UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

ANÁLISIS DE MÉTODOS DE RECONOCIMIENTO FACIAL BAJO EL SISTEMA OPERATIVO ANDROID

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

Autor
Bach. Bernal Leyva Alex Yuri

Asesor
Mg. Samillán Ayala Alberto Enrique

Línea de Investigación:

Ciencias de la computación

Pimentel – Perú
2018

ANÁLISIS DE MÉTODOS DE RECONOCIMIENTO FACIAL BAJO EL SISTEMA OPERATIVO ANDROID

Aprobación de la Tesis

Dr. Gutierrez Gutierrez Jorge Luis

Presidente del jurado de tesis

Ing. Mejía Cabrera Heber Iván

Secretario del jurado de tesis

Ing. Tuesta Monteza Victor Alexci

Vocal del jurado de tesis

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

DEDICATORIA

Este proyecto de tesis está dedicado a las personas que me apoyaron a lo largo de mi carrera, que a pesar de todo dieron un granito de apoyo para lograr mis objetivos, principalmente dedicado a mis padres que me apoyaron en lo emocional.

A la persona que estuvo a mi lado dando apoyo y dándome su aliento que todo se puede lograr en esta vida solo se necesita esfuerzo y dedicación, y por último dedicado a mi familia que me apoyaron en todos los momentos altos y bajos.

A mis compañeros y docentes de clases de la USS que estuvieron presentes en mi formación.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

AGRADECIMIENTO

Agradezco primeramente a Dios por guiarme durante estos diez ciclos de mi carrera, dándome día a día sabiduría y fortaleza para seguir adelante y poder sobresalir en los obstáculos que se presentaron, pero a su vez haberme concedido muchas experiencias extraordinarias.

Agradezco a mis queridos padres Asunción y Dorelis por apoyarme para seguir estudiando una carrera profesional, por estar a mi lado en todo el transcurso de mi carrera, inculcándome valores y enseñándome a ser una excelente persona y así en lo largo de mi vida demostrar tanto en lo personal como en lo profesional.

Agradezco a mis dos hermanos por ser parte importante de mi vida y estar durante este tiempo apoyándome en cada decisión tomada llenando mis días de alegrías especialmente en los momentos difíciles.

Agradezco a mis profesores por dedicarnos parte de su tiempo y así brindarnos gran parte de su conocimiento, su apoyo incondicional y en especial su amistad.

Agradezco a la empresa SIEMPRESOFT por darme la oportunidad de formar parte de su organización, brindándome su apoyo y mucho conocimiento en la vida profesional.

Por otro lado, agradezco mucho a las personas que me apoyaron para el desarrollo de mi tesis brindándome su apoyo incondicional, y que siempre estuvieron presentes dando aliento de lograr los objetivos trazos como profesional.

¡Gracias!

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

INDICE

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Índice de Tabla

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Índice de Figura

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Resumen

Este trabajo plantea como objetivo desarrollar un prototipo de reconocimiento facial aplicando los diferentes algoritmos matemáticos para la identificación del rostro.

Por otro lado, para realizar la implementación de los métodos de reconocimiento facial se realizó un análisis de los métodos que mejores resultados han obtenido en los últimos estudios, verificando si cumplen con lo necesario en tema de orientación y rendimiento computacional.

De tal modo que los métodos seleccionados se tomaron diversos resultados de tesis y artículos científicos de base de datos reconocidos, de manera que se implementó el método de Patrón Binario Local y EigenFaces en lenguaje C y el prototipo en lenguaje java.

Teniendo como muestra 50 personas con 15 imágenes por personas teniendo un total de 750 imágenes en la base de datos que fueron entrenados en ambos métodos LBP Y EigenFaces, las imágenes fueron adquiridas en diferentes tipos de escenarios (mañana, tarde y noche) con diferente orientación de rostro. Para la clasificación solo se tomó 1 imagen en tiempo real por prueba realizada y dicha imagen no es almacenada en el dispositivo solo se utiliza para la clasificación.

Por otro lado, se implementó un clasificador de bajo nivel por motivo que los dispositivos inteligentes no cuentan con gran recurso computacional basado a ese motivo fue que se implementó en el método LBP el clasificador probabilístico Chi-cuadrado y para el método EigenFaces el clasificador de distancia euclidiana.

En las pruebas realizadas se obtuvo diferentes resultados altos siendo el más resaltante el método LBP con el 100% de reconocimiento en las pruebas realizadas en noche de manera frontal. Y un tiempo de respuesta de menor a un segundo lo durante los diferentes escenarios y en el caso de EigenFaces obtuvo un resultado menor de acierto.

Palabras clave: Reconocimiento facial, Métodos, Patrón Binario Local, EigenFaces, rendimiento, orientación.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Abstract

This work aims to develop a prototype facial recognition applying the different mathematical algorithms for the identification of the face.

On the other hand, to carry out the implementation of facial recognition methods, an analysis was made of the methods that have obtained the best results in the last studies, verifying if they comply with what is necessary in terms of orientation and computational performance.

In such a way that the selected methods took diverse results of theses and scientific articles of recognized data base, so that the method of Local Binary Pattern and EigenFaces was implemented in C ++ language and the prototype in Java language.

Taking as sample 50 people with 15 images per person having a total of 750 images in the database that were trained in both LBP and EigenFaces methods, the images were acquired in different types of scenarios (morning, afternoon and evening) with different orientation of face. For the classification, only 1 image was taken in real time per test performed and this image is not stored in the device, it is only used for classification.

On the other hand, a low-level classifier was implemented because smart devices do not have a large computational resource based on this reason, the Chi-square probabilistic classifier was implemented in the LBP method and the distance classifier was implemented for the EigenFaces method. Euclidean

In the tests carried out, different high results were obtained, the most outstanding being the LBP method with 100% recognition in the frontal night tests. And a response time of less than one second during the different scenarios and in the case of EigenFaces obtained a lower score.

Key Words

Facial recognition, Methods, Local Binary Pattern, EigenFaces, Performance, Orientation.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Introducción

Hoy en día las Tecnologías de la Información (TI) se desarrollan ligeramente de tal modo que muchas de estas tecnologías juegan un papel importante en las empresas. La tecnología biométrica brinda soporte a los procesos empresariales desde los niveles operativos hasta los de la alta dirección, apoyando conjuntamente a la toma de decisiones y seguridad de la información.

Las empresas desde las más pequeñas hasta las más grandes necesitan cuidar el acceso a la información por lo que se ha desarrollado una serie de técnicas biométricas para salvaguardar la filtración de información.

El reconocimiento facial es una de las técnicas más investigadas en la actualidad, que consiste en la interacción del hardware y software, de tal modo que optimice el proceso de identificación de usuario, a diferencia de otras técnicas biometrías es más costosa la falsificación de identidad.

Para lograr el reconocimiento facial se tiene que procesar la imagen, el cual es una manera más eficiente de persuadir los objetos y texturas del entorno, abriendo paso a la interacción de hombre-máquina.

De tal manera que ha surgido diferentes métodos de reconocimiento de rostro como por ejemplo por textura del rostro, por distancia de los puntos principales de la cara, por construcción de rostro en 3D, entre otros; tales métodos necesitan de plataforma tecnológica para la implementación tales como equipos con el suficiente recurso. De tal manera que existen equipos emergentes con escaso recurso tales como los dispositivos móviles, por el cual algunos métodos de reconocimiento facial consumen demasiado recurso y surge la necesidad de analizar que método consume poco recurso pero con una taza alta aceptación.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

CAPÍTULO I: PROBLEMA DE INVESTIGACIÓN

1.1. Situación Problemática

Hoy en día las Tecnologías de la Información (TI) se desarrollan ligeramente de tal modo que muchas de estas tecnologías juegan un papel importante en las organizaciones. (Akariman, Jati, & Novianty, 2015) la tecnología biométrica brinda soporte a los procesos empresariales desde los niveles operativos hasta los de la alta dirección apoyando conjuntamente a la toma de decisiones y seguridad de la información (Bellakhdhar, Loukil, & Abid, 2012) entre uno de los procesos se encuentran los del control de asistencia de personal, el cual está siendo soportado por distintos tipos de tecnologías de biometría(Gofman & Mitra, 2016), entre ellos podemos mencionar la biometría estática medidas por las características físicas. Las principales investigaciones y aplicaciones de esta rama de la biometría están fundadas en los sistemas biométricos de análisis de retina e iris, huellas digitales, geometría de la mano, reconocimiento de rostro. Por lo contrario, la biometría dinámica esta medida por el comportamiento de un individuo y dentro de esta biométrica está siendo investigada y aplicada a los sistemas de reconocimiento de voz y firma manuscrita(Stoimenov, Tsenov, Mladenov, & Member, 2016).

Los sistemas antes mencionados para su implementación requieren de plataformas tecnológicas tales como equipos con el suficiente recurso para la implementación del software, también surge la necesidad de implementar redes o servidores para el almacenamiento de la información, y tales tecnologías requieren la necesidad de alimentación de energía, ya que sin este elemento no podrían funcionar ningún equipo implementado(Bertok & Fazekas, 2016; Stoimenov et al., 2016); Sin embargo existen tecnologías emergentes de biometría y muchos de ellos soportadas en dispositivos móviles tales como: Proteus que esta implementado en sistema operativo Android donde el autor Gofman & Mitra, 2016, implementa la biometría multimodal teniendo en cuenta las características del rostro y voz, el investigador Bertok & Fazekas, 2016,

viendo el continuo lanzamiento teléfonos inteligentes con mayor capacidad y más fuerte que el anterior, adapto soluciones basadas en PC a plataformas móviles centrándose en la adaptación de los algoritmos de plataformas móviles Android con el objetivo de crear una arquitectura lógica de la aplicación teniendo el mismo código fuente para el sistema operativo Windows y Android(Bertok & Fazekas, 2016; Gofman & Mitra, 2016).

Por ejemplo Akariman et al., (2015) Para la identificación de la cara humana realiza una serie de procesos: como la recolección de las imágenes, luego realiza el pre-procesamiento de las imágenes, continuando con el proceso de la detección de la cara, pasa al proceso de la extracción de las características importantes de la imagen facial, por ultimo realiza el proceso de clasificación e identificaron de la imagen del rostro. Por ello en la investigación se ha utilizado el algoritmo Patrón Binario Local (LBP), como un algoritmo de reconocimiento facial permitiendo poder obtener resultados relativamente rápido. El cual fue implementada en un sistema operativo Android para lograr el reconocimiento en tiempo real, por otro lado, otras investigaciones realizaron la implementación de otro tipo de método e implementaron un clasificador aplicando redes neuronales.

Otro caso muy similar al anterior investigador realizaron los mismos procesos de reconocimiento facial donde el autor Stoimenov, Tsenov, Mladenov, & Member, (2016) En esta investigación utilizaron otro tipo de metodología diferente pero con resultados muy buenos, primeramente hicieron la recolección de imágenes, después para la identificación y ubicación de los puntos importantes del rostro utilizaron la librería de “Google Mobile Visión API”; para poder utilizar dichas librerías tuvieron que crear un proyecto de Android en Android Studio, luego realizaron un filtrado de imágenes y la transformaron a escala de grises para la reducción de dimensionalidad, luego como tercer paso para la extracción de características calcularon las distancias de los 8 puntos importantes del

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

rostro, y por último paso aplicaron redes neuronales de Feed-forward para la clasificación y la identificación del rostro.

Sin embargo se ha hecho investigaciones con diferentes métodos de reconocimiento de imágenes y a partir de ello surgen una variedad de problemas, por ejemplo: Stoimenov et al., (2016) Debido a que la mayoría de los teléfonos inteligentes tienen un poder computacional modesto, es mejor trabajar con un número reducido de puntos faciales. Otro problema que han surgido es que Akariman et al., (2015) para la implementación en dispositivos móviles para el reconocimiento facial ocurre el problema con los diferentes intensidades de luces, también variación de la distancia de la foto, y la captura de imágenes con diferente orientación de la cara.

Como vemos hay métodos con resultados aceptables y muy eficientes al reconocer un rostro, sin embargo, hay otros problemas relacionados con el rendimiento, como sabemos que el rendimiento en las aplicaciones en dispositivos móviles es muy importante ya que el consumo de recursos y tiempo en respuesta es muy crucial en dispositivos móviles con capacidades limitadas. Este autor (Lee, Kim, Kim, & Whangbo, 2013) realizo una investigación utilizando el Modelo de Forma Activa (ASM) con tres mejoras en el algoritmo donde tuvo buenos resultados en tiempo de respuesta y eficiencia.

Es por esto, que en este proyecto de investigación se plantea hacer un análisis de métodos de reconocimiento facial en rendimiento y orientación de la cara utilizando el sistema operativo Android en teléfonos inteligentes.

1.2. Formulación del problema

¿Qué método de reconocimiento facial se desempeña mejor en condiciones de orientación y rendimiento en tiempo real, para la implementación en dispositivos móviles bajo el sistema operativo Android?

1.3. Delimitación de la Investigación

El presente proyecto de investigación, ubicado dentro de las ciencias de la computación, se orienta al desarrollo de un prototipo software en dispositivos móviles Android que clasifique rostros con variación de ángulo de la cara y análisis de rendimiento de los algoritmos en ambiente no controlado.

Las delimitaciones de los algoritmos que hemos analizado para el reconocimiento de imágenes indican una tasa de reconocimiento eficiente, pero en un entorno controlado entonces esta investigación se realizó en ambientes no controlados, en tiempo real, donde se adicionara a la investigación la orientación o ángulo de la cara en horizontal y vertical, para posteriormente ver el porcentaje de acierto de reconocimiento del algoritmo. Esto se realizará en el tiempo de abril a diciembre del 2017.

1.4. Justificación e importancia de la investigación

Dado el interés y el valor que pueden tener ciertos recursos físicos e informáticos conocido como datos utilizados en las empresas y en la vida cotidiana, se vuelve una necesidad de crear e implementar tecnologías de protección tanto de acceso a los datos como de integridad. Las empresas emplean procesos para restringir el acceso solamente a cierto tipo de personas o para realizar un control de su personal. Sin embargo, las empresas pueden contratar personal para llevar un control de sus accesos o asistencias a las empresas o áreas de trabajo conllevando a generar gastos a dichas entidades.

Debido a los altos niveles de confiabilidad y seguridad que brinda la identificación de personas a través de las características únicas que posee el rostro, es que hoy en día las empresas que tercerizan servicio por un tiempo determinado ya sea corto o largo plazo, no disponen la suficiente área para implementar un control de asistencia completo, entonces surge la necesidad de implementar esta tecnología de reconocimiento facial en

dispositivos móviles, ya que no son muy costosos con facilidad de implantar este tipo de alternativa tecnología.

1.5. Limitaciones de la investigación

En la presente investigación tiene como limitación al momento de capturar las imágenes surgen los problemas de: espacio de memoria, ruido en la imagen, iluminación interno y externo, más de un rostro en la imagen, expresiones faciales, distancia de la persona con el dispositivo.

1.6. Objetivos de la investigación

Objetivo general

Analizar comparativamente métodos de reconocimiento facial bajo el sistema operativo Android para resolver problemas de rendimiento y orientación del ángulo de la cara.

Objetivos específicos

a) Diseñar el protocolo de adquisición de imágenes.
b) Seleccionar el método de reconocimiento de imágenes.
c) Realizar la implementación de los métodos de reconocimiento facial en un lenguaje de programación.
d) Analizar e interpretar los resultados de pruebas.

CAPÍTULO II: MARCO TEÓRICO

2.1 Antecedentes de estudios:

En el año 2004 el autor Moreno Diaz Ana Belen en la Universidad Politécnica De Madrid En La Facultad De Informática, realizo la tesis doctoral con el tema Reconocimiento Facial Automático Mediante Técnicas de Visión Tridimensional. En dicha tesis se pretende optimizar la robustez de las aplicaciones de Reconocimiento facial 3D creando técnicas que reduzca las condiciones de adquisición de imágenes relacionado con la intensidad de luz o posición de la fuente de luz al momento de la adquirir la imagen, y también las condiciones de posición o rotación del rostro.

Para la evaluación de las técnicas se crearon base de datos de imágenes faciales 3D. Dichas imágenes consiste en mallados tridimensionales que representa el rostro con la finalidad de lograr una mejor eficiencia y evadir los efectos perniciosos en el sistema causados por las circunstancias de sombras, orientación y variación de iluminación.

El sistema creado emplea información holística 3D del rostro, el reconocimiento facial se utilizaron el método de Análisis de componentes Principales (PCA) y para la clasificación la imagen se utilizó Support Vector Machines (SVM).

La base de datos contiene 579 imágenes faciales 3D correspondiente a 61 personas, tomadas 9 imágenes de cada individuo. Que está conformado por 2 imágenes frontal sin expresión, 4 imágenes presentando rotación en profundidad (2 en el eje horizontal y 2 en el eje vertical), y 3 imágenes mostrando diferentes expresiones.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

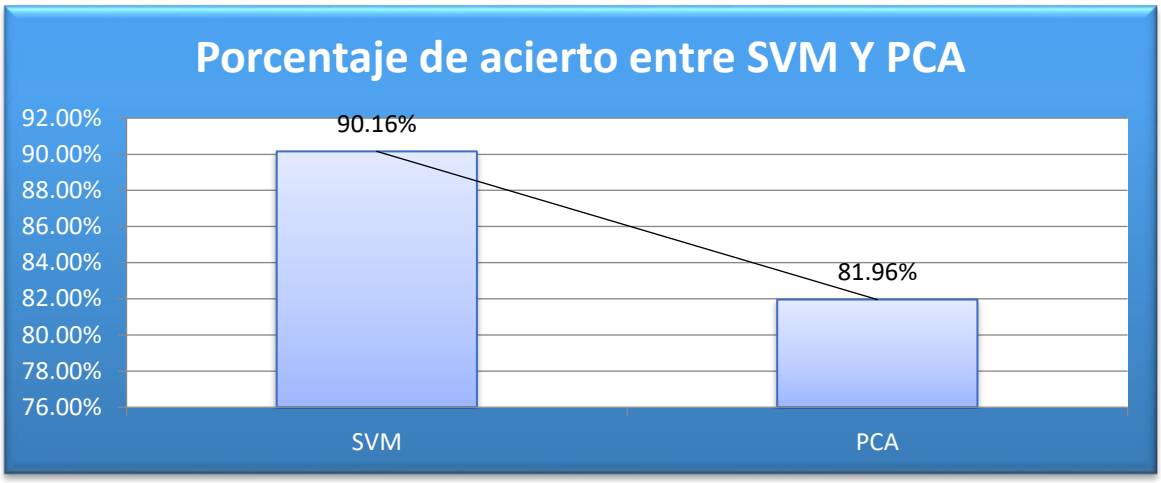


Figura 1: Porcentaje de acierto entre SVM Y PCA

Fuente: Ana Belén Moreno Díaz (2004). Resultados utilizando vectores de características en un experimento con una adquisición control.

Los resultados obtenidos en el uso de los vectores locales 3D para la clasificación en sus peores casos con diferencia de poses y expresiones faciales proporcionan el 78% de la tasa de aciertos con los mallados 3D. En su mejor caso cuando se ha entrenado imágenes con variación de pose y expresiones se alcanza el 90%. Entonces tiene una tasa de acierto comprendida entre 78% al 90% con el mallado 3D.

En el año 2015 los autores David Eduardo Espinoza Olguín, Peter Ignacio Jorquera Guillen en la Pontificia Universidad Católica De Valparaíso Facultad De Ingeniería Escuela De Ingeniería Informática, realizo la tesis con el tema "Reconocimiento Facial". Donde principalmente su problemática se deriva de su país Chile donde la biometría más utilizada es el reconocimiento de huella dactilar ya que es la menos costosa y fácil de utilizar, pero a su vez se han encontrado métodos para clonar la mano completa de la persona cosa que con el reconocimiento facial no sucedería ya que sería más difícil clonar los rastros de un rostro y también sería más costosa su clonación. Para dicha problemática se implementó un software de Reconocimiento Facial por medio de un lenguaje de programación utilizando una librería de Open CV/ EmguCV. Donde el software operara de dos formas. La primera se compara una imagen de una persona con otra

imagen de rostro la cual se requiere que sean muy similares para llegar a autentificar y verificación de caras. La segunda forma compara una imagen de una persona desconocida con las imágenes almacenadas en la base de datos para llegar a encontrar la identidad de la persona. Estudiaron 3 métodos de reconocimiento facial donde analizaron el método EigenFaces, FisherFaces y Patrón de Binario Local.

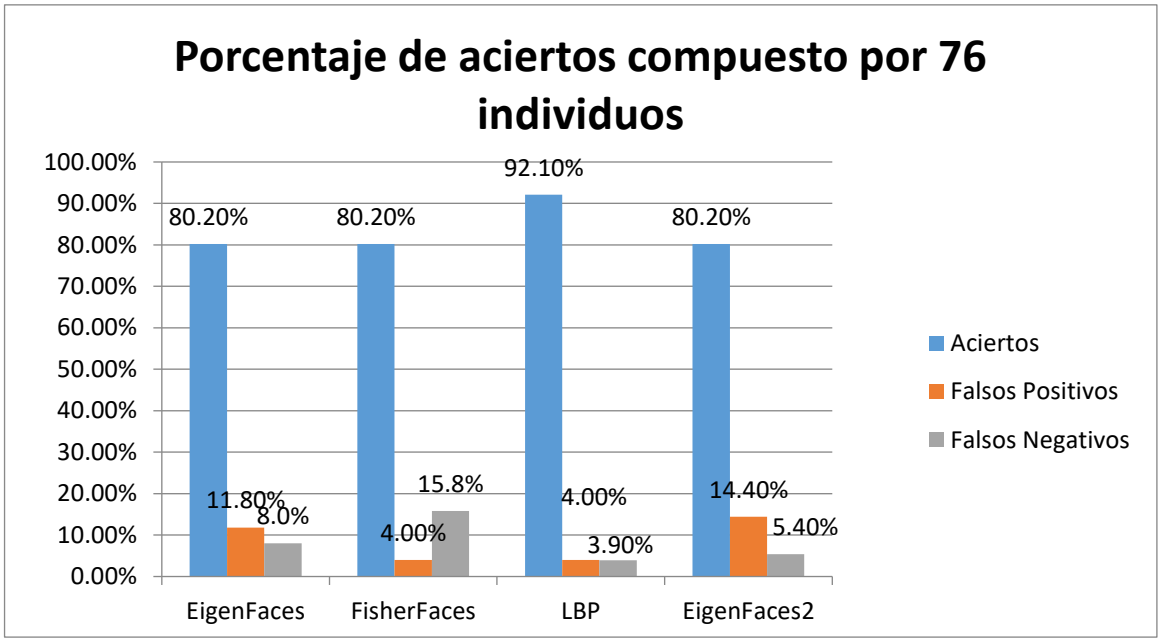


Figura 2: Porcentaje de aciertos compuesto por 76 individuos

Fuente: David Espinoza, Jorquera Guillen (2015). Resultados Estudio N°1. Generación 2012: 76 personas.

Teniendo como resultado en el 1° estudio compuesto de 76 imágenes de individuos distintos con tamaño de la foto 100x100 pixeles entrenada desde una foto tamaño carnet de 100x125 pixeles y todos los rostros tienen la misma expresión y están situados de manera frontal además que no hay cambio de entorno, como resultado para el método EigenFaces tiene el 80.2% de aciertos en el reconocimiento, Falsos Positivos de 11.8% y Falsos Negativos del 8%; para el otro método FisherFaces tiene el 80.2% de aciertos, Falsos negativos del 4% y Falsos Negativos del 15.8%; Y para el método LBP tuvo el 92.1% de aciertos, Falsos positivos de 4% y Falsos Negativos de 3.9%.

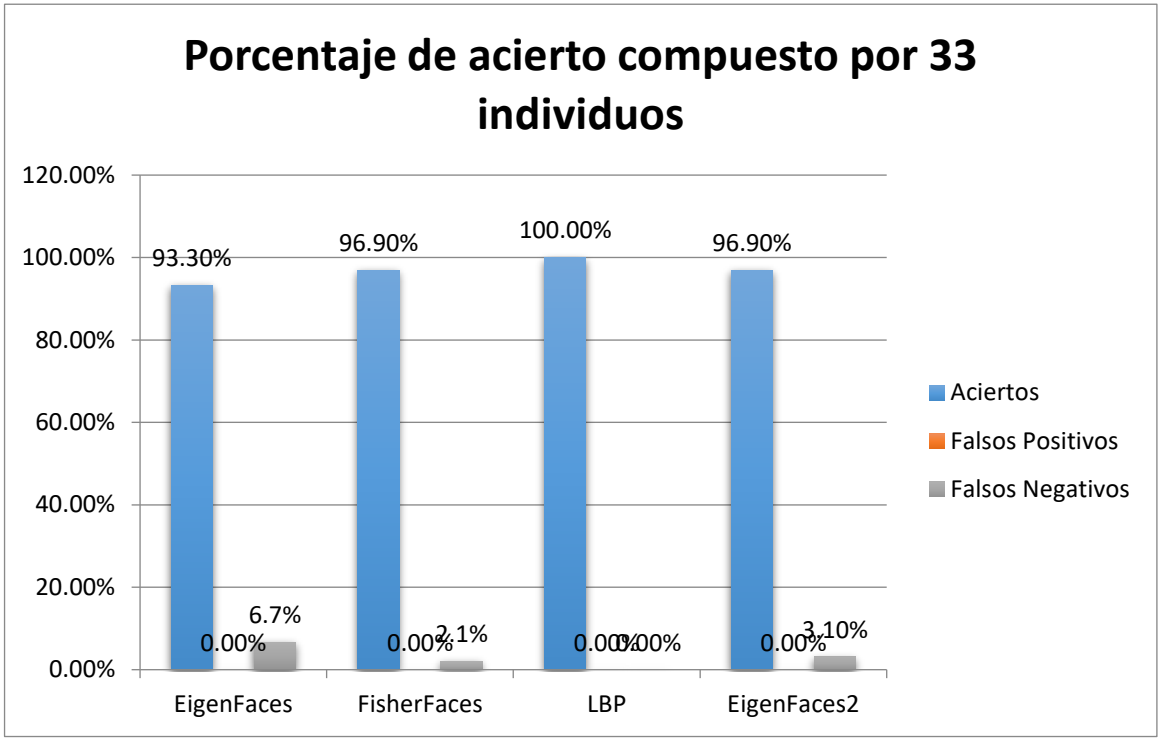


Figura 3: Porcentaje de acierto compuesto por 33 individuos

Fuente: David Espinoza, Jorquera Guillen (2015). Resultados Estudio N°2. Generación 2013: 33 personas.

En el 2° estudio está compuesto de 33 clases. Cada clase tiene 3 imágenes. La imagen de entrada en la mayoría de los casos fue una cuarta imagen distinta a la almacenada en la carpeta de entrenamiento, como resultado del estudio para el método de EigenFaces tiene el 93.3% de aciertos, Falsos Positivos 0% y Falsos negativos del 6.7%; para el método FisherFaces tuvo el 96.9% de aciertos, Falsos Positivos de 0% y Falsos negativos del 3.1% y para el otro método LBP se obtuvo los mejores resultados de los estudios realizados con el 100% de aciertos, Falsos Positivos de 0% y Falsos Negativos del 0%

Para concluir David Espinoza, Jorquera Guillen (2015) evaluaron los métodos mencionados anteriormente para que más adelantes personas interesadas en el tema de reconocimiento facial evalúen y saquen su propia conclusión de cuál de ellos deben implementar en su software.

En el año 2008 el autor Villa Palacios Sandra en la Universidad Peruana de Ciencias Aplicadas realizo la tesis con el tema “Sistema Automático de Reconocimiento de Rostro”. Tiene el objetivo general diseñar un sistema biométrico capaz de reconocer rostros en tiempo real y con algo nivel de confiabilidad. Para el desarrollo del sistema está conformado por un capturado de imágenes compuesto por una cámara web para la adquisición de imágenes, elaborando su propio ambiente controlado disminuir los problemas e inconvenientes con la iluminación. El prototipo de adquisición de imágenes tiene una base para el rostro, el interior de la base es de color blanco que permite resaltar las características de rostro y disminuir la tasa de error y mejor la segmentación a la hora del proceso de reconocimiento.

El algoritmo utilizado para el sistema se divide en cuatro partes: obtención de la imagen, pre-procesamiento de la imagen, extracción de características aplicando Transformada de Wavelet Discreta Bidimensional y comparación. En la primera parte de obtención de la imagen ya se describió anteriormente describiendo el uso de un ambiente controlado. En la segunda parte, se realiza el procesamiento de las imágenes necesarias (escala de grises a 8bits, filtros, mejora la iluminación) para el realce de las características para estar aptas para la segmentación del rostro. En la tercera parte, segmenta la imagen mediante proyecciones empezando por la aplicación de Wavelet y se repite 15 veces, ya que decidió adquirir 15 fotos las cuales serán almacenadas en la base de datos. Para la tercera parte, se procedió a utilizar el Análisis de Componentes Principales, para extraer las características principales de la imagen del rostro reflejadas en el vector las cuales fueron utilizados para la posterior comparación. Por último, la cuarta parte, se comparan los vectores obtenidos anteriormente con el nuevo vector desconocido adquirido de la imagen desconocida que no pertenece a la base de datos y mediante el cálculo de la distancia entre vectores logre identificar el rostro al cual más se asemeje

Para las pruebas y experimentos se utilizó una base de datos de 50 personas, tomado 15 fotos por cada persona, las cuales en la base de

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

datos almacenaba 750 imágenes con una resolución de 240 x 320 pixeles. La adquisición de las imágenes para la base de datos también fue en ambientes controlados tomadas en el mismo prototipo elaborado, pero con diferentes expresiones como sonreír y otros gestos. El autor no considero tomar pruebas de diferentes ángulos del rostro ya que todos son fotos frontales.

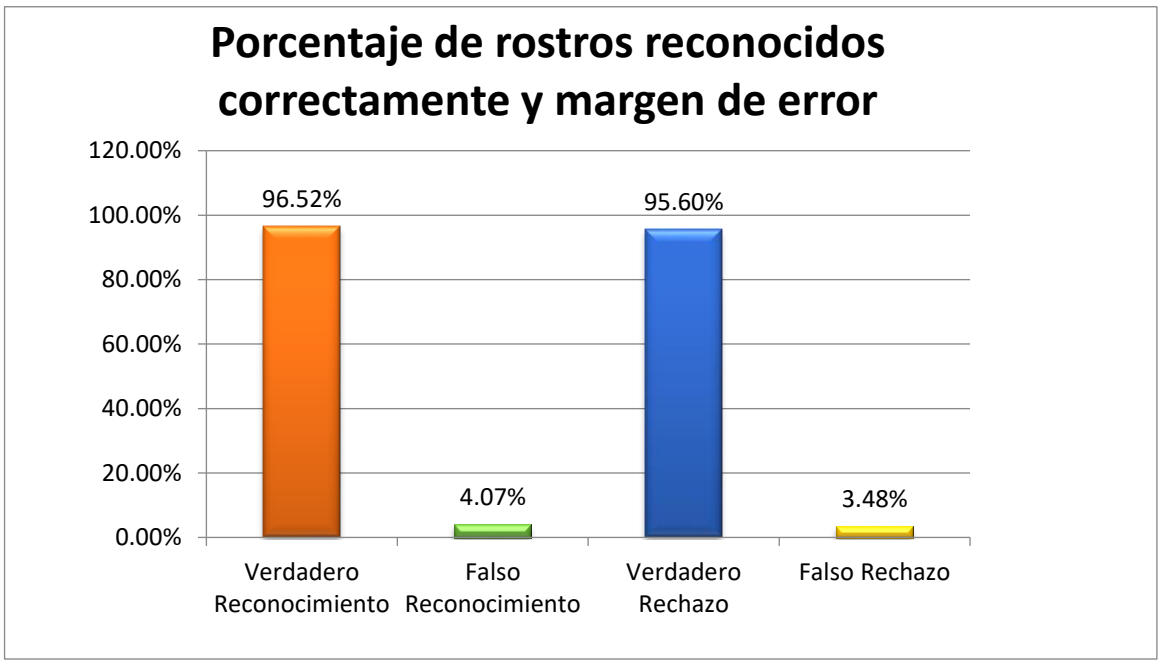


Figura 4: Porcentaje de rostros reconocidos correctamente y margen de error

Fuente: Villa Palacios Sandra (2008). Porcentaje de rostros reconocido correctamente y margen de error, con un umbral de 70.00%.

Se realizaron 3 tipos de pruebas con diferentes expresiones tomadas en tiempo real teniendo registrado previamente 50 rostros y se realizó pruebas con 15 rostros de individuos no registrados, se obtuvo un alto porcentaje de resultados correctos. Los resultados obtenidos para Verdadero Reconocimiento el 96.52%, Falso Reconocimiento el 4.07%, Verdadero Rechazo 95.60% y Falso Rechazo 3.48% con un umbral de 70.00%.

En conclusión, el uso de la Transformada Wavelet y el Análisis de Componentes Principales favorece a disminuir los datos obtenidos

perdiendo la menor cantidad posible de información, y reduciendo la carga computacional.

En el año 2015 el autor Yulian André Cama Castillo, en la Pontificia Universidad Católica del Perú, realizo la tesis con el tema “Prototipo computacional para la detección y clasificación de expresiones faciales mediante extracción de Patrones Binarios Locales”. Donde tiene como problema central las dificultades en el reconocimiento de expresiones automático en las imágenes digitales, siendo el rostro el principal transmisor de las emociones de una persona, la cual se propusieron el objetivo de investigación la implementación de un prototipo computacional de estudio y procesamiento de imágenes faciales que permita caracterizar el rostro de un individuo mediante el método Patrones Binarios Locales(LBP) a fin de reconocer el rostro en expresiones de felicidad, tristeza, miedo, asco, asombro.

Primeramente, en la primera fase primordial de adquisición de imágenes utilizo la librería de OpenCV para lograr mapear las imágenes en una clase, luego la imagen adquirida pasa por la segunda fase, que es el proceso de pre-procesamiento de la imagen para relevar las partes importantes. En este proceso pasa por dos sub-procesos que es la transformación de escala de grises, que ayuda a resaltar la luminosidad, la cual para la transformación también utilizaron la librería OpenCV ya que viene con dicha función. Luego el segundo sub-proceso es la ecualización de histograma, que estandariza la intensidad de pixeles mejorando la calidad de la imagen.

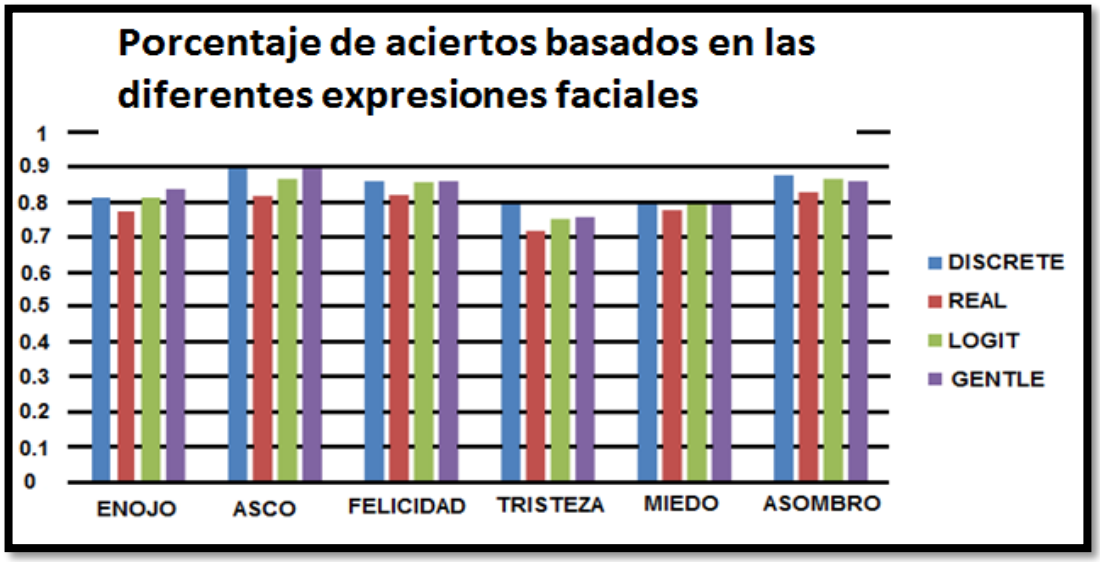
Luego en la tercera fase, pasaron al proceso de detección de rostro que es la base de la información para su posterior análisis y clasificación del rostro. Para este proceso utilizo el método propuesto por Viola-Jones. Este método utiliza la información de la imagen escala de grises y no requiere de información de la imagen a color y junto con el algoritmo AdaBoost permite que la tasa de detección positiva sea elevada con un costo computacional muy bajo.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Después de realizar el proceso de detección de rostro paso a la cuarta fase, donde se realizó la extracción de características de la cara con un proceso previo que es el recorte de las áreas que son irrelevantes como las orejas y el pelo, para finalmente pasar al proceso previo de escalado que consiste en la estandarización de la imagen que es la resolución para que cumpla las mismas características que la imagen de entrenamiento y así mejoro la tasa en el proceso de clasificación.

Una vez que se realizó los procesos previos con la imagen ya estandarizada y alineado, implemento el algoritmo Patrones Binarios Locales y un vector de baja dimensionalidad. Esto hace que el algoritmo transforme la imagen más rápido y mejore a la reducción del costo computacional.

Una vez que realizo la extracción de características paso a la fase cinco con el modelo de clasificación de expresiones. En este proceso se utilizaron la base de datos de imágenes Cohn-Kanade (CK) donde contiene 123 sujetos con imágenes de alta variedad de expresiones faciales. Dichas imágenes lo almacenaron en diferentes carpetas según las expresiones faciales. Para el modelo de clasificación utilizaron el método de aprendizaje por Boosting que es una forma más iterativa de clasificadores débiles donde se logró una clasificación más robusta y acertada.



Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Figura 5: Porcentaje de aciertos basados en las diferentes expresiones faciales

Fuente: Yulian André Cama Castillo (2015). Tasas promedio de reconocimiento por expresión tras variar el tipo de Boosting usado en el prototipo.

Finalmente, paso al proceso de experimentos. Para la primera prueba con variación del tipo de Boosting se alcanzó una precisión del 89.71% en la expresión de asco en las variantes Discrete Y Gentle, también se analizó los experimentos basado en LBP con un radio de 3 logrando superar con un máximo de 93.14% en expresiones de asombro y un mínimo de 83.14% en tristeza al LBP con radio 4.

En conclusión, llegaron que el método de Viola-Jones no toleraba rotaciones de 15 grados como máximo ya que si se requiere implementar un sistema se tendrá que mejorar el método de detección de rostro con un método más robusto en la rotación de la cara; también observaron que las variaciones más relevantes se dieron al ajustar el número de divisiones en la imagen LBP.

Segun Pratibha Sukhija, Sunny Behal, Pritpal Singh (2016), en su investigación realizo la comparación del algoritmo PCA con el algoritmo LDA.

El método PCA obtiene una tasa de reconocimiento para la base de datos del 97.14% siendo un índice alto, para la siguiente base de datos los valores son bajos teniendo el mismo número de clases el porcentaje más alto es de 84.16 considerando que el algoritmos PCA no es muy robusto para cualquier tipo de base de datos.

Tabla 1: Tasa de reconocimiento facial usando PCA en diferentes base de datos

Método	Base de datos	Numero de clases	Numero de clases Testeadas	Tasa de reconocimiento
PCA	ORL	10	3	97.14%
		20	4	91.44%
		30	3	87.60%
		40	2	81.87%
	UMIST	10	2	73.75%
		20	2	70.62%
		20	4	84.16%
	Indbase	10	3	58.71%
		10	4	60.00%
		10	5	70.00%

Fuente: Pratibha Sukhija, Sunny Behal, Pritpal Singh (2016). Tasa de reconocimiento facial usando PCA en diferentes bases de datos

El algoritmo LDA en las diferentes bases de datos a obtenido porcentajes muy deficientes por lo que el más alto con 76.25%, el cual no es muy recomendable por su gran tasa de error.

Tabla 2: Tasa de reconocimiento facial usando LDA de todas las bases de datos.

Método	Base de datos	Numero de clases	Numero de clases Testeadas	Tasa de reconocimiento
LDA	ORL	10	2	76.25%
		20	4	55.00%
		30	4	42.70%

		40	2	36.25%
	UMIST	10	2	28.75%
		20	2	63.12%
		20	4	60.83%
	Indbase	10	2	52.50%
		10	3	70.00%
		10	4	33.33%

Fuente: Pratibha Sukhija, Sunny Behal, Pritpal Singh (2016). Tasa de reconocimiento facial usando el método LDA de todas las bases de datos disponibles.

Segun Shivakumar Dalali, Suresh L. (2016), realizó una serie de pruebas al algoritmo implementado LBP, cambiando el número de vecindad y tamaño de los bloques. Teniendo como resultado casi perfecto con el 99.30% con el umbral más alto. Teniendo como muestra imágenes de rostros con diferentes orientaciones.

Tabla 3: Tasa de reconocimiento de imágenes sin ruido utilizando subbloques que no se superponen

Tamaño de bloque secundario no superpuesto en la imagen de entrada	**Tamaño de una nueva matriz para el LBP modificado.**	**Métodos modificados de LBP.**	**Valor de umbral (rangos).**	**% de la tasa de reconocimiento.**
2x2	26x21	P=8; R=4,3,2,1; Tamaño de bloque=10X13	50-60	99.30 %

3x3	17x14	P=8; R=4,3,2,1; Tamaño de bloque=10X13	20-30	99.11 %
4x4	13x11	P=8; R=3,2,1; Tamaño de bloque=7X10	10-20	99.01 %
5x5	10x8	P=8; R=2,1; Tamaño de bloque=6X8	10-20	99.00 %

Fuente: Shivakumar Dalali, Suresh L. (2016). Tasa de reconocimiento de imágenes sin ruido utilizando subbloques que no se superponen.

Las pruebas realizadas para el segundo experimento, se utilizó los mismos valores del caso anterior, teniendo como varianza que las imágenes de muestra contienen ruido, el cual realizó pruebas obteniendo buen resultado a pesar que las imágenes están movidas. Teniendo como resultado eficiente del 98.28 % de la tasa de reconocimiento.

Tabla 4: Tasa de reconocimiento de imágenes con ruido utilizando subbloques no superpuestos.

Tamaño de bloque secundario no superpuesto en la imagen de entrada	**Tamaño de una nueva matriz para el LBP modificado.**	**Métodos modificados de LBP.**	**Valor de umbral (rangos).**	**% de la tasa de reconocimiento .**
2x2	26x21	P=8; R=4,3,2,1; Tamaño de bloque=10X13	50-60	98.28%
3x3	17x14	P=8; R=4,3,2,1; Tamaño de	20-30	98.13%

		bloque=10X13		
4x4	13x11	P=8; R=3,2,1; Tamaño de bloque=7X10	10-20	98.05%
5x5	10x8	P=8; R=2,1; Tamaño de bloque=6X8	10-20	97.82%

Fuente: Shivakumar Dalali, Suresh L. (2016). Tasa de reconocimiento de imágenes con ruido utilizando subbloques no superpuestos.

Según Vázquez Míguez, Javier (2014), realizo en su investigación de los algoritmos de reconocimiento facial, el cual implemento tres algoritmos para comparar teniendo como resultado LBPH demostró ser el algoritmo con mejor porcentaje de acierto con el 97.50 %.

Tabla 5: Computación media de aciertos y tiempo entre algoritmos.

Algoritmo	**Media de aciertos**	**Tiempo necesario**
Eigenfaces	87.50%	5.628 s
Fisherfaces	92.50%	7.515 s
LBPH	97.50%	1.319 s

Fuente: Vázquez Míguez, Javier (2014). Comparación media de aciertos y de tiempo entre algoritmos

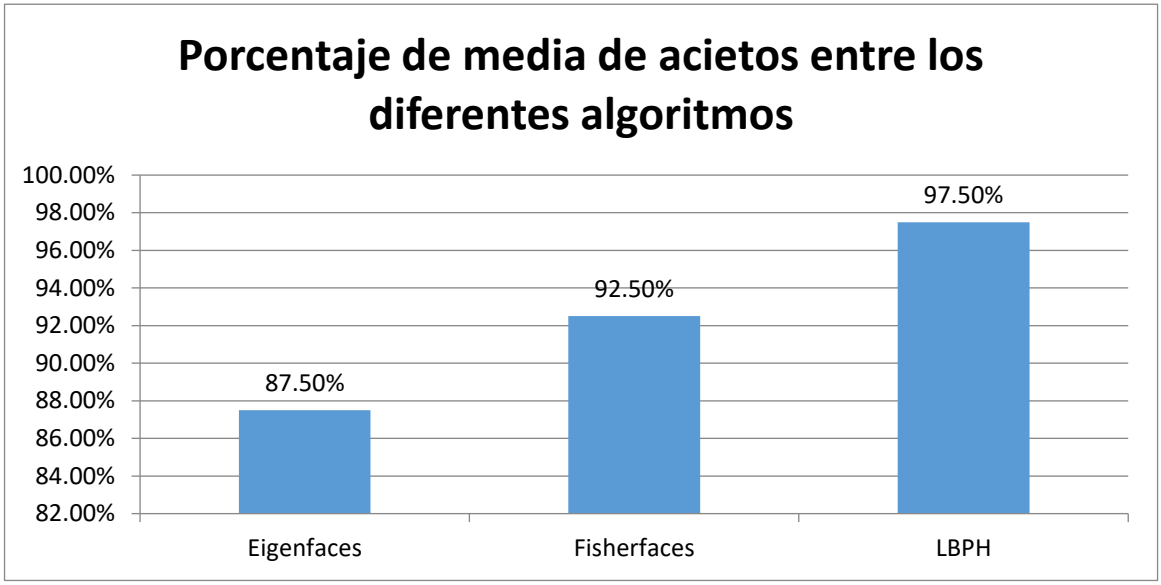


Figura 6: Porcentaje de acierto entre los diferentes algoritmos.

Fuente: Vázquez Míguez, Javier (2014). Comparación de los resultados medios de los tres algoritmos.

Sin embargo, también es la técnica más rápida tomando 1.319 segundos para su reconocimiento teniendo la gran diferencia con los otros algoritmos implementados.

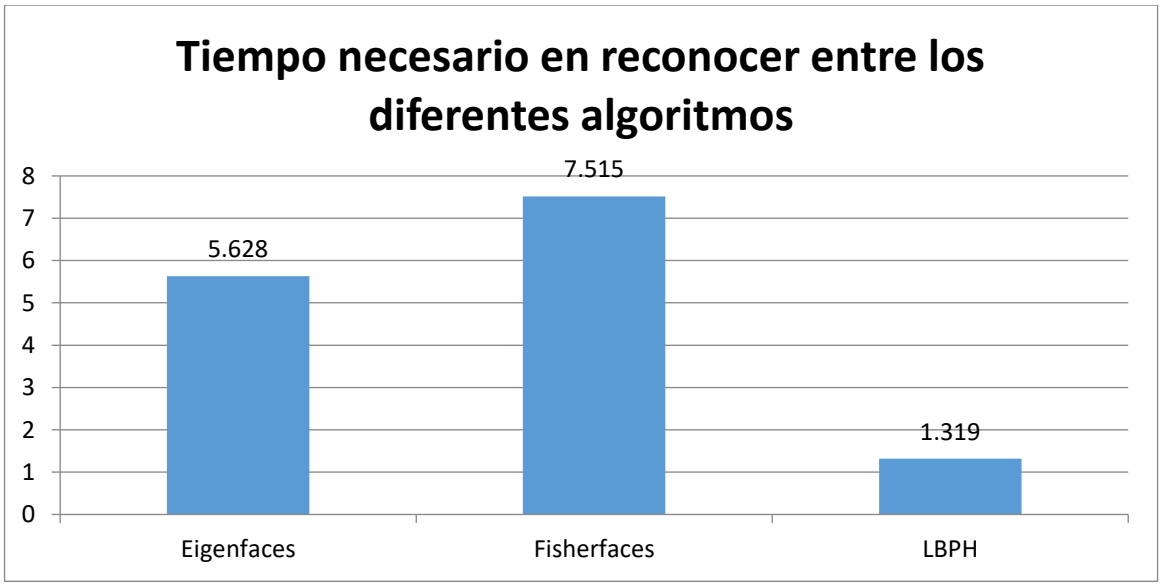


Figura 7: Comparación del tiempo medio en cada algoritmo.

Fuente: Vázquez Míguez, Javier (2014). Comparación del tiempo medio en cada algoritmo.

2.2 Estado del arte

En el año 2012 el autor Faten Bellakhdhar, Kais Loukil, Mohamed en la revista Journal Of Intelligent Computing, publico el artículo científico “SVM Classification for Face Recognition” Con respecto a la biometría facial existen varios métodos para realizar el procedimiento de reconocimiento de rostro, sin embargo, en las investigaciones realizadas existe algoritmos muy eficaces a la hora de procesar la imagen, pero ocurre el inconveniente en la etapa de la clasificación del rostro ya que existen una serie de variaciones en la imagen procesada, como es por el Angulo de la cara, sería un problema a la hora de la clasificación si ha existido una serie de variaciones de la orientación del rostro y como consecuencia trayendo problemas en el rendimiento de la aplicación, por eso Faten Bellakhdhar (2012) realizo un estudio centrándose en la clasificación para el reconocimiento facial, extrayendo primero las características de Wavelets de Gabor seguida por la fusión de características 2DPCA para la extracción de los puntos importantes y como clasificador SVM que se utilizó mediante la configuración de un conjunto de vectores de apoyo. Y como resultado se encontró un margen de error en comparación de rendimiento del 0.07% para el primer protocolo P1 y para el segundo protocolo P2 que es la evaluación de prueba de rendimiento tuvo un margen de error de 0.15%, como vemos el uso del clasificador SVM para la clasificación de caras tiene una gran influencia en el rendimiento de la aplicación y mejoramiento de la tasa de error.

En el año 2013 el autor Yong Lee & Cheong Ghil & Youngseop Kim & Taeg Keun Whangbo en la revista “Springer Science+Business Media New York” publicaron el artículo científico con el tema “Facial Landmarks Detection Using Improved Active Shape Model on Android Platform”. Donde utilizaron el método ASM y en base al presente algoritmo propusieron 3 mejoras para la extracción de puntos faciales bajo el sistema operativo Android. Sus mejoras fueron relacionadas, con la iniciación del algoritmo usando el

centro de los ojos, su segunda mejora fue obtener los 76 puntos faciales para formar el rostro facial y su tercera mejora fue ampliar el perfil unidimensional al perfil dimensional para la localización de la posición de cada punto facial.

Tabla 6: Porcentaje de tasa de reconocimiento de los dos métodos ASM.

Método	**Tasa de éxito**			**Éxito promedio proporción**	**Extracción promedio tiempo**
	FEI DB	JAFFE DB	Our DB		
ASM CLASICO	83.0% (166/200)	63.7 % (137/215)	84.5 % (279/330)	78.1 % (582/745)	1.612 s.
ASM PROPUESTO	93.5% (187/200)	71.6 % (154/215)	96.1 % (317/330)	88.3 % (658/745)	1.195 s.

Fuente: Yong-Hwan Lee & Cheong Ghil Kim & Youngseop Kim & Taeg Keun Whangbo (2013). Comparación de rendimiento y porcentaje de tasa de éxito de los dos métodos ASM.

El algoritmo propuesta en la base de datos de Our BD ha obtenido un porcentaje bueno del 96.10%, dicha base de datos ha sido creada de internet con la recolección de fotos de rostros mediante Google.

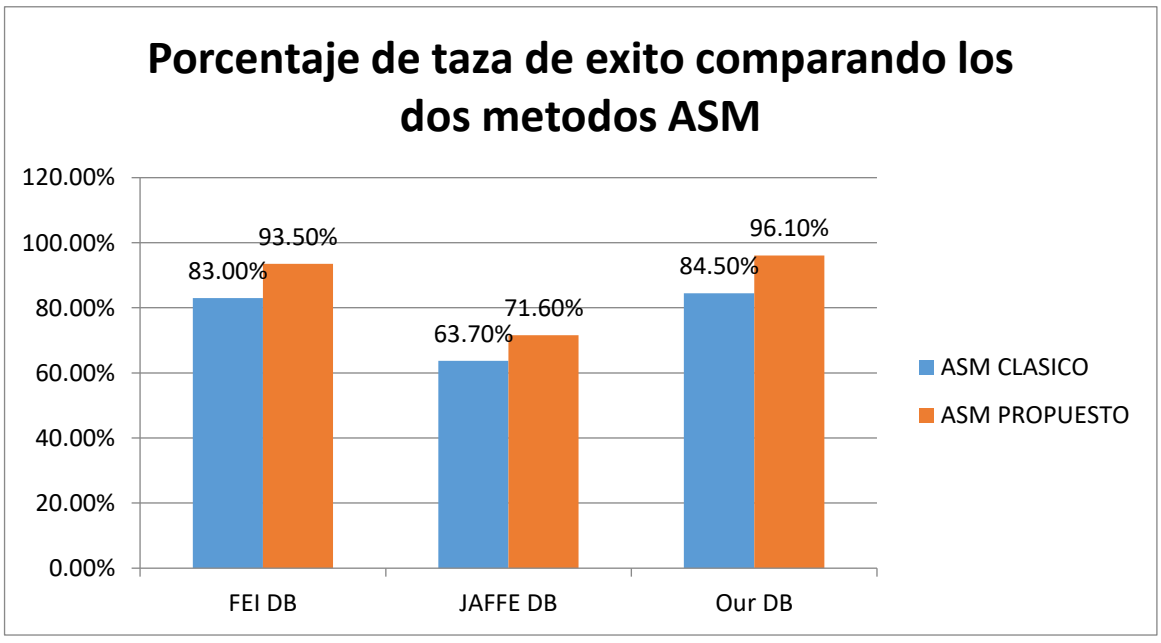


Figura 8: Porcentaje de tasa de exito comparando los dos metodos ASM.

Fuente: Yong-Hwan Lee & Cheong Ghil Kim & Youngseop Kim & Taeg Keun Whangbo (2013). Comparación del Algoritmos ASM clásico y ASM propuesto.

Con dichas mejoras en el método obtuvieron un resultado relevante del 10.2% mayor en precisión al modelo clásico ASM, y también mejoro el tiempo computacional en extracción con 0.417 segundos en promedio.

Asimismo, en la publicación del artículo de investigación en el año 2015, Akariman, Jati, Novianty, en la "Universidad De Telkom", con el tema "Face Recognition Based on the Android Device Using LBP Algorithm" en dicho artículo implemento el algoritmo Patrón Binario Local en dispositivos móviles para medir la exactitud del método en reconocimiento de imágenes con variación de luz, variación de distancia y variación del ángulo de la cara. De entrada, realizaron el proceso de entrenamiento de la cara iniciando en la detección del rostro para luego pasar al pre-procesamiento de la imagen cambiando de tamaño a 128 x 128 pixeles y transformando la imagen RGB a escala de grises, continuando con la extracción de características con el algoritmo LBP compara que el pixel que se encuentra en la imagen central con un valor de pixel vecino, los pixeles se comparan utilizando 3x3 vecinos, donde el centro se compara con 8 valores de pixeles circulantes donde si es menor cambia a 0 y si el pixel es mayor o igual cambia a 1 para posteriormente tener un vector para clasificarlos.

Como resultados de las pruebas alcanzo en este caso en diferentes escenarios con variación de iluminación, teniendo como resultado en la mañana el 90% de aciertos, por la tarde obtuvo el 80% y en la noche el 50%.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

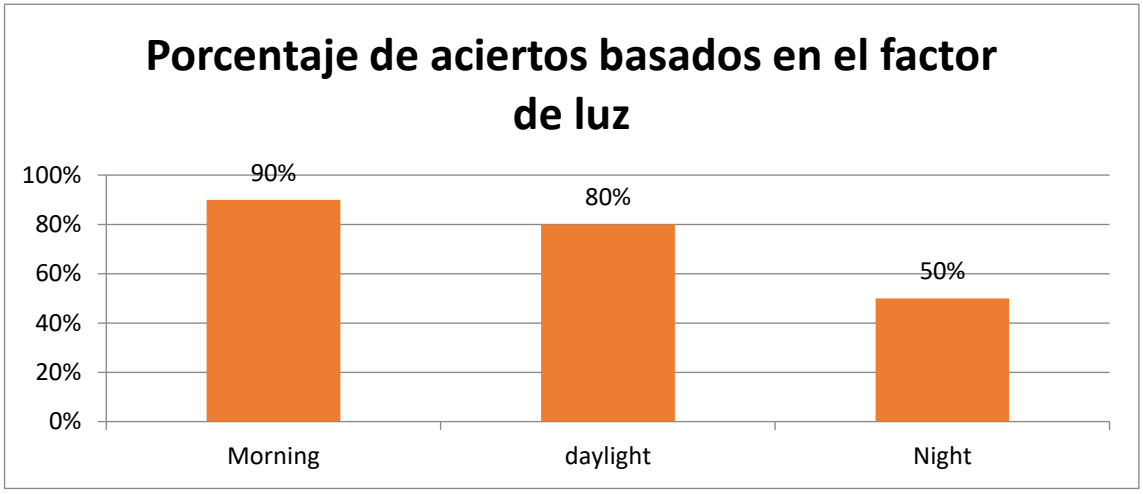


Figura 9: Porcentaje de acierto basados en el factor luz de la mañana, tarde y noche.

Fuente: Akariman et al., (2015). Pruebas basadas en factor de luz.

El investigador también realizo pruebas a diferentes distancias, teniendo buenos resultados para 25cm y 50 cm con el 90% de reconocimiento y el 80% de reconocimiento a 100 cm donde indica que se podría mejorar teniendo un dispositivo de más capacidad.

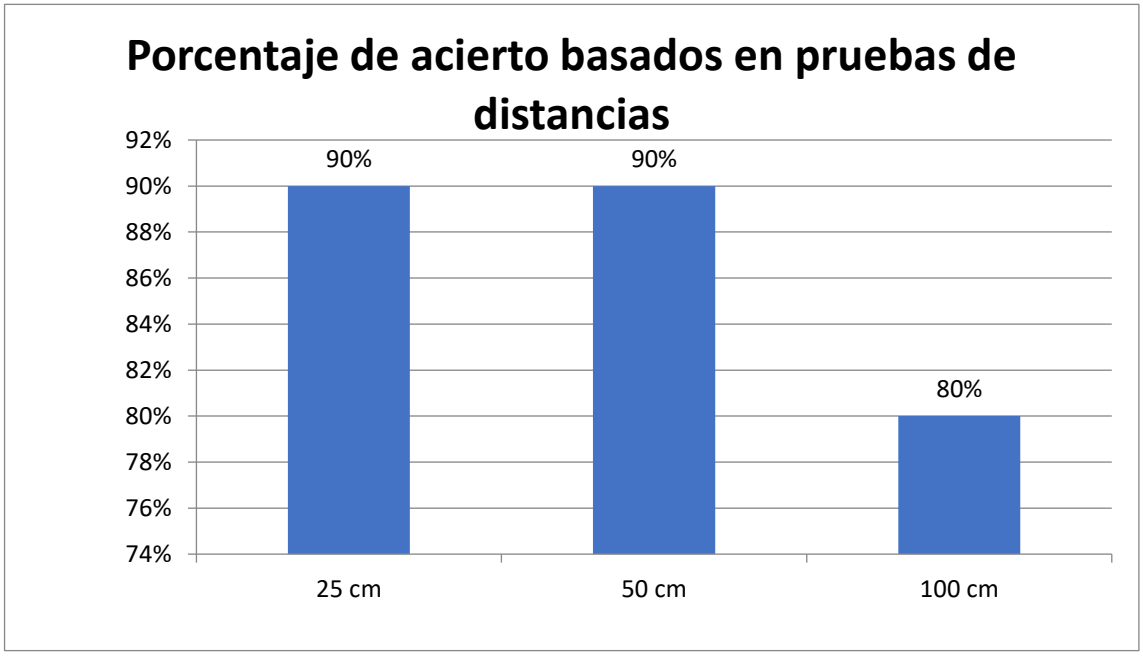


Figura 10: Porcentaje de acierto basados en pruebas de distancia con 25, 50 y 100 centímetros.

Fuente: Akariman et al., (2015). Pruebas basadas en la distancia.

Por último, realizo pruebas con orientación de rostro en forma horizontal, parte importante para esta investigación, donde para 0° y 30° sigue teniendo el 90 % de aciertos, indicando en la fuente que a partir de los 60° se empieza a perder el número de aciertos y a los 90° que es de perfil no logra tener ningún porcentaje de acierto. El cual muestra que el algoritmo LBP es robusto hasta los 30° sin pérdida de aciertos.

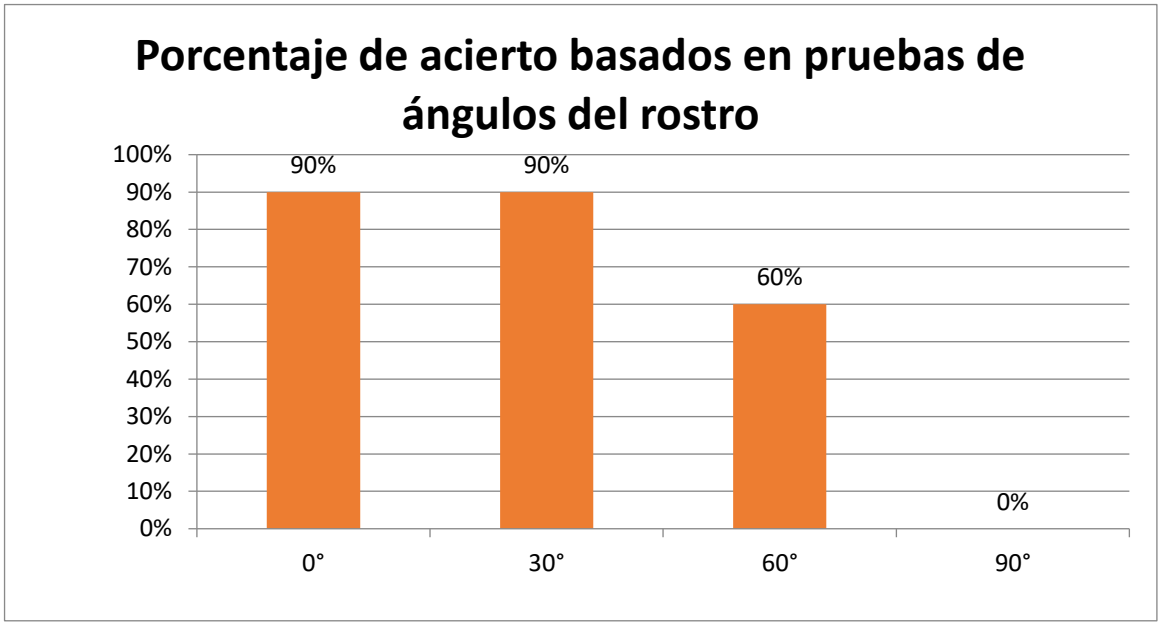


Figura 11: Porcentaje de acierto basados en pruebas de ángulo 0°, 30°, 60° y 90 °

Fuente: Akariman et al., (2015). Pruebas basadas en ángulo.

En el año 2016, Stoimenov, Georgi T. Tsenov, Valeri M. Mladenov en "Symposium on Neural Networks and Applications (Neurel)" publicaron su artículo de investigación con el tema "Face Recognition System in Android Using Neural Networks" en dicho artículo Implementaron una red neuronal con java en el entorno de teléfonos móviles Android, con la finalidad de reconocer rostros de fotos o fotogramas de videos. Siguiendo el procedimiento, extrajeron los datos de imagen sin procesar, luego realizaron filtrado de imágenes y transformar en escala de grises, continuando con la extracción de características específicas de la imagen facial "Google Mobile Visión API" y finalmente clasificaron las características extraídas anteriormente y relacionando la imagen con la

mejor clase de imagen de un conjunto de imágenes conocidos y representa una salida. Teniendo como experimentos de los estudios probaron con 15 imágenes, 5 para cada clase de persona, al tener los primeros 5 de la persona 1 que resulta a 1 0 0 salida, 5 para la persona dos con 0 1 0 salida y el final 5 para la persona 3 con resultado 0 0 1 salida. Los resultados son muy cercanos a las clases correctas como se puede ver cuando las salidas son las que estaban muy cerca de uno y el que debía ser cero eran muy cercanos a cero. Al redondear las salidas obtenemos resultados de clasificación correctos, lo que significa que el error es muy pequeño.

En el año 2016, Gofman, Mitra en “Communications Of The Acm” publicaron su artículo de investigación con el tema “Multimodal Biometrics For Enhanced Mobile Device Security” en dicho artículo implementaron un sistema biométrico multimodal en dispositivos inteligentes como ven que cuentan con cámara y micrófono facilitaba la captura de rasgos humanos, donde su sistema desarrollado lo llamaron Proteus basado en la cara y voz integrando nuevos algoritmos de autentificación biométrica multimodal optimizados para dispositivos móviles y una interfaz que permite al usuario registrar fácilmente sus rasgos faciales. Para la autentificación multimodal evaluaron la calidad de la muestra de la cara y de la voz. Proteus evalúa la calidad de la imagen facial basado en la luminosidad, nitidez y contraste mientras que la voz se evalúa por la grabación de relación señal-ruido (SNR). Proteus adapta los enfoques probabilísticos descrito en Vondrasek y Pollak para estimar la voz del ruido. La cara se detecta a través del algoritmo de Viola-Jones y la propia aplicación extrae la banda sonora. Como resultados obtenidos de fusión de nivel de puntuación es para cara el Equal Error Rate (EER) es el 21.17% con un tiempo de prueba de 0.065 sec. Para voz el EER es el 41.44% con un tiempo de prueba de 0.045 sec. Para fusión nivel de puntuación el EER es el 25.70% con tiempo de prueba de 0.108 segundos.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

2.3 Base teórica científicas

2.3.1 Inteligencia Artificial

Según Alberto García Serrano (2012) en el libro “Inteligencia Artificial: Fundamentos, practica y aplicaciones” nos dice que el aprendizaje automático es también una de las condiciones para que un ente sea considerado artificial. Si una maquina no puede aprender nuevas cosas, difícilmente será capaz de adecuar a los medios, la cual es exigible para cualquier ser dotado de inteligencia. La inteligencia artificial, son las líneas actuales de la investigación, que buscan que las maquinas sean capaces de hacer generalización a partir de un ejemplo sacado del medio. Por ejemplo, un niño en edad muy temprana aprende que si se cae al suelo se hace daño entonces para llegar a la generalización, antes tienes que caerse al suelo varias veces, eso se considera ejemplo de aprendizaje en Inteligencia Artificial, y a través de las redes neuronales trata de simular el cerebro humano.

Por otro lado, según John Haugeland (1988) en su libro La inteligencia artificial nos comenta que la inteligencia artificial primero debe tratar de entender el conocimiento y la destreza y todo lo que pueda adquirir, y después sobre esa base tratar de desenvolver su aprendizaje. Desde luego que es fundamental tener facilidad de aprendizaje para que logre tener un aprendizaje completo.

2.3.2 Imágenes Digitales

Según Passariello, G., & Mora, F. (1995). En su libro Imágenes médicas define que una imagen digital es la representación bidimensional de una imagen real, formada por un conjunto de puntos definidos denominado pixeles donde es la parte más pequeña que puede ser subdividida una imagen.

Según Hugo Rodríguez Alonso (2009) en su libro Imagen Digital: Conceptos básicos nos comenta que si se aplica zoom sobre la imagen

observaremos que está compuesto por una matriz de puntos o pixeles, donde cada matriz está compuesta por 0 y 1. Donde cada pixel guarda información de color pero en lenguaje máquina. Las imágenes o fotos digitales son guardadas en distintos formatos. Cada formato tiene una extensión diferente que especifica lo que contiene, las cuales más utilizados hoy en día son: BMP, GIF, JPG, TIF, PNG.

2.3.3 Procesamiento de imágenes digitales

Según José Jaime Esqueda Elizondo, Luis Enrique Palafox Maestre (2005) en su libro “Fundamentos de procesamiento de imágenes” nos dice que el procesamiento de imágenes digitales aparece en el instante en que se dispone de recursos tecnológicos para captar y manejar grandes cantidades de información en forma de valores para mejorar y resaltar las características principales de la imagen.

2.3.4 Biometría

Según Ortega García, J., Alonso Fernández, F. and Coomonte Belmonte, R. (2008). En su libro “Biometría Y Seguridad” indica que los dispositivos biométricos es un reconocedor de patrones que captura datos biométricos de una persona, extrayendo un grupo de características a partir de esos datos se compara con información ya almacenados anteriormente para lograr la identificación. Donde los sistemas biométricos pueden operar de tres formas distintas: registro identificación y verificación

La biometría es un proceso automático que realiza la identificación y verificación de un individuo, basándose en las características físicas y de comportamiento. Esta tecnología de reconocimiento facial es más difícil de falsificar ya que cada persona posee rasgos humanos distintos a otras como: voz, rostro, huellas dactilares, iris de ojos, retina, entre otros (redyseguridad.fi-p.unam.mx, 2011).

Según F. Serratosa (2016) en el libro “La Biometría para la Identificación de las Personas” nos dice que es una ciencia que estudia las diferentes

características y posiciones entre las partes del cuerpo para poder identificar o clasificar un individuo. En la actualidad existen varios rasgos biométricos tales como huella dactilar, reconocimiento de rostro, el iris, la mano o la firma.

2.3.5 Reconocimiento facial

Según J. Phillips, P. Grother, J. Micheals, Duane M., E. Tabassi, M. Bone (2003) en el libro “Face Recognition Vendor Test 2002” nos dice que el reconocimiento facial abarca numerosos campos y disciplinas que ahora son objeto de la investigación además de tener numerosas aplicaciones prácticas, es un comportamiento humano esencial para la comunicación entre personas.

Según Stan Z. Li, Anil K. Jain (2011) en el libro “Handbook of Face Recognition”, Donde nos menciona que el reconocimiento facial es un trabajo que lo seres humanos realizamos a diario. Viendo la amplia gama de disponibilidad de sistemas de escritorio a un bajo costo ha surgido el interés por el procesamiento de las imágenes digitales para los distintos objetivos como la seguridad pública, la necesidad de identificar para el acceso físico y lógico y la necesidad de análisis de cara para la gestión de datos multimedia.

Un sistema de reconocimiento facial puede trabajar en uno o ambos modos: el primer modo puede trabajar en la verificación de la cara o autentificación que implica la coincidencia de una imagen que ha sido comparada con otra imagen almacenada en una base de datos y el segundo modo que puede trabajar es en la identificación de la cara o reconocimiento del rostro especialmente cuando el número de rasgos es muy elevado.

2.3.6 Métodos Principales de reconocimiento facial

En los artículos de investigación analizados para el estado del arte se encontraron métodos que describimos a continuación:

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

2.3.6.1 Patrones binarios locales

Según Brahnam, S., Jain, L. C., Nanni, L., & Lumini, A. (2014). En su libro “Local Binary Patterns: New Variants and Applications”, menciona que en el dominio de 3 x 3 el algoritmo patrones binarios locales se limita a 8 pixeles vecinos al valor del pixel central. Donde hace un recorrido de forma horaria sobre el pixel central, si el pixel vecino es menor al pixel central su valor cambia a 0, en cambio si el pixel vecino es mayor o igual que su valor cambia a 1. Una descripción más formal del operador LBP se obtiene mediante:

$$f_{LBP}(X) = \sum_{J=0}^{7} b(I_j - I_C)2^j$$

Según Stan Z. Li, Anil K. Jain (2011). En su libro “Handbook of Face Recognition”, nos indica que una vez obtenida la imagen marcada LBP y el histograma LBP se define como:

$$H_I = \sum_{x,y} I\{f_I(x\,, y) = i\}, i = 0, \dots, n-1$$

En la cual n es el número de etiquetas del operador LBP.

2.3.6.1 EigenFaces

Según Stan Z. Li, Anil K. Jain (2011). En su libro “Handbook of Face Recognition”, nos define que el método de reconocimiento facial eigenface se basa fácilmente en las características faciales. Esto lleva a una mejora en el rendimiento del reconocimiento facial por capas que describen los rasgos faciales. Donde cada imagen se transforma en un vector para posteriormente se realice un análisis de componentes principales.

2.3.6.2 Modelo de Forma Activa(ASM)

Emami, S. and Levgen, K. (2012). Mastering OpenCV with Practical Computer Vision Projects. Nos dice que es un modelo estadístico de variación de formas. Con la cual el modelo se forma a través de las combinaciones de variaciones de formas. Con el fin de construir un modelo de la forma del rostro con varios puntos clave que conforman la cara. Hay 76 puntos importantes en el rostro tales como la boca, la nariz, las cejas, el límite de la cara, y los ojos.

2.3.6.2 Análisis de Componentes Principales (PCA)

Según Luis Gómez Chova (2004), nos indica que PCA es una técnica de análisis multivariante, siendo como objetivo adquirir un conjunto de variantes ordenadas según la importancia de las características considerado como una rotación del procedimiento de coordenadas original al nuevo sistema de coordenadas ortogonales.

2.3.7 Redes Neuronales:

Este término de redes neuronales está siendo utilizado para muchas investigaciones, pero en esta investigación lo utilizamos en la fase de clasificación de imágenes faciales la cual existe una serie de definiciones. Por ejemplo:

Las redes neuronales se comportan como las neuronas del cerebro humano que pueden ser entrenadas para cumplir diversas funciones complejas en software de visión artificial, tales como reconocimiento de patrones, extracción de características y filtrados de imágenes entre otros (Eleyan, Demirel, 2007).

Una unidad de proceso de red neuronal está compuesta por capas: la cual encontramos la capa de entrada, la capa oculta es la sumatoria o cálculo de todas las entradas y por último la capa de salida es igual a todas las

conexiones de salida con el mismo valor (Pedro Viñuela & Ines Galvan, 2004).

2.3.8 Máquina de Vectores de Soporte(SVM)

Emami, S. and Levgen, K. (2012). Mastering OpenCV with Practical Computer Vision Projects. Nos menciona que es un algoritmo de reconocimiento de patrones incluido como algoritmos de aprendizaje creado especialmente para la clasificación binaria. Sin embargo, la primera tarea a realizar antes de clasificar tenemos que entrenar a nuestro clasificador. La cual no es un trabajo sencillo porque requiere una gran cantidad de datos para entrenar al sistema, pero no significa que mientras más grande sea la base de datos mejores resultados se obtendrán.

2.3.9 Lenguajes de programación:

2.3.9.1 Python

Según Guido van Rossum (2009) Python es un lenguaje de programación fuerte y fácil de manipular, con una estructura fuerte y de alto nivel con una orientación a la programación orientado a objetos, y la facilidad de reutilizar código de Python ya que te permite guardar en módulos que puedes utilizar en otros programas.

Python cuenta con una librería llamada Python Imaging Librery para el procesamiento de imágenes compartible con muchos formatos de archivos, actualmente está en la versión 1.1.7 compartible con Python 1.5.2(pypi.python.org).

2.3.9.2 Java

Java implementa la tecnología basada en el lenguaje orientado a objetos trabajando con sus datos como objetos y con una interfaz para esos objetos. Soportando tres características resaltantes: polimorfismo, herencia y encapsulación de datos (Erick. G. Coronel Castillo,2010).

2.3.10 Entornos de programación:

2.3.10.1 OpenCV

Emami, S. and Levgen, K. (2012). En su libro Mastering OpenCV with Practical Computer Vision Projects. OpenCv tiene un conjunto de funcionalidades perfeccionado para la detección de objetos, donde los más conocidos son los detectores de características utilizando el detector de caras de Violo-Jones. Pero sin embargo existen distintos métodos detectores de características que hacen únicos.

2.3.10.2 Android studio:

El conocido Android Studio es un entorno de desarrollo integrado, potente editor de código, actualmente con más funcionalidades. La cual existen diferentes funcionalidades de la API Android Visión Móvil (developer.android.com).

La librería FaceDetector.Face la cual proporciona la funcionalidad de detección de rostro y también nos da a conocer el ángulo del rostro con los métodos conocido como pose (*EULER_X*, *EULER_Y* o *EULER_Z*), por otro lado existe un método EyesDistance() donde nos da la distancia entre los ojos y otro de los métodos importantes es el método getMidPoint() donde nos da el punto medio entre los ojos (developers.google.com).

2.4 Definición de los términos básicos

Imagen. Es la representación de un elemento o cosa que se puede percibir con la vista. La cual está compuesta por una variedad de colores donde nos da una simulación de la realidad.

Pixel. Es la unidad más pequeña de una imagen digital. Cada pixel es una unidad homogénea, y con el conjunto de pixeles con diferentes variedades de color dan como resultado una imagen.

Recursos computacionales. Es cualquier componente físico, que conectado a un sistema informático con disponibilidad limitada. Es la parte interna o parte hardware donde se está ejecutando un sistema.

Procesamiento de imágenes. Es una técnica para el mejoramiento de los aspectos de las imágenes o fotografías, haciendo resaltar la parte más importante o mejorando la calidad de la imagen digital.

Ambiente no controlado. Es un área donde las condiciones no son estables y no se puede manipular las intensidades de iluminación.

Método. Son procesos formales que forman parte de una rutina o costumbre que el hombre realiza y lo repite inconscientemente.

Algoritmo. Es la descripción más precisa de los caminos que llevan a la solución de un problema planteado. La descripción de un algoritmo afecta en la entrada, proceso y salida. Por ese motivo que se compara con una función matemática.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

CAPÍTULO III: MARCO METODOLÓGICO

3.1 Tipo y diseño de investigación

Tipo de investigación

Investigación cuantitativa, debido a que en el presente proyecto se usaron indicadores que brindan información con un nivel de medida, que se utilizó de manera estadística y realizar comparaciones de resultados según los experimentos realizados para dar solución a problemas reales.

Diseño de la investigación

Diseño cuasi-experimental, porque el proyecto se asemeja a una investigación científica, que realizo la manipulación de las variables dependientes, con diseño de pruebas Post-Test.

3.2 Población y muestra

3.2.1 Población

En el presente proyecto de investigación hemos tomado referencia de tesis y artículos analizados para sacar nuestra población y muestra. Se capturo 15 fotos a 50 personas, teniendo como número de individuos a personas que se realizó el proceso de adquisición de imágenes teniendo un total de 750 imágenes que fueron procesadas y almacenadas en la base de datos para su posterior clasificación

3.2.2 Muestra

En la presente investigación se utilizó una muestra de tipo censal, obtenidas en la población, utilizando toda la población de imágenes obtenidas en tiempo real para el entrenamiento y clasificación.

3.3 Hipótesis

Para el reconocimiento facial se diseñó un prototipo en el cual consta de los métodos necesarios para lograr tener el mejor porcentaje de éxito sin descuidar los parámetros relacionados con el procesamiento, como tardar 35 segundos en entregar una respuesta ya que se requiere del resultado en el menor tiempo.

3.4 Variables

Variable dependiente

Rendimiento computacional.

Variable independiente

Método de reconocimiento facial

3.5 Operacionalización

En la siguiente tabla 7 describimos las variables dependientes e independientes incluyendo su unidad de medida.

Tabla 7: ***Variables dependientes e independientes***

Tipo	Variable	Indicadores	Formula
Independientes	Método de reconocimiento facial.	Tiempo promedio que tarda en reconocer a las personas.	$TP = \frac{\sum Tiempo\ x\ prueba}{Total\ de\ pruebas}$
		Porcentaje de aciertos en reconocimiento de personas	Precisión: $P = \frac{\lvert \#TAciertos * 100 \rvert}{\lvert \#Tmuestras \rvert}$

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Dependientes	Rendimiento Computacional.	Promedio total de recurso consumido en memoria RAM.	$PR = \frac{\sum Recurso\ consumido}{Total\ de\ ejecuciones}$

Fuente: Elaboración propia.

3.6 Abordaje metodológico, técnicas e instrumentos de recolección de datos

3.6.1 Abordaje metodológico

El presente proyecto aborda una investigación cuasi-experimental ya que se desea encontrar una solución a la problemática encontrada, realizando varios experimentos a toda la muestra como un solo grupo. Como se mencionó anteriormente la muestra será toda la población para la realización de los experimentos.

3.6.2 Técnicas de recolección de datos

Observación de campo: Mediante esta técnica de observación se emplea en todos los procesos de desarrollo en la tesis en cual presentará en estos puntos:

Verificar los resultados de la implementación de los métodos seleccionados para el reconocimiento facial en entornos no controlados bajo el sistema operativo Android

3.6.3 Instrumentos de recolección de datos

Ficha de Observación

En el desarrollo de la tesis se registrarán los sucesos presentado de la observación en que se registrar en un documento los pasos que se va

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

desarrollar ya que la ficha de la observación está relacionada entre la hipótesis y los hechos reales.

Guía de observación de campo

En el desarrollo de la investigación de tema de tesis se apuntará los incidentes detallando el nivel de avance y resultados preliminares, con el fin de resaltar qué acontecimiento surgió cuando se implementa los métodos definidos en la investigación, en parte del pre-procesamiento, procesamiento y clasificador de las características del rostro.

3.7 Procedimiento para la recolección de datos

Basado a la técnica de recolección de datos voy a detallar las especificaciones:

Basado a la muestra solo escogeré aleatoriamente 15 imágenes de rostros de diferentes personas, ya que la muestra es de tipo censal como se ha especificado y esto son las respectivas características:

Numero de Pixeles de la cámara.

Gama del dispositivo móvil

Lugar de la toma fotográfica.

Tiempo de la toma fotográfica.

Hora en la que fue la toma fotográfica

Tipo de iluminación (Luz solar o luz eléctrica)

Ruidos en la imagen

3.8 Análisis estadístico e interpretación de los datos

Se realiza un procesamiento estadístico descriptiva utilizando la aplicación IBM SPSS, para la secuencia de la presentación, análisis e interpretación

de los resultados finales de la investigación establecido en los indicadores definidos en la operacionalización. En donde detalla los datos obtenidos de los resultados en promedio realizando cuadros estadísticos para mejorar el análisis de los resultados obtenidos.

3.9 Principios éticos

Confidencialidad: Los códigos de ética hacen énfasis ante la seguridad y el resguardo de los participantes como la identidad, como también los informantes de la investigación.

Derechos de Autor: Todo documento e información utilizada para el desarrollo de tema de tesis estará referenciado y citados con sus respectivos autores como participante del trabajo.

3.10 Criterios de rigor científico

Fiabilidad: El proyecto de investigación se realizó el cumplimiento de las expectativas mencionadas en su contenido y se implementara con los estándares de desarrollo.

Consistencia: El proyecto de investigación representará material consistente, fiable y certificado por la comunidad científica.

Validez: Los datos alcanzados por el estudio o proyecto serán revisados y examinados por los ingenieros especialistas en el tema para determinar su autenticidad.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

CAPÍTULO IV: ANÁLISIS E INTERPRETACIÓN DE LOS RESULTADOS

4.1 Resultados en tablas y gráficos

Introducción

En el presente capitulo se muestran las tablas y figuras de los experimentos realizados en el prototipo final y describir los resultados obtenidos de dichos experimentos realizados, teniendo en cuenta que se midieron tanto el porcentaje de precisión como el tiempo que tarda en reconocer a las personas.

Las imágenes entrenadas y experimentales son en ambiente no controlado con variación de iluminación ya descrito en el protocolo de adquisición de imágenes.

Considerando que la base de datos de imágenes de entrenamiento contenía toda la muestra completa de las 50 personas con 15 fotos por individuo. Teniendo un total de 750 imágenes entrenadas en todos los métodos comparados.

4.1.1 Tiempo promedio que tarda en reconocer a las personas.

En este experimento se realizó las pruebas de tiempo durante la mañana, tarde y en la noche, en los diferentes escenarios dados, se calculó el tiempo promedio en que tarde en reconocer, a una distancia de 50 cm y en ambiente no controlado. Para cada escenario se realizó pruebas a las personas de manera frontal.

Se realizaron los mismos tipos de experimentos para la comparación de los diferentes métodos implementados.

Se realizaron pruebas en la mañana, tarde y noche a 15 personas por escenario, para calcular el tiempo promedio que tardó en reconocer. Esto se realizó igual para ambos métodos implementados.

Para la prueba de reconocimiento solo se capturo 1 foto por persona que no se almaceno en el dispositivo y solo se utilizó para el reconocimiento y obtener el tiempo de respuesta.

Tabla 8: ***Tiempos de respuesta en segundos de los métodos implementados.***

	Método LBP	Método EigenFaces
MAÑANA	0.454	0.774
	0.511	0.628
	0.459	0.749
	0.453	0.682
	0.407	0.812
	0.463	0.843
	0.519	0.732
	0.422	0.874
	0.488	0.862
	0.437	0.791
	0.456	0.767
	0.431	0.814
	0.463	0.736
	0.446	0.612
	0.427	0.735
TARDE	0.958	1.193
	0.819	1.060
	0.952	1.025
	0.853	0.839
	0.905	1.164
	0.963	0.896
	0.919	0.978
	1.012	1.177
	0.988	1.127
	0.933	0.852
	0.95	0.943
	0.931	0.990
	1.041	1.040
	0.946	0.990
	0.927	1.032
NOCHE	0.954	0.878

	0.815	0.957
	1.079	1.029
	0.982	0.944
	0.997	1.127
	1.065	1.143
	0.919	1.094
	0.922	1.190
	1.012	1.081
	0.988	1.175
	0.983	1.168
	1.055	1.105
	0.934	1.073
	1.05	1.154
	0.943	1.187

Fuente: Elaboración Propia

Para sacar el tiempo promedio por escenario se aplicó la fórmula siguiente:

$$TP = \frac{\sum Tiempo\ x\ prueba}{Total\ de\ pruebas}$$

El tiempo promedio en reconocer a la persona se muestra en el siguiente en la tabla 9, indicando el tiempo promedio en la mañana, tarde y noche.

Tabla 9: ***Resultados promedio según escenario de prueba***

	METODO 1 LBP	METODO 2 EIGENFACES
MAÑANA	0.456	0.761
TARDE	0.940	1.020
NOCHE	0.980	1.087

Fuente: Elaboración propia.

Para obtener los resultados de pruebas se tuvo en cuenta el tiempo que se tardó por individuo en reconocer y se digitalizo los tiempos de todas las personas que se utilizó para las pruebas.

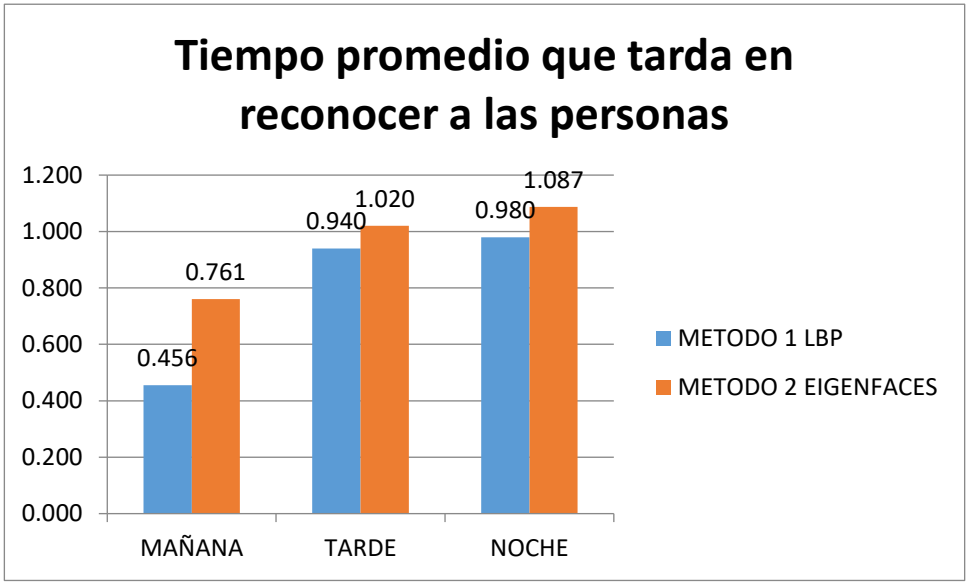


Figura 12: *Tiempo promedio que tarda en reconocer según los escenarios de prueba (mañana, tarde y noche).*

Fuente: Elaboración propia.

Los resultados obtenidos en la Figura 12 para la mañana de 7:00 AM a 9:00 AM en que se realizó las pruebas se obtuvo el tiempo promedio de 0.456 segundos (s) para el primer método LBP, por lo contrario, el segundo método obtuvo 0.761 segundos.

Para la tarde de 1:00 PM a 3:00 PM en que se realizó las pruebas se obtuvo el tiempo promedio de 0.940 segundos y para el segundo método EigenFaces se obtuvo 1.020 segundos demorando más que el primer método.

Para la noche de 7:00 PM a 9:00 PM teniendo el tiempo promedio de 0.980 ms para el primer método LBP y para el segundo método se obtuvo 1.087 segundos.

4.1.2 Porcentaje de aciertos en reconocimiento de personas

El presente experimento se realizó en los diferentes escenarios (mañana, tarde y noche) con la muestra completa y entrenada, las pruebas se

realizaron en los horarios ya descritos anteriormente, dichas pruebas se realizaron en ambiente no controlado.

Para la prueba de reconocimiento solo se capturo 1 foto por persona que no se almaceno en el dispositivo y solo se utilizó para el reconocimiento considerando lo mismo para todos los métodos implementados.

Se tuvieron que utilizar las mismas personas en los mismos escenarios para la comparación de los métodos, se utilizó a 20 personas para la mañana, las mismas 20 para la tarde y las mismas 20 para la noche. Esto se hizo para ambos métodos implementados.

El porcentaje de falsos positivos y falsos negativos se utilizó para lograr alcanzar el 100% de precisión.

Para hallar el porcentaje de precisión se utilizó la siguiente formula:

$$P = \frac{|\#TAciertos * 100|}{|\#Tmuestras|}$$

En la siguiente tabla 10 muestra el porcentaje de aciertos de las 20 personas en la mañana tarde y noche comparada el método LBP y EigenFaces, se realizó pruebas de manera frontal.

Tabla 10: ***Porcentaje de aciertos y error en reconocer a las personas en los 3 escenarios (mañana, tarde y noche) de manera frontal.***

	Método 1 LBP			Método 2 EigenFaces		
Escenario y Orientación	Acierto	Falso Positivo	Falso negativo	Acierto	Falso Positivo	Falso negativo
Mañana 0°	90%	10%	0%	75%	15%	10%
Tarde 0°	85%	10%	5%	75%	15%	10%
Noche 0°	100%	0%	0%	85%	10%	5%

Fuente: Elaboración propia

La Figura 13 muestra de manera comparativa ambos métodos comparados para un mejor análisis visualizando que el método 1 LBP obtuvo el 100%

en la noche a comparación del método 2 EigenFaces que obtuvo el 85% de acierto.

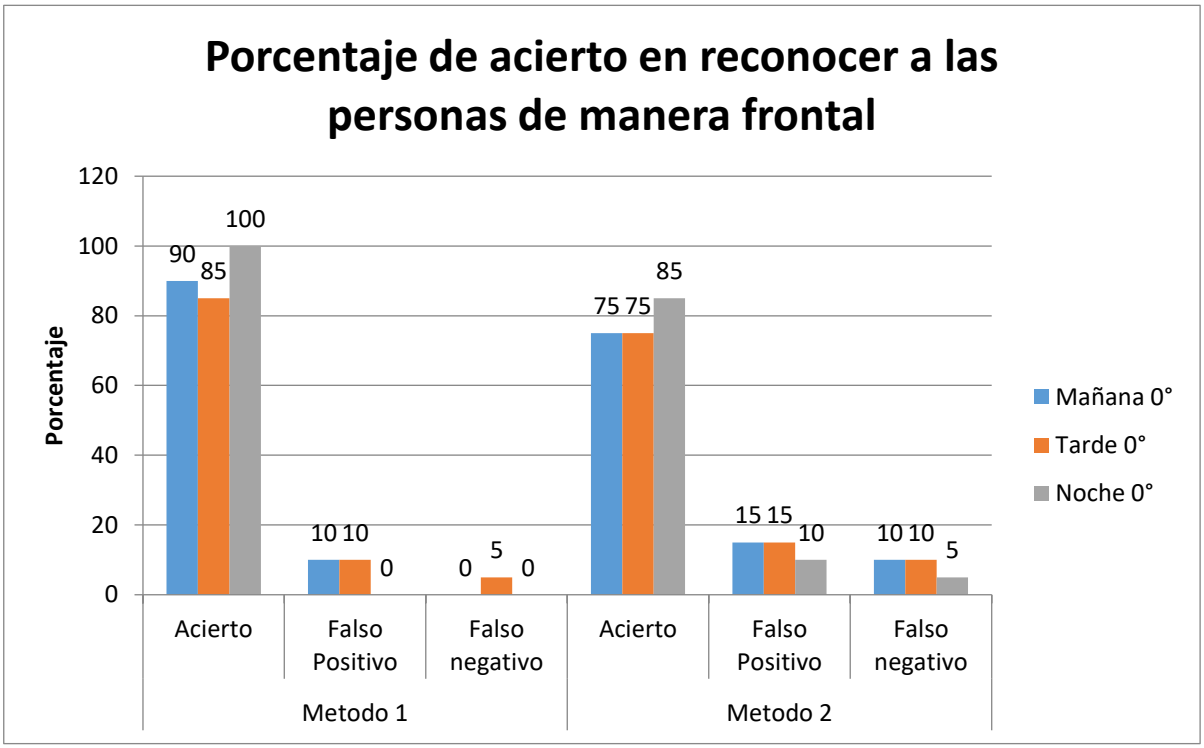


Figura 13: ***Porcentaje de aciertos y error en reconocer a las personas en los 3 escenarios (mañana, tarde y noche) de manera frontal.***

Fuente: Elaboración propia

La siguiente tabla 11 compara las pruebas con una orientación del rostro de 30° en los diferentes escenarios. Dichas pruebas se realizaron iguales a la anterior con diferencia de que en esta vez se evaluara con orientación del rostro de 30° para realizar una comparación de ambos métodos.

Tabla 11: ***Porcentaje de aciertos y error en reconocer a las personas en los 3 escenarios (mañana, tarde y noche) con orientación horizontal de 30°.***

	Método 1 LBP			Método 2 EIGENFACES		
Escenario y orientación	Acierto	Falso Positivo	Falso negativo	Acierto	Falso Positivo	Falso negativo
Mañana 30°	80%	20%	0%	75%	15%	10%
Tarde 30°	75%	25%	0%	75%	25%	0%
Noche 30°	85%	10%	5%	70%	15%	15%

Fuente: Elaboración propia

En la Figura 14 siguiente se muestra de manera comparativa ambos métodos comparados teniendo como escenario en la noche el método LBP obtiene el mayor porcentaje de acierto con un 85% de reconocimiento. Sin embargo, para el método 2 EigenFaces obtuvo menor porcentaje con el 70% de reconocimiento.

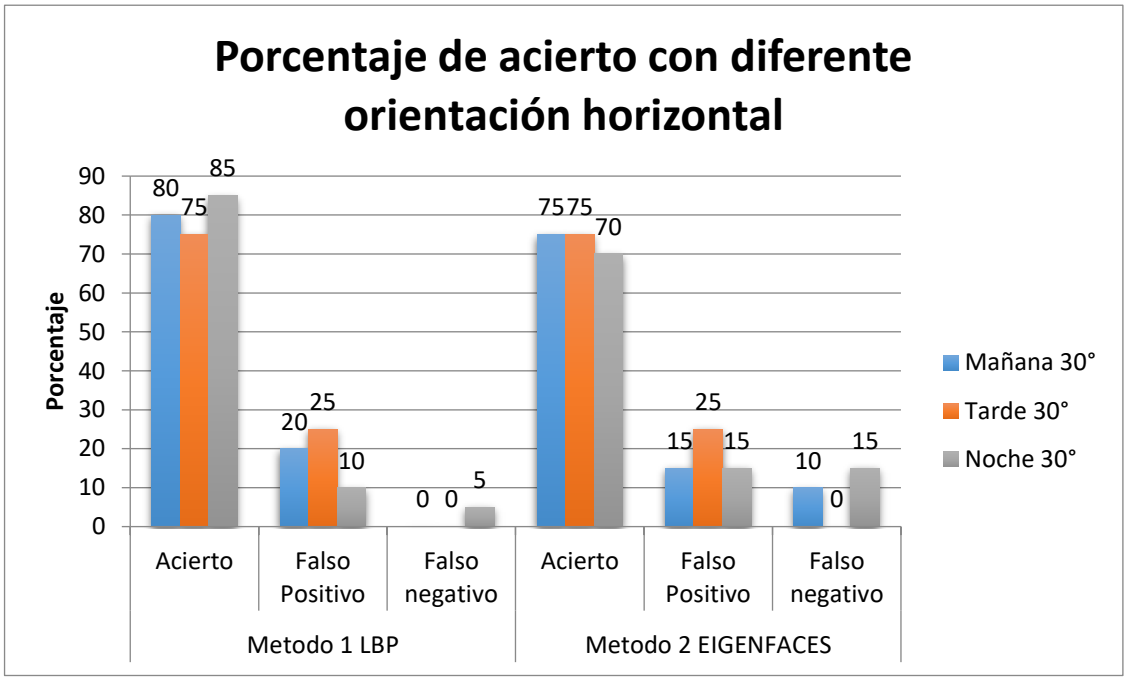


Figura 14: ***Porcentaje de aciertos y error en reconocer a las personas en los 3 escenarios (mañana, tarde y noche) con orientación horizontal de 30°.***

Fuente: Elaboración propia

La tabla 12 muestra compara los métodos con orientación de 60° en los diferentes escenarios (mañana, tarde y noche). Las pruebas se realizaron igual que las anteriores utilizando las 20 personas para los 3 escenarios como se visualiza hasta el momento mientras más sea el grado de orientación va perdiendo para ambos algoritmos el porcentaje de acierto.

Tabla 12: ***Porcentaje de aciertos y error en reconocer a las personas en los 3 escenarios (mañana, tarde y noche) con orientación horizontal de 60°.***

	Método 1 LBP			Método 2 EIGENFACES		
Escenario y orientación	Acierto	Falso Positivo	Falso negativo	Acierto	Falso Positivo	Falso negativo
Mañana 60°	60%	30%	10%	65%	30%	5%
Tarde 60°	60%	25%	15%	55%	35%	10%
Noche 60°	65%	20%	15%	55%	35%	10%

Fuente: Elaboración propia

La Figura 15 muestra comparativamente los resultados con 60° de orientación de manera horizontal. Teniendo un índice del 65% de acierto el método LBP y un falso positivo del 30% durante la mañana mientras que el método 2 EigenFaces logro tener durante la mañana el 65% de acierto y un falso positivo del 30% reconociendo a una persona por otra.

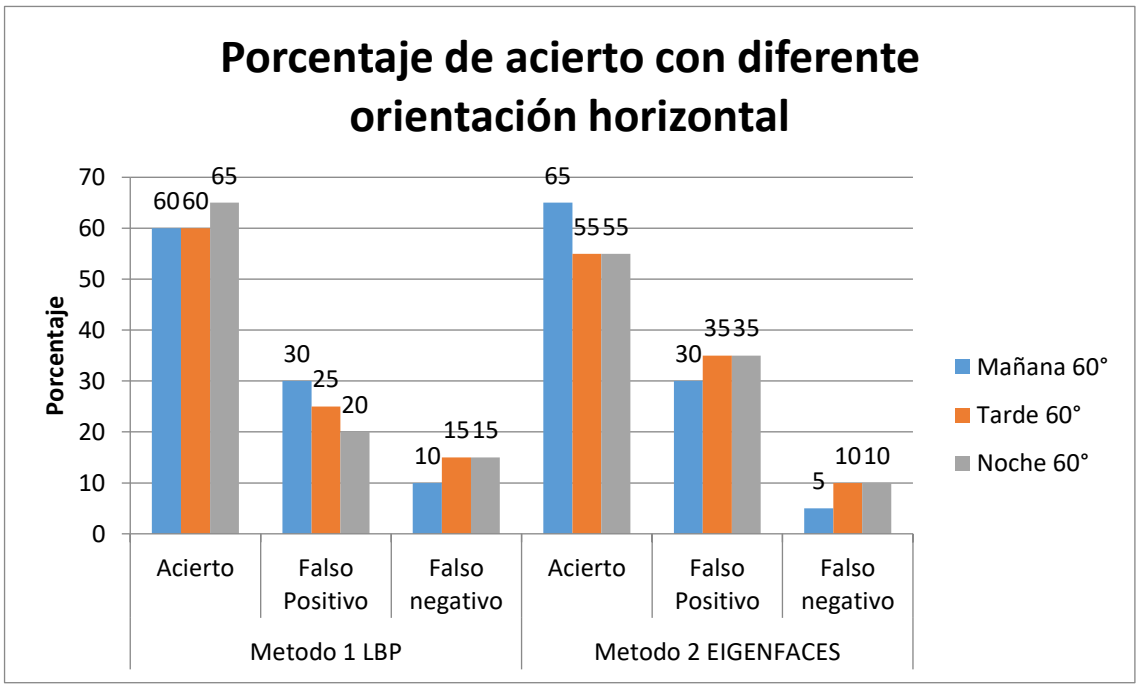


Figura 15: ***Porcentaje de aciertos y error en reconocer a las personas en los 3 escenarios (mañana, tarde y noche) con orientación horizontal de 60°.***

Fuente: Elaboración propia

Para el caso de prueba con orientación de rostro a 90° (perfil). Las pruebas se realizaron igual a las anteriores solo con variación de la orientación obteniendo resultados resaltantes. Logrando un porcentaje de acierto del 0% esto debido al clasificador en cascada por motivo que no identificaba el rostro.

Tabla 13: ***Porcentaje de aciertos y error en reconocer a las personas en los 3 escenarios (mañana, tarde y noche) con orientación horizontal de 60°.***

	Método 1 LBP			Método 2 EIGENFACES		
Escenario y orientación	Acierto	Falso Positivo	Falso negativo	Acierto	Falso Positivo	Falso negativo
Mañana 90°	0%	0%	100%	0%	0%	100%
Tarde 90°	0%	0%	100%	0%	0%	100%
Noche 90°	0%	0%	100%	0%	5%	95%

Fuente: Elaboración propia.

En la siguiente Figura 16 muestra comparativamente ambos métodos comparados teniendo para LBP el 0% de acierto y un falso negativo del 100% en los diferentes escenarios por otro lado EigenFaces obtuvo en la mayoría de sus escenarios el 100% y un falso positivo del 5% por una identificación errónea.

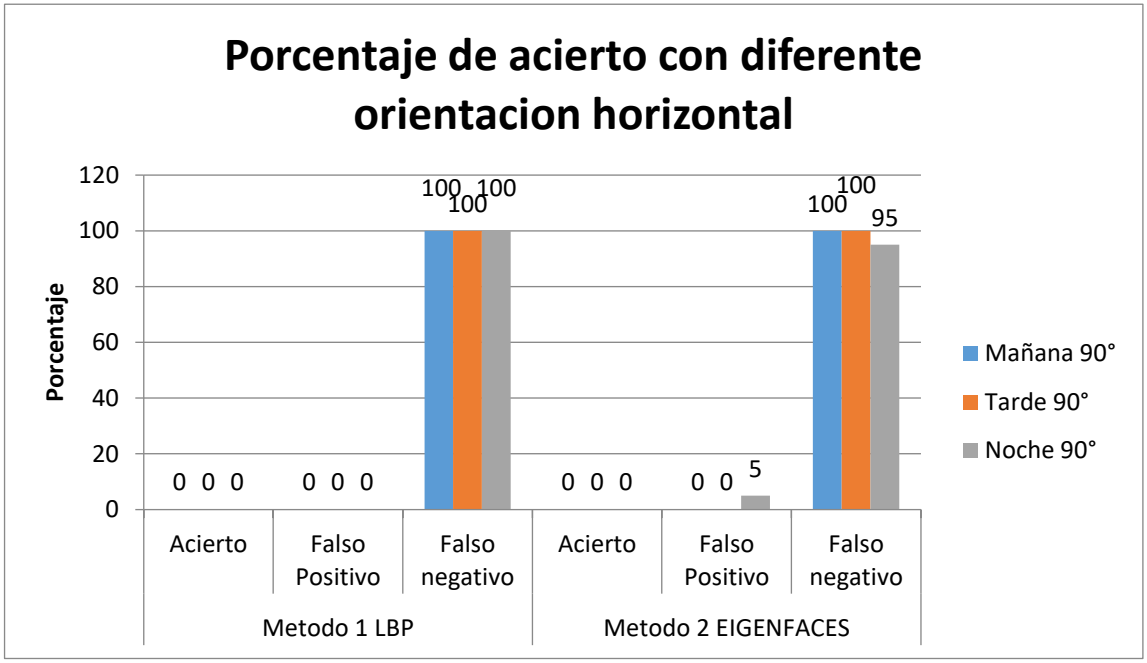


Figura 16: ***Porcentaje de aciertos y error en reconocer a las personas en los 3 escenarios (mañana, tarde y noche) con orientación horizontal de 90°.***

Fuente: Elaboración propia

En la tabla 14 muestra las pruebas realizadas con orientación de rostro de manera vertical hacia arriba y abajo en los diferentes escenarios que se describen en la tabla. Logrando un índice alto el método 1 LBP con el 85%. Para esta prueba no se tuvieron inconvenientes con el Clasificador en Cascada.

Tabla 14: ***Porcentaje de aciertos y error en reconocer a las personas en los 3 escenarios (mañana, tarde y noche) con orientación vertical.***

	Método 1 LBP			Método 2 EIGENFACES		
Escenario y orientación	Acierto	Falso Positivo	Falso negativo	Acierto	Falso Positivo	Falso negativo
Mañana Vertical Arriba	80%	15%	5%	75%	15%	10%
Tarde Vertical Arriba	85%	10%	5%	75%	15%	10%
Noche Vertical Arriba	90%	10%	0%	85%	15%	0%
Mañana	70%	10%	20%	75%	15%	10%

Vertical Abajo						
Tarde Vertical Abajo	70%	15%	15%	75%	15%	10%
Noche Vertical Abajo	70%	10%	20%	60%	20%	20%

Fuente: Elaboración propia

En la Figura 17 muestra comparativamente ambos métodos logrando visualizar con un alto índice del 90% el método 1 LBP con orientación vertical hacia arriba. Por otro lado, el método EigenFaces obtuvo unos porcentajes menos pero significativo para orientación con el 85% en la noche de manera vertical hacia arriba.

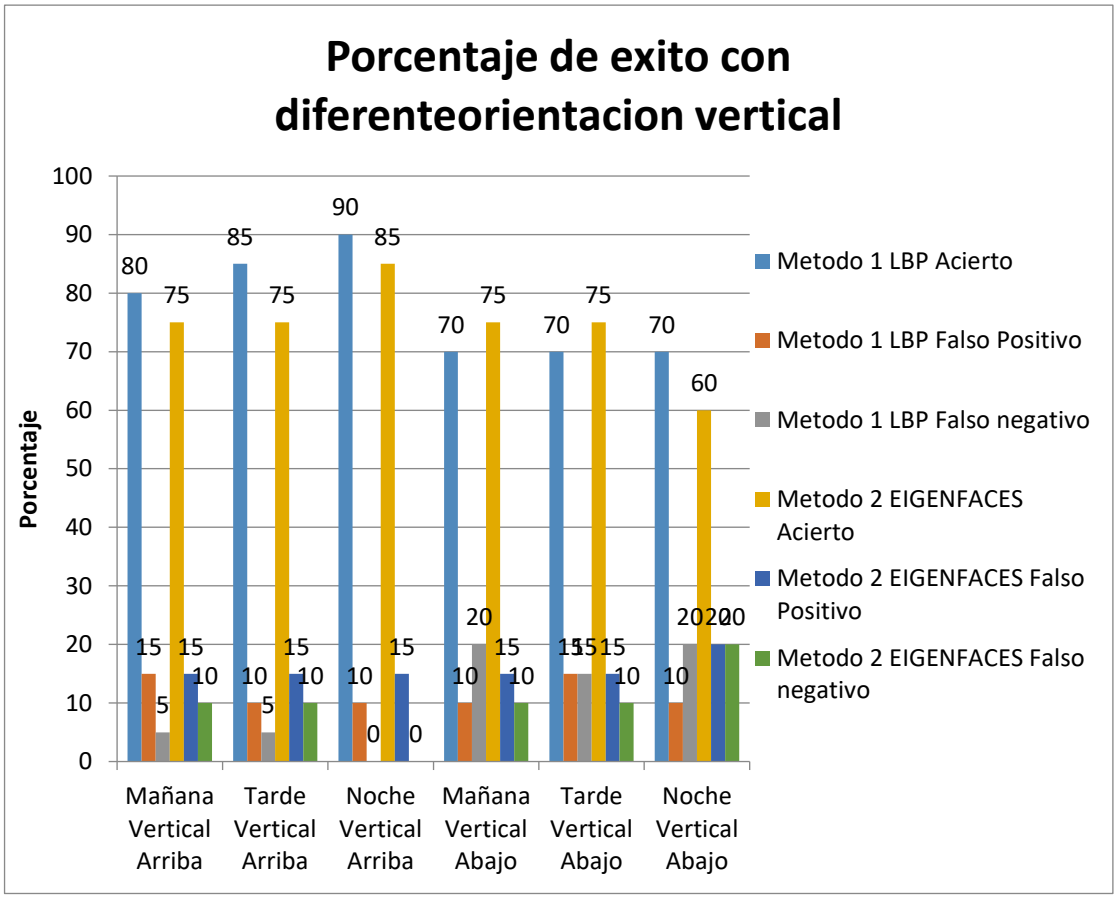


Figura 17: ***Porcentaje de aciertos y error en reconocer a las personas en los 3 escenarios (mañana, tarde y noche) con orientación vertical.***

Fuente: Elaboración propia

Por otro lado, con orientación hacia abajo EigenFaces logro superar por una persona más en reconocimiento durante el día con el 75% y LBP en la noche supera con el 70%.

4.1.3 Promedio total de recurso consumido en memoria RAM

En el presente experimento se midió el rendimiento computacional de la aplicación a la hora de reconocer a las personas. Estas pruebas se realizaron en ambiente no controlado. Con el fin de medir el consumo de la memoria RAM. Para eso se efectuó la siguiente operación para ello se evalúa desde el momento que identifica un rostro hasta que clasificador indique la persona reconocida.

$$PR = \frac{\sum Recurso\ consumido}{Total\ de\ ejecuciones}$$

Para esto se tuvo 7 pruebas por escenario de las mismas personas, de las 7 pruebas se obtiene el promedio de memoria que consume descrito en MB.

Para esto la figura que muestra a continuación, son algunos de los valores obtenidos en el escenario de la mañana con el método LBP.

Figura 18: ***Memoria RAM consumida en la mañana con el método LBP***

Fuente: Elaboración propia

En la figura siguiente, muestra uno de las 7 pruebas para evidenciar el gasto de la memoria que se consume durante la tarde.

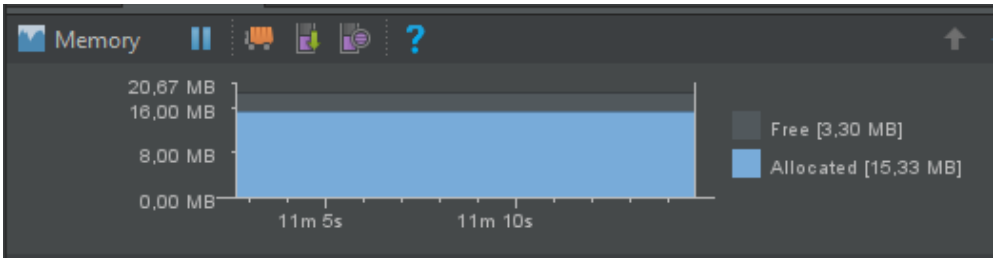


Figura 19: ***Memoria RAM consumida en la tarde***

Fuente: Elaboración propia

En la figura siguiente, nos muestra el gasto de memoria RAM durante la noche, esto realizado en un ambiente no controlado solo con iluminación artificial.

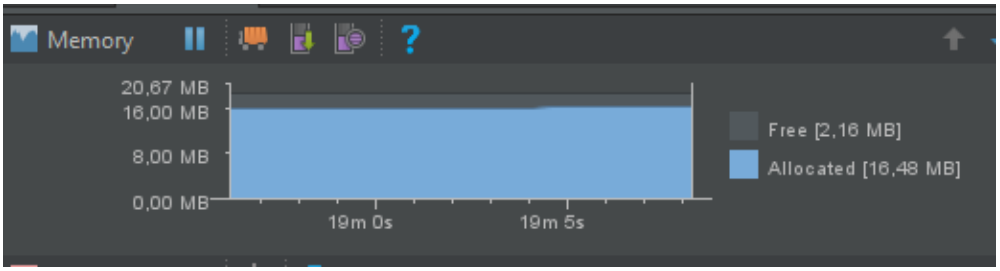


Figura 20: ***Memoria RAM consumida en la noche***

Fuente: Elaboración propia.

A continuación, se presenta los resultados del consumo de recursos de memoria RAM con el método 1 LBP, indicando cada prueba cuantos MB se consume durante el reconocimiento facial.

Tabla 15: ***Consumo de memoria RAM con el método LBP según los escenarios (mañana, tarde y noche).***

METODO LBP			
Escenario	**Nombre**		**Consumo de memoria RAM (MB)**
Mañana	Prueba 1	LILI_BERNAL	15.01

Mañana	Prueba 2	ROSSMERY_BERNAL	16.16
Mañana	Prueba 3	ASUNCION_BERNAL	14.97
Mañana	Prueba 4	DORELIS_LEYVA	15.53
Mañana	Prueba 5	ALEXANDER_BERNAL	16.14
Mañana	Prueba 6	ANTHONY_DAULE	15.76
Mañana	Prueba 7	CONSUELO_LEYVA	15.97
Tarde	Prueba 1	LILI_BERNAL	15.33
Tarde	Prueba 2	ROSSMERY_BERNAL	14.45
Tarde	Prueba 3	ASUNCION_BERNAL	14.99
Tarde	Prueba 4	DORELIS_LEYVA	15.32
Tarde	Prueba 5	ALEXANDER_BERNAL	16.00
Tarde	Prueba 6	ANTHONY_DAULE	15.54
Tarde	Prueba 7	CONSUELO_LEYVA	14.76
Noche	Prueba 1	LILI_BERNAL	16.48
Noche	Prueba 2	ROSSMERY_BERNAL	15.75
Noche	Prueba 3	ASUNCION_BERNAL	16.06
Noche	Prueba 4	DORELIS_LEYVA	16.24
Noche	Prueba 5	ALEXANDER_BERNAL	15.77
Noche	Prueba 6	ANTHONY_DAULE	15.93
Noche	Prueba 7	CONSUELO_LEYVA	15.55

Fuente: Elaboración propia.

La tabla 16 muestra los resultados de memoria consumida con el método EigenFaces, indicando cuando de memoria consume con la finalidad de comparar ambos métodos implementados.

Tabla 16: ***Consumo de memoria RAM con el método EigenFaces según los escenarios (mañana, tarde y noche).***

METODO EIGENFACES			
Escenario	**Nombre**		**Consumo de memoria RAM (MB)**
Mañana	Prueba 1	LILI_BERNAL	15.45
Mañana	Prueba 2	ROSSMERY_BERNAL	16.55
Mañana	Prueba 3	ASUNCION_BERNAL	16.97
Mañana	Prueba 4	DORELIS_LEYVA	15.53
Mañana	Prueba 5	ALEXANDER_BERNAL	16.13
Mañana	Prueba 6	ANTHONY_DAULE	15.76

Mañana	Prueba 7	CONSUELO_LEYVA	16.97
Tarde	Prueba 1	LILI_BERNAL	15.37
Tarde	Prueba 2	ROSSMERY_BERNAL	16.95
Tarde	Prueba 3	ASUNCION_BERNAL	16.99
Tarde	Prueba 4	DORELIS_LEYVA	15.32
Tarde	Prueba 5	ALEXANDER_BERNAL	17.46
Tarde	Prueba 6	ANTHONY_DAULE	15.54
Tarde	Prueba 7	CONSUELO_LEYVA	14.75
Noche	Prueba 1	LILI_BERNAL	16.41
Noche	Prueba 2	ROSSMERY_BERNAL	16.77
Noche	Prueba 3	ASUNCION_BERNAL	17.03
Noche	Prueba 4	DORELIS_LEYVA	16.29
Noche	Prueba 5	ALEXANDER_BERNAL	16.72
Noche	Prueba 6	ANTHONY_DAULE	16.98
Noche	Prueba 7	CONSUELO_LEYVA	15.51

Fuente: Elaboración propia.

A continuación, mostramos el consumo promedio de recursos según los escenarios de mañana, tarde y noche aplicando la formula anterior descrita.

Tabla 17: ***Consumo promedio de memoria RAM.***

ESCENARIOS	METODO 1 LBP	METODO 2 EIGENFACES
MAÑANA	15.65	16.19
TARDE	15.20	16.05
NOCHE	15.97	16.53

Fuente: Elaboración propia.

Posteriormente se visualiza comparativamente en la figura siguiente con finalidad de comparar ambos métodos, considerando que no existe gran variación entre los métodos LBP Y EigenFaces.

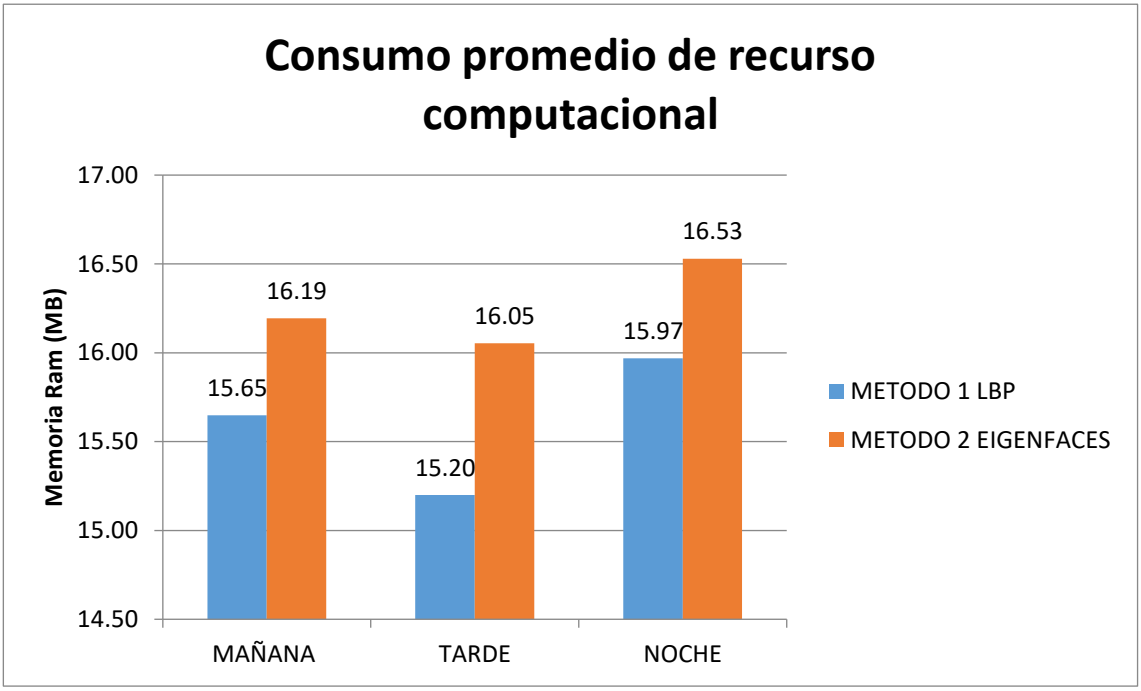


Figura 21: ***Consumo de Recurso Computacional***

Fuente: Elaboración propia

En la presente figura anterior se visualizó que durante la noche el consumo de memoria RAM en el método LBP es más elevado con un promedio de 15.97 MB, en caso contrario en la tarde consume menos memoria con 15.20, sin embargo, ambos resultados no existen gran diferencia. Por otro lado, en el método EigenFaces durante la noche consume 16.53 MB mayor que el método LBP.

4.2 Discusión de resultados

4.2.1 Tiempo promedio que tarda en reconocer a las personas.

Los resultados obtenidos en el tiempo de reconocimiento en los diferentes escenarios demostraron que en la mañana el método 1 LBP reconoce en promedio 0.456 segundos, reconociendo a la persona más rápido que el método 2 EigenFaces con una gran diferencia de 0.761 segundos.

En cambio, en la noche LBP demoro un poco más en reconocer teniendo como tiempo promedio 0.980 ms de manera que comparada con

EigenFaces también demora más tiempo que el método 1 con una aproximación de 1,087ms.

Por otro lado, al momento de tener toda la muestra al abrir la aplicación el método implementado con LBP tardaba un promedio de 35 segundos en cargar las fotos entrenadas de manera que si comparado con EigenFaces se tuvo problemas en algunos casos que el dispositivo no respondía.

4.2.2 Porcentaje promedio de aciertos en reconocer a las personas.

Los resultados de las pruebas realizas en la mañana, tarde y noche teniendo el entrenamiento completo con las 50 personas, se visualizó que las pruebas de manera frontal en la noche tuvieron el 100% de manera que la iluminación artificial en todas las pruebas realizadas en la noche ayuda a mejor el porcentaje de reconocimiento. Por otro lado, comparado con EigenFaces que obtuvo el máximo porcentaje durante la noche el 85 % por motivo que en la noche el método 1 LBP es mejor que EigenFaces por lograr un mejor porcentaje de aciertos.

Por otro lado, como si visualizan en la tabla de resultados se llegó a ver mientras la orientación horizontal sea mayor el clasificador en Cascado iba perdiendo la identificación de los rostros. Esto se consideró para ambos métodos. Sin embargo, mientras que el ángulo sea menor a los 60° se obtiene un porcentaje por encima del 50 % dependiendo el escenario.

En otro punto las pruebas realizadas de manera vertical lograron tener mejor porcentaje que las pruebas horizontales, de tal forma que el clasificador en cascada si lograba encontrar el rostro de la persona con orientaciones verticales y logrando una clasificación del 90% en la noche hacia arriba para el método LBP y muy similar el método EigenFaces con el 85%.

4.2.3 Promedio total de recurso consumido en memoria RAM.

Los resultados obtenidos para el método LBP el consumo de recursos en memoria RAM para los distintos escenarios no fueron grandes variantes

por motivo que los resultados obtenidos varían dentro de 1 MB de diferencia. Sin embargo, el cambio de consumo de memoria RAM es brusco en la mañana como se visualiza en la figura de resultados de memoria consumida con gran aumento de memoria, pero no supera al resultado más elevado. Como resultado del promedio de las pruebas en la noche fue el más elevado con 15.97 MB que para un dispositivo de gama alta y gama media lo podría utilizar, el cual dicha cantidad de memoria puede ser soportada por ambas gamas.

Por otro lado, el método EigenFaces aumentaba aproximadamente 0.60 MB más que el método 1 el cual el consumo es mayor en los diferentes escenarios fue en la noche con 16.53 MB lo es soportada por un dispositivo de gama alta el cual la gama media no podría soportar por motivo que toma mucho más tiempo en realizar el entrenamiento con una muestra superior a 750 imágenes.

CAPÍTULO V: PROPUESTA DE INVESTIGACIÓN

5.1 Estimación de Recursos del proyecto

En esta parte se indicará los diferentes recursos que ha sido utilizado para la creación del proyecto, mostrando las características tanto física como lógica.

5.1.1 Recursos Físicos

Recursos Físicos del ordenador: Las características donde se desarrolla gran parte del proyecto son:

Procesador Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz

10 GB de memoria RAM

Disco duro de 500GB

Características del Dispositivo móvil: Este dispositivo es indispensable para la ejecución y pruebas del proyecto. Las características del dispositivo utilizado son:

Marca y modelo: Samsung Galaxy S8 SM-G950F

Procesador: Octa-Core Exynos 8895 Chipset

Gráfica: ARM Mali-G71 MP20 550Mhz

Pantalla: 5.8" pulgadas 1440 x 2960 píxeles

Cámara de 12 Megapíxeles

Android 7.0 Nougat

Almacenamiento de 64 GB

RAM de 4GB

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

5.1.2 Recursos de software

Sistema Operativo Windows 7: Sistema Operativo instalado en el dispositivo portátil.

Android Studio 2.3.3: Entorno de desarrollo del proyecto con la última versión actual hasta la fecha.

Java SE Development Kit 8u144: Se utilizó la última versión descargada de la propia página web.

OpenCv 3.1.0: Librería utilizada para el desarrollo del proyecto, utilizando la versión estable descargada de la propia página de OpenCv.

5.2 Diseño y Desarrollo del protocolo de Adquisición de imágenes.

5.2.1 Requerimientos de fotos:

a) El dispositivo hace la captura de fotos con un tamaño por defecto de 1280 x 720 pixeles.

b) El recorte de la parte importante de la foto se hizo con un tamaño de 128x128 pixeles.

c) La distancia la cual se hizo la captura de imágenes fue de 50cm de distancia desde el dispositivo hasta el rostro de la persona.

d) La imagen del rostro se guardó a colores.

e) La imagen guardada de tipo de formato jpg.

f) El nombre de la imagen guardada en la base de datos de imágenes de rostros esta concatenado por nombre_apellido_numerodefoto.extensión. Ejemplo JUAN_PEREZ_23.jpg.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

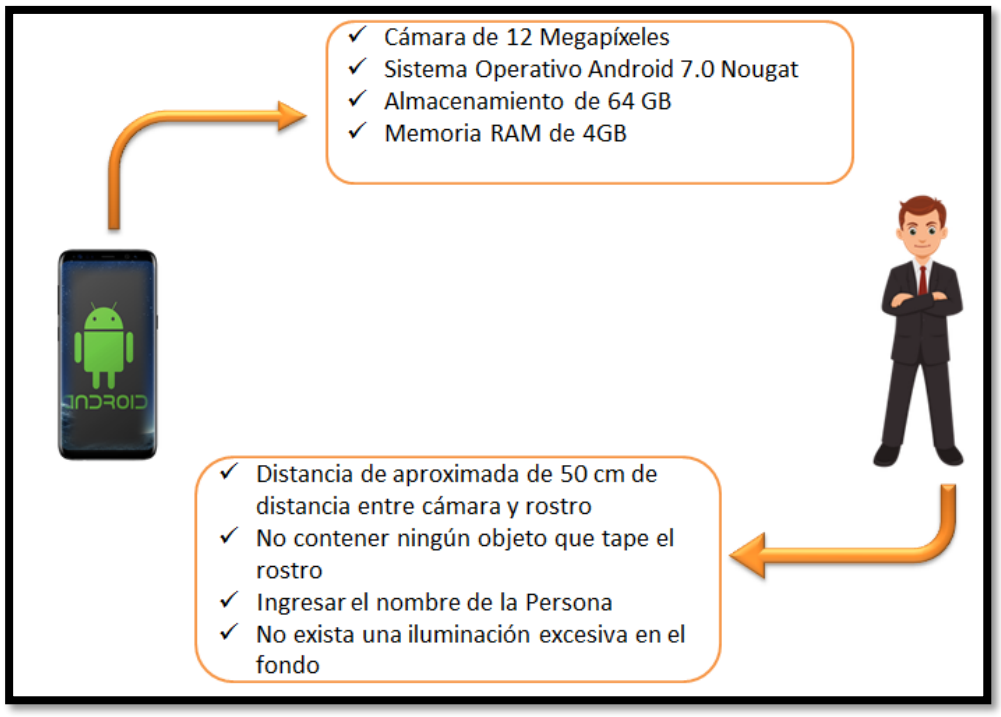


Figura 22: ***Requerimientos de la persona y dispositivo para la captura de imágenes.***

Fuente: Elaboración propia.

En la Figura 22 se describe detalladamente los requerimientos que tiene que cumplir la persona y el dispositivo de manera que se desarrolle correctamente el protocolo de adquisición de imágenes.

5.2.2 Captura de imágenes

Las imágenes adquiridas de rostros fueron capturadas en diferentes orientaciones para su entrenamiento, en la Figura 23 muestra las orientaciones de cara la cual se adquirió 5 orientaciones de rostro.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

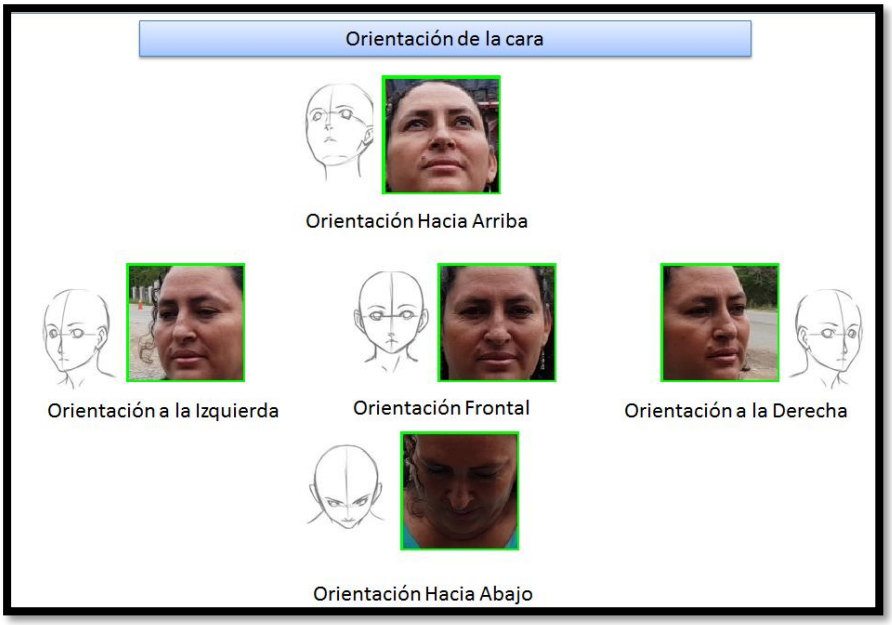


Figura 23: ***Orientaciones de la cara.***

Fuente: Elaboración propia.

Se consideró 30° 60° y 90 grados horizontal según un artículo de investigación que trata de medir el nivel de acierto que tiene con las diferentes orientaciones de rostro (Qawlan Akariman, 2015).

Se consideró también realizar el entrenamiento con fotos de manera vertical por motivo que según lo investigado ningún investigador realizo el entrenamiento de manera vertical el cual un rostro puede existir para el momento de la clasificación variación de manera vertical por lo que se optó realizar el entrenamiento y las pruebas de manera vertical.

La primera foto capturada para el entrenamiento fue tomada en orientación frontal 0° grados, luego la segunda foto fue tomada en orientación a la derecha 30° grados, la tercera foto fue tomada en orientación a la izquierda 30°, la cuarta foto fue tomada en orientación Hacia Arriba hasta el límite de giro, y la quinta foto es tomada en orientación hacia abajo también hasta el límite de giro de la cabeza.

Las fotos adquiridas fueron tomadas en diferentes tipos de ambientes no controlados, el cual se tomaron 5 fotos en la mañana de 7:00 AM a

9:00 AM y almacenados en el dispositivo, luego a la misma persona se tomó 5 fotos más en la tarde en el rango de hora 1:00 PM a 3:00 PM con la iluminación de la luz solar al aire libre y se tomaron 5 fotos más en la noche teniendo como rango de hora de 7:00 PM a 9:00 PM con iluminación artificial , según las orientaciones descritas anteriormente (Qawlan Akariman , Agung Nugroho Jati , Astri Novianty, 2017).

Las fotos almacenas en la base de datos fueron tomadas sin uso del flash a una distancia de 50 cm, ver Figura 24. Las fotos adquiridas para el entrenamiento fueron un total de 15 fotos en diferentes ambientes no controlados.

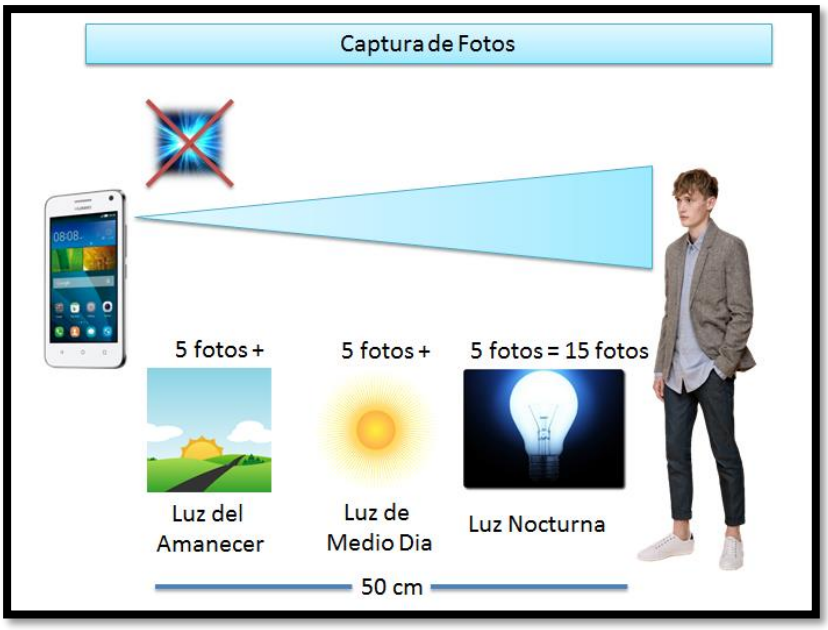


Figura 24: ***Condiciones y escenarios para la captura de fotos.***

Fuente: Google.

Para la identificación y adquisición de la parte de interés (rostro) se realizó el siguiente proceso descrito en la Figura 25, la cual se describirá por partes.

Figura 25: ***Proceso de adquisición de rostro.***

Fuente: Elaboración propia.

Para el desarrollo del protocolo de adquisición de imágenes se tuvo que construir una interfaz que utiliza la cámara posterior del dispositivo y con ayuda de la librería OpenCv para poder manipular la cámara. La funcionalidad, como ya se ha comentado esta implementado en clases Java, mientras que la interface de Usuario está en XML. Estas interfaces llamadas Layout, contienen los componentes de la interfaz mostrando la ubicación, tamaño y nombre del componente para poder llamarlo desde la clase Java. Como podemos ver en la Figura 26, se visualiza el componente de la cámara utilizando la librería ya mencionada, como se visualiza es la interfaz construida del prototipo de reconocimiento facial.

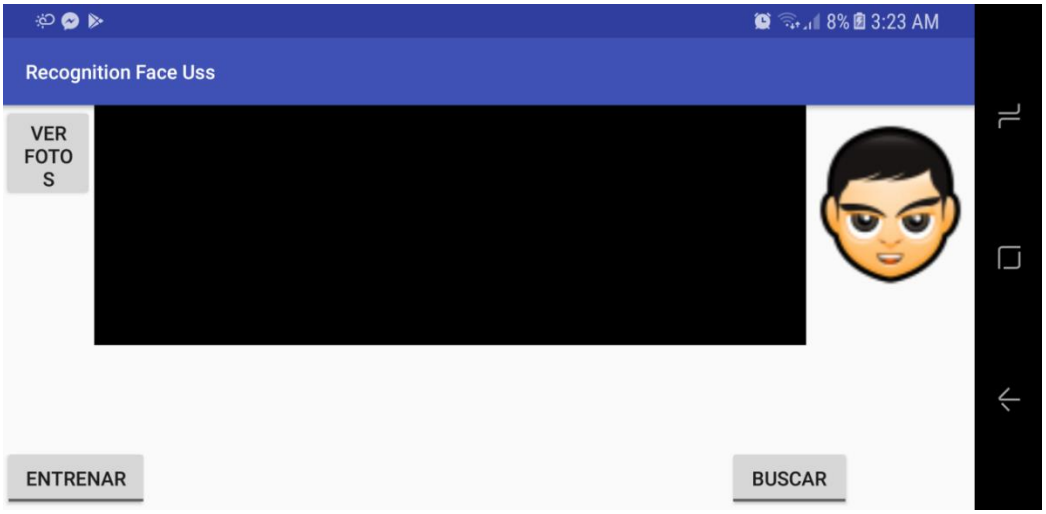


Figura 26: ***Interfaz principal del proyecto.***

Fuente: Elaboración propia.

Para conectar el componente de la cámara (JavaCamaraView) a la clase CamaraActivity se declaró una variable llamado mOpenCvCameraView de tipo CameraBridgeViewBase, con la finalidad de recibir los datos para poder realizar los diferentes procesos para el reconocimiento facial.

Figura 27: ***Conexión el componente del XML con Java.***

Fuente: Google.

5.2.3 Identificación del rostro

En la primera etapa se realiza la identificación del rostro. Para ello gracias a la librería OpenCV se puede realizar la identificación con el algoritmo de Viola-Jones apoyado en características Haar o utilizando LBP.

El primer paso para esta etapa se tiene que cargar previamente el algoritmo de un archivo con formato XML almacenado dentro del proyecto Android creado. A continuación, en el Figura 28 se muestra el cargado del archivo XML

Luego de haber cargado el archivo XML como paso importante debemos convertir la imagen en escala de grises. Para optimizar la identificación del rostro se puede modificar unos parámetros.

Factor a escala: Valor que determina cuanto se reduce la imagen para detectar el rostro. Mientras menos sea el factor mayor precisión tendrá, pero tardara más en detectar. En este caso el factor fue de 1.1 que se redujera el 10%.

Número de vecinos: Parámetro que indica el número de pixeles vecinos. Mientras menor sea el número de vecinos menor será la eficiencia del algoritmo, pero con mayores detecciones. En este caso 3 vecinos es el número con mayor número de aciertos.

Tamaño máximo y mínimo del rostro: Parámetro que indica el tamaño mínimo y máximo del rostro. En este caso se indicó que el tamaño mínimo del rostro sea el 20% del tamaño.

A continuación, para la selección del algoritmo que mejor destaque en identificación se revisó información donde según Diego Zapatero Olmedillo (2016), en su tesis Herramientas de Reconocimiento Facial de Emociones en Android, realizo una comparación de los dos algoritmos Haar y LBP con las diferentes versiones existentes, comparando la luminosidad y velocidad de identificación.

Donde se comparó con 40, 200 y 450 de luminosidad donde el algoritmo Haar con la versión haarcascade_frontalface_alt obtuvo un promedio de 38.0 milisegundos (ms) en las distintas intensidades de luminosidad en cambio grande es la diferencia en tiempo con el algoritmo LBP con la versión lbpcascade_frontalface donde con luminosidad 40 se obtuvo 14.7

ms, con luminosidad 200 obtuvo 15.8 ms y con luminosidad 450 se obtuvo 16.3 ms sacando un promedio de 14.9 ms en velocidad de identificación.

Con la comparación de los algoritmos se apreció que ambos algoritmos son rápidos para la identificación. Sin embargo, el algoritmo LBP fue enormemente más rápido identificando que el algoritmo de Viola-Jones. Por esta razón fue que se utilizó el algoritmo LBP.

```
InputStream is = getResources().openRawResource(R.raw.lbpcascade_frontalface);
File cascadeDir = getDir("cascade", getApplicationContext().MODE_APPEND);
File mCascadeFile = new File(cascadeDir, "lbpcascade_frontalface.xml");
FileOutputStream os = new FileOutputStream(mCascadeFile);
```

Figura 28: ***Carga del algoritmo***

Fuente: Elaboración propia.

Una vez cargado el algoritmo se utilizó el clasificador para detectar caras a multiescala para eso se utilizó la librería OpenCV, donde utilizo el método detectMultiScale de la clase CascadeClassifier, donde previamente se ingresó como parámetros como se visualiza en la figura 29, la imagen convertida en escala de grises ecualizada, el factor de escala, número de vecinos y tamaño máximo y mínimo.

```
MatOfRect faces = new MatOfRect();
if (cascadeClassifier != null) {
        cascadeClassifier.detectMultiScale(grayscaleImage, faces, 1.1, 3, 2,
        new Size(absoluteFaceSize, absoluteFaceSize), new Size());
}
```

Figura 29: ***Parámetros para la detección de caras a multiescala.***

Fuente: Elaboración propia.

Una vez encontrada los rostros en la imagen en tiempo real, en el código de la Figura 30 se encargará de dibujar un rectángulo a su alrededor, para eso se aplicó un for para recorrer los pixeles de la imagen de color y el

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

método rectangle de la clase Imgproc de OpenCV se encargó de dibujar el rectángulo en la imagen.

```
Rect[] facesArray = faces.toArray();
for (int i = 0; i <facesArray.length; i++) {
        Imgproc.rectangle(mat, facesArray[i].tl(), facesArray[i].br(), new Scalar(0, 255, 0, 255), 3);
}
```

Figura 30: ***Código utilizado para dibujar el rectángulo sobre la cara identificada***

Fuente: Elaboración propia.

Por ejemplo, en la Figura 31, se muestra la identificación del rostro dibujando un rectángulo alrededor de la cara detectada según el algoritmo de identificación de rostro muestra un desempeño optimo en ambiente no controlado.

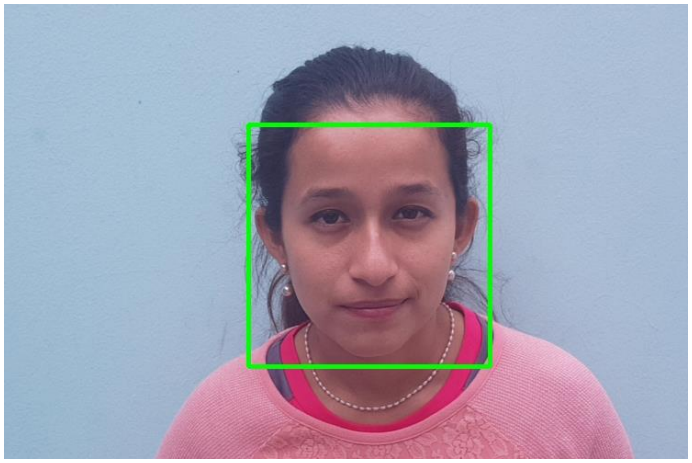


Figura 31: ***Identificación de rostro.***

Fuente: Elaboración propia.

Escala de Gris

Este paso consiste en transformar la imagen con intensidades de Rojo, Verde y Azul por pixel que serían de 3 canales, convirtiendo a escala de grises de 1 canal que varían de 0 a 255 intensidades de grises. Siendo entonces un paso importante para el pre procesamiento de la imagen con

la finalidad central la información en una dimensión y mostrar las diversas texturas de la imagen.

Para la conversión de escala de grises se utilizó la ecuación ponderada para la conversión de RGB a escala de grises (M.Sc Jimy Alexander Cortés Osorio, 2011).

Como sabemos el ojo percibe diferentes intensidades de luz en función del color que se observa, esto es debido a la respuesta del ojo al espectro visible la cual se puede observar en la figura 32, por esa razón el cálculo de la escala de grises o luminancia de la imagen debe realizarse como una media ponderada de las distintas componentes de color de cada pixel

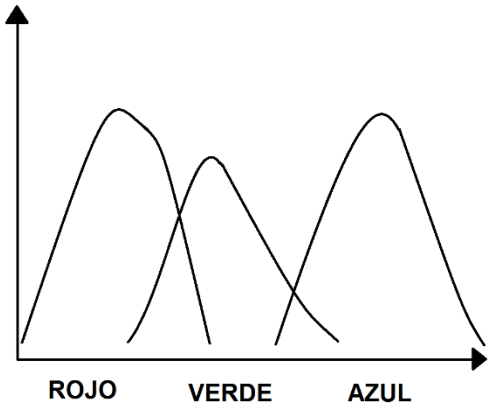


Figura 32: ***Longitud de ondas***

Fuente: Elaboración propia

Los factores de ponderación de cada componente de color nos indican la sensibilidad del ojo humano a las frecuencias del espectro cercanas al rojo, verde y azul. Para realizar la conversión basta con aplicar la siguiente ecuación (Juan José Ortuño López, 2016).

Ecuación Matemática

$$\boldsymbol{Y = R * 0.3 + G * 0.59 + B * 0.11}$$

Y: Es el resultado obtenido de la ecuación ya convertido en escala de gris.

R: Es la intensidad de color Rojo obtenido del pixel

G: Es la intensidad de color Verde obtenido del pixel

B: Es la intensidad de color Azul obtenido del pixel

Donde al efectuar la ecuación con los valores de los pixeles RGB (Rojo, Verde y Azul) se obtendrá un nuevo valor Y. De esta manera la imagen completa será convertida en una imagen en escala de gris convirtiendo pixel a pixel de color RGB a escala de gris de un byte.

Ejemplo de la ecuación matemática:

Color ingresado

Transformación de RGB a escala de gris

$$Y = 237 * 0.3 + 206 * 0.59 + 187 * 0.11$$

$$Y = 71.1 + 121.54 + 20.57$$

$$\boldsymbol{Y = 213}$$

Conversión en escala de gris

Esta conversión a escala de grises se implementó en el método Abstracto al implementar CvCameraViewListener2 llamado **onCameraFrame** que en la imagen, se muestra en la figura 33 el

pseudocódigo de la implementación del método para convertir a escala de grises.

```
Inicio
        Imagen_RGB;
        Imagen grayscaleImage = Convertir (CvtColor( Imagen_RGB a Escala_grises Aplica formula matematica))
        Return grayscaleImage
Fin
```

Figura 33: ***Pseudocódigo de escala de grises.***

Fuente: Elaboración propia.

Donde la primera línea de código se obtiene los datos de la cámara en tiempo real en 3 canales RGB; los datos que ingresan son de Tipo Mat que representa una matriz dimensional que puede almacenar imágenes de una solo canal o multicanal eso quiere decir que puede tener datos de las imágenes a color o en escala de grises. La segunda línea utilizamos la conversión de OpenCV ya implementado la ecuación matemática donde se transformará la imagen a color a escala de grises.

Para este método CvtColor se ingresa la imagen a color y de salida se almacenará en la variable grayscaleImage de tipo Mat y también se envía el tipo de conversión que se desea hacer en este caso se realizó la conversión a escala de grises de 8 bits.

Por ejemplo, en la Figura 34 se muestra la conversión de la imagen original a imagen en escala de grises.

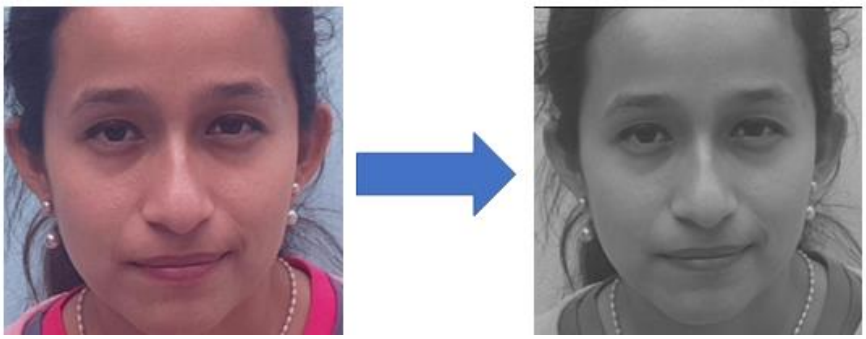


Figura 34: ***Conversión a escala de grises.***

Fuente: Elaboración propia.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Ecualización de Histogramas

El paso siguiente que se empleó fue la ecualización de histogramas en la imagen transformada a escala de grises con la finalidad de estandarizar el contraste y brillo de la imagen, esta parte es muy importante en esta investigación ya que estamos trabajando en un ambiente no controlado. Con este paso realizado se mejoró la identificación del rostro, por lo que se está eliminando el ruido de las imágenes (exceso de brillo).

La finalidad de la ecualización de histogramas es que la imagen ingresada en escala de grises tenga una distribución uniforme sobre todos los pixeles en escala de grises.

La primera fórmula es calcular el histograma de la imagen, que es la suma total de la misma intensidad del pixel.

$$n_k = h[r_k]$$

H = histograma

K = 0 – 255

N = Número total de pixeles de la imagen

En la Figura 35 se visualiza un ejemplo de intensidades de pixeles para calcular la formula.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

2	1	5	5	8	9	5	5
8	2	3	3	2	1	6	6
6	2	2	2	4	2	1	1
5	8	4	2	1	2	4	2
3	2	2	1	8	1	2	1
6	3	5	3	1	2	1	2
8	3	6	3	1	1	2	5
9	6	2	6	8	3	5	7

Figura 35: ***Ejemplo de intensidades de pixeles.***

Fuente: Elaboración propia.

Una vez que se tiene las intensidades de pixeles de una imagen en una matriz se realizó la sumatoria de todos los pixeles con la misma intensidad, para ello se tiene cuantas veces se repite la misma intensidad, por ejemplo, a continuación, se sumó todos los pixeles de la misma intensidad y se obtuvo un histograma como se visualiza en la Figura 36.

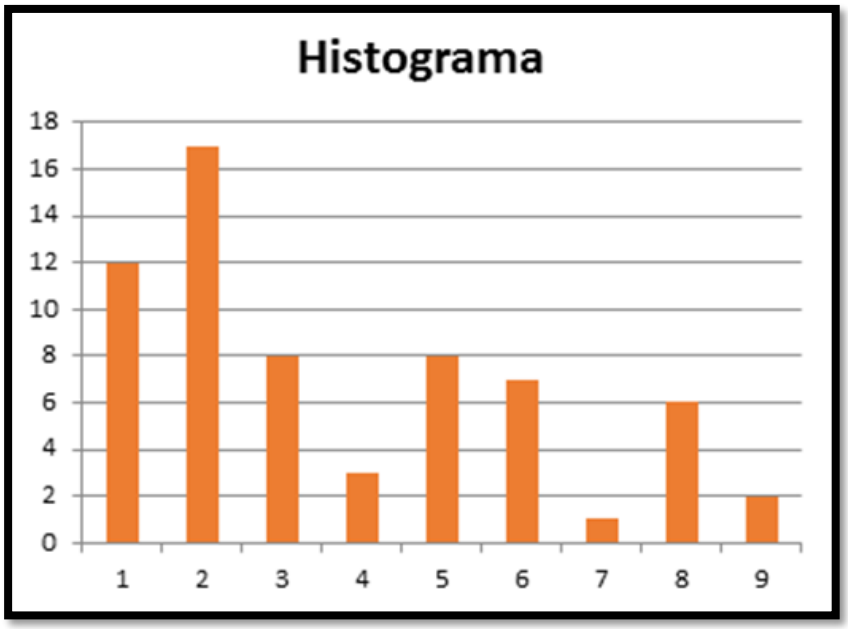


Figura 36: ***Histograma de ejemplo.***

Fuente: Elaboración propia.

Una vez obtenido el histograma de la imagen se calculará el histograma calculado, que es la siguiente formula:

$$H(a) = \sum_{I=1}^{MN} H\,(i-1) + h(i)$$

MN = Total de pixeles

H = Histograma

i = intensidad

El histograma acumulado se calcula con el total de la intensidad de los pixeles más la sumatoria del pixel anterior. Por ejemplo, en la Figura 37 se visualiza el histograma acumulado del histograma generado anteriormente.

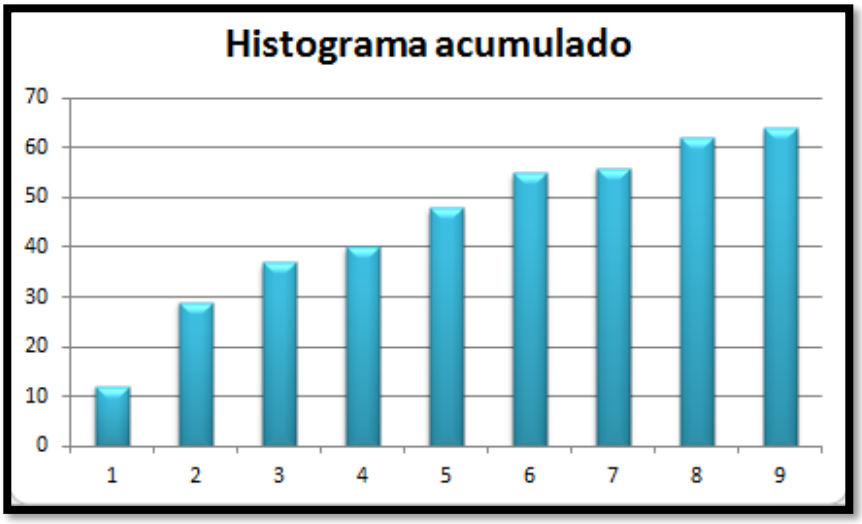


Figura 37: ***Histograma acumulado***

Fuente: Elaboración propia.

Continuando con las operaciones para lograr la ecualización del histograma se desarrolló la siguiente formula:

$$Feq(a) = \left[H(a)\,.\frac{k-1}{MN}\right]$$

Una vez efectuada la operación matemática del histograma acumulado por k -1 entre el total de pixeles de la imagen se obtiene una nueva matriz de la imagen.

Una vez terminado este proceso la ecualización de histogramas crea una nueva imagen donde los niveles de intensidades se normalizar en el rango de 0 a 255. El resultado de la ecualización del histograma mejorando las intensidades de brillo excesivo y maximiza el contraste de una imagen sin perder la estructura de la imagen.

5.2.4 Construcción de la base de datos y almacenamiento de la muestra

Una vez construido el proceso de identificación, se realizó la toma de la muestra de la investigación y posteriormente se necesitó almacenar las imágenes en el dispositivo, con su respectivo nombre de la foto, con la finalidad de construir así la base de datos de los rostros a entrenar. Primeramente, las fotos tomas fueron en ambiente que ya se indicó anteriormente con un total de 15 fotos por persona. Para el nombre de las fotos se especificó anteriormente en los requerimientos de las fotos, ya que es importante al momento de clasificarlos para mostrar el respectivo nombre de la persona identificada.

Para la construcción de la base de datos de las imágenes se tuvo que crear una carpeta en el dispositivo, que en este caso se llamó FACERECOGNITIONA, donde se almaceno todas las fotos de la muestra y también se creó un archivo .txt donde están los nombres o etiquetas de las personas registradas.

Donde el sistema necesita la imagen del rostro y el nombre de la persona, para este paso como se ve en la figura 38, se creó un método con el nombre add donde recibe la imagen de tipo Mat y el nombre que se indicó. Por lo tanto, lo siguiente fue crear una imagen de tipo BitMap del mismo

tamaño y pixeles de la imagen ingresada, luego se reduce la imagen indicando el tamaño de la imagen, que en este caso es de 128 x 128 pixeles, una vez que se realizó el paso anterior se crea el nombre según los requerimientos de la foto, para posteriormente crear el nuevo archivo (imagen) con la extensión .jpg y así concluyendo el paso de almacenar la imagen del rostro.

```
public void add(Mat m, String description) {
  Bitmap bmp= Bitmap.createBitmap(m.width(), m.height(), Bitmap.Config.ARGB_8888);
  Utils.matToBitmap(m,bmp);
  bmp= Bitmap.createScaledBitmap(bmp, WIDTH, HEIGHT, false);
  FileOutputStream f;
  try {
    String nombres[] = description.toString().split(" ");
    StringBuilder s = new StringBuilder();
    for (String nom : nombres) {
      s.append(nom);
      s.append("_");
    }
    f = new FileOutputStream(mPath + s + "" + count + ".jpg",true);
    count++;
    bmp.compress(Bitmap.CompressFormat.JPEG, 100, f);
    f.close();
  } catch (Exception e) {
    Log.e("error",e.getCause()+" "+e.getMessage());
    e.printStackTrace();
  }
}
```

Figura 38: ***Almacenamiento de las imagines.***

Fuente: Elaboración propia.

Una vez que se realizó el proceso de almacenar las imágenes se procedió a tomar las fotos. Como se mencionó en la muestra las fotos adquiridas para el entrenamiento se realizó durante la mañana teniendo como promedio las 8:15 AM, donde se capturo el primer conjunto de fotos con 5 fotos por persona, tomando a 50 personas de diferentes edades, en las diferentes orientaciones de rostro ya indicadas, haciendo un total de 250 fotos, en ambiente no controlado en el cual las personas no poseían ningún artículo en el rostro, como se puede apreciar en la figura 39.

Figura 39: ***Fotos adquiridas en la mañana.***

Fuente: Elaboración propia

El segundo conjunto de fotos para el entrenamiento se consideró las mismas personas que fueron tomas en el primer escenario, por lo tanto, las personas tampoco poseían algún objeto en el rostro, teniendo como promedio la 12:46 PM del medio día, en ambiente no controlado y el cielo despejado, también se capturo 5 fotos de diferentes orientaciones de rostro a las 50 personas, haciendo un total de 250 fotos más para el entrenamiento. Como se puede apreciar en la Figura 40.

Figura 40: ***Fotos adquiridas al medio día.***

Fuente: Elaboración propia.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Para el último conjunto de fotos para el entrenamiento fueron las mismas 50 personas tomadas con anterioridad, teniendo como promedio las 7:28 PM de la noche, las fotos fueron capturadas en ambiente no controlado no más que un foco tomada como iluminación artificial, en este caso tampoco las personas poseían algún objeto en el rostro, por lo tanto se capturo 5 fotos por persona en las diferentes orientaciones, haciendo un total de 250 fotos más que se agregan al entrenamiento como se muestra en la Figura 41.

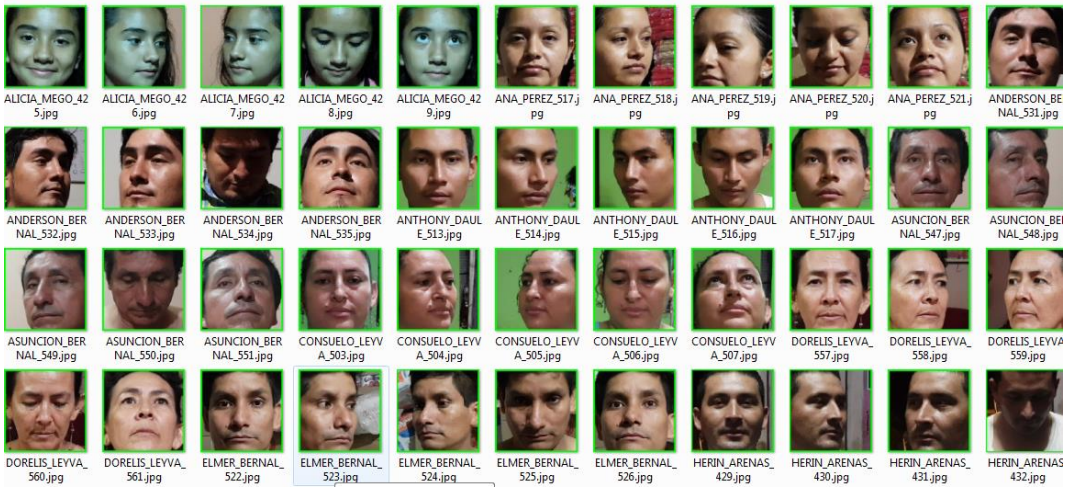


Figura 41: ***Fotos adquiridas en la noche.***

Fuente: Elaboración Propia

Se tomaron un total de 750 fotos para el entrenamiento, teniendo 15 fotos por persona, donde cada posición del rostro contiene 3 fotos almacenadas, cada una en diferente escenario.

5.3 Selección de métodos de reconocimiento de imágenes

A base de las tesis y artículos científicos revisados en esta investigación se realizó un análisis de los resultados de los diferentes antecedentes como se puede visualizar en el capítulo 2.

Para los métodos implementados se optó implementar para ambos el mismo clasificador en cascada para poder identificar el área de interés, de manera que según Diego Zapatero Olmedillo (2016), en su tesis Herramientas de Reconocimiento Facial de Emociones en Android, realizo una comparación de los dos algoritmos de clasificación en cascada HaarCascade y LBPCascade teniendo un mejor desempeño el algoritmo LBPCascade y también realiza una identificación más precisa.

El método Patrón Binario Local ha alcanzado resultados más altos logrando llegar hasta el 100% según el análisis y en otros casos logrando aproximarse al 100% en las diferentes investigaciones, cumpliendo con algunas características de orientación, rendimiento e iluminación el cual había sido seleccionado para su implementación en un dispositivo inteligentes (Javier Eslava Ríos, 2013).

El método Patrones Binarios Locales para la clasificación utiliza el algoritmo probabilístico que implementa la fórmula matemática de Chi-cuadrado. Dicho clasificador es considerado como clasificador de texturas según la documentación de OpenCV 3.3.1.

Tabla 18: ***Método 1 LBP (Identificador de rostro, extractor de características y clasificador).***

Método	Identificación	Extracción	Clasificación
Método 1	LBPCascade	LBP Operador	Chi-Cuadrado

Fuente: Elaboración propia.

Por otro lado, se implementará el método de EigenFaces utilizando la técnica de PCA que consiste en reducir la dimensionalidad en un conjunto de datos y facilitándolo al clasificador. (David Eduardo Espinoza Olguín, 2015). Por motivo que los recursos en dispositivos móviles son muy reducidos se optó por dicho método.

Y para el clasificador el mismo autor David Eduardo Espinoza Olguín, 2015 implemento para el método de EigenFaces aplica la fórmula matemática de Distancia Euclidiana ya que no consume demasiados recursos.

Tabla 19: ***Método 2 EigenFaces (Identificador de rostro, extractor de características y clasificador).***

Método	Identificación	Extracción	Clasificación
Método 2	LBPCascade	EigenFaces	Distancia Euclidiana

Fuente: Elaboración propia.

Los métodos descritos se implementaron con la finalidad de realizar que método de reconocimiento facial se desempeña mejor en dispositivos móviles sin tener demasiado inconveniente con el consumo computacional por motivo que los recursos son escasos.

5.4 Implementación de los métodos de reconocimiento facial

Según el análisis de los métodos investigados, que tengan el mejor número de aciertos considerando que los recursos son limitados ya que fue implementado en Dispositivos Inteligentes, como se mostró anteriormente los mejores métodos para reconocimiento facial y que fue implementado en el dispositivo móvil por su alto índice de aciertos es el método Local Binary Pattern (LBP) y también el método Eigenfaces

La implementación de los métodos se realizó en java, C, utilizando java para él procesamiento y envió de un vector de datos (imágenes) conjuntamente con un vector de los nombres o identificadores. Los algoritmos fueron implementados en lenguaje C.

5.4.1 Método 1 Patrón Binario Local

Extracción de Características.

Local Binary Pattern se utiliza para la descripción de la cara. El procedimiento consiste en distinguirse la textura del rostro con la finalidad de resaltar las descripciones faciales.

Los patrones binarios locales, son sencillos, rápidos de calcular y muestran un alto número de aciertos con cambios de iluminación (Claire L. Witham, 2017).

El algoritmo también llamado LBP o LBP operador se describe mediante la siguiente formula:

$$LBP(x_C, y_C) = \sum_{P=0}^{P-1} 2^P S(i_p - i_c)$$

Dónde:

$\boldsymbol{x_c, y_c}$ = Es el pixel central

$\boldsymbol{i_c}$ = Intensidad del pixel central en escala de grises

$\boldsymbol{i_p}$ = Intensidad del pixel vecino

$\boldsymbol{S}$ = Función

$$S(x) = \begin{cases} 1, & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Donde el algoritmo para cada pixel de la imagen reconoce 8 pixeles vecinos alrededor del pixel central, teniendo una dimensión de 3x3.

Por lo tanto, se determinó que cada pixel vecino se verifico la intensidad por medio de su valor:

Si el pixel vecino es mayor o igual que el pixel central es 1; si el pixel vecino es menor que el pixel central es 0.

Por ejemplo, como se muestra en la imagen 42 el valor del pixel central es 189 entonces el pixel vecino 192 es mayor por lo que el valor es 1; como se ve en el segundo caso el pixel vecino es 185 por lo tanto el valor es 0 remplazando la misma posición del valor del pixel. De manera que se codifico en forma horaria en una cadena binaria formada de 8 números (Yulian André Cama Castillo, 2015).

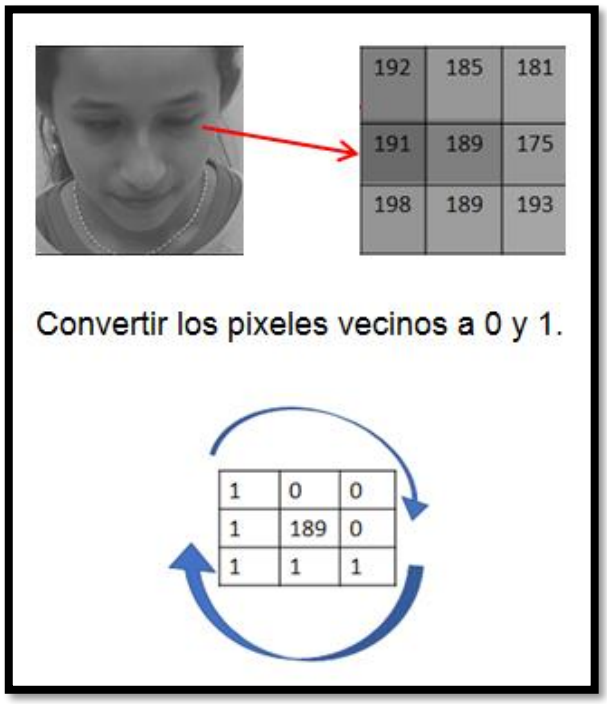


Figura 42: ***Proceso para la creación del operador Patrón Binario Local con una vecindad de 3x3.***

Fuente: Elaboración propia.

Una vez realizado la transformación a binario se calculó el decimal por ejemplo como se visualiza en la imagen 43 teniendo como resultado del ejemplo un decimal de 241 para el mejor desarrollo del algoritmo.

1	0	0	0	1	1	1	1
1	2	4	8	16	32	64	128

LBP = 1+16+32+64+128=241

Figura 43: ***Conversión de los vecinos 3x3 a decimales.***

Fuente: Elaboración propia

Para la implementación de la fórmula matemática se realizó el siguiente procedimiento explicado en la Figura 44.

```
Inicio
        Entrada matriz;
        Generar matriz_LBP [filas_matriz - 2, columna_matriz - 2];
        Para (i = 1 ; i < filas_matriz - 1; Aumentar i) Hacer
                Para (j = 1; j < columnas_matriz ; aumentar j) Hacer
                        pixel_centro = elemento_matriz[f,c];
                        codigo_LBP = 0;
                        codigo_LBP |= (elemento_matriz(i-1,j-1) >= pixel_centro ) << 7;
                        codigo_LBP |= (elemento_matriz(i-1,j) >= pixel_centro ) << 6;
                        codigo_LBP |= (elemento_matriz(i-1,j+1) >= pixel_centro ) << 5;
                        codigo_LBP |= (elemento_matriz(i,j+1) >= pixel_centro ) << 4;
                        codigo_LBP |= (elemento_matriz(i+1,j+1) >= pixel_centro ) << 3;
                        codigo_LBP |= (elemento_matriz(i+1,j) >= pixel_centro ) << 2;
                        codigo_LBP |= (elemento_matriz(i+1,j-1) >= pixel_centro ) << 1;
                        codigo_LBP |= (elemento_matriz(i,j-1) >= pixel_centro ) << 0;
                        elemento_matriz_LBP[ i, j ] = codigo_LBP ;
                Fin Para
        Fin Para
Fin
```

Figura 44: ***Pseudocódigo para la generación de los Patrones Binarios Locales.***

Fuente: Elaboración propia.

El pseudocódigo descrito esta implementado en la siguiente Figura 45 que se muestra, visualizando la implementación de la fórmula matemática, la cual esta comentado las acciones programadas.

```
template <typename _Tp> static
void olbp_(InputArray _src, OutputArray _dst) {
   // obtener matrices
   Mat src = _src.getMat();
   // asignar memoria para resultado
   _dst.create(src.rows-2, src.cols-2, CV_8UC1);
   Mat dst = _dst.getMat();
   // cero la matriz de resultados
   dst.setTo(0);
   // calcular patrones
   for(int i=1;i<src.rows-1;i++) {
      for(int j=1;j<src.cols-1;j++) {
         _Tp center = src.at<_Tp>(i,j);
         unsigned char code = 0;
         code |= (src.at<_Tp>(i-1,j-1) >= center) << 7;
         code |= (src.at<_Tp>(i-1,j) >= center) << 6;
         code |= (src.at<_Tp>(i-1,j+1) >= center) << 5;
         code |= (src.at<_Tp>(i,j+1) >= center) << 4;
         code |= (src.at<_Tp>(i+1,j+1) >= center) << 3;
         code |= (src.at<_Tp>(i+1,j) >= center) << 2;
         code |= (src.at<_Tp>(i+1,j-1) >= center) << 1;
         code |= (src.at<_Tp>(i,j-1) >= center) << 0;
         dst.at<unsigned char>(i-1,j-1) = code;
      }
   }
}
```

Figura 45: ***Implementación de la fórmula matemática.***

Fuente: Elaboración propia.

Luego la imagen o fotos del rostro se divide en diferentes ventanas, la cual cada ventana genero diferentes histogramas, indicando que cada histograma de las ventanas es único, que se va concatenando en secuencia, con el fin de obtener un vector de histogramas, por lo que guardara la información espacial. Donde el proceso se muestra a continuación:

En la Figura 46. Por ejemplo: es la imagen de entrada de tamaño 128*128. Donde la imagen completa es representada M * N dicha imagen es procesada de manera que se empieza a iterar a través de

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

los pixeles para generar ventanas más pequeñas de toda la imagen que contiene información relevante del rostro.

Figura 46: ***Imagen pre-procesada de entrada.***

Fuente: Elaboración propia.

$$R = M * N$$

R = Imagen completa de 16 384 pixeles

M = Alto de la imagen

N = Ancho de la imagen

En este caso se generaron 8 x 8 ventanas con un total de 64 ventanas generadas de toda la imagen ingresada de M * N = 16 384 pixeles, cada ventana tiene un rango de 16 pixeles en la fila y 16 pixeles en la columna con un total de 256 pixeles por ventana, entonces al multiplicar 64 ventanas por 256 pixeles se obtiene R que es el total de pixeles.

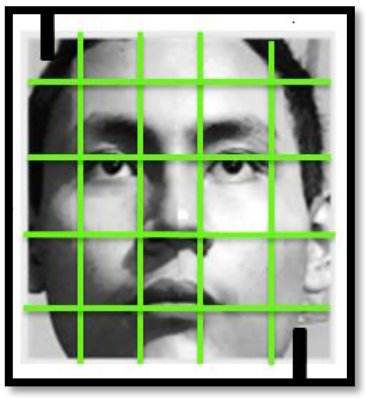


Figura 47: ***Imagen dividido en ventanas.***

Fuente: Elaboración propia.

El cual la imagen está representada en la siguiente fórmula matemática.

$$R = (V = X * Y) * K$$

Dónde:

V = Ventana generada

X = Ancho de la ventana de la imagen

Y = Alto de la ventana de la imagen

K = Numero de Ventanas

R = Foto completa de 128 * 128 pixeles

Las ventanas son generadas como una matriz de izquierda a derecha como se muestra en la Figura 48.

Esto está reflejado en el código que se visualiza a continuación se aplicó dos for para recorrer toda la imagen pixel a pixel

```
    for(int i = 0; i < grid_y; i++) {
            for(int j = 0; j < grid_x; j++) {
```


Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

```
                Mat      src_cell      =      Mat(src,
    Range(i*height,(i+1)*height),
    Range(j*width,(j+1)*width));
            }
    }
```

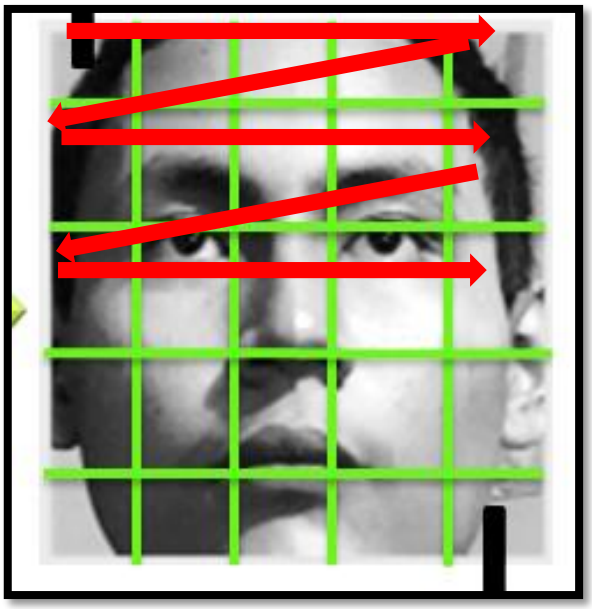


Figura 48: ***Secuencia de generación de ventanas.***

Fuente: Elaboración propia.

Esto con la finalidad de tener un histograma de 8 bits por cada uno de las ventanas donde tiene descripción importante del rostro.

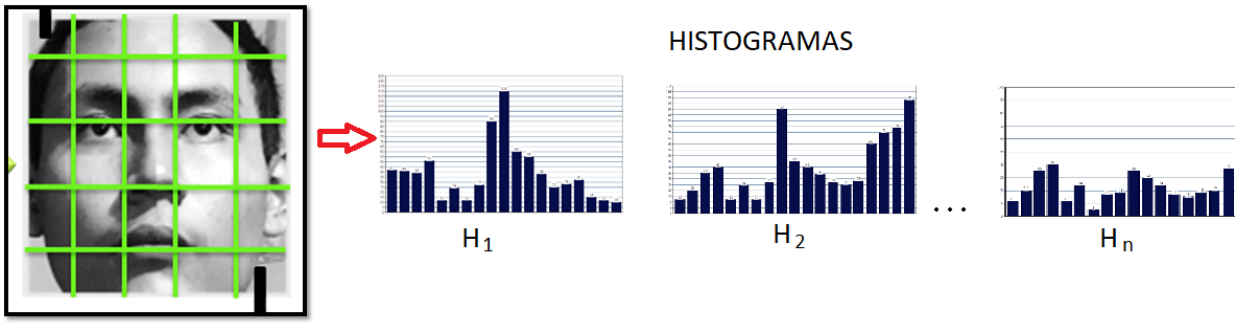


Figura 49: ***Histogramas de ventana.***

Fuente: Elaboración propia.

Donde la Figura 49 se representa en la siguiente función: la ventana 1 es igual al histograma 1, la ventana 2 es igual al histograma 2 y así sucesivamente dependiendo el número de ventanas será igual el número de histogramas.

$$V_1 V_2 \dots V_n = H_1 H_2 \dots H_n$$

Entonces el descriptor de toda la imagen es concatenar todos los histogramas, por lo tanto, si tenemos Vn la dimensión de los histogramas será Hn. El cual el vector de histogramas sirve para comparar los vectores tanto de la imagen de entrada como las ya entrenadas y poder clasificarlos. Este proceso se realiza tanto para la clasificación como para el entrenamiento.

Para ello se tuvo que implementar el método spatial_histogram donde realiza el procedimiento descrito anteriormente. Esto se hizo con la finalidad de que los histogramas espaciales o las ventanas generadas guarden la información resaltante de la textura o puntos importantes del rostro. Para eso la figura siguiente se comenta las principales acciones.

```
    static Mat spatial_histogram(InputArray _src, int
    numPatterns,
                                 int    grid_x,    int
```

```
grid_y, bool)
{
    Mat src = _src.getMat();
    // calcular el tamaño del parche LBP
    int width = src.cols/grid_x;
    int height = src.rows/grid_y;
    // asignar memoria para el histograma espacial
    Mat result = Mat::zeros(grid_x * grid_y,
numPatterns, CV_32FC1);
    // devolver la matriz con ceros si no se
dieron datos
    if(src.empty())
        return result.reshape(1,1);
    // fila de resultado inicial
    int resultRowIdx = 0;
    // iterar a través de la grilla
    for(int i = 0; i < grid_y; i++) {
        for(int j = 0; j < grid_x; j++) {
            //genera las Ventana
            Mat src_cell = Mat(src,
Range(i*height,(i+1)*height),
Range(j*width,(j+1)*width));
//Calcula histograma de la ventana
Mat cell_hist = histc(src_cell, 0, (numPatterns-
1), true);
            // copiar a la matriz de resultados
Mat result_row = result.row(resultRowIdx);

cell_hist.reshape(1,1).convertTo(result_row,
CV_32FC1);
            // aumentar el recuento de filas en la
matriz de resultados
```


Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

```
                resultRowIdx++;
            }
        }
        // resultado de devolución como vector de características reformado
        return result.reshape(1,1);
    }
```

Figura 50: ***Código del método histograma espacial***

Fuente: Elaboración propia

El método donde se implementó todos los procedimientos descritos se llama train donde su finalidad es entrenar a todas las imágenes que se ingresen con su respectiva etiqueta y generar un histograma global por foto para posteriormente realizar una clasificación y mostrar el nombre de la persona que se desea identificar en el sistema. En la siguiente figura esta comentado los pasos importantes que realiza el método.

```
    void LBPH::train(InputArrayOfArrays _in_src, InputArray _in_labels, bool preserveData) {
        if(_in_src.kind() != _InputArray::STD_VECTOR_MAT && _in_src.kind() != _InputArray::STD_VECTOR_VECTOR) {
            String error_message = " Las imágenes se esperan como InputArray::STD_VECTOR_MAT (a std::vector<Mat>) or _InputArray::STD_VECTOR_VECTOR (a std::vector< std::vector<...> >).";
            CV_Error(Error::StsBadArg, error_message);
        }
        if(_in_src.total() == 0) {
            String error_message = format("Se entregaron datos de entrenamiento vacíos.
```

```
Necesitarás más de una muestra para aprender un modelo.");
        CV_Error(Error::StsUnsupportedFormat, error_message);
    } else if(_in_labels.getMat().type() != CV_32SC1) {
        String error_message = format("Las etiquetas se deben dar como un entero (CV_32SC1). Expected %d, but was %d.", CV_32SC1, _in_labels.type());
        CV_Error(Error::StsUnsupportedFormat, error_message);
    }
    // obtener el vector de matrices
    std::vector<Mat> src;
    _in_src.getMatVector(src);
    // obtener la etiqueta de la matriz
    Mat labels = _in_labels.getMat();
    // verificar si los datos están bien alineados
    if(labels.total() != src.size()) {
        String error_message = format("El número de muestras (src) debe ser igual al número de etiquetas (etiquetas). Was len(samples)=%d, len(labels)=%d.", src.size(), _labels.total());

        CV_Error(Error::StsBadArg, error_message);
    }
    // si este modelo debe ser entrenado sin preservar los datos antiguos, elimine los datos del modelo anterior
    if(!preserveData) {
        _labels.release();
```


Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

```
            _histograms.clear();
        }
        // añadir etiquetas a la matriz de etiquetas
        for(size_t labelIdx = 0; labelIdx < labels.total(); labelIdx++) {

_labels.push_back(labels.at<int>((int)labelIdx));
        }
        // almacenar los histogramas espaciales de los datos originales
        for(size_t sampleIdx = 0; sampleIdx < src.size(); sampleIdx++) {
            // calcular la imagen lbp
            Mat lbp_image = olbp(src[sampleIdx], _radius, _neighbors);
            // obtener histograma espacial de esta imagen lbp
            Mat p = spatial_histogram(
                    lbp_image, /* lbp_image */
                    static_cast<int>(std::pow(2.0, static_cast<double>(_neighbors))),
                    _grid_x, /* grid size x */
                    _grid_y, /* grid size y */
                    true);
            // agregar a las plantillas XML
            _histograms.push_back(p);
        }
}
```

Figura 51: ***Código del método train (Entrenamiento).***

Fuente: Elaboración propia

Según la Figura 51, _histogram.push_back(p) se encarga de almacenar los histogramas generados a la matriz de histogramas.

Para el entrenamiento se utilizó la base de datos obtenida anteriormente en la adquisición de imágenes, en diferentes ambientes no controlados y sin ningún uso de objetos en el rostro. Las imágenes entrenadas son en diferentes posiciones como se muestra en la Figura 52.

Figura 52: ***Fotos adquiridas para el entrenamiento***

Fuente: Elaboración propia

Para eso se tuvo que implementar un método para convertirlos de RGB a escala de grises y entrenarlas inmediatamente.

Para el entrenamiento se implementó el método que se muestra en el Pseudocódigo, donde se envía

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

```
Inicio
        imagen = Imagenes_Filtradas;
        Vector_imagenes[total_imagenes];
        Vector_nombres[total_imagenes];
        Contador = 0;
        Para (Archivo imagen : imagenes_filtradas) Hacer
                Cargar_Imagen;
                nombre = Sacar_nombre_de_la_imagen;
                imagen_grey = Convertir_escala_de_grises(imagen);
                Vector_imagenes[contador] = imagen_grey;
                Vector_nombres[contador] = nombre ;
                Aumentar Contador;
                Reconocimiento_facial.Entrenar(Vector_imagenes,Vector_nombres),
        Fin Para
Fin
```

Figura 53: ***Pseudocódigo del método implementado para el entrenamiento***

Fuente: Elaboración propia

Clasificador de rostro.

Para el clasificador se implementó la fórmula de chi-cuadrado según menciona la documentación de OpenCV 3.3.1 este modelo matemático se usa para la comparación de histogramas de textura. Por ello se decidió implementarlo como clasificador por motivo que este clasificador probabilístico no consume muchos recursos y realiza un cálculo muy rápido por lo que unido con LBP se convierte en un método atractivo para un sistema en tiempo real (Javier Eslava Ríos, 2013).

Para este caso de reconocimiento facial el chi-cuadrado se utiliza para calcular la distancia de dos vectores creados a partir de un histograma generado de las imágenes de entrenamiento y la imagen nueva a reconocer.

Para la comparación de dos rostros, se compara dos histogramas con la siguiente formula:

$$d(H_1, H_2) = 2 * \sum_{I} \frac{(H_1(I) - H_2(I))^2}{H_1(I) + H_2(I)}$$

H_1 = Histograma de la imagen entrenada

H_2 =Histograma de la nueva imagen a clasificar

Este método al momento de ingresar la nueva imagen se genera el histograma para realiza una comparación con el histograma de la imagen ya entrenada y predice cuál de las imágenes se parecen más para posteriormente devolver la etiqueta con el nombre registrado.

```
Inicio
        Obtener imagen_entrada;
        histograma_espacial = Obtener el histograma espacial de la imagen_entrada;
        encontrar el vecino más cercano imagen_entrada;
        Para (Muestraid = 0 ; muestra < tamaño_histogramas ; aumentar muestraid ) hacer
                Diferencia = Comparar_Histograma( tamaño_histogramas [Muestraid], histograma_espacial);
                etiqueta = nombre_etiqueta[muestraid];
                retorna etiqueta, diferencia
        Fin Para
Fin
```

Figura 54: ***Pseudocódigo del método implementado***

Fuente: Elaboración propia

El pseudocódigo mostrado en la Figura 54 describe las acciones que se realizan para la clasificación.

Por eso en la Figura 55, se muestra la parte en que primeramente la imagen nueva que ingresa a ser identificada se convierte a una imagen de tipo IplImage y recortando del mismo tamaño de las imágenes entrenadas. Para posteriormente enviar la imagen a predecir y también las variables n que es la que almacenara la etiqueta de la persona reconocida y p la probabilidad de reconocimiento.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

```
IplImage ipl = MatToIplImage(m,WIDTH, HEIGHT);
faceRecognizer.predict(ipl, n, p);
```

Figura 55: ***Código del envió de la imagen a reconocer***

Fuente: Elaboración propia.

Una vez que se envía la imagen al método predict, primeramente, verifica si ya se entrenó las imágenes de entrenamiento, es importante entrenar por lo que se comparó los histogramas de las imágenes entrenadas con la nueva imagen, por eso el código implementado primeramente verifico que se haya realizado el entrenamiento y si no lo es retornara error o -1.

Después se obtiene la imagen espacial y realiza una comparación con los histogramas entrenados. Estas acciones se comentaron las partes importantes del código que se muestra en la Figura 56.

```
void LBPH::predict(InputArray _src,
Ptr<PredictCollector> collector) const {
    if(_histograms.empty()) {
        // Lanzar error si no hay datos (o
simplemente devolver -1?)
        String error_message = "Este modelo LBPH
aún no se ha calculado. ¿Llamaste al método del
tren?";
        CV_Error(Error::StsBadArg, error_message);
    }
    Mat src = _src.getMat();
    // obtener el histograma espacial de la imagen
de entrada
    Mat lbp_image = olbp(src, _radius,
_neighbors);
    Mat query = spatial_histogram(
            lbp_image,
```

```
            static_cast<int>(std::pow(2.0,
static_cast<double>(_neighbors))),  /*  número  de
patrones posibles */
            _grid_x, /* tamaño de cuadrícula x */
            _grid_y, /* tamaño de cuadrícula y */
            true);
    // encontrar el vecino más cercano
    collector->init((int)_histograms.size());
    for  (size_t  sampleIdx  =  0;  sampleIdx  <
_histograms.size(); sampleIdx++) {
        //Esta  parte  es  donde  se  realiza  la
comparación de los histogramas
        double                dist                =
compareHist(_histograms[sampleIdx],          query,
HISTCMP_CHISQR_ALT);
        int                label                =
_labels.at<int>((int)sampleIdx);
       //obtiene la etiqueta y la probabilidad
        if             (!collector->collect(label,
dist))return;
    }
}
```

Figura 56: ***Código del método predict***

Fuente: Elaboración propia

Para calcular la distancia entre histogramas se utilizó la fórmula del chi-cuadrado. Parte de la formula implementada se visualiza en la siguiente figura.

Previamente se va a describir el código de la figura de la operación matemática. Primero se realizó la resta de los vectores de histogramas

desde la primera posición elevado al cuadrado hasta la última, luego dividido entre la sumatoria de los histogramas, dicho resultado se va sumando con la finalidad de hallar la distancia de los dos histogramas comparados. Una vez obtenidos el resultado de la operación se multiplico dos veces el mismo resultado obteniendo la distancia de los histogramas.

```
if ((method == CV_COMP_CHISQR) || (method ==
CV_COMP_CHISQR_ALT)) {
            for (j = 0; j < len; j++) {
                double a = h1[j] - h2[j];
                double b = (method ==
CV_COMP_CHISQR) ? h1[j] : h1[j] + h2[j];
                if (fabs(b) > DBL_EPSILON) {
                    result += a * a / b;
                }
            }
        }

if (method == CV_COMP_CHISQR_ALT) {
            result *= 2;
 }
```

Figura 57: ***Código de la fórmula matemática de Chi-cuadrado***

Fuente: Elaboración propia

Una vez que se obtuvo la distancia de los histogramas se ingresa al método collect la etiqueta y la distancia. Aquí se verifico que la distancia sea menor que el threshold (umbral) y ve si la distancia del resultado es menor que el mínimo de la distancia, si es así retorna el resultado predicho.

```
bool StandardCollector::collect(int label,
double dist) {
```


Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

```
        if (dist < threshold)
        {
            PredictResult res(label, dist);
            if (res.distance < minRes.distance)
                minRes = res;
            data.push_back(res);
        }
        return true;
    }
```

Figura 58: ***Método collect.***

Fuente: Elaboración propia

Para realizar la clasificación de rostros se utilizó la muestra entrenada anteriormente para los métodos a comparar. Para esto se tuvo que capturar una nueva imagen del rostro pero que no fue almacenada solo se utilizó para el proceso de clasificación y lograr reconocer el nombre de la persona.

Para este paso se utilizó la misma interfaz de adquisición de imágenes, pero con la diferencia que esta vez no se realizara el entrenamiento sino se ara la búsqueda de la persona. Adquiriendo la nueva imagen y en cuestión de milisegundos obtener los resultados como el nombre de la persona y la probabilidad. Esto se visualiza en la Figura 59, 60 y 61. Para eso también la foto adquirida en ambiente no controlado con 50 cm de distancia de la cámara al rostro, esto se utilizó para la comparación de los diferentes métodos.

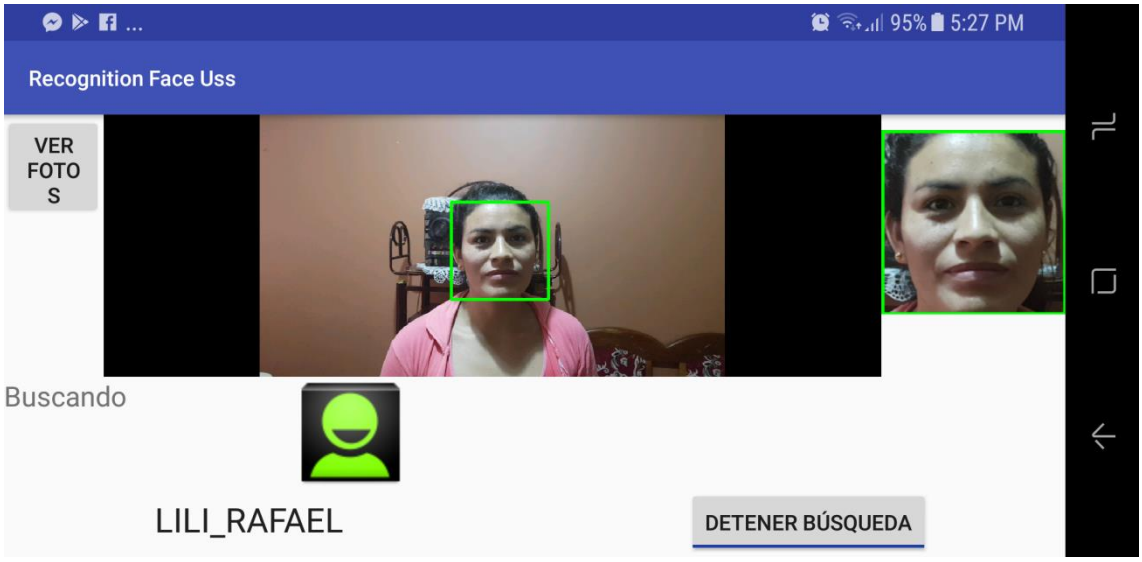


Figura 59: ***Clasificación de rostro frontal***

Fuente: Elaboración propia

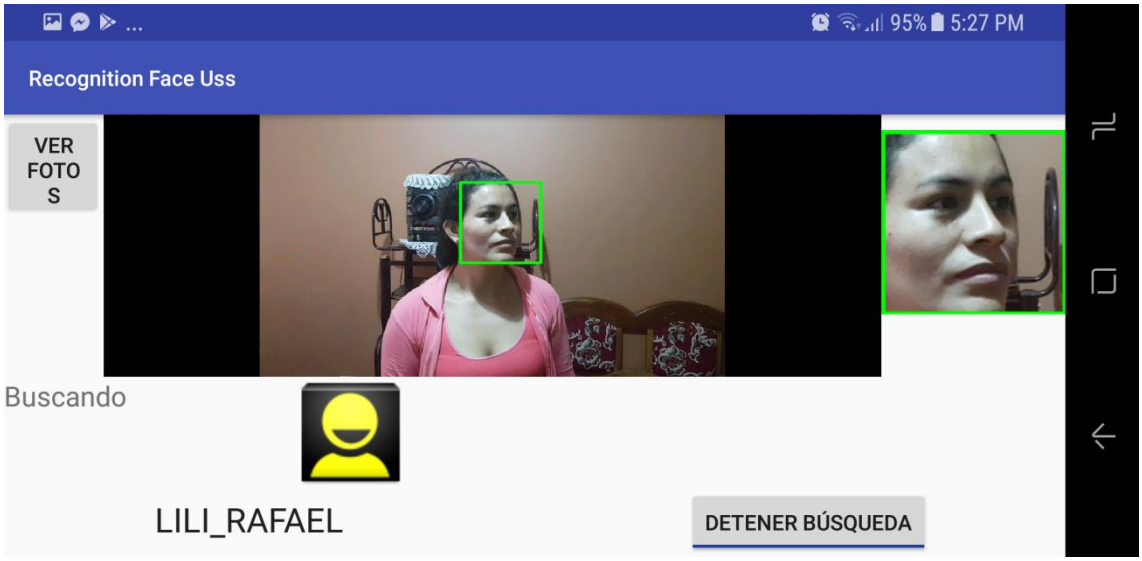


Figura 60: ***Clasificación de rostro horizontal.***

Fuente: Elaboración.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

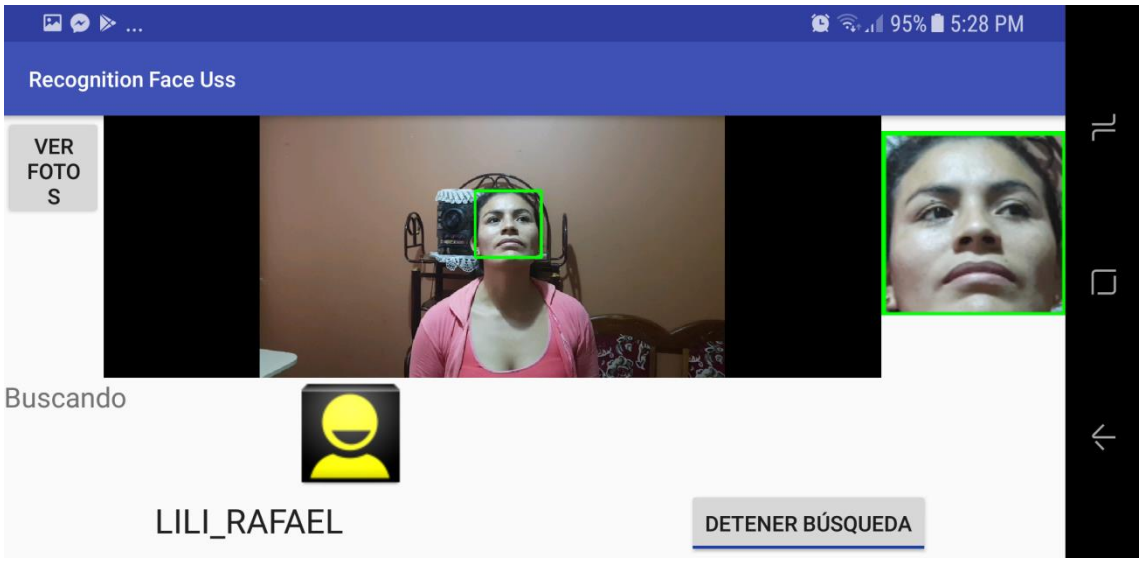


Figura 61: ***Clasificación de rostro vertical***

Fuente: Elaboración propia

El primer método con el extractor de características LBP y clasificador chi-cuadrado que se implementó en el prototipo logro clasificar correctamente de manera frontal, horizontal y vertical como se muestra en las figuras anteriores y logro un desempeño muy óptimo logrando reconocer en milisegundos a la persona.

5.4.2 Método 2 EigenFace

Extracción de características

Eigenfaces adoptó un enfoque holístico para el reconocimiento facial, el cual es una imagen facial es un punto de un espacio de imagen de alta dimensión y se localiza una representación de menor dimensión, donde la clasificación es más fácil.

El subespacio de dimensiones inferiores se encuentra con Análisis de Componentes Principales (PCA), que identifica los ejes con varianza máxima. Por lo tanto, se aplicó una proyección de clase específica con un análisis discriminante lineal al reconocimiento facial.

Teniendo como objetivo capturar las variaciones existentes entre las diferentes imágenes de rostros.

Para el reconocimiento de rostro se desea encontrar las componentes principales de la distribución de rostros, o los vectores propios de la matriz de covarianza del conjunto de imágenes de entrenamiento, considerando que una imagen de la base de datos es vector en un espacio dimensional muy alto. Los vectores propios son establecidos, cada uno representando una diferente cantidad de variación entre los rostros (Edwin Arturo Vega Aquino, 2011).

Primeramente, se calcula la media μ

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Dónde:

n: Total de pixeles de la imagen.

i: La posición del pixel.

x: Representa el pixel.

Primeramente, la imagen en escala de grises es una matriz de 1 canal, la cual lo utilizamos los valores de cada pixel para hallar la media. Según la fórmula matemática la media se calcula sumando todos los pixeles de la imagen entre el total de pixeles sumados. Ejemplo: En la siguiente matriz de 3x3 existen diferentes intensidades de pixeles.

75	67	98
45	56	42
51	36	88

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Figura 62 ***Matriz de intensidades de pixeles.***

Fuente: Elaboración propia

Se aplica la fórmula de la media a la matriz

$$\mu = \frac{1}{9} 75 + 67 + 98 + 45 + 56 + 42 + 51 + 36 + 88$$

$$\mu = \frac{1}{9}(558)$$

$$\boldsymbol{\mu = 62}$$

Teniendo como la media de toda la matriz 62 el cual sirve para el cálculo de la covarianza.

Para visualizar la variabilidad de los datos y la información relativa se estima la matriz de covarianza S

$$S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T$$

Dónde:

μ = Media calculada de la matriz.

x = Pixel de la imagen.

i = Posición del pixel.

n = Total de Pixeles

Donde la matriz en **i** representa el comportamiento de la covarianza en el vector. Por ejemplo: se aplicó el cálculo de la covarianza para la matriz de la imagen 62 donde se efectúa el desarrollo de la formula.

$$S = \frac{1}{9}[[75-62][75-62] + [67-62][67-62] + [98-62][98-62] + [45-62][45-62] + [56-62][56-62] + [42-62][42-62] + [51-62][51-62] + [36-62][36-62] + [88-62][88-62]]$$

$$S = \frac{1}{9}[13][13] + [5][5] + [36][36] + [-17][-17] + [-6][-6] + [-20][-20] + [-11][-11] + [-26][-26] + [26][26]$$

$$S = \frac{1}{9}[169 + 25 + 1296 + 289 + 36 + 400 + 121 + 676 + 676]$$

$$S = \frac{1}{9}[3688]$$

$$S = 409.8$$

El resultado de la matriz de covarianza de S es 409.8. Una vez realizado la covarianza de S. Se procede a realiza el cálculo de los valores propios y vectores propios de *S.*

$$Svi = \lambda_i v_i \quad , \quad i = 1,2,3, \dots n$$

Dónde:

λ = Lambda

v_i = Vectores propios

Por ejemplo: donde S es el resultado de la covarianza y v_i es los vectores propios

$$[409.8]\begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 363.1\ x \\ 363.1\ y \end{bmatrix} = \begin{bmatrix} \lambda x \\ \lambda y \end{bmatrix}$$

$$363.1\ x = \lambda x$$

$$363.1\ y = \lambda y$$

$$\lambda_1 = 363.1 \qquad \lambda_2 = 363.1$$

Se ordena los autovectores que provienen por su valor propio. Los K componentes principales son los vectores propios pertinente a los K valores propios más grandes.

Crear W de forma que sus columnas son los vectores propios correspondientes a los valores propios más relevantes eso significa con mayor energía (Roger Gimeno Hernández, 2010)

Los *K* principales componentes del vector observado *x* están representada por:

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

$$y = w^T(x - u)$$

Dónde:

$W = \nu_1\nu_2\nu_3 \dots \nu_K$

μ = Media

Se despeja la formula damos como por ejemplo: que W se multiplica por el vector observado dando $w^T x$ y a la vez también W se multiplica por el negativo de la media dando $-w^T u$

$$y = w^T x - w^T u$$

$$wy = ww^T x - ww^T u$$

$$wy = x - u$$

$$wy + u = x$$

$$\boldsymbol{x = wy + u}$$

Una vez que se obtuvo el resultado $x = wy + u$ que es la reconstrucción de PCA, donde W sigue siendo los vectores propios.

El cual el método EigenFaces realiza el reconocimiento facial proyectando la muestra de imágenes de entrenamiento en el subespacio de PCA.

De igual manera la imagen consultada es proyectada en el subespacio de PCA. Para posteriormente el clasificador identifique el rostro consultado.

Las fórmulas matemáticas fueron implementadas en el método train() el cual recibe la imagen que fue almacenada en la base de datos de

entrenamiento y también el nombre o etiqueta de la persona, el cual esta descrito paso a paso la figura siguiente del método train()

```
void Eigenfaces::train(InputArrayOfArrays _src,
InputArray _local_labels) {
    if(_src.total() == 0) {
        String error_message = format("Empty training
data was given. You'll need more than one sample to
learn a model.");
        CV_Error(Error::StsBadArg, error_message);
    } else if(_local_labels.getMat().type() !=
CV_32SC1) {
        String error_message = format("Labels must be
given as integer (CV_32SC1). Expected %d, but was
%d.", CV_32SC1, _local_labels.type());
        CV_Error(Error::StsBadArg, error_message);
    }
    // Asegúrese de que los datos tengan el tamaño
correcto
    if(_src.total() > 1) {
        for(int i = 1; i <
static_cast<int>(_src.total()); i++) {
            if(_src.getMat(i-1).total() !=
_src.getMat(i).total()) {
                String error_message = format("¡En el
método Eigenfaces todas las muestras de entrada
(imágenes de entrenamiento) deben ser del mismo
tamaño! Expected %d pixels, but was %d pixels.",
_src.getMat(i-1).total(), _src.getMat(i).total());
                CV_Error(Error::StsUnsupportedFormat,
error_message);
            }
        }
```


Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

```
    }
    // Obtener etiquetas
    Mat labels = _local_labels.getMat();
    // observaciones en fila
    Mat data = asRowMatrix(_src, CV_64FC1);

    // número de muestras
   int n = data.rows;
    // afirman que hay tantas muestras como etiquetas
    if(static_cast<int>(labels.total()) != n) {
        String error_message = format("The number of
samples (src) must equal the number of labels
(labels)!    len(src)=%d,    len(labels)=%d.",    n,
labels.total());
        CV_Error(Error::StsBadArg, error_message);
    }
    // borrar los datos del modelo existente
    _labels.release();
    _projections.clear();
    // cantidad de clips de componentes para ser
válida
    if((_num_components <= 0) || (_num_components >
n))
        _num_components = n;

    // Realizar el PCA
    pca(data,        Mat(),        PCA::DATA_AS_ROW,
_num_components);
    // Copiar los resultados de PCA
    _mean = pca.mean.reshape(1,1); // Almacenar el
vector malo
    _eigenvalues   =   pca.eigenvalues.clone();   //
```


Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

```
    Valores propios por fila
        transpose(pca.eigenvectors,  _eigenvectors);  //
    eigenvectores por columna
        // Almacenar etiquetas para predicción
        _labels = labels.clone();
        // guardar proyecciones
        for(int  sampleIdx  =  0;  sampleIdx  <  data.rows;
    sampleIdx++) {
            Mat  p  =  LDA::subspaceProject(_eigenvectors,
    _mean, data.row(sampleIdx));
            _projections.push_back(p);
        }
    }
```

Figura 63: ***Código de entrenamiento para el método EigenFaces***

Fuente: Elaboración propia

Para la fase de entrenamiento utilizo la misma cantidad de imágenes y las mismas fotos de rostros de ambos métodos por motivo que fue el mismo clasificador en cascada.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

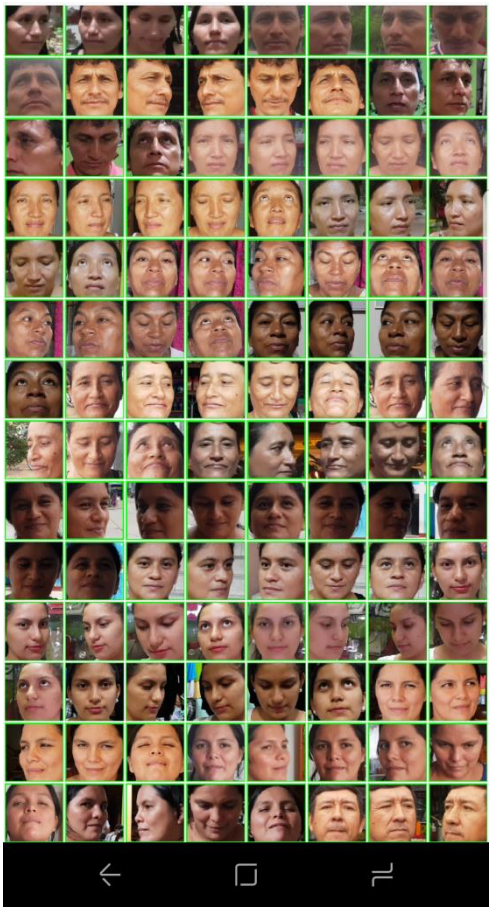


Figura 64: ***Base de datos de entrenamiento del método EigenFaces***

Fuente: Elaboración propia

Considerando que a la hora de realizar el entrenamiento al momento de entrenar las imágenes cuando la base de datos ya contenía las 750 fotos de las 50 personas la aplicación tarda aproximadamente 2 minutos en entrenarlas todas. Hasta en caso contrario la aplicación tarda buen tiempo que el dispositivo emite una alerta que la aplicación dejo de funcionar, esto solo ocurrió cuando ya se tenía toda la base de entrenamiento.

Clasificador de rostro

Para reconocer la imagen del nuevo rostro que se ingresó se tiene que obtener la distancia euclidiana de cada vector de la matriz de PCA de

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

la base de datos de entrenamiento. La siguiente distancia se obtiene aplicando la siguiente formula:

$$D_k = \|\Omega - \Omega_k\|^2$$

D = Distancia euclidiana obtenida.

Ω_k = Posición de la muestra

Ω = Muestra nueva

Se obtiene tantas distancias posibles dependiendo el tamaño de la base de datos de imágenes.

Para la clasificación la menor distancia euclidiana será la imagen a la que más se asemeje en la base de datos, y toma la posición de la imagen para identificar el nombre de la persona.

Para la clasificación del rostro se ingresa un nuevo rostro diferente a las fotos entrenadas, pero si con el mismo tamaño que se entrenó, el cual para recibir la nueva imagen donde se implementa la fórmula matemática el método predict(), en el presente método primeramente se obtiene la matriz de la imagen, luego se verifica que exista una matriz de imágenes ya entrenadas, de no ser así no se puede realizar una clasificación si no existe imágenes entrenadas.

```
void Eigenfaces::predict(InputArray _src,
Ptr<PredictCollector> collector) const {
    // Obtener datos
    Mat src = _src.getMat();
    // asegúrese de que el usuario esté pasando los
datos correctos
    if(_projections.empty()) {
        // lanzar error si no hay datos (¿o
```


Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

```
simplemente devolver -1?)
        String error_message = " Este modelo de
Eigenfaces aún no se ha calculado. ¿Llamaste a
Eigenfaces :: train?";
        CV_Error(Error::StsError, error_message);
    }       else       if(_eigenvectors.rows       !=
static_cast<int>(src.total())) {
        // Verificar la alineación de datos solo para
mensajes de excepción más claros
        String error_message = format("Tamaño de
imagen de entrada incorrecto. Motivo: las imágenes de
entrenamiento y prueba deben ser del mismo tamaño.
Expected an image with %d elements, but got %d.",
_eigenvectors.rows, src.total());
        CV_Error(Error::StsBadArg, error_message);
    }
    // proyecto en el subespacio de PCA
    Mat   q   =   LDA::subspaceProject(_eigenvectors,
_mean, src.reshape(1, 1));
    collector->init(_projections.size());
    for   (size_t   sampleIdx   =   0;   sampleIdx   <
_projections.size(); sampleIdx++) {
        double dist = norm(_projections[sampleIdx],
q, NORM_L2);
        int label = _labels.at<int>((int)sampleIdx);
        if (!collector->collect(label, dist))return;
    }
}
```

Figura 65: Método predict

Fuente: Elaboración propia

Para hallar la matriz del subespacio de PCA se implementó el método subspaceProject() el cual esta descrito paso a paso en la Figura 66.

```
Mat LDA::subspaceProject(InputArray _W, InputArray
_mean, InputArray _src) {
    // Obtener matrices de datos
    Mat W = _W.getMat();
    Mat mean = _mean.getMat();
    Mat src = _src.getMat();
    // obtener el número de muestras y la dimensión
    int n = src.rows;
    int d = src.cols;
    // asegúrese de que los datos tengan la forma
correcta
    if(W.rows != d) {
        String error_message = format("Wrong shapes for
given matrices. Was size(src) = (%d,%d), size(W) =
(%d,%d).", src.rows, src.cols, W.rows, W.cols);
        CV_Error(Error::StsBadArg, error_message);
    }
    // asegúrese de que la media sea correcta si no
está vacía
      if(!mean.empty() && (mean.total() != (size_t) d))
  {
          String error_message = format("Wrong mean
  shape for the given data matrix. Expected %d, but was
  %d.", d, mean.total());
          CV_Error(Error::StsBadArg, error_message);
      }
      // crear matrices temporales
      Mat X, Y;
      // asegúrese de operar con el tipo correcto
      src.convertTo(X, W.type());
```


Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

```
    // seguro de hacer, debido a la afirmación
anterior
    if(!mean.empty()) {
        for(int i=0; i<n; i++) {
            Mat r_i = X.row(i);
            subtract(r_i, mean.reshape(1,1), r_i);
        }
    }
    //finalmente calcule la proyección como Y = (X -
media) * W
    gemm(X, W, 1.0, Mat(), 0.0, Y);
    return Y;
}
```

Figura 66: Método subspaceProject

Fuente: Elaboración propia

Dicho código en la figura anterior se retorna la imagen proyectada para posteriormente realizar el cálculo de la distancia euclidiana con las imágenes en la base de datos logrando mostrar a que persona se aparece más según la menor distancia que se obtenga de todas las fotos de rostro entrenadas.

El método de Eigenfaces se utilizó la misma muestra o fotos adquiridas con el clasificador en cascada LBPCascade ya que el método Eigenfaces también utiliza el mismo clasificador, el segundo método implementado demando de mucho más tiempo al momento de reconocer y teniendo un índice de reconocimiento más bajo que el primer método.

CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

Inicialmente el uso del algoritmo desarrollado LBPCASCADE_FRONTALFACE permitió realizar correctamente la detección del rostro de manera frontal en ambos métodos, principalmente LBP logro en la noche obtener el 100% de acierto y un falso negativo y falso positivo del 0%, por otro lado, de manera vertical se desarrolló eficientemente en los diferentes escenarios teniendo como porcentaje para LBP el 90% de reconocimiento durante la noche y el 85% para el método EigenFaces. Sin embargo, se notó que el algoritmo no soporta una iluminación excesiva o rotación del rostro más de los 60°, por lo que si se desea mejorar el sistema de reconocimiento facial se debe desarrollar un algoritmo más robusto en tema de orientación de rostro de manera horizontal.

En la fase de selección, se concluye que el método LBP es superior en tiempo de respuesta de tal modo que obtuvo durante la mañana 0.456 segundos considerando a diferencia de EigenFaces un buen tiempo de respuesta de tal modo que EigenFaces demoro 0.761 segundos. El método LBP cumple con las principales características y necesidades para la implementación en dispositivos inteligentes, logrando a diferencia de otros métodos un tiempo de respuesta aceptable.

En la siguiente fase de implementación el método LBP para el tema de reconocimiento facial en dispositivos móviles mostro un consumo computacional durante la mañana 15.65 MB, en la tarde 15.20 MB y en la noche 15.97 MB, mientras para el caso de EigenFaces durante la mañana 16.19 MB, en la tarde 16.05 y en la noche 16.53 MB. Esto significa que, LBP consume menos recursos computacionales obteniendo un consumo aceptable. También se concluye que, si se disminuye la dimensionalidad de imagen, ya que existe información irrelevante se logra reducir el consumo computacional por lo que hace viable la aplicación.

Finalmente, por experimento se logró tener resultados altos y bajos, de manera que los resultados de acierto se logró tener un alto índice en los diferentes escenarios probados, logrando hasta un alto porcentaje de reconocimiento de manera vertical teniendo para LBP el más alto índice en la noche con 90% de acierto con orientación hacia arriba y en el caso de EigenFaces también obtuvo resultados similares con el 85 % de acierto durante la noche con orientación hacia arriba. Esto significa que ambos métodos se asemejan en orientación vertical; y los resultados bajos fue por motivo del identificador que no soportaba los 90° de orientación del rostro, teniendo un falso negativo del 100% en ambos métodos.

6.2 Recomendaciones

Con la finalidad de mejorar el protocolo de adquisición de imágenes sería recomendable que se pueda combinar el algoritmo LBPCASCADE_FRONTALFACE Y LBPCASCADE_PROFILEFACE de manera que se pueda probar si puede identificar tanto frontal como de perfil.

Se recomienda tener más fotos de entrenamiento de rostros a 90° grados, por lo que se desea mejorar el índice de reconocimiento facial de perfil para que el algoritmo LBP cumpla con todas las perspectivas de un algoritmo de reconocimiento facial.

Por otro lado, surge la necesidad cuando no existe una iluminación que enfoque algunas características del rostro, por lo que se recomienda trabajar en los nuevos dispositivos inteligentes que tienen sensores infrarrojos con la condición de realizar nuevos experimentos a explorar.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Referencias

Akariman, Q., Jati, A. N., & Novianty, A. (2015). Face recognition based on the Android device using LBP algorithm. ICCEREC 2015 - International Conference on Control, Electronics, Renewable Energy and Communications, 166–170.

Claire L. Witham (2017). Automated face recognition of rhesus macaques. Journal of Neuroscience Methods.

Bellakhdhar, F., Loukil, K., & Abid, M. (2012). SVM Classification for Face Recognition, 3(4), 178–184.

Bertok, K., & Fazekas, A. (2016). Face recognition on mobile platforms. 2016 7th IEEE International Conference on Cognitive Infocommunications.

Gofman, M. I., & Mitra, S. (2016). Multimodal biometrics for enhanced mobile device security. Communications of the ACM, 59(4), 58–65.

Lee, Y. H., Kim, C. G., Kim, Y., & Whangbo, T. K. (2013). Facial landmarks detection using improved active shape model on android platform. Multimedia Tools and Applications, 1–10.

Stoimenov, S., Tsenov, G. T., Mladenov, V. M., & Member, S. (2016). Face Recognition system in Android Using Neural Networks. Symposium on Neural Networks and Applications (NEUREL), 13–16.

Moreno Diaz Ana (2004). Reconocimiento Facial Automático Mediante Técnicas de Visión. Universidad Politécnica De Madrid En La Facultad De Informática.

David Eduardo Espinoza Olguín, Peter Ignacio Jorquera Guillen (2015). Reconocimiento Facial. Pontificia Universidad Católica De Valparaíso Facultad De Ingeniería Escuela De Ingeniería Informática.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Villa Palacios Sandra (2008). Sistema Automático de Reconocimiento de Rostro. Universidad Peruana de Ciencias Aplicadas.

Yulian André Cama Castillo (2015). Prototipo computacional para la detección y clasificación de expresiones faciales mediante extracción de Patrones Binarios Locales. Pontificia Universidad Católica del Perú.

José Jaime Esqueda Elizondo, Luis Enrique Palafox Maestre (2005). Fundamentos de procesamiento de imágenes. Universidad Autónoma de Baja California.

Ortega García, J., Alonso Fernández, F. and Coomonte Belmonte, R. (2008). Biometría y seguridad. Madrid: Fundación Rogelio Segovia para el Desarrollo de las Telecomunicaciones.

Serratosa, F. (2008). La biometría para la identificación de las personas. Universitat Oberta de Catalunya.

Phillips, P. J., Grother, P., Micheals, R., Blackburn, D. M., Tabassi, E., & Bone, M. (2003). Face recognition vendor test 2002. In Analysis and Modeling of Faces and Gestures, 2003. AMFG 2003. IEEE International Workshop.

Brahnam, S., Jain, L. C., Nanni, L., & Lumini, A. (2014). Local binary patterns: new variants and applications. Springer Berlin Heidelberg.

Emami, S. and Levgen, K. (2012). Mastering OpenCV with Practical Computer Vision Projects. Birmingham: Packt Pub.

Rodríguez, H. (2009). Imagen digital: conceptos básicos. Marcombo.

Passariello, G., & Mora, F. (1995). Imágenes médicas. Baruta: Equinoccio, Ediciones de la Universidad Simón Bolivar.

García, A. (2012). Inteligencia artificial: fundamentos, práctica y aplicaciones. Rc Libros.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Haugeland, J. (1988). La inteligencia artificial. Siglo XXI.

Jain, A. and Li, S. (2011). Handbook of face recognition. London: Springer.

Guido van Rossum (2009). El tutorial de Python. Python Software Foundation

Eleyan, A., & Demirel, H. (2007). Pca and lda based neural networks for human face recognition. In Face Recognition. InTech.

Erick G. Coronel Castillo (2010). Lenguaje de programacion Java. Macro

Pedro Viñuela & Ines Galvan (2004). Redes de neuronas artificiales: un enfoque práctico. Pearson.

Luis Gomez Chova (2004). Pattern Recognition Methods for Crop Classification from Hyperspectral Remote Sensing Images. Dissertation.com

Edwin Arturo Vega Aquino (2011). Detección Y Seguimiento De Rostros. Universidad Carlos III de Madrid.

Roger Gimeno Hernández (2010). Estudio De Técnicas De Reconocimiento Facial. Universidad Politécnica de Catalunya

Javier Eslava Ríos (2013). Reconocimiento Facial En Tiempo Real. Universidad Autónoma de Madrid.

Pythonware.com (2017). Python Imaging Library (PIL) http://www.pythonware.com/products/pil/

Google Developers. (2017). Face Detection Concepts Overview | Mobile Vision | Google Developers. https://developers.google.com/vision/face-detection-concepts

Android Studio (2017). Conoce Android Studio | Android Studio. Developer.android.com. https://developer.android.com/studio/intro/index.html?hl=es-419

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Anexos

Anexos 1: Integración de la librería OpenCv

Para la integración de las librerías de OpenCv necesariamente se tiene que descargar el paquete completo de OpenCv del enlace http://opencv.org/releases.html

Primeramente, en esta parte se va a describir los pasos de integración de OpenCv. Como se visualiza en la imagen (Anexo Imagen 1), se creó un proyecto en Android Studio con la versión API 26 y la mínima con la API 24. Una vez creado el proyecto en Android se va a Nuevo -> Importar Modulo y una vez hecho eso se va abrir una nueva ventana donde indicaremos la ruta donde está el paquete de OpenCv.

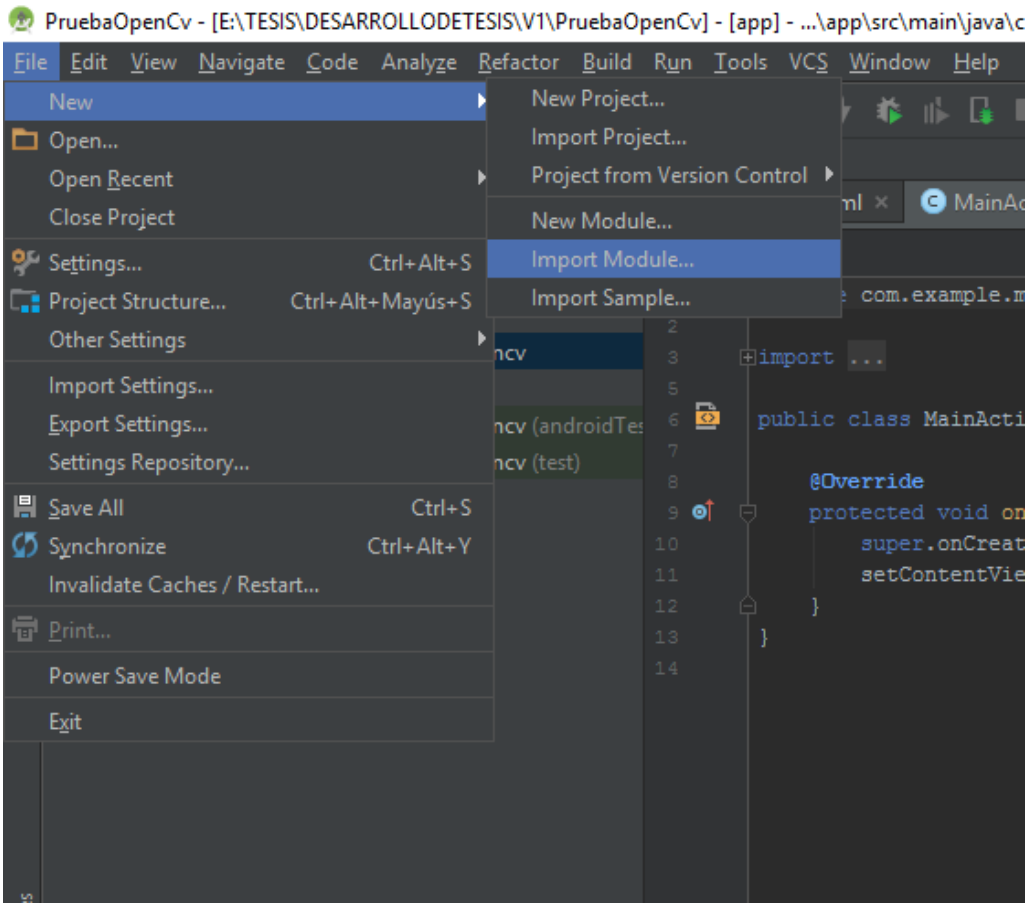


Figura 67: Procedimiento de importación de Modulo OpenCV

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

Fuente: Elaboración propia

Una vez abierto la ventana buscamos la ubicación donde se encuentra las librerías de OpenCV

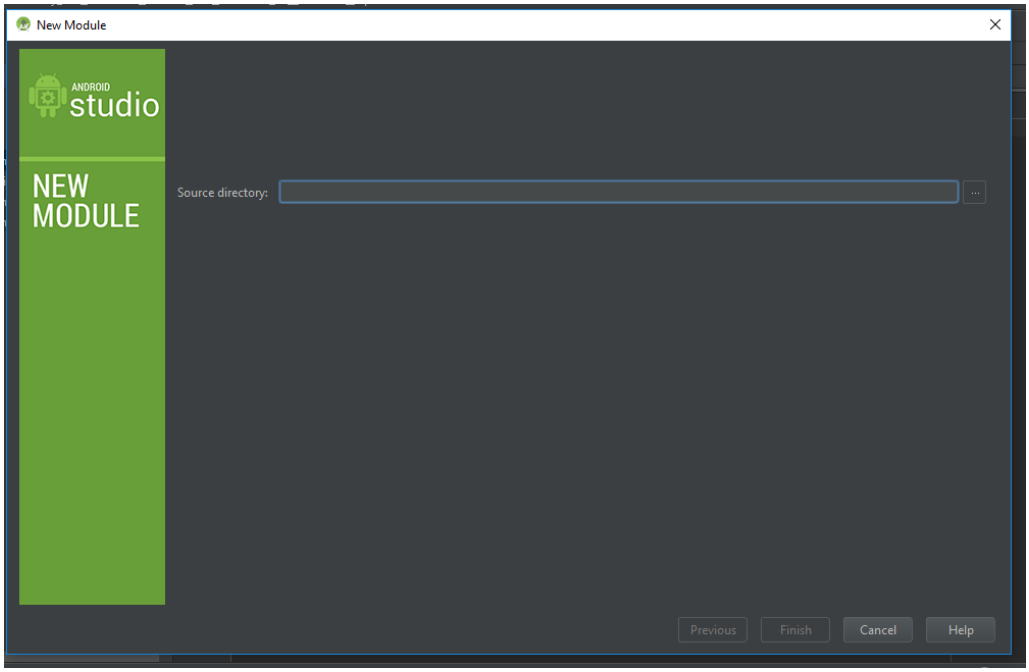


Figura 68: Ruta del módulo OpenCV

Fuente: Elaboración propia

Una vez importado la librería verificamos las versiones para no tener problemas posteriormente

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

```
apply plugin: 'com.android.library'

android {
    compileSdkVersion 14
    buildToolsVersion "25.0.0"

    defaultConfig {
        minSdkVersion 8
        targetSdkVersion 21
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.txt'
        }
    }
}
```

Figura 69: Verificación de las versiones

Fuente: Elaboración propia

Nos dirigimos a la ventana dependencias para agregar el nuevo modulo

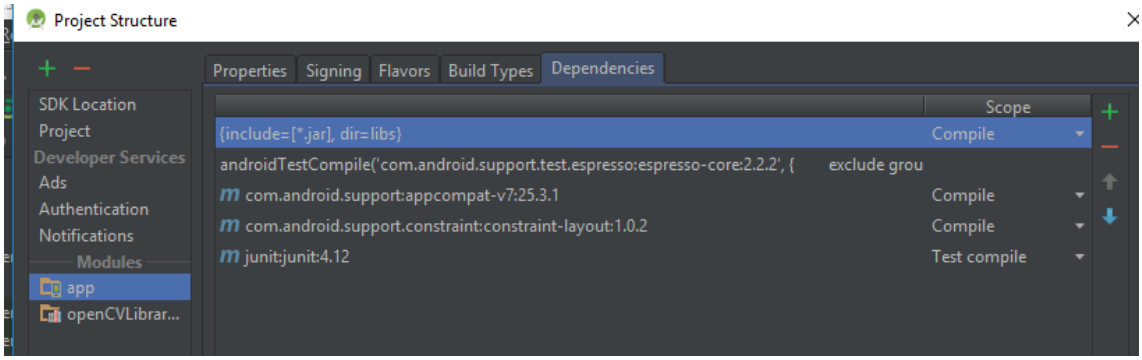


Figura 70: Registro de dependencia del modulo

Fuente: Elaboración propia

Una vez echo clic en el símbolo más, hacemos clic en la opción Molule Dependencia como se muestra en la imagen.

Posteriormente seleccionamos el módulo OpenCV en este caso la versión 3.1.0.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

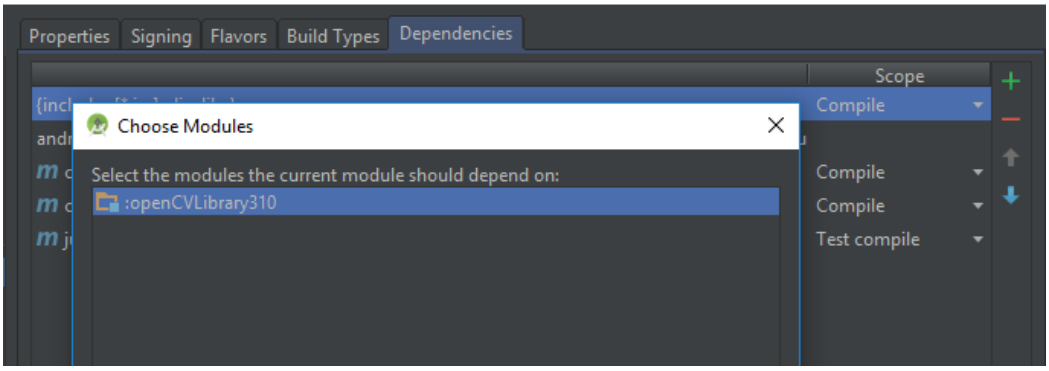


Figura 71: Registro de Dependencia del módulo.

Fuente: Elaboración propia.

Una vez echo lo anterior, ahora si nos dirigimos a la opción Nuevo -> Folder -> JNI Folder para crear lo siguiente.

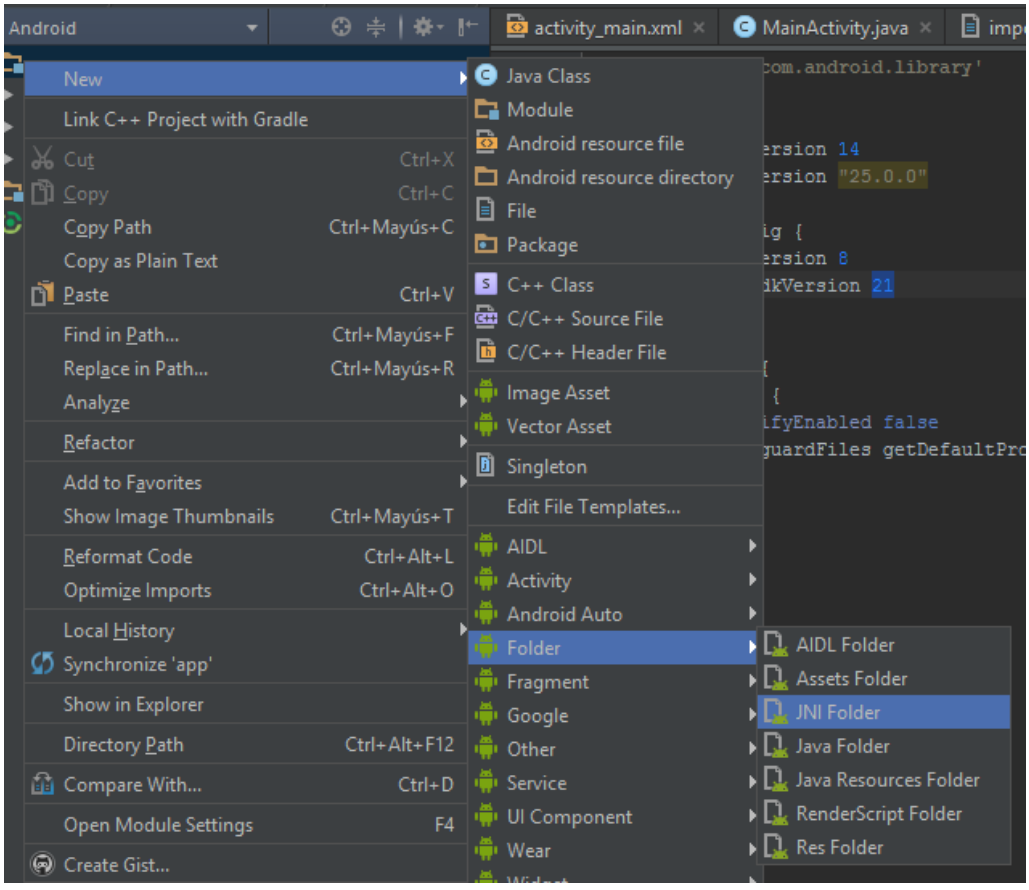


Figura 72: Procedimiento para importar estructura.

Fuente: Elaboración propia.

Ahora creamos la nueva librería o carpeta donde se guardará la estructura de OpenCV para poder utilizar los métodos y también que se puedan ejecutar.

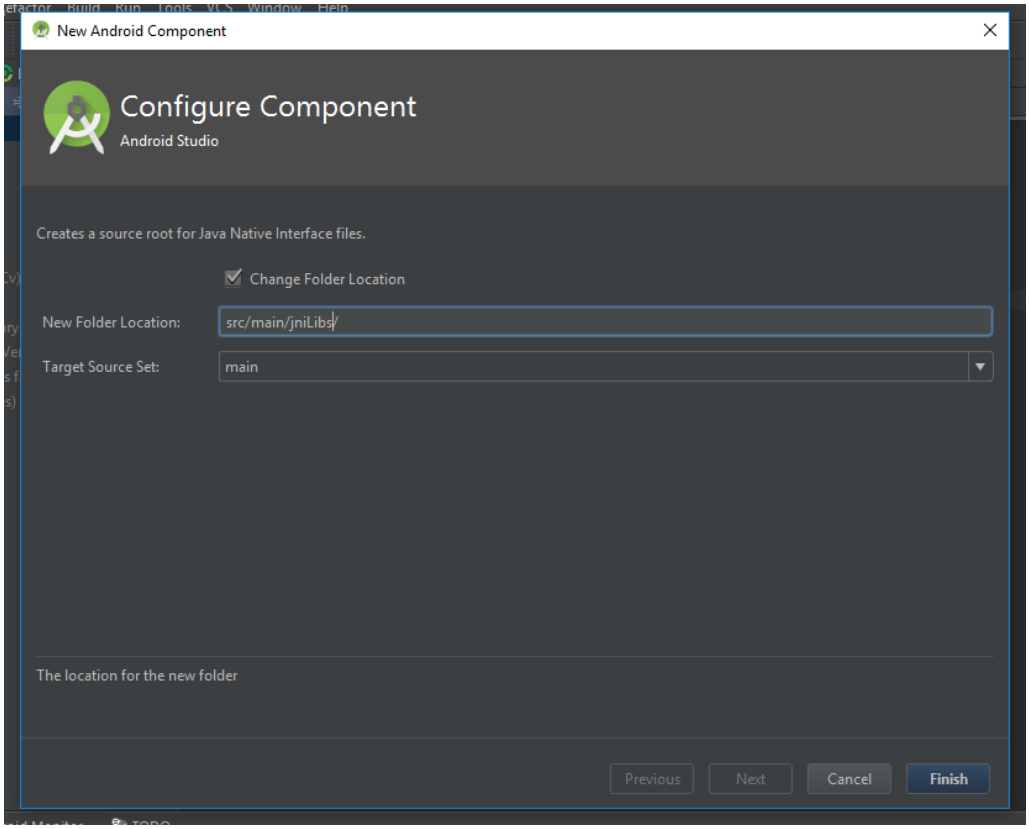


Figura 73: Ruta de ubicación de los archivos Java Native Interface.

Fuente: Elaboración propia.

Una vez creada copiamos las siguientes carpetas dependiendo de la arquitectura del dispositivo móvil para poder utilizar los métodos de OpenCV ya implementados.

Ahora si verificamos si OpenCV está respondiendo con el siguiente método que se muestra en la siguiente imagen.

Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

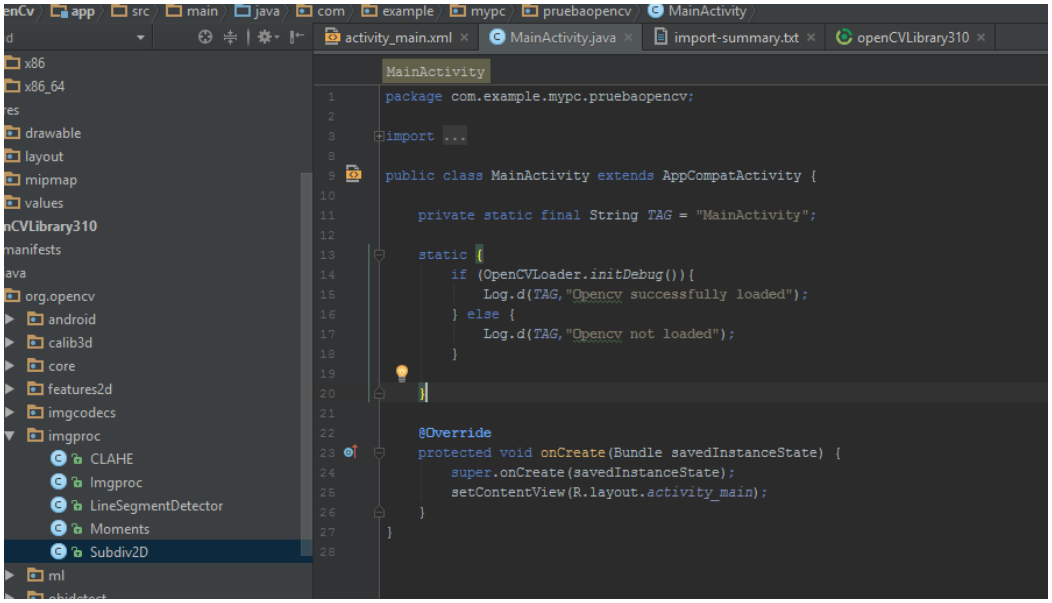


Figura 74: Código para empezar a utilizar OpenCV.

Fuente: Elaboración propia.

Anexo 2: Código fuente del proyecto

Clase reconocimiento facial.

La clase Reconocimiento facial es la que se encarga enviar las imágenes de entrenamiento y recibir los datos de predicción.

```
public final static int MAXIMG = 100;
FaceRecognizer faceRecognizer;
String mPath;
int count=0;
RegistroPersonas registroPersonas;
 static  final int WIDTH= 128;
 static  final int HEIGHT= 128;;
 private int mProb=999;

   ReconocimientoFacial(String path) {
       faceRecognizer =
com.googlecode.javacv.cpp.opencv_contrib.createFisherFaceRecognizer(0
,Double.MAX_VALUE);
       mPath=path;
   registroPersonas = new RegistroPersonas(mPath);
   }
```

```
public void add(Mat m, String description) {
   Bitmap bmp= Bitmap.createBitmap(m.width(), m.height(),
Bitmap.Config.ARGB_8888);

   Utils.matToBitmap(m,bmp);
   bmp= Bitmap.createScaledBitmap(bmp, WIDTH, HEIGHT, false);

   FileOutputStream f;
   try {
      String nombres[] = description.toString().split(" ");
      StringBuilder s = new StringBuilder();
      for (String nom : nombres) {
         s.append(nom);
         s.append("_");
      }
      f = new FileOutputStream(mPath + s + "" + count + ".jpg",true);
      count++;
      bmp.compress(Bitmap.CompressFormat.JPEG, 100, f);
      f.close();


   } catch (Exception e) {
      Log.e("error",e.getCause()+" "+e.getMessage());
      e.printStackTrace();


   }
}

public boolean train() {

   File root = new File(mPath);

       FilenameFilter pngFilter = new FilenameFilter() {
          public boolean accept(File dir, String name) {
             return name.toLowerCase().endsWith(".jpg");

       };
       };

       File[] imageFiles = root.listFiles(pngFilter);
       MatVector images = new MatVector(imageFiles.length);
       int[] labels = new int[imageFiles.length];

       int counter = 0;
       int label;

       IplImage img=null;
```


Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

```
        IplImage grayImg;

        int i1=mPath.length();

        for (File image : imageFiles) {
            String p = image.getAbsolutePath();
            img = cvLoadImage(p);
            if (img==null)
                Log.e("Error","Error al cargar imagen");
            Log.i("image",p);
            int i2=p.lastIndexOf("_");
            int i3=p.lastIndexOf(".");
            int icount=Integer.parseInt(p.substring(i2+1,i3));
            if (count<icount) count++;

            String description=p.substring(i1,i2);
            if (registroPersonas.get(description)<0)
          registroPersonas.add(description, registroPersonas.max()+1);
            label = registroPersonas.get(description);
            grayImg = IplImage.create(img.width(), img.height(),
IPL_DEPTH_8U, 1);
            cvCvtColor(img, grayImg, CV_BGR2GRAY);
            images.put(counter, grayImg);
            labels[counter] = label;
            counter++;
        }
        if (counter>0)
            if (registroPersonas.max()>1) {
                faceRecognizer.train(images, labels);
            }
    registroPersonas.Save();
return true;
}

public boolean canPredict() {
    if (registroPersonas.max()>1)
        return true;
    else
        return false;

}

public String predict(Mat m) {
    if (!canPredict())
        return "";
    int n[] = new int[1];
    double p[] = new double[1];
    IplImage ipl = MatToIplImage(m,WIDTH, HEIGHT);
```


Campus Universitario Km. 5 Carretera a Pimentel - Chiclayo - Perú
Teléf: (+51)(74) 481610 / 481620 - Fax: 203861
www.uss.edu.pe

```
    faceRecognizer.predict(ipl, n, p);

    if (n[0]!=-1)
     mProb=(int)p[0];
    else
       mProb=-1;
// if ((n[0] != -1)&&(p[0]<95))
    if (n[0] != -1)
       return registroPersonas.get(n[0]);
    else
       return "Cara desconocida";
}



  IplImage MatToIplImage(Mat m,int width,int heigth) {
       Bitmap bmp=Bitmap.createBitmap(m.width(), m.height(),
Bitmap.Config.ARGB_8888);
       Utils.matToBitmap(m, bmp);
       return BitmapToIplImage(bmp,width, heigth);
  }

IplImage BitmapToIplImage(Bitmap bmp, int width, int height) {
    if ((width != -1) || (height != -1)) {
       Bitmap bmp2 = Bitmap.createScaledBitmap(bmp, width, height, false);
       bmp = bmp2;
    }
    IplImage image = IplImage.create(bmp.getWidth(), bmp.getHeight(),
IPL_DEPTH_8U, 4);
    bmp.copyPixelsToBuffer(image.getByteBuffer());
    IplImage grayImg = IplImage.create(image.width(), image.height(),
IPL_DEPTH_8U, 1);
    cvCvtColor(image, grayImg, opencv_imgproc.CV_BGR2GRAY);
    return grayImg;
}
```

Figura 75: Clase principal del proyecto de reconocimiento facial.

Fuente: Elaboración propia

Método implementado para el cálculo del histograma

```
static Mat

histc_(const Mat& src, int minVal=0, int maxVal=255,
```

```
bool normed=false)
{
    Mat result;
    // Estabilizar el número de bits.
    int histSize = maxVal-minVal+1;
    float range[] = { static_cast<float>(minVal),
static_cast<float>(maxVal+1) };
    const float* histRange = { range };
    // calcular histograma
    calcHist(&src, 1, 0, Mat(), result, 1, &histSize,
&histRange, true, false);
    // normalizar
    if(normed) {
        result /= (int)src.total();
    }
    return result.reshape(1,1);
}
```

Figura 76: Código del cálculo del histograma.

Fuente: Elaboración propia