DESARROLLO DE SISTEMA PARA LA DETECCIÓN DE EMOCIONES UTILIZANDO VISIÓN ARTIFICIAL E INTELIGENCIA ARTIFICIAL APLICADO A ENTORNOS EDUCATIVOS.

Juan David Pinzón Naranjo

Universidad de Pamplona

Ingeniería en telecomunicaciones
departamento EEST facultad de ingenierías y arquitectura

Pamplona
2020

II

DESARROLLO DE SISTEMA PARA LA DETECCIÓN DE EMOCIONES UTILIZANDO VISIÓN ARTIFICIAL E INTELIGENCIA ARTIFICIAL APLICADO A ENTORNOS EDUCATIVOS.

Juan David Pinzón Naranjo

Trabajo de grado presentado como requisito para optar al título de Ingeniero de telecomunicaciones

Director:
Nydia Susana Sandoval
Universidad de Pamplona

Universidad de Pamplona
Ingeniería en telecomunicaciones
departamento EEST facultad de ingenierías y arquitectura
Pamplona
2020

"Gracias a mis padres, que siempre creyeron en mí.
a mi familia que siempre me apoyo en las buenas y en las malas.
También un agradecimiento a mis docentes de colegio y universidad por formarme como una persona íntegra honesta.

A mis amigos que siempre estuvieron en los proyectos más importantes en la formación como persona.
"

Resumen

Inicialmente este proyecto trata de centrarse en el reconocimiento anímico de personas por medio de sus expresiones faciales o características especiales en su rostro. La primera etapa del proyecto fue una de las más importantes debido a que el software es capaz de detectar la cara de cualquier persona a través de una imagen o cámara externa. Esta detección de rostro se llevó a cabo por medio de un algoritmo de reconocimiento de rostros. En la segunda etapa se identifica qué base datos se va a utilizar, en este caso se implementó una que maneja de 400 a 700 imágenes de muestra por cada emoción para el entrenamiento de algoritmos. En la tercera etapa se entrena el algoritmo para poder identificar las partes esenciales del rostro de la persona para así poder determinar el estado de ánimo que esta persona muestra, los cuales se pueden diferenciar entre seis estados iniciales de ánimo básicos. En La última etapa se implementa una interfaz fácil de manejar que contenga toda la información relevante de los procesos que se llevan a cabo en la toma de datos y el muestreo.

Palabras clave: Detección, facial, anímico, reconocimiento, algoritmo, imagen, emoción.

.

.

Abstract

Initially this project tries to focus on the emotional recognition of people through their facial expressions or special features on their face. The first stage of the project was one of the most important because the software is capable of detecting the face of any person through an external image or camera. This face detection was carried out by means of a face recognition algorithm. In the second stage, the database to be used is identified. If possible, one that handles 400 to 700 sample images for each emotion for algorithm training. In the third stage, the algorithm is trained to identify the essential parts of the person's face in order to determine the state of mind that this person shows, which can be differentiated between six basic initial states of mind. In the last stage, an easy-to-use interface is implemented that contains all the relevant information on the processes that are carried out in data collection and sampling.

Keywords: Detection, facial, psychic, recognition, algorithm, image, emotion.

Agradecimientos

Mis más sinceros agradecimientos:

A mi madre Emilse Naranjo y padre Ulpiano Pinzón por formar parte de mi formación profesional y apoyarme incondicionalmente, por darme siempre fuerzas para salir adelante como una persona íntegra.

A mi hermano Edwar Pinzón por estar siempre pendiente de mi en mis momentos más difíciles y apoyarme siempre.

A mis amigos, gracias por enseñarme a compartir más como persona y a dejar atrás los malos entendidos y aprender a disculpar.

A la profesora Nydia Susana por ser parte de este grandioso proyecto de grado.

A mis profesores por formar las bases en las diferentes materias para así afrontar este proyecto de grado.

A todas las personas que conocí días tras en la universidad de pamplona y de las cuales me llevo un gran recuerdo: Eiver, Adilson, Richard, Alejandro, los insociables.... Debo ser la única persona del mundo a la que se le acaba la hoja de dedicatorias, pero en fin son demasiadas personas, ojalá pudiera decir a todos algo más ya que los llevo en mi corazón.

Capítulo 1 Contenido

Índice de Figuras

Capítulo 1 Introducción

La revolución tecnológica actual y el interés multidisciplinar por desarrollar métodos y productos que mejoren los resultados obtenidos de la aplicación del reconocimiento de patrones y clasificación de rasgos faciales facilita hoy en día la protección de la información, pero también ayuda a la detección de emociones. Hoy en día hay muchos sistemas implementados a nivel mundial de detección de emociones. Los cuales se pueden encontrar en diferentes aplicaciones del celular o en una cámara de seguridad en la cual tienes que hacer algún gesto para permitir el acceso. La idea de este proyecto nace con hacer más fácil la detección de una emoción utilizando bases de datos creadas con rostros en diversas posiciones, para obtener resultados favorables y a partir de un algoritmo detectar el rostro de la persona para proceder a indicar ciertos parámetros y así proceder a utilizar modelos para predecir una emoción.

1.1 Planteamiento del Problema

La Universidad de Pamplona no cuenta con un sistema que permita identificar de manera adecuada los estados de ánimo de los estudiantes en tiempo real, debido a que hay poco interés en saber los estados de ánimo de los estudiantes, y a partir de ello que el docente tome otra metodología con el estudiante.

Uno de los principales problemas que hay hoy en día es la ausencia de softwares que puedan interactuar o captar las expresiones de los estudiantes. Estas falencias nacen con el poco interés de mejorar el proceso de relación entre el estudiante y el docente, ya que los jóvenes presentan en su rostro señales físicas que pueden ser captadas y adaptadas a procesos tecnológicos que puedan convertir una emoción del rostro en características fáciles de fácil manejo para un sistema automático que le indique qué proceso debería seguir. Hoy en día se cuenta con más estudiantes con problemas de aprendizaje en diferentes materias debido a sus estados de ánimo.

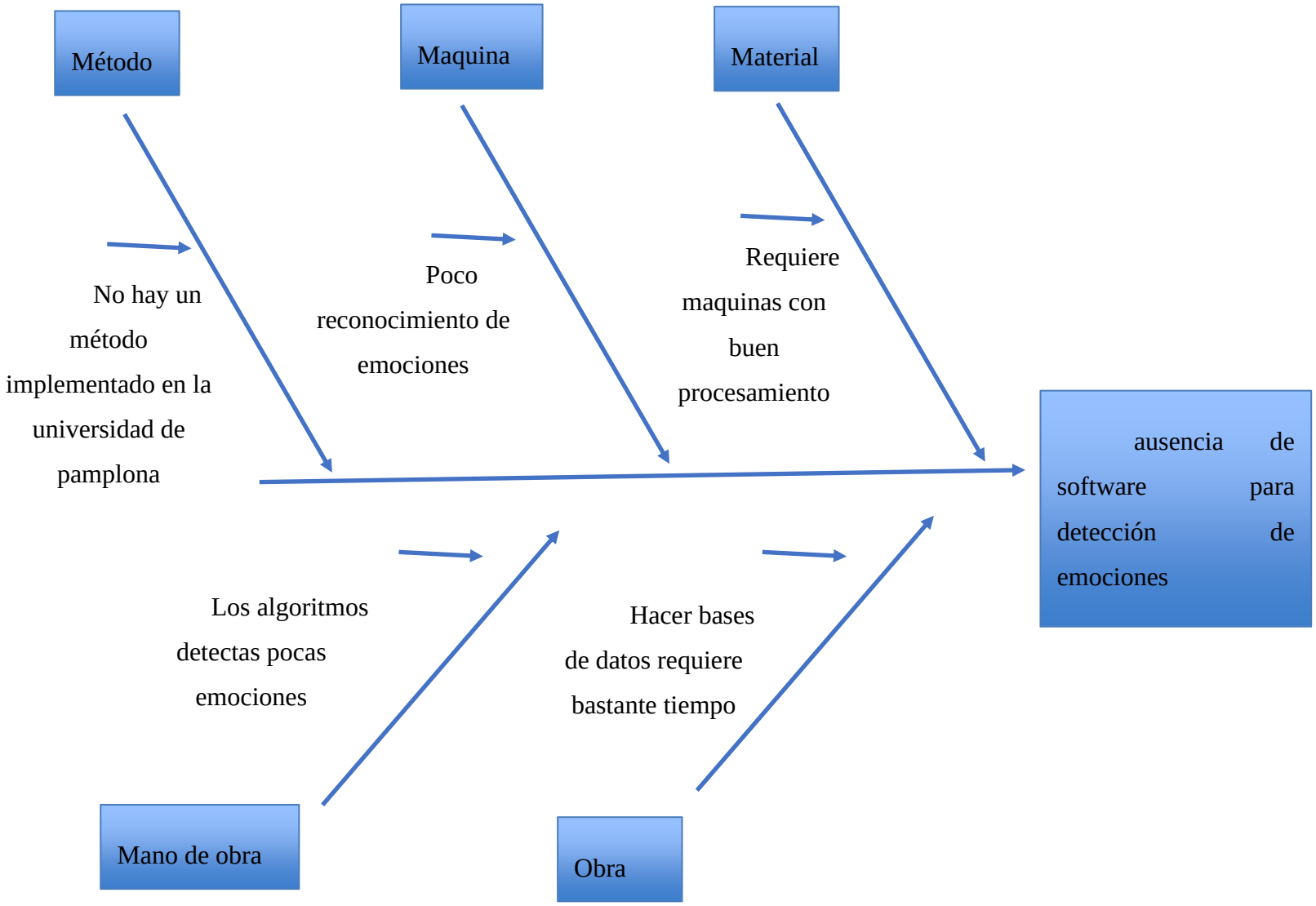


Figura 1.1 Diagrama causa y efecto

Fuente:(Autor)

1.2 Justificación y Objetivos

Lo que se busca con este trabajo es desarrollar un software capaz de identificar el estado de ánimo de una persona utilizando algoritmos de aprendizaje automático para la detección de rostros, los cuales relacionan características morfológicas que tienen todas las personas en su rostro y que no son iguales. Estos rasgos faciales permiten identificar diferentes descriptores a partir de modelos que relacionan diferentes puntos en toda la parte frontal del rostro.

El entrenamiento de algoritmos para el reconocimiento de patrones a través de modelos ya establecidos como lo es el modelo LBPH que genera resultados favorables en cuanto a al entrenamiento de emociones que en este caso se relacionan las emociones básicas principales.

Llevar a cabo el diseño de una interfaz relacionando los parámetros de entrada y salida generará una comunicación más sencilla entre usuario y sistema de detección de emociones.

1.2.1 Objetivo General

Implementar un sistema capaz de detectar el rostro de una persona e identificar los estados de ánimo básicos mediante diferentes algoritmos de visión e inteligencia artificial aplicado a entornos educativos.

1.2.2 Objetivos Específicos

- Establecer el reconocimiento facial en imágenes mediante el diseño de un algoritmo para la identificación únicamente de los rostros.
- Implementar base de datos para entrenamiento de algoritmos que relacionan cualidades morfológicas faciales.
- Detectar estados de ánimo relacionando diferencias o tipos de clasificadores que se puedan identificar en una imagen que contenga un rostro.
- Implementar una interfaz que muestre los estados de ánimo de los estudiantes y facilite la ejecución para los usuarios.
- Validar el funcionamiento de la interfaz y de los algoritmos de detección de estados de ánimo colocando en marcha este sistema en un entorno educativo con estudiantes y profesores.

Capítulo 2 Marco teórico

En el capítulo se presentarán parte de las herramientas, conceptos e historia de la detección facial.

2.1 Historia de la detección facial

La detección de rostros se ha desarrollado durante décadas. En la década de 1950, el primer experimento de detección de rostros se realizó desde una perspectiva psicológica. Desde entonces se han realizado otros estudios, como la interpretación de diversas expresiones faciales, la interpretación de emociones o la percepción de gestos. Durante las décadas de 1970 y 1980, se utilizaron plantillas y medidas de características geométricas de varias partes del rostro para detectar y reconocer rostros. Desde entonces, la investigación sobre la detección de rostros ha continuado: por ejemplo, en 1994, Yang y Huang propusieron un método de capas basado en el conocimiento que puede perfeccionar la detección de rostros en cada etapa. Li y sus colegas propusieron un método para detectar y rastrear rostros en videos en color al combinar la detección de piel en color con modelos faciales para detectar rostros en el área de la piel (García del Prado et al. 2020).

Woodrow Wilson Bledsoe puede considerarse un pionero de esta tecnología porque comenzó a estudiar un sistema para clasificar los rasgos faciales mediante la tabla RAND en 1960. El sistema utiliza un lápiz óptico y coordenadas para identificar los ojos, la nariz o la boca de una persona, pero sigue siendo un proceso muy manual. Diez años más tarde, llegaron Goldstein, Harmon y Lesk, quienes introdujeron estos rasgos faciales en detalle y comenzaron a desarrollarse en la dirección de la precisión del reconocimiento facial. Posteriormente, a finales de la década de 1980, gracias a Sirovich y Kirby, se aplicó el álgebra lineal (*Presente y Futuro Del Reconocimiento Facial*, n.d.).

Ya en 1991, Turk y Pentland desarrollaron una tecnología capaz de detectar rostros en fotografías llamando a esto como reconocimiento. El reconocimiento facial se ha convertido gradualmente en un levante de mercado en aplicaciones de seguridad, especialmente aquellas que involucran a países. En 2001, nació el marco de detección de objetos Viola-Jones, que propuso un algoritmo para detectar objetos en imágenes y se utilizó inmediatamente para detectar rostros con éxito. En la década actual ha surgido el aprendizaje automático y, hasta el día de hoy, son la mejor forma de detectar rostros humanos mediante potentes sistemas que pueden almacenar información en la nube. No fue hasta 2010 que esta tecnología entró al público a través de Facebook. A medida que el público va tomando conciencia de esta tecnología, se ha convertido en un tema controvertido porque muchos ciudadanos y los medios de comunicación están cuestionando la privacidad de este sistema. Para reducir esta controversia, Facebook anunció en septiembre de 2019 que cancelaría el uso del reconocimiento facial por defecto, permitiendo a sus usuarios decidir si usar el reconocimiento facial (*Presente y Futuro Del Reconocimiento Facial*, n.d.).

En muchos campos, con el único propósito de detección o como paso previo al reconocimiento de rostros, se desarrollan programas para la detección automática de rostros y se requiere la mayor precisión posible. Por ejemplo, en el campo de la seguridad, la detección automática de rostros puede ser de gran ayuda para los investigadores porque puede filtrar automáticamente imágenes o videos en función de la presencia o ausencia de rostros (García del Prado et al. 2020).

2.2 Conceptos generales

2.2.1 La imagen digital

Pixel es la abreviatura de palabras en inglés "Elemento de imagen" es el elemento más pequeño de la imagen en las cuales sé que pueden la información de lo cual está compuesta. En el modelo matemático de la imagen, los píxeles se identifican por su centro para que puedan representar el píxel como un punto (x, y) en el plano. Las imágenes digitales están compuestas por grupos píxeles, el valor de intensidad o brillo de cada píxel relacionado. Imagen digital de matriz bidimensional, de modo que cada elemento la matriz corresponde a cada píxel de la imagen (Herrero Vez & Villena Román, 2010).

En una imagen digital en blanco y negro de 1 bit cada píxel puede tener 2 valores: blanco o negro absoluto y en una imagen en blanco y negro de 2 bits cada píxel puede tener 4 valores: blanco, negro, gris claro y gris oscuro. En una imagen en blanco y negro de 8 bits (1 byte u octeto) cada píxel puede representar 256 valores desde el blanco absoluto al negro pasando por una gama intermedia de valores de gris. El ojo humano sólo es capaz de discernir simultáneamente un rango tonal relativamente pequeño por lo que con 8 bits se tiene suficiente información para representar con suficiente fidelidad la realidad que lo rodea (*Profundidad de Bits*, n.d.).

Depende del rango de cada valor se puede distinguir entre los siguientes tipos de imágenes:

1. Imagen binaria: solo forman el rango del valor de blanco y negro [0,1].

2. Imagen gris: hasta 256 niveles gris, por lo que su rango está en [0, 255].

3. Imagen en color: sus píxeles se han cuantificado utilizando tres componentes separados, estos tres pueden formar a todo color en componentes básicos.

Una imagen digital en tamaño de grises no es más que una matriz de números donde cada valor indica el nivel de gris de un píxel. Por ejemplo, la imagen de la segunda posición siguiente es una matriz de tamaño 256×256 donde los niveles del gris de la imagen varían desde 0 correspondiente al color negro hasta 255 correspondiente al color blanco. Los valores comprendidos entre 0 y 255 se escriben en binario con 8 bits, por eso esta imagen necesita 1 byte por píxel.

Figura 2.1 Imagen original

Fuente:(Autor)

Figura 2.2 Imagen escala de grises

Fuente:(Autor)

Figura 2.3 Imagen binaria

Fuente:(Autor)

2.2.2 Compresión de una imagen

La compresión de una imagen digital genera que dicha imagen ocupe menos espacio para poder aplicar diferentes técnicas y así la imagen pueda viajar más rápido por internet, que ocupe menos memoria en discos duros, etc. Las técnicas de compresión de una imagen digital están basadas en la descomposición en valores singulares de una matriz (Matas & Rojas, 2010).

Dada A una matriz real de dimensiones m×n, existe una matriz U ortogonal de orden m, una matriz S diagonal de dimensiones m×n y una matriz V ortogonal de orden n de forma que

$$A = USV^t \qquad (1)$$

Los elementos no nulos de la diagonal de S se denominan valores singulares de A y son números no negativos, pues son la raíz cuadrada de los autovalores no nulos de la matriz AtA o de AAt, ya que ambos coinciden salvo el autovalor nulo, si es que lo tienen. Las columnas de U y V se denominan vectores singulares por la izquierda y por la derecha respectivamente. Esta descomposición matricial se conoce como SVD (Singular Value Decomposition). Una propiedad muy interesante de la descomposición SVD de una matriz A es

Que el número de valores singulares no nulos de A coincide con el rango de dicha matriz. Es decir que una matriz A es de rango r y que los valores singulares no nulos están ordenados en orden decreciente (Matas & Rojas, 2010):

$$\mathtt{sl} \geq \mathtt{s2} \geq \cdots \geq \mathtt{sr} > 0.$$

Se puede comprobar que

$$A = USV^t = s_1 u_1 v_1^t + s_2 u_2 v_2^t + \cdots + s_r u_r v_r^t \qquad (2)$$

Donde sj es un escalar (el valor singular correspondiente), mientras que uj es la columna j-ésima de la matriz U y vj es la columna j-ésima de la matriz V. Si una matriz tiene rango r, entonces, según la ecuación (1), se podrá expresar como la suma

De r matrices de rango unidad. Es lógico pensar que, si los valores singulares son muy pequeños, entonces la suma de la ecuación (1) se podrá truncar en el momento en que la magnitud de estos valores singulares sea suficientemente pequeña, obteniendo de esta forma aproximaciones de la matriz A original. En efecto esto es así, ya que si se tiene.

$$A_1 = s_1 u_1 v_1^t \qquad (3)$$

Esta matriz es de rango 1 y es la matriz de rango 1 más parecida a A. así se llega a:

$$A_1 = s_1 u_1 v_1^t + s_2 u_2 v_2^t \qquad (4)$$

Esta matriz es de rango 2 y también resulta ser la matriz de rango 2 más parecida a A. Y así sucesivamente, de modo que la matriz de rango k más parecida a la original es la matriz

$$A_k = s_1 u_1 v_1^t + s_2 u_2 v_2^t + \cdots + s_k u_k v_k^t \qquad (5)$$

Siendo $1 \leq k \leq r$. Una definición precisa de lo que entendemos por matriz más «parecida» a la original puede encontrarse en noble.

2.2.3 Espacios de color

El espacio de color es un sistema de reproducción de color, es decir, la organización específica de colores en una imagen o video. Depende del modelo de color y de los dispositivos físicos que permiten representaciones de color reproducibles, como las aplicadas a señales analógicas (televisión en color) o representaciones digitales. El espacio de color puede ser arbitrario y se pueden asignar colores específicos y construir matemáticamente de acuerdo con el sistema.

El modelo de color es un modelo matemático abstracto que describe cómo representar colores como números, generalmente representados como tres o cuatro valores o componentes de color (RGB YCbCr y HSV son modelos de color). Sin embargo, un modelo de color que no tiene una función de mapeo asociada con un espacio de color absoluto es más o menos un sistema de color arbitrario que no está conectado a un sistema de reproducción de color.

2.2.2.1 Espacios de color RGB

El modelo de color RGB (1931) proviene del inglés Red-Green-Blue, que se basa en la síntesis acumulativa de la intensidad de la luz del rojo, verde y azul para lograr diferentes colores. Cada color aparece en los componentes principales del espectro rojo, verde y azul. El modelo se basa en un sistema de coordenadas cartesianas, que forma un cubo con valores RGB en 3 esquinas. El cian, el magenta y el amarillo están en las otras tres esquinas, el negro está en el origen y el blanco está en el vértice más alejado del origen (Herrero Vez & Villena Román, 2010).

La CIE (Comisión Internacional de Iluminación) define la longitud de onda del color primario de la luz como rojo 700 nm, verde 546,1 nm y azul 435,8 nm.

La figura siguiente se muestra una representación gráfica del espacio de color RGB, que está formado por un cubo unitario con ejes R, G y B.

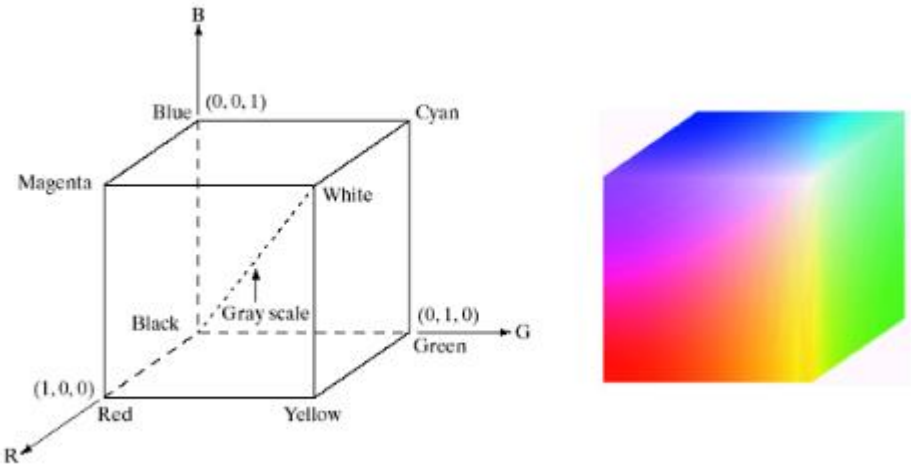
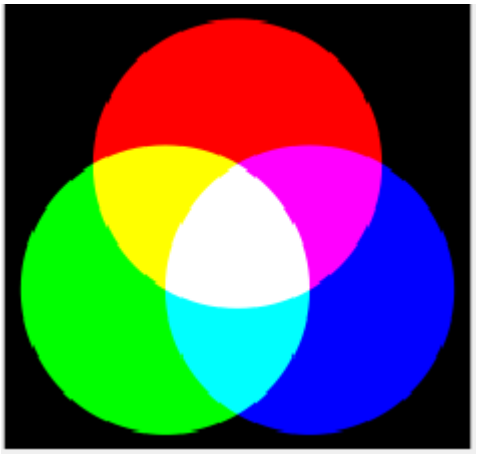


Figura 2.4 Diagrama esquemático del cubo RGB y el cubo a 24 bits

Fuente:(“Procesamiento Digital de Imágenes,” 1996)

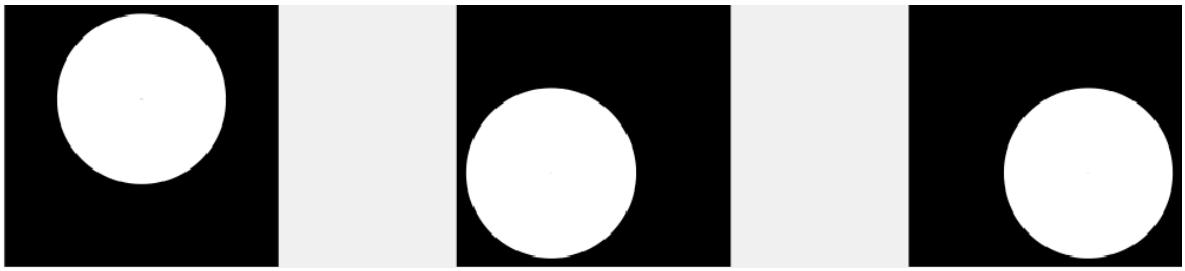
En la línea que conecta los puntos blancos y negros, la escala de grises (puntos con valores RGB iguales) varía de negro a blanco. Los diferentes colores en el modelo son puntos sobre o dentro del cubo, y están definidos por un vector que se extiende desde el origen. La imagen representada en el modelo RGB consta de 3 imágenes componentes, cada una de las cuales es un color primario. La cantidad de bits utilizados para representar cada píxel en el espacio RGB se denomina profundidad de píxel. Considere una imagen en la que las imágenes roja, azul y verde son todas imágenes de 8 bits. En estas condiciones, se dice que la profundidad de cada píxel de color RGB (tupla (R, G, B)) es de 24 bits. El número total de colores en una imagen RGB de 24 bits es (28) 3 = 16.777.216 (“Procesamiento Digital de Imágenes,” 1996).



(a)

Figura 2.5 (a) Espacios de color

Fuente: (Molinero Díez, 2010)



(b) **(c)** **(d)**

Figura 2.6 (b)Plano rojo (c)Plano verde (d) Plano azul

Fuente: (Autor)

Se pueden utilizar tres filtros sensibles al rojo, verde y azul para obtener imágenes en color. Al visualizar una escena en color con una cámara monocromática equipada con uno de estos filtros, el resultado es una imagen monocromática cuya intensidad es proporcional a la respuesta del filtro. Si se repite este proceso para cada filtro, se generarán 3 sub imágenes monocromas, que son la descomposición de los tres colores RGB de la escena de color, este proceso se puede visualizar en la figura anterior.

2.2.2.2 Modelo de color HSV

Su componente principal es la tonalidad (Hue), saturación y valor. Las coordenadas que siguen inmediatamente son coordenadas cilíndricas, el espacio que define el color es la pirámide inferior. Hexagonal

El modelo de color HSV es una conversión no lineal del modelo RGB a coordenadas cilíndricas, por lo que cada color está definido por las siguientes expresiones:

- Matiz: el ángulo que representa el matiz, normalmente definido entre 0 y 360 grados.
- Saturación: El nivel de saturación del color, entre 0 y 1. 0 significa que no hay saturación (blanco), y el más alto es 1, que es el tono de toda su intensidad. El porcentaje suele ser 0% -100%.

- Brillo: el nivel de brillo está entre 0 y 1. 0 es negro; de lo contrario, es 0. 1. Blanco. Como la saturación, su porcentaje puede estar entre 0% y 100%. Por tanto, el 50% representa un nivel medio o normal de brillo de color (*Modelo HSV - EcuRed*, n.d.).

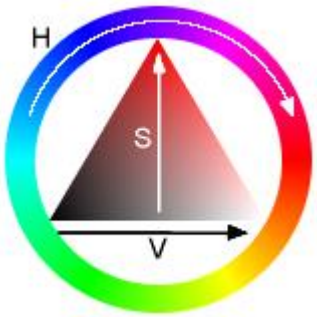


Figura 2.7 Diagrama de representación gráfica del modelo HSV

Fuente:(*Modelo HSV - EcuRed*, n.d.)

2.2.4 Procesamiento de la imagen

El procesamiento de imágenes tiene como objetivo mejorar la apariencia de la imagen y hacer que ciertos detalles que desea notar sean más obvios en la imagen. La imagen puede haber sido producida de diversas formas, por ejemplo, fotográfica o electrónicamente mediante un monitor de televisión. El procesamiento de imágenes generalmente se puede realizar en una computadora mediante métodos ópticos o digitales. La tecnología de pretratamiento tiene como objetivo mejorar los atributos de la imagen para facilitar los pasos posteriores del análisis de la imagen. El método mejorado puede basarse en operaciones puntuales u operaciones vecinas.

El objetivo principal de la tecnología de mejora de imágenes es hacer frente a la imagen para hacerla más adecuada para aplicaciones específicas o posprocesamiento. Por tanto, la utilización de una técnica u otra depende del problema específico a resolver. Los métodos de mejora de la imagen se pueden dividir en dos campos diferentes: métodos en el dominio de la frecuencia y métodos en el dominio espacial. El primero se basa en la transformada de fourier de la imagen modificada, mientras que el segundo se basa en la manipulación directa de las propiedades con las que cuenta una imagen como lo es su tamaño o sus píxeles, en los cuales se puede modificar para así tener una expresión diferente.

2.2.4.1 Operaciones puntuales

Las operaciones puntuales son conversiones uno a uno, es decir, son independientes del píxel 'q' en la posición (i, j) en función del píxel 'p' de otra imagen, pero en la misma posición, es decir, (i, j).

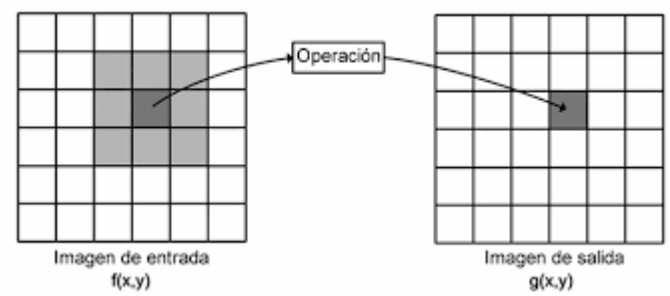


Figura 2.8 Transformación espacial sobre una imagen

Fuente: (Imágenes, n.d.)

2.2.4.2 Modificación del contraste

El ajuste de contraste asigna el valor de intensidad de la imagen a todo el rango de visualización del tipo de datos. Una imagen con buen contraste tiene diferencias obvias entre blanco y negro. Para ilustrar este punto, la figura siguiente (b) tiene un contraste pobre y el valor de intensidad está limitado a la mitad del rango. La imagen (c) tiene un mayor contraste y su valor de intensidad ocupa todo el rango de intensidad [0, 255]. En una imagen de alto contraste, las luces se ven más brillantes y las sombras se ven más oscuras.

(a)

(b **c)**

Figura 2.9 (a) Imagen original (b)Contraste bajo (contraste alto

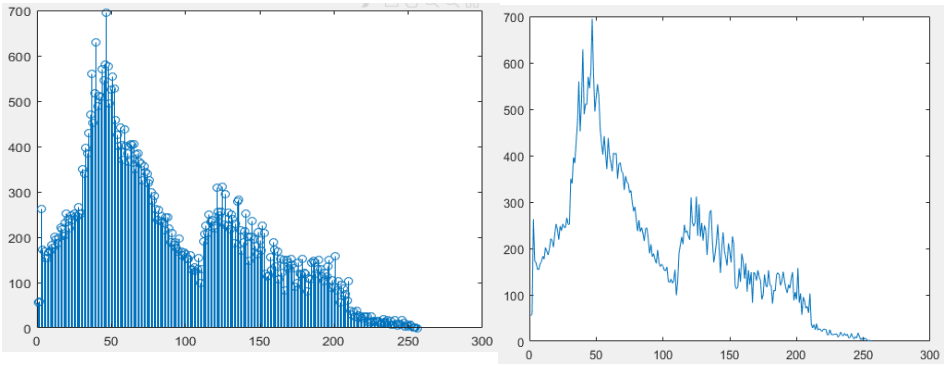
Fuente: (García del Prado et al., 2020)

2.2.4.3 Modelado del histograma

El histograma es una representación de frecuencia que indica el valor relativo de cada color en la imagen. Por tanto, la modificación del histograma intenta actuar sobre la imagen completa de una manera global para ajustar el histograma a una forma específica. El propósito de la ecualización del histograma es tener el mismo número de píxeles en todos los niveles de gris. La especificación del histograma se utiliza para obtener mejores resultados y así indicar la imagen de salida que contiene los resultados finales. La operación basada en

histograma puede ser elaborada a partir de un histograma local: De esta forma, el contorno del histograma se adaptará a los atributos locales de la imagen (Herrero Vez & Villena Román, 2010).

(a)



(b) **(c)**

Figura 2.10 (a) Imagen original (b)Contorno del histograma (c)Histograma de la imagen

Fuente: (Autor)

2.2.4.4 Operaciones entre imágenes

Las operaciones entre imágenes incluyen el valor del píxel P [i j] a partir de determinar el valor del píxel P [i j] Píxeles P'[i, j] y P' '[i, j], donde P' y P '' Son dos imágenes diferentes. Los puntos principales, algunos de ellos, son muy importantes en el procesamiento de imágenes. Por ejemplo, la suma permite superponer imágenes y promediarlas para reducir el ruido aleatorio adicional. La resta elimina la interferencia adicional y también funciona para el movimiento entre imágenes de una misma escena. La multiplicación permite eliminar

partes si el producto está hecho con máscara, sólo se conserva la imagen de abajo (Herrero Vez & Villena Román, 2010).

2.2.4.5 Operaciones espaciales

Es una operación aplicada a la imagen para mejorar o simplificar los detalles espaciales para mejorar los efectos visuales de una imagen. Los ejemplos comunes incluyen la aplicación de diferentes filtros para mejorar el detalle de los bordes en una imagen o reducir o eliminar patrones de ruido. La acción espacial es una operación "local" que modifica el valor de cada píxel en función del valor de los píxeles circundantes. Se definen en el entorno vecindad del punto a convertir.

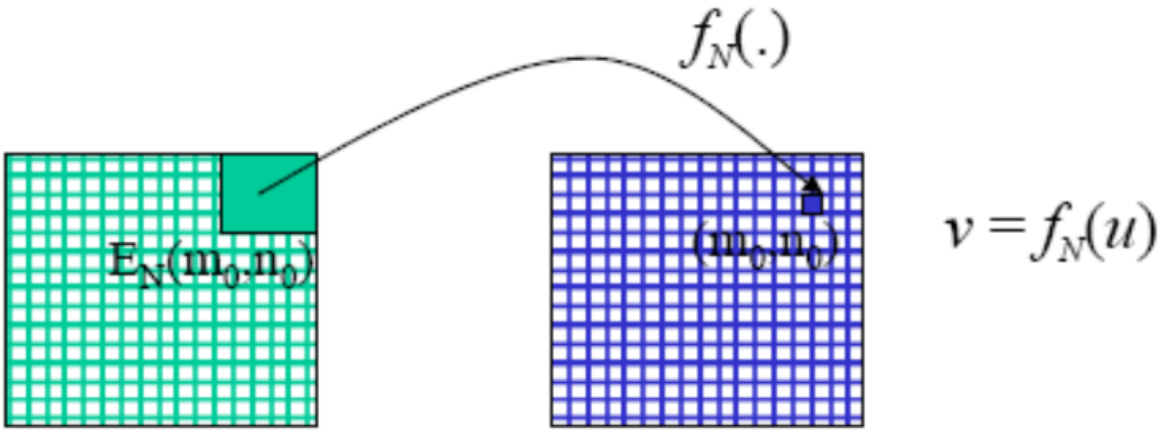


Figura 2.11 Operación espacial

Fuente: (Herrero Vez & Villena Román, 2010)

La frecuencia espacial define la magnitud del cambio. La cantidad de datos por unidad de distancia en un área determinada de la imagen. El área de la imagen donde el valor de la imagen cambia poco o cambia gradualmente se denomina área de baja frecuencia. El área donde ocurren grandes cambios o una transición rápida se denomina área de alta frecuencia.

2.2.4.6 Ruido en imágenes

Es la data no deseada que contamina a una imagen. g(x,y)=f(x,y)+r(x,y) el origen puede estar tanto en el proceso de adquisición de la imagen (errores en los sensores), como en el de transmisión (debido a interferencias en el canal de transmisión). Tipos de ruido en la imagen existen distintos modelos de ruido, según las funciones de densidad de probabilidad que sigan sus intensidades dentro de una imagen.

A continuación, se observa un ejemplo de ruido encontrado en imágenes.

2.2.4.6.1 Ruido impulsivo (o sal y pimienta)

Se produce normalmente en la cuantificación que se realiza en el proceso de digitalización y está dada por la siguiente gráfica en la cual se observa un cambio brusco en las diferencias entre (a) y (b).

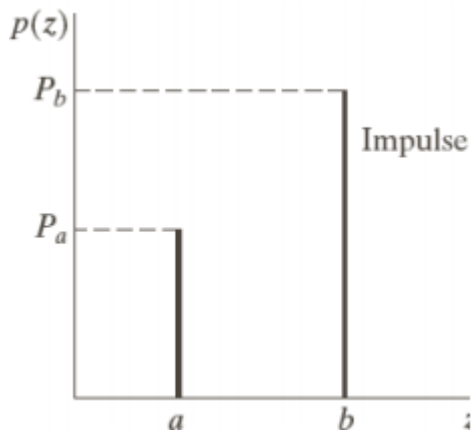
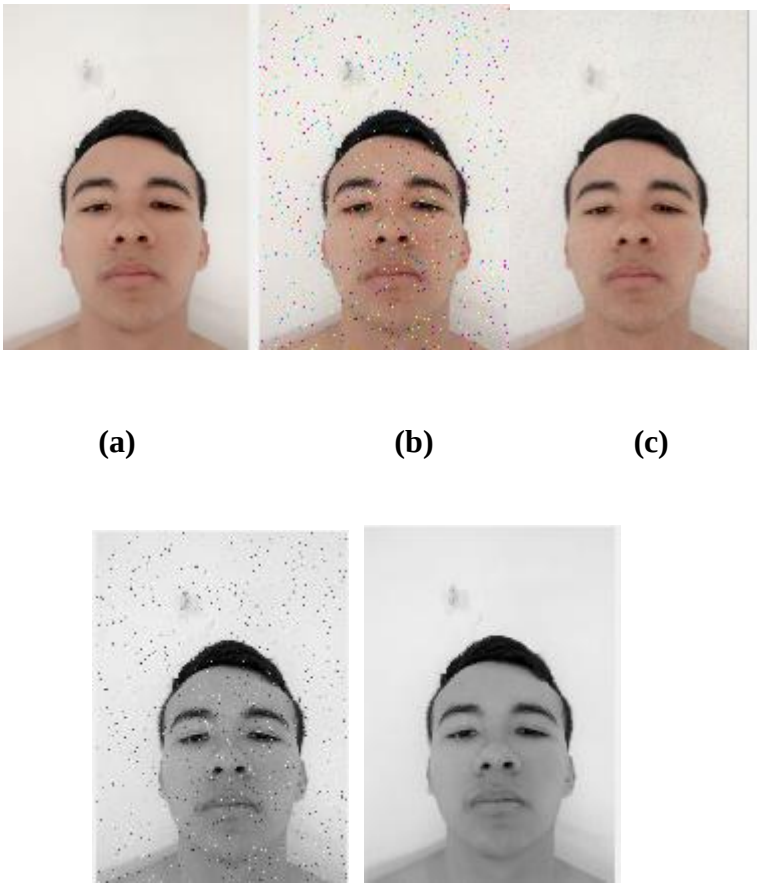


Figura 2.12 (a) Ruido impulsivo

Fuente: (Tema, 2017)



(a) **(b)** **(c)**

(d) **(e)**

Figura 2.13 (a) Imagen original (b)Imagen con ruido impulsivo (c)Imagen filtrada (d)Imagen con ruido impulsivo en escala de grises (e)Imagen filtrada

Fuente: (Autor)

2.2.3.7 Filtros paso alto

Destacan las altas frecuencias, eliminan las bajas frecuencias y mejora las características lineales. Dejar pasar las altas frecuencias dará como resultado imágenes llenas de bordes y discontinuidades, y la eliminación de los términos de baja frecuencia, reducirá significativamente el contraste general de la imagen.

(a) (b)

Figura 2.14 (a) Imagen original en escala de grises (b) Filtro pasa altas

Fuente: (Autor)

2.2.3.8.1 Dilatación

Su finalidad es rellenar pequeños huecos y espacios de menor tamaño o más pequeños que los elementos estructurales. El resultado de la dilatación es un conjunto de puntos barridos por la matriz que identifica el píxel de la imagen que se está procesando, y algunos puntos coinciden con una de las imágenes originales. Esta es una operación extensa, es decir, el resultado contiene la imagen original pero modificada en algunos pixeles.

(a) *(b)*

Figura 2.15 (a) Imagen original en escala de grises (b) Filtro dilatación

Fuente: (Autor)

2.2.3.8.2 Erosión

Su finalidad es eliminar grupos de pixeles en la matriz que identifica el píxel de la imagen que se está procesando, como pequeñas islas o baches. Siempre que todos los puntos estén incluidos, la salida de erosión es un conjunto de puntos barridos por el centro del elemento estructural. Esta es una operación anti expansión, es decir, la imagen contiene el resultado final como se observa en la siguiente figura.

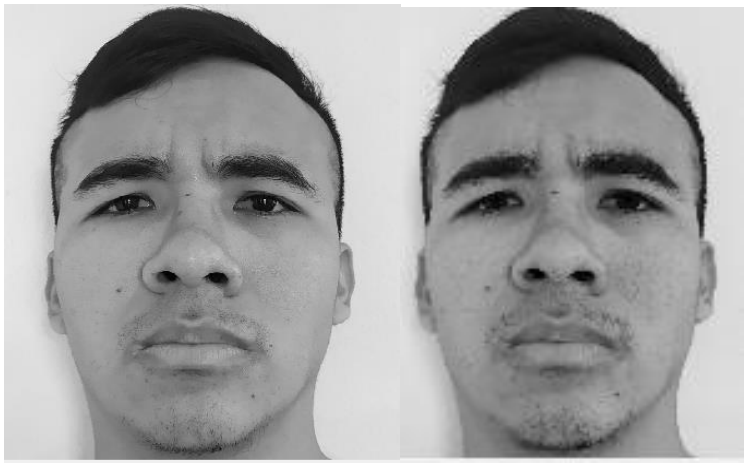


Figura 2.16 (a) imagen original en escala de grises (b) imagen filtro erosión

Fuente: (Autor)

2.2.3.8.3 Apertura

La apertura es una combinación de erosión y dilatación, que se considera una operación anti expansión, se conserva la imagen original. Algunos de sus efectos son: suavizar contornos, eliminar protuberancias, separar objetos en puntos estrechos y suavizar bordes.

(a) **(b)**

Figura 2.17 (a) Imagen original en escala de grises (b) Imagen filtro apertura

Fuente: (Autor)

2.2.3.8.4 Cierre

El cierre es una combinación de expansión y erosión. Algunos de sus efectos son: rellenar huecos y grietas, eliminar entradas y conectar objetos adyacentes difuminando la imagen un poco como se observa en la siguiente figura.

(a) (b)

Figura 2.18 (a) Imagen original en escala de grises (b) Imagen filtro cierre

Fuente: (Autor)

2.2.3.8 Técnicas basadas en los bordes

La idea detrás de la mayoría de las tecnologías de detección de bordes es el cálculo de los operadores de bypass locales, porque si el nivel de gris entre un píxel y sus vecinos cambia bruscamente, el píxel pertenece al borde. Para localizar los contornos en la escena, después de utilizar la segmentación, se utilizará la tecnología de detección de bordes. Por tanto, es necesario realizar otras tareas en la etapa de preprocesamiento de imágenes. Sin

embargo, estas técnicas no pueden determinar finalmente el objeto en la imagen. La presencia de ruido, el efecto de las sombras, la falta de iluminación uniforme hace que los contornos sean incompletamente continuos y cerrados sobre el objeto como se observa en la siguiente figura.

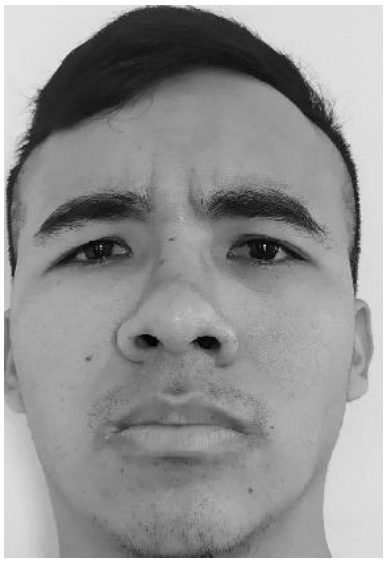
(a) (b)

Figura 2.19 (a) Imagen original (b) Contornos

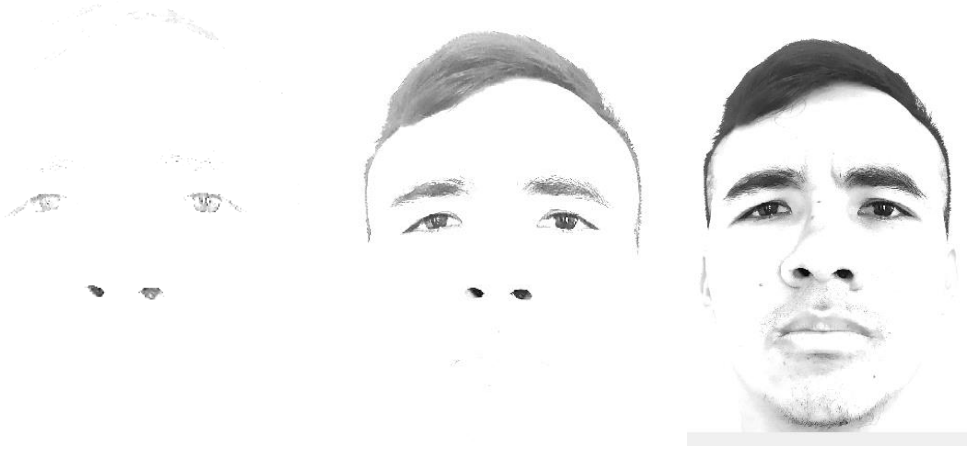
Fuente: (Autor)

2.2.3.9 Umbralización

El umbral de imagen es el proceso de buscar el mejor promedio, este promedio distingue entre objetos en el fondo y objetos en primer plano en la imagen. Este umbral es el histograma de la imagen el cual se divide en dos puntos (o valores) máximos. En la mayoría de los casos debido a la complejidad de los gráficos, es difícil encontrar el valor gráficamente estos histogramas. Es por eso que se utilizan métodos paramétricos y no paramétricos para indicar el modelo. Un ejemplo de la umbralización es el área independiente de la imagen corresponde al objeto que se requiere analizar. Esta separación se basa en el cambio de intensidad entre el píxel del objeto y el píxel de fondo Para distinguir el píxel de interés del resto de píxeles, se compara el valor de intensidad de cada píxel con un umbral. Una vez que los píxeles importantes están correctamente separados, se pueden establecer en un cierto valor para su identificación (es decir, se pueden asignar (negro), (blanco) o cualquier valor que se adapte a sus necesidades).



(a)



(b) (c) (d)

Figura 2.20 (a) Imagen original (b) Umbral al 100 (c) Umbral al 127 (d) Umbral al 150

Fuente: (Autor)

2.3 Detección de rostros

Es el proceso de detección de rostros, es decir, dada una imagen, determinar si contiene una sub imagen que representa un rostro humano y así poder ubicarla para su posterior procesamiento teniendo en cuenta los diferentes tratamientos para no perder información. Las emociones para ser consideradas como básicas son:

- Poseer sentimientos específicos distintivos.
- Manifestar propiedades motivacionales y organizativas de funciones adaptativas.
- Tener un sustrato neural específico y distintivo.
- Tener una expresión o configuración facial específica y distintiva.

En general, detectar rostros en imágenes tiene múltiples aspectos problemáticos. En los cuales se puede encontrar los siguientes:

- Problemas de iluminación (no uniformidad).
- Condiciones generales de la imagen (ruido, fondo).
- Cantidad desconocida de caras en la imagen.
- Pose y orientación de la cara.
- Presencia de gafas, barba, gorro, etc.
- Expresión de la cara.

Hoy en día hay muchos códigos o algoritmos en los cuales se aplica el rendimiento de las características de una imagen en el cual depende de la detección de rostros de la escena a considerar. Una descripción de alto nivel de ellos es la posibilidad de clasificaciones de algoritmos de detección de rostros variando o descartando pixeles.

La identificación y el reconocimiento facial conlleva dos fases en visión artificial. Primero, se debe encontrar a una persona en la imagen a examinar y después llevar a la identificación de dicha persona comparándola con una base de datos, todo ello de forma automática. Es importante diferenciar ambos procesos y para ello se representa en la siguiente figura.

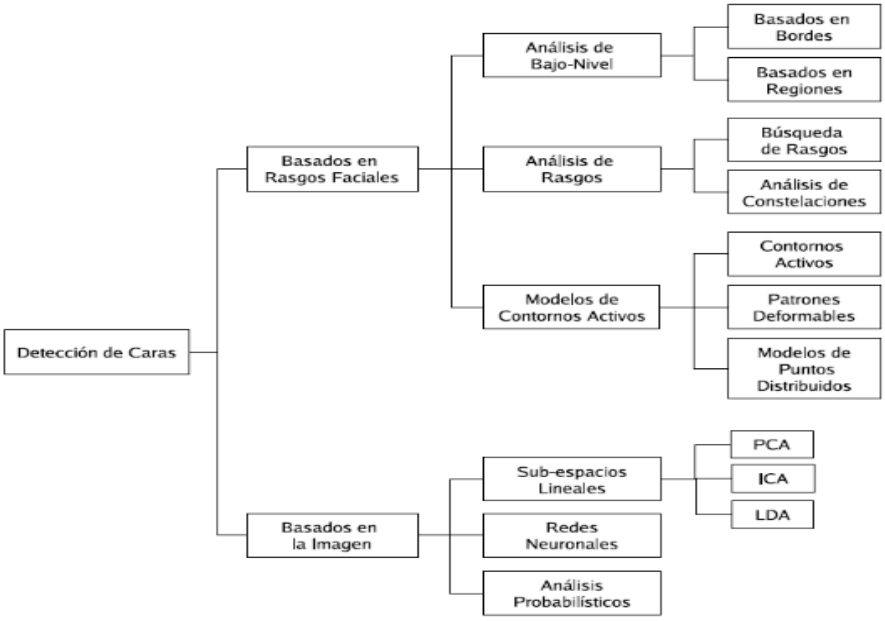


Figura 2.21 Diagrama de detección de caras

Fuente:(Herrero Vez & Villena Román, 2010)

2.3.1 Viola -Jones

El algoritmo Viola-Jones propuso un método para detectar objetos en 2001 con unos resultados bastante favorables teniendo altos rangos en las características de iluminación y escala.

Es uno de los algoritmos más utilizados en la detección de rostros actualmente. El método Viola Jones es un método de aproximación basado en la apariencia. Se divide en dos etapas, la primera etapa es el aprendizaje del clasificador basada en una gran cantidad de ejemplos positivos y la segunda etapa es la detección mediante un clasificador el cual contiene imágenes conocidas (Ekman, 2005).

Viola y Jones deciden utilizar el valor de cada pixel y basar el algoritmo en características más simples dentro de la imagen. de esta manera las características se pueden utilizar para codificar datos que con una cantidad finita de datos de entrenamiento se tienen mejores resultados (Mihai, 2014).

2.3.2 Detección

Se recomienda utilizar imágenes intermedias para calcular rápidamente característica de rectángulo. Esta imágen se llama imagen integral, es decir una representación de imagen del mismo tamaño que la imagen original, donde cada la suma de puntos se calcula como la suma de los píxeles de arriba y de la izquierda (Chandra et al., 2018).

$$ii(x,y) = \sum_{x \leq x', y' \leq y'} i(x', y') \qquad (6)$$

En la ecuación, ii(x,y) es la imagen integral e i(x,y) es la imagen original.

Para implementar (1), se utilizan el siguiente par de ecuaciones:

$$s(x,y) = s(x,y-1) + i(x,y) \qquad (7)$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \quad (8)$$

Donde s(x,y) es la suma acumulativa de las filas, siendo s(x,-1=0) e ii(-1,y)=0. Con este par de ecuaciones se puede calcular la imagen integral en una sola pasada de la imagen original.

Con la imagen integral ya calculada, cualquier suma rectangular puede ser calculada utilizando cuatro puntos de la matriz. En el siguiente ejemplo, el valor del punto 1 es el de la suma de todos los píxeles de la zona A, el de 2 es A + B, el de 3 es A + C, y el de 4 es A + B + C + D, por lo que, si queremos conocer la suma de los píxeles dentro del rectángulo D, se puede calcular cómo 4+1-(2+3).

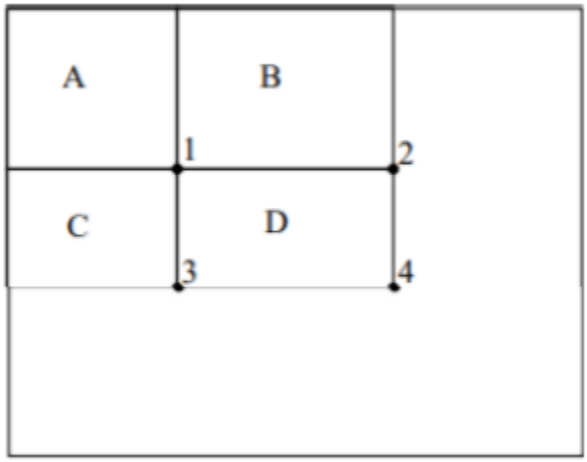


Figura 2.22 Ejemplo imagen integral

Fuente:(Mihai, 2014)

2.3.3 Entrenamiento del clasificador

Para el entrenamiento de clasificadores, Viola y Jones usan una variante de AdaBoost para un pequeño conjunto de funciones. La idea original de este algoritmo es mejorar el rendimiento de la clasificación de algoritmos de aprendizaje simples.

AdaBoost es un método de clasificación que combina varios clasificadores básicos para formar un clasificador único, más complejo y preciso. La idea se basa en la propuesta de que, siempre que tenga una cantidad suficiente de muestras de entrenamiento, puede combinar varios clasificadores simples (la precisión de cada clasificador es ligeramente mayor que la de un clasificador aleatorio) para formar un clasificador de mayor precisión (Chandra et al., 2018).

El algoritmo Viola Jones usa AdaBoost mezclando una serie de clasificadores AdaBoost (como cadenas de filtros). Cada filtro es un clasificador AdaBoost separado, compuesto por varios clasificadores débiles. Si cada uno de estos filtros en el área de

aceptación de imágenes indica que un rostro humano no está calificado, el área se clasificará inmediatamente como un rostro no humano (Hirvin Gonzalez, 2019).

En este algoritmo, se selecciona cada ciclo que promueve características entre todas las demás características potenciales y, finalmente, la clasificación final será una combinación lineal de clasificaciones iniciales débiles.

Dado un conjunto de imágenes $(x_{i,}y_i),\ldots,(x_{n,}y_n)$ donde y_i=0,1 para muestras negativas y positivas respectivamente.

Inicializar los pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_{i=0,1}$ donde m es el número muestras negativas y l es el número de muestras positivas.

Normalizar los pesos.

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \qquad (9)$$

Donde wt es la distribución de probabilidad. Por cada característica j, entrena un clasificador hj el cual es restringido por una simple característica. El error es evaluado con respecto a wt.

$$\epsilon j = \sum_{i} w_{i|hj(xj)-yi|} \qquad (10)$$

En la ecuación (9) se escoge el clasificador con menor error y en la ecuación (10) se actualizan los pesos.

$$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i} \qquad (11)$$

Donde ei=0 si el ejemplo xi se clasifica correctamente y 1 en caso contrario.

$$h(x) = \{1, \quad si \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \ \ 0, \quad en\ caso\ contrario \qquad (12)$$

Donde

$$\alpha_t = \frac{1}{\beta_t} \qquad (13)$$

2.3.4 Clasificador en cascada

En lugar de utilizar el proceso de Adaboost para construir un solo clasificador, se puede construir un clasificador más pequeño y efectivo para repeler muchas ventanas negativas (es decir, aquellas que no tienen ninguna instancia del objeto de búsqueda), mientras conserva casi todas las ventanas positivas (Es decir, aquellas instancias que contienen el objeto de búsqueda). Estos clasificadores más simples se usan para rechazar la mayoría de las ventanas de búsqueda y solo aquellos con la mayor probabilidad de encontrar un clasificador de caras reduciendo el número de errores positivos (Hirvin Gonzalez, 2019).

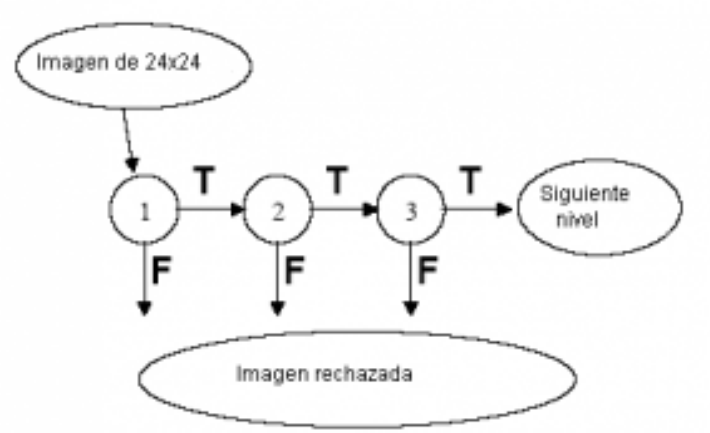


Figura 2.23 Clasificador en cascada

Fuente: (Miao, 2014)

Se obtiene un clasificador en cascada, cada uno de ellos recibe entrenamiento AdaBoost, y luego se ajusta su umbral para minimizar la cantidad de errores negativos. La cascada de entrenamiento Viola-Jones tiene 38 etapas y más de 6000 funciones, en promedio, cada uno solo evalúa 10 características en la ventana de búsqueda.

2.3.5 Pre procesado de la imagen

En la etapa del reprocesado se trata de convertir a cada rostro detectado previamente para que la extracción de características sea más efectiva y que facilite las operaciones. En esta etapa está el escalado, el recorte y la ecualización para después de completar la detección de rostros.

La escala implica proporcionar un cierto tamaño para las caras para que todas las caras tengan las mismas proporciones. Para hacer esto, se recomienda la distancia entre los centros de los dos ojos. Usando esta distancia como referencia para el proceso de zoom se puede dar cuenta de que las proporciones de todas las imágenes son muy similares, simplificando así la tarea de comparación.

1.1.1.1 2.3.5.1 Distancia Euclídea

La distancia euclidiana es una de las medidas más básicas para calcular la distancia. La distancia se define como la distancia directa entre dos puntos del plano. El ejemplo más claro es la distancia entre dos puntos en un plano bidimensional con coordenadas x y y. Si tenemos dos puntos P1 y P2 con coordenadas (x1, x1) y (x2, y2) respectivamente, el cálculo de la distancia euclidiana entre ellos será (Chandra et al., 2018).

$$d_E(P_1, P_{2)} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (14)$$

En general, la distancia euclídea entre dos puntos P=(P1,P2,…,Pn) y Q=(Q1,Q2,…,Qn) en el espacio euclídeo n-dimensional vendría definida

$$d_E(P, Q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \qquad (15)$$

2.4 Modelo LBP

La idea principal de LBP operador de Patrones Binarios Locales es resumir la estructura local de una imagen mediante las cualidades de cada píxel con sus píxeles vecinos. Se toma un píxel como centro y se limita el valor de los vecinos. Es decir, si la intensidad del píxel que está en el centro es mayor que su vecino, este se denota con un cero, y si es al contrario, la intensidad del vecino es mayor o igual a la intensidad del vecino central, entonces se denota con un uno(Tutor & Fern, 2015).

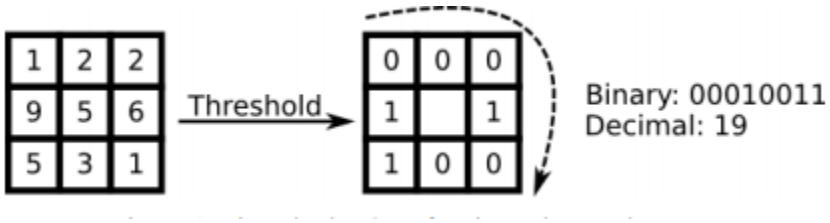


Figura 2.24 Ejemplo de cómo funciona el operador LBP

Fuente: (Tutor & Fern, 2015)

Descripción del algoritmo

Una descripción más formal del operador LBP se puede dar como:

$$LBP(x_{c,}y_c) = \sum_{p=0}^{p-1} 2^p s(i_p - i_c) \qquad (16)$$

con $(x_{c,}y_c)$como el píxel central con intensidad i_c; e i_n la intensidad del píxel vecino. s es la función signo definida como:

$$s(x) = \{1, \quad si\ x \geq 0\ 0, \quad e.0.c$$

Esta descripción permite capturar con detalles y con mucha precisión cada una de las imágenes. Para un punto $(x_{c,}y_c)$, la posición de su vecino $(x_{p,}y_p)$, $p^{\epsilon p}$, puede ser calculado como:

$$x_p = x_c + R\ cos\ cos\ \frac{2\pi p}{p} \qquad (17)$$

$$y_p = y_c + R\ sin\ sin\ \frac{2\pi p}{p} \qquad (18)$$

donde R es el radio del círculo que forman los vecinos y P es el número de puntos de muestra. El operador es una extensión de la codificación original LBP, por eso a veces es llamado LBP Extendido. Si los puntos coordenados en el círculo no corresponden a las coordenadas de la imagen, el punto debe ser interpolado. Open implementa una interpolación bilineal:

$$f(x,y) \approx\ [1 - xx]\begin{bmatrix} f(0,0)\ f(0,1) \\ f(1,0)\ \ f(1,1) \end{bmatrix}\begin{bmatrix} 1-y \\ y \end{bmatrix} \qquad (19)$$

Se denomina al operador LBP como robusto frente a transformaciones en escala de grises monótonas. Por último, se incorpora la información espacial en el modelo de identificación facial. La representación propuesta es dividir la imagen LBP en regiones locales y extraer la información del histograma de cada una. El vector de características espaciales mejorado es obtenido mediante la concatenación de los histogramas locales. Estos histogramas son llamados como Histogramas de Patrones Binarios Locales (Mihai, 2014).

2.5 Estados de ánimo

A través de los años se ha llevado a cabo diversos intentos de predecir los estados de ánimo de una persona a través de dispositivos electrónicos por eso se hace conocer como comienza a darse a conocer cada una de las etapas.

2.5.1 Historia

Antes de la aparición de Ekman (los antropólogos, incluida Margaret Mead), generalmente creían que las expresiones faciales y las emociones que representaban estaban determinadas por la cultura: la gente aprendió a hacer y leer expresiones faciales. Ekman comenzó a probar esta idea en 1968. Viajó a Papúa Nueva Guinea para estudiar las expresiones faciales de los miembros de la remota tribu Forr, donde aprendió que pueden reconocer constantemente las emociones en las expresiones faciales al mirar las fotos de otras personas. Incluso si la tribu no ha estado expuesta a ninguna cultura externa, no hay cultura.

Los hechos han demostrado que las expresiones faciales son transculturales. Su investigación muestra que tanto el mundo occidental como el mundo oriental utilizan ciertas expresiones faciales comunes. Esta lista de expresiones faciales comunes publicada por Ekman en 1972 contiene seis emociones básicas. (*LAS SEIS EMOCIONES BÁSICAS* | ., n.d.)

En 1991 el psicólogo Izard quiso plasmar las diferentes emociones que podía sentir los seres humanos, inició creando una lista con los requisitos que debían cumplir aquellas emociones para ser consideradas como básicas las cuales son:

- Tener una expresión o configuración facial específica y distintiva.
- Poseer sentimientos específicos distintivos.
- Derivar de procesos biológicos evolutivos.
- Manifestar propiedades motivacionales y organizativas de funciones adaptativas.

Con todo esto, Izard llegó a la conclusión de que había 8 emociones que cumplían con cada uno de los requisitos anteriores (Placer, interés, sorpresa, tristeza, ira, asco, miedo y desprecio). Pero finalmente fue Ekman, otro de los autores relevantes en el estudio de la emoción quien determinó según los requisitos anteriores 6 emociones básicas: ira, alegría, asco, tristeza, sorpresa y miedo. Mucho antes de que en los años 90 estos psicólogos determinarán las características que diferencian cada emoción, otros intelectuales también repararon en el origen genético de algunas expresiones faciales.

Aristóteles dijo “Hay expresiones de la cara características que son observables para acompañar la cólera, el miedo, la excitación erótica y todas otras pasiones” (Herrero Vez & Villena Román, 2010).

2.5.2 Propiedades de los estados de ánimo básicos

2.5.2.1 Asco

Según la teoría de Ekman, el disgusto es una de las emociones básicas. Significa una posibilidad (real o imaginaria) de rechazar o evitar la ingesta de sustancias peligrosas con propiedades contaminantes. El sentimiento subjetivo es una de las grandes insatisfacciones y una de las obvias aversiones por la estimulación ofensiva. Muestra un papel fisiológico central en diferentes problemas gastrointestinales y náuseas. La activación en vivo también aumenta generalmente.

Esta es la emoción más desagradable entre las seis. Tiene una poderosa capacidad para regular nuestro comportamiento, especialmente en la comida. La función del disgusto no solo es protectora desde el momento en que se procesa, sino también para el futuro, porque es un cincel en nuestra memoria (*6 Emociones Básicas, Latidos de Un Lenguaje Universal - La Mente Es Maravillosa*, n.d.).

2.5.2.2 El miedo

El miedo es la emoción más estudiada en animales y humanos. Se trata de un estado emocional negativo o repugnante, que tiene un alto poder de activación y puede fomentar el evitar y evitar situaciones peligrosas. La experiencia hace que la gente se sienta muy nerviosa, pero también preocupada por su seguridad y salud. La relevancia fisiológica nos muestra el rápido aumento de la activación y la preparación para el vuelo. La actividad del corazón aumenta bruscamente y la actividad respiratoria se acelera, lo que resulta en una respiración superficial e irregular (Ruse, 2010).

2.5.2.3 Alegría

De todas las emociones básicas, la felicidad es quizás la más positiva: está directamente relacionada con el placer y la felicidad. Por ejemplo, esto se basa en la resolución de ciertos objetivos personales o en la reducción del malestar. Por la forma en que lo expresamos, parece que no solo es un reflejo de nuestro estado interno, sino que no tiene ningún efecto en nuestra supervivencia. Sin embargo, la felicidad es uno de los sistemas del cuerpo para fomentar la acción. Además, también es una recompensa por acciones que son beneficiosas para uno mismo. Cuando realizamos una acción que satisface la meta, es decir, cuando se desencadena la alegría, por ello, la acción se repetirá para restaurar esa sensación de placer. Puede que sea el refuerzo más natural que tenemos (Ekman, 2005).

2.5.2.4 Tristeza

De todas las emociones básicas basadas en la psicología, la tristeza puede ser la más negativa. Esta emoción significa una disminución del estado de ánimo y una disminución significativa de las actividades cognitivas y conductuales. Aunque esta emoción tiene mala reputación, su papel es tan importante o incluso más importante que otras emociones básicas. El propósito del duelo es actuar cuando el sujeto es impotente o no puede tomar ninguna acción directa. Un ejemplo es la pérdida de un ser querido. El duelo reduce los niveles de

actividad, por lo que el organismo intenta ahorrar recursos y evitar esfuerzos innecesarios (*Las 6 Emociones Básicas Según La Psicología*, n.d.).

2.5.2.5 Ira

Suele tener diferentes características en otras personas. La ira genera poder. Podemos imaginar que, en una batalla, el hecho de que un oponente golpee a otro oponente aumentará su ira. La ira es el producto de la diferencia entre el estado de ocurrencia y otro estado que deseamos que ocurra. Por otro lado, la mentalidad de concesión habitual es una sensación de frustración. Esta situación lleva a la gente a insistir en identificar al culpable. De lo contrario, también puede ser quien ocupe tu puesto. La razón de esto es que es una emoción que necesita deshacerse de ella con urgencia (Ekman, 2005).

2.5.2.6 Sorpresa

La sorpresa se puede definir como una reacción provocada por algo imprevisto, novedoso o extraño. Es decir, cuando apareció el estímulo, los sujetos no pensaron en su predicción o plan. La experiencia subjetiva que sigue es una sensación de incertidumbre y un estado de persona con pensamientos en blanco (Ruse, 2010).

Tabla 2-1 Expresiones faciales.

Fuente:(Bartual González, 2017)

Emoción	Definición	Expresión
Ira 	El antagonismo hacia una persona o un objeto a menudo se sentía después de que usted siente que ha sido agraviado u ofendido	La reducción de cejas, apretar y estrechar los labios, los ojos mirando fijamente, apretando los párpados inferiores, con menos frecuencia,

		empujando la mandíbula hacia delante
Asco 	Desagrado intenso o condena causada por algo ofensivo o repulsivo	La reducción de cejas, curvando el labio superior, arrugando la nariz
Tristeza 	Sentimiento de infelicidad o tristeza	Los párpados caídos, la reducción de las esquinas de la boca, labios fruncidos, los ojos bajos.
Sorpresa 	Sensación de Malestar o sorpresa ante un hecho inesperado	Levantando las cejas altas (que puede causar arrugas en la frente), abriendo los ojos como platos, dejando caer la mandíbula.

	Sensación de aprehensión provocada por la percepción de peligro, amenaza o imposición de dolor.	Levantando las cejas/ dibujar las cejas juntas, tensando los párpados inferiores, que se extiende horizontalmente labios, la boca ligeramente abierta.
	Agradable sensación de satisfacción y bienestar	Sonriendo- tirando hacia arriba las comisuras de los labios, contrayendo los músculos grandes orbitales alrededor de los ojos.

2.6 Herramientas de desarrollo de código

Las plataformas de desarrollo son demasiado importantes hoy en día. Por eso se van a resaltar cuáles son las más importantes y en las cuales es más complejo a la hora importar librerías u otras ayudas que mejoren los resultados desde la etapa inicial a la final.

2.6.1 Herramientas

2.6.1.1 Anaconda

Anaconda es una suite de código abierto que contiene una serie de aplicaciones, bibliotecas y conceptos diseñados para desarrollar ciencia de datos usando Python. Por lo general, la distribución de Anaconda es una distribución de Python que puede actuar como administrador de entorno, administrador de paquetes y tiene una colección de más de 720

paquetes de código abierto (*Anaconda Distribution: La Suite Más Completa Para La Ciencia de Datos Con Python*, n.d.).

2.6.1.1.1 Características de Anaconda

- Documentación gratuita, de código abierto, muy detallada y una comunidad sólida.
- Multiplataforma (Linux, macOS y Windows).
- Le permite instalar y administrar paquetes de ciencia de datos, dependencias y entornos usando Python de una manera muy simple.
- Se utiliza varios IDE como Jupyter, JupyterLab, Spyder y RStudio para desarrollar proyectos de ciencia de datos.
- Tiene herramientas como Dask, numpy, pandas y Numba para analizar datos.

- Permite la visualización de datos usando Bokeh, Datashader, Holoviews o Matplotlib.

- Se relaciona con varias aplicaciones de aprendizaje automático y los modelos de aprendizaje.

- Anaconda Navigator es una interfaz gráfica de usuario GUI bastante simple, pero tiene un gran potencial. Puede usar Python para administrar paquetes relacionados con la ciencia de datos desde la terminal de una manera avanzada.
- Proporciona la capacidad de acceder a recursos de aprendizaje más avanzados.
- Elimina las dependencias de paquetes y los problemas de control de versiones.
- Está equipado con herramientas que le permiten crear y compartir documentos que contienen código con compilación en tiempo real, ecuaciones, descripciones y comentarios.
- Permite compilar Python en código de máquina para una ejecución rápida.

- Ayuda a escribir algoritmos paralelos complejos para realizar tareas.
- Es compatible con la informática de alto rendimiento.

2.6.1.1.2 Instalación de anaconda por pasos

Descargar el instalador de Anaconda en la página oficial www.anaconda.com.

Verificar que no se tenga nada instalado como un mencionada o algún otro programa relacionado

Leer los términos de la licencia y dale en siguiente.

Seleccionar una instalación dependiendo de las preferencias

Seleccionr una carpeta de destino para instalar Anaconda

Seleccionar todas las características al gusto del programador y dar en las siguientes, el mismo instala complementos que se necesitan en caso de que no luego se realizan por comando.

2.6.1.2 Python

Python es un lenguaje de programación interpretado cuya filosofía hace referencia a la legibilidad de su código y trabaja con múltiples plataformas para hacer más fácil el manejo a usuarios.

Python es un lenguaje multiparadigma, ya que soporta programación orientada a objetos, programación imperativa, programación funcional.

2.6.1.2.1 Características de Python

- Python es un lenguaje de programación de múltiples paradigmas. Esto significa que, en lugar de obligar a los programadores a adoptar un estilo de

programación específico, permite múltiples estilos: programación orientada a objetos, programación imperativa y programación funcional.

- Apoya otros paradigmas utilizando extensiones.
- Python utiliza escritura dinámica y recuento de referencias para la gestión de la memoria.

2.7 Estado del arte

En los últimos años se han llevado a cabo diferentes investigaciones acerca de lo que se denomina detección facial. Siendo una de las décadas con mayor acercamiento a la tecnología la cual ha facilitado la implementación en algunos países sistemas que relacionan el rostro con bases de datos para así llevar a cabo el seguimiento de las personas (Cañego, 2017).

Partiendo de esta idea nacen diferentes campos de lo que se denomina la detección del análisis morfológico facial y aplicado a reconocimientos de estados de ánimo.

En el 2019 se llevó a cabo la creación de un sistema que indicaba qué emociones presentaba una persona utilizando el algoritmo de Viola -Jones para el reconocimiento de patrones.

Este sistema implementa el reconocimiento anímico de personas a partir de sus expresiones faciales en el software de LabVIEW y Matlab y a partir de ello se llevó a cabo en varias fases. En la primera fase el software detectó cualquier rostro en un video, en la segunda fase se procesó y adaptó al estado de ánimo a partir de este proceso se implementó la gráfica final que mostraba el estado de ánimo (Tfg et al., 2019).

En el 2016 se implementó un sistema para la detección de rostros a partir de sus cualidades faciales como el aspecto de sus ojos, nariz y boca en un sistema de Android. Teniendo un porcentaje de aprobación de bastante acogimiento ya que el uso de detectores faciales es muy común hoy en día (Zapatero Olmedillo, 2016).

En 2019 se dio un diseño que cuenta con un sistema que es capaz de reconocer el rostro de

una persona y mostrar el porcentaje de acierto que tiene con respecto a las imágenes del rostro del usuario que sirven para que el sistema pueda identificar a la persona. Utilizando la librería OpenCV de Python mediante el algoritmo de Viola-Jones y el algoritmo de Patrones Binarios Locales (LBP), se realizó un interfaz que es capaz de identificar a tantos usuarios se encuentren registrados en la base de datos (Hirvin Gonzalez, 2019).

En 2018, en la Universidad Politécnica Salesiana del Ecuador, desarrollaron un sistema de acceso usando tecnología RFID y reconocimiento facial. Cuenta con un directorio donde se encuentran alrededor de 310 fotografías almacenadas y así poder realizar la comparación para luego arrojar el resultado de la verificación (Büyükçolpan & Tol, 2019)

En los últimos años, las aplicaciones de detección de rostros que utilizan el procesamiento de imágenes digitales y el reconocimiento de patrones se han desarrollado rápidamente. Una de las razones de este crecimiento es la creciente demanda de la tecnología en aplicaciones de seguridad y vigilancia utilizadas en diferentes campos de la detección facial. Pero en el campo de las aplicaciones para dispositivos móviles, esto también es fundamental para la detección de emociones, porque muchas de estas aplicaciones utilizan la detección de rostros en fotos o vídeos (García del Prado et al. 2020).

“Al experimentar una emoción existen diferentes medios a través de los cuales podemos comunicar y expresar aquello que estamos sintiendo. Estos medios incluyen componentes fisiológicos y conductuales, como lo son los cambios faciales, gestuales y de lenguaje que ocupamos al expresarnos. Las diferencias individuales, como la personalidad y el sexo, influyen en la forma y la intensidad de la manifestación de una emoción” (Zapatero Olmedillo, 2016).

Capítulo 3 Metodología

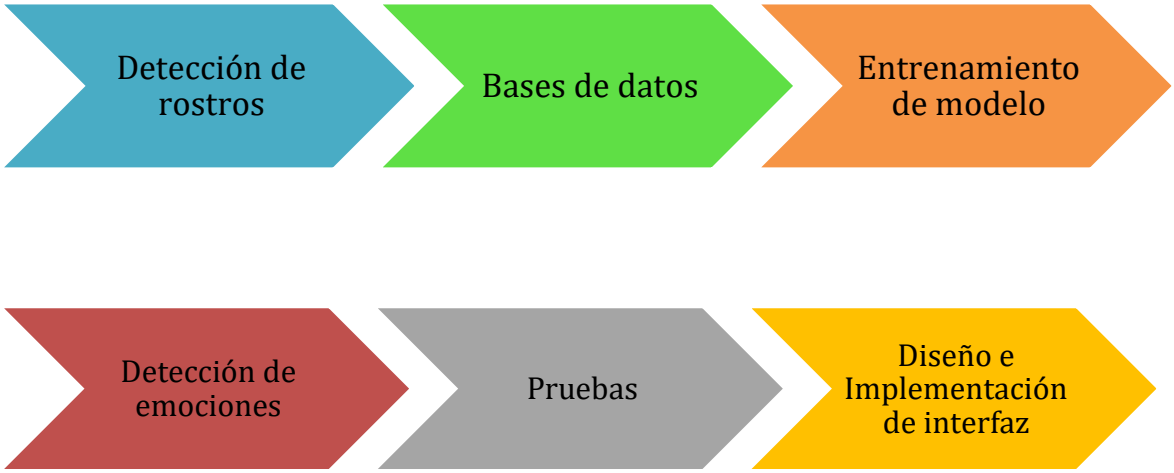


Figura 3.1 Metodología

Fuente(autor)

Para la detección de un rostro es necesario crear un algoritmo que a partir de una función ya establecida facilite la localización de una cara en una imagen omitiendo distintos objetos en un entorno, y a partir de la detección de rostros generar un algoritmo para crear una base de datos con las emociones más representativas que contiene un rostro.

Para una optimización en el tiempo se propone llevar a cabo el entrenamiento a partir de métodos ya establecidos para general el modelo. en este caso se establece que parámetros generan mejores resultados y que modelo daría mejor respuesta con menor tiempo de entrenamiento y que características maneja para proceder a lo que sería la detección de emociones teniendo en cuenta el tiempo de respuesta en tiempo real sin necesidad de cambiar mucho en los entornos del ambiente.

Evaluar las condiciones de la detección con distintos parámetros ya establecidos para observar cual da mejor respuesta teniendo en cuenta la mayor cantidad de emociones

posibles.

Posteriormente a obtener la detección de emociones llevar a cabo pruebas con las personas que se entrenó el modelo dependiendo de la base de datos y así obtener los valores de aceptabilidad que permiten evaluar qué tan eficiente es.

Finalmente diseñar una interfaz de fácil interacción que contenga la parte de detección en tiempo real y que permita visualizar a la persona cuál es su estado de ánimo presente.

Capítulo 4 Diseño e implementación

En este capítulo se van a resaltar parte de los requerimientos utilizados y cómo se desarrollan algunas técnicas de diseño e implementación de algoritmos.

4.1 Capturando Rostros

La detección de rostros es una técnica que permite encontrar en una imagen, el rostro o cara de una o varias personas, mientras que ignora el fondo de la imagen u otros objetos que estén presentes dentro de ella, pero en muchos casos pueden aparecer falsos positivos.

Está detección de rostros está presente en muchas aplicaciones que se usan durante el día, por ejemplo, en facebook con las imágenes que se postean que ya detectan el rostro y lo que muchas veces solo pide adicionar el nombre de las personas que allí aparecen. También en algunos filtros de Instagram en donde se necesita detectar el rostro para que los filtros que se requiera aplicar sean empleados de manera más compleja.

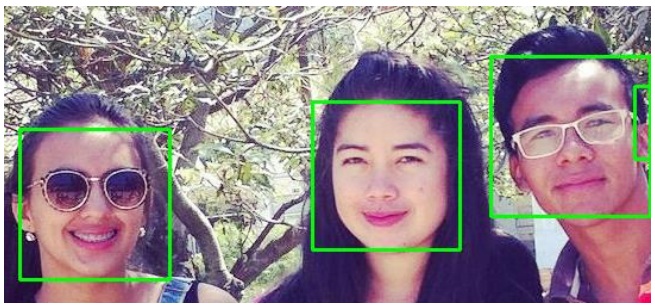


Figura 4.1 Ejemplo detección de rostros

Fuente (Autor)

4.1.1 Detección de rostro en una imagen

Esta no es una técnica fácil para el computador, por ello se necesita que este 'aprenda' para lo cual se implementa machine Lear Ning o aprendizaje automático.

Para implementar estas técnicas se necesita de una gran cantidad de imágenes para entrenar a un clasificador, estos clasificadores. El clasificador se implementa para que este pueda encontrar las diferencias entre la presencia de un objeto y la no presencia del mismo. Por ejemplo, para realizar un detector de rostros se necesitan imágenes positivas e imágenes negativas las cuales tienen y no tienen rostros. Luego se procede a la extracción de características de todas las imágenes, para después utilizar un enfoque de machine learning y proceder con el entrenamiento. Finalmente se podrá obtener el clasificador que ya indica los rostros detectados.

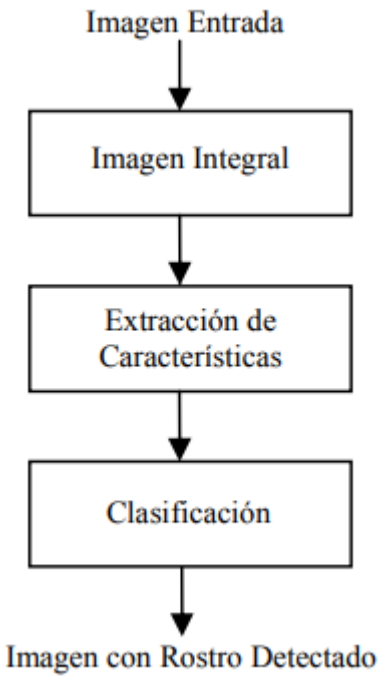


Figura 4.2 Proceso de detección de rostros en una imagen

Fuente:(*Detección de Rostros Basados En Filtros Haar + Adaboost*, n.d.)

4.1.2 Imagen integral

Es la representación que permite extraer de forma rápida características a diferentes tamaños ya que no se trabaja directamente con los datos de intensidad de la imagen si no con una imagen acumulativa que se implementa a partir de operaciones, todo esto se puede observar más detallado en el capítulo 2 detección de rostros.

4.1.3 Extracción de características

Primeramente, el algoritmo requiere muchas imágenes positivas (imágenes donde se observe una cara) e imágenes negativas (imágenes donde no se encuentre una cara) para entrenar un clasificador. Se necesita extraer características de él. Para esto, se utilizan las características de haar que se observan en la imagen siguiente.

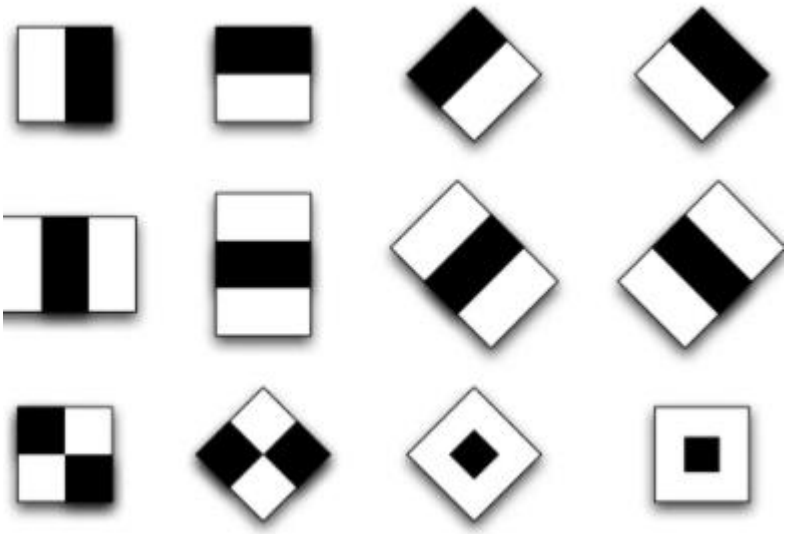


Figura 4.3 Características Haar

Fuente: (Cañego, 2017)

Son como un núcleo convolucional para dar mejor resultado. Cada característica es un único valor que se obtiene al restar la suma de píxeles en el rectángulo blanco de la suma de píxeles en el rectángulo de color negro.

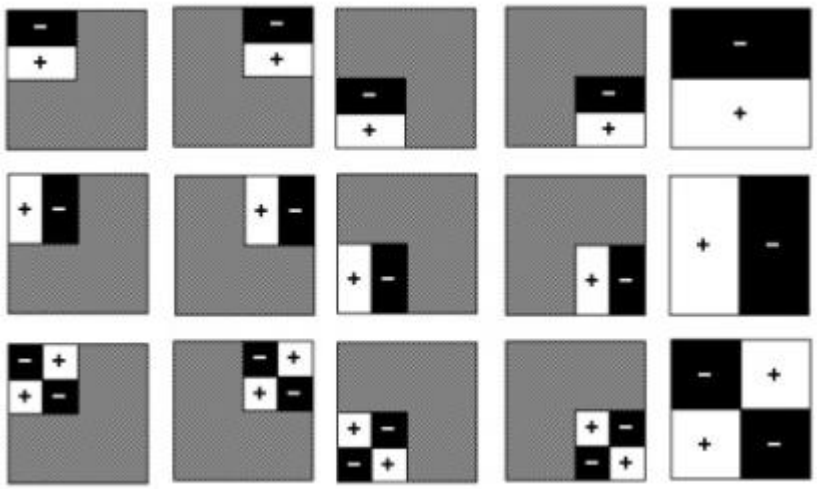


Figura 4.4 Filtros Haar rotados, trasladados y con cambios de escala

Fuente: (Gordillo et al., 2015)

En el estudio planteado por Viola-Jones, las características se definen sobre una ventana de búsqueda básica de un marco de 24x24 píxeles, lo que da opción a más de 160000 características posibles. Todos los tamaños y ubicaciones posibles de cada núcleo se usan para calcular las características, para cada cálculo de características, se necesitan encontrar la suma de píxeles en rectángulos blanco y negro

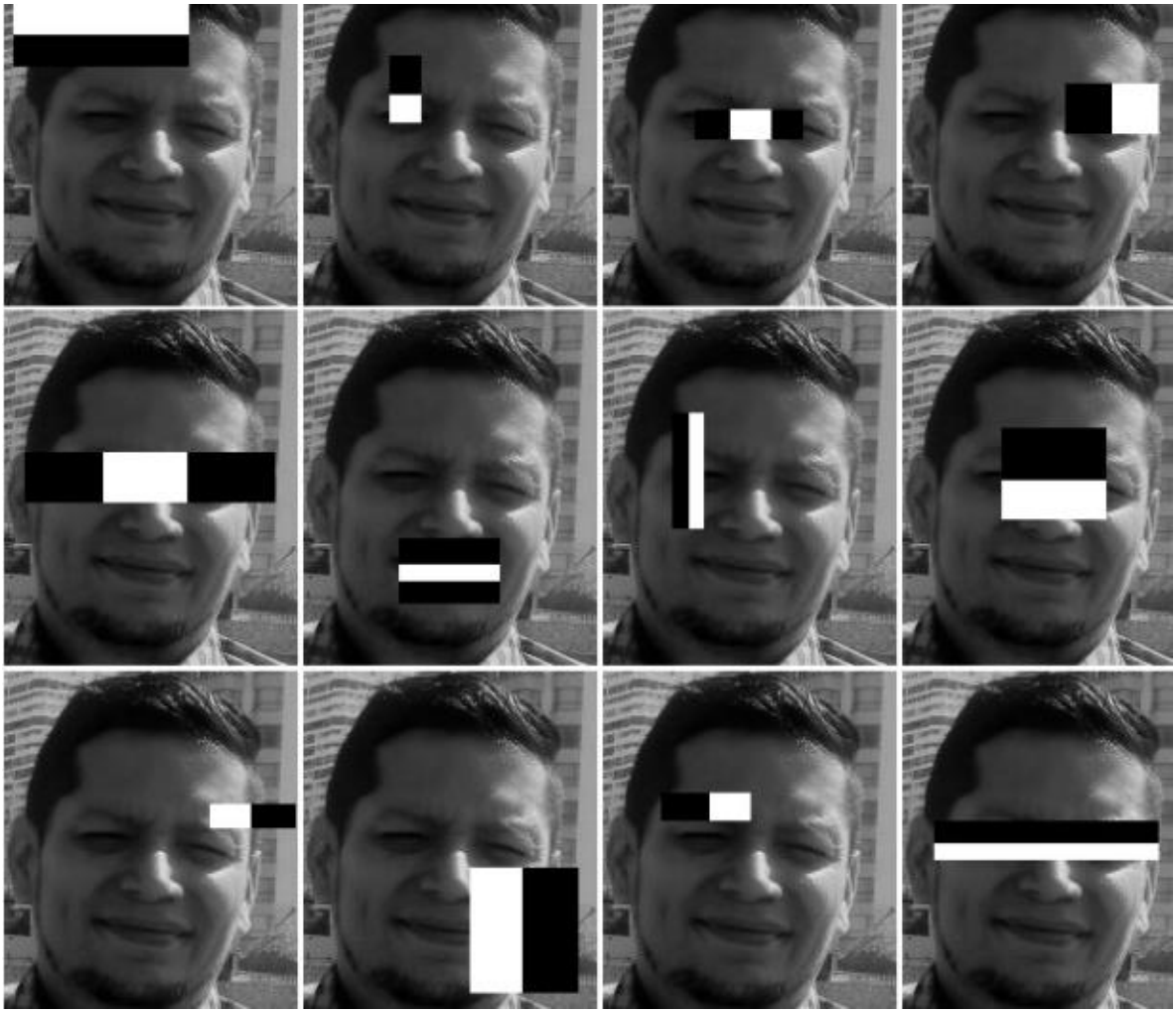


Figura 4.5 Filtros Haar escalados en diferentes posiciones de la imagen

Fuente: (Cañego, 2017)

4.1.4 Clasificación

En una imagen digital, la mayoría de la parte de la región de una imagen es una región sin cara. Entonces, es mejor idea tener un diseño simple para verificar si una ventana no es una región del rostro.

Por esto se introdujo el concepto de cascada de clasificadores boosting. El boosting se genera para poder tomar una serie de clasificadores débiles y combinarlos para construir un clasificador fuerte con la precisión de detección. Es requerido realizar un proceso de entrenamiento supervisado para implementar la cascada de clasificadores. Este diseño se realiza mediante un algoritmo basado en AdaBoost, un meta-algoritmo adaptativo de aprendizaje automático.

Adaboost es un conjunto de procedimientos efectivos para buscar un número pequeño de buenas características que no tienen un cambio significante. En una subventana de una imagen, el número total de Haar-like features es muy grande, mucho más grande que el número de pıxeles. Con tal de asegurar una clasificación rápida, el proceso de aprendizaje debe excluir a la gran mayoría de las características disponibles, y centrarse en un conjunto pequeño de características críticas (Gámez Jiménez, 2009).

4.1.5 Código detección de caras

Para emplear la detección de rostros con haarcascade en OpenCV vamos a necesitar del módulo detectMultiScale que ayudará a detectar los objetos de acuerdo al clasificador que se utilice. Esto nos permitirá obtener un rectángulo delimitador en donde se encuentre el objeto que se desea encontrar dentro de una imagen, para ello se deben especificar algunos argumentos.

Imagen: Es la imagen en donde va a actuar el detector de rostros.

ScaleFactor: Este parámetro nos especifica que tanto va a ser reducido el tamaño de la imagen. Por ejemplo, si se ingresa 1.1, quiere decir que se va a ir reduciendo la imagen en 10%, con 1.3 se reduce el 30%, creando de esta manera una pirámide de imágenes. Se debe tener en cuenta que si se da un número muy alto se pierden algunas detecciones. Mientras que para valores algo pequeños como por ejemplo 1,01 se llevará mayor tiempo de procesamiento, ya que se encontrarán más imágenes a analizar, además de que pueden incrementar además incrementan los falsos positivos

El parámetro Min Neighbors especifica cuántos vecinos debe tener cada rectángulo

Una explicación es cuando se tiene una pequeña ventana que va a ir pasando por una imagen digital buscando caras y en este caso se puede encontrar que al final de todo el proceso ha identificado varios rostros en la misma área, pero puede que muchos de ellos correspondan a la misma persona. Entonces este parámetro, hace relación a todos esos rectángulos delimitadores de un mismo rostro. Por lo cual min Neighbors especifica el número mínimo de cuadros de vecinos, que debe tener un rostro para que quede detectado como tal.

Se debe tener en cuenta que entre más alto sea el valor que se coloque, menos cantidad de caras se detectarán, mientras que si se da un valor muy bajo se pueden presentar falsos positivos y encontrar regiones que no pertenecen a una cara.

Primero se importan las librerías en las cuales se tienen os, cv2 y inútils

```
import imutils
import cv2
import os
```

Figura 4.6 Módulos

Fuente (Autor)

OpenCV nos ofrece clasificadores pre entrenados no solo de rostros de personas si no de características como objetos específicos.

El módulo proporciona un contenedor para módulos específicos de la plataforma, tales como posix, nt y mac. La interfaz de programación para las funciones disponibles en todas las plataformas debe ser la misma, por lo que usar el módulo nos ofrece cierta medida de portabilidad. Sin embargo, no todas las funciones están disponibles en todas las plataformas.

Imutils es un conjunto de funciones que ha sido creado para realizar tareas básicas de procesamiento de imágenes de una manera más fácil.

Lo primero es crear las carpetas con las imágenes de cada emoción dentro de la carpeta Data.

```
dataPath = 'E:\python3\Data'
emotionsPath = dataPath + '/' + emotionName

if not os.path.exists(emotionsPath):
    print('Carpeta creada: ',emotionsPath)
    os.makedirs(emotionsPath)
```

Figura 4.7 Creación de carpetas

Fuente (Autor)

Se especifica que se va a realizar un video en tiempo real y se asigna el detector de rostros que se va a usar y finalmente se establece un contador count en 0 para que se cuenten todos los rostros que se pueden ir almacenando.

```
cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)

faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
count = 0
```

Figura 4.8 Detector de rostros

Fuente (Autor)

En la siguiente imagen se observa cómo se lee cada fotograma y este a su vez será redimensionado, se trasforma a escala de grises y para tener una imagen del fotograma intacto se implementa frame.copy() y se asigna a auxFrame.

```
while True:

    ret, frame = cap.read()
    if ret == False: break
    frame =  imutils.resize(frame, width=640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    auxFrame = frame.copy()
```

Figura 4.9 Lectura de cada fotograma

Fuente (Autor)

En la variable faces se guardan todos los rostros que se pueden detectar y para mirar las propiedades de cada uno de ellos se pasa al for, en él se dibuja un rectángulo, luego se recorta cada rostro, se redimensiona, para que todas las caras guardadas tengan el tamaño mismo y se guarda cada rostro en la carpeta que se creó. Finalmente se incrementa en 1 el contador y se visualiza.

```
faces = faceClassif.detectMultiScale(gray,1.3,5)

for (x,y,w,h) in faces:
    cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
    rostro = auxFrame[y:y+h,x:x+w]
    rostro = cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
    cv2.imwrite(emotionsPath + '/rotro_{}.jpg'.format(count),rostro)
    count = count + 1
cv2.imshow('frame',frame)
```

Figura 4.10 Almacenamiento de rostros

Fuente(Autor)

Se establece una tecla para romper el proceso

```
k =  cv2.waitKey(1)
if k == 27 or count >= 500:
    break
```

Figura 4.11 Establecimiento de tecla para finalizar

Fuente (Autor)

Se indica que ha finalizado el video en tiempo real y se cierran las ventanas.

```
cap.release()
cv2.destroyAllWindows()
```

Figura 4.12 Finalización de primer código

Fuente (Autor)

4.2Entrenamiento

4.2.1 Código entrenamiento de modelo LBPH

El entrenamiento se lleva a cabo diferenciando modelos para obtener mejores resultados en este caso el que mejor obtiene resultados es el método LBPH el cual genera un entrenamiento rápido dependiendo de la cantidad de imágenes en la base de datos.

```
7   def obtenerModelo(method,facesData,labels):
8
9       if method == 'LBPH': emotion_recognizer = cv2.face.LBPHFaceRecognizer_create()
10      # Entrenando el reconocedor de rostros
11      print("Entrenando ( "+method+" )...")
12      inicio = time.time()
13      emotion_recognizer.train(facesData, np.array(labels))
14      tiempoEntrenamiento = time.time()-inicio
15      print("Tiempo de entrenamiento ( "+method+" ): ", tiempoEntrenamiento)
16
17      # Almacenando el modelo obtenido
18      emotion_recognizer.write("modelo"+method+".xml")
```

Figura 4.13 Creación de función

Fuente (Autor)

Se crea la función obtener Modelo, para esta función se necesita de method que es el nombre del método a emplear y observar cual es más eficiente, faces Data el array en donde se almacenarán todas las caras con sus diferentes emociones y finalmente labels, que son las etiquetas de cada uno de los rostros en este caso se van observado que resultados se obtiene y cuantas emociones se podrían guardar.

```
def obtenerModelo(method,facesData,labels):
```

Figura 4.14 Función

Fuente(autor)

Se usa el método LBPH y se asignará cv2.face.LBPHFaceRecognizer_create() a emotion_recognizer.

```
if method == 'LBPH': emotion_recognizer = cv2.face.LBPHFaceRecognizer_create()
```

Figura 4.15 Lectura de modelo

Fuente(Autor)

De la línea 11 a 15 se entrena al reconocedor, para ello en la línea 11 imprime un mensaje de 'Entrenando'. En la línea 12 tiempo inicial en inicio, en la línea 13 se entrena el reconocedor, para ello se coloca emotion recognizer.train y dentro de los paréntesis se especifica el array en donde se guardan todas las caras y las etiquetas de cada una de estas caras. En la línea 14 se toma el tiempo actual menos el tiempo de inicio para conocer la cantidad de tiempo en que se llevó el entrenamiento. En la línea 15 se imprime el método usado y el tiempo que le llevó entrenar.

```
11      print("Entrenando ( "+method+" )...")
12      inicio = time.time()
13      emotion_recognizer.train(facesData, np.array(labels))
14      tiempoEntrenamiento = time.time()-inicio
15      print("Tiempo de entrenamiento ( "+method+" ): ", tiempoEntrenamiento)
```

Figura 4.16 Entrenamiento

Fuente (Autor)

Después de que se entrena el reconocedor, se procede al almacenamiento con emotion_recognizer.write,

```
emotion_recognizer.write("modelo"+method+".xml")
```

Figura 4.17 almacenamiento de modelo

Fuente(Autor)

Se indica la ruta en donde se encuentra.

```
dataPath = 'E:\python4\Data'
emotionsList = os.listdir(dataPath)
print('Lista de personas: ', emotionsList)
```

Figura 4.18 Ruta

Fuente (Autor)

Se declaran los arrays vacíos labels y faceData en ellos se almacenará todos los rostros de las carpetas con una etiqueta asociada según la emoción observando las características. Se declara label en 0, para ayuda con el valor almacenado en labels.

```
labels = []
facesData = []
label = 0
```

Figura 4.19 Etiquetas

Fuente (Autor)

Con un for recorremos cada elemento del array emotionsList. Luego se asigna a emotionsPath la ruta completa.

```
for nameDir in emotionsList:
    emotionsPath = dataPath + '/' + nameDir
```

Figura 4.20 Recorrido de cada elemento

Fuente(Autor)

Se recorre cada una de las imágenes contenidas en emotions Path, es decir que por cada carpeta de emociones se leerán todas las caras contenidas en ella. En la línea 33 se almacena el valor de label en el array labels, y después se lee cada imagen y se transforma a grises, gracias a que se está usando 0 como segundo argumento de cv2.imread. Una vez que se haya leído cada cara de la carpeta se aumenta en 1 label para que la próxima carpeta con otra emoción tenga otro valor.

```
31      for fileName in os.listdir(emotionsPath):
32          #print('Rostros: ', nameDir + '/' + fileName)
33          labels.append(label)
34          facesData.append(cv2.imread(emotionsPath+'/'+fileName,0))
35          #image = cv2.imread(emotionsPath+'/'+fileName,0)
36          #cv2.imshow('image',image)
37          #cv2.waitKey(10)
38      label = label + 1
```

Figura 4.21 Recorrido de cada una de las imágenes

Fuente(Autor)

Se llama la función obtener Modelo que se creó en un principio, en ella se especifica el método que se usará para entrenar el reconocedor, seguido del array en donde se han almacenado los rostros y finalmente cada una de las etiquetas correspondientes.

```
obtenerModelo('LBPH',facesData,labels)
```

Figura 4.22 Modelo generado

Fuente(Autor)

4.3 Reconocimiento emoción

4.3.1 Código reconocimiento de rostros

Se importa OpenCV que es utilizado para la parte de visión artificial, os y numpy con un alias np.

Se crea una función y como parámetros se tendrá a emotion que puede ser la emoción descrita. Dependiendo de estos se puede visualizar una imagen que represente el estado diferente a la derecha de los fotogramas

```
if emotion == 'Tristeza': image = cv2.imread(

def emotionImage(emotion):
```

Figura 4.23 Función Emoción

Fuente (Autor)

Dependiendo del método se asignará a emotion recognizer cv2.face.LBPHFaceRecognizer_create().y se lee el modelo correspondiente a LBPH previamente entrenado en los anteriores procesos.

```
method = 'LBPH'

if method == 'LBPH': emotion_recognizer = cv2.face.LBPHFaceRecognizer_create()
```

Figura 4.24 Lectura de modelo

Fuente (Autor)

Se especifica la ruta del directorio data.

```
dataPath = 'E:\python3\Data'
imagePaths = os.listdir(dataPath)
print('imagePaths=',imagePaths)
```

Figura 4.25 Ruta de directorio

Fuente (Autor)

Se indica que vamos a realizar un video en tiempo real.

```
cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)
```

Figura 2.26 video en tiempo real

Fuente (Autor)

Se lee el detector de rostros.

```
faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
```

Figura 4.27 lectura del detector de rostros

Fuente(autor)

Se lee cada fotograma, y se ayuda de gray.copy() para crear una copia.

```
ret,frame = cap.read()
if ret == False: break
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
auxFrame = gray.copy()
```

Figura 4.28 Lectura de cada fotograma

Fuente (Autor)

Se concatenan dos imágenes, la primera frame y la segunda será un área de color negro de 300 pixeles de ancho y 480 de alto que se creó con np.zeros

```
nFrame = cv2.hconcat([frame, np.zeros((480,300,3),dtype=np.uint8)])
```

Figura 4.29 Concatenamiento

Fuente (Autor)

En faces se almacenan todas las caras detectadas.

```
faces = faceClassif.detectMultiScale(gray,1.3,5)
```

Figura 4.30 almacenamiento de caras

Fuente (Autor)

Luego se extrae la información donde se almacenaron las caras en donde se encuentran las coordenadas x, y, ancho y alto de cada rostro detectado. Se recortan los rostros de la imagen, se redimensiona y finalmente se va a utilizar emotion_recognizer.predict entre paréntesis el rostro recortado y redimensionado.

En la última línea se visualiza el valor obtenido de result sobre la cara detectada.

```
for (x,y,w,h) in faces:
    rostro = auxFrame[y:y+h,x:x+w]
    rostro = cv2.resize(rostro,(150,150),interpolation= cv2.INTER_CUBIC)
    result = emotion_recognizer.predict(rostro)

    cv2.putText(frame,'{}'.format(result),(x,y-5),1,1.3,(255,255,0),1,cv2.LINE_AA)
```

Figura 4.31 extracción de características

Fuente (Autor)

Se compara el segundo valor almacenado en result con 60. Si este valor es menor a 60, quiere decir que se ha reconocido una emoción, por lo cual se visualiza el nombre de la emoción reconocida, además de un rectángulo. Mientras tanto en la línea 51 se llama a la función, y a ella se le entrega imagePaths[resul que contiene el nombre de la emoción reconocida y se concatena frame e image que es la imagen que contiene otra imagen a la que se le haría referencia el estado encontrado.

Cuando result es mucho mayor a 60, no se reconoció emociones por lo tanto se especifica que se muestre el texto 'No identificado' y se dibujará un rectángulo.

```
46
47          if method == 'LBPH':
48              if result[1] < 60:
49                  cv2.putText(frame,'{}'.format(imagePaths[result[0]]),(x,y-25),2,1.1,(0,255,0),1,cv2.LIN
50                  cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
51                  image = emotionImage(imagePaths[result[0]])
52                  nFrame = cv2.hconcat([frame,image])
53              else:
54                  cv2.putText(frame,'No identificado',(x,y-20),2,0.8,(0,0,255),1,cv2.LINE_AA)
55                  cv2.rectangle(frame, (x,y),(x+w,y+h),(0,0,255),2)
56                  nFrame = cv2.hconcat([frame,np.zeros((480,300,3),dtype=np.uint8)])
```

Figura 3.32 Cumplimiento de modelo

Fuente (Autor)

Se visualiza nFrame , se especifica que cuando se presione una tecla el proceso se detenga

```
    cv2.imshow('nFrame',nFrame)
    k = cv2.waitKey(1)
    if k == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Figura 4.33 visualización y cierre

Fuente (Autor)

4.4 desarrollo de interfaz

4.4.1 código interfaz

Ahora se va a desarrollar lo que sería la interfaz utilizando Tkinter que es un binding de la biblioteca gráfica Tcl/Tk la cual Proporciona un conjunto de herramientas robustas e independiente de la plataforma para administrar ventanas de manera más sencilla. A continuación, se anexa una parte en donde se tiene en cuenta el título los escudos de la universidad y parámetros de tamaño de visualización dependiendo del contenido.

```
from tkinter import *
from tkinter import messagebox
from tkinter import filedialog
from PIL import Image, ImageTk as itk

class Ventana():

    def __init__(self):

        self.ventana = Tk()
        self.ventana.title("Reconocimiento de Emociones")
        self.ventana.configure(background='white')
        self.ventana.geometry("1000x700")

        TituloTesis = Label(self.ventana, text="DESARROLLO DE SISTEMA PARA LA DETECCION DE EMOCIONES UTILIZANDO", background='white
        TituloTesis.place(x= 220, y = 40)
        TituloTesis = Label(self.ventana, text="VISION E INTELIGENCIA ARTIFICIAL APLICADO A ENTORNOS EDUCATIVOS", background='white
        TituloTesis.place(x= 220, y = 70)
        TituloFacultad = Label(self.ventana, text="INGENIERIA EN TELECOMUNICACIONES - DEPARTAMENTO EEST", background='white', ancho
        TituloFacultad.place(x= 270, y = 120)
        TituloFacultad = Label(self.ventana, text="FACULTAD DE INGENIERIAS Y ARQUITECTURA", background='white', anchor="center", fo
        TituloFacultad.place(x= 340, y = 150)

        btn_cargar = Button(self.ventana,text='Subir Imagen', command=self.cargar_imagen, relief=RAISED)
        btn_cargar.place(x = 130, y = 280)
        btn_comparar = Button(self.ventana,text='Comparar Imagen', command=self.comparar_imagen, relief=RAISED)
        btn_comparar.place(x = 700, y = 280)
        btn_salir = Button(self.ventana, text = " Salir", command = self.salir, relief=RAISED)
        btn_salir.place(x = 500, y = 670)
```

Figura 4.34 Código interfaz

Fuente:(Autor)

Capítulo 5 Resultados

5.1 detección de rostros

Inicialmente se logra la detección de rostros en una imagen utilizando lo que es Haarcascade el cual se observa en el capítulo anterior, implementación. Para así observar qué cualidades tiene y cuántas caras puede detectar.

Se observa que se detectan 32 caras de 35 personas presentes en la imagen, obteniendo buenos resultados.

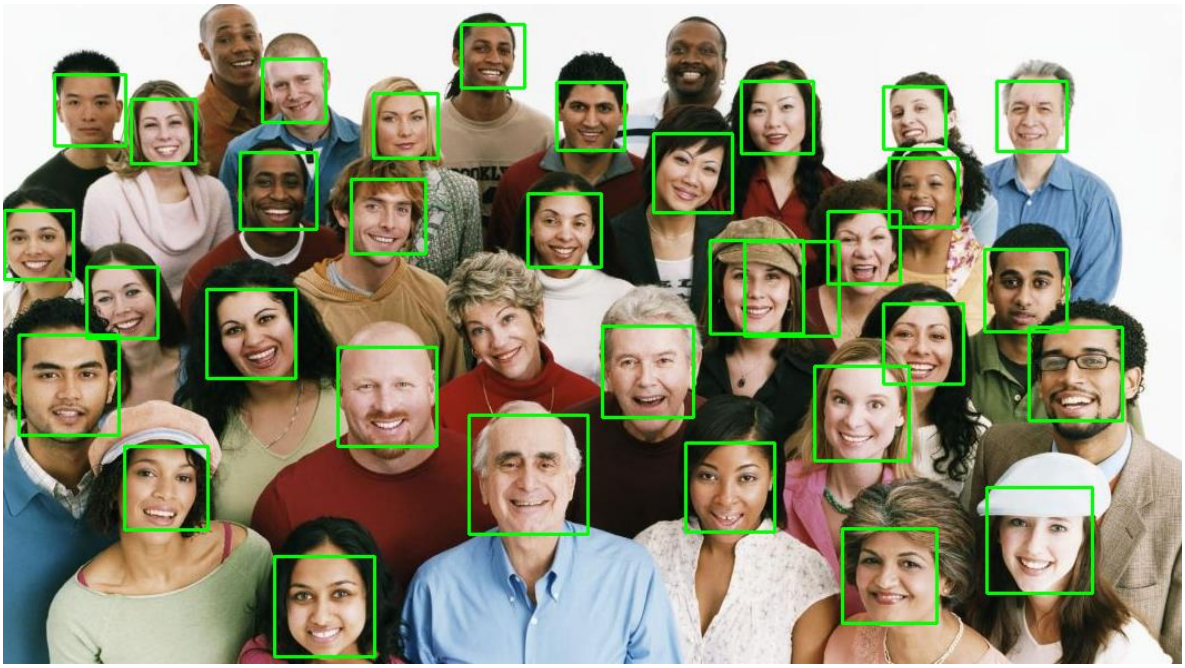


Figura 5.1 Detecciones de rostros

Fuente (Hirvin Gonzalez, 2019)

Se observa bastante aprobación en cuanto a cambios de entorno ya que hay detectores de rostros como los detectores HOG que son sensibles a cambios de ángulos o de luz.

Figura 5.2 Detecciones de rostros con cambios en entorno

Fuente (Autor)

De igual manera se mantiene dando buenos resultados en cuanto a cambios en el rostro como lo es una persona que use gafas.

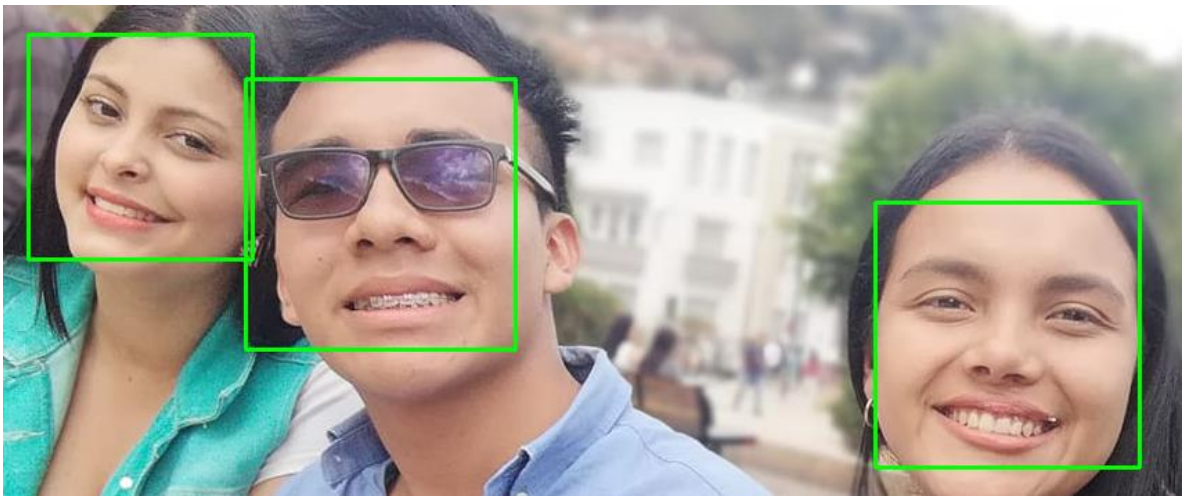


Figura 5.3 Detecciones de rostros con cambios

Fuente (Autor)

5.2 bases de datos para entrenamiento de algoritmos

Se implementó una base de datos teniendo en cuenta cada una de las emociones y haciendo de manera más notoria lo que es una representación de las características del rostro para que se asemeje a cada emoción clara para el entrenamiento.

Figura 5.4 Base de datos con cada una de las emociones

Fuente (Autor)

Figura 5.5 Base de datos emoción alegría.

Fuente (Autor)

Figura 5.6 Base de datos emoción ira.

Fuente(Autor)

Figura 5.7 Base de datos emoción miedo.

Fuente(Autor)

Figura 5.8 Base de datos emoción neutro.

Fuente (Autor)

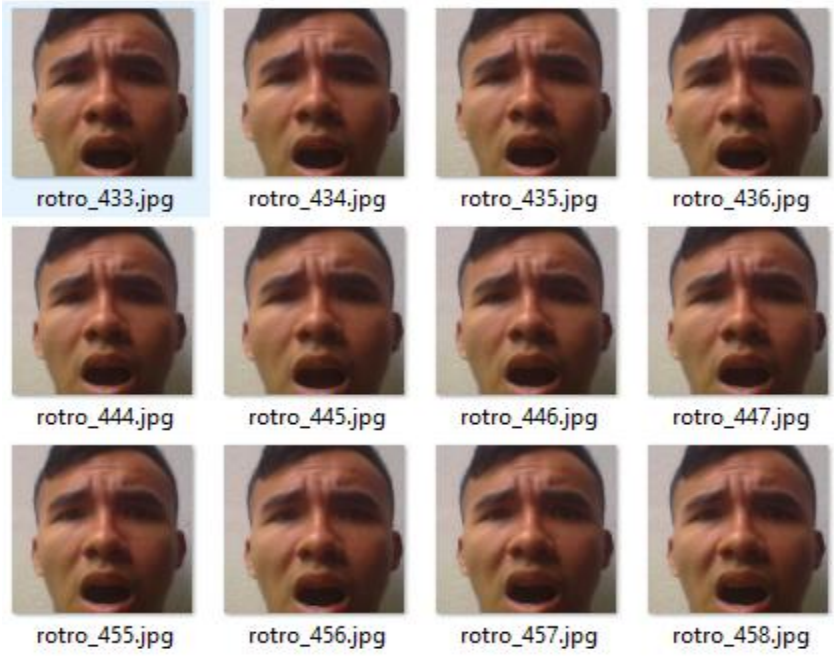


Figura 5.9 Base de datos emoción sorpresa.

Fuente (Autor)

Figura 5.10 Base de datos emoción tristeza

Fuente (Autor)

Figura 5.11 Base de datos emoción asco

Fuente (Autor)

5.3 Detección de estados de ánimo

En las imágenes siguientes se observa el resultado de las pruebas del algoritmo de detección de emociones. En la parte superior encontramos lo que sería la detección de la emoción correspondiente a cada uno de los rostros y va variando de acuerdo a la etiqueta que se le dio a cada emoción. Por ejemplo, en la primera imagen se observa que tiene la etiqueta 0 y un valor de confianza de 36, este valor de confianza entre más se acerque a 0 indica que tiene mayor similitud con los datos que tiene el modelo de acuerdo al entrenamiento que se llevó a cabo con la base de datos.

Se observa que el algoritmo tiene bastante fluidez en cuanto a la detección de la emoción sin importar cambios abruptos de una emoción a otra. Pero en términos de cambios de luz influye bastante ya que si se tiene una mejor cámara y se obtiene un mejor modelo a partir de imágenes de más calidad se tendrán mejores resultados.

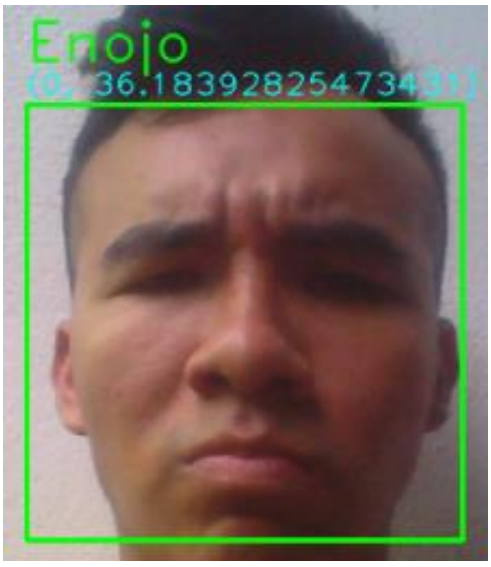


Figura 5.12 Detección de emoción enojo

Fuente (Autor)

Figura 5.13 Detección de emoción alegría

Fuente (Autor)

Figura 5.14 Detección de emoción tristeza

Fuente (Autor)

Figura 5.15 Detención de emoción sorpresa

Fuente (Autor)

Figura 5.17 Detención de emoción Neutro

Fuente (Autor)

5.4 Interfaz

Final mente se implementó una interfaz en la que se cuenta con 3 ventanas. En la primera ventana se encuentra lo que es el inicio del sistema y en el se observa el escudo de la universidad y el título del proyecto.

4.4.2 Manual de uso

Inicialmente la interfaz cuenta con tres ventanas principales en la cuales se puede desplazar para mayor facilidad y entendimiento entre la maquina y la persona que la utilice.

Partiendo de una ventana principal donde se encuentra la información correspondiente del usuario creador y de la universidad a la que pertenece.

Como primeras medias el sistema no se puede sobrecargar con demasiadas informaciones de la base de datos, o dependiendo del equipo en el que se esté trabajando va a tener mejores resultados en el procesamiento de la información para obtener un modelo.

el modelo creado tiene que estar especificado en la misma ruta donde se encuentra el .py y también donde se encuentra la base de datos.

se pueden modificar las rutas dentro del .py para así facilitar la ubicación de las sub carpetas.

cuando se crea la base de datos hay que tener en cuenta el entorno donde se está trabajando, ya que demasiada luz afecta la detección de la emoción.

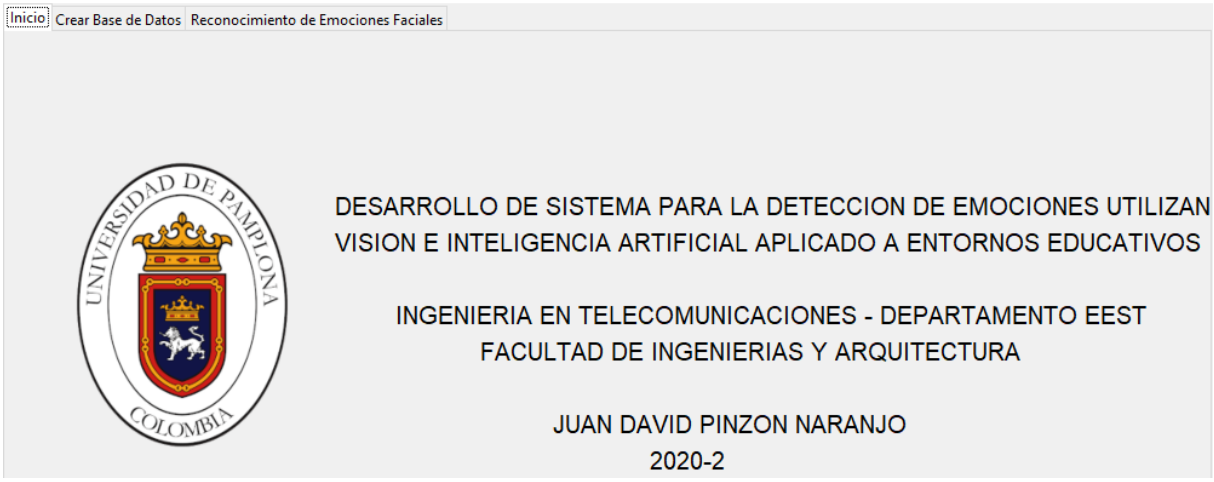


Figura 5.16 Inicio

Fuente (Autor)

En la segunda ventana se ha creado 7 diferentes botones para indicar que emociones quiere almacenar en la base de datos. También se implementó un botón para hacer el proceso

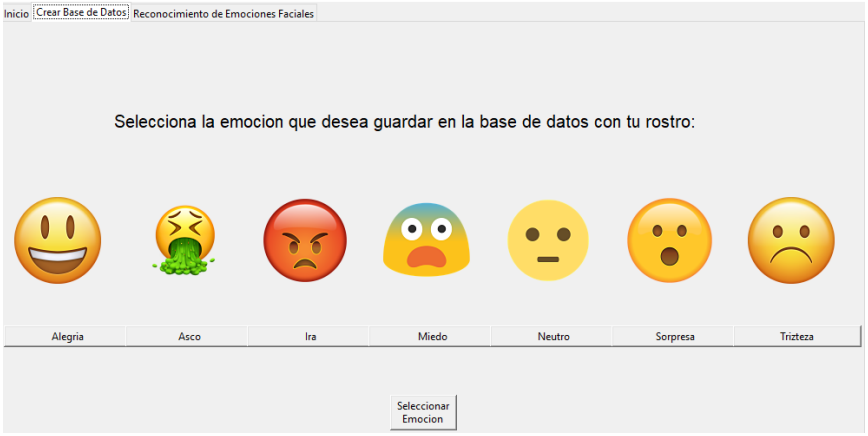


Figura 5.17 Crear base de datos

Fuente (Autor)

En la última ventana se encuentra lo que es en botón para crear nuestro modelo y el botón para generar lo que es la detección de la emoción y un botón final para salir

Figura 5.18 Reconocimiento de Emociones Faciales

Fuente (Autor)

Conclusiones

El Machine Learning (ML) como aprendizaje automático ha mejorado en un porcentaje muy alto los resultados de detección de rostros. Con esto se comprueba que la tecnología de detección de rostros y emociones está presente y actualizándose. Las diferentes técnicas de detección de rostros siempre se basan en descartar objetos presentes en una imagen o video, y así poder indicar un rostro a partir de técnicas ya implementadas, como las propuestas por viola-Jones a partir de haarcascades que dan excelentes resultados en diferentes entornos y con rostros poco comunes.

Generar una base de datos propia de cada emoción para el entrenamiento del modelo ayuda a mejorar los resultados, ya que podemos ir diferenciando cuantas imágenes se necesitan para el entrenamiento sin necesidad de desperdiciar procesamiento. A la hora de generar bases de datos de emociones es importante realizar las cualidades morfológicas que más se adecuen a cada emoción.

Se implementó de la mejor manera posible la detección de las emociones faciales en las cuales encontramos alegría, tristeza, ira, asco, sorpresa y miedo. Pero no solo se lograron identificar estas emociones en tiempo real, sino que también se logró identificar una 7 emoción la cual es neutro, esta emoción se puede observar a veces cuando se está de frente al computador, en la cual el rostro no genera muchos cambios. De esta manera se logró potenciar al máximo todo el proyecto.

Se diseñó una interfaz sencilla teniendo en cuenta todas las características en cuanto a detección de rostros, creación de bases de datos y detección de emociones para la interacción con cada estudiante que esté dispuesto a usarla. Se le dio un toque universitario teniendo en cuenta la universidad de pamplona.

Bibliografía

6 emociones básicas, latidos de un lenguaje universal - La Mente es Maravillosa. (n.d.). Retrieved October 28, 2020, from https://lamenteesmaravillosa.com/6-emociones-basicas-latidos-de-un-lenguaje-universal/

Anaconda Distribution: La Suite más completa para la Ciencia de datos con Python. (n.d.). Retrieved November 17, 2020, from https://blog.desdelinux.net/ciencia-de-datos-con-python/

Bartual González, R. (2017). *Detección facial y reconocimiento anímico mediante expresiones faciales*. 48. http://eds.a.ebscohost.com.ezproxy.umng.edu.co/eds/pdfviewer/pdfviewer?vid=2&sid=8970c06b-b7e2-423a-92d7-b6b7ad56afe9%40sessionmgr4006

Básica, E., Red, D. E. U. N. A., Tradicional, C., Neuronal, Y. C., Fundamentos, T., Redes, D. E. L. A. S., Artificiales, N., Red, T., Organizing, S., & Red, M. A. P. Y. (n.d.). *REDES NEURONALES ARTIFICIALES- Xabier Bosogain Olabe.pdf*. http://cvb.ehu.es/open_course_ware/castellano/tecnicas/redes_neuro/contenidos/pdf/libro-del-curso.pdf

Cañego, N. &. (2017). *Personas Mediante Reconocimiento Facial Aplicado a Videovigilancia ”*.

Ekman, P. (2005). Basic Emotions. In *Handbook of Cognition and Emotion* (pp. 45–60). John Wiley & Sons, Ltd. https://doi.org/10.1002/0470013494.ch3

García del Prado, N., González Castro, V., Alegre, E., & Fidalgo Fernández, E. (2020). *Comparación de métodos de detección de rostros en imágenes digitales*. 976–982. https://doi.org/10.17979/spudc.9788497497749.0976

Herrero Vez, T., & Villena Román, J. (tutor). (2010). *Sistema automático de detección y etiquetado de caras en imágenes*. 178. http://e-archivo.uc3m.es/bitstream/handle/10016/11170/PFC Tamara Herrero Vez.pdf?sequence=1

Imágenes, P. D. E. (n.d.). *3. procesamiento de imágenes*. 42–72.

Izaurieta, F., & Saavedra, C. (1999). Redes Neuronales Artificiales. *Charlas de Fisica*, 1–15. https://doi.org/10.1016/S0210-5691(05)74198-X

Las 6 emociones básicas según la psicología. (n.d.). Retrieved October 28, 2020, from https://www.psicologia-online.com/las-6-emociones-basicas-segun-la-psicologia-4205.html

LAS SEIS EMOCIONES BÁSICAS | . (n.d.). http://germansalinas.blogspot.com/2014/02/las-seis-emociones-basicas.html

Máquinas de Vector Soporte (Support Vector Machines, SVMs). (n.d.). Retrieved November 18, 2020, from https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines

Modelo HSV - EcuRed. (n.d.). https://www.ecured.cu/Modelo_HSV

Perfil, E., & Rocha, P. (n.d.). *El Perfil del Estudiante en la Educación 4.0*.

Presente y futuro del reconocimiento facial. (n.d.). Retrieved October 16, 2020, from https://www.beedigital.es/tendencias-digitales/historia-y-evolucion-del-reconocimiento-facial/

Procesamiento digital de imágenes. (1996). *Perfiles Educativos*, *72*.

Ruse, M. (2010). Charles Darwin. In G. Oppy (Ed.), *The History of Western Philosophy of Religion* (pp. 161–174). Acumen Publishing Limited.

https://doi.org/10.1017/UPO9781844654666.013

Tfg, T. D. E. L., Autor, T., Salam, E., & Data, S. J. (2019). *Deep learning para el reconocimiento facial de emociones básicas*.

Trujillano, J., March, J., & Sorribas, A. (2004). Aproximación metodológica al uso de redes neuronales artificiales para la predicción de resultados en medicina. *Medicina Clinica*, *122*(SUPPL. 1), 59–67. https://doi.org/10.1157/13057536

UNIDAD 4 REDES NEURONALES - INTELIGENCIA ARTIFICIAL. (n.d.). Retrieved October 30, 2020, from https://sites.google.com/site/mayinteligenciartificial/unidad-4-redes-neuronales

Universidad de Sevilla. (2016). *Tema 4 : Segmentación de imágenes Segmentación de imágenes*. http://alojamientos.us.es/gtocoma/pid/tema4.pdf

Zapatero Olmedillo, D. (2016). *Herramienta de reconocimiento facial de emociones en Android*. 2–5. http://oa.upm.es/44722/

Büyükçolpan, & Tol, B. panjang jalan. (2019). No 主観的健康感を中心とした在宅高齢者における 健康関連指標に関する共分散構造分析Title. ثبثبث, ث ثقثقثق(ققثق), 1–121. https://www.bps.go.id/dynamictable/2018/05/18/1337/persentase-panjang-jalan-tol-yang-beroperasi-menurut-operatornya-2014.html

Cañego, N. &. (2017). *Personas Mediante Reconocimiento Facial Aplicado a Videovigilancia ”*.

Chandra, M. L. R., Kumar, B. V., & Sureshbabu, B. (2018). IoT enabled home with smart security. *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing, ICECDS 2017*, 1193–1197. https://doi.org/10.1109/ICECDS.2017.8389630

Detección de Rostros basado en Filtros Haar + Adaboost. (n.d.). Retrieved December 8,

2020, from https://carlosjuliopardoblog.wordpress.com/2017/05/12/filtros-haar-deteccion-de-rostros/

Ekman, P. (2005). Basic Emotions. In *Handbook of Cognition and Emotion* (pp. 45–60). John Wiley & Sons, Ltd. https://doi.org/10.1002/0470013494.ch3

Gámez Jiménez, C. V. (2009). *Diseño y Desarrollo de un Sistema de Reconocimiento de Caras*. 160.

García del Prado, N., González Castro, V., Alegre, E., & Fidalgo Fernández, E. (2020). *Comparación de métodos de detección de rostros en imágenes digitales*. 976–982. https://doi.org/10.17979/spudc.9788497497749.0976

Gordillo, F., Ángel Pérez, M., Mestas, L., Salvador, J., Arana, J. M., & López, R. M. (2015). Diferencias en el Reconocimiento de las Emociones en Niños de 6 a 11 Años11Este trabajo ha sido realizado gracias a una ayuda a la investigación de la Universidad Camilo José Cela (I+D+i Research Grants). *Acta de Investigación Psicológica*, *5*(1), 1846–1859. https://doi.org/10.1016/s2007-4719(15)30005-3

Hirvin Gonzalez, S. V. (2019). *Reconocimiento Facial utilizando Viola-Jones y Patrones Binarios*. *23*, 1. file:///C:/Users/SBryan/Downloads/126-Artículo-287-2-10-20190912.pdf

Matas, A. R., & Rojas, A. C. (2010). Trabajando con imágenes digitales en clase de Matemáticas. *Gaceta de La Real Sociedad Matemática Española*, *13*(2), 317–336.

Mihai, A.-E. (2014). Sistema De Reconocimiento Facial. *Zaguan.Unizar.Es*, 75. http://zaguan.unizar.es/TAZ/EUCS/2014/14180/TAZ-TFG-2014-408.pdf

Molinero Díez, G. (2010). Segmentación De Imágenes En Color Basada En El Crecimiento De Regiones. *Etsi*, *Ingeniería de telecomunicación*, 5–10.

Profundidad de bits. (n.d.). Retrieved December 9, 2020, from

http://cefire.edu.gva.es/pluginfile.php/199804/mod_resource/content/0/contenidos/006/profundidad_de_bits.html

Tema, P. I. D. (2017). *Tema 3: Filtros*. 1–37. http://alojamientos.us.es/gtocoma/pid/tema3-1.pdf

Tutor, S. Z., & Fern, E. (2015). *Autor: María Sierra Zapata Tutor: Eduardo Fernández Camacho*.

Anexo Código Programación

```
# coding: utf-8


# In[ ]:



import cv2

import os

import imutils

import numpy as np

import time

from tkinter import *

from tkinter import ttk

import tkinter as tk

from tkinter import messagebox


class Ventana:
```

```
def __init__(self):


    self.ventana = Tk()

    self.ventana.title("Reconocimiento Automatico de Emociones")

    self.ventana.geometry("1000x500")

    selected = IntVar()

    self.cuaderno = ttk.Notebook(self.ventana)


    seleccion = IntVar()


    self.pestana1 = ttk.Frame(self.cuaderno)

    self.cuaderno.add(self.pestana1, text="Inicio")


    TituloTesis = Label(self.pestana1, text="DESARROLLO DE SISTEMA PARA LA DETECCION DE EMOCIONES UTILIZANDO", anchor="center", font=("Arial Bold",15))

    TituloTesis.place(x= 270, y = 130)

    TituloTesis = Label(self.pestana1, text="VISION E INTELIGENCIA ARTIFICIAL APLICADO A ENTORNOS EDUCATIVOS", anchor="center", font=("Arial Bold",15))
```

```
        TituloTesis.place(x= 270, y = 160)

        TituloFacultad = Label(self.pestana1, text="INGENIERIA EN TELECOMUNICACIONES - DEPARTAMENTO EEST", anchor="center", font=("Arial Bold",15))

        TituloFacultad.place(x= 320, y = 220)

        TituloFacultad = Label(self.pestana1, text="FACULTAD DE INGENIERIAS Y ARQUITECTURA", anchor="center", font=("Arial Bold",15))

        TituloFacultad.place(x= 390, y = 250)

        TituloNombre = Label(self.pestana1, text="JUAN DAVID PINZON NARANJO", anchor="center", font=("Arial Bold",15))

        TituloNombre.place(x= 450, y = 310)

        TituloAno = Label(self.pestana1, text="2020-2", anchor="center", font=("Arial Bold",15))

        TituloAno.place(x= 530, y = 340)


        img_U =PhotoImage(file="portada.png")

        lbl_imagen = Label(self.pestana1, image=img_U, anchor="center")

        lbl_imagen.place(x=50, y=100)



        self.pestana2 = ttk.Frame(self.cuaderno)
```

```
self.cuaderno.add(self.pestana2, text="Crear Base de Datos")


etiqueta = tk.Label(self.pestana2, text="""Selecciona la emocion que desea guardar en la base de datos con tu rostro:""", font=("Arial Bold",15), justify = tk.LEFT, padx = 100)


emo1 = tk.Radiobutton(self.pestana2,text='Alegria', indicatoron = 0, width = 20,padx = 0, value=1, variable=seleccion)

img_emo1 =PhotoImage(file="alegria.png")

lbl_imagen1 = Label(self.pestana2, image=img_emo1, anchor="center")

lbl_imagen1.place(x=15, y=200)

emo2 = tk.Radiobutton(self.pestana2,text='Asco', indicatoron = 0, width = 20,padx = 0, value=2, variable=seleccion)

img_emo2 =PhotoImage(file="asco.png")

lbl_imagen2 = Label(self.pestana2, image=img_emo2, anchor="center")

lbl_imagen2.place(x=160, y=200)

emo3 = tk.Radiobutton(self.pestana2,text='Ira', indicatoron = 0, width = 20,padx = 0, value=3, variable=seleccion)

img_emo3 =PhotoImage(file="ira.png")

lbl_imagen3 = Label(self.pestana2, image=img_emo3, anchor="center")

lbl_imagen3.place(x=300, y=200)
```

```
        emo4 = tk.Radiobutton(self.pestana2,text='Miedo', indicatoron = 0, width = 20,padx = 0, value=4, variable=seleccion)

        img_emo4 =PhotoImage(file="miedo.png")

        lbl_imagen4 = Label(self.pestana2, image=img_emo4, anchor="center")

        lbl_imagen4.place(x=440, y=200)

        emo5 = tk.Radiobutton(self.pestana2,text='Neutro', indicatoron = 0, width = 20,padx = 0, value=5, variable=seleccion)

        img_emo5 =PhotoImage(file="neutro.png")

        lbl_imagen5 = Label(self.pestana2, image=img_emo5, anchor="center")

        lbl_imagen5.place(x=580, y=200)

        emo6 = tk.Radiobutton(self.pestana2,text='Sorpresa', indicatoron = 0, width = 20,padx = 0, value=6, variable=seleccion)

        img_emo6 =PhotoImage(file="sorpresa.png")

        lbl_imagen6 = Label(self.pestana2, image=img_emo6, anchor="center")

        lbl_imagen6.place(x=720, y=200)

        emo7 = tk.Radiobutton(self.pestana2,text='Trizteza', indicatoron = 0, width = 20,padx = 0, value=7, variable=seleccion)

        img_emo7 =PhotoImage(file="tristeza.png")

        lbl_imagen7 = Label(self.pestana2, image=img_emo7, anchor="center")

        lbl_imagen7.place(x=860, y=200)
```

```
        def clicked():

            emocion=seleccion.get()

            if (emocion==1):
                messagebox.showinfo('Mensaje', 'Por favor, Sonria por 5 segundos')
                emotionName = 'Alegria'
                dataPath = 'E:\Tesis\Data' #Cambia a la ruta donde hayas almacenado Data
                emotionsPath = dataPath + '/' + emotionName

                if not os.path.exists(emotionsPath):
                    print('Carpeta creada: ',emotionsPath)
                    os.makedirs(emotionsPath)

                cap = cv2.VideoCapture(0)

                faceClassif                                                    =
cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
                count = 0
```

```
while True:

    ret, frame = cap.read()
    if ret == False: break
    frame =  imutils.resize(frame, width=640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    auxFrame = frame.copy()

    faces = faceClassif.detectMultiScale(gray,1.3,5)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
        rostro = auxFrame[y:y+h,x:x+w]
        rostro = cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
        cv2.imwrite(emotionsPath + '/cara_{}.jpg'.format(count),rostro)
        count = count + 1
    cv2.imshow('Capturando',frame)
```

```
            k =  cv2.waitKey(1)
            if k == 27 or count >= 200:
                break

        cap.release()
        cv2.destroyAllWindows()

    if (emocion==2):
        messagebox.showinfo('Mensaje', 'Por favor, Cara de Asco por 5 segundos')
        emotionName = 'Asco'
        dataPath = 'E:\Tesis\Data' #Cambia a la ruta donde hayas almacenado Data
        emotionsPath = dataPath + '/' + emotionName

        if not os.path.exists(emotionsPath):
            print('Carpeta creada: ',emotionsPath)
            os.makedirs(emotionsPath)

        cap = cv2.VideoCapture(0)
```

```
faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
count = 0

while True:

    ret, frame = cap.read()
    if ret == False: break
    frame =  imutils.resize(frame, width=640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    auxFrame = frame.copy()

    faces = faceClassif.detectMultiScale(gray,1.3,5)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
        rostro = auxFrame[y:y+h,x:x+w]
        rostro = cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
```

```
            cv2.imwrite(emotionsPath + '/cara_{}.jpg'.format(count),rostro)
            count = count + 1
        cv2.imshow('Capturando',frame)

        k =  cv2.waitKey(1)
        if k == 27 or count >= 200:
            break

    cap.release()
    cv2.destroyAllWindows()

if (emocion==3):
    messagebox.showinfo('Mensaje', 'Por favor, Enojese por 5 segundos')
    emotionName = 'Ira'
    dataPath = 'E:\Tesis\Data' #Cambia a la ruta donde hayas almacenado Data
    emotionsPath = dataPath + '/' + emotionName

    if not os.path.exists(emotionsPath):
        print('Carpeta creada: ',emotionsPath)
```

```
            os.makedirs(emotionsPath)

        cap = cv2.VideoCapture(0)

        faceClassif                                                    =
cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
        count = 0

        while True:

            ret, frame = cap.read()
            if ret == False: break
            frame =  imutils.resize(frame, width=640)
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            auxFrame = frame.copy()

            faces = faceClassif.detectMultiScale(gray,1.3,5)

            for (x,y,w,h) in faces:
```

```
                    cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
                    rostro = auxFrame[y:y+h,x:x+w]
                    rostro = cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
                    cv2.imwrite(emotionsPath + '/cara_{}.jpg'.format(count),rostro)
                    count = count + 1
                cv2.imshow('Capturando',frame)

                k =  cv2.waitKey(1)
                if k == 27 or count >= 200:
                    break

            cap.release()
            cv2.destroyAllWindows()

        if (emocion==4):
            messagebox.showinfo('Mensaje', 'Por favor, Cara de Miedo por 5 segundos')
            emotionName = 'Miedo'
            dataPath = 'E:\Tesis\Data' #Cambia a la ruta donde hayas almacenado Data
```

```
emotionsPath = dataPath + '/' + emotionName


if not os.path.exists(emotionsPath):
    print('Carpeta creada: ',emotionsPath)
    os.makedirs(emotionsPath)


cap = cv2.VideoCapture(0)


faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
count = 0


while True:


    ret, frame = cap.read()
    if ret == False: break
    frame =  imutils.resize(frame, width=640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    auxFrame = frame.copy()
```

```
                faces = faceClassif.detectMultiScale(gray,1.3,5)


                for (x,y,w,h) in faces:
                    cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
                    rostro = auxFrame[y:y+h,x:x+w]
                    rostro = cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
                    cv2.imwrite(emotionsPath + '/cara_{}.jpg'.format(count),rostro)
                    count = count + 1
                cv2.imshow('Capturando',frame)


                k =  cv2.waitKey(1)
                if k == 27 or count >= 200:
                    break


            cap.release()
            cv2.destroyAllWindows()
```

```
            if (emocion==5):
                messagebox.showinfo('Mensaje', 'Por favor, Manten tu cara sin expresion por 5 segundos')
                emotionName = 'Neutro'
                dataPath = 'E:\Tesis\Data' #Cambia a la ruta donde hayas almacenado Data
                emotionsPath = dataPath + '/' + emotionName

                if not os.path.exists(emotionsPath):
                    print('Carpeta creada: ',emotionsPath)
                    os.makedirs(emotionsPath)

                cap = cv2.VideoCapture(0)

                faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
                count = 0

                while True:

                    ret, frame = cap.read()
```

```
                    if ret == False: break
                    frame =  imutils.resize(frame, width=640)
                    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                    auxFrame = frame.copy()

                    faces = faceClassif.detectMultiScale(gray,1.3,5)

                    for (x,y,w,h) in faces:
                        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
                        rostro = auxFrame[y:y+h,x:x+w]
                        rostro                                                    =
cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
                        cv2.imwrite(emotionsPath + '/cara_{}.jpg'.format(count),rostro)
                        count = count + 1
                    cv2.imshow('Capturando',frame)

                    k =  cv2.waitKey(1)
                    if k == 27 or count >= 200:
                        break
```

```
            cap.release()
            cv2.destroyAllWindows()

        if (emocion==6):
            messagebox.showinfo('Mensaje', 'Por favor, Cara de Sorpresa por 5 segundos')
            emotionName = 'Sorpresa'
            dataPath = 'E:\Tesis\Data' #Cambia a la ruta donde hayas almacenado Data
            emotionsPath = dataPath + '/' + emotionName

            if not os.path.exists(emotionsPath):
                print('Carpeta creada: ',emotionsPath)
                os.makedirs(emotionsPath)

            cap = cv2.VideoCapture(0)

            faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
            count = 0
```

```
while True:

    ret, frame = cap.read()
    if ret == False: break
    frame =  imutils.resize(frame, width=640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    auxFrame = frame.copy()

    faces = faceClassif.detectMultiScale(gray,1.3,5)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
        rostro = auxFrame[y:y+h,x:x+w]
        rostro = cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
        cv2.imwrite(emotionsPath + '/cara_{}.jpg'.format(count),rostro)
        count = count + 1
    cv2.imshow('Capturando',frame)
```

```
            k =  cv2.waitKey(1)
            if k == 27 or count >= 200:
                break

        cap.release()
        cv2.destroyAllWindows()

    if (emocion==7):
        messagebox.showinfo('Mensaje', 'Por favor, Trizte por 5 segundos')
        emotionName = 'Trizteza'
        dataPath = 'E:\Tesis\Data' #Cambia a la ruta donde hayas almacenado Data
        emotionsPath = dataPath + '/' + emotionName

        if not os.path.exists(emotionsPath):
            print('Carpeta creada: ',emotionsPath)
            os.makedirs(emotionsPath)

        cap = cv2.VideoCapture(0)
```

```
        faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
        count = 0

        while True:

            ret, frame = cap.read()
            if ret == False: break
            frame =  imutils.resize(frame, width=640)
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            auxFrame = frame.copy()

            faces = faceClassif.detectMultiScale(gray,1.3,5)

            for (x,y,w,h) in faces:
                cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
                rostro = auxFrame[y:y+h,x:x+w]
                rostro = cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
```

```
            cv2.imwrite(emotionsPath + '/cara_{}.jpg'.format(count),rostro)
            count = count + 1
        cv2.imshow('Capturando',frame)

        k =  cv2.waitKey(1)
        if k == 27 or count >= 200:
            break

    cap.release()
    cv2.destroyAllWindows()


btn = Button(self.pestana2, text=" Seleccionar \n Emocion ", command=clicked)

etiqueta.place(x= 30, y= 100)
emo1.place(x= 5, y = 350)
emo2.place(x= 145, y = 350)
emo3.place(x= 285, y = 350)
emo4.place(x= 425, y = 350)
```

```
            emo5.place(x= 565, y = 350)

            emo6.place(x= 705, y = 350)

            emo7.place(x= 845, y = 350)

            btn.place(x= 450, y = 430)


            self.pestana3 = ttk.Frame(self.cuaderno)

            self.cuaderno.add(self.pestana3, text="Reconocimiento de Emociones Faciales")


            img_UE =PhotoImage(file="encabezado.png")

            lbl_Enca = Label(self.pestana3, image=img_UE, anchor="center")

            lbl_Enca.place(x=150, y=100)


            def clicked_M():


                def obtenerModelo(method,facesData,labels):

                    if method == 'LBPH': emotion_recognizer = cv2.face.LBPHFaceRecognizer_create()

                    # Entrenando el reconocedor de rostros

                    #print("Entrenando ( "+method+" )...")
```

```
    inicio = time.time()
    emotion_recognizer.train(facesData, np.array(labels))
    tiempoEntrenamiento = time.time()-inicio
    print("Tiempo de entrenamiento ( "+method+" ): ", tiempoEntrenamiento)

    # Almacenando el modelo obtenido
    emotion_recognizer.write("modelo"+method+".xml")

dataPath = 'E:\Tesis\Data' #Cambia a la ruta donde hayas almacenado Data
emotionsList = os.listdir(dataPath)
#print('Lista de personas: ', emotionsList)

labels = []
facesData = []
label = 0

for nameDir in emotionsList:

    emotionsPath = dataPath + '/' + nameDir
```

```
            for fileName in os.listdir(emotionsPath):
                #print('Rostros: ', nameDir + '/' + fileName)
                labels.append(label)
                facesData.append(cv2.imread(emotionsPath+'/'+fileName,0))
                #image = cv2.imread(emotionsPath+'/'+fileName,0)
                #cv2.imshow('image',image)
                #cv2.waitKey(10)
            label = label + 1

        obtenerModelo('LBPH',facesData,labels)
        messagebox.showinfo('Mensaje', 'Modelo LBPH entrenado exitosamente!')

    def clicked_R():

        def emotionImage(emotion):
            # Emojis
            if emotion == 'Alegria': image = cv2.imread('Emoji/alegria.png')
            if emotion == 'Asco': image = cv2.imread('Emoji/asco.png')
```

```
            if emotion == 'Ira': image = cv2.imread('Emoji/ira.png')
            if emotion == 'Miedo': image = cv2.imread('Emoji/miedo.png')
            if emotion == 'Neutro': image = cv2.imread('Emoji/neutro.png')
            if emotion == 'Sorpresa': image = cv2.imread('Emoji/sorpresa.png')
            if emotion == 'Tristeza': image = cv2.imread('Emoji/tristeza.png')
            return image

        # ----------- Métodos usados para el entrenamiento y lectura del modelo ----------
        method = 'LBPH'

        if method == 'LBPH': emotion_recognizer = cv2.face.LBPHFaceRecognizer_create()

        emotion_recognizer.read('modelo'+method+'.xml')
        # -----------------------------------------------------------------------------

        dataPath = 'E:\Tesis\Data' #Cambia a la ruta donde hayas almacenado Data
        imagePaths = os.listdir(dataPath)
        #print('imagePaths=',imagePaths)
```

```
cap = cv2.VideoCapture(0)
faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')

while True:

    ret,frame = cap.read()
    if ret == False: break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    auxFrame = gray.copy()

    nFrame = cv2.hconcat([frame, np.zeros((480,300,3),dtype=np.uint8)])

    faces = faceClassif.detectMultiScale(gray,1.3,5)

    for (x,y,w,h) in faces:
        rostro = auxFrame[y:y+h,x:x+w]
        rostro = cv2.resize(rostro,(150,150),interpolation= cv2.INTER_CUBIC)
```

```
                    result = emotion_recognizer.predict(rostro)


                    cv2.putText(frame,'{}'.format(result),(x,y-
5),1,1.3,(255,255,0),1,cv2.LINE_AA)


                    # LBPHFace
                    if method == 'LBPH':
                        if result[1] < 60:
                            cv2.putText(frame,'{}'.format(imagePaths[result[0]]),(x,y-
25),2,1.1,(0,255,0),1,cv2.LINE_AA)
                            cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
                            image = emotionImage(imagePaths[result[0]])
                            nFrame = cv2.hconcat([frame,image])
                        else:
                            cv2.putText(frame,'No identificado',(x,y-
20),2,0.8,(0,0,255),1,cv2.LINE_AA)
                            cv2.rectangle(frame, (x,y),(x+w,y+h),(0,0,255),2)
                            nFrame =
cv2.hconcat([frame,np.zeros((480,300,3),dtype=np.uint8)])
```

```
            cv2.imshow('nFrame',nFrame)
            k = cv2.waitKey(1)
            if k == 27:
                break

        cap.release()
        cv2.destroyAllWindows()

    def salir():
        self.ventana.destroy()

    btn_2 = Button(self.pestana3, text=" Entrenar Modelo", command=clicked_M)
    btn_2.place(x= 100, y = 330)

    btn_3 = Button(self.pestana3, text=" Reconocimiento de Emociones", command=clicked_R)
    btn_3.place(x= 700, y = 330)

    btn_salir = Button(self.pestana3, text = "SALIR", command = salir, relief=RAISED)
```

```
        btn_salir.place(x = 450, y = 400)


        self.cuaderno.pack(expand=1, fill='both')
        self.ventana.mainloop()



ven=Ventana()
```