# UNIVERSIDAD POLITÉCNICA SALESIANA
# SEDE QUITO

**CARRERA:**
**INGENIERÍA DE SISTEMAS**

**Trabajo de titulación previo a la obtención del título de:**
**Ingenieros de Sistemas**

**TEMA:**
**APLICACIÓN PARA EL PROCESAMIENTO DIGITAL DE IMÁGENES DE ROSTROS.**

**AUTORES:**
**ROGER AUGUSTO AUQUI MORENO**
**CRISTIAN RICARDO PIGUAVE OCHOA**

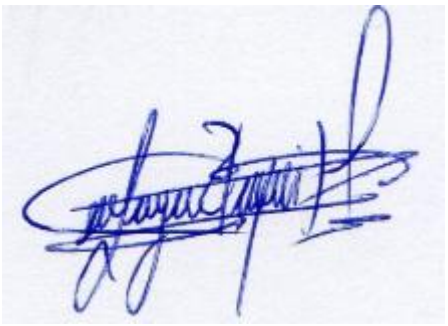
**TUTOR:**
**JULIO RICARDO PROAÑO ORELLANA**

**Quito, agosto del 2019**

## CESIÓN DE DERECHOS DE AUTOR

Nosotros, Roger Augusto Auqui Moreno con documento de identificación N.º 1724559420 y Cristian Ricardo Piguave Ochoa con documento de identificación N.º 1725214892, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación con el tema: APLICACIÓN PARA EL PROCESAMIENTO DIGITAL DE IMÁGENES DE ROSTROS, mismo que ha sido desarrollado para optar por el título de INGENIEROS DE SISTEMAS en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.



……..
ROGER AUGUSTO
AUQUI MORENO

CI: 1724559420

……...
CRISTIAN RICARDO
PIGUAVE OCHOA

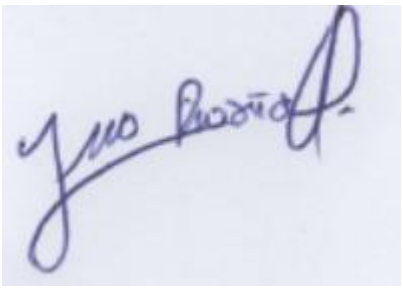
CI: 1725214892

Quito, agosto del 2019

## DECLARATORIA DE COAUTORIA DEL TUTOR

Yo, declaro que bajo mi dirección y asesoría fue desarrollado el Artículo Académico, con el tema: APLICACIÓN PARA EL PROCESAMIENTO DIGITAL DE IMÁGENES DE ROSTROS realizado por Roger Augusto Auqui Moreno y Cristian Ricardo Piguave Ochoa, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana para ser considerados como trabajo final de titulación.

Quito, agosto del 2019.



……..
JULIO RICARDO PROAÑO ORELLANA
CI: 0103909412

## DEDICATORIA

Dedico este trabajo en primer lugar a Dios por haberme dado la vida y las fuerzas durante todo el periodo universitario para poder continuar en los momentos más difíciles y guiarme de la mejor manera en una etapa más de mis estudios.

A mis amados padres Adriana y Augusto, por guiarme por el camino del bien, que con gran esfuerzo me entregaron este hermoso regalo que es el estudio, con su amor, con su dedicación y su sabiduría, son y serán siempre el ejemplo de lucha y perseverancia a seguir con todo lo que me proponga.

Y a mi familia y amigos que siempre estuvieron pendientes de todo este proceso y siempre me motivaban y me brindaban una palabra apoyo cuando más lo necesitaba.

Roger Augusto Auqui Moreno

## DEDICATORIA

En primer lugar, agradezco a mi madre María Ochoa por tener el temple para poder guiarme por el camino correcto, darme el estudio y animarme en esas noches enteras de desvelo por un proyecto o un examen con un café caliente de aliento en las madrugadas.

En segundo lugar, a mi padre Eugenio Piguave quien me enseño que con paciencia y dedicación se puede lograr cada una de las metas que me proponga.

En tercer lugar, a mis hermanos Paul Piguave y Saelyn Piguave quienes forman parte de mi día a día, para que sepan que obtener un título universitario no es fácil, pero se puede.

Cristian Ricardo Piguave Ochoa

## AGRADECIMIENTOS

Agradecemos a la Universidad Politécnica Salesiana que ha contribuido en nuestra formación profesional y personal, a nuestro tutor de proyecto de titulación Ingeniero Julio Proaño por habernos orientado y motivado para poder realizar satisfactoriamente nuestro trabajo.

Roger Augusto Auqui Moreno

Cristian Ricardo Piguave Ochoa

# APLICACIÓN PARA EL PROCESAMIENTO DIGITAL DE IMÁGENES DE ROSTROS.

Roger Auqui Moreno[1], Cristian Piguave Ochoa[2],Julio Proaño Orellana[3]

## Resumen

El presente trabajo consiste en el desarrollo de una aplicación para el procesamiento digital de imágenes de rostros. Por otro lado, debido a la complejidad de características que tiene una imagen digital de un rostro, se propone una arquitectura para la automatización del proceso de identificación. El proceso se divide en 3 etapas, en la primera se obtiene un video de 15 segundos de duración, en donde la aplicación obtiene una secuencia de fotos y las procesa filtrando las imágenes que contengan un rostro claro y que se puedan identificar claramente los descriptores faciales como son ojos, nariz y boca. Todas estas imágenes se almacenan con una estructura de carpetas nombre de la persona y todas las fotos se evalúa con respecto a los 68 puntos de referencia de un rostro. En la segunda etapa se procede identificar las características principales de la imagen utilizando algoritmos de clasificación en cascada y el algoritmo knn. Finalmente, en la tercera etapa permite detectar y reconocer el rostro. Los resultados nos muestran una precisión del 86.6%.

Palabras Clave: Detección de rostros, entrenamiento, reconocimiento facial, visión artificial.

## Abstract

The present work is about the development of an application for the digital processing of images of faces. On the other hand, due to the complexity of the characteristics of a digital image of a face, an architecture is proposed for the automation of the identification process. The process is divided in 3 stages, in the first one a video of 15 seconds of duration is obtained, where the application obtains a sequence of photos and processes them filtering the images that contain a clear face and that can clearly identify the facial descriptors such as eyes, nose and mouth. All these images are saved with a folder structure name of the person and all photos are evaluated against the 68 reference points of a face. In the second stage, the main features of the image are identified using cascade classification algorithms and the knn algorithm. Finally, in the third stage it is possible to detect and recognize the face. The results show an accuracy of 86.6%.

Keywords: Facial recognition, training, face detection, artificial vision.

[1]Ingeniero de Sistemas, Estudiante de Ingeniería de Sistemas – Universidad Politécnica Salesiana – sede Quito. Autor para correspondencia: rauqui@est.ups.edu.ec

[2] Ingeniero de Sistemas, Estudiante de Ingeniería de Sistemas – Universidad Politécnica Salesiana – sede Quito. Autor para correspondencia: cpiguave@est.ups.edu.ec

[3]Máster Universitario en Tecnologías Informáticas Avanzadas, Ingeniero Eléctrico, Docente – Universidad Politécnica Salesiana – sede Quito. Docente para correspondencia: jproanoo@ups.edu.ec

## 1. Introducción

En la actualidad el estudio de la visión artificial está marcando un paradigma en el procesamiento de imágenes digitales, con la finalidad de extraer información del mundo real partiendo de imágenes o videos por medio del computador. El procesamiento digital de imágenes de rostros en la actualidad es una de las funcionalidades con más enfoque centrándose en el reconocimiento facial para campos como la biometría, seguridad y reconocimiento de patrones. Una de las razones más importantes que ha llevado a este crecimiento, son la necesidad cada vez mayor de aplicaciones de seguridad y vigilancia utilizadas en diferentes ámbitos [1].

El principal inconveniente al implementar procesos de reconocimiento de uno o varios rostros reside en la gran variación entre cada una de las imágenes que puede llegar a tener una misma persona, estas discrepancias son producto, de diferentes tipos de luz artificial, luz natural o también conocida como luz fluorescente dependiendo de la variación del ambiente en el que fue captada, también varía por diferentes peinados, maquillaje e inclusive uso de accesorios, por lo que puede llegar a causar que la misma persona pueda parecer diferente entre varias imágenes [2]. Para los humanos esta tarea es natural y es realizada constantemente sin preocuparse por el cómo, el trasladar esta tarea a una maquina conlleva a uno de los principales objetivos propuestos en este estudio [2].

Actualmente existen varios sistemas de aprendizaje automático y visión artificial o visión por computadora, orientados al reconocimiento facial, los cuales utilizan técnicas y algoritmos especializados para detectar la ubicación de los principales hitos faciales ojos, nariz y boca [3] [4].

En este artículo se propone el desarrollo de una aplicación para el procesamiento de imágenes digitales con la finalidad de generar modelos entrenados la cual permite detectar e identificar rostros. Para ellos se propone una metodología compuesta por 3 etapas la primera se refiere a la extracción de imágenes de rostros mediante un video, la segunda consiste en el entrenamiento de modelos y finalmente el reconocimiento facial.

## 2. Trabajos relacionados.

Hoy en día existen varias técnicas para el reconocimiento facial. Algunas de ellas se enfocan en la detección de rostros y en la búsqueda de patrones de los principales hitos faciales [5] [6]. Con estas obras [7] [8] se centraron en los aspectos teóricos de la construcción de un sistema estable para el reconocimiento facial.

En [9] se desarrolló asignar instrucciones de computador a través del seguimiento de la mirada, sin la necesidad de utilizar el ratón y teclado, usando como sistema de entrada la cámara web del computador. La extracción de la región de cara y ojos es usando como base para la técnica Harr cascade de la librería *OpenCV* [10], simulando el movimiento del ratón con la mirada y con un parpadeo la acción de clic. Realizando estas acciones en un teclado virtual mostrado en el monitor, obteniendo más de un 95% en la precisión de rastreo de la mirada.

En [11], se enfocaron en el reconocimiento facial mediante la alineación del rostro y la extracción de características faciales por medio de la técnica *Face landmark* de la librería *dlib* [12] usando una *Convolutional Neural Networks* (CNN). Se utiliza CNN que permita ubicar y detectar las características faciales como son ojos, nariz y boca, procediendo con alinear y recortar el rostro de una imagen digital.

En [13] se enfocaron en comparar el rendimiento de dos librerías *OpenCV* y

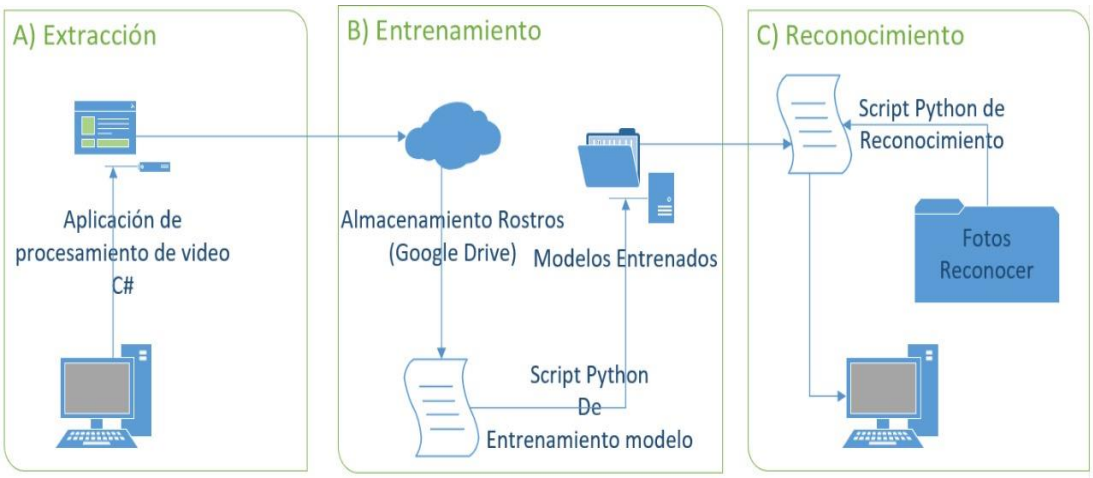



**Figura 1.** Arquitectura

*dlib* de visión artificial construyendo sobre sus bases dos sistemas de reconocimiento facial simples, basados en las técnicas de cascada la búsqueda de rostros y posterior reconocimiento realizadas en *Python*. Obteniendo como resultado que tiene un mejor rendimiento la librería OpenCV en la detección de rostros.

La mayoría de los trabajos relacionados con el reconocimiento facial requiere de la construcción de un dataset con imágenes de rostros y de una etiqueta (nombre de la persona).

En este trabajo se propone una arquitectura para la automatización del proceso de identificación facial con la extracción de imágenes mediante un video con la finalidad de obtener una secuencia de imágenes de una misma persona con diferentes gestos.

## 3. Metodología

La metodología utilizada en esta investigación consta de tres fases:

a) Extracción de imágenes de rostros mediante un video.

b) Entrenamiento de modelos.

c) Reconocimiento facial como se lo observa en la Figura 1.

### I. Extracción de imágenes de rostros

Iniciando con la extracción de imágenes, se optó grabar videos de 15 segundos de rostros, en formato .mp4 con una distancia focal de 25 a 55 milímetros y un ángulo de visión de 60 a 25 grados [14], se indica los movimientos específicos del rostro como se muestra en la Tabla 1.

Tabla 1 Movimientos de la cabeza.

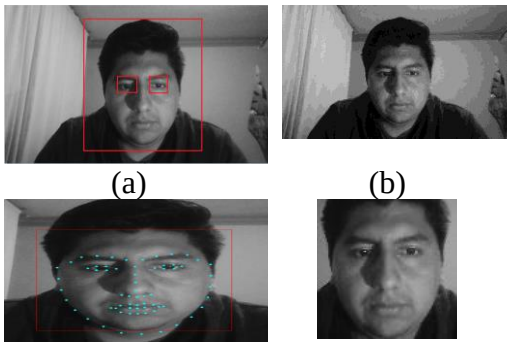
| Ítem | Detalle |
|---|---|
| 1 | Vista de frente a la cámara |
| 2 | Movimiento de cabeza de izquierda a derecha. |
| 3 | Movimiento de cabeza de derecha a izquierda. |
| 4 | Movimiento de cabeza de arriba hacia abajo |
| 5 | Movimiento de cabeza de abajo hacia arriba. |
| 6 | Vista de frente a la cámara aplicando diferentes expresiones faciales como son alegre, triste, enojado, serio entre otros. |

A continuación, se procede a extraer las imágenes con los rostros provenientes de un video, se desarrolló una aplicación de escritorio en *C#* permitiendo seleccionar un video en formato .mp4 del cual se extrae la secuencia de fotos que contiene el mismo como se muestra en la **Figura *2***.

**Figura 2.** Secuencia de fotos extraídas de video.

Posteriormente se valida que cada una de las imágenes contenga un rostro, aplicando el algoritmo para la detección de objetos desarrollados por la librería *EmguCV* llamado Harr-cascade, utilizamos un modelo pre-entrenado de OpenCV denominado haarcascade_frontalface el cual es una técnica para la detección de objetos, este método es utilizado para detectar varios aspectos como el/los rostros, para detectar los ojos en conjunto y los ojos por separado (izquierdo y derecho) de cada una de las fotografías utilizamos [15].

Las técnicas haarcascade righteye 2splits y haarcascade lefteye 2splits como se observa en la Figura 3(a), estas validación de detección se implementaron para validar que el rostro de la imagen sea aceptable como lo muestra en la Figura 3(b).



(a) (b)

(c) (d)

**Figura 3.** Rostros obtenidos

Adicionalmente, se implementó un método para complementar la validación, se utiliza *FacemarkLBF* es una librería que contiene *EmguCV* el cual permite detectar 68 puntos en el rostro encontrando los hitos faciales y con ello determinar si el rostro contiene 2 ojos, boca, nariz y 2 cejas como lo muestra en Figura 3(c).

Si el rostro de la imagen detectada cumple con el proceso de validación, se obtiene la ubicación del rostro de la imagen original generando un objeto de coordenadas con los puntos x (píxel inicial en x), y (píxel inicial en y), width (ancho de rostro detectado) y heigth (alto de rostro detectado).
Luego extraemos de la fotografía original solo el rostro utilizando los valores anteriormente mencionados y generamos la imagen como lo muestra la Figura 3(d) en una nueva dimensión de 100x100 pixeles.

El proceso de validación del rostro detectado se la realiza con la técnica Harr-cascade implementado un método en C# llamado "validarRostro()". Este método es el encargado de realizar la validación mencionada anteriormente.

El registro de la imagen del rostro válido, utiliza el método implementado en C# denominado "registrarRostro()", Este método se encarga de registrar la imagen en formato .jpg en una carpeta compartida la cual está en constante sincronización con la nube (Google Drive) obteniendo como finalidad los siguientes pasos representados en pseudocódigo como se observa en la Figura 4.

```
Algoritmo 1: Extracción de imágenes.
Variables
      entero   contador
      boolean   rostros,68puntos

1:     Inicio (Extracción de imágenes)
2:     Seleccionar video
3:     Crear un directorio
4:     Ingresar etiqueta (nombre de la persona) al
       directorio.
5:     Extraer imágenes
6:     contador=Cantidad de imágenes de fotograma del
       video.
7:     for i =1 in i >= contador
8:        rostros=localizar rostros usando
          facesDetected.
9:        68puntos=localizar 68 puntos del rostro
          usando FacemarkLBF.
10:    if (rostros = true && 68puntos = true)
11:          Desplegar "Rostro encontrado"
12:          Guardar imagen
13:       else if
14:          Desplegar "Rostro no encontrado"
15:          Descartar imagen
16:       end if
17:    end for
18:    Fin (Extracción de imágenes)
```

**Figura 4.** Pseudocódigo extracción de imágenes.

## II. Entrenamiento de modelos.

Para el entrenamiento de modelos se plantea utilizar una estructura de carpetas la cual tiene una carpeta global, una subcarpeta con una etiqueta (nombre de la persona) y en dentro de ella la secuencia de imágenes obtenidas del video de la persona como se lo observa en la Figura 5.

Luego se procede a ejecutar el script desarrollado en Python llamado "TrainFotosModelo.py" el cual nos permite generar un modelo de entrenamiento con los rostros registrados en la carpeta compartida, el entrenamiento del modelo se la realiza por el método "train()" este método se encarga de generar el modelo clasificado aplicando el algoritmo KNN [16].

Clasificando el nombre de las personas con sus imágenes, con la finalidad de obtener un modelo supervisado "TrainedFacesModelo.clf" el cual posteriormente se almacena en una carpeta en la nube (Google Drive).

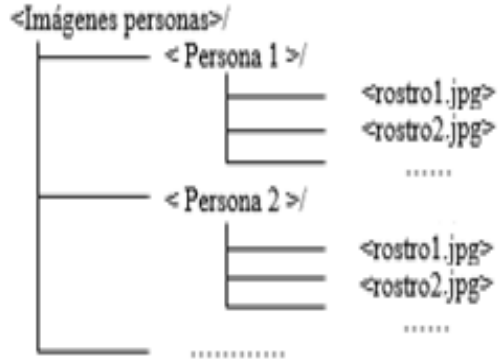



**Figura 5.** Estructura de archivos con imágenes a entrenar.

A continuación, se detalla el modelo entrenado en pseudocódigo como se lo observa en la Figura 5.

```
Algoritmo 2: Entrenar modelo
Variables
entero  i, j, ixpersona, personas, urostros, vecinos
urostros=cantidad de rostros por imagen.
ixpersona=cantidad de imágenes por persona.
personas=cantidad de personas en carpeta global.

1:    Inicio (Entrenar modelo)
2:    Seleccionar directorio de carpetas global
3:    Leer personas
4:    for j=1 in j >= personas
5:        leer ixpersona
6:        ingresar cantidad de vecinos más
          cercanos
7:        leer vecinos = 2
8:        for i =1 in i >= ixpersona
9:            if urostros!=1
10:               Desplegar "Incorrecto"
11:               Remover imagen a carpeta
                  no_validos
12:           else if
13:               Desplegar "Correcto"
14:               Crear clasificación con
                  estructura de carpetas en
                  árbol con KNN.
15:               Ubicar nombre persona e
                  imagen correcta.
16:           end if
18:       end for
19:   end for
20:   Guardar modelo entrenado
21:   Fin (Entrenar modelo)
```

**Figura 6.** Pseudocódigo entrenar modelo

## III. Reconocimiento facial.

Cuando se obtiene el modelo clasificado con el entrenamiento de los rostros se ejecuta el script desarrollado en Python denominado "reconoce.py", el cual nos permite procesar una imagen e identificar el rostro de o las personas enmarcándolas en un cuadro azul y el nombre de la persona en dentro del mismo, si la persona no estuviera registrada en el modelo entrenado me detectara el rostro y enmarcara en un cuadro azul y desplegando la palabra "Desconocido" en dentro del mismo.

El proceso de identificación de los rostros se la realiza por el método "predict()", este método selecciona la ruta en donde se encuentra el modelo entrenado, se localiza el rostro en la imagen que va a ser procesada para el reconocimiento y con ello se codifica la imagen para proceder a calcular la distancia entre el ítem a clasificar y el

resto de ítems del dataset entrenado, si el resultado es exitoso se obtiene una etiqueta con el nombre de la carpeta en donde la coincidencia con la imagen de entrenamiento se dio y modifica la imagen enmarcando el rostro y la etiqueta con el nombre de la persona como se muestra en la Figura 7.

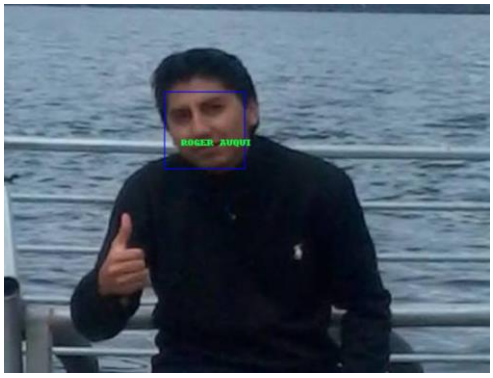



**Figura 7.** Resultado exitoso

Caso contrario si no se encontró similitud con las imágenes registradas en el modelo se modifica la imagen enmarcando el rostro que se procesó y etiquetándolo con "Desconocido" como se muestra en la Figura 8.


**Figura 8.** Sin resultado de reconocimiento.

A continuación de detalla el reconocimiento facial en pseudocódigo como se observa en la Figura 9.

```
Algoritmo 3: Reconocimiento facial.
Variables
        entero urostros, vecinos
        urostros=cantidad de rostros en  una
        imagen.
1:      Inicio (Reconocimiento facial)
2:        Cargar modelo entrenado
3:        Seleccionar imagen
4:        Ubicar rostro en imagen
5:        if (urostros == 0)
6:          Desplegar "Rostros no detectados".
7:        else if
8:          Desplegar "Rostros detectados"
9:          Buscar rostros similares con el modelo
            entrenado.
10:         Utilizar KNN para encontrar el vecino
            más cercano.
11:         Calcular distancia euclidiana más
            cercana.
12:         if (rostro == true)
13:           Obtener etiqueta
14:           Dibujar recuadro alrededor del rostro
              con etiqueta "Nombre persona".
15:         else if
16:           Dibujar recuadro alrededor del rostro
              con etiqueta "Desconocido".
17:         end if
18:       end if
19:     Fin (Reconocimiento facial)
```

**Figura 9.** Pseudocódigo reconocimiento facial.

## 4. Evaluación y resultados.

Para validar que el sistema cumpla con un nivel de precisión aceptable se realizó algunas pruebas experimentales utilizando un dataset obtenido en la extracción de imágenes de videos solicitados a 25 personas en ambientes no controlados, el cual se detallara a continuación.

Tabla 2 Dataset utilizado

| Ítems | Obtenidos | Correctos | Incorrectos |
|---|---|---|---|
| Cantidad de videos | 25 | 10 | 15 |
| Cantidad de imágenes | 2100 | 1810 | 290 |

### I.Tiempo de entrenamiento

Con la finalidad de evaluar el tiempo de entrenamiento del modelo se utilizaron 2 computadoras con similares características como se muestra en la Tabla 3 y tomar el tiempo de

entrenamiento que se obtuvieron de cada una de ellas.

Tabla 3.Características generales de computadoras utilizadas.

| Maquina 1 | Maquina 2 |
|---|---|
| Procesador: Intel Core i7-7700HQ CPU 2,80GHz<br>RAM: 16,0 GB<br>Tipo Sistema: 64 bits<br>GPU: NVIDIA GeForce GTX 1060 with Max-Q Design 6GB | Procesador: Intel Core i7-6500U CPU 2,50GHz<br>RAM: 16,0 GB<br>Tipo Sistemas: 64 bits<br>GPU: AMD Radeon R5 M335 4GB |

Para realizar la prueba se tomó como base a 7 personas para el entrenamiento con 100, 200, 250 imágenes por cada una de ellas y se tomó el tiempo de entrenamiento del modelo por 3 ocasiones obteniendo los resultados en la Tabla 4.

Tabla 4.Resultado de entrenamiento.

| # | Prueba1 (700 imágenes) | Prueba 2 (1400 imágenes) | Prueba3 (1750 imágenes) |
|---|---|---|---|
| Pc 1 | 9 min 08seg | 19min 21seg | 23min 09seg |
| Pc 2 | 9 min 48seg | 20min 09seg | 23min 54seg |

Tras la realización de las pruebas de entrenamiento del modelo generado desde cada computador utilizado para la prueba, fue posible calcular el tiempo de diferencia que se obtuvo en cada entrenamiento en el cual fue un promedio de 44.33 segundos de diferencia, tomando en cuenta estos resultados puede aumentar o disminuir según las características del computador.

## II. Modificación en imágenes.

Con la finalidad de validar cual imagen es más adecuada para la generación del modelo de entrenamiento para una mejor identificación del rostro y modificar con ello el sistema.

Para ello vamos a generar los modelos de entrenamiento con diferentes fuentes de imágenes las cuales cada grupo de imágenes se va a usar en RGB (imagen a color) y Gray (imagen a escala de grises), cada fuente de imágenes está compuesta por 5 personas y cada una de ella contiene 100 fotografías.

Después de generar cada modelo entrenado aplicamos el proceso de reconocimiento para cada modelo y con ello validar el porcentaje de precisión y error de cada uno. Para ello se generó un conjunto de 25 imágenes a reconocer el cual consiste en 5 imágenes por personas entrenada en el modelo a ser evaluado.

Tras la realización de las pruebas de reconocimiento aplicados a cada modelo obtenemos los siguientes resultados como se observa en la Figura 10.

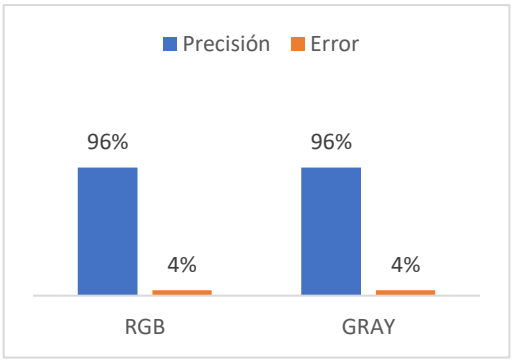



**Figura 10.** Tipos de imágenes.

De esta forma fue posible determinar que el error es de un 4% en todos los modelos entrenados, por tanto, los resultados fueron muy similares, con ello podemos definir que la librería dlib implementada se centra en la obtención del rostro indistintamente del tipo de imagen.

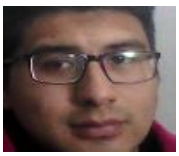
## III. Modificación de dimensión de imágenes.

Con la finalidad de validar cual es la dimensión más apropiado para la generación del modelo de entrenamiento para una mejor identificación del rostro y modificar con ello el sistema y almacenar las imágenes apropiadas.

Para ello se generó los modelos de entrenamiento con diferentes fuentes de imágenes las cuales cada grupo de imágenes van a tener diferentes dimensiones, las dimensiones que se va a manejar son:

Dimensión original (como se extrae del video), se observa en la Figura 11 (a).Dimensión recortada hasta hombros (300x300px) como se observa en la Figura 11 (b), detectamos la ubicación del rostro de la imagen original y la recortamos tratando de obtener hasta los hombros de la persona. Dimensión solo rostro detectamos la ubicación del rostro y cortamos alrededor del mismo obteniendo una imagen de dimensiones (100x100px) como se observa en la Figura 11 (c).

(a)



(b) (c)

**Figura 11.** Escala de imágenes

Cada fuente de imágenes está compuesta por 5 personas y cada una de ella contiene 100 fotografías.

Después de generar cada modelo entrenado con las imágenes con diferentes dimensiones antes mencionadas, aplicamos el proceso de reconocimiento y con ello validar el porcentaje de precisión y error de cada uno.

Para ello se generó un conjunto de 25 imágenes a reconocer el cual consiste en 5 imágenes por personas entrenada en el modelo a ser evaluado.

Tras la realización de las pruebas de reconocimiento aplicados a cada modelo se obtuvieron los resultados mostrados en la Figura 12.

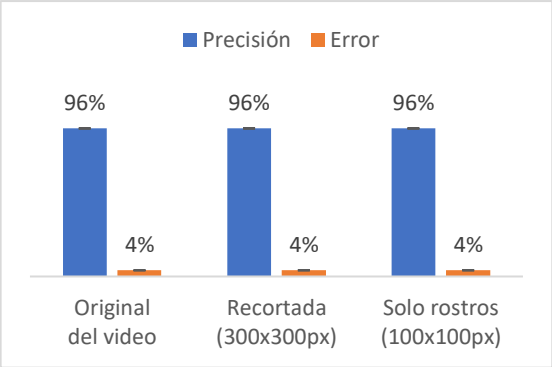



**Figura 12.** Resultados de precisión y error con distintas dimensiones de imágenes

De esta forma fue posible determinar que el error obtenido es de un 4% en todos los modelos entrenados, con ello podemos definir que la librería dlib implementada se centra en la obtención del rostro indistintamente de la dimensión de la imagen a registrar en el modelo entrenado y su posterior reconocimiento.

## IV. Variación de umbral (distancia euclidiana)

Con la finalidad de validar cual es el umbral más apropiado para la generación del modelo de entrenamiento mejorando la identificación del rostro.

Para ello se genera los modelos de entrenamiento con una fuente de imágenes compuesto de 5 personas y con 100 fotografías por cada una de ellas. Y los valores de umbral que serán utilizados son: 0.4, 0.5 y 0.6. Después de generar cada modelo entrenado aplicamos el proceso de reconocimiento y con ello validar el porcentaje de precisión y error de cada uno. Para ello se generó un conjunto de 30 imágenes a reconocer el cual consiste en 5 imágenes por personas entrenada en el modelo a ser evaluado, adicional en el conjunto de datos se

añadirán las imágenes de 5 personas desconocidas.

Tras la realización de las pruebas de reconocimiento aplicados a cada modelo, obtenemos las siguientes matrices de confusión como se observa en la Tabla 5 utilizando un umbral de 0.4, en la Tabla *6* con un umbral de 0.5 y en la Tabla *7* con un umbral de 0.6, indicando las siguientes nomenclaturas.

P = persona.
Des = desconocido.
Tot = total.

Tabla 5. Matriz de confusión de un modelo entrenado con umbral de 0.4.

| P | 1 | 2 | 3 | 4 | 5 | Des | Tot |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 4 | 5 |
| 2 | 0 | 1 | 0 | 0 | 0 | 4 | 5 |
| 3 | 0 | 0 | 3 | 0 | 0 | 2 | 5 |
| 4 | 0 | 0 | 0 | 5 | 0 | 0 | 5 |
| 5 | 0 | 0 | 0 | 0 | 5 | 0 | 5 |
| Des | 0 | 0 | 0 | 0 | 0 | 5 | 5 |
| Tot | 1 | 1 | 3 | 5 | 5 | 15 | 30 |

Tabla 6.Matriz de confusión de un modelo entrenado con umbral de 0.5.

| P | 1 | 2 | 3 | 4 | 5 | Des | Tot |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 4 | 5 |
| 2 | 0 | 4 | 0 | 0 | 0 | 1 | 5 |
| 3 | 0 | 0 | 5 | 0 | 0 | 0 | 5 |
| 4 | 0 | 0 | 0 | 5 | 0 | 0 | 5 |
| 5 | 0 | 0 | 0 | 0 | 5 | 0 | 5 |
| Des | 0 | 0 | 0 | 0 | 1 | 4 | 5 |
| Tot | 1 | 4 | 5 | 5 | 6 | 15 | 30 |

Tabla 7.Matriz de confusión de un modelo entrenado con umbral de 0.6.

| P | 1 | 2 | 3 | 4 | 5 | Des | Tot |
|---|---|---|---|---|---|---|---|
| 1 | 5 | 0 | 0 | 0 | 0 | 0 | 5 |
| 2 | 0 | 4 | 1 | 0 | 0 | 0 | 5 |
| 3 | 0 | 0 | 5 | 0 | 0 | 0 | 5 |
| 4 | 0 | 0 | 0 | 5 | 0 | 0 | 5 |
| 5 | 0 | 0 | 0 | 0 | 5 | 0 | 5 |
| Des | 0 | 0 | 1 | 1 | 1 | 2 | 5 |
| Tot | 5 | 4 | 7 | 6 | 6 | 2 | 30 |

A partir de los resultados obtenidos se puede realizar un análisis y calcular los valores porcentuales de precisión y error de clasificación (MisClasification Error) [17] como se observa en la Tabla 8, con los resultados obtenidos podemos definir que para un óptimo reconocimiento el sistema podría utilizar el valor de umbral de reconocimiento de 0.5 o 0.6.

Tomando en cuenta que los valores de error entre los umbrales 0.5 y 0.6, son muy similares pero los errores de las pruebas son distintos, esto quiere decir, cuando el sistema está configurado para generar el modelo con el umbral de 0.5, el reconocimiento de una persona que no fue entrenada con una característica en el rostro como por ejemplo los lentes, la evaluación y el reconocimiento acertara y retorna Desconocido, pero a su vez si el sistema está configurado con un umbral de 0.6 estas características si se toman en cuenta y retornara la identificación correcta.

A su vez cuando el sistema este configurado con un valor de umbral de reconocimiento de 0.6, las imágenes procesadas para el reconocimiento con personas que no se encuentren en el modelo entrenado retornara la similitud más cercana fallando en el reconocimiento.

Tabla 8.Resultados de error de clasificación de los modelos por umbral.

| Umbral | Error de clasificación |
|---|---|
| 0,4 | 33.33% |
| 0,5 | 20% |
| 0,6 | 13.33% |

## V. Evaluación con varios dataset.

Con este experimento se busca validar y encontrar la precisión que tiene nuestra arquitectura propuesta para la automatización de reconocimiento facial.

En estos dataset se definieron varias características para la obtención de imágenes como es: número total de personas, número de imágenes por persona, número total de imágenes, género, raza, rango de edad, barba, formato de imagen, cámara utilizada, Iluminación: artificial, natural y fluorescente.

Obteniendo un dataset con escenarios definidos y una cierta cantidad de imágenes como se observa en la Tabla 9.

Tabla 9 Datasets utilizados

| Dataset | Personas | Imágenes por persona | Total, de imágenes |
|---|---|---|---|
| AT&T | 72 | 20 | 1440 |
| Yales faces | 15 | 10 | 150 |
| CSW | 10 | 10 | 100 |
| Grimace | 18 | 20 | 360 |
| Prima Inrialpes | 155 | 80 | 12400 |

Se procede a descartar imágenes en las que no se encontraron un rostro válido como se observa en la Tabla 10.

Tabla 10 Imágenes descartadas

| Dataset | Imágenes correctas | Imágenes descartadas |
|---|---|---|
| AT&T | 1415 | 25 |
| Yales faces | 148 | 2 |
| CSW | 98 | 2 |
| Grimace | 159 | 10 |
| Prima Inrialpes | 10439 | 1931 |

Se encontró que al descartar imágenes de estos dataset es debido a la iluminación y posición del rostro, evitando encontrar los principales hitos faciales como se observa en la Figura 13 del dataset de prima inrialpes y del dataset de yales faces descarta la imagen por la iluminación perdiendo la ubicación de una de las cejas como se observa en la Figura 14.

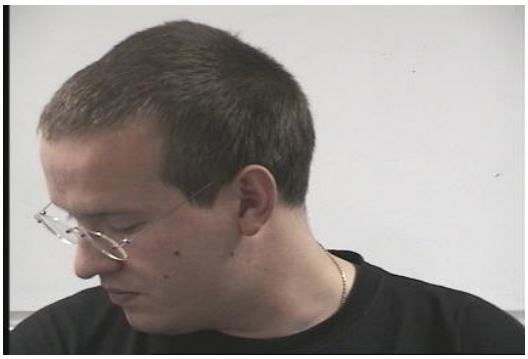


**Figura 13.** Imagen descartada en dataset prima inrialpes[18]

**Figura 14.** Imagen descartada del dataset yales faces [19]

Evaluando cada uno de los dataset antes mencionados se obtuvieron un valor de exactitud como se muestra la Tabla 11 encontrando, que nuestra arquitectura para la automatización de reconocimiento facial propuesta tiene un promedio de exactitud del 86.38%.

Tabla 11 Valor de exactitud por dataset

| Dataset | Exactitud |
|---|---|
| AT&T | 84,50% |
| Yales faces | 86,30% |
| CSW | 87,90% |
| Grimace | 88,50% |
| Prima Inrialpes | 84,50% |
| Arquitectura propuesta | 86,60% |

## 5. Conclusiones

La estructura que se utilizó para el sistema, se desarrolló para obtener el mejor potencial utilizando las librerías de OpenCV con su extensión para C# llamada EmguCV y la librería dlib, considerando que las librerías mencionadas son robustas por separado, aprovechando lo mencionado, se utilizó la

librería EmguCV para el procesamiento y detección de rostros para generar las imágenes necesarias para el entrenamiento del modelo y la librería dlib para generar el modelo de reconocimiento y el proceso de reconocimiento.

La aplicación propuesta permite generar una gran base de datos para generar el modelo de entrenamiento, ya que al extraer los rostros desde un video nos permite generar una gran secuencia de fotogramas, las cuales nos ayuda para almacenar el rostro de una persona desde varias perspectivas .Dicho esto el modelo de entrenamiento sería posible identificar a una persona y disminuir el porcentaje de error del sistema que actualmente es de un 13.33%, resultado obtenido al realizar los experimentos antes mencionados.

El modelo de reconocimiento facial de la librería dlib indica que tiene una precisión 86.6% por medio de varios experimentos por ejemplo se cambió el umbral de 0.4 hasta 0.6 se pudo evidenciar que sin importar la dimensión de la imagen que se ingresó a entrenar el sistema es fiable y reconoce con un 86.6% a la persona si esta entrenada.

Al implementar en la estructura la ejecución del entrenamiento del modelo para el reconocimiento facial por medio de un script desarrollado en Python utilizando la librería dlib, nos orientamos a una futura estructura en la cual podamos ejecutar varios scripts de entrenamiento de modelos con diferentes fuentes de imágenes, los cuales podrían ser generados desde un web service y los mismos almacenarlos en un servidor. Y con ello acceder a los modelos dependiendo como puedan ser utilizados.

A su vez, en la estructura también se implementó el reconocimiento facial de la misma manera que se genera el modelo, esto quiere decir, desde un script de reconocimiento desarrollado en Python utilizando la librería dlib, con ello ejecutamos el script mencionado ya sea desde un aplicativo web, móvil o de escritorio, enviando como parámetros la información necesaria en el caso de la aplicación desarrollada seria la imagen la cual va a ser procesada para el reconocimiento facial.

## 6. Referencias


[1] R. Gimeno Hernández, J. R. Morros, and R. Barcelona, “ESTUDIO DE TÉCNICAS DE RECONOCIMIENTO FACIAL,” 2010.

[2] R. DE Rostros and N. Y. Algoritmos De Reconocimiento De Objetos De La Biblioteca Opencv Edison Rene Caballero Barriga, “APLICACIÓN PRÁCTICA DE LA VISIÓN ARTIFICIAL PARA EL.”

[3] “CAPÍTULO 1 INTRODUCCIÓN.”

[4] R. Trapiella Pino Autor Carlos Cerrada Somolinos Director, “Desarrollo y evaluación de técnicas de visión artificial apliocadas a un problema de reconocimiento facial.,” 2017.

[5] OpenCV, “OpenCV: Detección de rostro utilizando Cascadas Haar.” [Online]. Available: https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html. [Accessed: 25-Jun-2019].

[6] H. A. M. Alhamzawi, “Faces and eyes Detection in Digital Images Using Cascade Classifiers,” Comput. Eng. Appl. J., vol. 7, no. 1, pp. 57–66, 2018.

[7] “Face detection using OpenCV and Python: A beginner’s guide - Blogs

SuperDataScience - Big Data | Analytics Careers | Mentors | Success.” [Online]. Available: https://www.superdatascience.com/blogs/opencv-face-detection. [Accessed: 14-Jul-2019].
[8] “Detección de rostros con OpenCV y Python.” [Online]. Available: https://github.com/davisking/dlib/tree/master/python_examples. [Accessed: 14-Jul-2019].
[9] C. Z. Li, C. K. Kim, and J. S. Park, “The indirect keyboard control system by using the gaze tracing based on haar classifier in opencv,” Proc. - 2009 Int. Forum Inf. Technol. Appl. IFITA 2009, vol. 2, no. 1, pp. 362–366, 2009.
[10] “OpenCV.” [Online]. Available: https://opencv.org/. [Accessed: 25-Jun-2019].
[11] S. Sharma, K. Shanmugasundaram, and S. K. Ramasamy, “FAREC - CNN based efficient face recognition technique using Dlib,” Proc. 2016 Int. Conf. Adv. Commun. Control Comput. Technol. ICACCCT 2016, no. 978, pp. 192–195, 2017.
[12] “Biblioteca dlib C ++.” [Online]. Available: http://dlib.net/. [Accessed: 25-Jun-2019].
[13] N. Boyko, O. Basystiuk, and N. Shakhovska, “Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library,” Proc. 2018 IEEE 2nd Int. Conf. Data Stream Min. Process. DSMP 2018, pp. 478–482, 2018.
[14] “La Distancia Focal Explicada con Ejemplos.” [Online]. Available: https://www.dzoom.org.es/la-distancia-focal-todo-lo-que-necesitas-saber-explicado-con-ejemplos/. [Accessed: 14-Jul-2019].
[15] K. N. K. Kumar, H. Natraj, and T. P. Jacob, “Motion activated security camera using Raspberry Pi,” Proc. 2017 IEEE Int. Conf. Commun. Signal Process. ICCSP 2017, vol. 2018-Janua, pp. 1598–1601, 2018.
[16] S. Taneja, C. Gupta, S. Aggarwal, and V. Jindal, “MFZ-KNN-A modified fuzzy based K nearest neighbor algorithm,” Proc. - 2015 Int. Conf. Cogn. Comput. Inf. Process. CCIP 2015, pp. 1–5, 2015.
[17] “RPubs - Matriz de Confusión - Evaluación de modelos de predicción.” [Online]. Available: https://rpubs.com/chzelada/275494. [Accessed: 27-Jun-2019].
[18] “Head Pose Image Database.” [Online]. Available: http://www-prima.inrialpes.fr/perso/Gourier/Faces/HPDatabase.html. [Accessed: 21-Jul-2019].
[19] “Yale Face Database | vision.ucsd.edu.” [Online]. Available: http://vision.ucsd.edu/content/yale-face-database. [Accessed: 21-Jul-2019].