UNIVERSIDAD NACIONAL DE LOJA

FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

“IMPLEMENTACIÓN DE INTERFAZ GRÁFICA PARA LA CLASIFICACIÓN Y RECONOCIMIENTO DE ROSTROS MEDIANTE EL PROCESAMIENTO DIGITAL DE SEÑALES.”

TESIS DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTOR:

BRYAN MANUEL NAGUA GONZÁLEZ.

DIRECTOR:

ING. MARCELO FERNANDO VALDIVIEZO CONDOLO, MG.SC.

LOJA – ECUADOR
2019

CERTIFICACIÓN

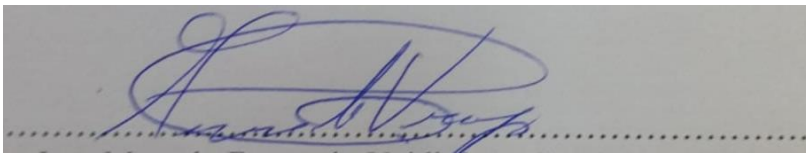
Ing. Marcelo Fernando Valdiviezo Condolo, Mg. Sc.

DIRECTOR DE TESIS

CERTIFICA:

Haber dirigido, asesorado, revisado y corregido el presente trabajo de tesis, en su proceso de investigación cuyo tema versa en "**IMPLEMENTACIÓN DE INTERFAZ GRÁFICA PARA LA CLASIFICACIÓN Y RECONOCIMIENTO DE ROSTROS MEDIANTE EL PROCESAMIENTO DIGITAL DE SEÑALES**", previa a la obtención del título de **Ingeniero en Electrónica y Telecomunicaciones**, realizado por el señor: **Bryan Manuel Nagua González** , la misma que cumple con la reglamentación y políticas de investigación, por lo que autorizo su presentación y posterior sustentación y defensa.

Loja, 22 de agosto de 2019

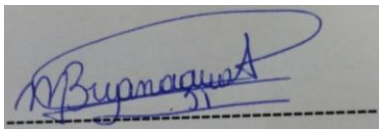


Ing. Marcelo Fernando Valdiviezo Condolo, Mg. Sc.
DIRECTOR DEL TRABAJO DE TESIS

AUTORÍA

BRYAN MANUEL NAGUA GONZÁLEZ, declaro ser el autor del presente trabajo de titulación y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales por el contenido de la misma.

Adicionalmente acepto y autorizo a la Universidad Nacional de Loja, la publicación de mi tesis en el Repositorio Institucional – Biblioteca Virtual.

Firma: 

Cédula: 1105771644

Fecha: 23 de octubre del 2019

CARTA DE AUTORIZACIÓN

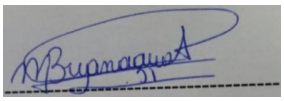
CARTA DE AUTORIZACIÓN DE TESIS POR PARTE DEL AUTOR, PARA LA CONSULTA, REPRODUCCIÓN PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO.

BRYAN MANUEL NAGUA GONZÁLEZ, declaro ser autor de la tesis titulada: "**IMPLEMENTACIÓN DE INTERFAZ GRÁFICA PARA LA CLASIFICACIÓN Y RECONOCIMIENTO DE ROSTROS MEDIANTE EL PROCESAMIENTO DIGITAL DE SEÑALES**" como requisito para optar al grado de: **INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES**; autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Digital Institucional:

Los usuarios pueden consultar el contenido de este trabajo en el RDI, en las redes de información del país y del exterior, con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia de la tesis que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los veinte y tres días del mes de octubre del dos mil diecinueve.

Firma: 

Autor: Bryan Manuel Nagua González

Cédula: 1105771644

Dirección: Loja, (Las Palmas) **Correo Electrónico:** bmnaguag@unl.edu.ec

Teléfono: 072722948 **Celular:** 0985763613

Director de Tesis: Ing. Marcelo Fernando Valdiviezo Condolo, Mg.Sc.

Tribunal de Grado: Ing. Manuel Augusto Pesantez González, Mg.Sc.

Ing. Marco Augusto Suing Ochoa, Mg. Sc.

Ing. Luis Eduardo Rodríguez Montoya, Mg. Sc.

DEDICATORIA

A Dios por bendecirme en cada etapa de mi vida personal y estudiantil. A mi madre, Chelita, quien es el motor de mi vida y quien supo apoyarme incondicionalmente en los momentos difíciles.

A mi padre, Manuel, por brindarme su tiempo, cariño, confianza y seguridad para alcanzar mis metas.

A mi hermana, Karen, y a mi sobrina, Darla, quienes con su afecto y comprensión me han motivado día a día.

A mi abuela, Liduvina, quien con su inmenso amor y cariño fue pilar fundamental en el transcurso de mi vida.

Bryan Manuel

AGRADECIMIENTO

En primer lugar, agradezco a Dios por la salud y la familia que me ha regalado, por su amor reflejado en cada bendición brindada y, por ser mi soporte, especialmente en las situaciones adversas.

A mis padres, Graciela y Manuel, quienes diariamente, con su trabajo y esfuerzo se han sacrificado para proporcionarme siempre lo mejor, por la formación moral y por el amor a Dios que me han inculcado desde etapas tempranas de mi vida.

A todos los docentes que formaron parte de mi vida universitaria, quienes supieron compartir con paciencia y dedicación toda su experiencia, y quienes me brindaron sabio consejo en los momentos adecuados.

De manera especial, quiero agradecer al Ing. Marcelo Valdiviezo, por todo el tiempo brindado, por compartir de manera desinteresada todo su conocimiento y por todo el apoyo académico brindado.

A Soraya, por su ayuda en el transcurso de toda la carrera, en el presente trabajo y en todos los momentos que lo necesitaba.

A mis amigos y compañeros de clase, quienes con su apoyo, alegría y ocurrencias me ayudaron a sobrellevar las situaciones complicadas del proceso de estudio.

Bryan Manuel

TABLA DE CONTENIDO

Pág.

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE TABLAS

Pág.

ACRÓNIMOS

CMU	Carnegie Mellon Univesity, Universidad de Carnegie Mellon
CNN	Convolucional Neural Network, Red Neuronal Convolucional
DCT	Discrete Cosine Transform, Transformada Discreta del Coseno
FERET	Face Recognition Technology, Tecnología de Reconocimiento Facial
GMM	Gausssian Mixture Models, Modelos de Mezclas Gaussianas
HOG	Histograms of Oriented Gradients, Histogramas de Gradientes Orientados
ICA	Independet Componenet Analysis, Análisis de Componentes Independientes
IEEE	Institute of Electrical and Electronics Engineers, Instituto de Ingenieros Eléctricos y Electrónicos.
JVM	Java Virtual Machine, Máquina Virtual Java
KNN	K-Nearest Neighbours, K-vecinos más cercanos
LDA	Linear Discriminant Analysis, Análisis Discriminante Lineal
MIT	Massachusetts Institute of Technology, Instituto de Tecnología de Massachusetts
OpenCV	Open Source Computer Vision Library, Visión por Computodora de Código Abierto
ORL	Olivetti Datasearch Laboratory, Laboratorio de Búsqueda de Datos Olivetti
PCA	Principal Component Analysis, Análisis de Componentes Principales
RAM	Random Access Memory, Memoria de Accesos Aleatorio
SK-LEARN	Sci-Kit Learn, Aprendizaje Sci-Kit.
SVM	Support Vector Machine, Máquina de Vectores de Soporte.

1. TÍTULO

“IMPLEMENTACIÓN DE INTERFAZ GRÁFICA PARA LA CLASIFICACIÓN Y RECONOCIMIENTO DE ROSTROS MEDIANTE EL PROCESAMIENTO DIGITAL DE SEÑALES.”

2. RESUMEN

En la presente investigación se desarrolló el análisis de dos técnicas fundamentales de reconocimiento facial, PCA (Principal Component Analysis) y LDA (Linear Discriminant Analysis), con la finalidad de elegir la técnica más eficiente para su posterior implementación en una interfaz gráfica desarrollada en el lenguaje de programación Python. El sistema de reconocimiento facial estará constituido por dos etapas principales, en la primera etapa el administrador podrá ingresar imágenes captadas desde la videocámara para almacenar el rostro de un usuario nuevo; en la segunda etapa el usuario podrá acceder al sistema capturando una fotografía de su rostro la cual será comparada con una base de datos previamente creada y se mostrará en pantalla la identificación y foto del usuario.

Para realizar un análisis comparativo entre las técnicas mencionadas anteriormente, se desarrolló una etapa de entrenamiento utilizando las bases de datos ORL (Olivetti Datasearch Laboratory) y MIT (Massachusetts Institute of Technology); observando el efecto que produce el incremento del conjunto de imágenes en los resultados finales.

Posteriormente, para comparar el rostro del usuario de entrada con los rostros presentes en la base de datos, se realizó una etapa de clasificación utilizando dos algoritmos clasificadores KNN, uno creado por el autor y otro desarrollado por la librería SK-Learn de Python; así mismo, se evaluaron los resultados obtenidos al aplicar cada uno de estos algoritmos.

Finalmente, la evaluación de los resultados se realizó utilizando la librería SK-Learn, esta herramienta permite obtener porcentajes de precisión y valores de error cuadrático medio de los modelos desarrollados, dichos parámetros son los que permitirán realizar una selección óptima del modelo a implementar.

Palabras clave: PCA, LDA, Reconocimiento facial, SK-Learn, ORL, MIT, Detección, Clasificador, Normalización.

ABSTRACT

In this work, the analysis of two mainly facial recognition techniques, PCA and LDA, was developed in order to choose the most efficient technique for its future implementation in a graphical interface developed in the Python programming language. The facial recognition system will consist of two main stages, in the first stage the administrator can enter images captured from the videocamera to store the face of a new user; in the second stage the user can access the system by capturing a photograph of his face which will be compared with the databases previously created and the identification and photo of the user will be displayed.

To perform a comparative analysis between the techniques mentioned above, a training stage was developed using ORL and MIT databases; observing the effect that produces the increase of the set of images in the final results.

Subsequently, to compare the face of the input user with the faces present in the database, a classification stage was carried out using two KNN classification algorithms, one created by the autor and another developed by the Python SK-Learn library.

Finally, the evaluation of the results was carried out using the SK-Learn library, this tool allows to obtain precisión percentages and mean Square error values of the developed models, these parameters are those that will allow an optimal selection of the model to be implemented.

Keywords: PCA, LDA, Face recognition, SK-Learn, ORL, MIT, Detection, Classifier, Normalization.

3. INTRODUCCIÓN

Se debe considerar que el reconocimiento de rostros es una acción que el ser humano realiza sin gran dificultad, pero esto no es así cuando se habla de sistemas computarizados; ya que el proceso de emular la facilidad del ser humano para clasificar y reconocer distintos objetos ha sido un tema de investigación profunda en los últimos años, y gracias al avance tecnológico se han implementado algoritmos cada vez más sencillos usando distintas técnicas para incrementar su precisión.

Así mismo, el reconocimiento de rostros generalmente se fundamenta en una extensa base de datos para realizar las comparaciones necesarias, ocasionando un elevado consumo de recursos del procesador; por esta razón se hace trascendental el uso de técnicas que permitan minimizar dicho consumo.

Los sistemas de reconocimiento facial tradicionales pueden evidenciar cuatro partes fundamentales, las cuales se pueden estructurar como se observa en la figura 1. Cabe recalcar que la etapa de normalización no es indispensable en el proceso de reconocimiento facial, pero es de gran relevancia para lograr una optimización de los resultados.(Cazorla Martínez, 2016)

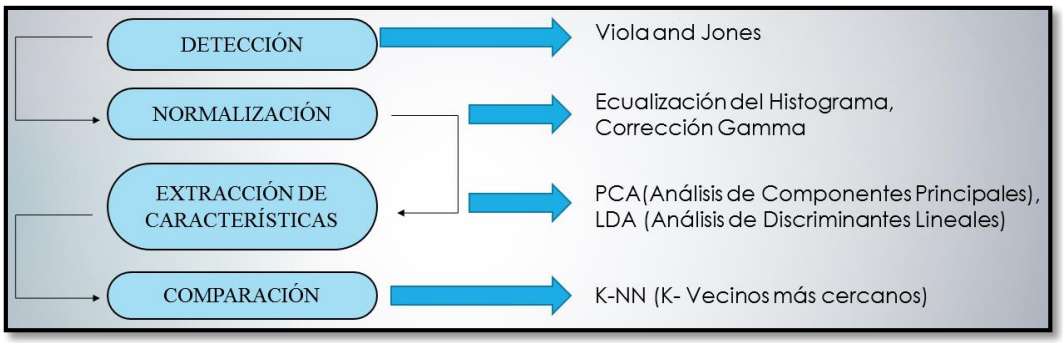


Figura 1. *Diagrama de las etapas de un sistema de reconocimiento de rostros.*
Fuente: *Autor*

Se considera como fase inicial la detección de áreas de la imagen que contengan un rostro (reconocimiento facial) o cierto patrón determinado (reconocimiento de objetos) con el propósito de extraer dicha área y analizar una menor cantidad de datos. La importancia de la etapa de detección radica en la implicación que ésta tiene en las fases siguientes, ya que el detectar erróneamente un rostro producirá un fallo en todo el proceso. (Cazorla Martínez, 2016)

La etapa de normalización se realiza con la finalidad de optimizar resultados ya que permite analizar imágenes cuyas características de luminosidad, contraste y dimensión sean distintas entre sí, para lo cual se localiza las componentes principales del rostro y mediante transformaciones geométricas se normaliza dichas componentes. (Cazorla Martínez, 2016)

Existen distintas técnicas para la extracción de características como pueden ser PCA (Principal Component Analysis), LDA (Linear Discriminant Analysis), DCT (Discrete Cosine Transform), pero todas pretenden extraer una matriz característica de cada rostro para ser comparada posteriormente. (Cazorla Martínez, 2016)

El objetivo de la fase final es identificar al usuario que está ejecutando el sistema, para lo cual se realiza las etapas de detección, normalización y extracción de características sobre la imagen de dicho usuario y se compara con los datos previamente almacenados en la base de datos; la identificación del usuario estará dada por el rostro que posea el mayor porcentaje de características similares.

Esta manera de identificar usuarios puede ser utilizada en distintas áreas como lectores biométricos inteligentes, smartphones o cámaras de seguridad, como ejemplo de este último se debe mencionar que la Alcaldía de la ciudad de Quito ha anunciado hace pocos días, la implementación de un sistema de videovigilancia con reconocimiento facial hasta finales del presente año, el cual permitirá capturar el rostro de una persona, almacenarlo en una base de datos y encontrar similitudes en dicha base; esto posibilitará conocer con exactitud la hora, fecha y lugar por el cual una persona transita.(Rodríguez, 2019). Así mismo, a nivel nacional existe un pequeño número de empresas del sector privado que ofrecen servicios de lectores biométricos basados en reconocimiento facial; por los ejemplos indicados se puede señalar que la implementación de este tipo de sistemas actualmente está dando sus primeros pasos en Ecuador.

Sin embargo, considerando que los sistemas de reconocimiento facial se realizan con bases de datos que contienen rostros de personas extranjeras, es necesario efectuar un ajuste de parámetros e implementar distintas técnicas que permitan optimizar dichos resultados en nuestro entorno, motivo por el cual en el presente trabajo se establecen los objetivos descritos a continuación.

Objetivos

Objetivo General

Desarrollar e implementar una interfaz gráfica que permita capturar imágenes en tiempo real y compararlas con una base de datos previamente creada aplicando técnicas de procesamiento digital de señales para la clasificación y reconocimiento de rostros.

Objetivos Específicos

- Seleccionar el modelo de reconocimiento de rostros más eficiente a través del estudio de diversos algoritmos para detectar características esenciales de cada rostro.
- Implementar el algoritmo seleccionado para reconocimiento de rostros a través del lenguaje de programación Phyton
- Desarrollar una etapa de entrenamiento y evaluación del modelo a través de una base de datos existente para verificar el correcto desempeño del algoritmo.
- Desarrollar una interfaz gráfica que permita realizar el reconocimiento facial usando una base de datos con imágenes capturadas en tiempo real.

4. REVISIÓN DE LITERATURA

El reconocimiento de rostros es una de las técnicas estudiadas en el área de la Inteligencia Artificial, que busca entrenar a una computadora para que sea capaz de realizar dicha actividad en la cual los seres humanos poseemos una eficiencia mayor; es decir, una computadora será capaz de detectar, extraer, almacenar y comparar las características físicas propias de cada individuo (datos biométricos[1]), y de esta manera crear una identidad digital para cada usuario. Existen diversos lenguajes de programación en los cuales se puede desarrollar y entrenar algoritmos de reconocimiento facial, a continuación se detallarán los más utilizados de acuerdo a los datos presentados por GitHub ("GitHub: The top 10 programming languages for machine learning," 2019), IEEE, ("The Top Programming Languages 2019 - IEEE Spectrum," 2019) y TIOBE("October Headline: Top 8 of the TIOBE index quite stable for the last 15 years," 2019); los cuales analizan y califican periódicamente el crecimiento de la popularidad de cada lenguaje; mostrando de esta manera los lenguajes de programación más utilizados en el área de Machine Learning.

4.1 Lenguajes de Programación

4.1.1 *Python*

Python es un lenguaje de programación de alto nivel, interpretado y multipropósito. Su popularidad ha ido creciendo constantemente en los últimos años y es considerado actualmente como uno de los lenguajes de programación más utilizados en el desarrollo de software. Python puede utilizarse en varios sistemas operativos como Windows, Mac OS y Linux.

Se debe recalcar que grandes empresas como Walt Disney, NASA, Google, Yahoo! y Nokia hacen gran uso de este lenguaje para desarrollar sus productos y servicios; demostrando que Python puede ser usado en distintos sectores. Este lenguaje es considerado potente, flexible y con una sintaxis clara y concisa.

[1] Dato biométrico o rasgo biométrico es un dato personal que posee características físicas, fisiológicas o conductuales y que aporta una identificación única a cada persona. (Serratosa, 2012)

Python es de código abierto, lo cual permite que cualquier persona pueda contribuir a su desarrollo y divulgación; así mismo, no es necesario pagar licencia para distribuir software desarrollado en este lenguaje. (Fernández Montoro, 2012).

Al ser un lenguaje de programación interpretado, ocupa menos tiempo a la hora de desarrollar el programa, ya que ahorra el proceso de compilar y enlazar. (Universidad de Granada, 2019).

Python es un lenguaje con tipado dinámico; es decir, el programador no debe declarar cada variable como un tipo específico (Entero, cadena, flotante), el tipo de cada variable se fija al momento de asignar su valor. (Fernández Montoro, 2012).

A continuación se detallan las librerías de Python más utilizadas en proyectos de Machine Learning: ("The State of the Octoverse: machine learning," 2019)

- Numpy, un paquete con soporte para operaciones matemáticas en datos multidimensionales, fue el paquete más importado, utilizado en casi tres cuartos de los proyectos de aprendizaje automático y ciencia de datos.
- Scipy, un paquete para computación científica, pandas, un paquete para administrar conjuntos de datos y matplotlib, una biblioteca de visualización, se utilizan en más del 40% de los proyectos de aprendizaje automático y ciencia de datos.
- Scikit-learn es un paquete popular de aprendizaje automático que contiene implementaciones de una gran cantidad de algoritmos de aprendizaje automático; casi el 40% de los proyectos lo utilizan.
- Tensorflow, un paquete para trabajar con redes neuronales, se usa en casi una cuarta parte de los paquetes.

4.1.2 Java

Java es un lenguaje de programación de propósito general orientado a objetos desarrollado por Sun Microsystems. Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. Una desventaja del lenguaje C++ es su falta de seguridad, pero C y C++ son lenguajes más difundidos; por ello, Java se diseñó para ser parecido a C++ y así brindar un rápido y fácil aprendizaje.

Java elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el garbage collector (reciclador de memoria dinámica). No es necesario preocuparse de liberar memoria, el reciclador se encarga de ello y, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria.

Java es parcialmente interpretado; es decir, aproximadamente el 20% del código Java es interpretado por la JVM (Java Virtual Machine), pero es un 20% muy importante. Tanto la seguridad de Java como su habilidad para ser ejecutado en múltiples plataformas se deben a que los pasos finales la compilación se maneja localmente. ("Principales Características de Java," 2019)

Además, Java es seguro; es decir, el código Java pasa varias pruebas antes de ejecutarse en una máquina. El código atraviesa un verificador de byte-codes que comprueba el formato de los fragmentos de código y aplica un verificador de teoremas para detectar fragmentos de código ilegal (código que falsea punteros, viola derechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto). ("Características de Java – Manual de Java," 2019)

En cuanto a Machine Learning, existen diferentes librerías para el entrenamiento y evaluación de este tipo de sistemas; a continuación se detalla algunas de ellas:

- Smile: un sistema rápido y completo para llevar a cabo aprendizaje automático, NLP (Natural Language Processing), álgebra lineal, gráficos, interpolación y sistema de visualización en Java y Scala.
- H20: una plataforma en memoria para aprendizaje automático distribuido y escalable que funciona en la infraestructura de big data existente, o en la parte superior de los clústeres Hadoop o Spark existentes.
- EasyML: un sistema de flujo de datos de uso general diseñado para facilitar la aplicación de algoritmos de aprendizaje automático a tareas del mundo real.

4.1.3 *Lenguaje C, C++, C#*

C es un lenguaje de nivel medio; se debe mencionar que los lenguajes de alto nivel se asemejan a la forma de razonar del ser humano, aislando al programador de los detalles técnicos (referentes a la máquina física), permitiendo que este tipo de lenguajes sean poco

eficientes; por el contrario, los lenguajes de bajo nivel controlan directamente la circuitería del ordenador, pudiendo obtenerse con ellos la eficiencia máxima (sin embargo, resultan incómodos y poco portables). Así las ventajas de los lenguajes de alto nivel respecto a los de bajo nivel son su sencillez, uniformidad y portabilidad.

Sin embargo, un programa escrito en un lenguaje de alto nivel debe ser traducido a un lenguaje que entienda la máquina antes de poder ser ejecutado. Esto se conoce como compilación (si traducen el programa completo a código máquina antes de ejecutar cualquiera de las instrucciones) o interpretación (si se recorre el programa tomando instrucciones una a una en pequeños grupos que se traducen y ejecutan).

Un compilador o intérprete, también es un programa que acepta como datos de entrada un programa en alto nivel (programa fuente) y genera como resultado un programa en lenguaje máquina (programa objeto).

En un nivel intermedio se sitúa el lenguaje C, permitiendo beneficiarse de las ventajas de ambos tipos de lenguajes, y reduciendo sus inconvenientes; por lo que se considera de propósito general. El lenguaje C ha sido utilizado para el desarrollo de diversas aplicaciones: sistemas operativos, hojas de cálculo, gestores de bases de datos, entre otras.

Es un lenguaje portable, es decir, es independiente del hardware, por lo que programas escritos en C son fácilmente trasportables a otros sistemas.

Sin embargo, este lenguaje carece de algunas ventajas que proporcionan otros lenguajes, como son: recolección de basura, encapsulamiento, soporte para programación orientada a objetos, funciones anidadas, polimorfismo y solo dispone de un soporte rudimentario para la programación genérica. Además, comparado con otros lenguajes, puede resultar un poco lento desarrollar en C y el mantenimiento puede ser tan costoso como inexperto sea el programador, ya que depende casi exclusivamente de su habilidad (Massó Gomez, 2012).

En el área de Machine Learning, la familia de lenguajes C ofrece las siguientes librerías:

- Tensorflow: el marco de aprendizaje automático ampliamente utilizado de Google con API para una amplia variedad de idiomas.
- Turi Create: una biblioteca que simplifica el desarrollo de modelos personalizados de aprendizaje automático para desarrolladores nuevos en el campo.

- LightGBM: diseñado por Microsoft para ayudar a aumentar la velocidad y la eficiencia del entrenamiento del modelo de aprendizaje automático.
- OpenCV: OpenCV es una biblioteca de visión por computadora de código abierto que tiene toneladas de módulos como detección de objetos, reconocimiento facial y realidad aumentada. ("GitHub: The top 10 programming languages for machine learning," 2019)

4.1.4 *JavaScript*

JavaScript es un lenguaje de programación de scripts (secuencia de comandos) orientado a objetos. Los scripts son en su mayoría interpretados; por tanto, es necesario contar con un intérprete para ejecutar código Javascript, el intérprete que se utiliza generalmente se incluye en el navegador de internet. Cada navegador tiene un intérprete Javascript, que varía en función del navegador. Si está utilizando Internet Explorer, el intérprete es llamado JScript (en versión 9 el intérprete es llamado Chakra), en Mozilla Firefox se llama SpiderMonkey y en Google Chrome es el motor V8. (Menéndez, n.d.)

Se debe recalcar que JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. JavaScript es un lenguaje que se incorpora dentro de la página Web, formando parte del código HTML.

A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java. Legalmente, JavaScript es una marca registrada de la empresa Sun Microsystems. (Eguíluz Pérez, n.d.) Además, Javascript no es orientado a objetos, esto quiere decir que no existe la necesidad de crear clases al momento de programar, tal como se realiza en los lenguajes de programación estructurada como C o Pascal. Java es mucho más complejo, aunque también más potente, robusto y seguro. Tiene más funcionalidades que Javascript y las diferencias que los separan son lo suficientemente importantes como para distinguirlos fácilmente. (EcuRed, 2010).

Así mismo, JavaScript posee librerías para procesos de Machine Learning, las cuales se muestran a continuación:

- TensorFlow.js: En 2019 se ha convertido en la base de todos los proyectos de Javascript de Machine Learning debido a su núcleo integral de álgebra lineal y sus capas de aprendizaje profundo.
- Brain.js: ciertamente se califica como la biblioteca de aprendizaje profundo más fácil en el ámbito de Machine Learning, incluidas las del ecosistema Python; además, es una de las bibliotecas de Javascript Deep Learning en evolución más activa.("Top Javascript Machine Learning libraries in 2019 - Towards Data Science," 2019).
- Bloques AI: un editor del tipo arrastrar y soltar que tiene como objetivo permitir que cualquiera pueda crear modelos de Machine Learning (también requiere la instalación de Python y tensorflow).
- ml5.js: tiene como objetivo hacer que el aprendizaje automático pueda ser utilizado por artistas y estudiantes sin conocimientos técnicos al ofrecer acceso a algoritmos y modelos de aprendizaje automático en el navegador. ("GitHub: The top 10 programming languages for machine learning," 2019)

4.2 Detección de Rostros

Es la técnica fundamental para desarrollar aplicaciones de reconocimiento y seguimiento de rostro o reconocimiento de expresiones faciales y de gestos, ya que para un funcionamiento correcto de las mismas se requiere que inicialmente el rostro sea detectado en la imagen. (Yang, Luo, Loy, & Tang, 2016)

Con la finalidad de comprender de mejor manera las técnicas de detección de rostros, estas se pueden clasificar en dos grandes grupos: Las basadas en Análisis del rostro completo y las basadas en Análisis de Rasgos Característicos del rostro. Las primeras se basan en clasificadores, los cuales analizarán una base de datos representativa para aprender cual figura corresponde a un rostro y cual no. Los métodos más conocidos de este tipo son: Redes Neuronales Convolucionales (Convolutional Neural Networks, CNN), Máquina de Vectores de Soporte (Support Vector Machine, SVM) y Análisis de Componentes Principales (Principal Component Analysis, PCA). La desventaja de estos

sistemas es su complejidad elevada pero se compensa con la precisión obtenida en etapas finales.

Paralelamente, las técnicas basadas en el análisis de rasgos característicos del rostro se sub-dividen en tres niveles. En el nivel inferior, se trata de detectar los rostros asemejándolos a manchas que tenga forma de una cara frontal, a través de valores de pixeles grises, movimiento o color. En el nivel medio, se buscan características que se mantengan fijas aunque existan cambios en las condiciones de luz u orientación del rostro. En el nivel superior, se pretende encontrar características de un rostro como ojos, nariz, boca y contornos de la cara para posteriormente usar modelos deformables o modelos de distribución de puntos. (Hatem, Beiji, & Majeed, 2015).

En la figura 2 se presenta una distribución detallada de las técnicas de detección de rostros:

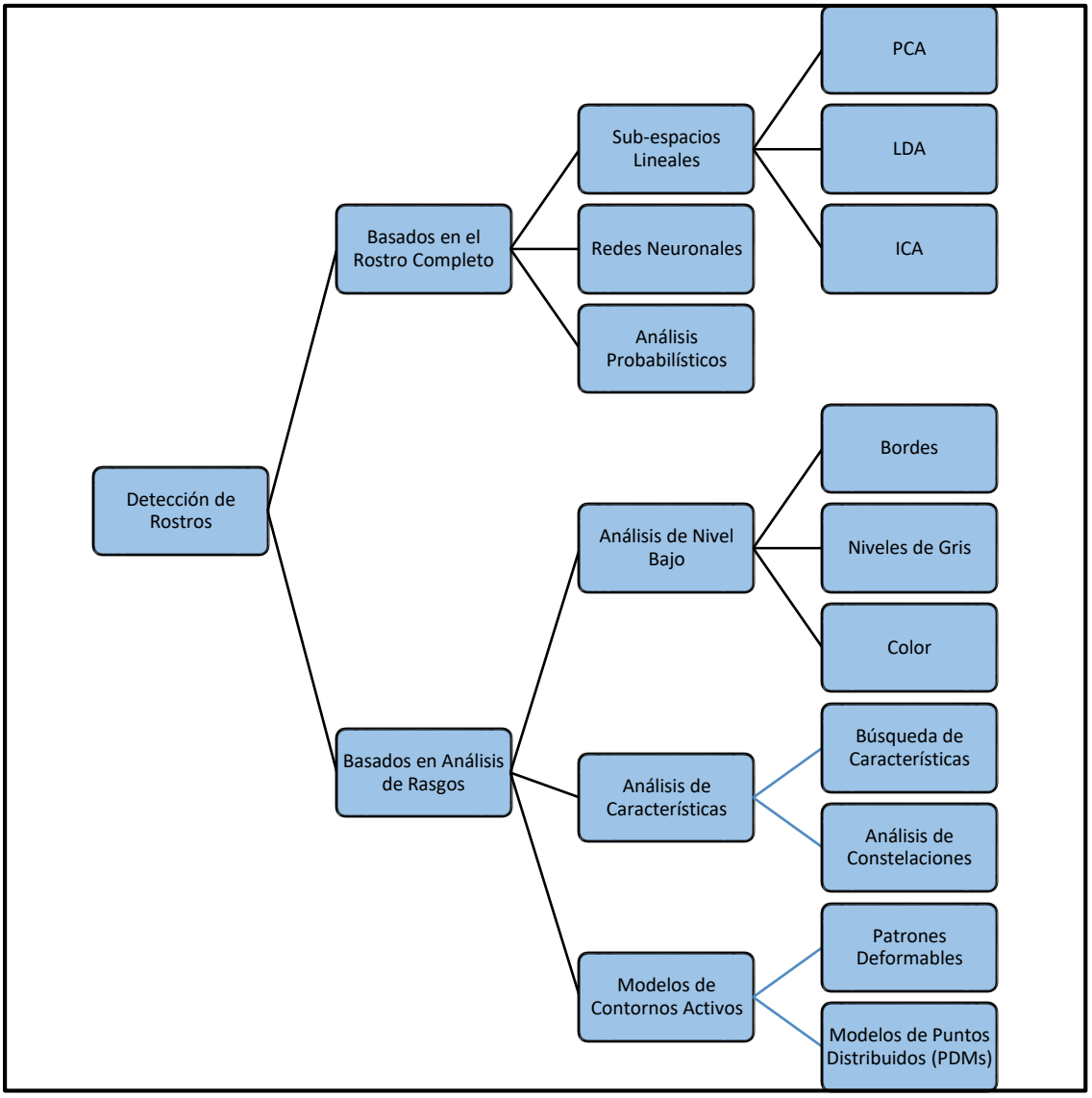


Figura 2. *Detección facial de acuerdo a distintos enfoques*
Fuente: *Autor*

4.2.1 *Detector Facial Viola and Jones*

Este algoritmo de detección fue uno de los métodos que tuvo el mayor impacto en la década de 2000 (Viola & Jones, 2002). Este detector puede ejecutarse también en tiempo real gracias a sus tres ideas principales: Imagen integral, Aprendizaje de clasificadores AdaBoost y Estructura de filtros en cascada. (Zhang & Zhang, 2010).

La imagen integral es un algoritmo que trabaja dividiendo una cuadrícula (imagen) en un sub-conjunto o sub-rectángulo en el que calcula de manera rápida la suma de sus valores. Viola y Jones aplicaron este concepto para calcular las características *Haar-like* de forma

rápida. (Zhang & Zhang, 2010). *Haar-like* es una característica simple rectangular que se calcula sobre un área especial de la imagen, en donde la suma de los pixeles debajo de las áreas blancas se restan de la suma de los pixeles debajo de las áreas negras.(Hirvin Gonzalez, 2019)

La imagen integral se construye de la siguiente manera:

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

donde $ii(x,y)$ es la imagen integral en la ubicación de píxeles (x,y) , y los valores correspondientes a (x',y') son los valores de la imagen original.

A continuación, para extraer las características de la imagen se aplica filtros *Haar-like*[2], los cuales usan una ventana básica de 24 x24 pixeles que se recorre a través de toda la imagen en búsqueda de características *Haar-like*. (Hirvin Gonzalez, 2019). Los filtros *Haar-like* generan características de borde, puntos y líneas a través de captura de contraste entre regiones con la finalidad de formar una codificación de diferencia de intensidades en la imagen. La detección de rostros usando filtros Haar-Like se presenta en la figura 3. (Caballero Barriga, 2017)

Figura 3. *Convolución del filtro Haar-like con una imagen integral*
Fuente: (Caballero Barriga, 2017)

Finalmente, para ejecutar la etapa de clasificación se utiliza un clasificador en cascada, este clasificador pretende elaborar una serie de clasificadores fuertes más pequeños que

[2] Los filtros *Haar-like* reciben este nombre ya que están basados en características *Haar-like.*

puedan filtrar la mayoría de sub-ventanas negativas tratando de almacenar la mayor cantidad de sub-ventanas positivas. Este enfoque permite aumentar la eficiencia de la detección ya que una gran cantidad de sub-ventanas se descartarán en etapas iniciales del detector. (Zhang & Zhang, 2010). En la figura 4 se muestra el funcionamiento de los clasificadores en cascada *Haar*.

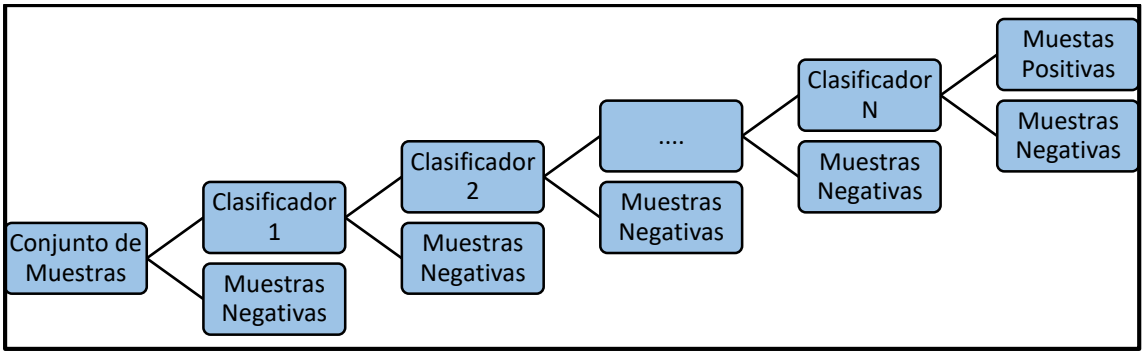


Figura 4. *Estructura de los clasificadores en Cascada*
Fuente: (Van Dung, Vavilin, Jo, Hoang, & Vavilin Kang-Hyun Jo Ulsan, 2012)

Como se puede observar en la figura 4, las muestras consideradas finalmente como positivas deben atravesar un gran número de clasificadores durante la detección. Cada clasificador deberá decidir de forma binaria si la muestra analizada se mantiene para el siguiente nodo o si se descarta inmediatamente. Generalmente, los nodos finales poseen un mayor número de clasificadores débiles ya que en las etapas finales las muestras positivas que se analizan aumentan y se debe rechazar una cantidad mínima de muestras negativas. (Zhang & Zhang, 2010)

4.2.2 HOG (Histogram of Oriented Gradient)

El método se basa en la evaluación de histogramas locales bien normalizados de orientaciones de gradiente de imagen en una grilla densa. La idea básica es que la apariencia y la forma del objeto local a menudo se pueden caracterizar astante bien por la distribución de gradientes de intensidad locales o direcciones de borde, incluso sin un conocimiento preciso de las posiciones correspondientes de gradiente o borde. En la práctica, esto se implementa dividiendo la ventana de imagen en pequeñas regiones espaciales (celdas), para cada celda, acumulando un histograma 1-D local de direcciones de gradiente u orientaciones de borde sobre los píxeles de la celda. Las entradas combinadas del histograma forman la representación. Para una mejor invariancia a la iluminación, sombreado, etc., también es útil contrastar-normalizar las respuestas locales

antes de usarlas. Los bloques de descriptores normalizados son conocidos como histograma de descriptores de gradiente orientado (HOG). El mosaico de la ventana de detección con una cuadrícula densa (de hecho, superpuesta) de descriptores HOG y el uso del vector de características combinadas en un clasificador de ventana basado en SVM convencional proporciona el sistema de detección de rostros humanos. (Dalal, Histograms, & Triggs, 2009).

4.2.3 Máquinas de Soporte Vectorial (SVM, Support Vector Machine)

Una Máquina de Soporte Vectorial es un sistema de aprendizaje automático utilizado para resolver problemas de clasificación y regresión de manera eficiente. Las SVM se basan en la Teoría de Aprendizaje Estadístico y son capaces de clasificar muestras en dos posibles conjuntos: "positivos" y "negativos", que en el caso de detecciones de rostros corresponden a "rostros" y "no rostros" respectivamente. Para ello, se necesita un entrenamiento previo de la máquina.

Se toman dos conjuntos de datos diferentes para el entrenamiento del SVM. El proceso de aprendizaje consiste en que una vez obtenido el primer modelo de detección, se compara con el conjunto de imágenes de entrenamiento. Si en este punto una imagen entrenada como positiva no es detectada significa que se tiene un falso negativo y si, por el contrario, una imagen que no presenta un rostro es detectada, se tiene un falso positivo. Lo que se hace a continuación es reentrenar el modelo añadiendo los falsos positivos o los falsos negativos detectados al conjunto correspondiente. Este proceso de reentrenamiento se repite tantas veces como sea necesario hasta obtener una precisión razonable del detector. (Yilber Sisco, 2017)

4.2.4*Patrones Binarios Locales (LBP, Local Binary Pattern)*

El operador LBP es uno de los descriptores de textura con mejor rendimiento y se ha utilizado ampliamente en varias aplicaciones. Ha demostrado ser altamente discriminativo y sus ventajas clave son su invariabilidad a los cambios de nivel de gris monótonos y la eficiencia computacional, que lo hacen adecuado para exigentes tareas de análisis de imágenes.

La idea de utilizar LBP para la descripción de la cara está motivada por el hecho de que las caras pueden verse como una composición de micropatrones que están bien descritos

por dicho operador. (Ahonen, Hadid, & Pietikä, 2006). Para cada píxel en una imagen, se produce un código binario al limitar su valor con el valor del píxel central. Cada píxel vecino se compara con el píxel central, y aquellos cuyas intensidades exceden los píxeles centrales se marcan como "1", de lo contrario, como "0". De esta manera obtenemos características simples de puntos circulares que consisten solo en bits binarios. Se crea un histograma para recoger las ocurrencias de diferentes patrones binarios. La versión básica del operador LBP considera solo los ocho vecinos de un píxel, pero la definición se ha ampliado para incluir todos los vecindarios circulares con cualquier número de píxeles con la finalidad de manejar las texturas a diferentes escalas.(Mu, Yan, Liu, Huang, & Zhou, n.d.)

4.2.5 *Redes Neuronales Artificiales*

Los algoritmos que se basan en redes neuronales contienen varias propiedades entre las que se encuentran las habilidades de aprender, adaptarse, agrupar y generalizar. Sus operaciones se realizan con un procesamiento en paralelo, constituido por muchas unidades de procesamiento sencillas llamadas neuronas, las capas ocultas de neuronas y las conexiones interneuronales. (Jiménez Ochoa, 2016).

El conocimiento de la red se adquiere con un proceso denominado entrenamiento; este proceso, durante la ejecución de las capas utilizadas, almacena las funciones que mejor describen las características faciales en una variable comúnmente conocida como peso. En el transcurso del entrenamiento estos pesos se irán descartando para evitar el sobreajuste de la red. Implícitamente, las neuronas de la red se activan siempre que se supera un umbral, el cual viene dado por una función de activación como: función escalón, tangente hiperbólica o sigmoide. La neurona puede ser lineal o no lineal dependiendo de la función utilizada.(Peña & Orellana, 2018).

4.3 Normalización

En la fase de normalización se realiza procesamiento sobre la información obtenida en la fase anterior con el objetivo de preparar la imagen para ejecutar la Extracción de Características de manera óptima (Eslava Ríos, 2013). Las técnicas más comunes de normalización en imágenes se describen a continuación.

4.3.1 *Ecualización del Histograma*[3]

En ciertos casos, imágenes que aparentemente son similares entre sí, no lo son si se analiza el valor de intensidad de sus pixeles, debido a la variación de luminosidad y contraste. Para evitar este tipo de inconvenientes se aplica técnicas de ecualización del histograma que permiten extender estos valores a lo largo de todo el rango del histograma, en lugar de trabajar con imágenes en las que los valores de intensidad se concentren en una sola región. Una vez ecualizado el histograma las imágenes resultantes presentan mayor contraste y menor variabilidad lumínica entre ellas (Eslava Ríos, 2013).

La ecualización del histograma se inicia calculando el histograma acumulado de la imagen, el cual está dado por:

$$H(i) = \sum_{k=0}^{i} h(k) \qquad (2)$$

donde $h(k)$ es el histograma de la imagen e i es el nivel de intensidad (0 a 255).

Para que el histograma sea totalmente plano, el histograma de cada nivel de gris debe ser:

$$G(i') = (i' + 1)\frac{NM}{256} \qquad (3)$$

Donde $N\ y\ M$ son las dimensiones de la imagen, y 256 es el número de niveles con los que se representa la imagen, en este caso $2^{8\ bits} = 256\ niveles$

Ya que se pretende que $G(i') = H(i)$, se tiene que:

$$(i' + 1)\frac{NM}{256} = H(i) \quad \therefore \quad i' = \frac{256}{NM}H(i) - 1 \qquad (4)$$

Recordando que los niveles de gris deben ser únicamente valores enteros (Esqueda Elizondo, 2005), se tiene que:

$$i_{nuevo} = Parte\ entera\left[\frac{256}{NM}H(i_{anterior}) - 1\right] \qquad (5)$$

[3] Histograma de una imagen es una representación entre el número de pixeles en cierto nivel de gris versus los niveles de gris (255 en la mayoría de los casos). (Atienza, 2011)

Este procedimiento tiene como finalidad lograr que el número de pixeles para cada nivel de gris (0 a 255) sea lo más parejo posible, es decir, que exista el mismo número de pixeles en cada nivel de gris; como se muestra en la figura 5.

Debido a lo mencionado anteriormente, este proceso permite mejorar la precisión de los sistemas de reconocimiento facial ya que logra atenuar los cambios de luminosidad que se pueden presentar al extraer la información correspondiente al rostro. La imagen debe convertirse a escala de grises previo a la normalización del histograma.(Cazorla Martínez, 2016).

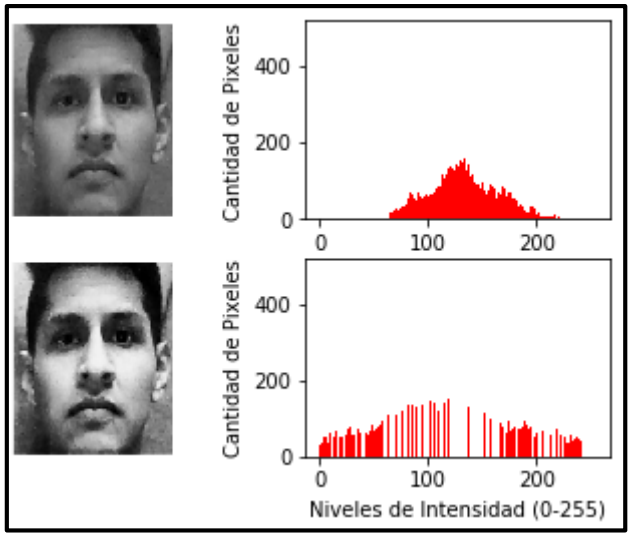


Figura 5. *En la parte superior imagen original oscura, en la parte inferior imagen ecualizada*
Fuente: *Autor*

4.3.2 *Corrección Gamma*

La corrección gamma o función potencia permite modificar el contraste de una imagen, logrando un incremento del mismo en niveles de gris oscuros (próximos a 0) o en niveles claros (próximos a 255) dependiendo del factor de corrección gamma seleccionado. (Bhandari, Kumar, & Singh, 2015).

Una función de potencia general tiene la forma:

$$s = cr^{\gamma} \qquad (6)$$

Donde s es el nuevo valor del pixel, c es una constante positiva de escalamiento que permite obtener valores de salida dentro de un rango deseado, 0 a 255 en este caso, r es el valor actual del pixel y γ es una constantes positiva que permitirá modificar el valor

del pixel. Como se puede observar en la figura 6, con un gamma inferior a 1, las entradas que poseen niveles de intensidad oscuros se convierten en salidas notablemente más claras, pero ocurre lo contrario para un factor gamma superior a 1, donde las entradas que tiene niveles de intensidad claros adoptan un nivel de intensidad oscuro en la salida. Se puede apreciar que cuando gamma es igual a 1, no existe ningún cambio en los niveles de intensidad a la salida. (Gonzalez & Woods, 2017)

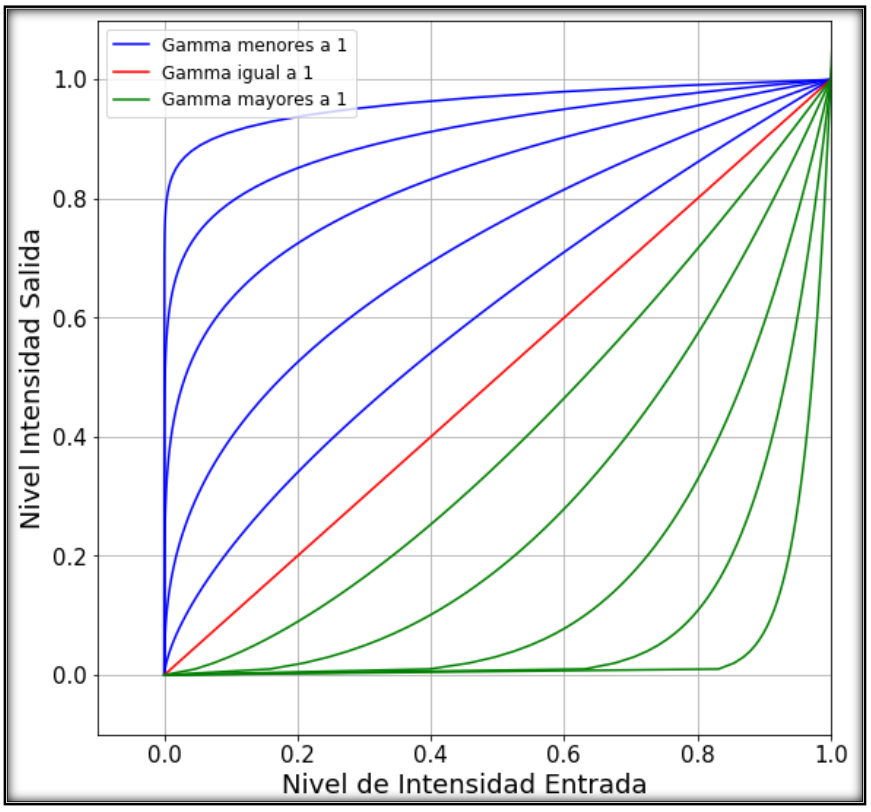


Figura 6. *Gráfica de la variación de intensidad de salida respecto a la entrada usando distintos alores de gamma*
Fuente: (Gonzalez & Woods, 2017)

Para aplicar la corrección gamma en una imagen, se utiliza la siguiente fórmula(Aranguren Zapata & Vela Asin, 2012):

$$Imagen_{nueva} = 255\left(\frac{Imagen_{entrada}}{255}\right)^{\gamma} \qquad (7)$$

En donde, Imagen nueva hace referencia al valor nuevo de intensidad de cada pixel, Imagen de entrada hace referencia al valor de intensidad actual de cada pixel y γ es el factor de corrección.

En la figura 7 se muestra los resultados al aplicar un factor de corrección gamma = 0.25, en la parte superior se observa la fotografía antes de normalizar con su respectivo histograma y en la parte inferior el resultado de la normalización. Se observa en los histogramas que, al normalizar la imagen, la intensidad de gris se concentra en los niveles más bajos (tonos oscuros).

En la figura 8, se realiza el mismo procedimiento, pero aplicando un factor de corrección gamma = 2. Se aprecia de acuerdo a los histogramas que la intensidad de gris se concentra en los niveles más altos (tonos claros).

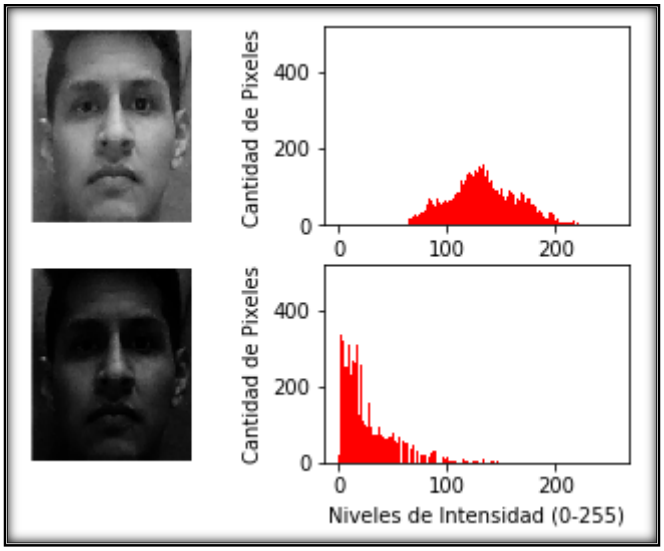


Figura 7. *Resultado de normalización al aplicar gamma = 0.25*
Fuente: *Autor*

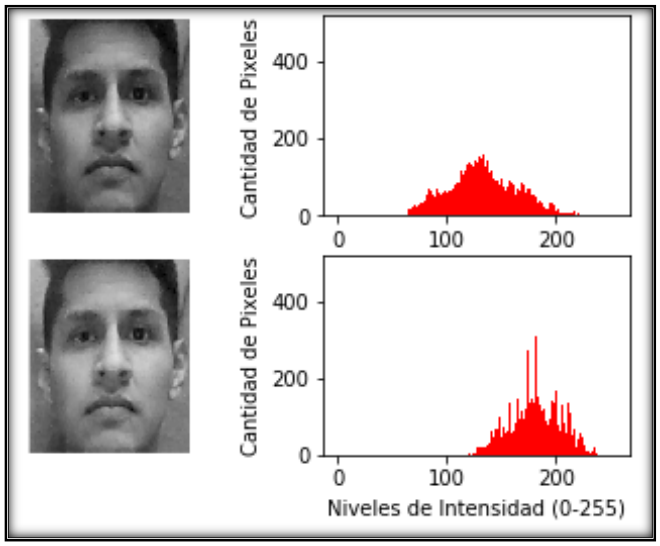


Figura 8. *Resultado de normalización al implementar gamma = 2*
Fuente: *Autor*

4.3.3 *Laplacciano*

Para definir el laplaciano digital el requisito principal es que los valores que determinan los píxeles exteriores sean negativos mientras que el valor que determina al píxel central sea positivo, dado que el laplaciano es una derivada de segundo orden, la suma de todos sus valores debe ser cero, por lo que resulta demasiado sensible al ruido, invalidándola para su uso en la práctica de detección de bordes, ya que además de esto, produce bordes dobles y es incapaz de detectar direcciones de borde.

El laplaciano puede utilizarse de forma más general en la búsqueda de ubicación de bordes utilizando las propiedades de paso por cero y en segundo plano podría establecer si un píxel se encuentra en una zona clara u oscura, es decir la adición del laplaciano en los bordes de la imagen original restaura la variación general de la escala de grises. (Huamán Layme & Del Carpio Salinas, 2007).

También mejora la imagen aumentando localmente el contraste en las discontinuidades, puede describirse como un filtro paso alto, el cual solo permite el paso de altas frecuencias al tiempo que elimina las bajas frecuencias correspondientes a la variación general gradual del brillo. (Russ & Brent Neal, 2017).

4.3.4 *Operaciones de Rango*

Las operaciones de rango determinan los píxeles que tienen los tonos más claros y más obscuros a través de operadores de máximo y mínimo, colocando el valor de ese tono en el píxel central. Estos pueden ser utilizados como una alternativa para encontrar y eliminar ruido claro u oscuro, produciendo de esta manera un filtro de mejora de bordes en la imagen. Para el caso de búsqueda de puntos oscuros, el algoritmo realiza una comparación entre el valor de brillo mínimo en una región pequeña central y el mismo valor dentro de un anillo circundante, si el resultado de esta comparación supera algún umbral establecido previamente, el valor de dicho píxel central se retiene, caso contrario se eliminaría. Para los puntos claros se utiliza el procedimiento invertido.

El filtro adaptivo será más eficiente entre más información disponible se encuentre de los objetivos ya que de esta manera se pueden determinar las características relevantes del mismo, separándolas del ruido de fondo. (Russ & Brent Neal, 2017).

4.4 Extracción de Características

En esta etapa se extraen distintas características propias del rostro humano para ser clasificadas, analizadas y comparadas en pasos posteriores. En la etapa de extracción de características se procura obtener los valores que aportan información relevante sobre la cara, descartando valores que, en la fase de comparación, pueden dar lugar a confusión o que simplemente no aportan información. A continuación se describirán los algoritmos Eigenfaces y Fisherfaces que para realizar la extracción de características trabajan con Análisis de Componentes Principales (Principal Component Analysis, PCA) y Análisis de Discriminantes Lineales (Linear Discriminant Analysis, LDA) respectivamente (Cazorla Martínez, 2016).

Estos algoritmos buscan clasificar los rostros en un nuevo sub-espacio vectorial. En la figura 9 se puede observar a la izquierda una imagen de entrenamiento que se proyecta en un nuevo subespacio vectorial. (Cazorla Martínez, 2016).

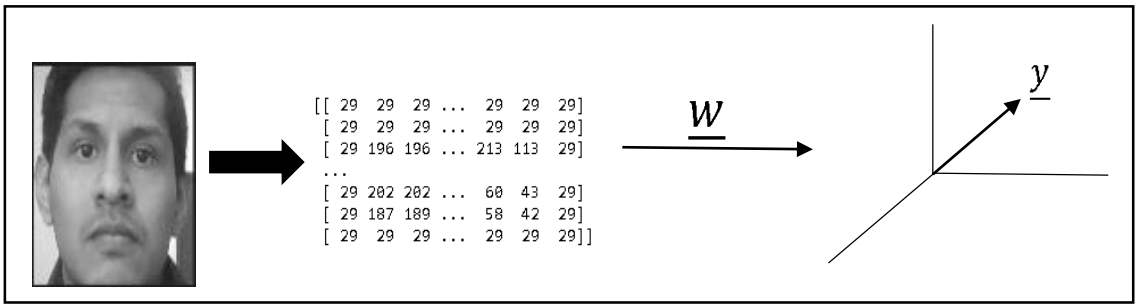


Figura 9. *Esquema que muestra la proyección vectorial de una imagen al nuevo subespacio.*
Fuente: (*Cazorla Martínez, 2016)*

4.4.1 *Eigenfaces*

Lawrence Sirovich y Robert M. Kirby aplicaron por primera vez la técnica PCA en 1987 para representar imágenes de rostros en una dimensión. Su idea se basaba en calcular el mejor sistema de coordenadas para comprimir una imagen de un rostro, donde cada coordenada es una imagen llamada *eigenpictures* (imágenes propias). Este concepto fue mejorado y reformulado en años posteriores por Matthew Turk y Alex Pentland, los cuales desarrollaron el método de reconocimiento denominado *Eigenfaces* (Caras Propias). Se debe aclarar que PCA y *Eigenfaces* son términos distintos, ya que el método PCA es un método general para analizar datos mientras que *Eigenfaces* es un método de reconocimiento facial que utiliza PCA con ciertas modificaciones (Slavković & Jevtić, 2012). En la figura 10 se muestra las *Eigenfaces* correspondientes a la base de datos ORL.

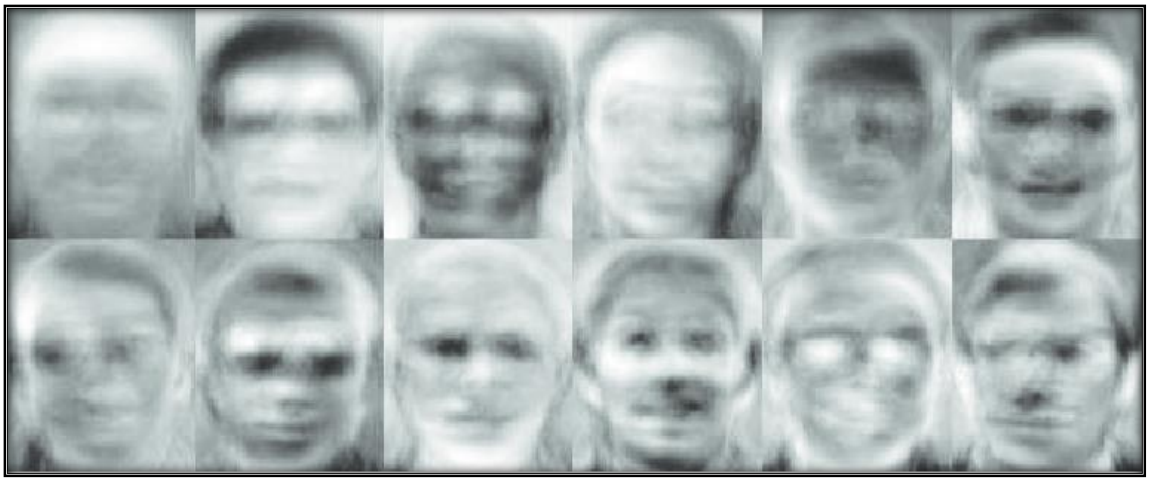


Figura 10.*Eigenvectores de los 12 principales eigenvalores de la base de datos ORL*
Fuente: (Chihaoui, Elkefi, Bellil, & Ben Amar, 2016)**.**

4.4.1.1 *Cálculo de Eigenfaces*

Una imagen de una cara representada mediante una matriz cuadrada $N\ x\ N$ puede considerarse como un vector N^2, por lo cual una imagen típica de 256 por 256 pixeles será un vector de dimensión 65536, o visto de otra forma, una imagen de este tamaño será un punto en un espacio de 65536 dimensiones. Un grupo de imágenes formará una nube de puntos en este gran espacio. PCA tiene como objetivo encontrar la base que defina de mejor manera la distribución de imágenes de rostros dentro de todo el espacio, denominado espacio de imágenes[4] por Kirby y Sirovich. Cada vector con una dimensión $d = N^2$ representa una imagen de $N\ x\ N$ que es una combinación lineal de vectores base del subespacio. Ya que dichos vectores son los pertenecientes a la matriz de covarianza[5] correspondiente al espacio original de imágenes, y también debido a que se asemejan a una cara, se les llama *Eigenfaces* (Narayana, 2015)

"Sea el conjunto de imágenes de caras $x_1, x_2, x_3 \dots x_m$ (considerando vectores columna de dimensión d se puede construir la matriz X de dimensiones $d\ x\ m$). La media del conjunto (o la cara media) se define $\mu = \frac{1}{m}\sum_{i=1}^{m} x_i$". Cada rostro varía de la media en función del vector $x_i - \mu$. PCA buscará un conjunto de m vectores ortonormales u_k,

[4] Espacio de imágenes hace referencia al nuevo sub-espacio vectorial en el cual las imágenes de los rostros han sido proyectadas.
[5] La covarianza es un valor que refleja en que cantidad varían dos variables respecto a su media. (Sánchez, 2006)

dentro del conjunto de vectores descritos anteriormente para representar la distribución de los datos. (Slavković & Jevtić, 2012)

Los vectores u_k son los vectores de la matriz de covarianza y los escalares λ_k son los valores propios de dicha matriz:

$$S = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)\,(x_i - \mu)^T = AA^T \qquad (8)$$

"Donde A sería la matriz *X* normalizada (la media restada de cada columna x_i). La matriz S tiene una dimensión de $d\ x\ d = (N^2\ x\ N^2)$", debido a la extensión de la matriz S, la tarea de extraer sus vectores y valores propios es computacionalmente imposible para imágenes de tamaño normal. Si en el espacio de imágenes el número de puntos es menor que la dimensión del mismo ($m < N^2$) , existirán como máximo $m - 1$ vectores propios significativos. Este problema se puede resolver realizando convenientemente las combinaciones lineales de las imágenes. Es decir, en lugar de calcular AA^T se debe considerar v_i los vectores propios de A^TA (y α_i sus valores propios) resultando:

$$A^TAv_i = \alpha_i v_i \qquad (9)$$

Multiplicando por A los dos extremos de la ecuación se tiene: variar

$$AA^TAv_i = \alpha_i Av_i \qquad (10)$$

Donde Av_i son vectores propios de $S = AA^T$

Con el análisis previo se obtiene la matriz A^TA de dimensión $m\ x\ m$ y se calculan los m vectores propios. (Slavković & Jevtić, 2012)

4.4.1.2 *Procedimiento de Eigenfaces*

Un algoritmo convencional de reconocimiento facial usando el método de *Eigenfaces* ejecuta el siguiente proceso general:

a. Adquisición de un conjunto inicial de imágenes o conjunto de entrenamiento, se debe añadir varias imágenes para cada persona, con distintas expresiones y cambios en la iluminación.

b. Calcular la matriz $m\,x\,m$ normalizada y encontrar sus vectores y valores propios, además se debe elegir los m' vectores propios con sus respectivos valores propios asociados más altos.

c. Calcular los m' *Eigenfaces* proyectando el conjunto de entrenamiento con los autovectores.

d. Para cada clase (individuo) se debe elegir como mínimo una (o promediando más de una) de sus imágenes y calcular el vector de clase Ω_k (proyectado en el espacio de imágenes).

e. Calcular el vector $\Omega = \mu_k(y - u)$ para cada nueva imagen y, y calcular la distancia existente a cada vector de clase.

4.4.2 *Fisherfaces*

Este método se basa en el hecho de que distintas imágenes de la misma persona, con la misma expresión facial pueden apreciarse drásticamente diferentes al ser iluminadas desde distintas direcciones. En sus inicios, *Fisherfaces* se fundamentaba en dos argumentos:

a. Si se captura distintas imágenes de una superficie de Lambert desde un punto de vista fijo, pero insertando cambios en la iluminación, éstas se proyectan en un sub-espacio lineal 3D perteneciente al espacio de la imagen.
b. En la práctica, existen zonas del rostro que pueden presentar variaciones de imagen a imagen, lo cual deriva en gran medida del sub-espacio lineal, y por lo tanto, reduce la fidelidad del reconocimiento. Estas variaciones se presentan debido a la presencia de sombra, reflejos y expresiones faciales (Armengot, 2006).

Cabe recalcar que el método de *Eigenfaces* también busca proyectar linealmente el espacio de las imágenes en un espacio de características con una menor dimensión y que dicho espacio sea intolerante a cambios en la expresión facial y variaciones de luz. Pero este método presenta un problema importante ya que al utilizar PCA, que pretende maximizar la dispersión, las variaciones no deseadas en los cambios de la expresión facial y de luz se mantienen. Además, los cambios ocasionados por la dirección del punto de

vista y la luz entre imágenes del mismo rostro son generalmente mayores a los cambios en la identidad del rostro. (Panda & Naik, 2015)

4.4.2.1 *Formulación*

Se intenta desarrollar un método de mayor fiabilidad que permita reducir la dimensionalidad del espacio de características usando la información obtenida del etiquetado del conjunto de aprendizaje. Para conseguir esta reducción en la dimensionalidad, las técnicas LDA utilizadas en *Fisherfaces* pretenden optimizar la dispersión obtenida de la proyección de datos. Existe una diferencia fundamental entre PCA y LDA, ya que el primero intenta encontrar los vectores que describen de mejor manera los datos, y el segundo busca los vectores que ofrecen una mayor discriminación entre clases luego de realizar la proyección (Armengot, 2006)

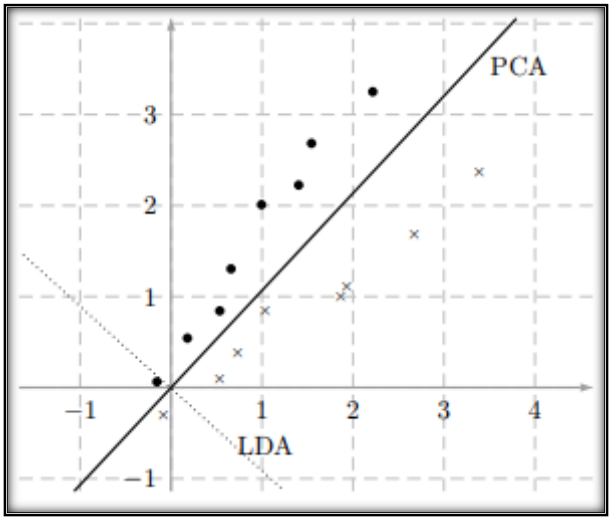


Figura 11. *Ejemplo de Proyección LDA en dos dimensiones.*
Fuente: *(Armengot, 2006)*

En la figura 11 se ilustra las ventajas de la técnica LDA aplicada en un espacio usual de dos dimensiones. En la técnica PCA, sería imposible discriminar los datos ya que, como se puede observar, los puntos se proyectan de forma mezclada alrededor de la recta que simboliza la dirección principal. LDA describe de manera más eficiente la dispersión entre clases ya que analiza la variación de los datos entre todas las imágenes y, además, analiza la variación de los datos dentro de las imágenes de cada usuario. (Panda & Naik, 2015). Generalmente, en el reconocimiento de rostros la matriz de dispersión intra-clase no tiene inversa (es singular) lo que representa un problema que se origina debido a la cantidad menor de imágenes en el conjunto de aprendizaje con respecto al número de

pixeles de cada imagen. *Fisherfaces* modifica el LDA convencional para solucionar este problema, proyectando el conjunto de imágenes en un espacio de dimensión menor donde la resultante S_w no es singular. En resumen, *Fisherfaces* hace uso de las dos técnicas descritas anteriormente, en primer lugar aplica la técnica PCA para disminuir la dimensión del espacio de imágenes y luego aplica el LDA convencional. (Smiatacz, 2013)

Considerando una matriz w de proyección, la cual está conformada por los autovectores ordenados ascendentemente, el objetivo es encontrar una w que maximice:

$$J\,(w) = \frac{w^T S_B w}{w^T S_w w} \tag{11}$$

Donde $J\,(w)$ es una matriz de proyección análoga a la matriz formada por los autovalores y autovectores en PCA, S_B se conoce como matriz de dispersión interclase, S_W se conoce como matriz de dispersión intraclase y w son los autovectores de S_B y S_w. La matriz de dispersión interclase describe la dispersión que tienen los diferentes usuarios entre sí, mientras que la matriz de dispersión intraclase describe la dispersión interna de la información de cada usuario. Suponiendo c clases, en donde cada clase tiene n muestras, S_B y S_W se definen como:

$$S_B = \sum_C n(\mu_C - \overline{x})(\mu_C - \overline{x})^T \tag{12}$$

$$S_W = \sum_C \sum_{i \in C} (x_i - \mu_C)(x_i - \mu_C)^T \tag{13}$$

Donde μ_C es la media de la clase (usuario) c y $\overline{x}$ es la media total, con lo cual se cumple que:

$$\mu_C = \frac{1}{n} \sum_{i \in C} x_i \tag{14}$$

$$\overline{x} = \frac{1}{n} \sum_i x_i = \frac{1}{n} \sum_c n\, \mu_c \tag{15}$$

Considerando S_T como la matriz de dispersión total,

$$S_T = \sum_i (x_i - \overline{x})(x_i - \overline{x})^T \tag{16}$$

Se cumple que (Panda & Naik, 2015):

$$S_T = S_W + S_B \tag{17}$$

4.4.2.2 *Procedimiento de Fisherfaces*

a. A partir de los vectores propios generalizados S_By S_w que corresponden a valores propios no nulos se construye la matriz de proyección.

b. Se proyecta la matriz anterior y se construye un vector de características para cada clase. Puede elegirse un vector por clase o promediarlos.

c. En ciertos casos, se debe reducir la dimensionalidad de los vectores de características y de los vectores de prueba para trabajar de mejor manera con S_B y S_w, optimizando la posterior clasificación.

d. Se toma como criterio de clasificación la distancia a los vectores de características.

4.4.3 *Transformada Discreta del Coseno (DCT, Discrete Cosine Transformation)*

La transformada discreta de coseno al igual que en PCA, utiliza niveles de intensidad para representar la imagen, obteniendo los valores de coeficientes más relevantes y ubicándolos en la parte superior izquierda de la matriz que realiza. De esta manera a través del uso de una pequeña cantidad de coeficientes es posible unir la mayor cantidad de información importante de la imagen sin afectar demasiado la imagen de salida. Al momento de encontrar todos los coeficientes de la DCT que sean necesarios, se escanean utilizando un sistema de Zigzag.

Con el uso de niveles de intensidad se ahorra de manera considerable el coste computacional de análisis y la utilización de la memoria en cada uno de los aspectos que se desarrollan en el programas, al reemplazar un sistema de NxM píxeles por un grupo de coeficientes. (Peláez Fernández & Ramón Morros, 2012).

4.4.4 *Proyecciones Preservando Localidad (LPP, Locality Preserving Projections)*

LPP es una técnica que al igual que LDA pretende disminuir la dimensión de los datos de entrada. El procedimiento que utiliza para construir una matriz de transformación es asociar los vectores a un sub-espacio, conservando su estructura local. El resultado de esto sería que tanto el "vecino" en el espacio original para un determinado dato será el mismo que el "vecino" en el nuevo sub-espacio. De esta manera las imágenes que no corresponden a un determinado individuo se encontraran alejadas del mismo, y las que estén acorde se encontraran cercanas entre sí. (Peláez Fernández & Ramón Morros, 2012)

4.5 Comparación

En esta última fase se compara la información obtenida con la que existía previamente en la base de datos. Los métodos de clasificación que se utilizan con mayor frecuencia en el estado del arte pueden dividirse en dos grupos: Medidas de Similitud o Distancia y Clasificadores; generalmente los clasificadores pueden utilizar distintas medidas de similitud. (Eslava Ríos, 2013).

En esta etapa se analiza la distancia entre el vector de la imagen capturada, a la cual se pretende realizar el reconocimiento, y los vectores que corresponden a las imágenes de entrenamiento. Es decir, se desea encontrar el vector de las imágenes de entrenamiento cuyo valor se encuentre más cercano al valor del vector de la imagen propuesta. En la figura 12 presenta un ejemplo de comparación de dos vectores, el vector y correspondiente a la imagen de entrenamiento y el vector $\hat{y}$ pertenece a la imagen analizada, ambos vectores alejados con una distancia d. El vector $\hat{y}$ que posea la menor distancia d respecto al vector de la imagen de entrenamiento corresponderá a la identidad del individuo de dicha imagen.

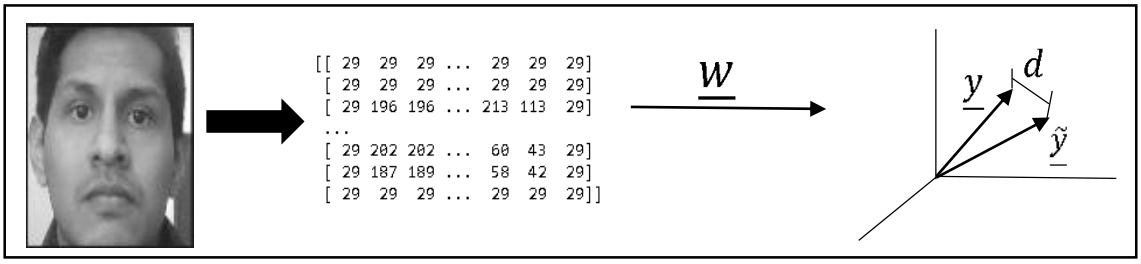


Figura 12. *Esquema que muestra la comparación de distancias entre dos vectores.*
Fuente: *(Cazorla Martínez, 2016)*

4.5.1 *Medidas de Similitud o Distancia*

4.5.1.1 *Distancia Euclideana*

Generalmente es usada para calcular la distancia directa entre dos puntos de un plano. Por ejemplo, dados dos puntos P_1 y P_2 con coordenadas x_1, y_1 y x_2, y_2 respectivamente, el cálculo de la distancia euclídea entre los mismos sería: (Eslava Ríos, 2013)

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (18)$$

4.5.1.2 *Chi-Square* ($\boldsymbol{\chi^2}$)

Esta distancia permite comparar distribuciones de probabilidad discretas[6]. Específicamente, en sistemas de reconocimiento biométrico, $\boldsymbol{\chi^2}$ permite calcular la distancia entre dos vectores que se han obtenido del análisis de sus histogramas. Dados dos histogramas S y M, la distancia *Chi-Square* se calcula así (Eslava Ríos, 2013):

$$\chi^2(S,M) = \sum_i \frac{(S_i+M_i)^2}{S_i+M_i} \qquad (19)$$

4.5.2 *Métodos de Clasificación*

4.5.2.1 *K-Nearest Neighbours* (Vecinos más Cercanos)

Este método también conocido como K-NN permite clasificar objetos a partir de las muestras de entrenamiento más cercanas al espacio de características. Este algoritmo encuentra los k vecinos más cercanos al objeto, para clasificarlos en función de la cantidad de los mismos, dentro del conjunto que contenga un mayor número de muestras cercanas (Eslava Ríos, 2013).

4.5.2.2 *Support Vector Machines* (SVM, Máquinas de Soporte Vectorial)

Estos modelos de aprendizaje son usados en regresión y clasificación, permiten representar distintas clases en un solo espacio para posteriormente localizar un híper-plano que divida dichas clases ubicándolas en distintas zonas, de esta manera, cuando un dato ingrese será ubicado y clasificado dentro de una de esas zonas. En *Support Vector Machines* la dimensión del espacio donde se representan las clases es igual al número de clases. En la figura 13 se muestra el ejemplo más sencillo de clasificación de dos clases. En este ejemplo, las clases (círculos negros y blancos) se dispersan en todo el plano de dos dimensiones (x_1 y x_2), mientras que los SVM intentan encontrar un híper-plano (en este caso una recta) que separe dichas clases de la mejor manera posible, esto es, una distancia considerable entre ambas clases. Este ejemplo es un caso ideal ya que existe gran separación entre clases, pero esto no es así en casos prácticos, usualmente las muestras de ambas clases se entremezclan originando problemas en la clasificación. La

[6] Una distribución de probabilidad discreta describe la probabilidad de ocurrencia de cada valor de una variable aleatoria discreta; un histograma es una representación común de este tipo de distribución de probabilidad. (Llinás Solano & Rojas Álvarez, 2017).

factibilidad de usar SVM depende principalmente del método de extracción de características empleado previamente (Eslava Ríos, 2013)

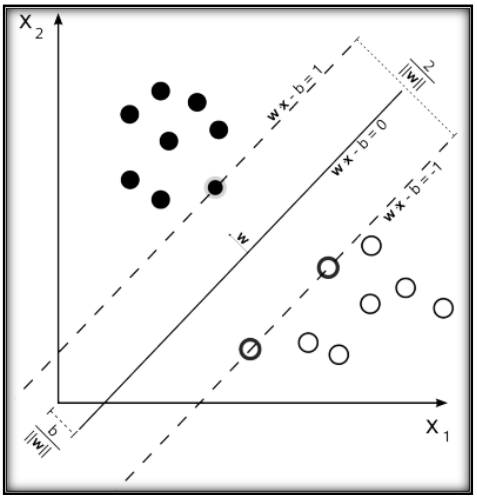


Figura 13. *Cálculo del plano que mejor divide las dos clases en un problema de dos dimensiones.*
Fuente: *(Eslava Ríos, 2013)*

4.5.2.3 *Gaussian Mixture Models* (GMM, Modelo de Mezclas Gaussianas)

En el campo biométrico los GMM se usan junto con *Expectation Maximization*[7] para aproximar los parámetros del mismo. Los GMM permiten representar, mediante sumas de componentes gaussianas, las funciones de densidad de probabilidad paramétricas (Eslava Ríos, 2013).

[7] *Expectation Maximization* o Algoritmo Esperanza-Maximización es un procedimiento que permite la maximización de una función de verosimilitud cuando los procedimientos estándar son numéricamente difíciles o inviables. El procedimiento consiste en definir una esperanza o expectativa en particular, y luego maximizarla. El procedimiento es iterativo, iniciándose en un cierto valor inicial de los parámetros y actualizando los valores en cada iteración. Los parámetros actualizados en cada iteración son los valores que maximizan la expectativa en esa iteración particular. (Train, 2014)

5. MATERIALES Y MÉTODOS

En el presente proyecto se pretende implementar una interfaz gráfica que sea capaz de clasificar y reconocer el rostro del usuario que ejecute el sistema, para realizar esta tarea previamente se debe implementar un algoritmo para el procesamiento de dicha imagen; existen diversos caminos para realizar el reconocimiento facial de una persona, sin embargo, en este trabajo se analizará las técnicas más conocidas de extracción de características y de clasificación de objetos con la finalidad de hacer un análisis comparativo y finalmente implementar el algoritmo que presente los resultados más precisos.

5.1 Selección de Modelos Principales

5.1.1 *Lenguaje de Programación*

Los lenguajes de programación más utilizados en reconocimiento facial se muestran en la tabla 1. Se eligió el lenguaje de programación Python debido a su estructura de código abierto, a las características que ofrece en cuanto a reconocimiento facial, por ejemplo: SK-Learn, numpy, matplotlib; además, actualmente es el lenguaje de programación que lidera las estadísticas de uso por parte de los programadores, especialmente en áreas relacionadas a la inteligencia artificial; dicha información ha sido recopilada por parte de GitHub, uno de los repositorios con mayor cantidad de usuarios y programadores a nivel mundial.(“Top programming languages in 2019: Python sweeps the board,” 2019). En el Anexo 3 se presentan las tablas de valoración de cada lenguaje de programación.

Así mismo, en la figura 14 se muestra una imagen comparativa de los lenguajes de programación más comunes según la página de IBM (International Bussines Machines Corporation).

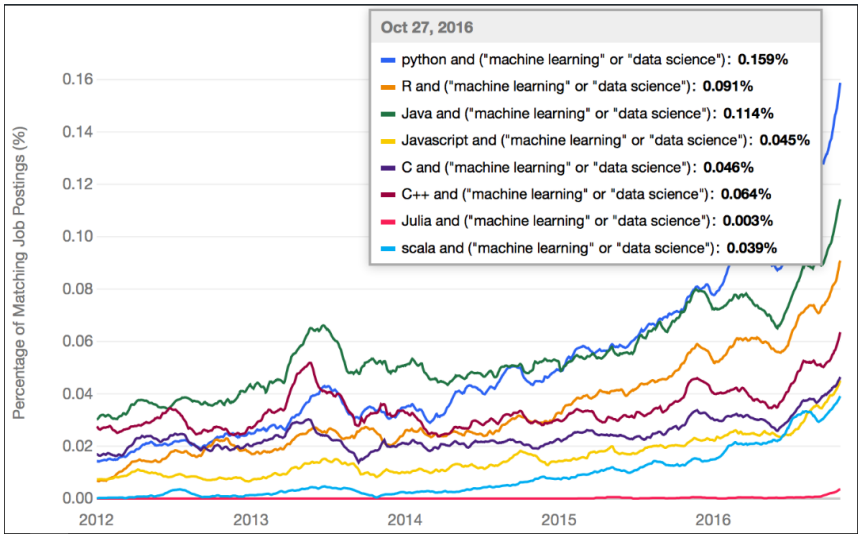


Figura 14. *Lenguajes de programación más usados según GitHub.*
Fuente: ("The Most Popular Language For Machine Learning Is," 2016)

Tabla 1. *Comparativa de Lenguajes de Programación*

Lenguaje de Programación	Características
Python (“About Python™ | Python.org,” 2019)	• Software de Código Abierto. • Actualmente calificado como lenguaje de programación más popular • Simplicidad y Versatilidad • Varias herramientas para el procesamiento de imágenes, como: escala de grises, recorte, enmascaramiento, rotación, convolución, además de muchos otros requisitos de procesamiento de imágenes normales. • Posee una de las bibliotecas más potentes y eficientes para modelos de reconocimiento facial, Scikit-Learn Machine Learning. • Se puede utilizar para crear funciones de procesamiento y reconocimiento de imágenes. • Existen varias bibliotecas construidas para que Python realice diferentes cálculos científicos contando el procesamiento de imágenes, el reconocimiento y la detección de movimiento. • Compatible con OpenCV.
C, C++, C# (“GitHub: The top 10 programming languages for machine learning,” 2019)	• Son potentes y se pueden utilizar en diversas aplicaciones, incluida la creación de procesamiento de imágenes y funcionalidades de reconocimiento. • Ofrece dos opciones para crear la función de procesamiento de imágenes, codificar todo desde cero mediante escribiendo los códigos manualmente o utilizar las bibliotecas existentes que están especialmente diseñadas para estos lenguajes de programación. • Está orientado a objetos y usa la plataforma .Net de Windows.
JavaScript (“GitHub: The top 10 programming languages for machine learning,” 2019)	• Este lenguaje es lo suficientemente potente como para realizar funcionalidades complejas. • Se puede usar para crear aplicaciones para el procesamiento de imágenes y el reconocimiento de imágenes. • Hay bibliotecas potentes que se pueden incorporar a Java con el fin de crear la función de reconocimiento de imágenes. (OpenCV)

5.1.2 *Algoritmo de Detección de Rostros*

Ya que en el presente trabajo se decidió desarrollar el proceso de reconocimiento facial en Python, es necesario implementar un algoritmo de detección que se encuentre disponible en la librería OpenCV de este lenguaje. En el anexo 4 se presenta la tabla de valoración de los algoritmos Viola and Jones y LBP, disponibles en OpenCV.

La mayor diferencia entre los detectores se encuentra en la precisión y velocidad de detección; para el presente estudio es necesario implementar el algoritmo que posea una tasa de detección mayor, por lo cual se eligió el algoritmo de Viola and Jones.

5.1.3 *Selección de las técnicas de Normalización*

En esta etapa se pretende minimizar el efecto de los cambios de iluminación que puedan existir en las diferentes fotografías, por lo cual se ha decido utilizar dos de las técnicas más comunes de mejora de contraste, las cuales son: Ecualización del Histograma y Corrección Gamma. La desventaja de la técnica de Corrección Gamma es que para elegir un índice adecuado de corrección, se debe evaluar los resultados con diferentes factores gamma, lo cual conlleva una gran cantidad de tiempo.

5.1.4 *Selección de la Técnica de Extracción de Características*

En la Tabla 3 se muestran diferentes algoritmos de Extracción de Características calificados de acuerdo a cuatro criterios: Complejidad Matemática, Entrenamiento Previo, Base de Datos, Procesamiento y Toma de Decisiones; en donde:

Para el criterio de Complejidad Matemática: 1 significa Complejidad Muy Alta y 5 significa sin Complejidad.

Para el criterio de Entrenamiento Previo: 1 significa Entrenamiento muy Complejo y 5 significa que No necesita entrenamiento.

Para el criterio de Base de Datos: 1 significa Cantidad de Datos almacenados muy grande y 5 significa Cantidad de Datos almacenados muy baja.

Para el criterio de Procesamiento y Toma de decisiones: 1 significa Muy Ineficiente y 5 significa Muy Eficiente.

La calificación puede estar entre 1 y 5, y la ponderación es el resultado de la calificación por el pese de cada criterio. (Pillajo Coka & Yaguana Montero, 2018)

Tabla 2. *Comparativa de distintas técnicas de Extracción de Características*

CRITERIOS / MÉTODOS	COMPLEJIDAD MATEMÁTICA		ENTRENAMIENTO PREVIO		BASE DE DATOS		PROCESAMIENTO Y TOMA DE DECISIONES		SUMA TOTAL
	0,4		0,1		0,3		0,2		
	CAL.	POND.	CAL.	POND.	CAL.	POND.	CAL.	POND.	
PCA	3	1,2	3	0,3	4	1,2	4	0,8	3,5
LDA	2	0,8	3	0,3	3	0,9	4	0,8	2,8
LPP	2	0,8	3	0,3	4	1,2	3	0,6	2,9

De acuerdo a la Tabla 2 se seleccionó la técnica PCA ya que obtiene una mayor calificación; así mismo, se decidió implementar la técnica LDA, debido a que generalmente esta técnica hace uso de los datos obtenidos por PCA. Cabe recalcar que el parámetro “Base de Datos” en el cual LDA obtiene un puntaje inferior a LPP no es de relevancia en el presente trabajo ya que la Base de Datos utilizada para todas las técnicas es la misma.

5.1.5 *Selección del algoritmo Clasificador*

Se decidió utilizar el algoritmo clasificador KNN debido a que es un algoritmo simple, fácil de explicar y comprender; sin embargo, posee una precisión elevada; es bastante tolerante al ruido y además, se encuentra disponible como parte de la librería SK-Learn de Python. Su principal desventaja es que su costo computacional está directamente relacionado a la cantidad de datos a analizar; no obstante, al utilizarlo sobre conjuntos de datos cuya dimensión ha sido reducida se convierte en un clasificador altamente eficaz. La tabla de valoración de los algoritmos clasificadores se muestra en el Anexo 5.

5.2 Descripción General

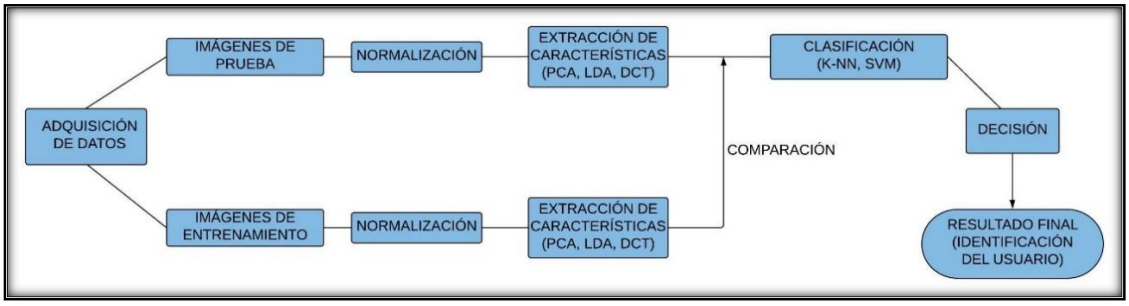


Figura 15. *Descripción del algoritmo de entrenamiento*
Fuente: *Autor*

En la figura 14 se muestra el proceso de entrenamiento del algoritmo; se debe señalar que para realizar el entrenamiento, se utilizará bases de datos ya creadas como ORL (Olivetti Datasearch Laboratory) y MIT (Massachusetts Institute of Technology); cuyas características se describirán más adelante; una vez entrenado el algoritmo se hará uso de la librería OpenCV de Python para realizar la adquisición de fotos de los distintos usuarios que ingresen al sistema; OpenCV realiza la detección del rostro del usuario basándose en el algoritmo Viola and Jones descrito anteriormente.

Para el entrenamiento del algoritmo, como primer paso se puede optar por un proceso de normalización de las imágenes, para el cual se analizará los resultados al aplicar técnicas de ecualización del histograma y corrección gamma sobre las imágenes de las base de datos ORL y MIT.

En la extracción de características de las imágenes se implementará el Análisis de Componentes Principales (PCA) y el Análisis de Discriminantes Lineales (LDA).

En la etapa de clasificación se utilizará el algoritmo clasificador KNN provisto por la librería SK-Learn del módulo OpenCV, adicionalmente, se ha decidido desarrollar un algoritmo clasificador basado en la técnica KNN, con la finalidad de comparar resultados entre ambos algoritmos. El algoritmo desarrollado por el autor se muestra en el Apéndice B.

Finalmente, se utilizará la librería SK-Learn para realizar una valoración numérica de los porcentajes de precisión y error cuadrático medio[8] obtenidos en el sistema desarrollado.

5.2.1 *Recopilación Bibliográfica*

Se ha tomado como referencia las investigaciones relacionadas al reconocimiento facial y, profundizando en la información descrita en el estado del arte, se pudo constatar que el reconocimiento de rostros presenta un umbral de precisión entre un intervalo de 90% y 99%, por lo que el algoritmo que se desarrollará en esta investigación deberá alcanzar resultados en torno a este umbral

5.2.1.1 *Recopilación de datos*

Se debe mencionar que para alcanzar los porcentajes de precisión del estado del arte, los autores de cada investigación trabajaron con distintas bases de datos y emplearon diferentes tiempos de entrenamiento, además, la capacidad de procesamiento variaba acorde al dispositivo utilizado, debido a estas razones no existe un porcentaje de precisión único.

5.2.1.2 *Análisis de Datasets*

Existen gran variedad de bases de datos que pueden ser obtenidas gratuitamente a través de internet, las cuales contienen imágenes de rostros humanos, de la misma persona en distintas posiciones; ejemplo de estos datasets son los propuestos por la ORL (Olivetti Dataserch Laboratory), Labeled Faces in the Wild, FERET (Face Recognition Technology), CMU (Carnegie Mellon University), MIT-CBCL, entre otras.

Tabla 3.*Características Principales de las Bases de Datos*

	Num. de usuarios	**Imag. por usuario**	**Dimensión (Pixeles)**	**Posición Frontal**	**Var. de Iluminación**	**Var. de pose**	**Var. De Gestos**
ORL	40	10	92 x 112	X	X		X
MIT	10	324	200 x 200	X	X	X	

[8] El error cuadrático medio mide el promedio de los errores al cuadrado; es decir, la diferencia entre el estimador y lo que se estima; en este caso, el estimador es el nombre del usuario que ingresa al sistema y lo que se estima es el nombre de usuario obtenido por el sistema.

Para el presente trabajo se ha elegido la base de datos presentada por ORL, la cual muestra imágenes en gris, 10 imágenes por persona de 40 personas, haciendo un total de 400 imágenes; dichas imágenes tiene dimensiones de 92 x 112 pixeles y todas son tomadas en posición frontal con cierto ángulo de inclinación del rostro.(AT&T Laboratories Cambridge, 2002)

En adición, se utilizó la base de datos MIT, la cual contiene dos datasets de entrenamiento y un dataset de prueba; en el dataset de entrenamiento se encuentran imágenes de la cara de 10 sujetos distintos. Las imágenes se encuentran en alta resolución e incluyen vista frontal, vista de medio perfil y vista de perfil completo. El segundo dataset de entrenamiento contiene imágenes renderizadas a partir de modelos de cabeza 3D de 10 sujetos. El dataset de prueba consiste de 200 imágenes por sujeto en las cuales se ha variado la iluminación, pose y el fondo.(Massachusetts Institute of Technology, 2005). Se utilizó 100 imágenes del primer dataset de entrenamiento, 10 imágenes por cada usuario.

Se analizará los resultados al trabajar solamente con el dataset ORL (40 usuarios) y posteriormente se comparará con los resultados obtenidos al añadir el dataset MIT (50 usuarios).

5.3 Fundamentos

5.3.1 *Bases del reconocimiento facial*

El rostro humano puede ser considerado como un conjunto de características únicas de cada individuo que nos permite identificar y clasificar a simple vista distintas personas, partiendo de este concepto, dichas características pueden ser extraídas digitalmente y ser utilizadas por un sistema de reconocimiento facial. Los rasgos más significativos que a su vez aportan la mayor cantidad de información para que un sistema pueda identificar y clasificar a cada individuo son los ojos, las cejas, las orejas, la nariz y la boca; por lo que se debe procurar en lo posible que estos rasgos sean captados en su totalidad al momento de almacenar la imagen del usuario.

Sin embargo, existen ciertos factores que pueden disminuir la precisión de un sistema de reconocimiento facial, uno de ellos es la presencia de vello facial (barba), o la presencia

de ciertas prendas de vestir al momento de capturar la imagen como gorras, gafas de sol y bufandas.

5.3.2 *Algoritmos*

Considerando la recopilación bibliográfica y el estado del arte del reconocimiento facial, se hará énfasis en las técnicas de extracción de características PCA y LDA; y de clasificación K-NN con los cuales se ha alcanzado los porcentajes mencionados anteriormente.

Cabe recalcar que para entrenar adecuadamente los algoritmos PCA y LDA se ha decidido dividir el dataset de imágenes en dos conjuntos:

- El primer dataset será de entrenamiento y contendrá 5 imágenes por usuario.
- El segundo dataset será de prueba y contendrá 5 imágenes por usuario.

5.3.2.1 *PCA*

El método PCA pretende extraer de manera no supervisada las características principales del rostro, almacenándolas en vectores de características PCA, los cuales son robustos al ruido y pequeñas rotaciones de la cabeza, pero los cambios de iluminación disminuyen considerablemente su precisión.

Inicialmente, se procede a representar cada imagen como una matriz de los valores de intensidad en gris para su posterior procesamiento. En la figura 15 se puede observar la representación de dichos valores extraídos de imágenes del mismo sujeto perteneciente al dataset ORL.

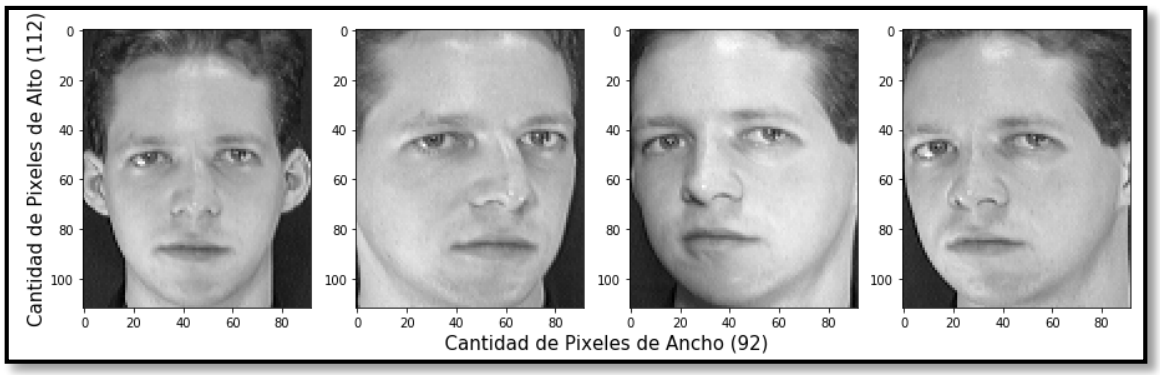


Figura 16. *Matrices de intensidades de gris graficadas en Python*
Fuente: *Autor*

A continuación, la etapa de extracción de características se trabaja sobre el grupo de entrenamiento. En esta etapa, las matrices son aplanadas, es decir, se redimensiona la matriz colocando todos sus valores de intensidades en una sola fila, por ejemplo, si una matriz tiene dimensión 92 x 122, luego de aplanarla su nueva dimensión será de 1 x [(92)(112)], lo que equivale a una dimensión de 1 x 10304.

Una vez aplanadas todas las matrices, se las coloca en una sola matriz de matrices, y se las ordena de tal manera que la nueva matriz tendrá una dimensión de [10304x200], en donde cada columna corresponderá a los datos de cada imagen.

Se calcula un vector de valores medios de cada fila de esta matriz, por lo que si se tiene 10304 filas, el vector de valores medios contendrá 10304 valores.

Posteriormente, se calculará la matriz de dispersión de todo el conjunto de entrenamiento, para lo cual simplemente se aplica la Ec. (9) y (10), esto debido a la elevada dimensionalidad [10304 x 10304] que poseería si se aplica la Ec. (8); por lo tanto la matriz de dispersión poseerá una dimensión de [número de fotos x número de fotos]

A continuación, se debe calcular los autovalores y autovectores de la matriz de dispersión, usando la librería Numpy de Python, se los ordena ascendentemente, y finalmente se proyecta los autovectores en la matriz de entrenamiento y en la matriz de test para obtener la matriz PCA train inicial y matriz PCA test inicial respectivamente.

Para reducir la dimensionalidad de las matrices PCA, se deberá elegir adecuadamente el número de autovectores que describan de mejor manera las características de cada usuario, el número elegido es de 53 autovectores, dicha elección se describirá posteriormente en la etapa de alineación. La nueva matriz poseerá entonces una dimensión de [#autovectores x 200 imágenes]. En la figura 16 y 17 se muestran las gráficas correspondientes a los 5 primeros autovectores y los 5 últimos autovectores; es decir, los más significativos y menos significativos respectivamente.

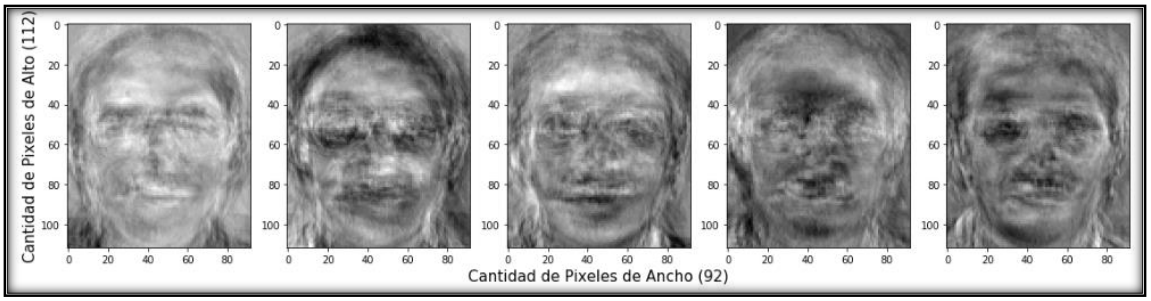


Figura 17. *Cinco autovectores más significativos del conjunto de imágenes ORL*
Fuente: *Autor*

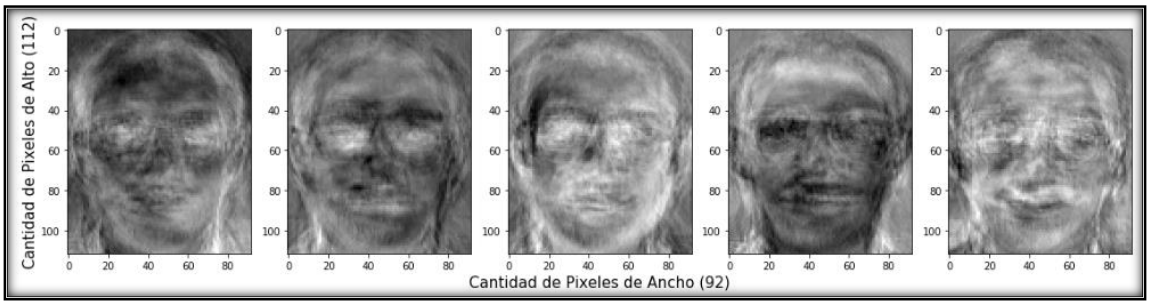


Figura 18. *Cinco autovectores menos significativos del conjunto de imágenes ORL*
Fuente: *Autor*

5.3.2.2 *LDA*

El método LDA extrae las características faciales de manera supervisada, es decir, tiene en cuenta las etiquetas de clase y maximiza la dispersión entre clases bajo la restricción de que la dispersión dentro de la clase se minimiza.

Este método toma como parámetro inicial la matriz PCA de entrenamiento con dimensión [#autovectores x 200 imágenes] calculada anteriormente, con la finalidad de aprovechar los cálculos que ya fueron efectuados y disminuir el coste computacional del algoritmo.

La matriz de train PCA es ordenada de tal forma que cada usuario posea su matriz respectiva con 5-submatrices (imágenes por usuario) dentro; es decir la nueva matriz tendrá una dimensión de [40 usuarios x 5 imágenes x # autovectores].

A continuación, para calcular la matriz de dispersión para cada usuario (40 matrices) se utiliza la Ec. (8), tomando en cuenta que la que la matriz de dispersión final tendrá una dimensionalidad [# autovectores x #autovectores], en donde el número de autovectores elegidos será menor a 200, lo cual hace posible utilizar dicha fórmula.

Posteriormente, se suma todas las matrices y se obtiene una nueva matriz que será nombrada Matriz de Dispersión Intraclase (Scattering Within), la cual tendrá una dimensión de [# autovectores x # autovectores], en la figura 20 se muestra esta matriz.

El siguiente paso consiste en calcular la Matriz de Dispersión Inter-Clase (Scattering Between), utilizando la Ec. (12), para lo cual es necesario previamente calcular un vector de medias (μ_c) por cada usuario y un vector de medias total $\overline{x}$.

La matriz de dispersión final está dada por la Ec. (17).

El paso final es similar al proceso realizado en PCA, y consiste en calcular los autovalores y autovectores de esta matriz de dispersión final, para posteriormente proyectarlos en la matriz de entrenamiento y en la matriz de test. La matriz resultante poseerá una dimensión igual al número de autovectores elegidos en PCA por el número de imágenes presentes en la base de datos [53 x 200]; en la figura 18 se muestran las primeras 5 Fisherfaces calculadas en la base de datos ORL.

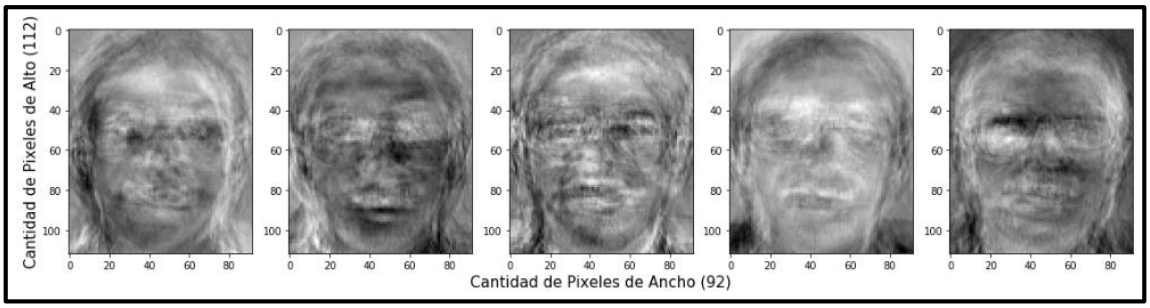


Figura 19. *Primeras 5 Fisherfaces de la base de datos ORL*
Fuente: *Autor*

Sin embargo, el algoritmo LDA permite disminuir incluso más esta dimensión, por lo que se debe elegir adecuadamente el número óptimo de autovectores que permita esta reducción, proceso que se describirá en la etapa de alineación del algoritmo.

5.4 Alineación

En esta etapa se seleccionarán adecuadamente el número de autovectores que optimicen los resultados del algoritmo, el proceso de selección se realizó para el método de Análisis de Componentes Principales (PCA), para el Análisis Lineal de Discriminantes (LDA), para cada normalización (Ecualización del Histograma, Correccción Gamma) y para las distintas cantidades de usuarios (40 usuarios y 50 usuarios); es decir, cada técnica analizada obtendrá su respectiva cantidad de autovectores como resultado de la alineación

Como proceso inicial se realiza la proyección de la matriz de entrenamiento con cada autovector acumulado; es decir, en primera instancia se proyecta la matriz de train con un autovector, luego con dos autovectores, y se continúa este proceso hasta proyectar todos los autovectores.

Posteriormente, se calcula el error cuadrático medio de cada proyección y se realiza una gráfica en función de este error versus el número de autovectores; en esta gráfica, bajo criterio del autor, se seleccionó una zona plana en la cual el error cuadrático medio no aumente en exceso pero el número de autovectores se reduzca en gran medida. El objetivo es disminuir la mayor cantidad posible el número de autovectores pero conservando un error cuadrático medio bajo.

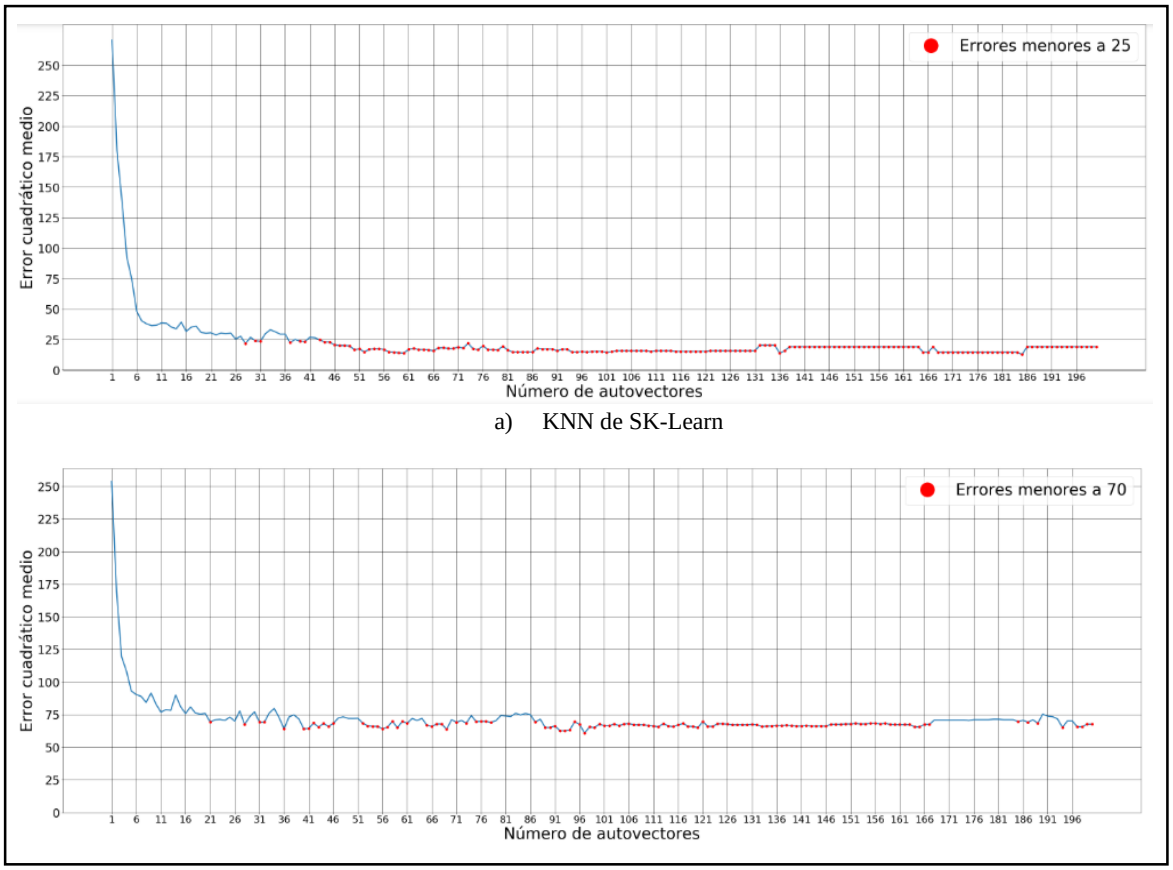
El error cuadrático medio se calcula al utilizar el clasificador SK-Learn y también con el clasificador desarrollado por el autor; además el análisis se realiza para uno, dos, tres, cuatro y cinco vecinos cercanos, y de la misma forma, para imágenes normalizadas, sin normalizar, con base de datos de 40 usuarios y 50 usuarios. Este análisis permitirá elegir minuciosamente todos los parámetros que optimicen adecuadamente el resultado final.

5.4.1 *Alineación de PCA*

En la Alineación de PCA para fotos sin normalizar, solo se mostrarán las gráficas correspondientes a 2 vecinos cercanos, las gráficas correspondientes a los demás vecinos cercanos se mostrarán en el Anexo 6.

5.4.1.1 *Para 40 usuarios*

A continuación, en la figura 19 se muestran las gráficas de error cuadrático medio obtenidas al usar clasificación KNN con 2 vecinos cercanos.



a) KNN de SK-Learn

b) KNN desarrollado por el autor

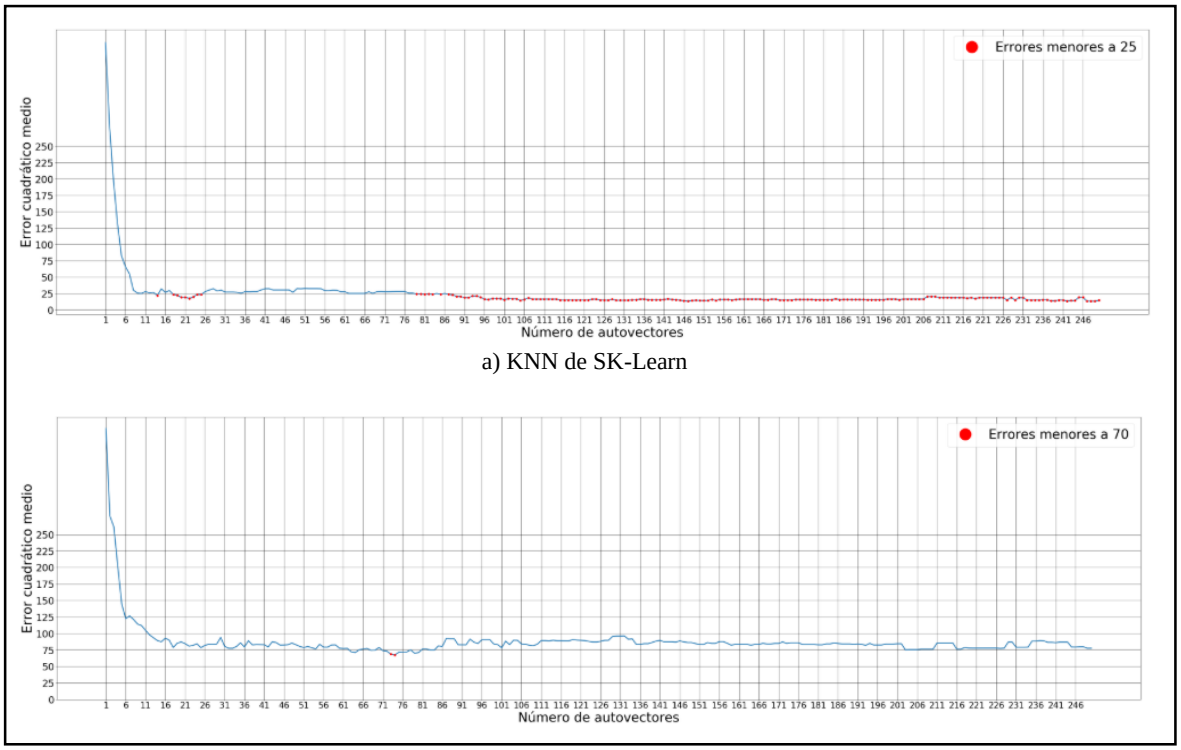
Figura 20. *Gráficas de Error Cuadrático Medio para fotos sin normalizar usando 40 usuarios*

Fuente: *Autor*

De las gráficas mostradas anteriormente se decidió utilizar un total de 53 autovectores para fotos sin normalizar.

5.4.1.2 *Para 50 usuarios*

En la figura 20 se puede observar el error cuadrático medio al usar 2 vecinos cercanos.



a) KNN de SK-Learn

b) KNN desarrollado por el autor

Figura 21. *Gráficas de Error Cuadrático Medio para fotos sin normalizar usando 50 usuarios*
Fuente: *Autor*

Al incrementar el dataset a 50 usuarios y analizando las gráficas anteriores, se decidió utilizar un total de 22 autovectores.

5.4.2 *Alineación de PCA aplicando Ecualización del Histograma*

Partiendo de la alineación realizada previamente, se evidenció que utilizando dos vecinos cercanos se obtiene un valor pequeño de error cuadrático medio, por lo que en las etapas posteriores correspondientes a PCA solamente se analizará la gráfica de errores cuadráticos medios correspondientes a esta cantidad de vecinos cercanos (2 vecinos); sin embargo, en las tablas de resultados se mostrarán los valores obtenidos al analizar con uno, dos, tres, cuatro y cinco vecinos cercanos.

5.4.2.1 *Para 40 usuarios*

La gráfica del error cuadrático medio al utilizar 2 vecinos cercanos se muestra en la figura 21, analizando esta gráfica se decidió utilizar 59 autovectores en fotos que presentan ecualización del histograma

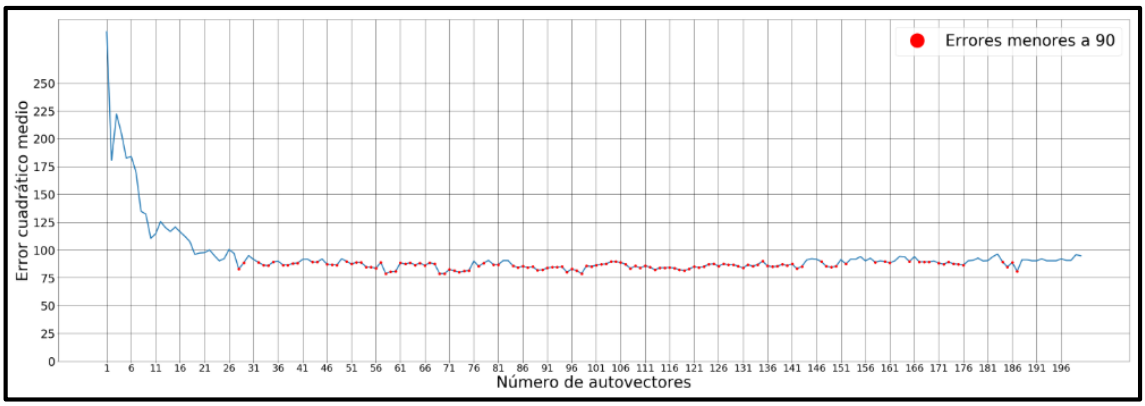


Figura 22. *Gráfica de Error Cuadrático Medio para fotos con Ecualización de Histograma usando 40 usuarios*

Fuente: *Autor*

5.4.2.2 *Para 50 usuarios*

En la figura 22 se muestra la curva de error cuadrático medio obtenida al clasificar usando 2 vecinos cercanos, a partir de los datos obtenidos se decidió usar 62 autovectores.

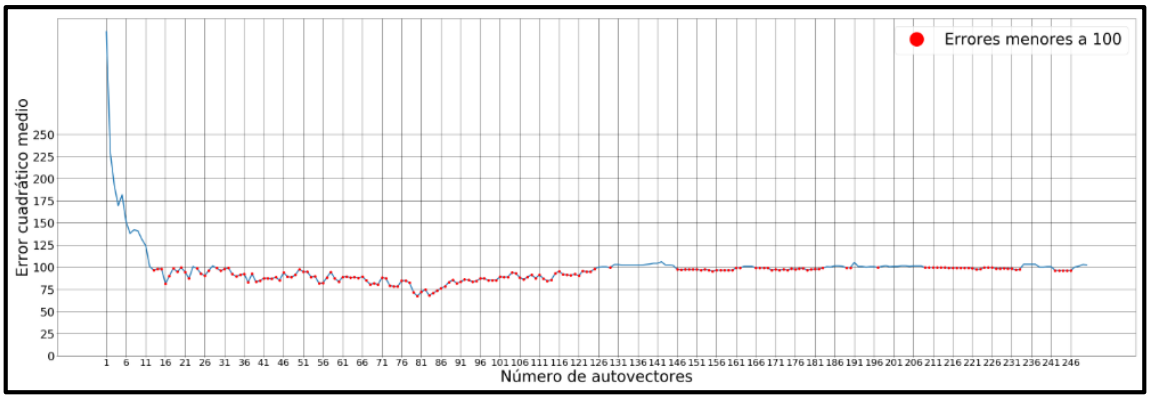


Figura 23. *Gráfica de Error Cuadrático Medio para fotos con Ecualización de Histograma usando 50 usuarios*

Fuente: *Autor*

5.4.3 *Alineación de PCA usando Corrección Gamma*

5.4.3.1 *Para 40 usuarios*

El paso previo a la elección del número adecuado de autovectores es la elección del factor de corrección gamma adecuado, para lo cual se realizaron distintas pruebas y finalmente se decidió utilizar un factor de corrección gamma igual a 11.

De la misma manera en la figura 23 se muestra la gráfica de los errores cuadráticos medios utilizando 2 vecinos cercanos para el clasificador KNN.

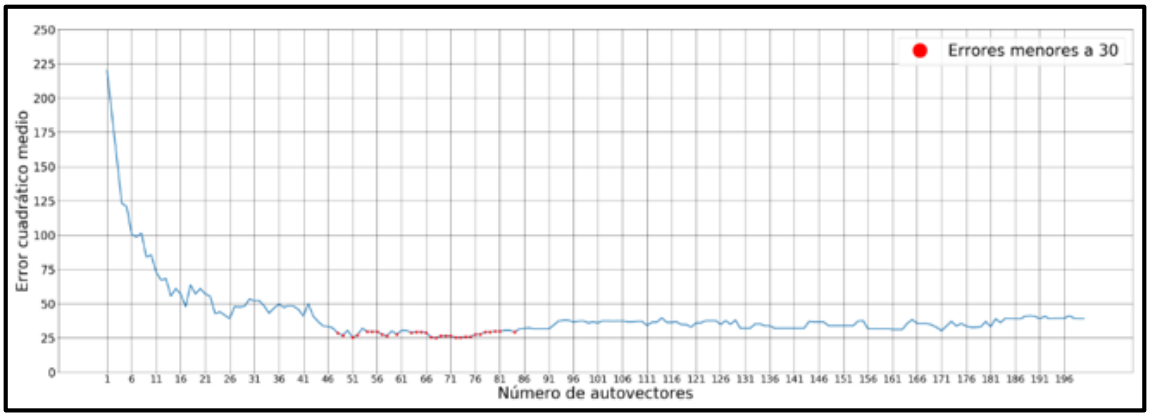


Figura 24. *Gráfica de Error Cuadrático Medio para fotos con factor de corrección gamma igual a 11 usando 40 usuarios*

Fuente: *Autor*

De los datos analizados se decidió elegir 68 autovectores para fotos normalizadas con factor de corrección gamma

5.4.3.2 *Para 50 usuarios*

Para el dataset de 50 usuarios se ha establecido un factor de corrección gamma igual a 60; en la figura 24 se muestra la gráfica de errores cuadráticos medios obtenida al usar dicho factor.

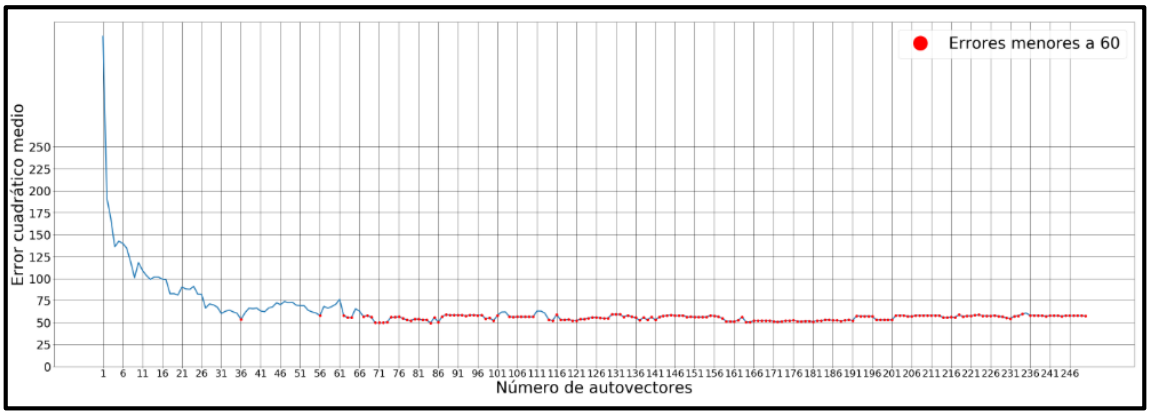


Figura 25. *Gráfica de Error Cuadrático Medio para fotos con factor de corrección gamma igual a 60 usando 50 usuarios*

Fuente: *Autor*

Con un dataset de 50 usuarios, se decidió utilizar 71 autovectores para un factor de corrección gamma igual a 60.

5.4.4 *Alineación de LDA*

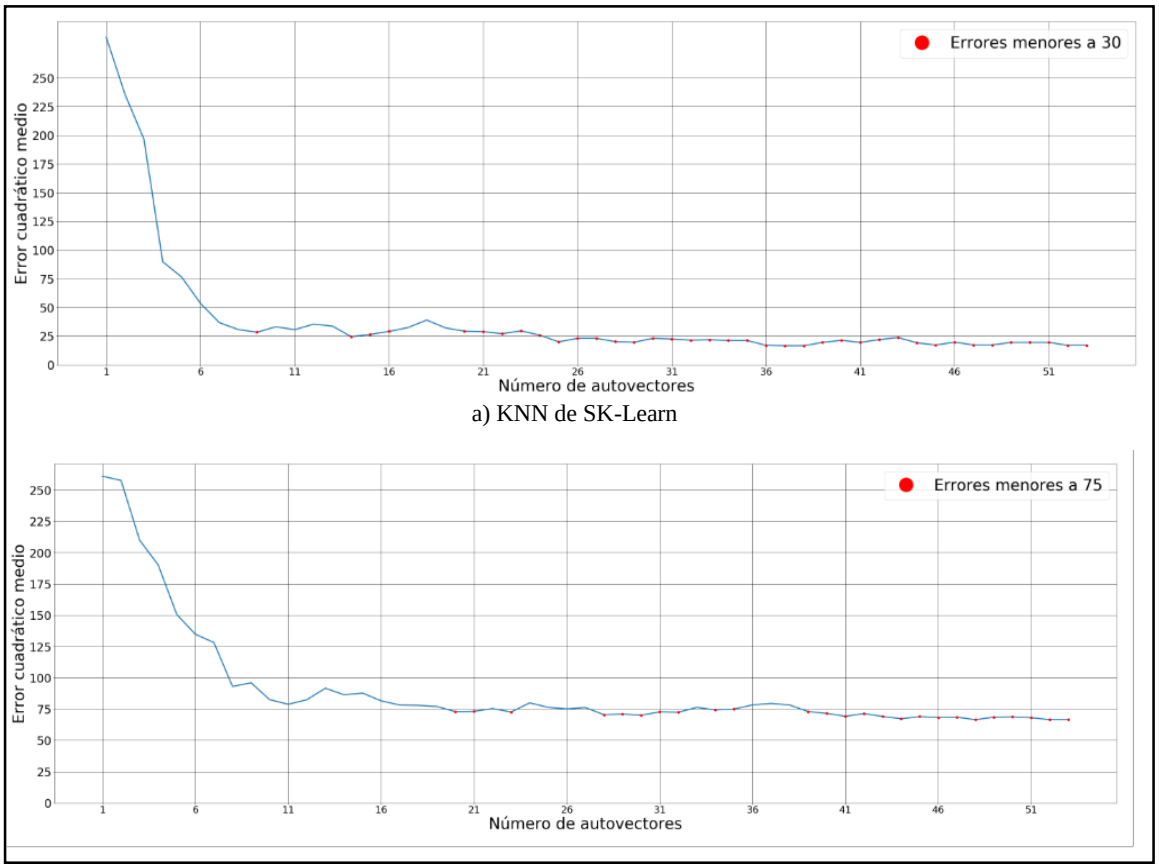
Para elegir el número adecuado de autovectores en LDA, se establece como número máximo de autovectores al número elegido en PCA; de esta manera se garantiza que la

reducción de dimensionalidad sea aún mayor. Este criterio se tomará tanto en las fotos normalizadas como en las fotos sin normalizar.

Así mismo, al igual que en la Alineación PCA, en la Alineación de LDA para fotos sin normalizar solo se mostrarán las gráficas correspondientes a 2 vecinos cercanos, las gráficas correspondientes a los demás vecinos cercanos se mostrarán en el Anexo 7.

5.4.4.1 *Para 40 usuarios*

En la figura 25 se muestran las gráficas de error cuadrático medio correspondientes a dos vecinos cercanos.



a) KNN de SK-Learn

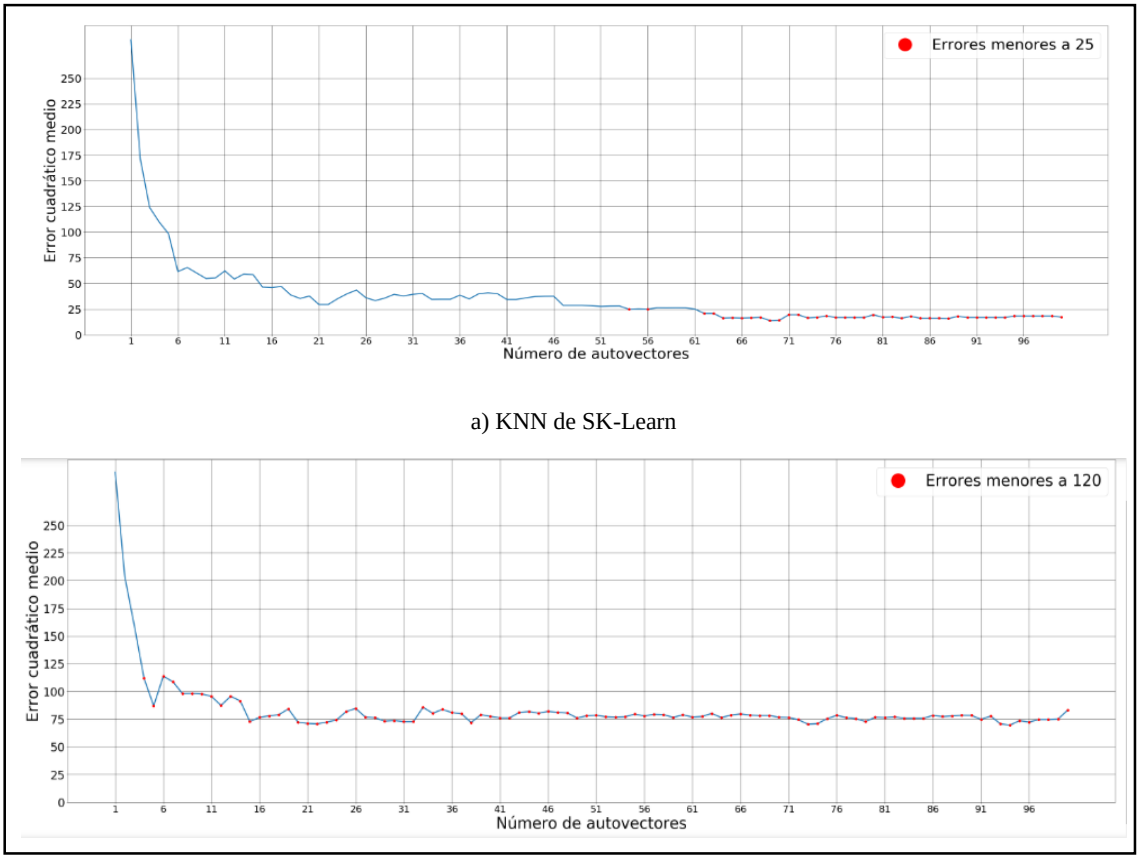
b) KNN desarrollado por el autor

Figura 26. *Gráficas de Error Cuadrático Medio en LDA para fotos sin normalizar usando 40 usuarios*
Fuente: *Autor*

Analizando las gráficas anteriores se decidió elegir un total de 38 autovectores para fotos sin normalizar.

5.4.4.2 *Para 50 usuarios*

Se decidió utilizar 100 autovectores como máximo con la finalidad de mejorar los errores cuadráticos medios obtenidos. En la figura 26 se observan las gráficas al analizar los datos con dos vecinos cercanos.



a) KNN de SK-Learn

b) KNN desarrollado por el autor

Figura 27. *Gráficas de Error Cuadrático Medio en LDA para fotos sin normalizar usando 50 usuarios*
Fuente: *Autor*

En el dataset de 50 usuarios se ha decido reducir la dimensionalidad a un número de 21 autovectores para fotos sin normalizar.

5.4.5 *Alineación de LDA con Ecualización del Histograma*

5.4.5.1 *Para 40 usuarios*

De la misma forma se decidió iniciar con el número de autovectores elegidos en el método PCA al usar ecualización del histograma, el cual fue de 53 autovectores. En la figura 27 se muestra la gráfica de errores cuadráticos medios correspondiente a esta normalización usando KNN con 2 vecinos cercanos.

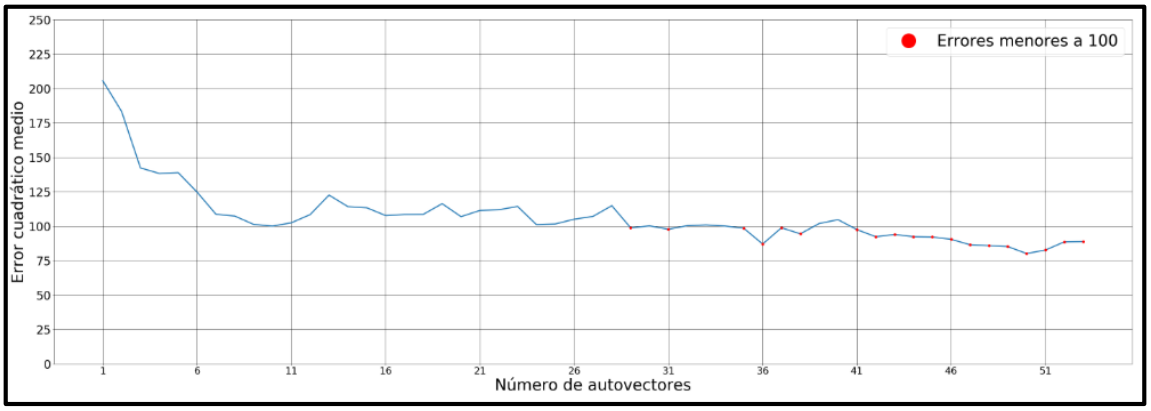


Figura 28. *Gráfica de Error Cuadrático Medio de LDA para fotos con Ecualización de Histograma usando 40 usuarios*

Fuente: *Autor*

A partir de la gráfica mostrada se decidió elegir 36 autovectores para optimizar los resultados.

5.4.5.2 *Para 50 usuarios*

Se inicia el proceso de análisis con 100 autovectores, en la gráfica 28 se presentan los errores cuadráticos medios con la finalidad de minimizar este número de autovectores. Luego del análisis de la gráfica se decidió reducir la dimensionalidad a 40 autovectores.

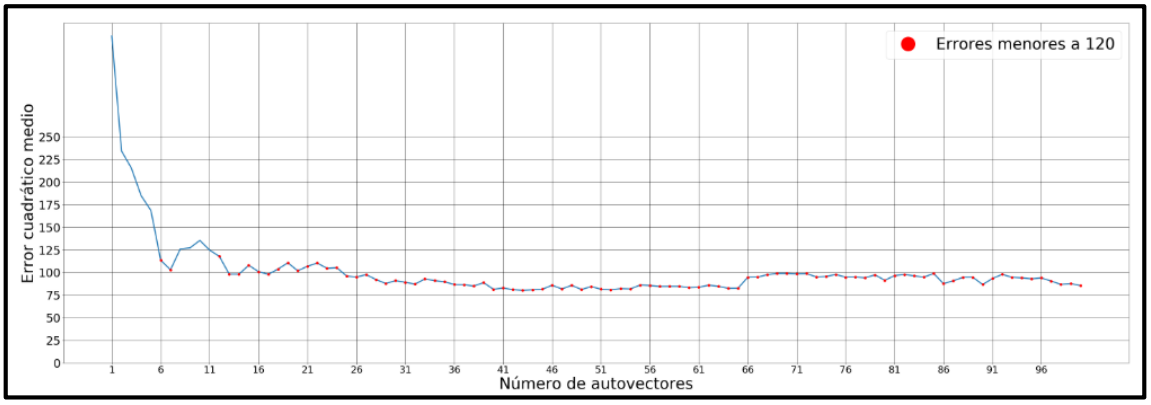


Figura 29. *Gráfica de Error Cuadrático Medio de LDA para fotos con Ecualización de Histograma usando 50 usuarios*

Fuente: *Autor*

5.4.6 *Alineación de LDA usando Corrección Gamma*

5.4.6.1 *Para 40 usuarios*

El paso previo a la reducción de dimensionalidad es elegir el factor de corrección gamma adecuado que minimice el error cuadrático medio, luego de analizar con diferentes factores se decidió elegir un factor de corrección igual a 16.

En la figura 29 se muestran los errores cuadráticos medios obtenidos al clasificar con 2 vecinos cercanos y al normalizar las imágenes con dicho factor de corrección.

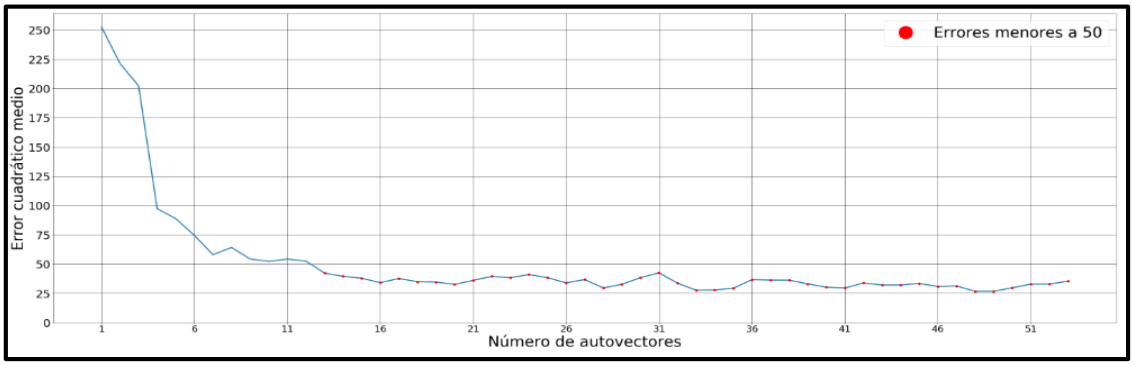


Figura 30. *Gráfica de Error Cuadrático Medio de LDA para fotos con factor de corrección gamma igual a 16 usando 40 usuarios*

Fuente: *Autor*

Analizando la gráfica 28, se decidió utilizar 34 autovectores en imágenes normalizadas con factor de corrección gamma para reducir la dimensionalidad de la matriz de dispersión.

5.4.6.2 *Para 50 usuarios*

Así mismo, usando 50 usuarios se eligió un factor de corrección gamma igual a 60. En la figura 30 se muestran los errores cuadráticos medios al usar este factor de corrección.

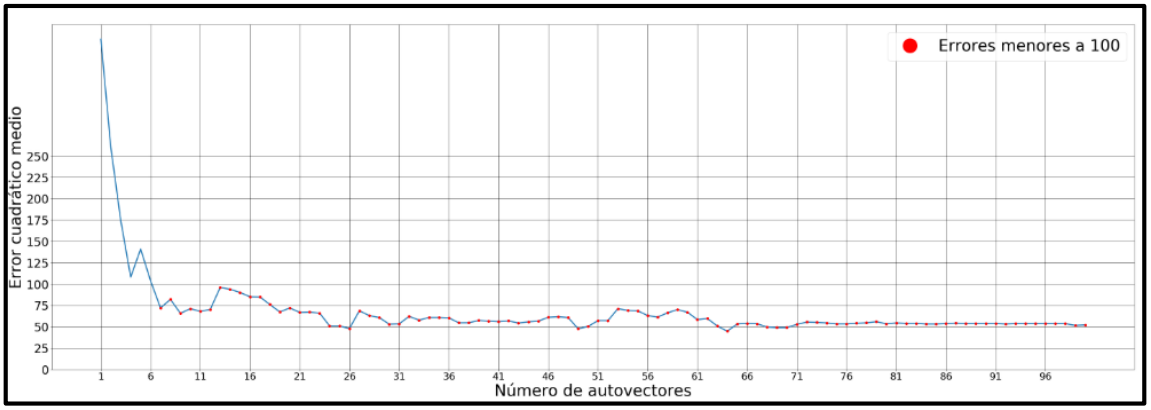


Figura 31. *Gráfica de Error Cuadrático Medio de LDA para fotos con factor de corrección gamma igual a 60 usando 50 usuarios*

Fuente: *Autor*

Analizando la figura 30 se ha decidido disminuir la cantidad de vectores propios a 64.

En la tabla 4 se muestra un resumen con el número de autovectores elegido para cada normalización, recordando que para todas las alineaciones se decidió utilizar 2 vecinos cercanos para la clasificación KNN

Tabla 4. *Resumen de la Cantidad de Autovectores Elegido en cada alineación*

		Num. de Autovectores entrada	Núm. de Autovectores reducido
PCA	40 Usuarios	200	53
	50 Usuarios	250	22
PCA con Ecualización	40 Usuarios	200	59
	50 Usuarios	250	62
PCA con Correc. Gamma	40 Usuarios	200	68, γ = 11
	50 Usuarios	250	71, γ = 60
LDA	40 Usuarios	53	38
	50 Usuarios	100	21
LDA con Ecualización	40 Usuarios	53	36
	50 Usuarios	100	40
LDA con Correc. Gamma	40 Usuarios	53	34, γ = 16
	50 Usuarios	100	64, γ = 60

5.5 Representación

Una vez entrenado correctamente el algoritmo y establecidos los parámetros para su funcionamiento óptimo, se procede a crear una base de datos propia con imágenes en tonos de gris capturadas desde la videocámara del computador portátil, dichas imágenes mostrarán solamente el rostro detectado del usuario, utilizando la librería OpenCV para Python. Para construir la base datos se utilizó una estructura de directorios en la que se creó un directorio con el nombre de cada usuario, dentro de cada directorio se almacenarán 10 imágenes en gris del usuario captadas por la videocámara; en la figura 31 se puede observar la detección adecuada del rostro usando OpenCV.

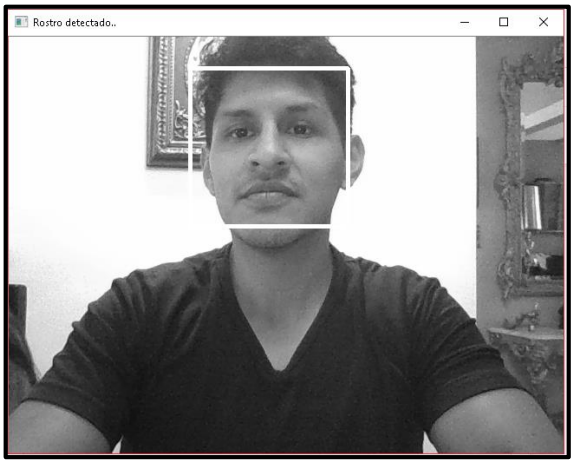


Figura 32. *Detección de Rostro a través de OpenCV*
Fuente: *Autor*

Las imágenes de cada usuario en la base de datos se almacenarán bajo el formato: "Apellido Nombre_0", "Apellido Nombre_1" hasta la imagen "Apellido Nombre_9" dentro de su directorio correspondiente, además, se separará las imágenes en conjuntos de entrenamiento y conjuntos de prueba como se muestra en la figura 32, posteriormente se extraerá los vectores de características del conjunto de entrenamiento usando PCA o LDA, los datos extraídos serán almacenados y utilizados en la futura clasificación.

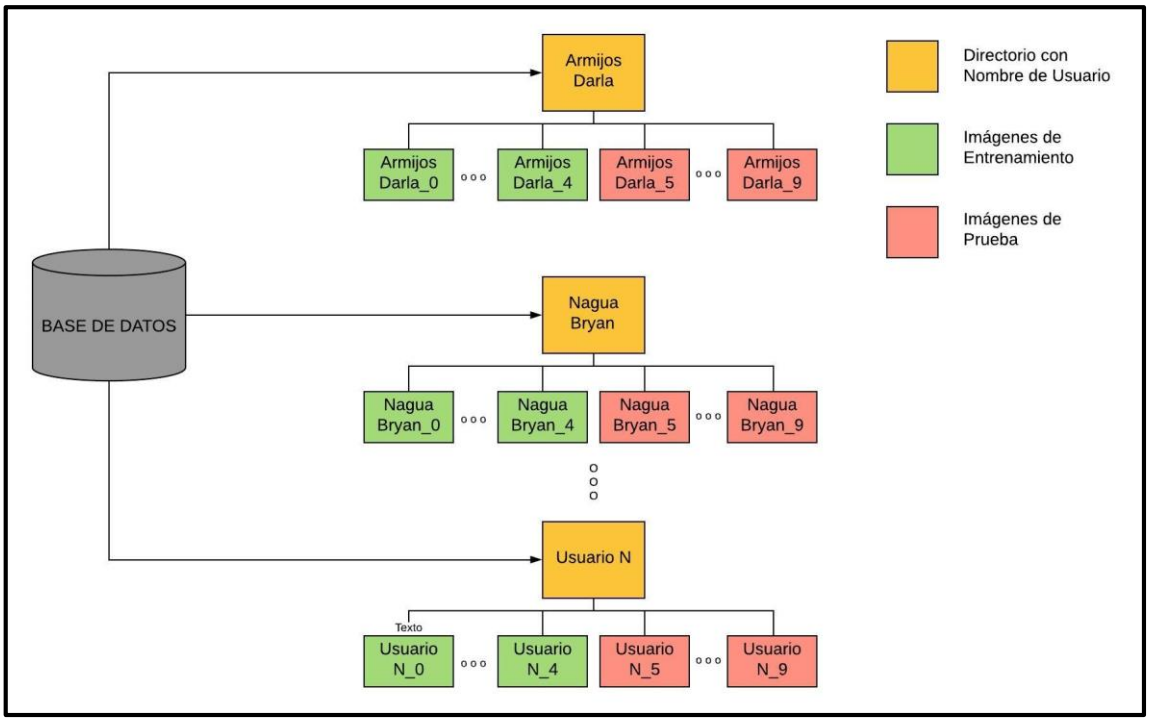


Figura 33. *Esquema de creación de directorios, grupos de entrenamiento y grupos de prueba*
Fuente: Autor

A continuación, se asignará el conjunto de imágenes de prueba como entrada del sistema, se repetirá el proceso de extracción de características; se comparará las características de las imágenes de entrenamiento con la imagen ubicada a la entrada del sistema mediante el algoritmo de clasificación K-NN y, finalmente, en la etapa de clasificación se devolverá como salida del sistema la identificación del usuario que presente la distancia mínima calculada.

6. RESULTADOS

Los resultados obtenidos al alinear el algoritmo con diferentes cantidades de usuarios y tipos de normalización se describirán a continuación.

6.1 Resultados usando PCA

6.1.1 *Para 40 usuarios*

En la tabla 5 se resumen los valores de error cuadrático medio y de precisión obtenidos de las gráficas de la figura 25. De color azul se muestran los valores obtenidos al utilizar el clasificador KNN de SK-Learn y de color naranja los valores obtenidos usando el clasificador KNN desarrollado por el autor. Esta nomenclatura se mantendrá en el desarrollo de todo el documento.

Tabla 5. *Valores de Precisión y Error cuadrático medio para fotos sin normalizar usando 40 usuarios*

K = 1 VECINO CERCANO				
	200 EIGEN	53 EIGEN	200 EIGEN	53 EIGEN
ERROR MEDIO	19.075	16.935	29.155	37.65
PRECISIÓN	0.925	0.925	0.87	0.855
K = 2 VECINOS CERCANOS				
	200 EIGEN	53 EIGEN	200 EIGEN	53 EIGEN
ERROR MEDIO	19.075	16.935	67.705	66.44
PRECISIÓN	0.925	0.925	0.73	0.72
K = 3 VECINOS CERCANOS				
	200 EIGEN	53 EIGEN	200 EIGEN	53 EIGEN
ERROR MEDIO	31.3	26.09	71.745	86.815
PRECISIÓN	0.85	0.87	0.705	0.675
K = 4 VECINOS CERCANOS				
	200 EIGEN	53 EIGEN	200 EIGEN	53 EIGEN
ERROR MEDIO	33.435	25.87	80.94	89.295
PRECISIÓN	0.865	0.86	0.665	0.66
K = 5 VECINOS CERCANOS				
	200 EIGEN	53 EIGEN	200 EIGEN	53 EIGEN
ERROR MEDIO	35.315	31.695	85.79	80.905
PRECISIÓN	0.845	0.845	0.62	0.62

6.1.2 *Para 50 usuarios*

En la tabla 6 se resumen los valores de error cuadrático medio y de precisión obtenidos de las gráficas de la figura 26.

Tabla 6. *Valores de Precisión y Error cuadrático medio para fotos sin normalizar usando 50 usuarios*

K = 1 VECINO CERCANO				
	250 EIGEN	22 EIGEN	250 EIGEN	22 EIGEN
ERROR MEDIO	14.684	17.34	39.868	60.224
PRECISIÓN	0.932	0.916	0.844	0.812
K = 2 VECINOS CERCANOS				
	250 EIGEN	22 EIGEN	250 EIGEN	22 EIGEN
ERROR MEDIO	14.684	17.34	77.636	80.732
PRECISIÓN	0.932	0.916	0.656	0.684
K = 3 VECINOS CERCANOS				
	250 EIGEN	22 EIGEN	250 EIGEN	22 EIGEN
ERROR MEDIO	28.896	23.456	91.364	100.46
PRECISIÓN	0.88	0.888	0.672	0.664
K = 4 VECINOS CERCANOS				
	250 EIGEN	22 EIGEN	250 EIGEN	22 EIGEN
ERROR MEDIO	28.192	23.44	98.712	102.32
PRECISIÓN	0.884	0.86	0.608	0.624
K = 5 VECINOS CERCANOS				
	250 EIGEN	22 EIGEN	250 EIGEN	22 EIGEN
ERROR MEDIO	48.732	27.868	98.556	116.524
PRECISIÓN	0.84	0.848	0.592	0.588

Para objeto de análisis, se tomará en cuenta los valores de precisión obtenidos al reducir la dimensionalidad de los datos.

De los resultados obtenidos en la tabla 5 correspondientes a 40 usuarios, se puede indicar que, para fotos sin normalización previa, el algoritmo clasificador KNN implementado por SK-Learn obtuvo un mayor porcentaje de precisión, el cual se encuentra dentro de los objetivos planteados en este trabajo.

Con 50 usuarios los porcentajes de precisión disminuyeron levemente debido al incremento de la varianza para mayor cantidad de rostros de diferentes personas

6.2 Resultados de PCA usando Ecualización del Histograma

6.2.1 *Para 40 usuarios*

Los valores de error cuadrático medio y de precisión obtenidos de la figura 27 se resumen en la tabla 7.

Tabla 7. *Valores de Precisión y Error cuadrático medio para fotos con ecualización del histograma usando 40 usuarios*

K = 1 VECINO CERCANO				
	200 EIGEN	59 EIGEN	200 EIGEN	59 EIGEN
ERROR MEDIO	62.055	80.39	32.94	36.35
PRECISIÓN	0.71	0.625	0.825	0.795
K = 2 VECINOS CERCANOS				
	200 EIGEN	59 EIGEN	200 EIGEN	59 EIGEN
ERROR MEDIO	62.055	80.39	94.745	80.555
PRECISIÓN	0.71	0.625	0.7	0.685
K = 3 VECINOS CERCANOS				
	200 EIGEN	59 EIGEN	200 EIGEN	59 EIGEN
ERROR MEDIO	63.785	74.235	95.68	108.64
PRECISIÓN	0.685	0.615	0.675	0.665
K = 4 VECINOS CERCANOS				
	200 EIGEN	59 EIGEN	200 EIGEN	59 EIGEN
ERROR MEDIO	64.775	65.67	111.63	104.255
PRECISIÓN	0.645	0.615	0.61	0.63
K = 5 VECINOS CERCANOS				
	200 EIGEN	59 EIGEN	200 EIGEN	59 EIGEN
ERROR MEDIO	75.055	79.01	119.22	110.49
PRECISIÓN	0.63	0.595	0.585	0.565

6.2.2 *Para 50 usuarios*

En la tabla 8 se muestran los valores obtenidos de error cuadrático medio y de precisión al aplicar ecualización del histograma a una base de datos de 50 usuarios.

Tabla 8. *Valores de Precisión y Error cuadrático medio para fotos con ecualización del histograma usando 50 usuarios*

K = 1 VECINO CERCANO				
	250 EIGEN	62 EIGEN	250 EIGEN	62 EIGEN
ERROR MEDIO	281.804	301.192	32.236	43.16
PRECISIÓN	0.46	0.404	0.856	0.844
K = 2 VECINOS CERCANOS				
	250 EIGEN	62 EIGEN	250 EIGEN	62 EIGEN
ERROR MEDIO	281.804	301.192	102.568	89.34
PRECISIÓN	0.46	0.404	0.748	0.764
K = 3 VECINOS CERCANOS				
	250 EIGEN	62 EIGEN	250 EIGEN	62 EIGEN
ERROR MEDIO	265.46	312.5	100.864	86.176
PRECISIÓN	0.456	0.388	0.732	0.744
K = 4 VECINOS CERCANOS				
	250 EIGEN	62 EIGEN	250 EIGEN	62 EIGEN
ERROR MEDIO	266.212	295.24	116.64	111.464
PRECISIÓN	0.44	0.392	0.696	0.708
K = 5 VECINOS CERCANOS				
	250 EIGEN	62 EIGEN	250 EIGEN	62 EIGEN
ERROR MEDIO	323.784	79.01	126.58	139.092
PRECISIÓN	0.36	0.595	0.664	0.664

Sin embargo, los porcentajes de precisión obtenidos se consideran bajos para una tarea de reconocimiento facial. Se debe mencionar que los porcentajes alcanzados para 50 usuarios fueron mejores que los conseguidos para 40 usuarios.

6.3 Resultados de PCA aplicando Corrección Gamma

6.3.1 *Para 40 usuarios*

En la tabla 9 se muestran los resultados conseguidos al emplear un factor de corrección gamma igual a 11

Tabla 9. *Valores de Precisión y Error cuadrático medio para fotos con corrección gamma usando 40 usuarios*

K = 1 VECINO CERCANO				
	200 EIGEN	68 EIGEN	200 EIGEN	68 EIGEN
ERROR MEDIO	289.5	276.325	20.43	13.085
PRECISIÓN	0.025	0.045	0.89	0.905
K = 2 VECINOS CERCANOS				
	200 EIGEN	68 EIGEN	200 EIGEN	68 EIGEN
ERROR MEDIO	289.5	276.325	38.89	24.905
PRECISIÓN	0.025	0.045	0.83	0.84
K = 3 VECINOS CERCANOS				
	200 EIGEN	68 EIGEN	200 EIGEN	68 EIGEN
ERROR MEDIO	285.575	454.75	44.77	38.1
PRECISIÓN	0.05	0.055	0.815	0.805
K = 4 VECINOS CERCANOS				
	200 EIGEN	68 EIGEN	200 EIGEN	68 EIGEN
ERROR MEDIO	295.845	388.455	59.275	44.84
PRECISIÓN	0.045	0.06	0.775	0.78
K = 5 VECINOS CERCANOS				
	200 EIGEN	68 EIGEN	200 EIGEN	68 EIGEN
ERROR MEDIO	313.36	408.39	65.405	49.525
PRECISIÓN	0.045	0.06	0.73	0.755

6.3.2 *Para 50 usuarios*

Los valores obtenidos al aplicar un factor de corrección gamma igual a 60 se resumen en la tabla 10.

Tabla 10. *Valores de Precisión y Error cuadrático medio para fotos con corrección gamma usando 50 usuarios*

K = 1 VECINO CERCANO				
	250 EIGEN	71 EIGEN	250 EIGEN	71 EIGEN
ERROR MEDIO	364.5	364.5	43.312	41.844
PRECISIÓN	0.02	0.02	0.88	0.864
K = 2 VECINOS CERCANOS				
	250 EIGEN	71 EIGEN	250 EIGEN	71 EIGEN
ERROR MEDIO	364.5	364.5	57.5	49.756
PRECISIÓN	0.02	0.02	0.792	0.804
K = 3 VECINOS CERCANOS				
	250 EIGEN	71 EIGEN	250 EIGEN	71 EIGEN
ERROR MEDIO	364.5	364.5	88.164	73.876
PRECISIÓN	0.02	0.02	0.796	0.812
K = 4 VECINOS CERCANOS				
	250 EIGEN	71 EIGEN	250 EIGEN	71 EIGEN
ERROR MEDIO	364.5	364.5	124.568	94.804
PRECISIÓN	0.02	0.02	0.708	0.736
K = 5 VECINOS CERCANOS				
	250 EIGEN	71 EIGEN	250 EIGEN	71 EIGEN
ERROR MEDIO	364.5	364.5	120.252	84.948
PRECISIÓN	0.02	0.02	0.688	0.704

Los porcentajes obtenidos para 40 usuarios fueron mejores que los alcanzados con 50 usuarios; además, en ambos casos los porcentajes y error cuadrático medio fueron notablemente mejores al usar el clasificador KNN desarrollado por el autor.

Se debe mencionar que usando 1 vecino cercano con 40 usuarios se alcanzó un porcentaje de 90.5 %

6.4 Resultados usando LDA

6.4.1 *Para 40 usuarios*

En la tabla 11 se muestran los valores obtenidos de error cuadrático medio y de precisión al usar clasificación KNN con 1, 2, 3, 4 y 5 vecinos cercanos.

Tabla 11. *Valores de Precisión y Error cuadrático medio en LDA para fotos sin normalizar usando 40 usuarios*

K = 1 VECINO CERCANO				
	53 EIGEN	38 EIGEN	53 EIGEN	38 EIGEN
ERROR MEDIO	16.935	16.575	37.65	32.92
PRECISIÓN	0.925	0.925	0.855	0.86
K = 2 VECINOS CERCANOS				
	53 EIGEN	38 EIGEN	53 EIGEN	38 EIGEN
ERROR MEDIO	16.935	16.575	66.44	78.185
PRECISIÓN	0.925	0.925	0.72	0.695
K = 3 VECINOS CERCANOS				
	53 EIGEN	38 EIGEN	53 EIGEN	38 EIGEN
ERROR MEDIO	26.09	21.765	86.815	87.98
PRECISIÓN	0.87	0.875	0.675	0.68
K = 4 VECINOS CERCANOS				
	53 EIGEN	38 EIGEN	53 EIGEN	38 EIGEN
ERROR MEDIO	25.87	26.52	89.295	91.185
PRECISIÓN	0.86	0.86	0.66	0.625
K = 5 VECINOS CERCANOS				
	53 EIGEN	38 EIGEN	53 EIGEN	38 EIGEN
ERROR MEDIO	31.695	32.445	80.905	89.71
PRECISIÓN	0.845	0.845	0.62	0.6

6.4.2 *Para 50 usuarios*

En la tabla 12 se muestran los resultados de precisión y error cuadrático medio utilizando LDA en imágenes sin normalizaciones previas.

Tabla 12. *Valores de Precisión y Error cuadrático medio en LDA para fotos sin normalizar usando 50 usuarios*

K = 1 VECINO CERCANO				
	100 EIGEN	21 EIGEN	100 EIGEN	21 EIGEN
ERROR MEDIO	17.292	29.496	45.016	48.092
PRECISIÓN	0.912	0.896	0.832	0.8
K = 2 VECINOS CERCANOS				
	100 EIGEN	21 EIGEN	100 EIGEN	21 EIGEN
ERROR MEDIO	17.292	29.496	83.012	71.06
PRECISIÓN	0.912	0.896	0.664	0.68
K = 3 VECINOS CERCANOS				
	100 EIGEN	21 EIGEN	100 EIGEN	21 EIGEN
ERROR MEDIO	23.836	50.592	85.884	95.756
PRECISIÓN	0.872	0.84	0.684	0.656
K = 4 VECINOS CERCANOS				
	100 EIGEN	21 EIGEN	100 EIGEN	21 EIGEN
ERROR MEDIO	25.632	47.38	93.452	99.58
PRECISIÓN	0.856	0.852	0.636	0.612
K = 5 VECINOS CERCANOS				
	100 EIGEN	21 EIGEN	100 EIGEN	21 EIGEN
ERROR MEDIO	25.596	51.54	103.952	108.392
PRECISIÓN	0.848	0.816	0.592	0.604

Se puede observar que los porcentajes obtenidos con 40 usuarios son superior a lo conseguidos con 50 usuarios; además, los mayores porcentajes se lograron al implementar el clasificador KNN de SK-Learn.

6.5 Resultados usando LDA con Ecualización del Histograma

6.5.1 *Para 40 usuarios*

En la tabla 13 se muestran la precisión y el error cuadrático medio al normalizar las imágenes mediante ecualización del histograma.

Tabla 13. *Valores de Precisión y Error cuadrático medio de LDA para fotos con ecualización del histograma usando 40 usuarios*

K = 1 VECINO CERCANO				
	53 EIGEN	36 EIGEN	53 EIGEN	36 EIGEN
ERROR MEDIO	81.595	103.685	32.605	50.51
PRECISIÓN	0.62	0.56	0.8	0.795
K = 2 VECINOS CERCANOS				
	53 EIGEN	36 EIGEN	53 EIGEN	36 EIGEN
ERROR MEDIO	81.595	103.685	88.755	87.01
PRECISIÓN	0.62	0.56	0.68	0.665
K = 3 VECINOS CERCANOS				
	53 EIGEN	36 EIGEN	53 EIGEN	36 EIGEN
ERROR MEDIO	72.055	94.875	108.735	122.245
PRECISIÓN	0.6	0.555	0.67	0.635
K = 4 VECINOS CERCANOS				
	53 EIGEN	36 EIGEN	53 EIGEN	36 EIGEN
ERROR MEDIO	70.475	94.255	108.84	110.67
PRECISIÓN	0.61	0.55	0.64	0.625
K = 5 VECINOS CERCANOS				
	53 EIGEN	36 EIGEN	53 EIGEN	36 EIGEN
ERROR MEDIO	74.51	91.81	102.725	107.8
PRECISIÓN	0.56	0.535	0.595	0.595

6.5.2 *Para 50 usuarios*

Aumentado el dataset a 50 usuarios, se obtuvieron los resultados mostrados en la tabla 14.

Tabla 14. *Valores de Precisión y Error cuadrático medio de LDA para fotos con ecualización del histograma usando 50 usuarios*

K = 1 VECINO CERCANO				
	100 EIGEN	40 EIGEN	100 EIGEN	40 EIGEN
ERROR MEDIO	330.216	321.456	43.028	55.984
PRECISIÓN	0.436	0.332	0.848	0.812
K = 2 VECINOS CERCANOS				
	100 EIGEN	40 EIGEN	100 EIGEN	40 EIGEN
ERROR MEDIO	330.216	321.456	85.388	84.672
PRECISIÓN	0.436	0.332	0.756	0.76
K = 3 VECINOS CERCANOS				
	100 EIGEN	40 EIGEN	100 EIGEN	40 EIGEN
ERROR MEDIO	317.4	2889.344	104.8	94.236
PRECISIÓN	0.416	0.348	0.728	0.74
K = 4 VECINOS CERCANOS				
	100 EIGEN	40 EIGEN	100 EIGEN	40 EIGEN
ERROR MEDIO	292.532	306.128	107.208	124.52
PRECISIÓN	0.4	0.324	0.708	0.704
K = 5 VECINOS CERCANOS				
	100 EIGEN	40 EIGEN	100 EIGEN	40 EIGEN
ERROR MEDIO	308.388	291.776	129.108	136.944
PRECISIÓN	0.388	0.304	0.656	0.656

Los resultados al aplicar ecualización del histograma en 40 usuarios fueron menores a los resultados de 50 usuarios; en ambos casos se obtuvieron los mejores resultados al implementar el clasificador KNN desarrollado por el autor.

6.6 Resultados usando LDA con Corrección Gamma

6.6.1 *Para 40 usuarios*

En la tabla 15 se detallan los resultados obtenidos al aplicar un factor de corrección gamma igual a 16.

Tabla 15. *Valores de Precisión y Error cuadrático medio en LDA para fotos con corrección gamma usando 40 usuarios*

K = 1 VECINO CERCANO				
	53 EIGEN	34 EIGEN	53 EIGEN	34 EIGEN
ERROR MEDIO	289.5	289.5	20.715	23.66
PRECISIÓN	0.025	0.025	0.9	0.86
K = 2 VECINOS CERCANOS				
	53 EIGEN	34 EIGEN	53 EIGEN	34 EIGEN
ERROR MEDIO	289.5	289.5	35.355	27.9
PRECISIÓN	0.025	0.025	0.825	0.82
K = 3 VECINOS CERCANOS				
	53 EIGEN	34 EIGEN	53 EIGEN	34 EIGEN
ERROR MEDIO	261.9	300.6	28.05	33.89
PRECISIÓN	0.045	0.025	0.835	0.815
K = 4 VECINOS CERCANOS				
	53 EIGEN	34 EIGEN	53 EIGEN	34 EIGEN
ERROR MEDIO	307.2	308.7	46.785	55.935
PRECISIÓN	0.03	0.03	0.755	0.76
K = 5 VECINOS CERCANOS				
	53 EIGEN	34 EIGEN	53 EIGEN	34 EIGEN
ERROR MEDIO	307.8	293.1	50.9	50.835
PRECISIÓN	0.025	0.025	0.75	0.76

6.6.2 *Para 50 usuarios*

Al incrementar el dataset y usando un factor de corrección gamma igual a 60 se obtuvieron los resultados descritos en la tabla 16.

Tabla 16. *Valores de Precisión y Error cuadrático medio en LDA para fotos con corrección gamma usando 50 usuarios*

K = 1 VECINO CERCANO				
	100 EIGEN	64 EIGEN	100 EIGEN	64 EIGEN
ERROR MEDIO	364.5	364.5	33.768	30.848
PRECISIÓN	0.02	0.02	0.86	0.86
K = 2 VECINOS CERCANOS				
	100 EIGEN	64 EIGEN	100 EIGEN	64 EIGEN
ERROR MEDIO	364.5	364.5	52.12	44.728
PRECISIÓN	0.02	0.02	0.804	0.8
K = 3 VECINOS CERCANOS				
	100 EIGEN	64 EIGEN	100 EIGEN	64 EIGEN
ERROR MEDIO	364.5	364.5	82.064	67.812
PRECISIÓN	0.02	0.02	0.8	0.776
K = 4 VECINOS CERCANOS				
	100 EIGEN	64 EIGEN	100 EIGEN	64 EIGEN
ERROR MEDIO	364.5	364.5	91.288	86.068
PRECISIÓN	0.02	0.02	0.732	0.704
K = 5 VECINOS CERCANOS				
	100 EIGEN	64 EIGEN	100 EIGEN	64 EIGEN
ERROR MEDIO	364.5	364.5	94.708	84.76
PRECISIÓN	0.02	0.02	0.696	0.688

Al aplicar dicho factor de corrección, los porcentajes de precisión alcanzados con 40 usuarios fueron ligeramente superiores a los logrados con 50 usuarios; además, existe una diferencia notable en cuanto a error cuadrático medio, por lo que la normalización aplicada a 40 usuarios se eligió como la más óptima. En ambos casos los mejores resultados se consiguieron aplicando el clasificador KNN desarrollado por el autor.

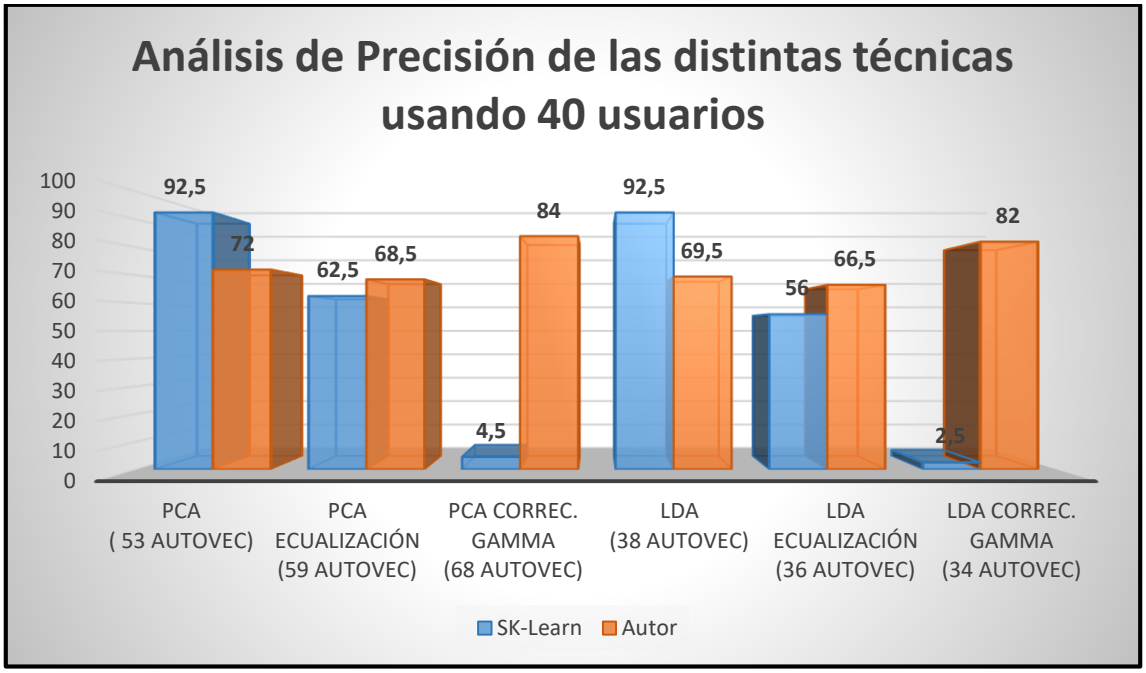
Para apreciar de mejor manera los resultados obtenidos en las tablas anteriores, se presenta, en la tabla 18, un resumen de los errores cuadráticos medios y de la precisión conseguida en cada normalización, tomando en cuenta que los resultados más óptimos se alcanzaron utilizando dos vecinos cercanos en la etapa de clasificación.

Tabla 17. *Resumen de Resultados obtenidos al aplicar diferentes normalizaciones*

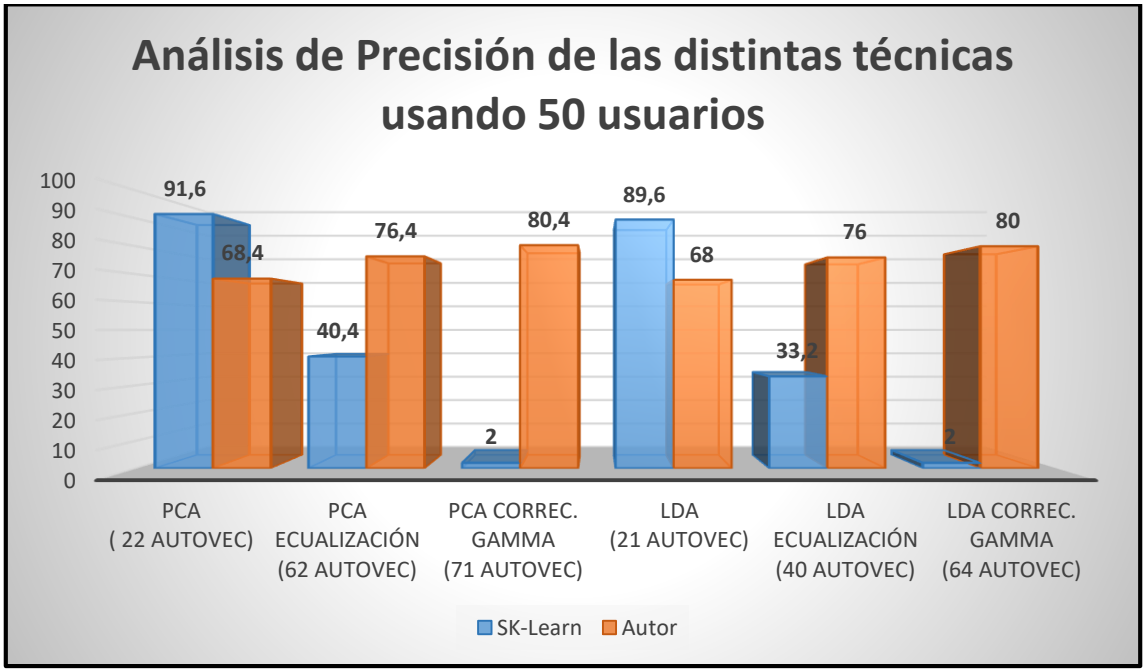
		Error Cuadrático Medio	Precisión
PCA	40 Usuarios, 53 autovec.	19.075	92.5 %
	50 Usuarios, 22 autovec.	17.34	91.6%
PCA con Ecualización	40 Usuarios, 59 autovec.	80.555	68.5 %
	50 Usuarios, 62 autovec.	89.34	76.4 %
PCA con Correc. Gamma	40 Usuarios, 68 autovec.	24.905	84 %, γ = 11
	50 Usuarios, 71 autovec.	49.756	80.4 %, γ = 60
LDA	40 Usuarios, 38 autovec.	16.575	92.5 %
	50 Usuarios, 21 autovec.	24.496	89.6 %
LDA con Ecualización	40 Usuarios, 36 autovec.	87.01	66.5 %
	50 Usuarios, 40 autovec.	84.672	76 %
LDA con Correc. Gamma	40 Usuarios, 34 autovec.	27.9	82 %, γ = 16
	50 Usuarios, 64 autovec.	44.728	80 %, γ = 60

Todos los valores mostrados en la tabla 17 corresponden a los mejores porcentajes de precisión y de error cuadrático medio obtenidos en cada etapa, para 40 y 50 usuarios; además, el color naranja corresponde a los casos en los que el algoritmo clasificador KNN desarrollado por el autor fue mejor que el clasificador implementado por SK-Learn.

En la figura 33 a) y b) se muestran los mejores porcentajes de precisión obtenidos al usar dos vecinos cercanos en el proceso de clasificación de las distintas técnicas analizadas para 40 y 50 usuarios respectivamente.



a)



b)

Figura 34. *Precisión de las técnicas estudiadas para a) 40 usuarios y b) 50 usuarios*

Fuente: *Autor*

7. DISCUSIÓN

7.1 Algoritmo PCA

Al aplicar previamente técnicas de normalización a las imágenes de la base de datos, se pudo evidenciar que el clasificador desarrollado por el autor produce mejores resultados frente al clasificador de SK-Learn.

No obstante, de entre todos los casos estudiados, el mayor porcentaje de precisión (92.5 %) se alcanzó al utilizar el clasificador KNN de SK-Learn sobre imágenes sin normalización previa.

En adición, en todas las variantes propuestas, la base de datos de 40 usuarios pertenecientes solamente a ORL logró resultados superiores frente a la base de datos combinada entre ORL y MIT.

7.2 Algoritmo LDA

De la misma manera, al aplicar técnicas de normalización sobre las imágenes, el algoritmo clasificador de SK-Learn reduce notablemente sus porcentajes de precisión, llegando hasta niveles inaceptables de 0,2 % con corrección gamma.

Así mismo, se alcanzó un porcentaje de precisión máximo de 92.5 % al entrenar imágenes sin normalización. Los mejores resultados se lograron con la base de datos de ORL (40 usuarios).

7.3 Comparación entre PCA y LDA

Contrastando los resultados adquiridos con 40 usuarios en PCA y LDA para fotos sin normalizar, se debe señalar que usando la técnica LDA se alcanzó un porcentaje de precisión igual a PCA; sin embargo, el método LDA permite una reducción de dimensionalidad considerablemente mayor (solamente 38 autovectores) y un error cuadrático medio menor.

Dicha reducción de dimensionalidad se traduce en un coste computacional y tiempo de ejecución notablemente bajo, aproximadamente 1,7 segundos, lo que permitiría ejecutar este algoritmo desde microcomputadores o chips que no posean características de procesamiento elevadas.

8. CONCLUSIONES

- El modelo de reconocimiento de rostros más eficiente fue Fisherfaces, alcanzando una precisión de 92.5% mediante Análisis de Discriminantes Lineales usando el clasificador KNN de la librería SK-Learn con dos vecinos cercanos, y la base de datos ORL sin previa normalización para la etapa de entrenamiento.
- El algoritmo clasificador desarrollado por el autor alcanzó su mayor porcentaje de precisión de 84% utilizando el método Eigenfaces para fotografías normalizadas con corrección gamma; mientras que el algoritmo clasificador de la librería SK-Learn de Python alcanzó su mayor porcentaje de precisión de 62.5% usando el método Eigenfaces para fotografías normalizadas con ecualización del histograma, en ambos casos los resultados se obtuvieron usando 40 usuarios.
- El algoritmo clasificador desarrollado por SK-Learn alcanza un porcentaje de precisión de 4.5 % en el mejor caso correspondiente a normalización con factor de corrección gamma igual a 11; esto debido a que SK-Learn realiza pre-procesamiento sobre las imágenes incluyendo normalización; es decir, las imágenes se normalizan nuevamente lo cual produce una pérdida elevada de datos correspondientes a características faciales.
- A partir de la etapa de entrenamiento y evaluación de los distintos modelos, se concluye que el algoritmo PCA logra resultados similares al algoritmo LDA usando el clasificador KNN de SK-Learn; en ambos casos se alcanza una precisión de 92.5% para imágenes sin normalizar; sin embargo, LDA permite una mayor reducción de dimensionalidad, reduciendo los 53 autovectores obtenidos en PCA a 38 autovectores; por lo tanto, utilizando LDA el costo computacional y tiempo de ejecución se reducen al momento de clasificar y reconocer un rostro, lo cual permite implementar este algoritmo en cualquier micro-computadora.
- En cada caso analizado se alcanzaron porcentajes de precisión mayores utilizando 1 vecino cercano en la etapa de clasificación; no obstante, se decidió utilizar dos vecinos cercanos con el propósito de brindar una confiabilidad mayor al algoritmo.

- La técnica de corrección gamma mostró mejores resultados frente a la técnica de ecualización del histograma utilizando el algoritmo clasificador desarrollado por el autor, llegando a porcentajes cercanos al 90% para Eigenfaces y Fisherfaces.
- A partir de las pruebas realizadas con la base de datos creada por el autor, se concluyó que, para lograr una detección exitosa, tanto el ambiente en el que se crea la base de datos como en el que se reconoce al usuario deben poseer características de iluminación similares
- Ejecutándose desde la interfaz gráfica, el tiempo que el sistema ocupa para entrenar adecuadamente el modelo cuando se ingresa un nuevo usuario es 3.3 segundos; y el tiempo que ocupa para reconocer a un usuario es de 1.7 segundos, utilizando un 15% de la memoria RAM y aproximadamente 38% del procesamiento del CPU; estos datos fueron registrados en una computadora con un procesador Intel Core i7 de 2.4 GHz de capacidad de procesamiento, memoria RAM de 8Gb, disco duro SATA de 1Tb HDD, y una tarjeta gráfica AMD Radeon R7 M270 con memoria DDR de 4 gigabytes.

9. RECOMENDACIONES

Las aplicaciones y los beneficios que pueden alcanzarse al utilizar sistemas de reconocimiento facial son numerosos; sin embargo, se debe tener precaución al momento de usar los mismos ya que el código de acceso es una fotografía de nuestro rostro. Es decir, estamos brindando a un tercero la posibilidad de almacenar datos únicos de nuestra persona.

En la actualidad existen diversas bases de datos que pueden ser descargadas gratuitamente desde internet, las cuales se han creado con fines investigativos y para entrenamiento de sistemas de reconocimiento facial; no obstante, dada la facilidad de acceso a esta información, diferentes grupos políticos, militares o empresas pueden utilizar estos datos con fines maliciosos.

Un gran ejemplo de esto es lo sucedido hace unos meses con la base de datos *MS Celeb* de *Microsoft*, esta empresa decidió eliminar completamente su base de datos de rostros, la cual se consideraba como una de las más extensas a nivel mundial. Según el *Financial Times*, los datos de *MS Celeb* han sido usados por compañías como *NVIDIA, Hitachi,IBM, Alibaba, Panasonic*, *Megvii*, *Sensetime* y organizaciones militares. (Sabán Antonio, 2019). El problema empieza cuando se descubrió que en dicha base de datos existían rostros de investigadores de seguridad y otras personas que no habían dado autorización para que sus datos sean almacenados en dicho conjunto de imágenes.

En adición, se debe mencionar el riesgo que existe al usar aplicaciones gratuitas que requieren del rostro de cada persona para aplicar filtros; cuando una persona acepta los términos y condiciones de estas aplicaciones permite a los creadores almacenar y manipular los datos de su rostro de la forma que ellos crean conveniente. Un ejemplo de esto es la aplicación *FaceApp*, la cual ha sido utilizada recientemente por un gran número de personas y que expresa en sus Términos de servicio que toda la información podrá ser cedida a empresas colaboradoras, del mismo grupo al que pertenece la firma desarrolladora y a terceros que se conviertan en afiliadas suyas.

Se recomienda analizar pacientemente respecto a qué personas estamos entregando nuestra información personal y que condiciones de uso estamos aceptando al momento de utilizar aplicaciones gratuitas, especialmente en nuestro medio ya que hasta la

actualidad no existe una legislación adecuada que sancione a compañías o terceros que utilicen nuestros datos mal intencionadamente.

10. BIBLIOGRAFÍA

About Python™ | Python.org. (2019). Retrieved October 21, 2019, from https://www.python.org/about/

Ahonen, T., Hadid, A., & Pietikä, M. (2006). *Face Description with Local Binary Patterns: Application to Face Recognition*. Retrieved from http://www.ee.oulu.fi/research/imag/texture/lbp/bibliography/

Aranguren Zapata, A., & Vela Asin, T. (2012). *Sistema de seguimiento de objetos mediante procesamiento digital de imágenes aplicado al control de robots autónomos*. Universidad Peruana de Ciencias Aplicadas. Retrieved from https://repositorioacademico.upc.edu.pe/bitstream/handle/10757/305116/aranguren_za-pub-delfos.pdf?sequence=1&isAllowed=y

Armengot, M. (2006). *Análisis comparativo de métodos basados en subespacios aplicados al reconocimiento de caras*. Valencia. Retrieved from https://www.researchgate.net/publication/237270192

AT&T Laboratories Cambridge. (2002). The ORL Database of Faces. Retrieved June 28, 2019, from https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Atienza, V. (2011). *El histograma de una imagen digital*. Valencia. Retrieved from https://riunet.upv.es/bitstream/handle/10251/12711/El%2520histograma%2520una%2520imagen%2520digital.pdf?sequence=1

Bhandari, A. K., Kumar, A., & Singh, G. K. (2015). Improved knee transfer function and gamma correction based method for contrast and brightness enhancement of satellite image. *AEU - International Journal of Electronics and Communications*, *69*(2), 579–589. https://doi.org/10.1016/J.AEUE.2014.11.012

Caballero Barriga, E. R. (2017). *Aplicación práctica de la visión artificial para el reconocimiento de rostros en una imagen, utilizando redes neuronales y algoritmos de reconocimiento de objetos de la biblioteca OpenCV*. Retrieved from http://repository.udistrital.edu.co/bitstream/11349/6104/1/CaballeroBarrigaEdisonRene2017.pdf

Características de Java – Manual de Java. (2019). Retrieved October 22, 2019, from https://www.manual-java.com/manualdejava/caracteristicas-de-java/

Cazorla Martínez, R. (2016). *Software para la Detección y el Reconocimiento de Rostros*. Retrieved from https://ddd.uab.cat/pub/tfg/2016/tfg_49339/Software_para_la_deteccio_n_y_el_reconocimiento_de_caras.pdf

Chihaoui, M., Elkefi, A., Bellil, W., & Ben Amar, C. (2016). A Survey of 2D Face Recognition Techniques. *Computers*, *5*(4), 21. https://doi.org/10.3390/computers5040021

Dalal, N., Histograms, B. T., & Triggs, B. (2009). Histograms of Oriented Gradients for Human Detection, 886–893. https://doi.org/10.1109/CVPR.2005.177ï

EcuRed. (2010). JavaScript. Retrieved October 22, 2019, from https://www.ecured.cu/JavaScript

Eguíluz Pérez, J. (n.d.). *Introducción a JavaScript*. Retrieved from www.librosweb.es

Eslava Ríos, J. (2013). *Reconocimiento Facial en tiempo real*. Universidad Autónoma de Madrid. Retrieved from http://repositorio.uam.es/handle/10486/13893

Esqueda Elizondo, J. J. (2005). *Fundamentos de Procesamiento de Imágenes*. Retrieved from https://s3.amazonaws.com/academia.edu.documents/39190415/FundamentosDeProcesamientoDeImagenes.pdf?response-content-disposition=inline%3B filename%3DFundamentos_de_procesamiento_de_imagenes.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYY

Fernández Montoro, A. (2012). *Python 3 al descubierto*. Retrieved from www.rclibros.es

GitHub: The top 10 programming languages for machine learning. (2019). Retrieved October 21, 2019, from https://www.techrepublic.com/article/github-the-top-10-programming-languages-for-machine-learning/?fbclid=IwAR0YeK_Cra7Kby1qBsWIB52eeKfsOb8twD6Y72tMBNEcYa-I9jm5SsXqab8

Gonzalez, R., & Woods, R. (2017). *Digital Image Processing*. (Prentince Hall, Ed.) (Second). New Jersey. Retrieved from http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/Digital_Image_Processing_2ndEd.pdf

Hatem, H., Beiji, Z., & Majeed, R. (2015). A Survey of Feature Base Methods for Human Face Detection. *International Journal of Control and Automation*, *8*(5), 61–78. https://doi.org/10.14257/ijca.2015.8.5.07

Hirvin Gonzalez, S. V. (2019). *Reconocimiento Facial utilizando Viola-Jones y Patrones Binarios*. Venezuela. Retrieved from file:///C:/Users/SBryan/Downloads/126-Artículo-287-2-10-20190912.pdf

Huamán Layme, J. A., & Del Carpio Salinas, J. A. (2007). *Localización, seguimiento y reconocimiento de rostros empleando métodos estadísticos PCA y HMME*. *Universidad Nacional de Ingeniería*. Universidad Nacional de Ingeniería. Retrieved from http://repositorio.uni.edu.pe/handle/uni/1003

Jiménez Ochoa, M. G. (2016). Desarrollo de un sistema de visión artificial para la detección de aglomeración de personas en un semáforo. Retrieved from https://dspace.unl.edu.ec/jspui/handle/123456789/11225

Llinás Solano, H., & Rojas Álvarez, C. (2017). *Estadística descriptiva y distribuciones de probabilidad*. (U. del Norte, Ed.). Barranquilla.

Massachusetts Institute of Technology. (2005). MIT-CBCL Face Recognition Database. Retrieved August 6, 2019, from http://cbcl.mit.edu/software-datasets/heisele/facerecognition-database.html

Massó Gomez, W. (2012). Utilizando Bibliotecas de Código C en Aplicaciones C#. *Revista Telem@tica*, *11*(2), 36–45. Retrieved from http://revistatelematica.cujae.edu.cu/index.php/tele

Menéndez, R. (n.d.). *JavaScript*. Retrieved from https://www.um.es/docencia/barzana/DAWEB/Lenguaje-de-programacion-JavaScript-1.pdf

Mu, Y., Yan, S., Liu, Y., Huang, T., & Zhou, B. (n.d.). *Discriminative Local Binary Patterns for Human Detection in Personal Album*. Retrieved from http://cv.cs.nthu.edu.tw/upload/activities/25/32.pdf

Narayana, K. V. (2015). *Enhanced Face Recognition based on PCA and SVM*. *International Journal of Computer Applications* (Vol. 117). Retrieved from www.ijcaonline.org

October Headline: Top 8 of the TIOBE index quite stable for the last 15 years. (2019). Retrieved October 21, 2019, from https://www.tiobe.com/tiobe-index/

Panda, R., & Naik, M. K. (2015). A novel adaptive crossover bacterial foraging optimization algorithm for linear discriminant analysis based face recognition. *Applied Soft Computing*, *30*, 722–736. https://doi.org/10.1016/J.ASOC.2015.02.021

Peláez Fernández, B., & Ramón Morros, J. (2012). *Reconocimiento de caras en entornos no controlados*. Cataluña. Retrieved from https://upcommons.upc.edu/bitstream/handle/2099.1/15642/Reconocimiento_de_caras_en_entornos_no_controlados,_Borja_Pelaez_Fernandez.pdf

Peña, M., & Orellana, J. (2018). *Red neuronal para clasificación de riesgo en cooperativas de ahorro y crédito*. Cuenca. https://doi.org/10.24133/cctespe.v13i1.710

Pillajo Coka, J. A., & Yaguana Montero, J. E. (2018). *Implementación de un Sistema de Soporte en Línea con Reconocimiento Facial como Mecanismo de Seguridad para Representantes de PFIZER*. Universidad de las Fuerzas Armadas. Retrieved from http://repositorio.espe.edu.ec/jspui/bitstream/21000/14161/1/T-ESPE-057692.pdf

Principales Características de Java. (2019). Retrieved October 22, 2019, from http://personales.upv.es/rmartin/cursoJava/Java/Introduccion/PrincipalesCaracteristicas.htm

Rodríguez, A. (2019, July 2). 5 zonas de Quito tendrán cámaras con reconocimiento facial. Retrieved from https://www.ecuavisa.com/articulo/noticias/nacional/505121-5-zonas-quito-tendran-camaras-reconocimiento-facial

Russ, J., & Brent Neal, F. (2017). *The Image Processing*. (T. & F. Group, Ed.) (Séptima). Carolina del Norte: Taylor & Francis Group.

Sabán Antonio. (2019). Microsoft elimina de Internet una de las mayores bases de datos públicas de caras, en pleno debate sobre el reconocimiento facial. Retrieved August 18, 2019, from http://www.diariotecnologia.es/microsoft-elimina-de-internet-una-de-las-mayores-bases-de-datos-pblicas-de-caras-en-pleno-debate-sobre-el-reconocimiento-facial

Sánchez, P. (2006). *Métodos Estadísticos Aplicados* (Primera). Barcelona.

Serratosa, F. (2012). *La biometría para la identificación de las personas*. Cataluña. Retrieved from https://s3.amazonaws.com/academia.edu.documents/51873502/Biometria_ES_Modulo_1.pdf?response-content-disposition=inline%3B filename%3DLa_biometria_para_la_identificacion_de_l.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYYGZ2Y53UL3A%2F20

Slavković, M., & Jevtić, D. (2012). Face Recognition Using Eigenface Approach*. *SERBIAN JOURNAL OF ELECTRICAL ENGINEERING*, *9*(1), 121–130. https://doi.org/10.2298/SJEE1201121S

Smiatacz, M. (2013). Eigenfaces, Fisherfaces, Laplacianfaces, Marginfaces-How to Face the Face Verification Task. https://doi.org/10.1007/978-3-319-00969-8_18

The Most Popular Language For Machine Learning Is. (2016). Retrieved October 21, 2019, from https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Language_Is_Best_For_Machine_Learning_And_Data_Science?lang=en&fbclid=IwAR2fpZVq1gcyX3tGMmqzXq1VC3z0-01rJdjE3wtP7I9inFPTczgB8ltUaLU

The State of the Octoverse: machine learning. (2019). Retrieved October 21, 2019, from https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/?fbclid=IwAR2nSujDfMY0Yl5HxZxeJWJVaqqh6-r1jrmyMJMMKZiBK0mkHakywMSIphA

The Top Programming Languages 2019 - IEEE Spectrum. (2019). Retrieved October 21, 2019, from https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019

Top Javascript Machine Learning libraries in 2019 - Towards Data Science. (2019). Retrieved October 22, 2019, from https://towardsdatascience.com/top-javascript-machine-learning-libraries-in-2019-cb63b95bdd10

Top programming languages in 2019: Python sweeps the board. (2019). Retrieved October 21, 2019, from https://jaxenter.com/python-ranking-2019-161880.html

Train, K. (2014). *MÉTODOS DE ELECCIÓN DISCRETA CON SIMULACIÓN* (Segunda Edición). Retrieved from https://eml.berkeley.edu/books/choice2nd/C14.pdf

Universidad de Granada. (2019). Python y sus características principales. Retrieved October 21, 2019, from https://osl.ugr.es/2018/12/14/python-y-sus-caracteristicas-principales/

Van Dung, H., Vavilin, A., Jo, K.-H., Hoang, V.-D., & Vavilin Kang-Hyun Jo Ulsan, A. (2012). Fast Human Detection based on Parallelogram Haar-like features. https://doi.org/10.1109/IECON.2012.6389212

Viola, P., & Jones, M. (2002). *Fast and Robust Classification using Asymmetric AdaBoost and a Detector Cascade*. Retrieved from http://papers.nips.cc/paper/2091-fast-and-robust-classification-using-asymmetric-adaboost-and-a-detector-cascade.pdf

Yang, S., Luo, P., Loy, C. C., & Tang, X. (2016). *WIDER FACE: A Face Detection Benchmark*. Hong Kong. Retrieved from http://openaccess.thecvf.com/content_cvpr_2016/papers/Yang_WIDER_FACE_A_CVPR_2016_paper.pdf

Yilber Sisco, F. M. (2017). *Minería de datos para la detección de rostros mediante el uso de Maquinas de Soporte Vectorial, Redes Neuronales Artificiales y Redes Neuronales Convolucionales*. Caracas. Retrieved from https://www.researchgate.net/publication/323666740_Mineria_de_datos_para_la_deteccion_de_rostros_mediante_el_uso_de_Maquinas_de_Soporte_Vectorial_Redes_Neuronales_Artificiales_y_Redes_Neuronales_Convolucionales

Zhang, C., & Zhang, Z. (2010). *A Survey of Recent Advances in Face Detection*. Retrieved from http://www.research.microsoft.com

11. ANEXOS

ANEXO 1: MANUAL DE USUARIO DEL SISTEMA DE RECONOCIMIENTO FACIAL

- Pantalla Inicial del Sistema

Una vez ejecutada la interfaz gráfica se mostrará la ventana de la figura 34, en la cual se tienen dos opciones principales: Base de Datos y Acceder al sistema

Figura 35. *Pantalla inicial del Sistema de Reconocimiento Facial*

- Gestión de la base de datos

Para conocer la información existente en la base de datos se debe hacer click en el botón **Base de Datos** e inmediatamente aparecerá la ventana de validación de la figura 35 que solicita la clave de administrador; la clave es **Cieyt2019.**

Figura 36. *Ventana de verificación de contraseña*

Luego de escribir la clave, se debe clicar en el botón **Acceder**. En caso de ingresar una clave errónea se presentará el mensaje de error de la figura 36.

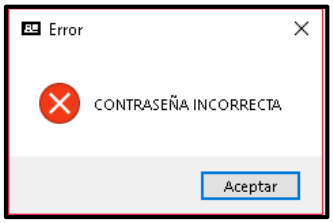


Figura 37. *Ventana emergente de error por contraseña incorrecta*

Si el usuario desea regresar a la pantalla principal debe hacer click en el botón **Atrás** de la ventana de verificación.

Cuando el usuario ingresa la clave correcta se presenta en pantalla una lista con los usuarios almacenados en la base de datos como se muestra en la figura 37.

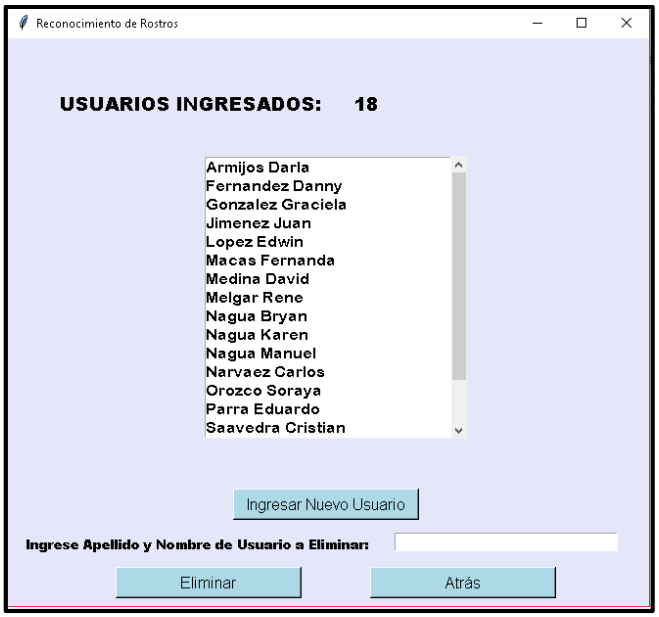


Figura 38. *Ventana de Gestión de la Base de Datos*

Además, existen dos opciones para modificar la base de datos: Ingresar un usuario nuevo y Eliminar un usuario existente.

- Ingresar un nuevo usuario

Se debe hacer click en el botón **Ingresar Nuevo Usuario** y a continuación se mostrará una pantalla con una secuencia de video obtenida por la videocámara en la cual se grafica un cuadro delimitador azul alrededor del rostro del usuario. La persona que desee ingresar debe ingresar su nombre en el siguiente formato: **Apellido Nombre,** por ejemplo: **Nagua Bryan;** posteriormente **debe colocarse en posición recta** a aproximadamente

40cm del lente de la cámara y hacer click en el botón **Tomar Foto.** En la figura 38 se presenta una detección correcta del rostro.

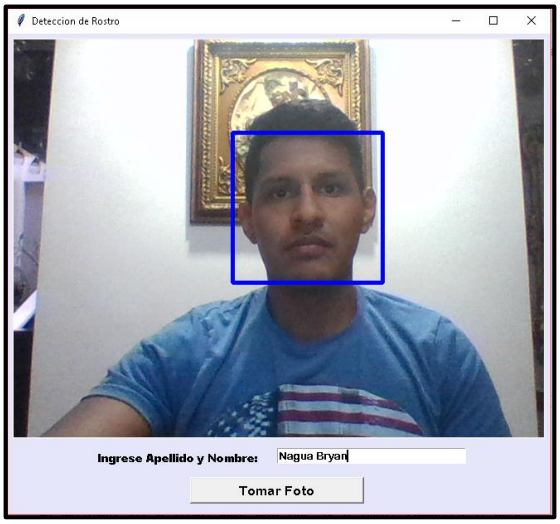


Figura 39. *Ventana de Ingreso de nuevo usuario*

Es importante que el usuario haga click en **Tomar Foto** una vez el cuadro delimitador se haya estabilizado; es decir, que solamente aparezca un cuadro delimitador. En la figura 39 se muestra un escenario no apropiado para almacenar las imágenes.

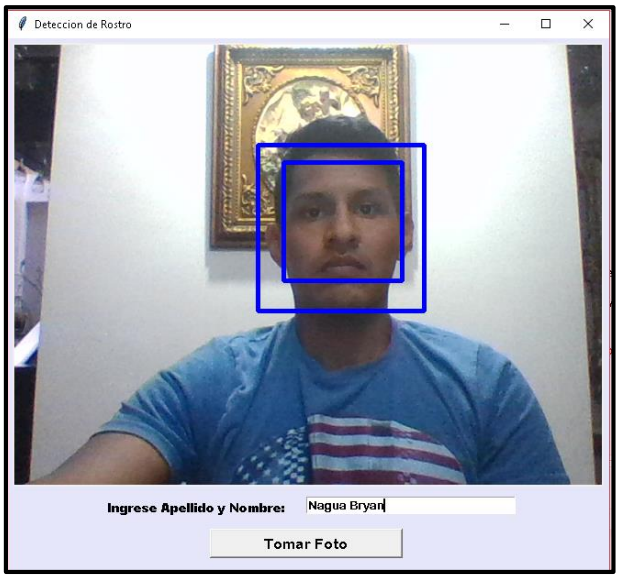


Figura 40. *Detección incorrecta del rostro en Ventana de Ingreso de nuevo usuario*

- Eliminar usuario existente

Para eliminar un usuario existente basta con ingresar en el cuadro **Ingrese Nombre y Apellido de Usuario a Eliminar** la identificación de un usuario mostrado en la lista con

el siguiente formato: **Apellido Nombre** y hacer click en el botón eliminar**.** Cabe recalcar que se debe escribir el nombre del usuario de la misma forma en la que se muestra en la lista, como se observa en la figura 40.

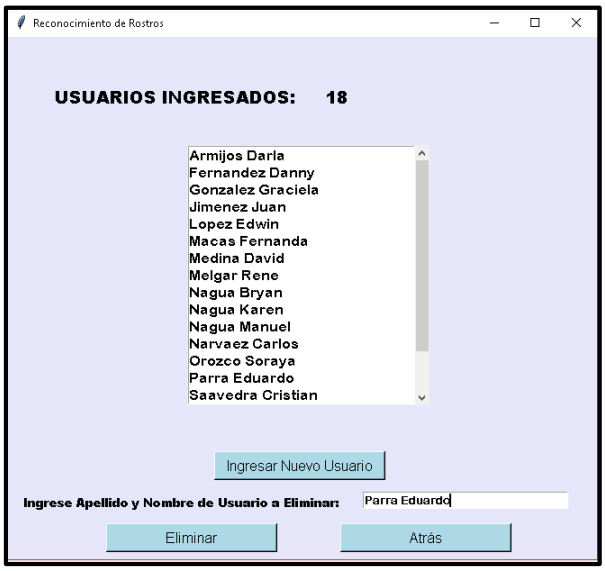


Figura 41. *Ventana Eliminar Usuario existente*

Si se escribió correctamente el nombre del usuario a eliminar se mostrará en pantalla el mensaje de confirmación de la figura 41.

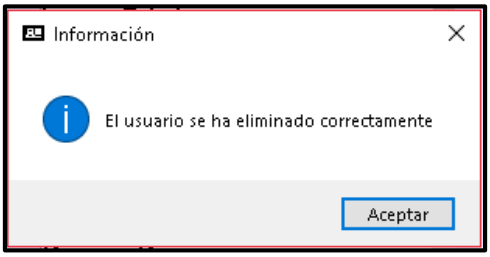


Figura 42. *Ventana emergente de información para usuario eliminado adecuadamente*

Por el contrario, si se escribió erróneamente el nombre de un usuario o se escribió el nombre de un usuario que no existe en la base de datos se presentará el mensaje de error de la figura 42.

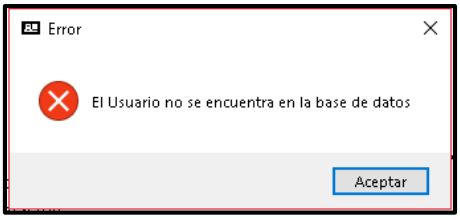


Figura 43. *Ventana emergente de error para eliminación incorrecta de usuario*

- Reconocimiento de un usuario

Para acceder al sistema de reconocimiento se debe hacer click en el botón **Acceder al Sistema** ubicado en la pantalla principal. Luego de hacer click se mostrará una pantalla similar a la mostrada en la ventana de Ingresar Nuevo Usuario; se debe seguir las mismas recomendaciones y hacer click en **Tomar Foto;** si se desea regresar a la pantalla inicial se debe hacer click en el botón **Atrás.** La pantalla de reconocimiento se observa en la figura 43.

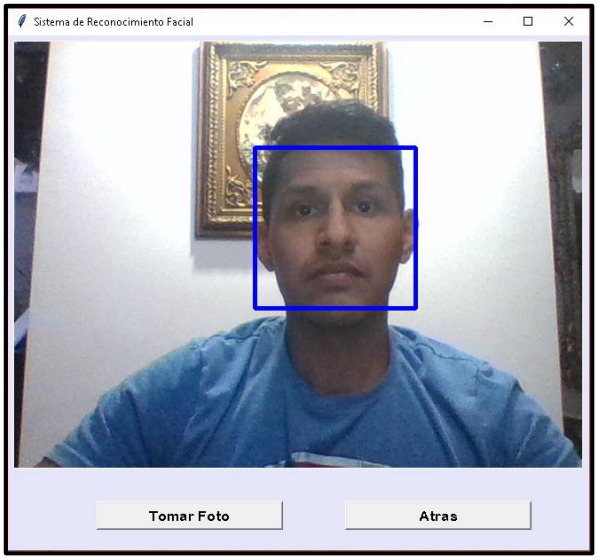


Figura 44. *Ventana de Reconocimiento de usuario*

Luego de tomar la foto existen dos posibilidades: Si el usuario que desea acceder al sistema se encuentra en la base de datos se mostrará en pantalla su fotografía y nombre como se muestra en la figura 44.

Figura 45. *Ventana con datos de usuario reconocido correctamente*

Al contrario, si el usuario que desea acceder al sistema **NO** se encuentra almacenado en la base de datos se mostrará la foto de la persona que intenta ingresar seguido de un mensaje **USUARIO DESCONOCIDO,** como se muestra en la figura 45.

Figura 46. *Ventana de Usuario desconocido*

El usuario deberá hacer clicar en el botón **Atrás** con lo que retornará a la pantalla de inicio del sistema.

ANEXO 2: MANUAL DEL PROGRAMADOR

ALGORITMO DE CLASIFICACIÓN KNN

```
################## CÁLCULO DE LA DISTANCIA EUCLIDEANA EXISTENTE ENTRE CADA IMAGEN ###################

def dist_eucl(train):        #Se crea una función que recibe el conjunto
                             #PCA de entrenamiento como parámetro de entrada
    a,b = train.shape        #Se almacena la dimensión de ese conjunto
    mat_dist = []
    dist_total = []
    diferencia = []
    for c in range(b):       #Se itera una cantidad de veces igual al numero de columnas
        for d in range (b):  #Se itera una cantidad de veces igual al número de columnas
            diferencia = [pow((train[:,c]-train[:,d]),2)] #Se calcula la diferencia de la columna
                                                          #con el resto de columnas
            suma = np.sum(diferencia, axis=1)   #Se suma todos los valores de diferencia
                                                #acorde a la ecuación de distancia euclideana.
            distancia = math.sqrt(suma)      #Se calcula la raiz cuadrada
            mat_dist.append(distancia)       #El resultado se almacena en una matriz que contiene
                                             #la distancia de la columna con respecto a las demás.
    dist_total.append(mat_dist)              #Esta matriz contiene la distancia de cada columna
                                             #con respecto a las demás.
    dist_total = np.asarray(dist_total)      #Se convierte a arreglo numpy
    return dist_total                        #Se retorna el arreglo final de distancias
```

```
#################### CÁLCULO DE LAS MENORES DISTANCIAS ENTRE CADA IMAGEN ##########################

def obt_vecinos(distancias, etiquetas, k):     #Función que recibe las distancias, etiquetas y número
                                               #de vecinos cercanos que se desea calcular
    o,p = distancias.shape                     #Se almacena la dimensión del conjunto de distancias.
    paso = int(math.sqrt(p))                   #Nueva cantidad de etiquetas
    mat_ord = []
    vecinos = []
    tuplas2 = []
    n_etiquetas = []
    for m in range(paso):
        n_etiquetas.extend(etiquetas)          #Se añade una matriz de etiquetas de usuario
                                               #a cada conjunto de distancias
    n_etiquetas = np.asarray(n_etiquetas)      #Se convierte a arreglo numpy
    tuplas2 = [(distancias[0][i], n_etiquetas[i]) for i in range(p)]     #Se empareja cada distancia
                                                                         #con su respectiva etiqueta de usuario
    a = len(tuplas2)                       #Se almacena la longitud del arreglo anterior.
    for x in range(0,a,paso):
        mat_paso = tuplas2[x:x+paso]       #Se agrupa adecuadamente los elementos de la matriz de tuplas
                                           #en subconjuntos que contienen las distancias entre cada columna
        mat_ord.append(mat_paso)           #Se añade cada subgrupo a una matriz principal.
    for c in mat_ord:
        c.sort()                           #Se ordena las tuplas de cada subgrupo (c) ascendentemente
    for d in mat_ord:
        a = d[:k+1]                        #Se selecciona las k+1 tuplas con distancia menor.
        a = np.asarray(a)                  #Se convierte a arreglo numpy
        a = a[:,1]                         #Se almacena solamente el número de la etiqueta
                                           #correspondiente a las k+1 distancias cercanas, descartando el
                                           #valor de la distancia.
        vecinos.append(a)                  #Se añade los vecinos mas cercanos de cada columna a una matriz.
    vecinos = np.asarray(vecinos)          #Se convierte a arreglo de numpy
    vecinos = np.delete(vecinos,(0), axis=1)  #Se elimina la primera columna de etiquetas por ser
                                              #la distancia existente al mismo punto (la distancia es 0)
    return vecinos,k                       #Se retorna el arreglo de vecinos y la cantidad de vecinos.
```

```
################ FUNCION QUE REALIZA VOTACION DE LA ETIQUETA A PRESENTAR ####################

def votacion(vec,k):    #Funcion que recibe las etiquetas de los vecinos mas cercanos
                        #y el número de vecinos.
    frecuencia = []
    lista = []
    resultado = []
    freq_ord = []
    for a in vec:           #Se recorre cada fila de la matriz vecinos
        b=a.tolist()        #Se convierte cada fila a lista
        lista.extend(b)     #Se añade cada fila a un arreglo
        for c in b:         #Se recorre el contenido de cada fila
            frecuencia.append(b.count(c))      #Se calcula la frecuencia de aparición de
                                               #cada etiqueta dentro de la fila y se añade
                                               #a una nueva matriz llamada frecuencia.

    tuplas3 = [(frecuencia[d], lista[d]) for d in range(len(frecuencia))]   #Se empareja el valor de la
                                                                            #frecuencia con su respectiva etiqueta.
    for x in range(0,len(tuplas3),k):
        h = tuplas3[x:x+k]                     #Se organiza la tupla creada en sub-grupos de k-elementos
        freq_ord.append(h)                     #Cada subgrupo se almacena en la matriz freq_ord.
    for c in freq_ord:                         #Se recorre las filas de la matriz freq_ord.
        c.sort(reverse = True)                 #Se ordena descendentemente acorde al valor de frecuencia.
                                               #(Las etiquetas con mayor frecuencia de aparición se ubicarán
                                               #al inicio de cada fila).
    for g in freq_ord:                         #Se recorre la nueva matriz ordenada
        resultado.append(int(g[0][1]))         #En la matriz resultado se almacena solamente la etiqueta
                                               #correspondiente al primer valor de cada fila.
    resultado = np.asarray(resultado)          #Se convierte a arreglo numpy
    return resultado                           #Se presenta la matriz con la etiqueta a presentar.
```

```
        ################## FUNCION QUE EJECUTA TODO EL PROCEDIMIENTO ######################

def principal(matriz_train, etiq,vecinos):    #Recibe como parámetros la matriz PCA de entrenamiento,
                                              #La matriz de etiquetas y el número de vecinos que
                                              #se desea calcular.
    dist = dist_eucl(matriz_train)            #Función que calcula distancia euclideana.
    vecinos_1, k = obt_vecinos(dist, etiq, vecinos)      #Función que obtiene las etiquetas de
                                                         #los vecinos cercanos.
    votos = votacion(vecinos_1, k)            #Función que decide la etiqueta a presentar.
    return votos                              #Se presenta dicha etiqueta.
```

ANEXO 3: TABLA DE VALORACIÓN DE LENGUAJES DE PROGRAMACIÓN

Lenguaje / Características	Python	JavaScript	Java	C, C#, C++
Código Abierto	X	X	X	X
Interpretado	X	X		
Multiplataforma	X	X	X	X
Orientada a Objetos	X	X	X	
Alto Nivel	X	X	X	
Top 1 según GitHub	X			
Top 1 según IEEE Spectrum	X			
Top 1 según TIOBE			X	
Facilidad de sintaxis	X			
Tipado dinámico	X			
Facilidad de aprendizaje	X			
Gran cantidad de modelos de ML* implementados	X			
Herramientas para creación de GUI	X	X	X	
Disponibilidad para trabajar con Google Colab	X			
Recolector de basura	X	X	X	
Mayor Velocidad				X
Mayor cantidad de librerías para ML*	X			
Mayor Seguridad			X	
Mayor Comunidad de Programadores				X
Gran cantidad de Libros con base técnica			X	X
Total	**16/20**	**7/20**	**9/20**	**5/20**

ML signfica Machine Learning*

ANEXO 4: TABLA DE VALORACIÓN DE ALGORITMOS DE DETECCIÓN

Algoritmo / Características	Viola & Jones	LBP
Mayor Precisión	X	
Mayor Velocidad		X
Mayor Cantidad de VP	X	
Menor Cantidad de FP	X	
Menor Cantidad de FN	X	
Mayor tolerancia a cambios de iluminación		X
Uso de clasificadores simples	X	
Implementados en Python	X	X
Mejor detección en tiempo real		X
Menor tiempo de entrenamiento		X
Elimina cálculo de imagen a escala múltiple	X	
Mayor tolerancia a rotación del rostro		X
Repercusión en trabajos relacionados con detección	X	
Total	**8/13**	**6/13**

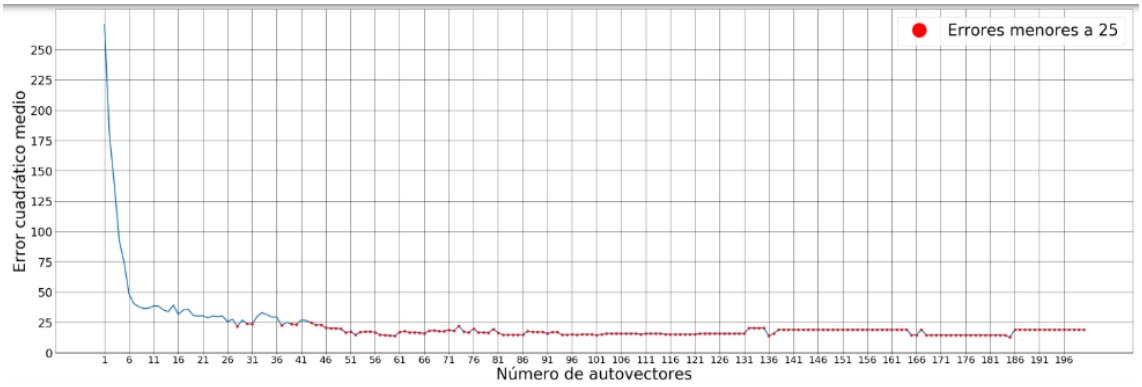
ANEXO 5: TABLA DE VALARACIÓN DE ALGORITMOS CLASIFICADORES

Algoritmo / Características	KNN	SVM	GMM
Mayor Precisión		X	
Mayor Velocidad	X		
Menor Complejidad Matemática	X		
Disponibles en Sk-Learn	X	X	X
Menor costo computacional en problemas de dimensionalidad reducida	X		
No necesita entrenamiento previo	X		
Se construye un modelo global		X	X
No necesita adaptación para más de dos clases	X		
Clasifica datos linealmente no separables		X	X
Más óptimo en espacios dimensionales grandes		X	X
Tolerante al ruido	X	X	X
Total	**7/11**	**6/11**	**5/11**

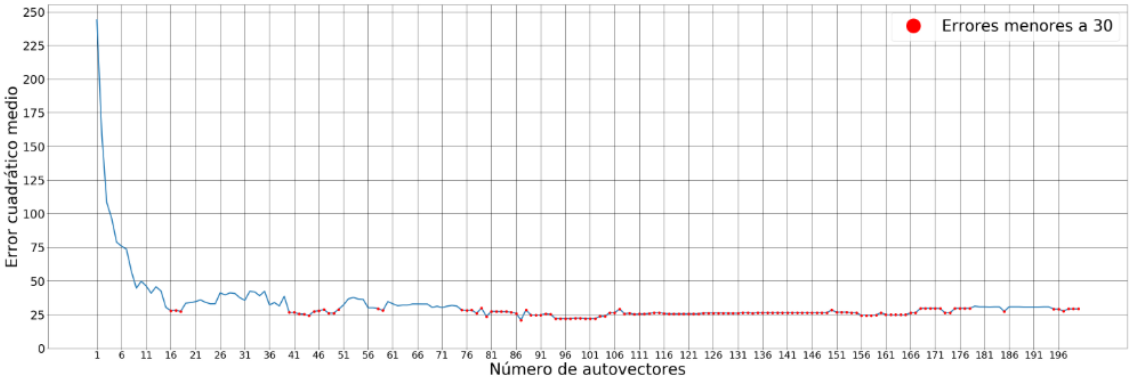
ANEXO 6: GRÁFICAS DE ALINEACIÓN CON UNO, TRES, CUATRO Y CINCO VECINOS CERCANOS USANDO PCA

- Para 40 usuarios

Utilizando un vecino cercano



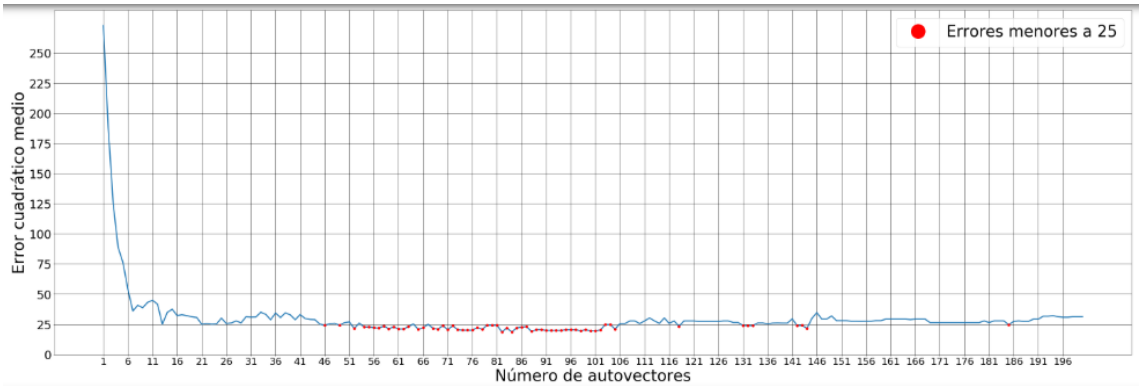
a) KNN de SK-Learn



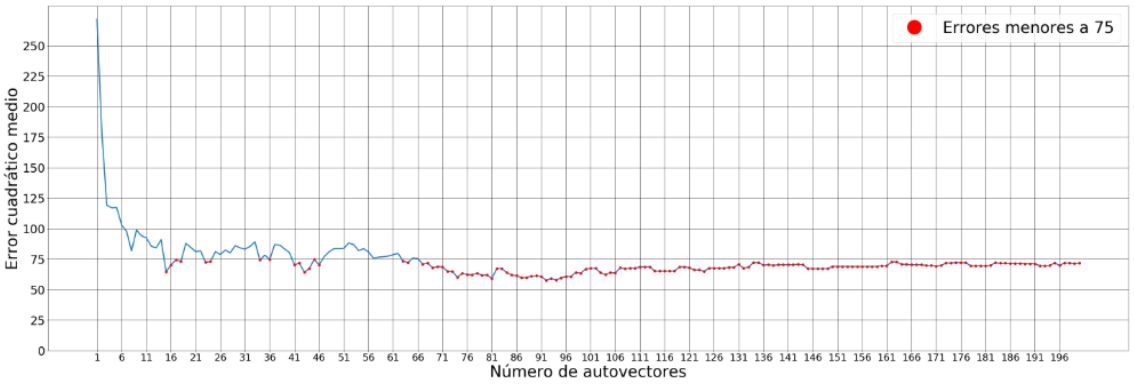
b) KNN desarrollado por el autor

Figura 47. *Gráficas de Error Cuadrático Medio de PCA para fotos sin normalizar usando 40 usuarios y 1 vecino cercano*

Utilizando tres vecinos cercanos



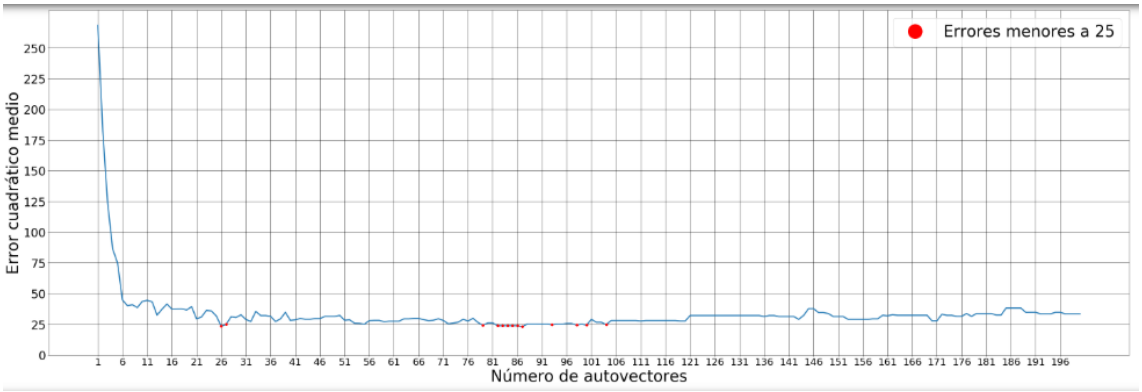
a) KNN de SK-Learn



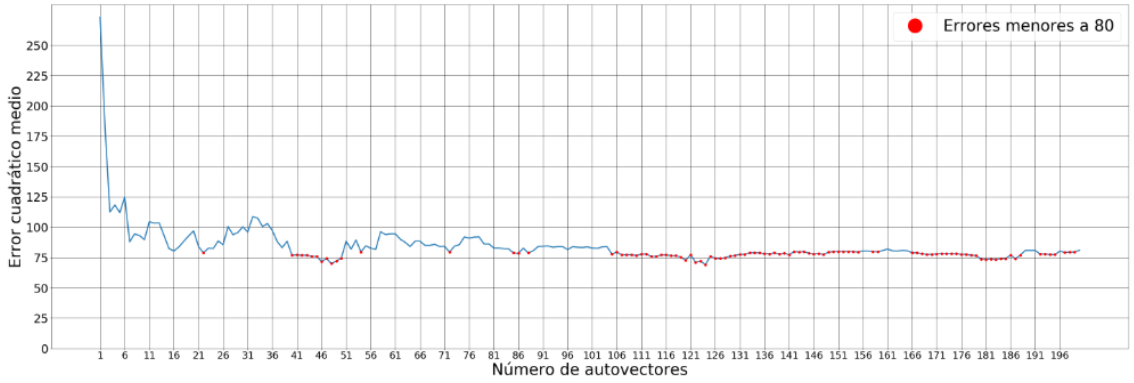
b) KNN desarrollado por el autor

Figura 48. *Gráficas de Error Cuadrático Medio de PCA para fotos sin normalizar usando 40 usuarios y 3 vecinos cercanos*

Utilizando cuatro vecinos cercanos



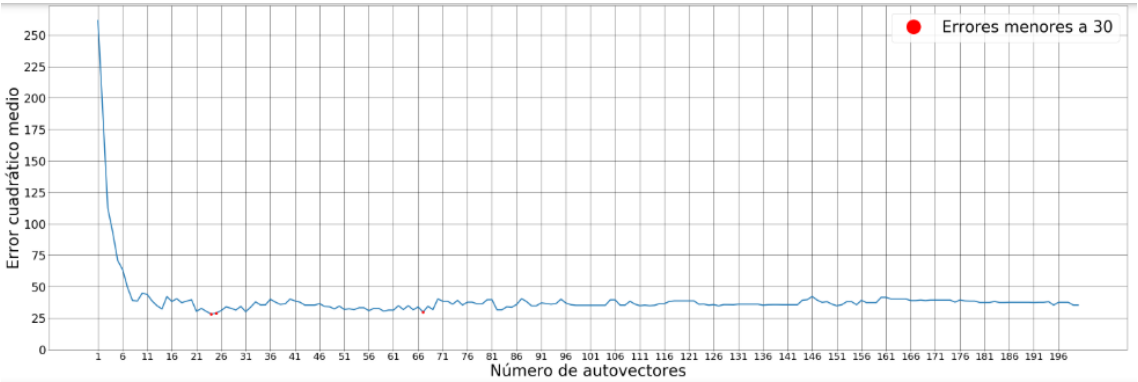
a) KNN de SK-Learn



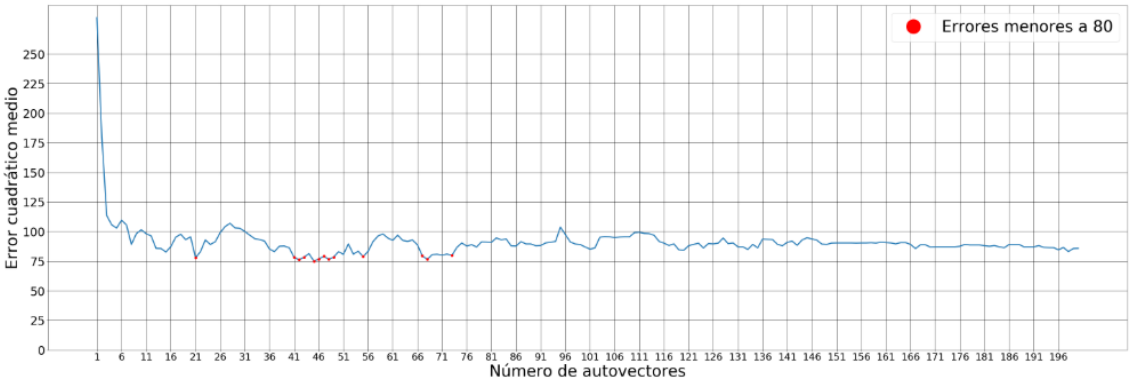
b) KNN desarrollado por el autor

Figura 49. *Gráficas de Error Cuadrático Medio de PCA para fotos sin normalizar usando 40 usuarios y 4 vecinos cercanos*

Utilizando cinco vecinos cercanos



a) KNN de SK-Learn

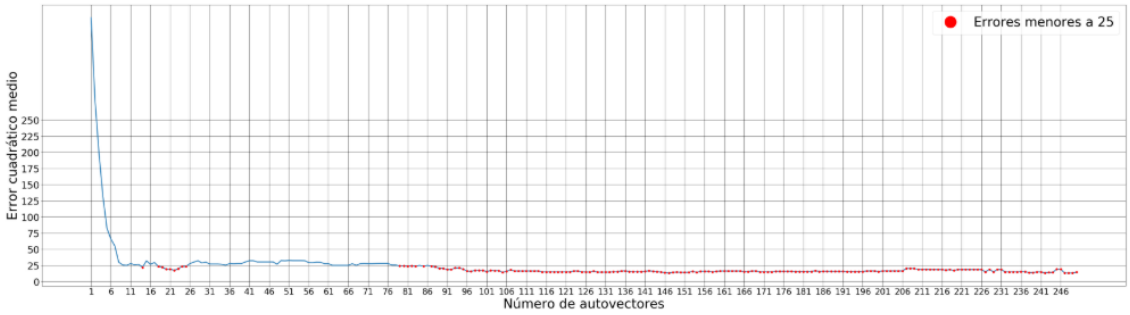


b) KNN desarrollado por el autor

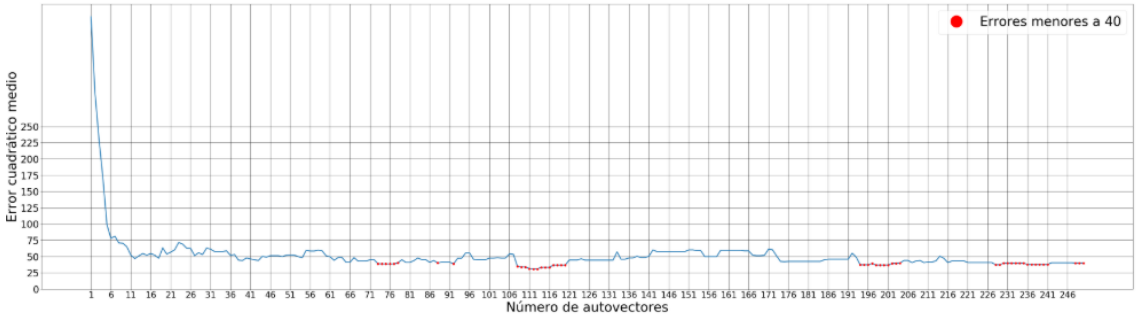
Figura 50. *Gráficas de Error Cuadrático Medio de PCA para fotos sin normalizar usando 40 usuarios y 5 vecinos cercanos*

- Para 50 usuarios

Utilizando un vecino cercano



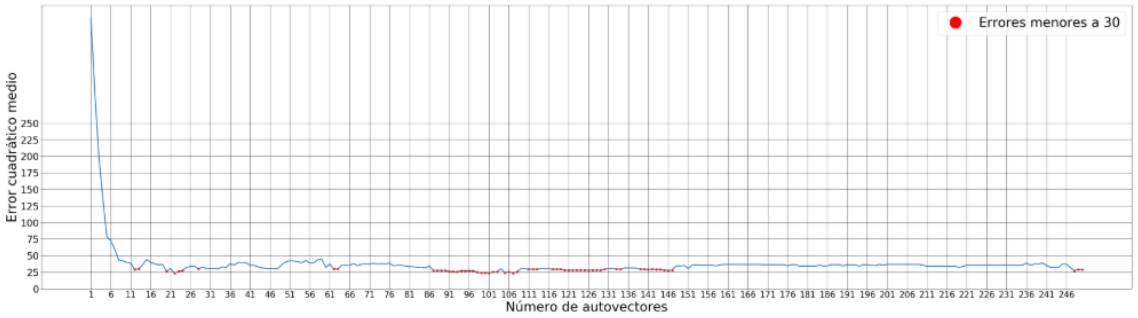
a) KNN de SK-Learn



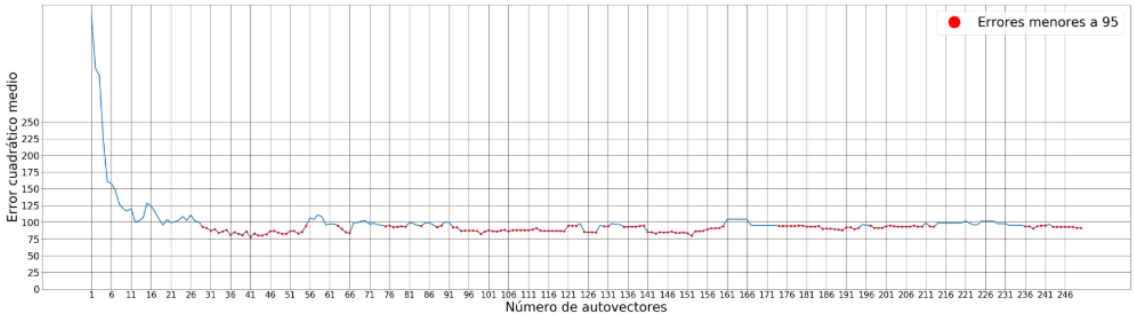
b) KNN desarrollado por el autor

Figura 51. *Gráficas de Error Cuadrático Medio de PCA para fotos sin normalizar usando 50 usuarios y 1 vecino cercano*

Utilizando tres vecinos cercanos



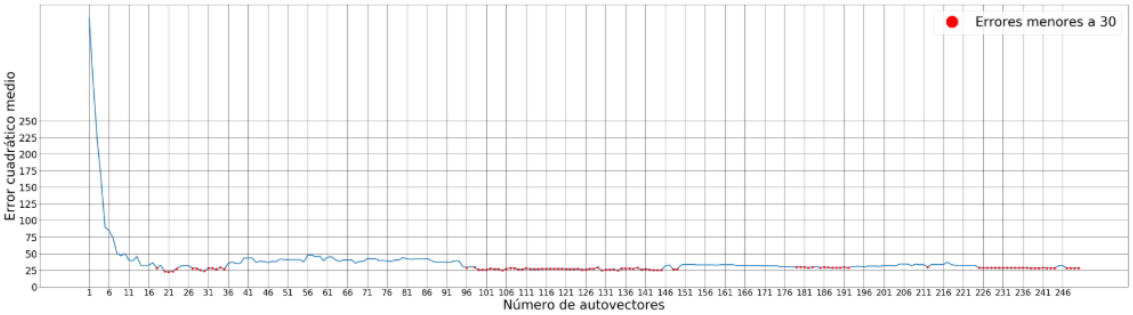
a) KNN de SK-Learn



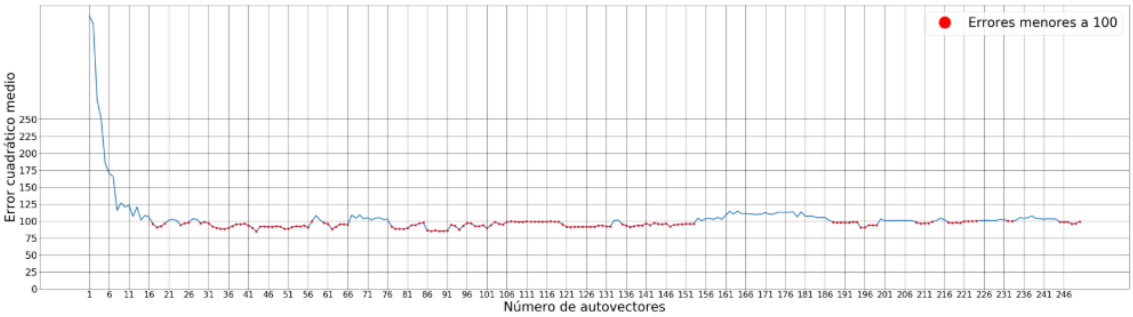
b) KNN desarrollado por el autor

Figura 52. *Gráficas de Error Cuadrático Medio de PCA para fotos sin normalizar usando 50 usuarios y 3 vecinos cercanos*

Utilizando cuatro vecinos cercanos



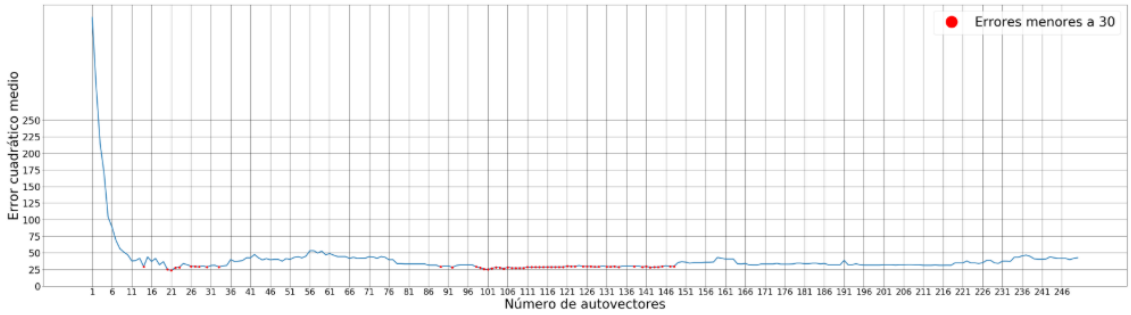
a) KNN de SK-Learn



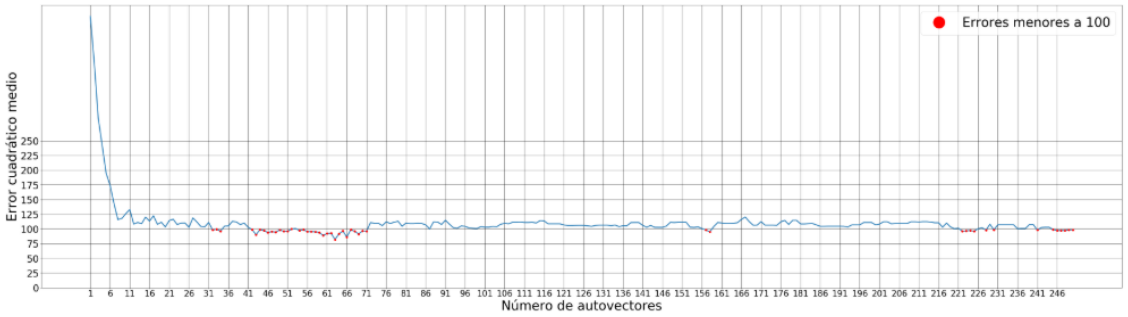
b) KNN desarrollado por el autor

Figura 53. *Gráficas de Error Cuadrático Medio de PCA para fotos sin normalizar usando 50 usuarios y 4 vecinos cercanos*

Utilizando cinco vecinos cercanos



a) KNN de SK-Learn



b) KNN desarrollado por el autor

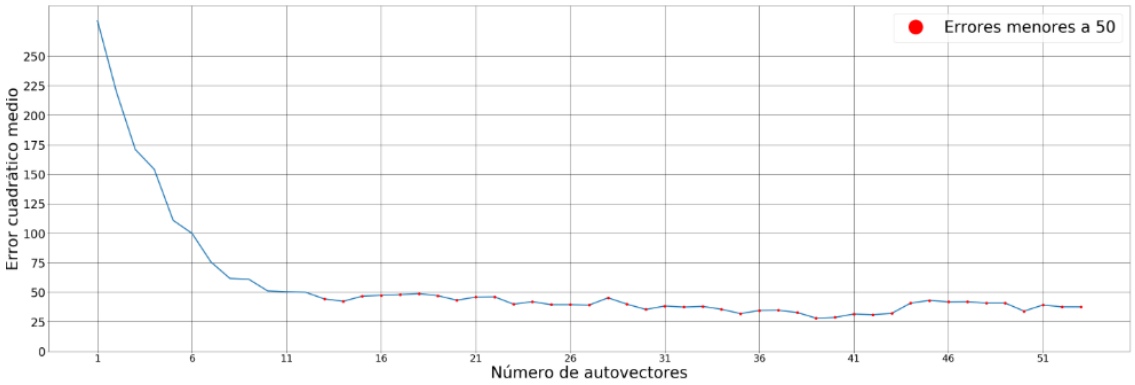
Figura 54. *Gráficas de Error Cuadrático Medio de PCA para fotos sin normalizar usando 50 usuarios y 5 vecinos cercanos*

ANEXO 7: GRÁFICAS DE ALINEACIÓN CON UNO, TRES, CUATRO Y CINCO VECINOS CERCANOS USANDO LDA

- Para 40 usuarios

Utilizando un vecino cercano

a) KNN de SK-Learn

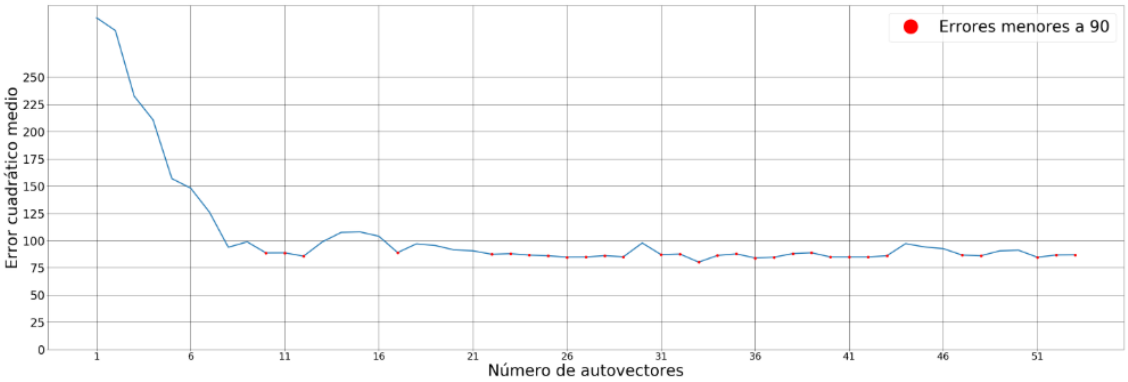


b) KNN desarrollado por el autor

Figura 55. *Gráficas de Error Cuadrático Medio de LDA para fotos sin normalizar usando 40 usuarios y 1 vecino cercano*

Utilizando tres vecinos cercanos

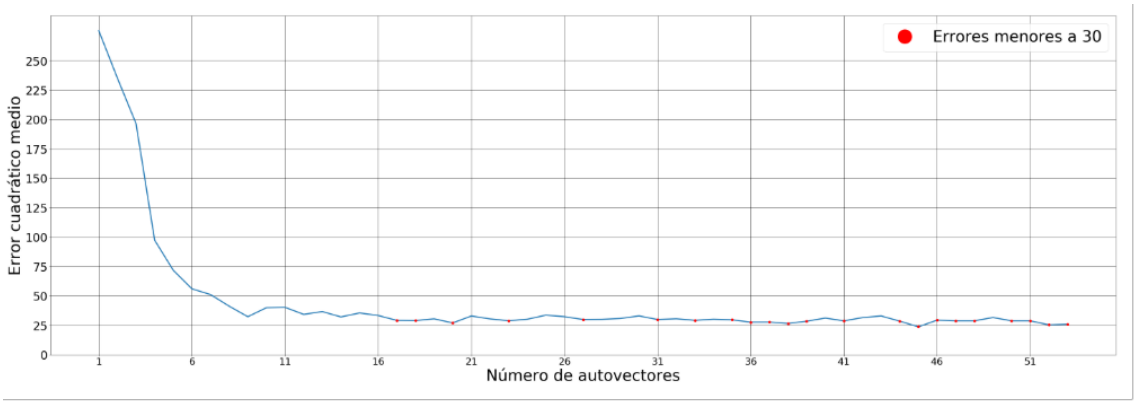
a) KNN de SK-Learn



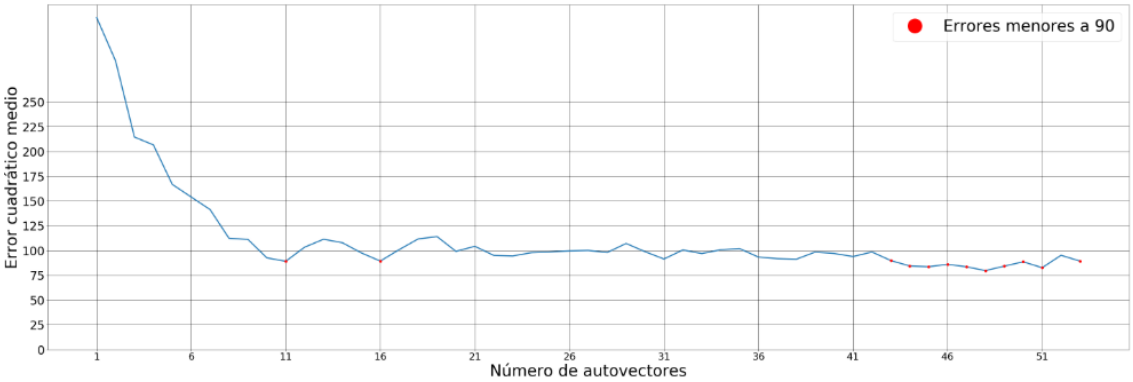
b) KNN desarrollado por el autor

Figura 56. *Gráficas de Error Cuadrático Medio de LDA para fotos sin normalizar usando 40 usuarios y 3 vecinos cercanos*

Utilizando cuatro vecinos cercanos



a) KNN de SK-Learn

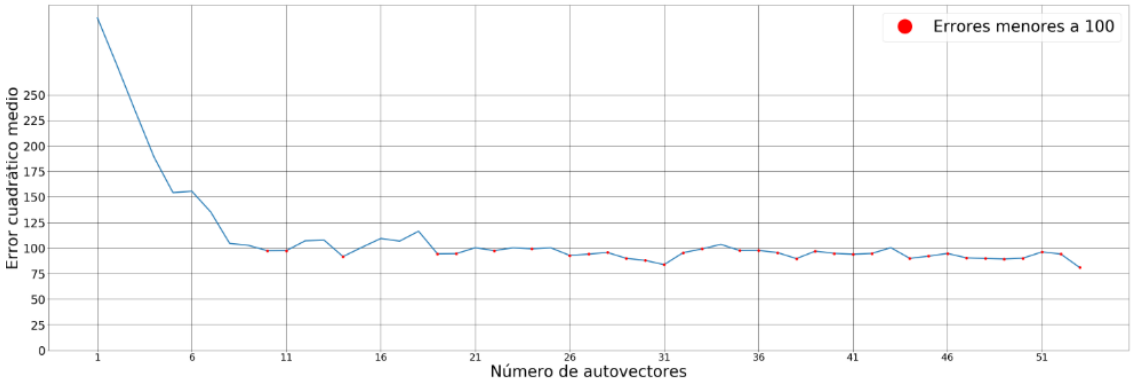


b) KNN desarrollado por el autor

Figura 57. *Gráficas de Error Cuadrático Medio de LDA para fotos sin normalizar usando 40 usuarios y 4 vecinos cercanos*

Utilizando cinco vecinos cercanos

a) KNN de SK-Learn

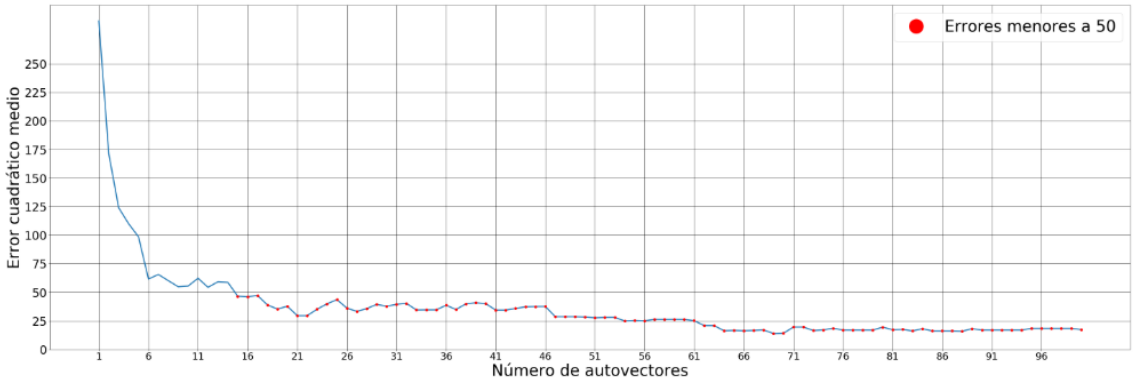


b) KNN desarrollado por el autor

Figura 58. *Gráficas de Error Cuadrático Medio de LDA para fotos sin normalizar usando 40 usuarios y 5 vecinos cercanos*

- Para 50 usuarios

Utilizando un vecino cercano

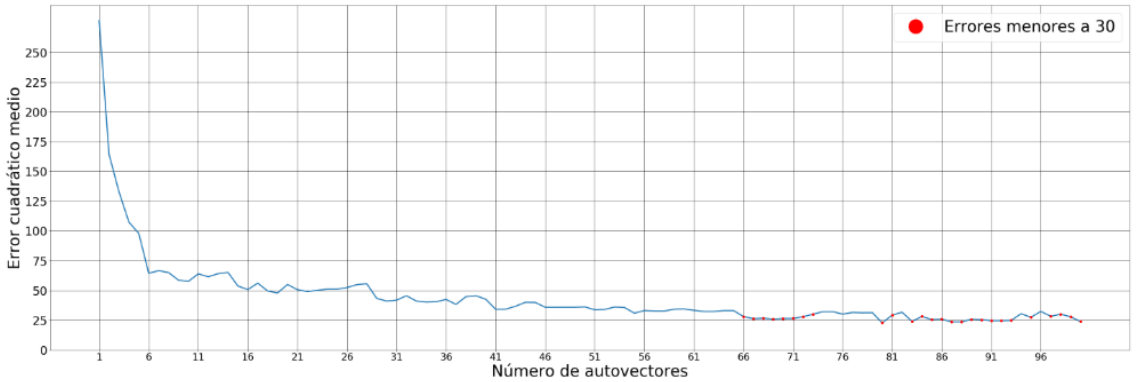


a) KNN de SK-Learn

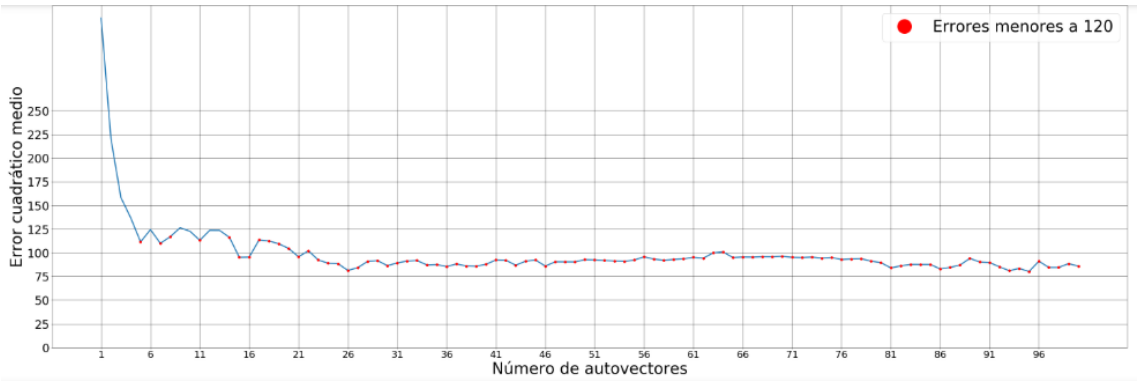
b) KNN desarrollado por el autor

Figura 59. *Gráficas de Error Cuadrático Medio de LDA para fotos sin normalizar usando 50 usuarios y 1 vecino cercano*

Utilizando tres vecinos cercanos



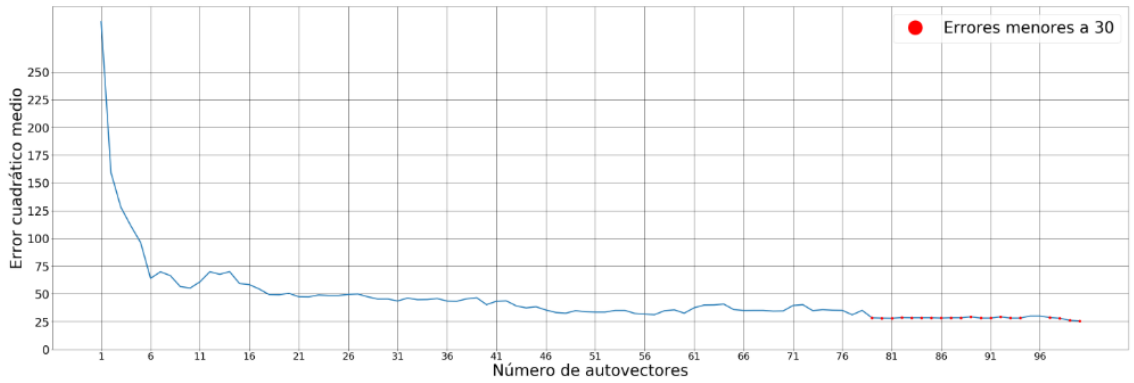
a) KNN de SK-Learn



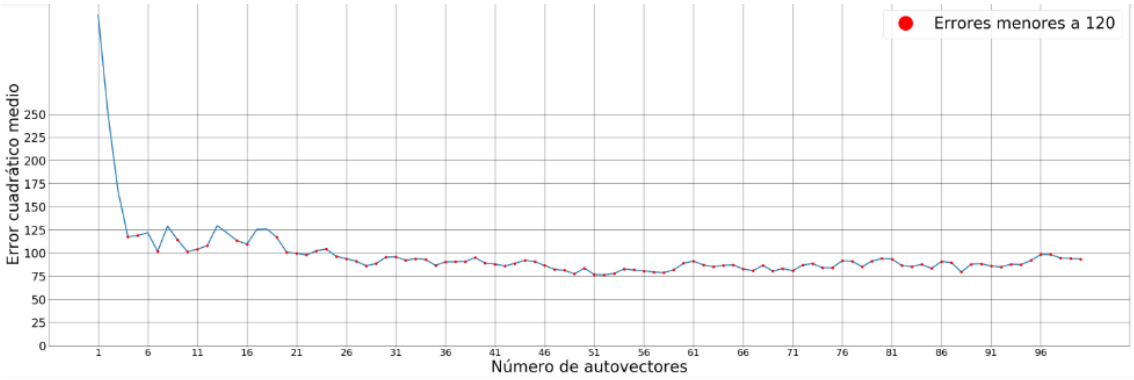
b) KNN desarrollado por el autor

Figura 60. *Gráficas de Error Cuadrático Medio de LDA para fotos sin normalizar usando 50 usuarios y 3 vecinos cercanos*

Utilizando cuatro vecinos cercanos



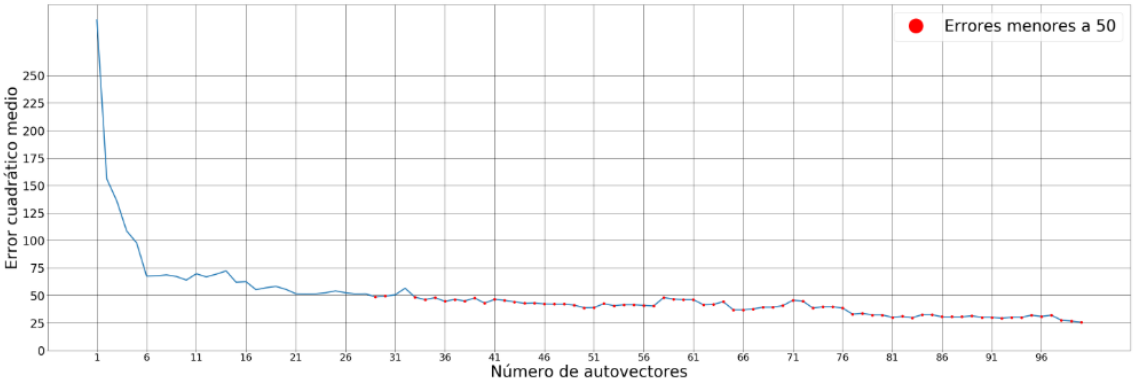
a) KNN de SK-Learn



b) KNN desarrollado por el autor

Figura 61. *Gráficas de Error Cuadrático Medio de LDA para fotos sin normalizar usando 50 usuarios y 4 vecinos cercanos*

Utilizando cinco vecinos cercanos



a) KNN de SK-Learn

b) KNN desarrollado por el autor

Figura 62. *Gráficas de Error Cuadrático Medio de LDA para fotos sin normalizar usando 50 usuarios y 5 vecinos cercanos*

ANEXO 8: TRABAJOS FUTUROS

El reconocimiento facial en dos dimensiones se basa en el reconocimiento de patrones de imágenes creados a partir de matrices bidimensionales de intensidades, por esta razón se producen errores de reconocimiento cuando existen cambios de rotación, de escala, o de intensidad de pixeles entre las imágenes de la base de datos y la imagen a detectar; además, se pierde una gran cantidad de información correspondiente a características faciales cuando se proyecta el rostro en un espacio de dos dimensiones. Por otra parte, los sistemas de reconocimiento facial en 2D podrían ser vulnerados al usar fotografías en lugar de rostros reales al momento de comparar los datos; siendo esta una desventaja considerable de seguridad. Por estas razones es necesario profundizar en la investigación de técnicas de reconocimiento tridimensional las cuales procesan información adquirida a través de digitalizadores 3D, conteniendo una gran densidad de puntos correspondientes a la profundidad del rostro y que disminuyen el efecto negativo de los factores mencionados anteriormente.

Como se mencionó anteriormente, la iluminación es uno de los principales problemas que afectan a la visión por computadora; los cambios de iluminación pueden generar sombras en los rostros a detectar, acentuando o disminuyendo características faciales importantes, lo cual provoca que se produzcan representaciones diferentes de un mismo rostro y finalmente un reconocimiento erróneo. Por lo mencionado anteriormente se puede optimizar el reconocimiento facial en dos dimensiones investigando métodos de normalización que mitiguen los cambios producidos en la imagen debido a la iluminación.

En adición, para construir una plataforma íntegra de visión por computador, se puede complementar el presente trabajo con un sistema de detección de rostros más eficiente, añadiendo sensores o cámaras situados en un punto específico, como por ejemplo en el edificio de laboratorios de la Facultad de Energía; esto permitiría establecer un control de acceso que mejore la seguridad y registre las personas que ingresen a cada laboratorio; así mismo, la información de registro de cada persona se puede almacenar en los servidores de la Unidad de Telecomunicaciones o en un servidor colocado dentro del mismo edificio.

Por otra parte, el reconocimiento facial también puede optimizar el sector educativo, añadiendo mejoras en los servicios virtuales; es decir, un profesor podría brindar clases, foros o conferencias online, y a través del reconocimiento facial se puede identificar al estudiante en todo momento, evitando la suplantación de identidad.

Otra forma de optimizar el sector educativo se puede realizar acelerando el proceso de matriculación de estudiantes a través de un sistema de reconocimiento facial ubicado en el sector administrativo de cada facultad, cada estudiante deberá acercarse a dicho sistema para capturar una imagen de su rostro y vincular sus datos académicos como: ciclo actual, ciclo siguiente, materias aprobadas, etc., los cuales estarán almacenados en los servidores de la facultad.

En cuanto a seguridad en el área académica, se puede implementar un sistema de reconocimiento facial que sirva a los docentes como medio de control al momento de asignar las calificaciones en el sistema de gestión académica; es decir, una ventana emergerá del navegador web para realizar una captura desde la videocámara, validando de esta manera la identificación de cada docente, lo cual evitará cualquier tipo de fraude por parte de personas no autorizadas.

Finalmente, se puede aportar en la conservación del medio ambiente al reducir el consumo de papel; por ejemplo, en las biblioteca o laboratorios de cada facultad se puede instalar sistemas de reconocimiento facial que almacenen la hora de entrada y la identidad de cada estudiante; evitando llenar hojas de registro.