UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

DISEÑO DE UN SOFTWARE PILOTO DE RECONOCIMIENTO FACIAL PARA EL CONTROL DE ASISTENCIA EN LA ESCUELA DE TELECOMUNICACIONES DE LA UNIVERSIDAD DE CARABOBO

MOSQUERA L. VANESSA A.
ROMERO V. ELVIS A.

Bárbula, 13 de diciembre del 2016

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE
TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

DISEÑO DE UN SOFTWARE PILOTO DE RECONOCIMIENTO FACIAL PARA EL CONTROL DE ASISTENCIA EN LA ESCUELA DE TELECOMUNICACIONES DE LA UNIVERSIDAD DE CARABOBO

TRABAJO ESPECIAL DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD DE CARABOBO PARA OPTAR AL TÍTULO DE INGENIERO DE TELECOMUNICACIONES

MOSQUERA L. VANESSA A.
ROMERO V. ELVIS A.

Bárbula, 13 de diciembre del 2016

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

CERTIFICADO DE APROBACIÓN

Los abajo firmantes miembros del jurado asignado para evaluar el trabajo especial de grado titulado «DISEÑO DE UN SOFTWARE PILOTO DE RECONOCIMIENTO FACIAL PARA EL CONTROL DE ASISTENCIA EN LA ESCUELA DE TELECOMUNICACIONES DE LA UNIVERSIDAD DE CARABOBO», realizado por los bachilleres MOSQUERA L. VANESSA A., cédula de identidad 20.452.827, ROMERO V. ELVIS A., cédula de identidad 21.136.637, hacemos constar que hemos revisado y aprobado dicho trabajo.

Firma
Prof. ELIMAR HERNÁNDEZ
TUTOR

Firma
Prof. CARLOS APONTE DEZZEO
JURADO

Firma
Prof. CARLOS MEJÍAS
JURADO

Bárbula, 13 de diciembre del 2016

Dedicatoria

Dedicado a Dios, mis padres,
hermano, familiares y demás seres
queridos por ser un pilar fundamental
y brindarme su apoyo durante
toda mi carrrera.

MOSQUERA L. VANESSA A.

Dedicado a Dios, mis padres,
abuelos, hermano, y demás
familiares por todo el apoyo
y la ayuda brindada durante
toda mi carrera.

ROMERO V. ELVIS A.

Agradecimientos

Le agradezco a Dios que gracias a él he llegado a esta estancia de mi carrera.

A Naurea y Jovino por ser los mejores padres del mundo y brindarme todo el apoyo necesario durante toda mi carrera y siempre creer en mí, a mi hermano Juan por su apoyo y ayuda durante todo mi proceso de formación. Mil gracias los amo.

A mis tios Sergia y Nicolás, a mis primos Chiquinquira, Alonzo y Ana, y demás familiares por su ayuda siempre tanto en los momentos buenos y malos.

A mi amigo Elvis, mejor compañero en este trabajo especial de grado no pude tener, gracias por todos estos años de amistad, ayuda y apoyo durante toda mi carrera. Feliz de haber logrado este objetivo juntos.

A Einer por brindarme su apoyo, comprensión y compañía durante este proceso.

A nuestra tutora Elimar Hernández por su motivación en este trabajo especial de grado.

A mis amigos, compañeros y todas aquellas personas que conocí durante toda mi carrera con los cuales compartí tantos momentos.

Mil gracias a todos. ***Mosquera L. Vanessa A.***

A Dios por brindarme la sabiduría y el entendimiento para lograr este objetivo.

A mis padres, abuelos, hermano y demás seres queridos por su apoyo siempre durante toda mi carrera.

A mis amigos, compañeros y todas aquellas personas que formaron parte de mi formación profesional.

A nuestra tutora Elimar Hernández por su apoyo incondicional para lograr esta meta.

Gracias a todos. ***Romero V. Elvis A.***

Índice general

Índice de figuras

Indice de tablas

Acrónimos

DSP	**D**igital **S**ignal **P**rocessing
LOTTT	**L**ey **O**rgánica del **T**rabajador, los Trabajadores y las Trabajadoras
OPENCV	**O**pen Source Computer **V**ision Library
LBPH	**H**istograma de **P**atrones **B**inarios **L**ocales
TIC	**T**ecnologías de la **I**nformación y la **C**omunicación
RGB	**R**ed **G**reen **B**lue
HSV	**H**ue **S**aturation **V**alue

DISEÑO DE UN SOFTWARE PILOTO DE RECONOCIMIENTO FACIAL PARA EL CONTROL DE ASISTENCIA EN LA ESCUELA DE TELECOMUNICACIONES DE LA UNIVERSIDAD DE CARABOBO

por

MOSQUERA L. VANESSA A. y ROMERO V. ELVIS A.

Presentado en el Departamento de Señales y Sistemas
de la Escuela de Ingeniería en Telecomunicaciones
el 13 de diciembre del 2016 para optar al Título de
Ingeniero de Telecomunicaciones

RESUMEN

El siguiente trabajo presenta el diseño de un software piloto que optimiza el registro de asistencia de la Escuela de Telecomunicaciones de la Universidad de Carabobo, brindando un mayor nivel fiabilidad por medio de la detección y reconocimiento facial de todo el personal que labora en el lugar, para ello, con la ayuda de la librería OpenCV se usó la tecnología de visión artificial, captura y comparación de imágenes, procesamiento digital de la mismas y de esta forma identificar a las personas. Además con el uso del lenguaje de programación Python se realizó una interfaz gráfica de usuario para la administración del software realizando pruebas para evaluar la eficiencia y eficacia del mismo.

Palabras Claves: OpenCV, Reconocimiento facial, Procesamiento digital de imágenes, Control de asistencia

Tutor: ELIMAR HERNÁNDEZ
Profesor del Departamento de Señales y Sistemas
Escuela de Telecomunicaciones. Facultad de Ingeniería

Capítulo I

Introducción

1.1. Motivación

En toda institución, pública o privada, la relación de trabajo entre empleador y trabajador está compuesta por un grupo de normas y valores que fungen en el buen desempeño de las actividades diarias. En tal sentido, asistir al lugar de trabajo es un deber que toda persona tiene; por otro lado, el jefe en cuestión es quien debe supervisar que se cumplan las normas, en especial la Ley Orgánica del Trabajador, los Trabajoderes y las Trabajadoras (LOTTT) que establece en su artículo 79 que una de las razones por la que se puede efectuar un despido justificado es por la inasistencia del personal a su lugar de trabajo [1].

Como se señaló, el cumplimiento del horario de trabajo es de gran importancia, por lo que el control de asistencia se debe llevar a cabo diariamente, y en muchas instituciones el registro diario que se lleva es de forma manual; es decir, no se hace un seguimiento exhautivo de su funcionamiento, caso que se puede prestar para fraudes en el desempeño del mismo.

La Escuela de Telecomunicaciones de la Universidad de Carabobo tiene personal docente y administrativo que prestan servicios dentro de la misma. Es importante resaltar, que actualmente se cuenta con un registro de asistencia para todo el personal, siendo computarizada para los profesores y de forma manual (llenado

de planilla) para los estudiantes que allí laboran. A todo esto, es el Director de la escuela quien debe supervisar los registros del personal docente y administrativo, lo que implica disposición de tiempo para analizar las asistencias y ninguna herramienta que le permita certificar que no se haya cometido fraude en las mismas. Es por ello, que se requiere de una nueva estrategia que permita automatizar todos los registros de asistencia, además de brindar fiabilidad en cuanto al cumplimiento de la misma.

En otro orden de ideas, la incorporación de tecnología de visión artificial en específico el reconocimiento facial al registro de asistencia que se diseñará para la Escuela de Telecomunicaciones aumentará de forma considerable la confiabilidad del mismo. Un sistema de reconocimiento facial debe permitir la identificación en forma automática de una persona en una imagen o video digital; bajo esta premisa, se requiere identificar la presencia de un rostro en una imagen digital para luego ser comparada en una base de datos previamente establecida y así verificar que realmente es el trabajador quien registra la asistencia [2].

Por ello, es conveniente resaltar algunos trabajos de investigación que guardan relación con el tema en cuestión y proporcionan información relevante en la solución de la problemática.

Respecto al tema, Malpica, G. y Mogollón, N. (2015) en su trabajo especial de grado de la Universidad de Carabobo, Valencia titulada «Desarrollo de un software para la detección de cadenas nacionales mediante la identificación de patrones de imágenes» señalaron que por medio de la tecnología de visión artificial se detecta un patrón de imagen en específico el de la coletilla de cadena nacional, lo cual es de gran relevancia ya que muestra los aspectos a tener en cuenta para la captura y comparación de imágenes [3].

Por su parte, Scarel, G. (2010) en su tesis de pregrado de la Universidad Nacional del Litoral Santa Fe, Argentina, denominada «Sistema de reconocimiento facial» describió las características a tener en cuenta cuando se quiere detectar el rostro e igualmente, muestra las diferentes técnicas que usan para tal fin [4].

Igualmente, Dra. Bracho M. (2008) en su trabajo de ascenso de la Universidad Central Lisandro Alvarado, Barquisimeto que lleva por nombre «Sistemas de reconocimiento de rostros para maggie», incorporó al robot Maggie la función de detectar rostros, lo que permite capturar la imagen del rostro, caracterizar la imagen, almacenar en una base de datos y comparar para identificar la persona; además, muestra los aspectos básicos a considerar en la captura de imagen [2].

Por todo lo expuesto, surge la necesidad de diseñar un software piloto de reconocimiento facial para el control de asistencia en la Escuela de Telecomunicaciones de la Universidad de Carabobo que brinde mayores niveles de seguridad, y que permita automatizar dicho proceso.

El desarrollo de un software de reconocimiento facial para el control de asistencia tiene un costo bajo de implementación debido a que solo se requiere la adquisición de una cámara web que son muy económicas en comparación a un sistema con tecnologías de reconocimiento de identidad para el control de asistencia, además su programación es por medio de software Open Source, lo que implica que no se tiene que hacer una inversión monetaria para su utilización.

Por otro lado, la implementación de un software de detección facial disminuye los trámites administrativos para la identificación y verificación del personal, debido a que se hace de forma automática; además, se puede ejecutar en cualquier organización debido a que será diseñado en un entorno de programación de software libre. Del mismo modo, motivará el valor y la conciencia del trabajo entre los empleados evitando la ausencia y el retraso de llegada de los mismos.

De igual modo, la visión artificial por computador es una rama de estudio del procesamiento digital de imágenes y video que tiene una amplia aplicación en nuestros días, como lo pueden ser sistemas inteligentes, sistemas de seguridad, entre otros. Es por ello, que el desarrollo de este trabajo servirá de apoyo para futuras aplicaciones que se deseen desarrollar en esta área.

1.2. Objetivos

1.2.1. Objetivo General

Diseñar un software piloto de reconocimiento facial para el control de asistencia en la Escuela de Telecomunicaciones de la Universidad de Carabobo.

1.2.2. Objetivos Específicos

1. Identificar los parámetros que intervienen en la detección de rostros y así generar criterios para el reconocimiento facial.

2. Componer un algoritmo utilizando las librerías de OpenCV, que permita la identificación y comparación de un rostro en un video generado por una cámara web.

3. Diseñar una interfaz gráfica para la administración del programa de registro de asistencia en la Escuela de Telecomunicaciones de la Universidad de Carabobo.

4. Evaluar el desempeño del programa de reconocimiento facial en el registro y control de asistencia.

1.3. Alcances

Mediante esta investigación se busca desarrollar un software piloto de reconocimiento facial para llevar el control de asistencia de forma automatizada en la Escuela de Telecomunicaciones de la Universidad de Carabobo, el cual permitirá tener el registro de horas diarias del personal, un campo para el llenado de observaciones, agenda sincronizada entre todo el personal; es decir, si se quiere transmitir un mensaje por este medio le podrá llegar a todo el personal. Aún sin conexión a Internet el software piloto se mantendrá operativo. Usando herramientas de software libre se diseñó una interfaz gráfica de usuario, con el propósito de mejorar la experiencia con el usuario.

Capítulo II

Marco Conceptual

2.1. Visión Artificial

La visión artificial tiene como finalidad la extracción de información del mundo físico a partir de imágenes, utilizando para ello una computadora.

Un sistema de visión artificial actúa sobre una representación de una realidad que le proporciona información sobre brillo, colores, formas, etc. Estas representaciones suelen estar en forma de imágenes estáticas, escenas tridimensionales o imágenes en movimiento [5].

2.1.1. Etapas de un Sistema de Visión Artificial

La visión artificial, en un intento de reproducir este comportamiento, define tradicionalmente cuatro etapas principales:

• Etapa 1, la cual es puramente sensorial, consiste en la captura o adquisición de imágenes digitales mediante algún tipo de sensor.

• Etapa 2, consiste en el tratamiento digital de las imágenes, con objeto de facilitar las etapas posteriores. En esta etapa de procesamiento previo es donde, mediante filtros y transformaciones geométricas, se eliminan partes indeseables de la

imagen o se realzan las partes de interés.

• Etapa 3, conocida como segmentación, consiste en aislar los elementos que interesan de una escena para comprenderla.

• Etapa 4, por último se encuentra la etapa de reconocimiento o clasificación, en la que se pretende distinguir los objetos segmentados, gracias al análisis de ciertas características que se establecen previamente para diferenciarlos.

Estas cuatro etapas no se siguen siempre de manera secuencial sino que, en ocasiones, suelen retroalimentarse, como se observa en la figura 2.1.

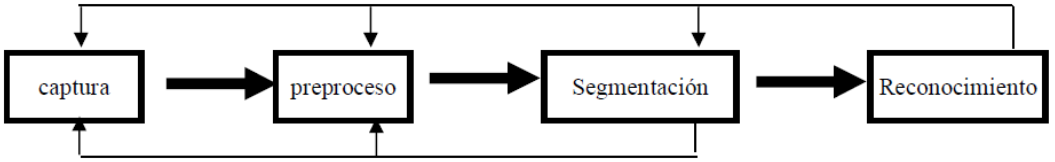


Figura 2.1: Etapas de un Sistema de Visión Artificial. Fuente: [5].

2.2. Procesamiento Digital de Imágenes

El procesamiento digital de imágenes, incluye un conjunto de técnicas que operan sobre la representación digital de una imagen, a objeto de destacar algunos de los elementos que conforman la escena, de modo que se facilite su posterior análisis, bien sea por parte de un usuario (humano) o un sistema de visión artificial. En general, las técnicas de procesamiento de imágenes son aplicadas cuando resulta necesario realzar o modificar una imagen para mejorar su apariencia o para destacar algún aspecto de la información contenida en la misma, o cuando se requiere, medir, contrastar o clasificar algún elemento contenido en la misma. También se utilizan cuando se requiere combinar imágenes o porciones de las mismas o reorganizar su contenido [6].

2.2.1. Imagen Digital

Una imagen en escala grises puede ser definida como una función bidimensional f(x,y), donde x e y son coordenadas espaciales, y el valor de f en un par de coordenadas dado es denominado intensidad o nivel de gris. Cuando x, y y f son cantidades discretas, la imagen es llamada imagen digital. De este modo, ésta puede ser representada como una matriz MxN elementos llamados píxeles [7].

2.2.2. Segmentación

En cualquier proceso de reconocimiento de imágenes, el paso fundamental es diferenciar claramente todos los elementos que componen la imagen. Esto es segmentar los diferentes objetos del fondo. Para ello se pueden aplicar multitud de procesos y técnicas distintas, dependiendo del tipo de imagen y el resultado deseado. Normalmente, la segmentación de una imagen monocromática se basa en las características de las tonalidades de gris, tales como la discontinuidad y la similitud. La discontinuidad busca líneas, bordes o puntos en función a cambios abruptos en la tonalidad de gris, mientras que la similitud establece regiones basándose en las relaciones espaciales o cromáticas que puede haber entre los píxeles que la forman [8].

2.3. Detección de Objetos

La detección de objetos en movimiento es una tarea común en aplicaciones de visión computacional, una de las estrategias más sencillas es realizando diferencias de imágenes sucesivas. Sin embargo, el ruido de las imágenes puede dar grandes diferencias donde no hay movimiento, otro problema si el obstáculo en movimiento se desplaza muy lentamente en relación a la frecuencia con que se captan las imágenes, la diferencia resultante puede ser tan pequeña que no se detecte el movimiento. Una alternativa es comparar las imágenes mediante un procesamiento

por regiones, en vez de hacerlo entre píxeles individuales, lo cual reduce la influencia de ruido. Otra forma es mediante detección de bordes que es más robusto ante los cambios de iluminación. El flujo óptico es otro método de estimación de movimiento que puede ser implementado mediante correlación, técnicas de gradiente o filtros sensibles a la velocidad. Este método solo es efectivo si el movimiento es significativo entre imágenes. También se emplea el método de comparar cada nueva imagen con una de referencia que contenga sólo la parte estática de la escena [9].

2.3.1. Detección Facial

La detección facial implica encontrar las áreas dentro de una imagen o video que contienen un rostro, es decir se trata de descartar todo lo que sea fondo, y así obtener la ubicación y tamaño exacto de la cara [10]. El proceso de identificación facial se divide básicamente en dos tareas: detección y reconocimiento.

El procedimiento metodológico seguido para la detección de rostros se basa en el algoritmo planteado por Viola Jones. Un esquema general para un sistema de detección facial se muestra en la figura 2.2. Esta se divide en tres etapas:

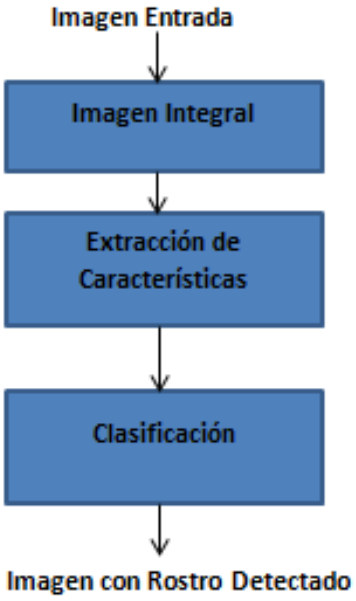


Figura 2.2: Esquema para un Sistema de Detección Facial. Fuente: [10]

• Etapa 1, en esta se realiza una transformación de la imagen de entrada generando una nueva, llamada imagen integral. Esta nueva representación de la imagen es introducida en el algoritmo Viola Jones.

• Etapa 2, en esta se realiza la extracción de características usando filtros con base Haar. Estos filtros pueden ser calculados eficientemente sobre la imagen integral, son selectivos en la orientación espacial y frecuencia, y permiten ser modificados en escala y orientación.

• Etapa 3, en esta última se usa boosting (Adaboot) para la construcción de clasificadores en cascada.

2.3.2. Comparación de Patrones

La combinación de diferentes datos dan forma a una arreglo de patrones; estos datos pueden ser numéricos, píxeles, sonido, entre otros. Los patrones son importantes en la visión artificial porque determinan la representación de un objeto en una imagen, si estos patrones son detectados, entonces es un indicador que ese objeto está ocurriendo en esa imagen. Lamentablemente estos patrones tienden a ser afectados por el ruido, la iluminación, la orientación, entre otros; es por ello que se tiende a extraer características de la imagen y realzarla para que sobresalga, y con ello utilizarla como patrón para la detección de estos mismos en otras imágenes [11].

2.3.3. Algoritmo Haar-Like Feature

En la detección de objetos, se necesitan tres etapas: extracción de características, generación de candidatos y decisión. Para el caso de la detección de rostros se trabaja con el algoritmo planteado por Viola-Jones [12] llamado Haar-Like Feature, este descriptor trabaja de forma diferente con respecto a otros, ya que no analiza pixel a pixel de la imagen, sino que analiza características, en específico se usan tres características Haar [13].

✓ Característica de 2 rectángulos

✓ Característica de 3 rectángulos

✓ Característica de 4 rectángulos

El algoritmo de Viola-Jones es eficiente dado que disminuye considerablemente el cálculo realizado para la detección, como se mencionó, no trabaja pixel a pixel, sino que utiliza técnicas como la de integral image o imagen integrada que permite la unión de dos o más pixeles para formar una característica, además de utilizar adaboost el cual es un algoritmo de aprendizaje, el cual sirve para generar datos de comparación entre características.

Con todo lo anterior, se llega al proceso final, el cual consiste en la cascada de decisión, es decir, se va analizando una a una las características y en base a los datos guardados el algoritmo decide si esa característica pertenece a un rostro o no, de ser negativo desecha dicha característica.

2.3.3.1. Imagen Integral

Esta imagen permite extraer de forma rápida características a diferentes escalas ya que no se trabaja directamente con los valores de intensidad si no con una imagen acumulativa que se construye a partir de operaciones básicas. A la hora de crear un sistema de detección facial resulta crucial encontrar un compromiso entre velocidad y eficiencia. Mediante el uso de una nueva representación de las imágenes, llamada imagen integral, Viola y Jones describen un método de evaluación de características de manera efectiva y a mayor velocidad [14].

Figura 2.3: Imagen Integral. Fuente:[15]

La imagen integral figura 2.3 y 2.4, en la localización (x,y), contiene la suma de los píxeles de la parte superior izquierda de la imagen y se puede calcular como se indica a continuación:

$$II(x,y) = \sum_{x' \leqslant x \, , y' \leqslant y} Im(x',y') \tag{2.1}$$

Donde II(x,y,) es la imagen integral e Im(x′,y′) es la imagen original a procesar.

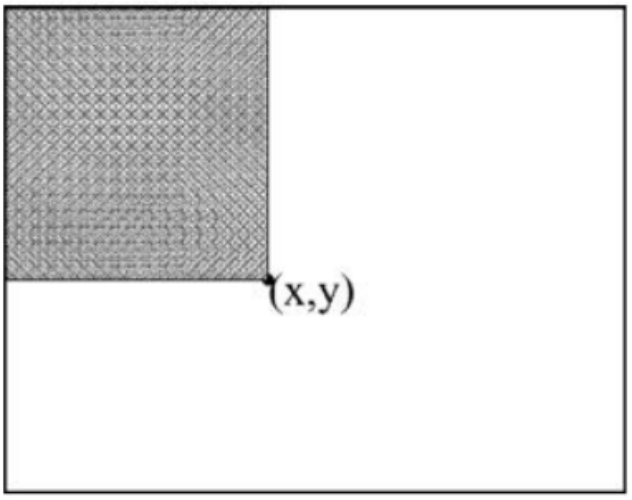


Figura 2.4: Píxeles utilizados para representar la imagen integral de un pixel (x,y). Fuente:[16]

2.3.3.2. Extracción de Características

En imágenes las características de cada objeto se extraen al aplicar ciertas funciones que permitan la representación y descripción de los objetos de interés de la imagen (patrones). La extracción de características es un paso en el reconocimiento de patrones en el cuál las medidas u observaciones son procesadas para encontrar atributos que puedan ser usados para asignar los objetos a determinada clase [15].

En la figura 2.5, se observan algunos de los filtros usados para la extracción de características.

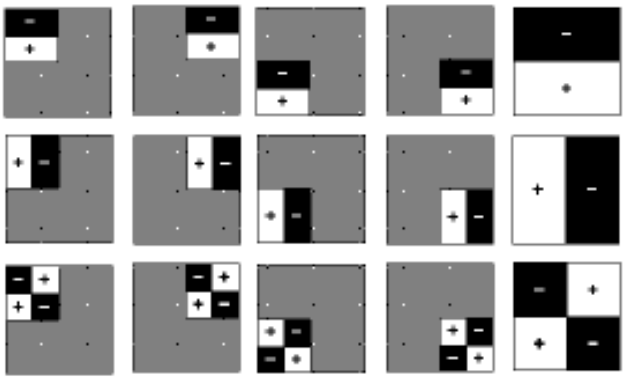


Figura 2.5: Filtros Haar rotados, trasladados y con cambios de escalas. Fuente:[15]

Cada característica se superpone sobre la imagen en todas las posiciones y todos los tamaños posibles. En la figura 2.6, se muestra la convolución de un filtro Haar con la imagen integral. De esta operación se puede extraer una característica en un tiempo constante sobre la imagen integral adicionando y sustrayendo los valores de los vértices para cada rectángulo. Para mayor claridad, en la figura la suma de los píxeles que forman el rectángulo D se puede calcular como:

$$\mathrm{sumD} = (4 + 1) - (2 + 3) \tag{2.2}$$

donde 1, 2, 3, 4 son los valores dados en la imagen integral en dichas localizaciones.

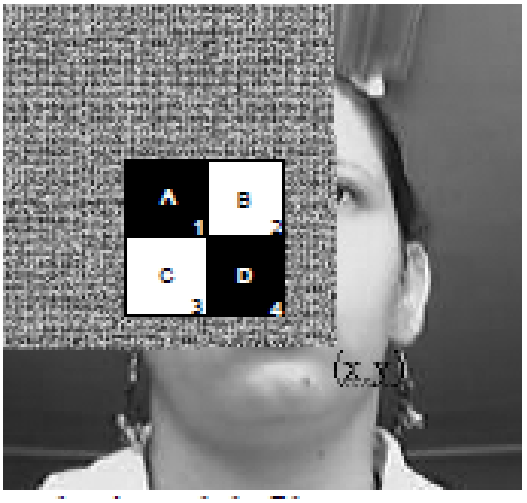


Figura 2.6: Convolución del Filtro Haar con una Imagen Integral. Fuente: [15]

Es decir, el resultado o valor de las característica se obtiene al restar a la suma de los píxeles debajo de la zona negra de la característica, la suma de todos los píxeles bajo la zona blanca de la característica [16]. Más ejemplos de esto se observa en la figura 2.7 para su correcta compresión.

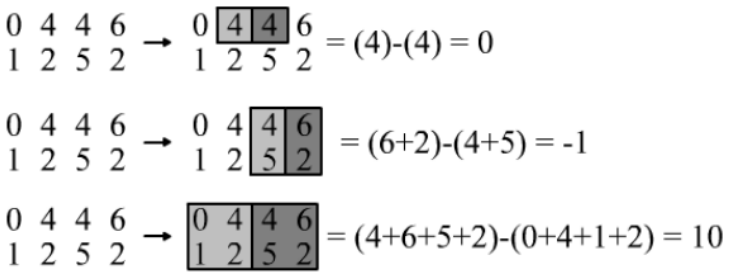


Figura 2.7: Cálculo de las Características de una Matriz. Fuente:[16]

2.3.3.3. Clasificación

Boosting es un método de clasificación que combina varios clasificadores básicos para formar un único clasificador más complejo y preciso. La idea se basa en la afirmación de que varios clasificadores sencillos, cada uno de ellos con una precisión ligeramente superior a una clasificación aleatoria, pueden combinarse para formar un clasificador de mayor precisión, siempre y cuando se disponga de un número suficiente de muestras de entrenamiento. En la figura 2.8, se muestra un esquema de un clasificador en cascada [15]. Dentro del algoritmo de detección este se encarga de asignar un conjunto de características dado a una clase con la que se encuentra una mayor similitud, de acuerdo a un modelo inducido durante el entrenamiento.

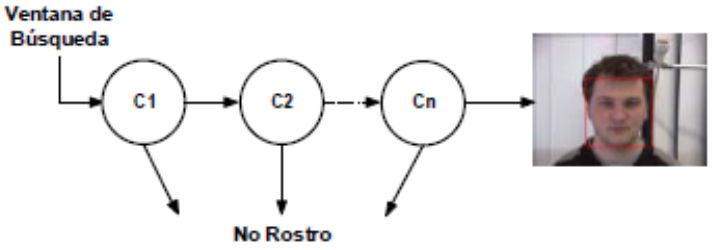


Figura 2.8: Clasificador en Cascada. Fuente:[15]

Para aplicar la técnica de boosting primero se debe establecer un algoritmo de aprendizaje sencillo (clasificador débil o base), que será llamado repetidas veces para crear diversos clasificadores base. Para el entrenamiento de los clasificadores base se emplea, en cada iteración, un subconjunto diferente de muestras de entrenamiento y una distribución de pesos diferente sobre las muestras de entrenamiento [17]. Finalmente, estos clasificadores base se combinan en un único clasificador que se espera sea mucho más preciso que cualquiera de los clasificadores base por separado.

En función de los clasificadores base que se utilicen, las distribuciones que se empleen para entrenarlos y el modo de combinarlos, podrán crearse distintas clases del algoritmo genérico de boosting. El algoritmo de boosting empleado por Viola y Jones en su trabajo es conocido como AdaBoost [15].

A continuación se muestra el algoritmo usado en el procedimiento de investigación. Este se encuentra planteado en [17].

• Dado un conjunto de imágenes $(x_1, y_1), ..., (x_n, y_n) donde, y_i = 0, 1$(2.3) para muestras negativos y positivos respectivamente.

• Inicializar los pesos

$$w_{1,i} = \frac{1}{2m}, \frac{1}{2l} \, para \, y_i = 0, 1 \tag{2.4}$$

donde m es el número de muestras negativas y l es el número de muestras positivas.

• Para t=1,...,T:

1. Normalizar los pesos

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \tag{2.5}$$

2. Seleccionar el mejor clasificador base respecto al peso del error:

$$\epsilon_t = min_{f,p,\theta} \sum_i w_i \mid h(x_i, f, p, \theta) - y_i \mid \tag{2.6}$$

3. Definir

$$h_t(x) = h(x, f_t, p_t, \theta_t) \text{ donde } f_t, p_t, \theta_t \tag{2.7}$$

son usadas para minimizar ϵ_t.

4. Actualizar los pesos:

$$w_{t+1,i} = w_{t,i}\ \beta_t^{1-e_i} \text{donde } e_i = 0 \tag{2.8}$$

si la muestra x_i es clasificada correctamente o $e_i = 1$ en otro caso, con

$$\beta_t = \frac{\epsilon_t}{1-\epsilon_t}. \tag{2.9}$$

5. El clafisicador robusto final es:

$$C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geqslant \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{en otro caso} \end{cases} \tag{2.10}$$

donde

$$\alpha_t = \log \frac{1}{\beta_t} \tag{2.11}$$

La figura 2.9 y 2.10 muestran un ejemplo del procedimiento que el algoritmo AdaBoost simple sigue para formar el clasificador final. Se tiene el siguiente conjunto de datos en donde un tipo de datos se representa por símbolos -.ªzules y el otro por símbolos .en rojo. En su primer intento, el clasificador realiza un çorte"tratando de separar los datos pero podemos observar como realiza tres errores: deja dos datos negativos dentro del conjunto positivo y un dato positivo no lo toma en cuenta. En su siguiente intento, el peso de estos tres datos en donde hubo error es incrementado, de modo que el nuevo clasificador le demás importancia a resolver bien estos puntos [18].

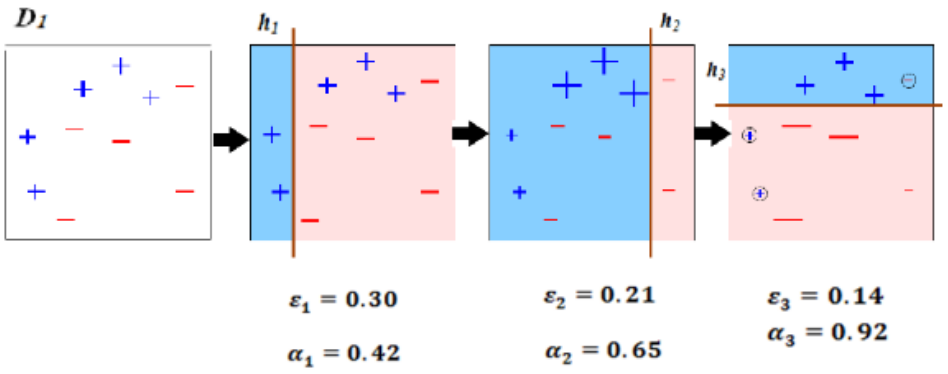


Figura 2.9: Ejemplo de procedimiento de AdaBoost. Fuente:[18]

En su segundo intento logra discriminar correctamente a los datos negativos que tenían más peso pero ahora incluye aquellos que tenían un bajo peso. Debido a esto, se incrementa el peso de estos errores y, por el contrario, aquellos datos que ya fueron correctamente clasificados reciben un peso menor. Se realiza un tercer clasificador que de igual manera tratará de hacer el mejor çorte"para resolver el problema. Finalmente se sumarán todos los clasificadores creados para crear la solución que, teóricamente, debería de ser mejor que cada clasificador por separado [18].

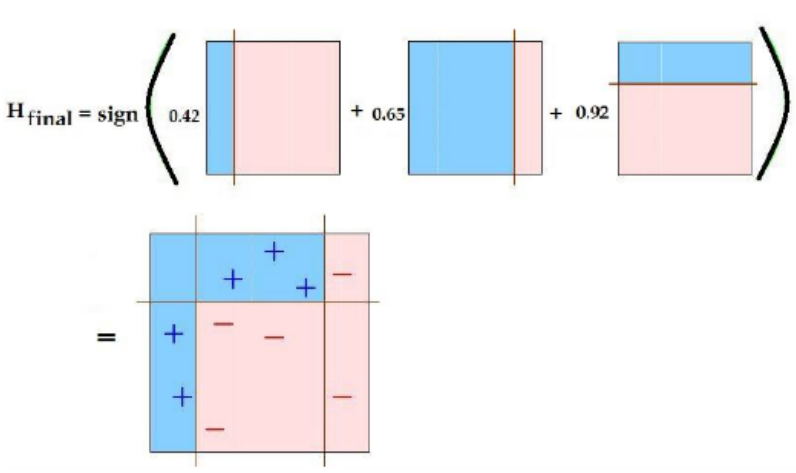


Figura 2.10: Clasificador final. Fuente:[18]

2.3.4. Reconocimiento Facial

El reconocimiento consiste en clasificar las características extraídas de cada rostro. Esta clasificación puede ser realizada de manera supervisada, en la cual un patrón de entrada es identificado como miembro de una clase predefinida, o de manera no supervisada, donde el patrón es asignado a una clase desconocida. Aquí, cada clase es un sujeto, por lo tanto, al clasificar las características se está indicando a qué sujeto pertenecen [4].

2.3.4.1. Patrones Binarios Locales (LBP)

El método de reconocimiento de rostros LBPH asigna etiquetas a cada uno de los pixeles de la imagen tomando en cuenta la distribución de los vecinos [19]. El funcionamiento LBPH que realiza para su respectivo reconocimiento de imágenes es el siguiente:

• Una máscara de tamaño determinado (8x8), recorre la imagen de manera iterativa seleccionando cada vez un pixel central y sus vecinos.

• Este pixel central se compara con cada uno de sus vecinos de forma ordenada. Se asigna un 1 cada vez que el pixel central sea menor que el pixel comparado y un 0 en el caso contrario, como se visualiza en la figura 2.11 y 2.12 y se muestra en las ecuaciones 2.12 y 2.13 las cuales permiten calcular LBP sobre una imagen en escala de grises. Donde P es el número de vecinos que se van a considerar, R es el tamaño del vecindario, gp y gc son los valores de gris del píxel central y cada uno de los p píxeles del vecindario respectivamente.

$$LBP_{P,R} = \sum_{p=0}^{7} s(g_p - g_c)2^p \tag{2.12}$$

$$s(x) = \begin{cases} 1, & si\, x \geqslant 0 \\ 0, & otro\, valor \end{cases} \tag{2.13}$$

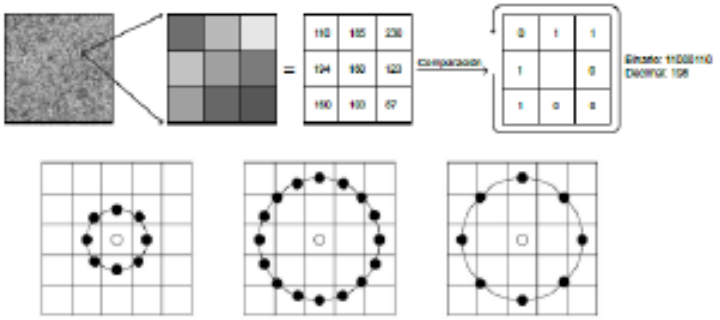


Figura 2.11: Obtención de los Parámetros LBP. Fuente: [20]

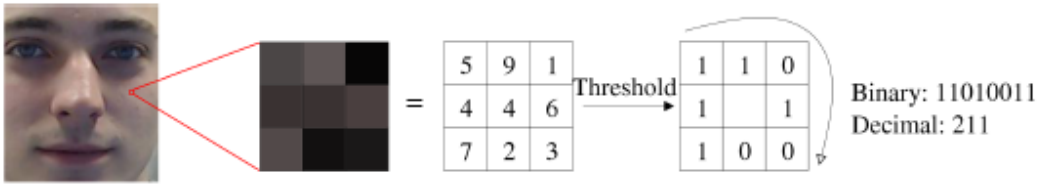


Figura 2.12: Operador LBP de Radio 1. Fuente: [4]

• El número binario resultante se convierte en un número decimal que es contado en el histograma para formar la descripción. El histograma de las etiquetas de todos los pixeles es posteriormente utilizado como una descripción de la textura de la imagen, que se indica en la ecuación 2.14 y 2.15. Donde n es el número de etiquetas diferentes producidos por el operador LBP.

$$H_i = \sum_{xy} I[LBP(x,y) = i],\ i = 0,...,n-1 \tag{2.14}$$

$$I(x) = \begin{cases} 1, & si\ x\ es\ verdadero. \\ 0, & otro\ valor. \end{cases} \tag{2.15}$$

La imagen resultante tiene aspecto de una imagen en blanco y negro que retiene los detalles de la imagen como muestra la figura 2.13.

Figura 2.13: Imágenes Transformadas mediante el Operador LBP. Fuente: [4]

Para la generación de histogramas, la imagen transformada se divide en regiones y se calcula el histograma por separado de cada una. Finalmente, todos los histogramas se concatenan para crear el vector de características como se observa en la figura 2.14. Dicho vector contendrá información sobre los valores de la imagen así como información local debido a la división en regiones. [20]. Mientras más regiones tenga la imagen, mayor sera la cantidad de información, pero mayor será a su vez el vector de características.

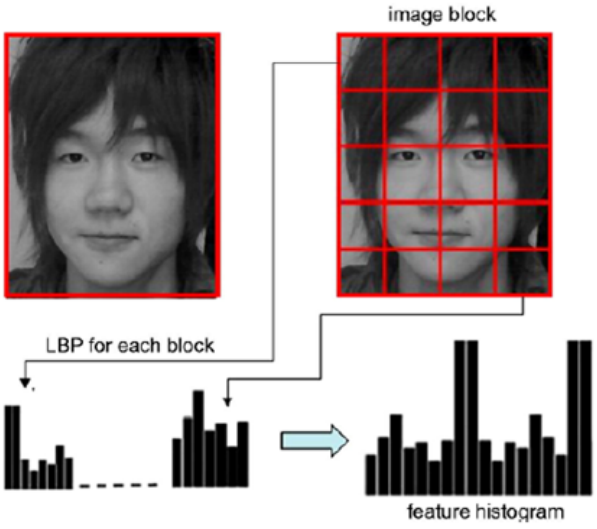


Figura 2.14: Cálculo de los Histogramas LBP de cada región. Fuente: [20]

2.3.4.2. Descripción de Rostros con LBP

El operador LBP es utilizado para describir regiones locales del rostro que después son agrupadas para formar una descripción global. Para formar la descripción global, la imagen del rostro se divide en diferentes regiones, a estas se les aplica el operador LBP consiguiendo descripciones independientes por región. Al final estas descripciones son concatenadas para construir una descripción global [21]. En la figura 2.15 se muestra algunas posibles divisiones.

Figura 2.15: Regiones divididas en un rostro. Fuente:[21]

Luego de que las regiones R_0,R_1,.....R_(m-1) del rostro son determinadas se aplica el operador LBP para calcular un histograma independiente para cada una de las m regiones, los m histogramas resultantes son combinados para crear el histograma mejorado espacialmente. Este histograma tiene un tamaño de mxn, siendo n la longitud de cada uno de los histogramas individuales [21].

A través del histograma mejorado espacialmente, se describe los rostros en tres niveles diferentes de localidad:

- A nivel de píxeles.
- Histogramas de cada región dividida en un rostro.
- Histograma global que es la unión de cada histograma de cada región dividida.

Después de realizar las descripciones de rostros a través del histograma mejorado espacialmente, se establece una medida que permite fijar el nivel de similitud entre el histograma del nuevo rostro, y los histogramas previamente guardados

[21]. Para ello se usa la medida Chi Cuadrado (ecuación 2.16), en donde se pone más énfasis a ciertas regiones del rostro.

$$\chi^2_w(x,\varepsilon) = \sum_{j,i} w_j \frac{(x_{i,j} - \varepsilon_{i,j})^2}{(x_{i,j} + \varepsilon_{i,j})} \tag{2.16}$$

Donde:

- x y epsilon son los histogramas mejorados espacialmente normalizados a ser comparados.
- Los índices i,j hacen referencia a la iénesima posición del histograma correspondiente a la jénesima región local.
- wj es el peso de la región j.

En la figura 2.16 se puede observar los valores que se asigna a las diferentes regiones del rostro. Se puede observar que hay más énfasis en el área de los ojos y la boca.

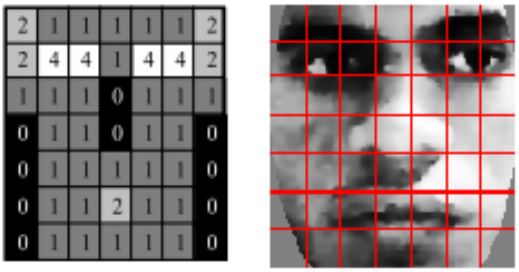


Figura 2.16: Pesos de las diferentes regiones del rostro. Fuente:[21]

Capítulo III

Procedimientos de la Investigación

En el presente trabajo se desarrolló un software piloto de reconocimiento facial para el registro de asistencia. Los objetivos planteados fueron logrados en fases como se muestra a continuación:

3.1. Fase I: Diagnóstico del Registro de Asistencias Actual

Se realizó un diagnóstico del registro de asistencia del personal que labora en la Escuela de Telecomunicaciones de la Universidad de Carabobo, en el que se pudo constatar que para el personal docente, el registro de asistencia es computarizado mientras que para el personal estudiantil como preparadores y becas servicios es de forma manual, a través del llenado de planillas.

Esta fase se culminó a través de la entrevista con el Director de la Escuela, quien nos proporcionó información detallada de cómo funciona actualmente el registro de asistencia, así como permiso para obtener información del personal que labora dentro de ella. A su vez la TIC nos indicó por su parte el proceso que llevan a cabo para el registro de asistencia en la Universidad de Carabobo así como los lineamientos y datos que toman en cuenta en su control tales como cédula, hora de entrada y hora de salida. Dentro de la información que se recopiló en la Escuela

por parte de la secretaria se tiene cédula, nombres, apellidos, fecha de nacimiento, escalafón, y entre otros datos pertenecientes al personal que labora dentro de ella.

Cabe recalcar, que para obtener los permisos de la TIC para realizar la conexión del software piloto con ellos, este debe cumplir con un período de pruebas, una vez finalizado si todo funciona de manera exitosa proceden a otorgar dichos permisos.

3.2. Fase II: Documentación Teórica

En esta etapa, se hizó un arqueo por una variedad de libros digitales, publicaciones, trabajos de grados, entre otros, con la intención de recabar información sobre el procesamiento de imágenes, video y su funcionamiento, en específico, sobre la detección y reconocimiento facial. Con lo cual, se identificaron los parámetros que intervienen en la detección facial es decir, a través de que parámetros el algoritmo Vioja Jones (Haar-Like Feature) se basa para decidir si es un rostro o no. Así como los criterios para el reconocimiento.

Entre los parámetros que intervienen en la detección facial y los criterios para el reconocimiento se tienen:

3.2.1. Características Haar

Todas las caras humanas comparten algunas propiedades similares. Estas regularidades pueden se igualadas mediante características Haar. Entre las propiedades comunes de la cara se tienen:

- La región del ojo es más oscura que las mejillas superiores.

En la figura 3.1 se observa la característica Haar aplicada al rostro en la región de los ojos. Donde el rectángulo blanco representa el área iluminada de las mejillas y el negro representa la región de los ojos.

Figura 3.1: Característica Haar en los ojos. Fuente:[22]

- La región del puente nasal es más brillante que los ojos.

En la figura 3.2 se observa la característica Haar aplicada al rostro en la región de la nariz. Donde el área blanca corresponde al tabique nasal que es las más claro que la región de los ojos.

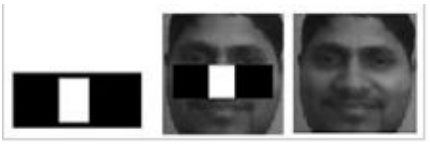


Figura 3.2: Característica Haar en la nariz. Fuente:[22]

3.2.2. Composición de propiedades que forman características faciales comparables

- Localización y tamaño: cejas, ojos, boca, puente de la nariz
- Valor: gradientes orientados de intensidades de píxeles

Las cuatro características que se corresponden con este algoritmo se buscan entonces en la imagen de una cara.

3.2.3. Características del rectángulo

- Valor = (píxeles en el área negra) - (píxeles en área blanca)
- Tres tipos: dos, tres, cuatro rectángulos.
- Por ejemplo: la diferencia de brillo entre los rectángulos blanco y negro sobre un área específica.

• Cada característica está relacionada con una ubicación especial en la subventana.

En la figura 3.3 se observan filtros Haar utilizados en algoritmo Viola Jones.

Figura 3.3: Rectángulos usados en el Viola Jones. Fuente:[22]

Es decir, el algoritmo Viola Jones busca en la imagen o video combinaciones de los patrones Haar. Cuando se desea detectar un rostro el algoritmo busca en la imagen o video la combinación de estos bloques que si se juntan se aproximan a un rostro como se observa en la figura 3.4.

Figura 3.4: Características Haar en el Rostro. Fuente:[23]

En la figura 3.5 se observan las características Haar seleccionadas para las facciones: cejas (características de 2 rectángulos), ojos (características de 2 rectángulos), nariz (características de 3 rectángulos) y boca (características de 4 rectángulos) con los cuales se procede a detectar el rostro de la persona. Cabe acotar, que estos son los más indicados para cada una de estas facciones.

Figura 3.5: Características Haar utilizadas para la detección facial. Fuente:[23]

3.2.4. Criterios para el Reconocimiento Facial

En la configuración (8,n) del operador LBP existen 255 patrones, de los cuales 197 no son patrones uniformes y 58 son uniformes. Esto significa una reducción del histograma del 76.86% por descripción. Con estos parámetros este método (LBP) calcula un histograma de 59 posiciones (58 para patrones uniformes y 1 para patrones no uniformes), en cada una de las 49 regiones divididas.

Luego de calculado los patrones, se procede a realizar un histograma para cada una de las 49 regiones en las que dividido el rostro. Estos histogramas son concatenados para formar la descripción global. En el momento de el cálculo de la medida de disimilitud, se le asigna a cada una de las regiones del rostro un peso correspondiente a la importancia de la información que contienen, de esta forma, las regiones de los ojos y la boca son las que tienen un peso mayor [24]. En la figura 3.6 se observa una representación de los pesos.

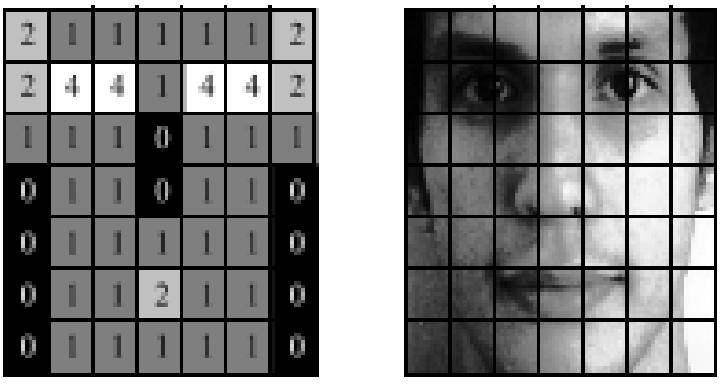


Figura 3.6: Configuración de los pesos. Fuente:[24]

En la figura 3.7 se observa una región específica del rostro con el histograma

de patrones calculado a su derecha para dos imágenes diferentes del rostro de un mismo sujeto.

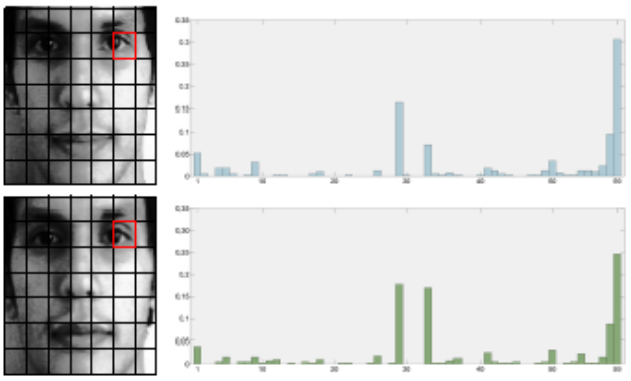


Figura 3.7: Histograma LBP calculado para una región del rostro. Fuente: [24]

Estos histogramas son juntados para tener un vector de 2891 posiciones que describe el rostro y este vector se almacena.

Cuando se va a realizar el reconocimiento se usa el vector almacenado previamente y se calcula un umbral de similitud entre el rostro de la base de datos y el rostro a reconocer. La identificación del rostro estará dada por el que tenga el valor de umbral más bajo.

3.3. Fase III: Composición del Algoritmo usando la Librería de OpenCV

3.3.1. Librerías Utilizadas

Entre las librerías y/o módulos utilizados en la composición del algoritmo de reconocimiento facial se tienen:

3.3.1.1. OpenCV

Librería de visión artificial de código abierto, utilizada para el procesamiento de imágenes, detección de objetos, reconocimiento de patrones y detección facial. Para esta última, utiliza los algoritmos Viola Jones. El cual se basa en: 1) Características rectangulares simples, llamadas Haar-Like Feature. 2) Para una rápida detección de características se utiliza una imagen integral. 3) Método de aprendizaje AdaBoost. 4) Clasificadores en cascada.

Las funciones OpenCV utilizadas en el algoritmo de reconocimiento facial fueron las siguientes:

• **cv2.VideoCapture**: Permite capturar video desde una cámara.

• **cv2.CascadeClassifier**: Carga el archivo cascada utilizado para detectar rostros.

• **detectMultiScale**: Se encarga de detectar objetos a partir del archivo cascade que se ha cargado. Se compone de los siguientes parámetros:

– Filtro: Contiene la imagen transformada a escala de grises.

– Factor de escala (scaleFactor): Determina el tamaño de los rostros a buscar en el video, generalmente se utiliza el valor de 1.1 para una buena detección o 1.2 para una detección más rápida, por ello se uso este valor en la realización del algoritmo.

– Vecinos cercanos (minNeighbors): Determina la fiabilidad del rostro detectado, por lo general es un valor de 3 el cual fue el establecido en el algoritmo realizado. También se puede establecer un valor más alto si se desea un rostro más confiable, pero algunos rostros en el video no podrán ser detectados.

El minNeighbors indica al detector cuantas veces tiene que pasar sobre la imagen para reducir el numero de falsos positivos en la imagen; esto se debe a que el analizador, hace un recorrido de la imagen en distintas posiciones y direcciones, así que de manera horizontal podemos deducir que algo se parece a un rostro, pero en la siguiente pasada aparecería otro objeto que no lo es.

– Banderas: Permite especificar si buscar todos los rostros o solo buscar al más grande. Se le pueden agregar los siguientes banderas y parámetros para aumentar la velocidad de búsqueda:

cv2.CASCADE_SCALE_IMAGE

cv2.cv.CV_HAAR_SCALE_IMAGE

- minsize: Tamaño mínimo del rostro a buscar. Este par de dimensiones en píxeles mejora el rendimiento cuando su valor es alto.

- maxsize: Tamaño máximo en píxeles del rostro a buscar. Este par de dimensiones en píxeles mejora el rendimiento cuando su valor es bajo.

• **cv2.createLBPHFaceRecognizer**: Crea el reconocedor de rostros basado en histogramas de patrones binarios locales.

• **cv2.destroyAllWindows**: Cierra todas las ventanas iniciadas.

• **cv2.cvtColor**: Transforma una imagen de color a escalas grises u otros. Para realizar las diferentes conversiones se utiliza:

cv2.COLOR_BGR2GRAY: Para convertir de RGB a GRAY es decir, de color a escala de grises.

cv2.COLOR_BGR2HSV: Para convertir de RGB a HSV.

• **cv2.rectangle**: Dibuja un rectángulo en una imagen.

• **cv2.imshow**: Muestra una imagen en una ventana específica.

• **cv2.waitKey**: Espera la pulsación de una determinada tecla para ejecutar una acción, o un tiempo determinado por delay en milisegundos.

• **cv2.equalizeHist**: Función que realiza la ecualización del histograma de una imagen de entrada.

• **cv2.putText**: Poner texto dentro de una imagen.

• **cv2.FONT_HERSHEY_SIMPLEX**: Parámetro de cv2.putText, identificador del tipo de fuente la cual es de tamaño normal utilizada en el texto de una imagen.

• **cv2.resize**: Cambia el tamaño de una imagen.

• **cv2.imwrite**: Guarda una imagen en una ruta específica.

3.3.1.2. Os

El módulo Os permite acceder a funcionalidades dependientes del sistema operativo.

3.3.1.3. Numpy

Librería con gran variedad de paquetes para el arreglo de vectores, matrices, etc.

3.3.1.4. PIL

Librería que agrega capacidades de procesamiento de imágenes para al intérprete Python.

3.3.1.5. Sys

Librería encargada de proveer variables y funcionalidades, directamente relacionadas con el intérprete.

3.3.1.6. Shutil

Módulo que consta de funciones para realizar operaciones de alto nivel con archivos y/o directorios.

3.3.1.7. Time

Librería con variables de tiempo.

Las librerías, módulos y funciones mencionados anteriormente son pertenecientes al lenguaje de programación Python utilizado para realización del algoritmo de reconocimiento facial, el cual es un software libre multiplataforma. Éstas deben ser instaladas mediante consola del ordenador que se está utilizando, algunas de estas solo requieren de su descarga e instalación para su funcionamiento.

Teniendo en cuenta la documentación anterior, se realizó un algoritmo en base a la librería de OpenCV que ejecute la detección y reconocimiento facial el cual se observa en la figura 3.15 y 3.16, para ello se siguió el siguiente procedimiento:

3.3.2. Entrenamiento e Inicialización

En el entrenamiento del clasificador Viola Jones llamado Haar-Like Feature, se utiliza una variante del algoritmo AdaBoost (Boosting) para seleccionar un pequeño grupo de características y entrenar el clasificador. Es decir, en la etapa de entrenamiento se aplican solo detección facial y procesamiento de imágenes.

La finalidad de la etapa de entrenamiento es construir un conjunto de imágenes del rostro que se han clasificado como conocidos, para ello primero se debe realizar una recopilación de diferentes capturas del rostro de la misma persona, ya que mientras más rostros se capturen mejor funcionará el algoritmo.

Para realizar la capturas del rostro para la base de datos se utilizan las siguientes funciones: cv2.VideoCapture(0), cv2.VideoCapture(0).read(), detectMultiScale() y cv2.CascadeClassifier("haarcascade_frontalface_default.xml") para activar la cámara y capturar el rostro de la persona, posteriormente estas imágenes se guardaran en la base de datos con la ayuda de cv2.imwrite().

En la figura 3.8 se observa una base de datos correspondiente a dicha etapa la cual es la encargada de permitir el ingreso de las imágenes con las que se realizó el

entrenamiento del sistema. Todas estas imágenes pertenecen a rostros que se desean identificar en el video.

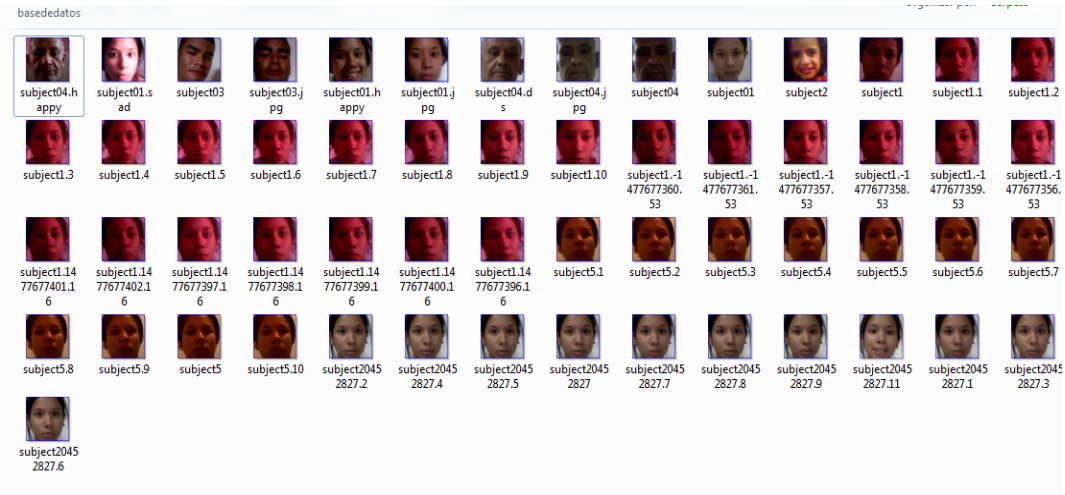


Figura 3.8: Base de Datos para un Conjunto de Imágenes de Rostros a Reconocer

Sobre cada una de estas imágenes ingresadas, el sistema realiza automáticamente un procesamiento en base al algoritmo de detección de rostro Haar Cascade, el cual genera un grupo final de imágenes normalizadas en tamaño y color (solo niveles grises) a través de las funciones cv2.resize() y cv2.cvtCOLOR() como se observa en la figura 3.9.

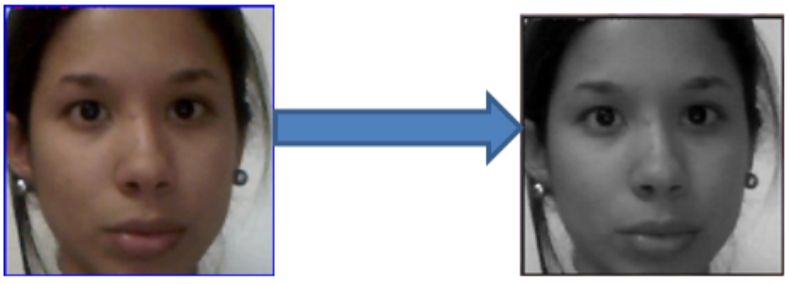


Figura 3.9: Imagen de Entrenamiento Preprocesada y Normalizada

El algoritmo inicia con una captura de video para ello, se utiliza la función vc=cv2.VideoCapture(0) la cual es propia de OpenCV, se debe colocar el número de dispositivo que se desea abrir 0 si se encuentra un sola cámara conectada, 1 si

se quiere hacer referencia a una segunda cámara conectada y así sucesivamente. Con ret, frame = vc.read() se captura y lee fotograma por fotograma, en el cual se buscará detectar la cara de la persona que se encuentra en frente como se observa en el figura 3.10.

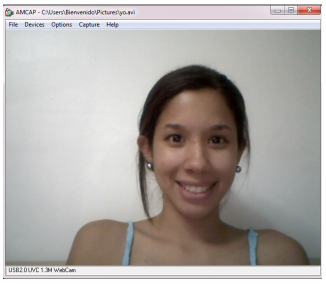


Figura 3.10: Ventana del Video Capture con cual inicializa el Algoritmo de Reconocimiento Facial.

3.3.3. Detección de Cara

Esto se realiza a través del algoritmo Viola Jones, para detectar si en el video se encuentra o no una cara, dicho algoritmo divide la imagen integral en subregiones de tamaños diferentes y utiliza una serie de clasificadores en cascada, cada una con un conjunto de características visuales.

En la detección de cara se utilizan los clasificadores en cascada que trabajan con características del algoritmo Haar-Like Feature el cual es llamado haarcascade_frontalface_default.xml en OpenCV, el cual permite identificar los rasgos en un video.

Después de que se realiza el procesamiento del video y la inicialización de los clasificadores, se puede detectar los rasgos en el video, para ello se utiliza la función detectMultiScale(image,scaleFactor,minNeighbors,flags,minSize) donde se configuraron los siguientes parámetros:

- image = gray.
- scaleFactor = 1.2
- minNeighbors = 3.

– flags = cv2.CASCADE_SCALE_IMAGE

– minSize= 150x150 pixeles.

Debido a que el área de interés es el rostro se identifican subregiones correspondientes con los rasgos presentes en la cara como son los ojos, las cejas, la nariz y la boca.

En la figura 3.11 se observa el rostro detectado en el video. Para dibujar el rectángulo sobre el objeto detectado (rostro) se utiliza cv2.rectangle() al cual se le configuran parámetros como las coordenadas del rostro y color del rectángulo a dibujar ya sea (255,0,0) para azul o (0, 255, 0) para verde.

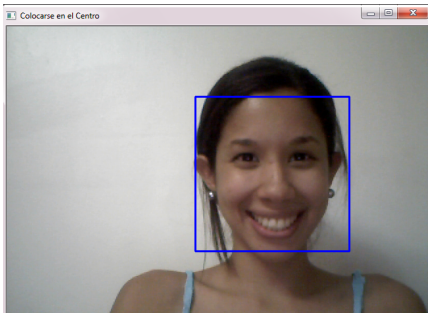


Figura 3.11: Detección de la cara en un video capture

3.3.4. Pre-Procesado

La segmentación de las zonas anteriormente mencionadas se realiza a través del algoritmo de Viola Jones debido a que este sólo procesa la información presente en una imagen en escala de grises, la imagen que se utiliza no será directamente la imagen capturada sino una representación de la imagen, llamada imagen integral. Es decir, en esta etapa se realiza el escalado y recorte de la imagen capturada a través de la captura de video. Se recortan las partes exteriores de la cara tales como: el cabello, la frente, las orejas y la barbilla o mentón. Debido a que se realiza un escalado no importa el tamaño del rostro detectado, es decir, si está más cerca o lejos de la cámara. El procesamiento del rostro consta de las etapas que se observan en la figura 3.12.

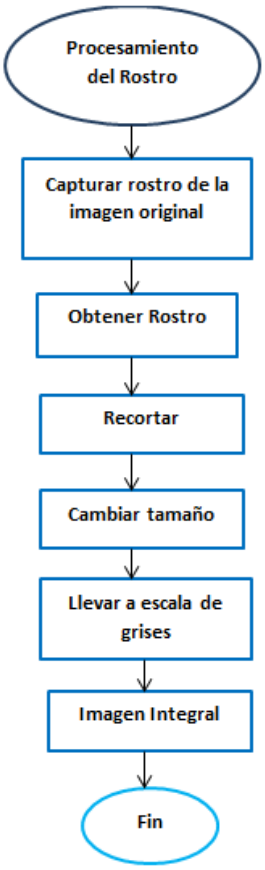


Figura 3.12: Esquema del procesamiento del rostro. Fuentes: Los Autores.

En la figura 3.13 se observa la imagen integral que luego pasará a la etapa de extracción de características para su posterior reconocimiento.

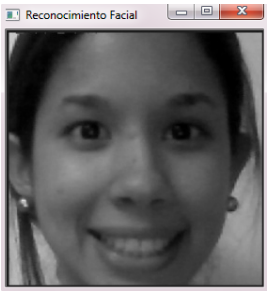


Figura 3.13: Imagen integral preprocesada para su posterior Reconocimiento.

Para realizar el procesamiento del rostro se utilizan las funciones de OpenCV tales como: cv2.resize(frame[y:y+h,x:x+w], (273, 273)) donde el primer parámetro

permite recortar la imagen original y el segundo cambiar el tamaño. También se utilizó cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) para llevar a escalas de grises la imagen original y así obtener la imagen integral.

3.3.5. Extracción de Características

Esta etapa se encarga principalmente de seleccionar los valores que realmente dan información de cara al reconocimiento y desechar aquellos que no aportan información relevante o incluso desechar también información que pueda provocar que sea difícil el reconocimiento facial entre el personal.

La técnica que se utilizó para la extracción de características fue la de histogramas de patrones binarios locales (LBPH); ésta destaca por su rapidez y simplicidad computacional, lo que permite analizar las imágenes en tiempo real.

El algoritmo de reconocimiento facial LBP (patrones locales binarios) cuenta con una clase llamada FaceRecognizer en OpenCV para su implementación la cual es llamada, cv2.createLBPHFaceRecognizer. El algoritmo LBPH tiene parámetros como el radio, números de vecinos, rejillas tanto verticales y horizontales, y el umbral, estos son configurados de acuerdo a la necesidad del usuario.

model = cv2.createLBPHFaceRecognizer(radio,vecinos,gridX,gridY,umbral)

Si no existe ninguna configuración el algoritmo trabaja con los siguientes valores por defecto: 1,8,8,8,100 para el radio, números de vecinos, rejilla para x, rejilla para y, y umbral.

Entre las funciones con que cuenta Face Recognizer que fueron utilizadas para la realización de software piloto se tienen:

• Train: la cual permite obtener un archivo donde se encuentran los valores de los histogramas de las imágenes de la base de datos.

• Predict: la cual permite comparar el histograma de una nueva imagen con el archivo de los histogramas almacenado, indica cual es la imagen reconocida y

entrega el valor de confianza de la predicción. Es decir, esta función compara los rostros que han sido captados en el entrenamiento con el rostro a reconocer.

Mientras más alto es el valor de la variable confianza , menos el reconocedor tiene confianza en el reconocimiento.

• Umbral: Permite establecer un valor base de predicción el cual fue establecido de 100.

3.3.6. Reconocimiento

Luego de que se tenga la imagen del rostro, se hará una comparación con las imágenes del personal almacenadas en una base de datos. En la figura 3.14 se observa la respuesta del algoritmo de reconocimiento tanto para una cara reconocida como una no reconocida.

```
>>> ================================ RESTART ================================
>>>
Introduzca su numero de identificacion.
 id:21136637
 Presione "s" cuando este listo en el centro
foto
Es incorrectamente reconocido
>>> ================================ RESTART ================================
>>>
Introduzca su numero de identificacion.
 id:20452827
 Presione "s" cuando este listo en el centro
foto
Fecha y hora: 11/08/16 19:29:16
Es correctamente reconocido
```

Figura 3.14: Respuesta del Algoritmo de Reconocimiento Facial

El software toma la decisión, es decir, si encontró una similitud en la base de datos se pasará a la siguiente etapa que consiste en registrar correctamente la asistencia. En la figura 3.15 se puede observar el diagrama de clases del algoritmo de reconocimiento facial y en la figura 3.16 su esquema general.

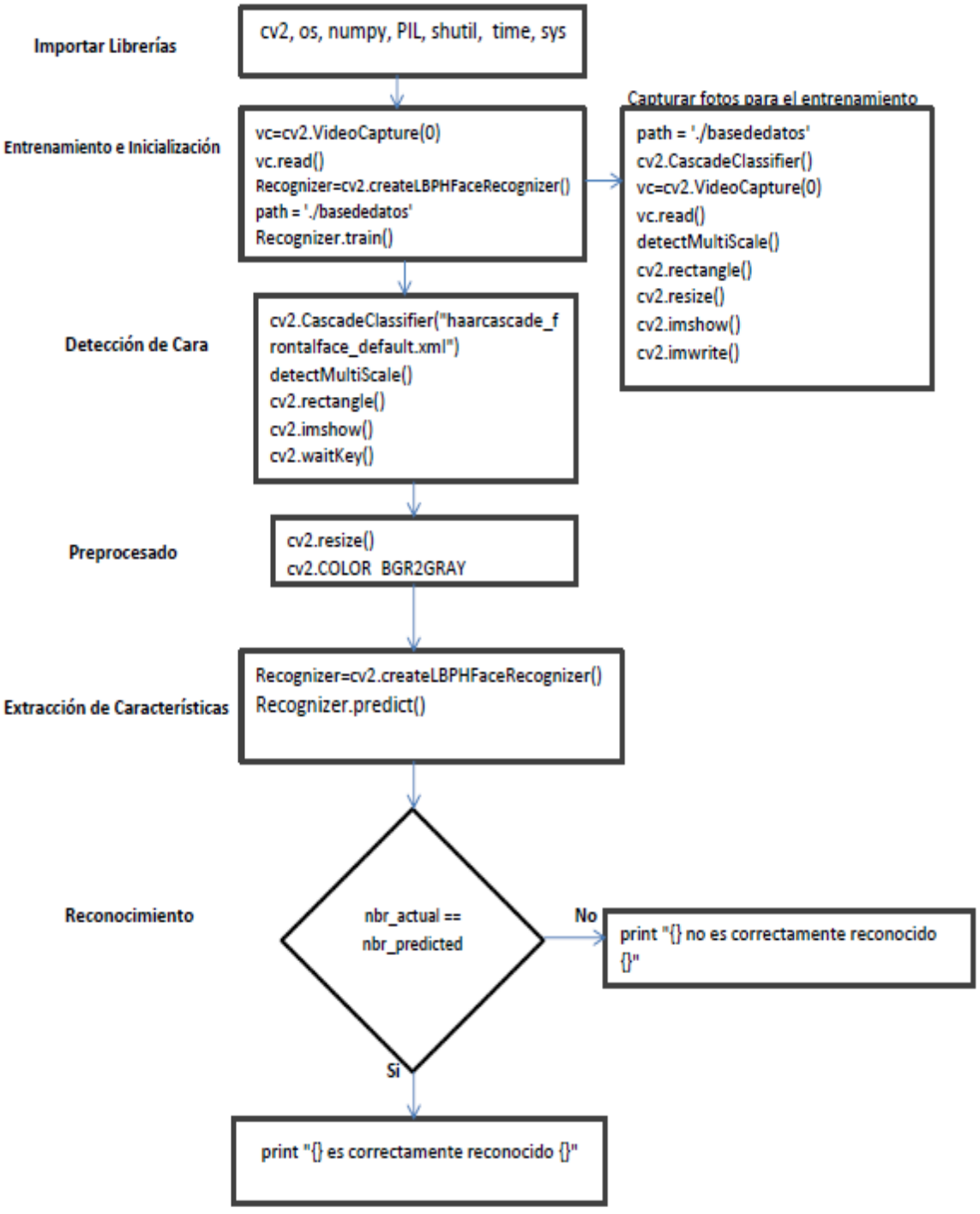


Figura 3.15: Diagrama de clases del Algoritmo de Reconocimiento Facial. Fuentes: Los Autores.

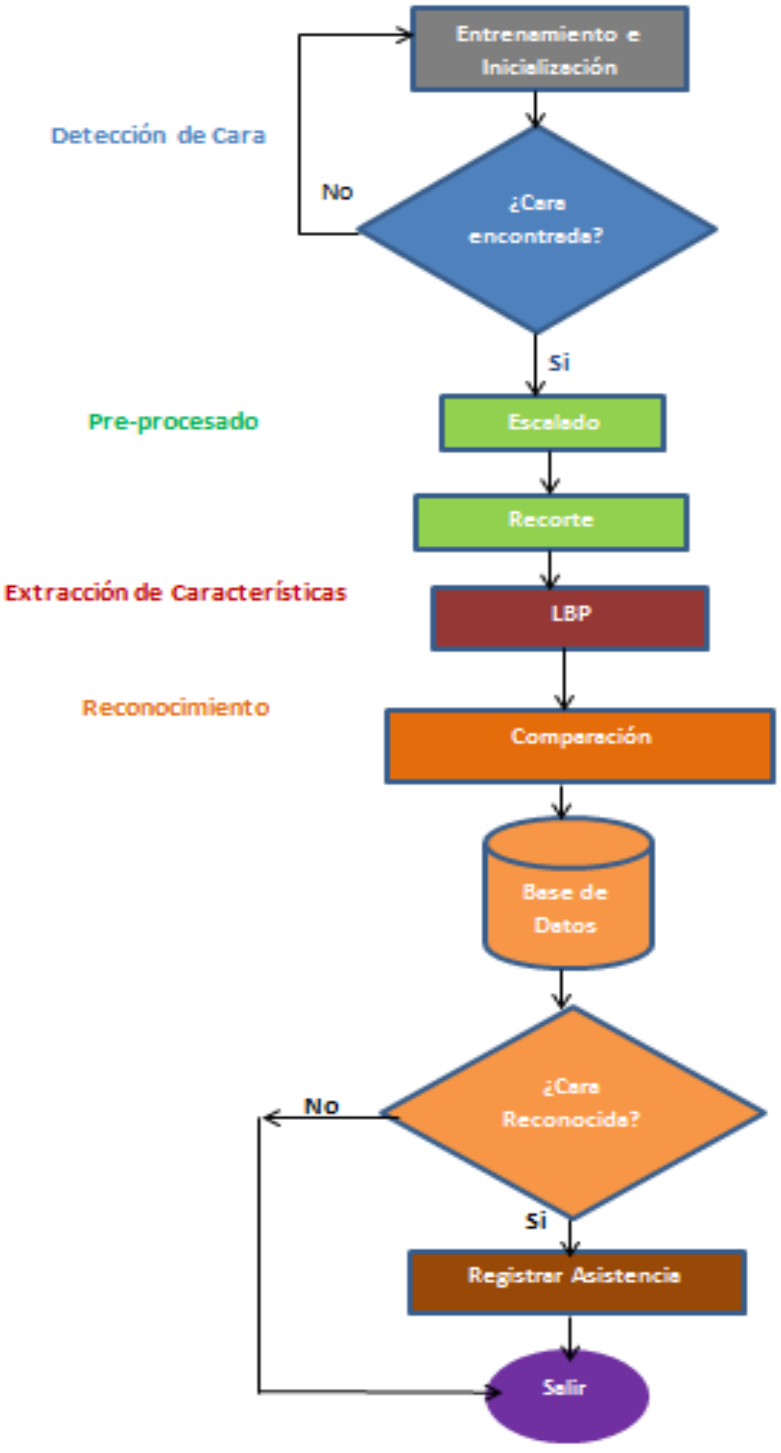


Figura 3.16: Esquema General del Algoritmo de Reconocimiento Facial para el Registro de Asistencia. Fuente: Los Autores.

3.4. Fase IV: Integración del Algoritmo de Reconocimiento Facial con el Registro de Asistencia

Luego de haber obtenido el algoritmo de reconocimiento facial, el mismo fue integrado al sistema de control de asistencia, donde después de haber autentificado correctamente a la persona, automáticamente se registra la asistencia.

Para el registro del personal y de asistencia se usó como herramienta un sistema de gestión de base de datos relacional, entre los diferentes lenguajes se decidió trabajar con el lenguaje PostgreSQL, debido a que es el gestor de bases de datos usados por TIC, además de ser un lenguaje de código abierto, lo que implica que su uso puede ser usado en cualquier instancia. Bajo el mismo contexto, se hizo el diseño de tres tablas relacionadas entre sí, basados en el esquema obtenido por TIC. Ver en el apéndice H

Entre los atributos que posee PostgreSQL y que fueron utilizados, se encuentran los siguientes:

insert: permite ingresar datos dentro de las tablas creadas.

select: permite recuperar datos dentro de la base de datos.

update: actualiza los registros dentro de las tablas.

Es importante señalar, que PostgresSQl es un lenguaje independiente, por tanto no está directamente relacionado con el entorno de programación Python, por lo que es imperativo utilizar una librería externa que sea capaz de conectar ambos lenguajes, para ello se trabajó con psycopg2, cuyo propósito es realizar la conexión a la base de datos, además de proporcionar un traductor para que el intérprete de Python pueda leer correctamente los datos.

```
cadenaConexion="host='localhost' dbname='tubasededatos' user='postgres' password='tucontraseña'"
```

Figura 3.17: Conexión entre Python y PostgreSQL a través de psycopg2.

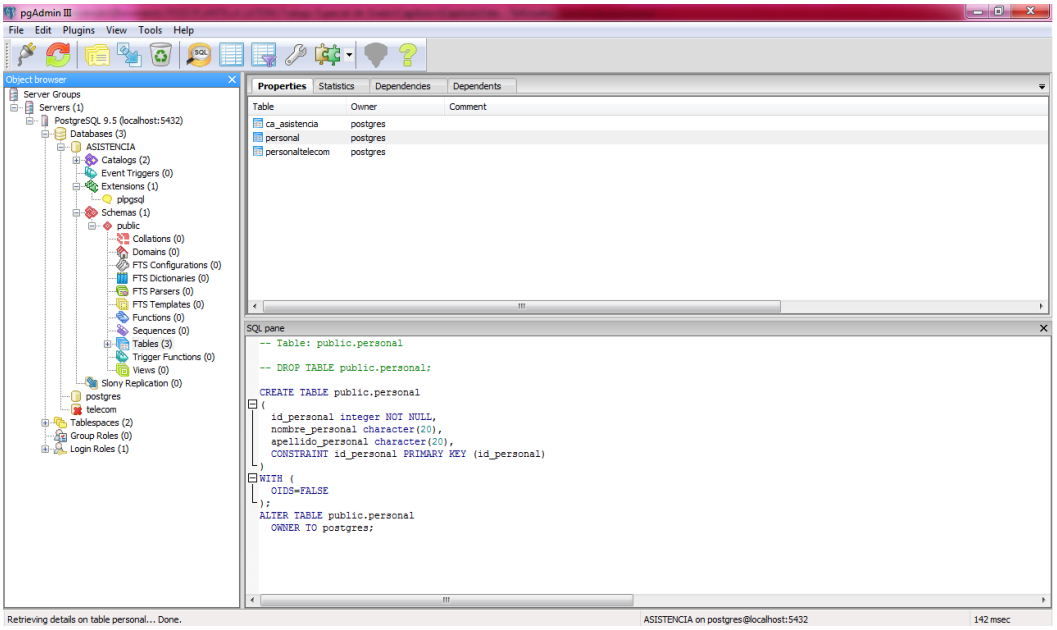


Figura 3.18: PostgreSQL. Herramienta utilizada para la realización de las bases de datos correspondientes al registro del personal y asistencia.

Para registrar la asistencia, sólo se debe ingresar el número de cédula dentro de la interfaz de usuario, posteriormente, se inicia automáticamente el programa de captura y comparación de imágenes. En caso de ser correcto el reconocimiento facial, la asistencia es enviada a la base de datos con el comando insert, el cual toma la cédula antes mencionada e inserta datos específicos como el nombre y apellido a quien pertenece la cédula de identidad, además de la hora de entrada y salida según apliquen las circunstancias e inmediatamente notifica al usuario que fue exitoso su registro, de lo contrario si no es perfectamente reconocido, aparecerá un mensaje señalando que no se pudo realizar el proceso de registro.

Por otro lado, para evitar la dependencia de la conexión a Internet para el registro de asistencia, se configuró el manejador de bases de datos PostgreSQL como servidor local, el mismo puede ser administrado de forma remota por medio de una red de área local. Además de evitar la dependencia a Internet, se estableció un sistema de respaldo para los datos almacenados en la misma y se logra también la administración remota, con lo cual se podrá en un principio generar las datas de asistencia, envió de notificaciones y registro de asistencias manual.

Dando pie a un sistema de control de asistencia escalable con el tiempo, debido a que, con solo disponer de la cámara web y un computador se puede realizar la instalación del software piloto de reconocimiento facial completo. Ver detalles del acceso remoto en el apéndice K.

3.5. Fase V: Desarrollo de una Interfaz Gráfica de Usuario

Utilizando el entorno de programación Python, se realizó una interfaz gráfica de usuario que permite al personal tener interacción directa de forma sencilla y práctica, por medio de la misma se podrá hacer el registro automatizado de la asistencia, ver información concerniente a su registro de asistencia, además de admitir la modificación del mismo, es decir, se podrá anexar, quitar o modificar datos del personal.

En la realización de las interfaces gráficas se utilizó QtDesigner esta herramienta permite que el diseño de la interfaz sea más rápida y sencilla, facilitando así la tarea de construir ventanas arrastrando botones, cuadros de textos, etiquetas, entre otros.

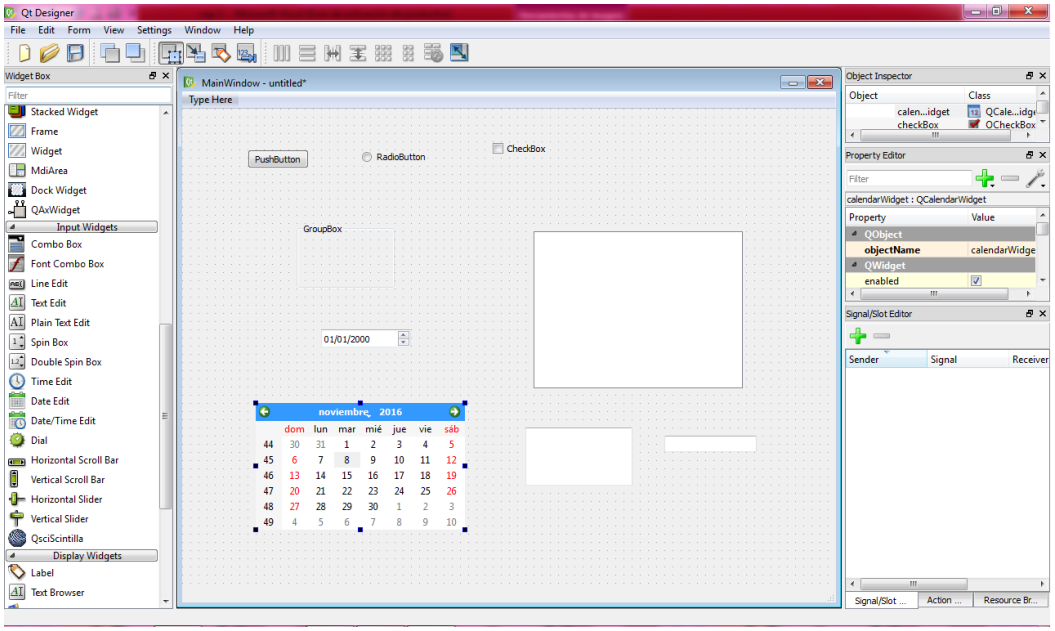


Figura 3.19: Qt Designer. Herramienta utilizada para la realización de las interfaces.

Los archivos creados QtDesigner se guardan con la extensión (.ui), para su posterior uso en Python debe realizarse su respectiva conversión a un archivo (.py) utilizando la herramienta pyuic, la cual se distribuye junto a PyQt. Con estos archivos solo se obtiene el diseño de la ventana por lo que se debe escribir un nuevo código (.py) en el cual se le atribuyen las funciones a los elementos que posee cada ventana.

La interfaz gráfica realizada se basó en una ventana principal que cuenta con el módulo de administración. Éste a su vez cuenta con las opciones: agregar personal, modificar personal, asistencia manual, generar data, notificaciones, manual de usuario y administrador, salir. Para observar con más detalles las interfaces anteriormente mencionadas ver el apéndice I y J.

3.6. Fase VI: Evaluar el Desempeño del Software

Con el fin de evaluar la eficiencia y eficacia del software piloto de reconocimiento facial para el control de asistencia se procedió a realizar una serie de pruebas de reconocimiento a distintas poblaciones en diferentes escenarios, con el fin de detectar posibles fallas que se puedan generar, para que estás puedan ser solventadas antes de su posterior instalación en la Escuela de Telecomunicaciones.

Para finalizar está fase se instaló en la Escuela de Telecomunicaciones el software piloto de reconocimiento facial para el control de asistencia con su respectiva cámara web, donde se registró al personal que allí labora.

Capítulo IV

Análisis, Interpretación y Presentación de los Resultados

4.1. Pruebas Realizadas

Las pruebas que se realizaron para evaluar el desempeño del software piloto de reconocimiento facial para el control de asistencia en la Escuela de Telecomunicaciones en cuanto a la eficiencia, eficacia y funcionamiento fueron las siguientes:

4.1.1. Prueba 1: Eficacia del reconocimiento facial con respecto al número de imágenes de entrenamiento

Se realizaron pruebas para el rostro de una persona considerando un número de imágenes de entrenamiento (fotos en la base de datos) correspondiente a 1, 2, 3, 4, 5, 6, 7, 8, y 9. Por otro lado, el número de muestras de la cantidad de veces que se capturó el rostro de dicha persona para su posterior comparación y reconocimiento fue de veinte.

En la siguiente tabla 4.1 se observan las 20 pruebas realizadas de reconocimiento para las cantidades mencionadas anteriormente de fotos en la base de datos, «Si» indica que fue reconocido y «No» que no fue reconocido el rostro.

Tabla 4.1: Reconocimiento para las 20 muestras.

Muestra	R.1	R.2	R.3	R.4	R.5	R.6	R.7	R.8	R.9
1	Si	Si	Si	Si	Si	Si	Si	Si	Si
2	Si	No	Si	Si	Si	Si	Si	Si	Si
3	No	Si	No	Si	Si	Si	Si	Si	Si
4	Si	Si	Si	Si	Si	Si	Si	Si	Si
5	Si	No	No	Si	Si	Si	Si	Si	Si
6	No	Si	Si	Si	Si	Si	Si	Si	Si
7	Si	Si	No	Si	Si	Si	Si	Si	Si
8	No	No	Si	Si	Si	Si	Si	Si	Si
9	No	Si	Si	Si	Si	Si	Si	Si	Si
10	Si	Si	Si	No	Si	Si	Si	Si	Si
11	No	No	No	Si	Si	Si	Si	Si	Si
12	No	Si	Si	Si	No	Si	Si	Si	Si
13	No	Si	Si	Si	Si	Si	Si	Si	Si
14	No	No	No	Si	Si	Si	Si	Si	Si
15	No	Si	Si	Si	Si	Si	Si	Si	Si
16	No	No	Si	No	Si	Si	Si	Si	Si
17	No	Si	Si	Si	Si	Si	Si	Si	Si
18	Si	No	Si	Si	Si	Si	Si	Si	Si
19	Si	Si	Si	Si	Si	Si	Si	Si	Si
20	Si	No	Si	Si	Si	Si	Si	Si	Si

Tabla 4.2: Eficacia del reconocimiento con diferentes número de imágenes de entrenamiento.

N° de Imágenes	Fallas	Aciertos	Porcentaje Fallas ٪	Porcentaje Aciertos ٪
1	11	9	55	45
2	8	12	40	60
3	5	15	33.33	66.67
4	2	18	10	90
5	1	19	5	95
6	0	20	0	100
7	0	20	0	100
8	0	20	0	100
9	0	20	0	100

En la tabla 4.2 se puede observar que a partir de 6 imágenes de entrenamiento se tiene un porcentaje de aciertos en el reconocedor del 100 ٪.Con estos resultados se fijaron la cantidad de imágenes que tiene la base de datos con la cual cuenta el programa.

Por ello, es necesario tener como mínimo 6 imágenes por cada individuo en la base de datos para un mejor entrenamiento del algoritmo y en consecuencia un eficaz reconocimiento.

4.1.2. Prueba 2: Eficacia del reconocimiento con respecto al número de muestras tomadas a una persona de sexo femenino variando su aspecto facial

Para probar el desempeño del reconocimiento facial se procedió a variar el aspecto de facial de una mujer en las siguientes situaciones: con maquillaje, sin maquillaje, peinada, despeinada, sonriendo, triste, seria, lengua afuera, con muecas, lentes de leer y de sol, boca cerrada, ojos cerrados y bufanda en la boca. El aspecto que tenía el rostro registrado en las imágenes de entrenamiento con la cual se hará la comparación de todos estos casos era peinada, ojos abiertos, seria, sin lentes y maquillada.

Tabla 4.3: Reconocimiento para una persona de sexo femenino.

Muestra	Reconocimiento
Maquillada	Si
Sin maquillaje	Si
Peinada	Si
Despeinada	Si
Sonriendo	Si
Triste	Si
Seria	Si
Lengua afuera	Si
Muecas	Si
Lentes de leer	Si
Lentes de sol	No
Boca cerrada	Si
Ojos cerrados	Si
Bufanda en la boca	Si
Bufanda en la boca y nariz	No

En la tabla 4.3 se puede apreciar que el reconocedor funciona de manera exitosa para todos los casos excepto para el caso en el cual con la bufanda se impide al

programa realizar la segmentación de la boca y la nariz de la persona ya que con solo los ojos no se pudo reconocer a la persona.

Así como también para el caso de los lentes de sol el reconocimiento fue incorrecto.

De estos resultados se puede decir que el reconocedor facial funciona de manera eficaz cuando realiza una segmentación de varias facciones de la persona.

4.1.3. Prueba 3: Eficacia del reconocimiento con respecto al número de muestras tomadas a una persona de sexo masculino variando su aspecto facial

Similar a la prueba anterior se realizó está, se varió el aspecto facial de un hombre en las siguientes situaciones: con gorra, sonriendo, triste, serio, lengua afuera, con muecas, lentes de leer y de sol, boca cerrada, ojos cerrados, con barba y sin barba. El aspecto que tenía el rostro registrado en las imágenes de entrenamiento con la cual se hará la comparación de todos estos casos era sin gorra, ojos abiertos, sonriendo, sin lentes y con barba.

Tabla 4.4: Reconocimiento para una persona de sexo masculino.

Muestra	Reconocimiento
Con gorra	Si
Sonriendo	Si
Triste	Si
Serio	Si
Lengua afuera	Si
Muecas	Si
Lentes de leer	Si
Lentes de sol	No
Boca cerrada	Si
Ojos cerrados	Si
Con barba	Si
Sin barba	Si

En la tabla 4.4 se puede apreciar que el reconocedor funciona de manera exitosa para todos los 11 casos (con gorra, sonriendo, triste, serio, lengua afuera, muecas,

lentes de leer, boca cerrada, ojos cerrados, con barba, sin barba) excepto para el caso en el cual el sujeto porta los lentes de sol ya que con tan sólo la boca y nariz el programa no pudo reconocer a la persona.

Con estos resultados se puede decir que el reconocimiento fue correcto en la mayoría de los casos.

Por ello, al momento de realizar la ejecución del programa de reconocimiento facial se le debe permitir a éste que realice una segmentación de todas las facciones de la persona. De manera tal, que se deben evitar posibles obstrucciones en el rostro de la persona a reconocer.

4.1.4. Prueba 4: Eficacia del reconocimiento con respecto a la luminosidad

Se realizaron pruebas de reconocimiento con respecto a la iluminación la cual es medida en Lux; para ello las imágenes de entrenamiento fueron tomadas en alta luminosidad (500 a 1000 Lux) mientras que las imágenes que se desean reconocer fueron tomadas en diferentes ambientes de luminosidad en un rango desde 20 a 2000 Lux.

En la figura 4.1 se observan las imágenes de las 2 personas utilizadas para realizar el reconocimiento en muy alta luminosidad. Es decir, niveles de luminosidad comprendidos entre 1000 a 2000 Lux.

Figura 4.1: Rostros en muy alta luminosidad.

En la figura 4.2 se observan las imágenes de las 2 personas utilizadas para realizar el reconocimiento en alta luminosidad. Es decir, niveles de luminosidad comprendidos entre 500 a 1000 Lux.

Figura 4.2: Rostros en alta luminosidad

En la figura 4.3 se observan las imágenes de las 2 personas utilizadas para realizar el reconocimiento en luminosidad media. Es decir, niveles de luminosidad comprendidos entre 200 a 500 Lux.

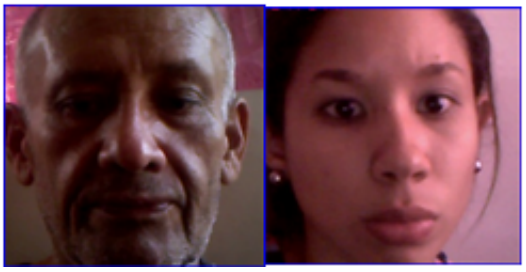


Figura 4.3: Rostros en luminosidad media

En la figura 4.4 se observan las imágenes de las 4 personas utilizadas para realizar el reconocimiento en baja luminosidad. Es decir, niveles de luminosidad comprendidos entre 100 a 200 Lux.

Figura 4.4: Rostros en baja luminosidad.

En la figura 4.5 se observan las imágenes de las 2 personas utilizadas para realizar el reconocimiento en muy baja luminosidad. Es decir, niveles de luminosidad comprendidos entre 20 a 50 Lux.

Figura 4.5: Rostros en muy baja luminosidad

Tabla 4.5: Reconocimiento con variaciones de luminosidad.

Luminosidad	Reconocimiento
Muy alta (1001 Lux)	Si
Muy alta (1020 Lux)	Si
Alta (702 Lux)	Si
Alta (689 Lux)	Si
Media (405 Lux)	Si
Media (310 Lux)	Si
Baja (101 Lux)	Si
Baja (130 Lux)	No
Baja (156 Lux)	No
Baja (178 Lux)	No
Muy baja (32 Lux)	No
Muy baja (25 Lux)	No

En la tabla 4.5 se observa que en ambientes con luminosidad muy alta, alta y media (200 a 2000 Lux) el programa reconoce a la persona mientras que en los ambientes con baja y muy baja luminosidad (20 a 200 Lux) el reconocimiento ocurre de manera incorrecta, esto es debido a que la imagen que se procesa con la captura de estas imágenes no permite segmentar con claridad las facciones de la persona.

Con los resultados obtenidos se puede decir que el software piloto de reconocimiento facial se ve afectado a los cambios de luminosidad sobre todo en ambientes

con baja y muy baja luminosidad. Por ello, es conveniente realizar el reconocimiento facial en ambientes con luminosidad muy alta, alta y media es decir, niveles de luminosidad comprendidos entre 200 a 2000 Lux.

4.1.5. Prueba 5: Eficacia del reconocimiento facial respecto a la distancia

Se realizaron pruebas colocando a 9 personas a diferentes distancias de la cámara para evaluar la capacidad para reconocer el rostro.

En la tabla 4.6 se observan los resultados obtenidos en el reconocimiento facial para el rango distancia comprendido desde 30 a 140 centímetros.

Tabla 4.6: Reconocimiento para diferentes distancias.

Muestra	30-40cm	50-60cm	70-80cm	90-100cm	110-120cm	130-140cm
1	Si	Si	Si	Si	Si	No
2	Si	Si	Si	Si	Si	No
3	Si	Si	Si	Si	No	No
4	Si	Si	Si	Si	Si	No
5	Si	Si	Si	No	No	Si
6	Si	Si	Si	Si	No	Si
7	Si	Si	Si	Si	Si	No
8	Si	Si	Si	Si	No	Si
9	Si	Si	Si	Si	Si	No

En la tabla 4.7 se observan los resultados obtenidos en el reconocimiento facial para el rango distancia comprendido desde 150 a 200 centímetros.

Tabla 4.7: Reconocimiento para diferentes distancias.

Muestra	150-160cm	170-180cm	190-200cm
1	No	No	No
2	No	No	No
3	No	No	No
4	Si	No	No
5	No	No	No
6	No	No	No
7	No	No	No
8	No	No	No
9	No	No	No

Tabla 4.8: Eficacia del reconocimiento con la distancia.

Distancia	Aciertos	Fallas	Porcentaje Aciertos%	Porcentaje Fallas%
30-40cm	9	0	100	0
50-60cm	9	0	100	0
70-80cm	9	0	100	0
90-100cm	8	1	88.89	11.11
110-120cm	5	4	55.56	44.44
130-140cm	3	6	33.33	66.67
150-160cm	1	8	11.11	88.89
170-180cm	0	9	0	100
190-200cm	0	9	0	100

En la tabla 4.8 se observa que para distancias comprendidas entre 30 a 80 cm no se presentaron fallas obteniéndose un porcentaje de aciertos del 100%.

Con estos resultados se puede decir que la distancia ideal para poder detectar y reconocer el rostro esta entre 30 a 90 cm porque en este rango de distancia es donde mejor se detecta el rostro permitiendo tener una captura con mayores detalles necesarios para el procesamiento y reconocimiento de la imagen.

Al momento de la captura de la imagen la persona debe colocarse preferiblemente a una distancia de 30cm a 90cm de la cámara cuando vaya a realizar el reconocimiento.

Para evitar que se detecten rostros que se encuentren atrás o de fondo a la persona que registrará su asistencia se estableció un tamaño mínimo y máximo del rostro a buscar, debido a esto los rostros que se encuentren a distancias mayores de 95 centímetros no serán detectados.

4.1.6. Prueba 6: Eficacia del reconocimiento con respecto a posiciones del rostro

Se realizaron pruebas de reconocimiento variando la rotación del rostro de tres persona como se observa en la figura 4.6. En ellas se observa que para los casos donde la cara se encuentra de lado a la izquierda y derecha así como hacia abajo e inclinada no ocurre una detección del rostro.

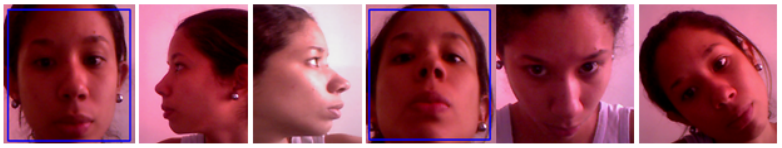


Figura 4.6: Posiciones del rostro utilizados para la prueba 7.

Tabla 4.9: Reconocimiento variando posiciones del rostro para la persona 1.

Posición o Rotación del Rostro	Reconocimiento
Frontal	Si
90° A la izquierda	No
90° A la derecha	No
Hacia arriba	Si
Hacia abajo	No
Inclinada	No

En la tabla 4.9 se observa que el reconocimiento ocurrió de forma correcta en los casos donde el rostro se encontraba frontal y hacia arriba.

Tabla 4.10: Reconocimiento variando posiciones del rostro para la persona 2.

Posición o Rotación del Rostro	Reconocimiento
Frontal	Si
90° A la izquierda	No
90° A la derecha	No
Hacia arriba	No
Hacia abajo	No
Inclinada	No

En la tabla 4.10 y 4.11 se observa que el reconocimiento ocurrió de forma correcta solo en el caso donde el rostro se encontraba frontal, en comparación con la persona 1 no ocurrió reconocimiento en el caso del rostro hacia arriba ya que la detección no fue exitosa.

Tabla 4.11: Reconocimiento variando posiciones del rostro para la persona 3.

Posición o Rotación del Rostro	Reconocimiento
Frontal	Si
90° A la izquierda	No
90° A la derecha	No
Hacia arriba	No
Hacia abajo	No
Inclinada	No

Con estos resultados se puede decir que al tener el rostro de la persona en una posición frontal el reconocimiento ocurre de manera correcta ya que el programa esta diseñado para detectar rostros que están de frente a la cámara.

4.1.7. Prueba 7: Eficacia del reconocimiento con respecto al período de tiempo de las imágenes de entrenamiento

Se le realizaron pruebas de reconocimiento a seis personas cuyas imágenes de entrenamiento fueron tomadas hace seis meses con la finalidad de observar si se presentaban fallas en el reconocimiento en ellas ya que en la actualidad en algunos casos las personas cambiaron su aspecto facial, tanto la imagen a reconocer como las del entrenamiento fueron capturadas en el mismo ambiente.

A continuación se explican en detalles como se encuentran en la actualidad las personas a las cuales se le realizó el reconocimiento. La persona 1,2,6 no variaron su aspecto facial, la persona 3 se realizó una micropigmentación en las cejas, la persona 4 adelgazó 7 kilos por ello su rostro sufrió un cambio, la persona 5 se incorporó lentes de contactos.

En la tabla 4.12 se observa que el reconocimiento fue correcto para 6 las personas e incorrecto para 0 personas.

Tabla 4.12: Reconocimiento con respecto al período de tiempo de las imágenes de entrenamiento

Muestra	Reconocimiento
1	Si
2	Si
3	Si
4	Si
5	Si
6	Si

Con estos resultados se puede decir que el reconocimiento facial es correcto siempre y cuando la persona no sufra cambios considerables y drásticos en su rostro como por ejemplo un accidente en su cara ó transcurra un período de tiempo muy extenso en el cual cambien sus facciones como por ejemplo que la persona envejezca. En caso de que esto ocurra se debe realizar una nueva captura de las imágenes de entrenamiento para la base de datos.

4.1.8. Prueba 8: Eficacia del reconocimiento con respecto a variaciones en el ambiente de las imágenes de entrenamiento

Se le realizaron pruebas de reconocimiento a una persona cuyas imágenes de entrenamiento fueron tomadas en diferentes ambientes, en la figura 4.7 se puede observar algunas fotos utilizadas. Cabe acotar, que la imagen a reconocer en algunos casos fue tomada en el mismo ambiente de las imágenes correspondientes con la base de datos.

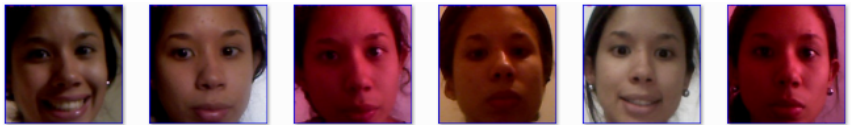


Figura 4.7: Imágenes de Entrenamiento

En la tabla 4.13 se observa que el reconocimiento fue correcto 4 veces e incorrecto 2 veces.

Tabla 4.13: Reconocimiento con respecto a variaciones en el ambiente de las imágenes de entrenamiento

Muestra	Reconocimiento
1	No
2	Si
3	Si
4	No
5	Si
6	Si

En la figura 4.8 se observa el ambiente de la imagen a reconocer el cual coincide con la muestra número 2 y 5. Cabe acotar que, todas las imágenes de entrenamiento fueron comparadas con fotos capturadas en este mismo ambiente para su posterior reconocimiento.

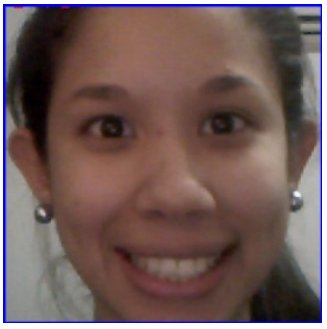


Figura 4.8: Ambiente de la imagen a reconocer.

En la figura 4.9 se observa la muestra 1 de entrenamiento ya una vez procesada la cual no permite distinguir con claridad algunas facciones del rostro esto es debido a que la captura de la imagen se realizó en un ambiente de baja luminosidad. El reconocimiento para este caso fue incorrecto.

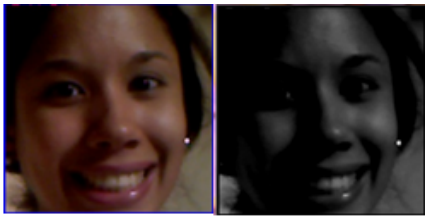


Figura 4.9: Imagen de entrenamiento de la muestra 1

En la figura 4.10 se observa la muestra 2 de entrenamiento ya una vez procesada la cual permite distinguir con claridad las facciones del rostro, la captura de la imagen se realizó en un ambiente de alta luminosidad. El reconocimiento para este caso fue correcto.

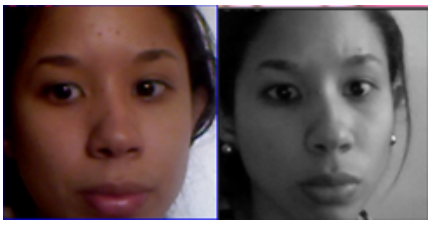


Figura 4.10: Imagen de entrenamiento de la muestra 2

En la figura 4.11 se observa la muestra 3 de entrenamiento ya una vez procesada la cual permite distinguir con claridad las facciones del rostro. El reconocimiento para este caso fue correcto.

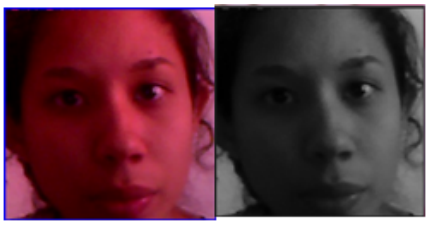


Figura 4.11: Imagen de entrenamiento de la muestra 3

En la figura 4.12 se observa la muestra 4 de entrenamiento ya una vez procesada la cual no permite distinguir con claridad algunas facciones del rostro esto es debido a que la captura de la imagen se realizó en un ambiente de baja luminosidad. El reconocimiento para este caso fue incorrecto.

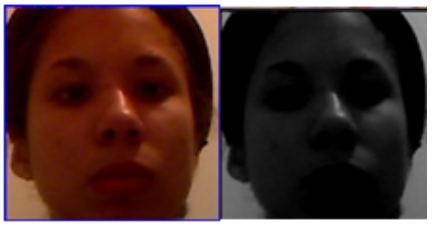


Figura 4.12: Imagen de entrenamiento de la muestra 4

En la figura 4.13 se observa la muestra 5 de entrenamiento ya una vez procesada la cual permite distinguir con claridad las facciones del rostro, la captura de la imagen se realizó en un ambiente de muy alta luminosidad. El reconocimiento para este caso fue correcto.

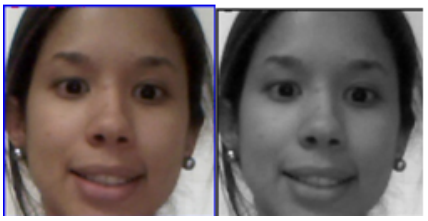


Figura 4.13: Imagen de entrenamiento de la muestra 5

En la figura 4.14 se observa la muestra 6 de entrenamiento ya una vez procesada la cual permite distinguir con claridad las facciones del rostro. El reconocimiento para este caso fue correcto.

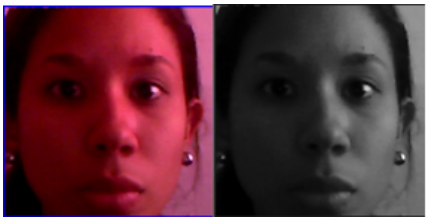


Figura 4.14: Imagen de entrenamiento de la muestra 6

Con estos resultados se puede decir que el reconocimiento facial es correcto en ambientes donde las imágenes de entrenamiento y a reconocer coincide como ocurrió en los casos de las muestras 2 y 5.

En los casos de las muestras 3 y 6 que fueron tomadas en ambientes diferentes el reconocimiento también fue correcto.

Para el caso 1 y 4 el reconocimiento fue incorrecto debido a que la luminosidad en donde se capturaron las imágenes afecto en el procesado de la imagen.

Por ello, al momento de realizar el reconocimiento éste se debe hacer en el mismo ambiente donde fueron tomadas las imágenes con las cuales se hará la comparación para un mayor nivel de confianza en el reconocimiento o en ambientes con

alta o muy alta luminosidad (500 a 2000 Lux) para evitar posibles fallas que pueda ocasionar los cambios de ambiente, luminosidad, etc en el procesado de la imagen.

4.1.9. Prueba 9: Eficacia del reconocimiento con respecto a cambios de cámara

Se realizaron pruebas de reconocimiento a una persona en la cual para 4 casos las imágenes de entrenamiento y a reconocer fueron tomadas con cámaras diferentes.

- Cámara 1: Cámara Web Havit Usb Hv-n632.
- Cámara 2: Cámara Web Markvision M-350K-MV.

Ver en el apéndice M las características de la cámara 1 con la cual se hará el reconocimiento y se utilizará en la Escuela de Telecomunicaciones y la cámara 2 con las cuales se realizó la captura de la imágenes de entrenamiento de las muestras 1, 2, 3 y 6.

En la figura 4.15 se puede observar dos imágenes de la misma persona capturadas con la cámara 2 y 1.

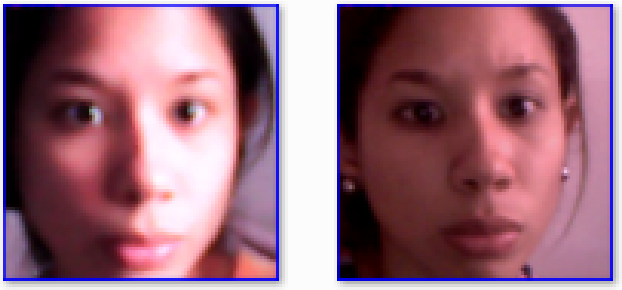


Figura 4.15: Imágenes capturadas con diferentes cámaras

En la tabla 4.14 se observa que sólo una muestra fue reconocida de forma incorrecta.

Tabla 4.14: Reconocimiento con respecto a cambios en la cámara

Muestra	Reconocimiento
1	Si
2	Si
3	Si
4	Si
5	Si
6	No
7	Si

En la figura 4.16 se observa la imagen de entrenamiento y a reconocer de la muestra 1 las cuales fueron tomadas con cámaras diferentes y en el mismo ambiente. El reconocimiento en este caso fue correcto.

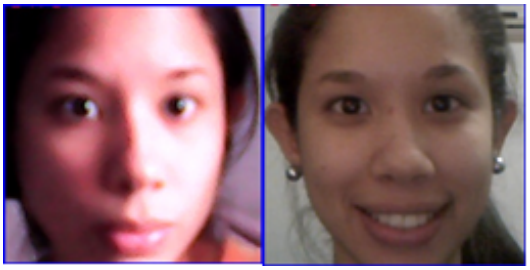


Figura 4.16: Imágenes de la muestra 1

En la figura 4.17 se observa la imagen de entrenamiento y a reconocer de la muestra 2 las cuales fueron tomadas con cámaras diferentes e iguales ambientes. El reconocimiento en este caso fue correcto.

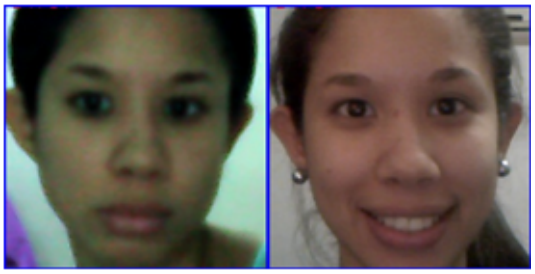


Figura 4.17: Imágenes de la muestra 2

En la figura 4.18 se observa la imagen de entrenamiento y a reconocer de la

muestra 3 las cuales fueron tomadas con cámaras diferentes y en el mismo ambiente. El reconocimiento en este caso fue correcto.

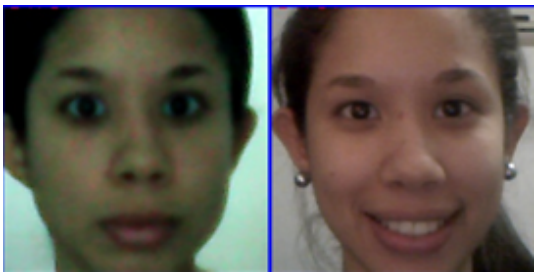


Figura 4.18: Imágenes de la muestra 3

En la figura 4.19 se observa la imagen de entrenamiento y a reconocer de la muestra 4 las cuales fueron tomadas con cámaras y ambientes iguales pero diferente luminosidad. El reconocimiento en este caso fue correcto.

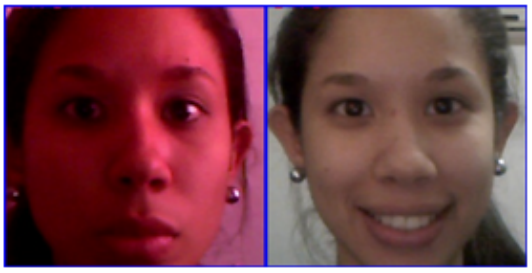


Figura 4.19: Imágenes de la muestra 4

En la figura 4.20 se observa la imagen de entrenamiento y a reconocer de la muestra 5 las cuales fueron tomadas con cámaras, ambientes y luminosidad iguales. El reconocimiento en este caso fue correcto.

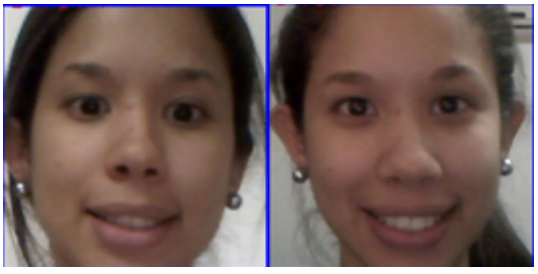


Figura 4.20: Imágenes de la muestra 5

En la figura 4.21 se observa la imagen de entrenamiento y a reconocer de la muestra 6 las cuales fueron tomadas con cámaras y luminosidad diferentes pero en el mismo ambiente. El reconocimiento en este caso fue incorrecto debido a que la persona realizó movimiento en el rostro al momento de la captura, el procesado de esta imagen de entrenamiento para un lado del rostro no se visualizó de forma correcta por el programa.

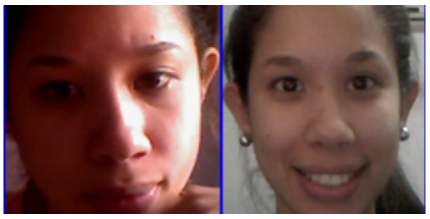


Figura 4.21: Imágenes de la muestra 6

En la figura 4.22 se observa la imagen de entrenamiento y a reconocer de la muestra 7 las cuales fueron tomadas con cámaras y ambientes iguales. El reconocimiento en este caso fue correcto.

Figura 4.22: Imágenes de la muestra 7

Con estos resultados se puede decir que el reconocimiento no se vio afectado a cambios de cámara para la captura de imágenes. Pero para evitar posibles fallas al momento de realizar el reconocimiento se debería utilizar la misma cámara tanto para capturar las imágenes de entrenamiento como a reconocer, esto le brinda un mayor nivel de confiabilidad al reconocer, ya que como podemos observar en algunas imágenes de rostros capturados con distintas cámaras la calidad de la imagen no es la misma.

La cámara 1 modelo Havit Usb Hv-n632 será la utilizada en la implementación del software piloto de reconocimiento facial en la Escuela de Telecomunicaciones

ya que posee mejor calidad y nitidez en las imágenes.

4.1.10. Prueba 10: Eficacia del reconocimiento con respecto a un número de muestras tomadas a una población

Se realizaron pruebas de reconocimiento facial a una población en específica tomando en cuenta diferentes ambientes, iluminación, escenarios, estados y aspectos del rostro de la persona los cuales fueron los siguientes:

• Población: Personal docente, administrativo y estudiantil (preparadores, becas servicios y estudiantes) pertenecientes a la Escuela de Telecomunicaciones.

• Luminosidad: Muy alta, alta, media.

• Ambientes: Oficinas, cubículos y laboratorios pertenecientes a la Escuela de Telecomunicaciones.

• Estados y aspectos del rostro: sonriendo, triste, bravo, lengua afuera, muecas, ojos cerrados, con lentes de leer y de sol, mitad de la cara, cabello en la cara, cabello suelto y amarrado, con gorra.

• Escenarios: Comparación con fotos impresas como por ejemplo de cédulas, carnets, etc. Otro de los escenarios que se tomó en cuenta que vale recalcar es una persona utilizando el número de cédula de otra para tratar de registrar la asistencia.

En esta prueba se le explicó y mostró al personal que labora en la Escuela de Telecomunicaciones el funcionamiento del software piloto de reconocimiento facial para el control de asistencia de manera tal que se fueran familiarizando con la herramienta.

Esta prueba se le realizó a 47 personas en un mismo día pertenecientes a la Escuela de Telecomunicaciones en cada uno de sus diferentes ambientes contando con la colaboración e ideas de ellos de posibles escenarios, estados y aspectos del rostro que se pudieran cambiar o variar para probar la eficacia del software piloto de reconocimiento facial. Todo esto con el fin de que la imagen que se desea reconocer fuera distintas a la imagen que registraron en la base de datos de entrenamiento.

Tabla 4.15: Reconocimiento a una población en específica.

Muestra	Reconocimiento
1	Si
2	Si
3	No
4	Si
5	Si
6	Si
7	Si
8	Si
9	Si
10	No
11	Si
12	Si
13	Si
14	Si
15	Si
16	Si
17	Si
18	Si
19	Si
20	No
21	Si
22	Si
23	Si
24	Si
25	Si
26	Si
27	Si
28	Si
29	Si
30	Si
31	Si
32	No
33	Si
34	Si
35	Si
36	Si
37	Si
38	Si
39	Si
40	Si
41	Si
42	Si
43	Si
44	Si
45	Si
46	Si
47	Si

En la tabla 4.15 se observan los siguientes resultados: de un total de 47 muestras tomadas, 43 de ellas fueron reconocidas de manera correcta (aciertos) y 4 de forma incorrecta (fallas), obteniéndose así un porcentaje de aciertos del 91.49% y de fallas del 8.51 % dichos valores fueron calculados con las ecuaciones 4.1 y 4.2. Los casos no reconocido 2 de ellos fueron debido a que la persona realizó movimientos del rostro al momento de la captura de la foto a reconocer mientras que los otros 2, uno de ellos fue debido a que la persona no registro primeramente su imagen en la base de datos con la cual se realiza la comparación y reconocimiento facial y el otro debido a que la persona utilizó una foto tipo carnet al momento de la captura de la imagen.

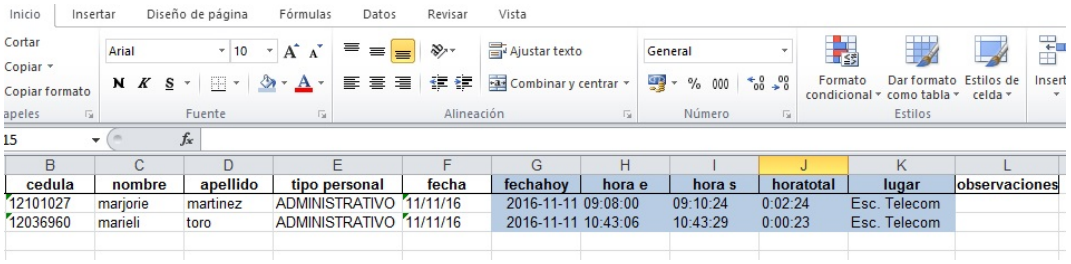
$$\text{Porcentaje de Aciertos} = \frac{\text{TotaldeAciertos}}{\text{TotaldeMuestras}} \text{x} 100 \tag{4.1}$$

$$\text{Porcentaje de Fallas} = \frac{\text{TotaldeFallas}}{\text{TotaldeMuestras}} \text{x} 100 \tag{4.2}$$

En la figura 4.23 se observa el registro de asistencia en pantalla de un grupo del personal que labora en la Escuela de Telecomunicaciones que colaboró para la realización de está prueba.

Figura 4.23: Registro de asistencia en pantalla del personal

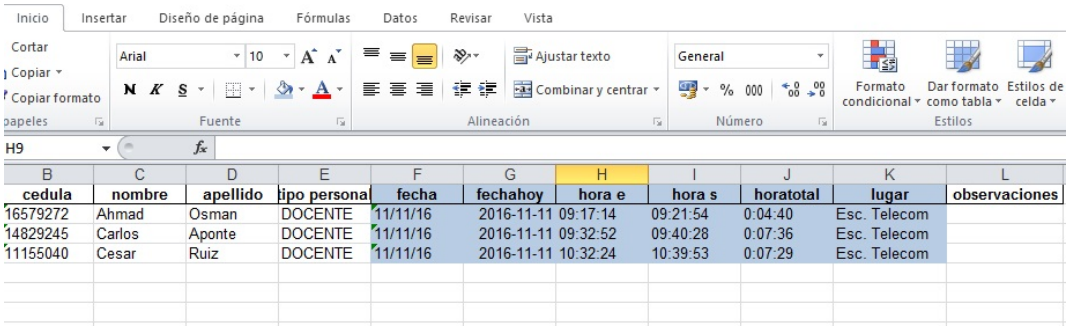
En la figura 4.24 se observa la data generada por parte del software para el personal administrativo que colaboró en la realización de la prueba.



cedula	nombre	apellido	tipo personal	fecha	fechahoy	hora e	hora s	horatotal	lugar	observaciones
12101027	marjorie	martinez	ADMINISTRATIVO	11/11/16	2016-11-11	09:08:00	09:10:24	0:02:24	Esc. Telecom	
12036960	marieli	toro	ADMINISTRATIVO	11/11/16	2016-11-11	10:43:06	10:43:29	0:00:23	Esc. Telecom	

Figura 4.24: Reporte de asistencia del personal administrativo

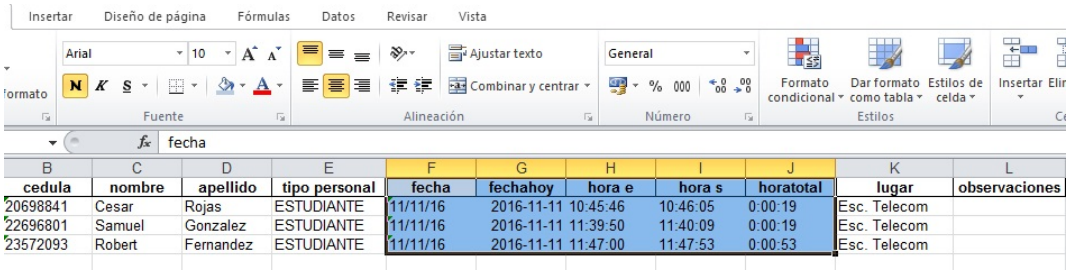
En la figura 4.24 se observa la data generada por parte del software para el personal docente que colaboró en la realización de la prueba.



cedula	nombre	apellido	tipo personal	fecha	fechahoy	hora e	hora s	horatotal	lugar	observaciones
16579272	Ahmad	Osman	DOCENTE	11/11/16	2016-11-11	09:17:14	09:21:54	0:04:40	Esc. Telecom	
14829245	Carlos	Aponte	DOCENTE	11/11/16	2016-11-11	09:32:52	09:40:28	0:07:36	Esc. Telecom	
11155040	Cesar	Ruiz	DOCENTE	11/11/16	2016-11-11	10:32:24	10:39:53	0:07:29	Esc. Telecom	

Figura 4.25: Reporte de asistencia del personal docente

En la figura 4.24 se observa la data generada por parte del software para el personal estudiantil que labora en la Escuela de Telecomunicaciones que colaboró en la realización de la prueba.



cedula	nombre	apellido	tipo personal	fecha	fechahoy	hora e	hora s	horatotal	lugar	observaciones
20698841	Cesar	Rojas	ESTUDIANTE	11/11/16	2016-11-11	10:45:46	10:46:05	0:00:19	Esc. Telecom	
22696801	Samuel	Gonzalez	ESTUDIANTE	11/11/16	2016-11-11	11:39:50	11:40:09	0:00:19	Esc. Telecom	
23572093	Robert	Fernandez	ESTUDIANTE	11/11/16	2016-11-11	11:47:00	11:47:53	0:00:53	Esc. Telecom	

Figura 4.26: Reporte de asistencia del personal estudiantil

Con estos resultados se puede decir que el software piloto de reconocimiento facial para el control de asistencia en la Escuela de Telecomunicaciones funciona de manera eficaz al tener un porcentaje de aciertos mayor al 90%en esta prueba.

4.1.11. Prueba 11: Desempeño del software piloto de reconocimiento facial para el control de asistencia

Se realizaron pruebas al software piloto de reconocimiento facial para evaluar su desempeño con respecto al registro de asistencia.

En la figura 4.27 se observa el registro de asistencia en pantalla para las personas en el cual su reconocimiento facial ocurrió con éxito donde se puede visualizar cédula, nombre, apellido, tipo de personal, fecha, hora de entrada y salida, cantidad de horas laboradas y lugar donde se registro la asistencia. Cabe acotar, que si la persona no fue reconocida no se registrará asistencia por ello del total de las 24 personas seleccionadas para la prueba realizada sólo se muestra en pantalla las 19 donde el reconocimiento fue correcto.

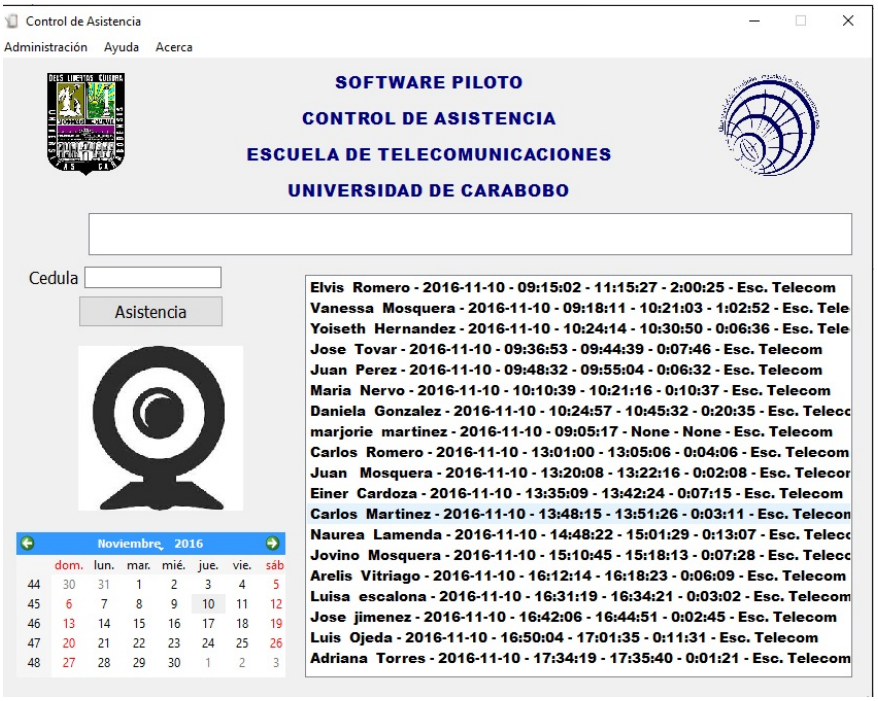


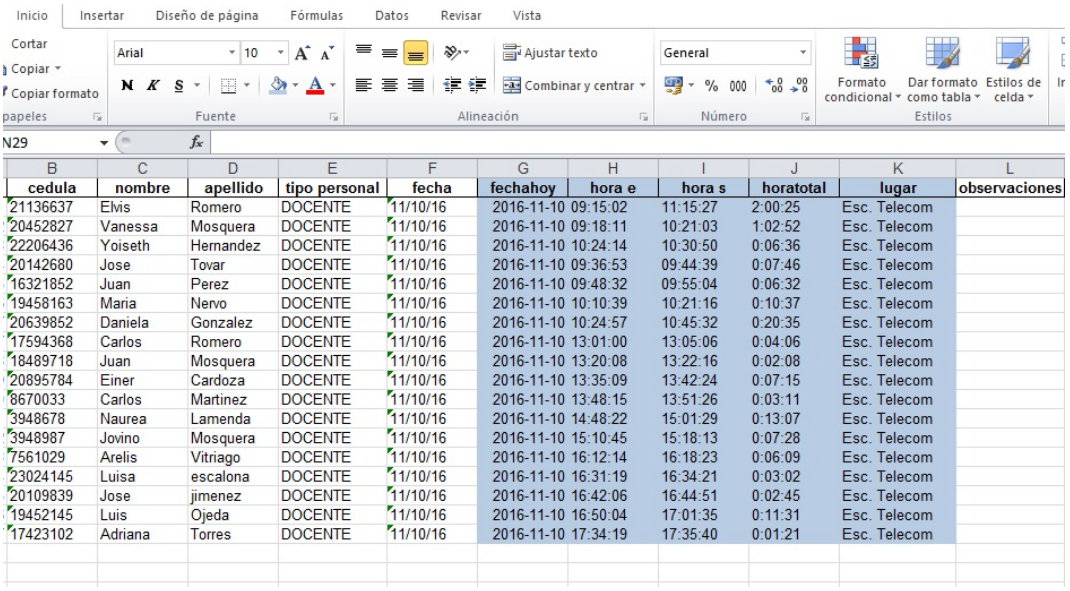
Figura 4.27: Registro de asistencia en pantalla

En la tabla 4.16 se observa que los casos donde el reconocimiento fue correcto la persona se encontraba registrada en la base de datos por lo tanto, su asistencia se registró con éxito. Cabe acotar, que la persona 5 no se encontraba registrada en la base de datos por ello su proceso de reconocimiento y asistencia no fue exitoso, en cuanto a las personas 6, 8, 9 y 10 su prueba consistió en registrarse haciéndose pasar por otra persona utilizando su número de cédula, el reconocimiento y registro de asistencia en esos casos fue incorrecto debido a que el rostro no correspondía con el de la persona a la cual le pertenecía ese número de cédula.

Tabla 4.16: Reconocimiento facial con el registro de asistencia

Muestra	Reconocimiento	Asistencia
1	Si	Si
2	Si	Si
3	Si	Si
4	Si	Si
5	No	No
6	No	No
7	Si	Si
8	No	No
9	No	No
10	Si	Si
11	No	No
12	Si	Si
13	Si	Si
14	Si	Si
15	Si	Si
16	Si	Si
17	Si	Si
18	Si	Si
19	Si	Si
20	Si	Si
21	Si	Si
22	Si	Si
23	Si	Si
24	Si	Si

En la figura 4.28 se observa el reporte final de la data generada de las 19 personas cuyo reconocimiento facial y registro de asistencia ocurrió correctamente.



cedula	nombre	apellido	tipo personal	fecha	fechahoy	hora e	hora s	horatotal	lugar	observaciones
21136637	Elvis	Romero	DOCENTE	11/10/16	2016-11-10	09:15:02	11:15:27	2:00:25	Esc. Telecom	
20452827	Vanessa	Mosquera	DOCENTE	11/10/16	2016-11-10	09:18:11	10:21:03	1:02:52	Esc. Telecom	
22206436	Yoiseth	Hernandez	DOCENTE	11/10/16	2016-11-10	10:24:14	10:30:50	0:06:36	Esc. Telecom	
20142680	Jose	Tovar	DOCENTE	11/10/16	2016-11-10	09:36:53	09:44:39	0:07:46	Esc. Telecom	
16321852	Juan	Perez	DOCENTE	11/10/16	2016-11-10	09:48:32	09:55:04	0:06:32	Esc. Telecom	
19458163	Maria	Nervo	DOCENTE	11/10/16	2016-11-10	10:10:39	10:21:16	0:10:37	Esc. Telecom	
20639852	Daniela	Gonzalez	DOCENTE	11/10/16	2016-11-10	10:24:57	10:45:32	0:20:35	Esc. Telecom	
17594368	Carlos	Romero	DOCENTE	11/10/16	2016-11-10	13:01:00	13:05:06	0:04:06	Esc. Telecom	
18489718	Juan	Mosquera	DOCENTE	11/10/16	2016-11-10	13:20:08	13:22:16	0:02:08	Esc. Telecom	
20895784	Einer	Cardoza	DOCENTE	11/10/16	2016-11-10	13:35:09	13:42:24	0:07:15	Esc. Telecom	
8670033	Carlos	Martinez	DOCENTE	11/10/16	2016-11-10	13:48:15	13:51:26	0:03:11	Esc. Telecom	
3948678	Naurea	Lamenda	DOCENTE	11/10/16	2016-11-10	14:48:22	15:01:29	0:13:07	Esc. Telecom	
3948987	Jovino	Mosquera	DOCENTE	11/10/16	2016-11-10	15:10:45	15:18:13	0:07:28	Esc. Telecom	
7561029	Arelis	Vitriago	DOCENTE	11/10/16	2016-11-10	16:12:14	16:18:23	0:06:09	Esc. Telecom	
23024145	Luisa	escalona	DOCENTE	11/10/16	2016-11-10	16:31:19	16:34:21	0:03:02	Esc. Telecom	
20109839	Jose	jimenez	DOCENTE	11/10/16	2016-11-10	16:42:06	16:44:51	0:02:45	Esc. Telecom	
19452145	Luis	Ojeda	DOCENTE	11/10/16	2016-11-10	16:50:04	17:01:35	0:11:31	Esc. Telecom	
17423102	Adriana	Torres	DOCENTE	11/10/16	2016-11-10	17:34:19	17:35:40	0:01:21	Esc. Telecom	

Figura 4.28: Data generada para la asistencia

Se realizó una prueba en la cual se muestra como se enviaría y observaría una notificación enviada. En la figura 4.29 se observa la ventana para enviar las notificaciones.

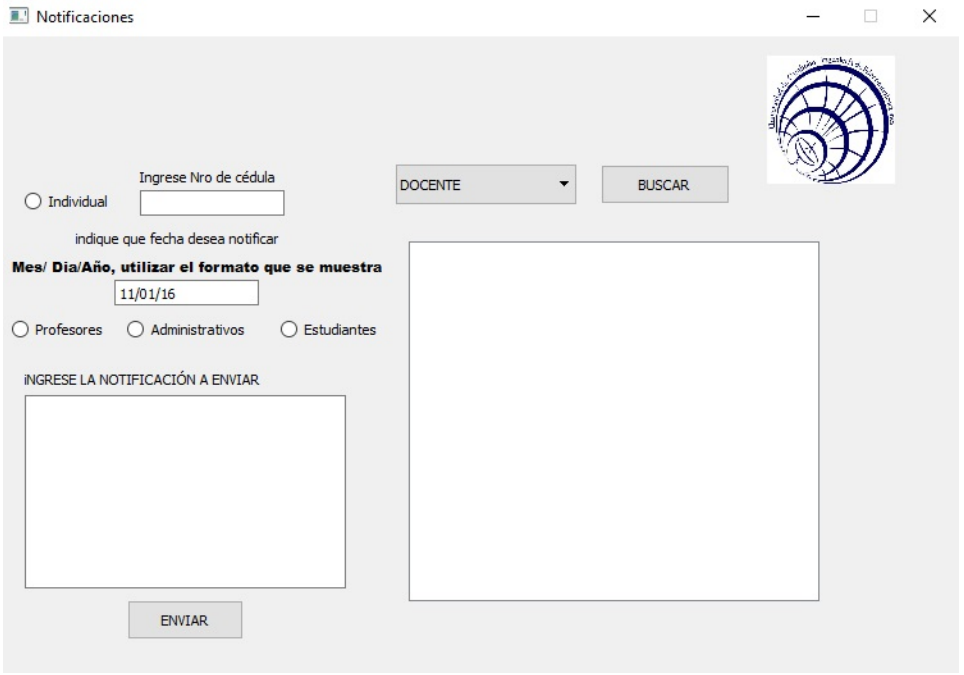


Figura 4.29: Ventana para el envío de notificaciones.

En la figura 4.30 se observa la notificación que se desea enviar la cual posteriormente se mostrará en la ventana principal como se observa en la figura 4.31.

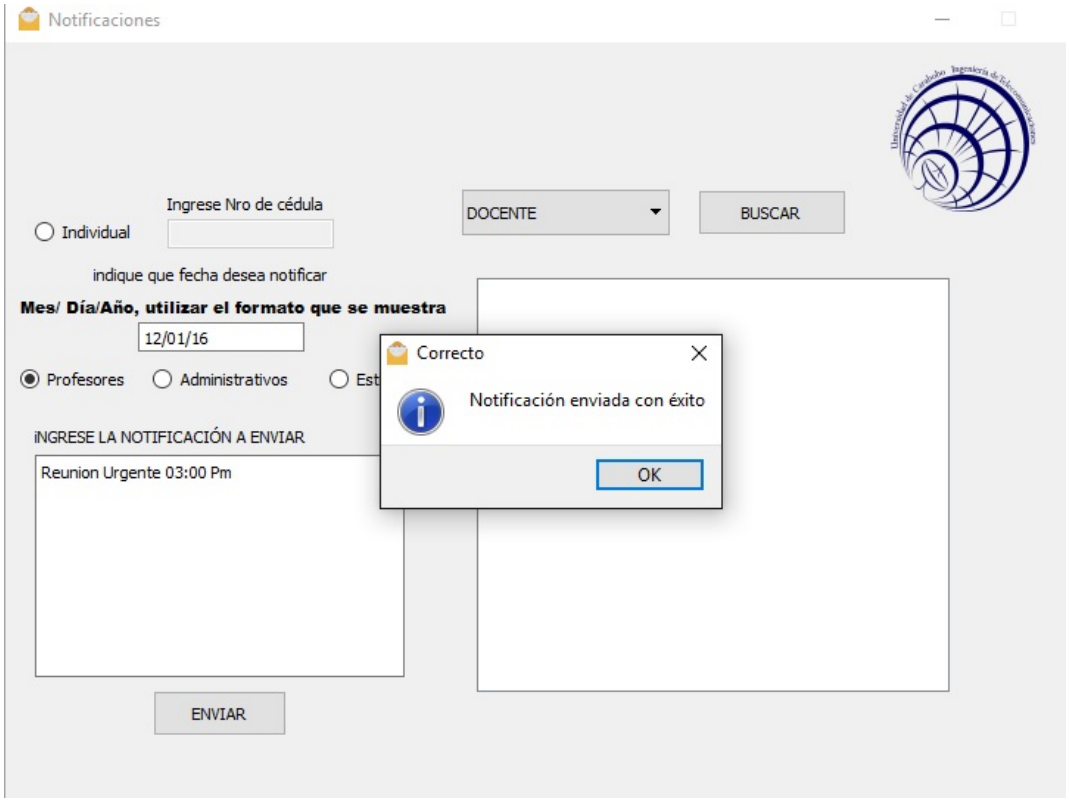


Figura 4.30: Notificación a enviar

Figura 4.31: Notificación en la ventana principal.

Para más detalles e información sobre el proceso de registro de asistencia ver los apéndices I y J.

Con estos resultados obtenidos se puede decir que el software piloto funciona de manera eficaz ya que al ser correctamente reconocido el rostro, el registro de asistencia será exitoso. En caso contrario es decir, que el reconocimiento sea incorrecto no se registrará asistencia, con esto se evitan suplantaciones de identidad por parte de las personas para registrar asistencias.

A su vez el software piloto es eficiente debido a que el consumo de recursos computacionales y monetarios es bajo en comparación con otros programas de reconocimiento facial ya que para su realización e implementación tan sólo requirió la compra de una cámara web y su programación fue por medio de software libre. También es eficiente debido a su rapidez de detección y a su servidor local el cual permite que se mantenga operativo sin conexión a Internet.

En el apéndice N se puede ver el presupuesto de los recursos que se requieren para la implementación del software piloto de reconocimiento facial para el control de asistencias en la Escuela de Telecomunicaciones de la Universidad de Carabobo.

Las condiciones mínimas de operación del software piloto de reconocimiento facial para el control de asistencia en la Escuela de Telecomunicaciones en la Universidad de Carabobo que se deben tomar en cuenta siempre son las siguientes:

- Tener como mínimo 6 imágenes en la base de datos de entrenamiento por individuo.
- Permitir realizar una segmentación de todas las facciones de la persona. Es decir, no taparse alguna parte del rostro al momento de la captura de la imagen.
- Colocarse en una posición frontal a la cámara.
- Colocarse a una distancia entre 30 a 90 cm para la captura de la imagen.
- Evitar tener personas al lado al momento de realizar la captura del rostro para registrar la asistencia.
- Si ocurren cambios en el rostro realizar una nueva captura de imágenes para la base de datos.

- Realizar la captura de las imágenes de entrenamiento y a reconocer en el mismo ambiente para un mayor nivel de confianza en el reconocimiento o en ambientes diferentes pero que cuenten con muy alta, alta o media luminosidad, es decir; niveles de luminosidad comprendidos entre 200 a 2000 Lux.

- Utilizar cámaras cuya resolución permita obtener imágenes de calidad.

- No moverse al momento de la captura de la imagen.

- No utilizar fotos digitales como por ejemplo: fotos tipo carnet o impresas para tratar de registrar su asistencia o la de otra persona ya que el software piloto no la reconocerá y de manera tal el registro no se realizará.

- Contar con energía eléctrica para que el software este operativo, si en algún momento no se cuenta con esta se debe realizar el proceso de asistencia manual que se explica en el apéndice J.

- Para la conexión remota se requiere que la PC tenga configurada la IP de forma manual.

Una vez tomadas en cuenta todas estas condiciones mínimas se obtiene una eficacia del 100% por parte del software piloto de reconocimiento facial para el control de asistencia de la Escuela de Telecomunicaciones de la Universidad de Carabobo.

Capítulo V

Conclusiones y Recomendaciones

5.1. Conclusiones

Se cumplió con el objetivo y aporte principal planteado en el trabajo especial de grado que consistía en diseñar un software piloto de reconocimiento facial para el control de asistencia en la Escuela de Telecomunicaciones de la Universidad de Carabobo.

• Se han analizado y comprendido los parámetros que interfieren en la detección y reconocimiento facial mediante el procesamiento de imágenes, el cual se basa en el algoritmo Viola Jones, Haar Like-Feature, imagen integral, extracción de características, filtros Haar y clasificadores en cascadas que pueden ser fundamentales al momento de reconocer el rostro de una persona.

• El software piloto fue diseñado en Python, el uso de este lenguaje de programación con las librerías OpenCV permite el desarrollo eficiente a bajo costo del sistema reconocimiento facial, además que simplifica la codificación en comparación con otros lenguajes como por ejemplo C++.

• El reconocimiento facial se basó en el algoritmo LBPH ya que es el más indicado al momento de realizar aplicaciones en tiempo real debido a su robustez

frente a cambios de iluminación siempre y cuando estos permitan realizar un correcto procesado de la imagen. Este operador facilita el entrenamiento a partir de pocas imágenes de un mismo individuo y no es necesario contar con una gran base de datos como lo requieren otros algoritmos que también son propios de la librería OpenCV.

• Se diseñó una interfaz gráfica que permite la administración del programa de registro de asistencia en la Escuela de Telecomunicaciones de la Universidad de Carabobo en entornos de programación como Python y su librería PyQt la cual cuenta con un gran cantidad de elementos para la realización de interfaces de usuario, siendo de fácil manejo.

• Con el diseño de este nuevo registro de asistencia se garantiza actualización y confiabilidad con respecto al diseño con el cual se cuenta actualmente ya que posee numerosas mejoras y ventajas tales como el reconocimiento facial, notificaciones, eliminación del registro manual para preparadores y becas servicios, permanece operativo aún sin conexión a Internet, entre otras.

• Se realizaron pruebas para evaluar el desempeño del software piloto en las cuales se han obtenido resultados satisfactorios tanto para el reconocimiento facial como para el registro de asistencias. De estas pruebas se establecen criterios y condiciones mínimas de operación como crear una base de datos con por lo menos 6 imágenes de entrenamiento, la distancia para el reconocimiento debe estar comprendida entre 30 a 90 centímetros, implementar el software piloto de reconocimiento facial en el mismo ambiente o en niveles luminosidad comprendidos entre 200 a 2000 Lux en el cual se creará la base de datos para prevenir posibles fallas en el reconocedor, etc. Una vez establecidos y tomados en cuenta todos estos criterios y condiciones mínimas, se garantiza una eficacia en el software piloto de reconocimiento facial para el control de asistencias del 100%.

5.2. Recomendaciones

- Leer los manuales de administración, usuario e instalación para un correcto uso del software piloto diseñado e implementando.

- Se recomienda que luego de la instalación del software piloto de reconocimiento facial en la Escuela de Telecomunicaciones se establezca un período de prueba de funcionamiento entre 3 a 4 meses para obtener posteriormente los permisos de conexión con la TIC y trabajar de forma conjunta con ellos.

- Tomar en cuenta todas las condiciones mínimas de operación del programa para un eficaz uso y funcionamiento del mismo.

- Crear la base de datos correspondiente con el entrenamiento en el mismo ambiente de luminosidad donde se realizará el reconocimiento facial. También se recomienda, actualizar cada cierto tiempo las imágenes del personal que se encuentran en la base de datos.

- Realizar trabajos a futuros en este misma línea de investigación del reconocimiento facial en la Escuela de Telecomunicaciones como por ejemplo, implementar un sistema de vigilancia utilizando la cámara con que cuenta la recepción.

- Realizar una interfaz o página web para la administración y uso del software piloto de reconocimiento facial para el control de asistencias para así acceder a este en cualquier lugar a través de la utilización del Internet.

- Implementar un software de reconocimiento facial para el control de asistencia en toda la Facultad de Ingeniería de la Universidad de Carabobo utilizando como base éste, que se ha diseñado para la Escuela de Telecomunicaciones.

Apéndice A

Código del Programa Principal

```
# -*- coding: cp1252 -*-
import psycopg2 #libreria para conectar la base de datos de POSTGRESQL con python
import pprint
import sys
import os
from PyQt4 import QtCore, QtGui, uic
import time
import cv2 #Libreria de OpenCV
import numpy as np
from PIL import Image
import shutil
from validar import * #modulo para importar la ventana de verificacion, para acceder
                      #a la ventana de administracion
from importa_base import * #modulo donde se cargan las fotos registradas en la carpeta
                           #base de datos
from datetime import datetime, timedelta, date

# Cargar nuestro archivo .ui
form_class = uic.loadUiType("principal.ui")[0]  #Toma el dise o de Qt designer y lo
                                        #muestra dentro del programa principal

class MyWindowClass(QtGui.QMainWindow, form_class):
def __init__(self, parent=None):
QtGui.QMainWindow.__init__(self, parent)
```

```
self.setupUi(self)
self.validar = Validar() #Llama al modulo validar, para hacer el llamado de la ventana

self.btn_Entrada.clicked.connect(self.btnAsistencia) #Definir Boton entrada, realiza
                    #la comparaci n y registra la asistencia

self.IniciarBase()

images, labels = images_and_labels(path) #Llame a la funci n images_and_labels y obtener
                            #las im genes de la cara y las etiquetas correspondientes
recognizer.train(images, np.array(labels))  #Realiza el entrenamiento

#Predeterminar una tama o fijo a la ventana principal -------
self.setFixedSize(850, 650) #fija valores predeterminado para las ventanas,

#Agrgar Logos a la Ventana Principal----
i1 = QtGui.QLabel(self)  #QLabel crea una etiqueta para almacenar texto o imagenes
i1.setGeometry(700, 35, 100, 100)#setGeometry() inserta la posicion y el tama o de la
#etiqueta
pixmap = QtGui.QPixmap('telecom.jpg')# QPixmap permite ubicar una imagen para ser insertada
                                    # dentro de la ventana
pixmap = pixmap.scaled(100, 100, QtCore.Qt.KeepAspectRatio) # scala la imagen en 100 x 100
                            #pixeles, para su insercion en la etiqueta
i1.setPixmap(pixmap)#relaciona el objeto con la etiqueta
i1.show() # muestra la imagen con las dimensiones antes se aladas

i2 = QtGui.QLabel(self)  #QLabel crea una etiqueta para almacenar texto o imagenes
i2.setGeometry(50, 35, 100, 100)#setGeometry() inserta la poscion y el tama o de la etiqueta
pixmap2 = QtGui.QPixmap('uc.jpg')#QPixmap permite ubicar una imagen para ser insertada dentro
                                    #de la ventana
pixmap2 = pixmap2.scaled(100, 100, QtCore.Qt.KeepAspectRatio) # scala la imagen en 100 x 100
                              #pixeles para su insercion en la etiqueta
i2.setPixmap(pixmap2)#relaciona el objeto con la etiqueta
i2.show() # muestra la imagen con las dimensiones antes se aladas

label_Foto = QtGui.QLabel(self)  #QLabel crea una etiqueta para almacenar texto o imagenes
label_Foto.setGeometry(80, 300, 160, 160)#setGeometry() inserta la posicion y el tamano
                                        #de la etiqueta
pixmap3 = QtGui.QPixmap('camara.jpg')#QPixmap permite ubicar una imagen para ser
                                    #insertada dentro de la ventana
```

```
pixmap3 = pixmap3.scaled(160, 160, QtCore.Qt.KeepAspectRatio) # escala la imagen en
                        #100 x 100 pixeles, para su insercion en la etiqueta
label_Foto.setPixmap(pixmap3)#relaciona el objeto con la etiqueta
label_Foto.show() # muestra la imagen con las dimensiones antes se aladas

# crear el botn administrar
administrarAction = QtGui.QAction(QtGui.QIcon('admin.png'), '&Administrar', self)
#inserta un icono y el nombre del boton

administrarAction.setShortcut('Ctrl+q') #Crea un acceso directo al boton administrar
administrarAction.setStatusTip('Administraci n del Personal')
# Muestra la acci n en la parte inferior de la pantalla

administrarAction.triggered.connect(self.btnAdministrar) #al presionar
#el botn administrar  abre la ventana de administrac n

#Crea el boton acerca de
acercaAction = QtGui.QAction('&Acerca de', self) #inserta un icono y el nombre del boton
acercaAction.setShortcut('Ctrl+p') #Crea un acceso directo al boton administrar
acercaAction.setStatusTip('informacion del software') # Muestra la accion en la parte
                                                      # inferior de la pantalla
acercaAction.triggered.connect(self.AcercaAutores) #al presionar el
#boton administrar abre la ventana de administrac n

self.statusBar()  # Para crear la clase de menu
menubar = self.menuBar() #Se le asigna un nombre al menu
AdministrarMenu = menubar.addMenu('&Administraci n') #Se crea un submenu desplegable
AdministrarMenu.addAction(administrarAction) #para crear los botones dentro del submenu
AyudaMenu = menubar.addMenu('&Ayuda') #Se crea un submenu desplegable
AcercaMenu = menubar.addMenu('&Acerca') #Se crea un submenu desplegable
AcercaMenu.addAction(acercaAction) #para crear los botones dentro del submenu

def AcercaAutores(self):
autores = QtGui.QMessageBox.about(self, 'Acerca', 'Este trabajo fue desarrollado
por los autores')

def IniciarBase(self):

cadenaConexion="host='localhost' dbname='basededatos' user='postgres' password=
'contrase a'"
```

```
self.con =psycopg2.connect(cadenaConexion)
self.cursor = self.con.cursor()
self.con.commit()

def btnAdministrar(self):
self.validar.show()

def btnAsistencia(self):
shutil.rmtree("fotoscapturadas")
pathdir='./fotoscapturadas' #Carpeta de captura de imagenes
os.mkdir(pathdir) #Crea la carpeta donde se encuentran las imagenes capturadas por
#la camara web

self.IniciarBase() #se inicia la conexi n con la base de datos
self.cursor2 = self.con.cursor()

self.notificaciones.clear() #borrar la barra de notificaciones

self.id = str(self.lineEdit.text()) #Toma el valor que ingresa el usuario
if len(self.id)!=0: #Evita errores al conectar a la base de de datos, ya que evita que el
                                        #usuario deje en blanco la casilla de cedula
self.fechahoy = time.strftime("%x") #registra la fecha actual
self.fechabusqueda = date.today()

#variable temporal que se usa para guardar el valor de la cedula registra en BD,
#se le asigna por defecto cualquier cosa, para el supuesto que la cedula no este registrada
self.cedulapersonal = '012422sfsfsa'
self.cursor.execute("SELECT cedula FROM telecom.personal_telecom where cedula =
'"+str (self.id)+"' ")
#se realiza una busqueda en la BD con la cedula ingresada
for i in self.cursor:
self.cedulapersonal = str(i[0]) #Si la busqueda es exitosa registra la cedula
#de la BD en esta variable, de lo contrario deja
#el valor por defecto definido anteriormente
self.fechabd = '1' #valor por defecto de variable a utilizar mas adelante
if self.id==self.cedulapersonal: #Compara la cedula registrada por el usuario con la de la BD, de
#ser iguales ejecuta la siguiente linea
self.cursor.execute("SELECT fecha FROM telecom.asistencia_telecom
where cedula =
'"+str (self.id)+"'")
```

```
#se toma la fecha de la BD asistencia, para hacer la comparacion
# Se guardan los datos en variables para registrar la asistencia
for i in self.cursor:
self.fechabd = str(i[0]) #Registra la fecha, de la BD asistencia para luego se comparada

if self.fechahoy != self.fechabd: #compara la fecha actual con la de BD, si son diferentes
#implica que no ha registrado la entrda en ese momento por tanto se inicia el registro
#de entrada
vc = cv2.VideoCapture(0) #Activa la camara web
while (True):  #Inicia el programa

ret,frame = vc.read() #Se lee y guarda un frame
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #Convirtiendo la imagen a blanco y negro

faces = face_cascade.detectMultiScale(gray, 1.2, 3,
flags = cv2.CASCADE_SCALE_IMAGE, minSize=(150,150))
#Si se encuentran rostros se buscan
#sus coordenadas y se guarda su posicion
#Se dibuja un rectangulo en las coordenadas del rostro de la persona
for (x,y,w,h) in faces:
cv2.putText(frame,'Quedese QUIETO!!', (x,y), cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,250),3,1)
cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
cv2.imshow('Posicionarse en el Centro',frame) #Se muestra la imagen
if cv2.waitKey(1000): #if cv2.waitKey(10) == ord('s'):
break

cv2.destroyAllWindows()

#Comienza la captura de imagenes
start = time.time()
count = 0 #Contador
while int(time.time()-start) <= 2:
ret,frame = vc.read() #Se lee y guarda un frame
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #Para cambiar la imagen a escala
#de grises
faces = face_cascade.detectMultiScale(gray, 1.2, 3,
flags = cv2.CASCADE_SCALE_IMAGE, minSize=(150,150)) #Si se encuentran rostros se
#buscan sus coordenadas y se guarda su posicion
#Se dibuja un rectangulo en las coordenas del rostro
for (x,y,w,h) in faces:
```

```
cv2.putText(frame,'Capturando!', (x,y), cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,250),3,1)
#Puntero que indica que se estan capturando las imagenes

count +=1
cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
resized_image = cv2.resize(frame[y:y+h,x:x+w], (273, 273))
if count%5 == 0:
cv2.imwrite(pathdir+'/'+'subject'+str(self.id)+'.jpg', resized_image ); #se guardan
#las fotos
#en la carpeta pathdir con el respectivo nombre i

cv2.imshow('Captura de Fotos',frame) #Se muestra la imagen
cv2.waitKey(10)

cv2.destroyAllWindows()
ahora = time.strftime("%c")

image_paths = [os.path.join(pathdir, f) for f in os.listdir(pathdir) if f.endswith('.jpg')]
# A ade las im genes con la extensi n .jpg en image_paths
for image_path in image_paths:
predict_image_pil = Image.open(image_path).convert('L') #Se lee la imagen y se convierte
#a escala grises
predict_image = np.array(predict_image_pil, 'uint8') #Se convierte el formato de la imagen
#en una matriz numpy
faces = face_cascade.detectMultiScale(predict_image) # Detectar la cara en la imagen
for (x, y, w, h) in faces:

nbr_predicted, conf = recognizer.predict(predict_image) #Etiquetas usada para el
#reconocimiento
nbr_actual = int(os.path.split(image_path)[1].split(".")[0].replace("subject", ""))
#Se comprueba si el reconocimiento es correcto mediante la comparaci n de las etiquetas
#nbr_actual y nbr_predicted, y su umbral de confianza
if nbr_actual == nbr_predicted and conf<100: #Mientras mayor sea el valor de la variable
#de confianza, menos el reconocedor tiene confianza en el reconocimiento.
#Se toman los datos para registrar la asistencia.
self.cedula = str(self.lineEdit.text())
self.fecha = time.strftime("%x")
self.hora_e = time.strftime("%X")
# Se cargan los datos indicados de la tabla, dicha tabla se encuentran los datos del personal
#registrado en la base de datos
```

```
self.cursor.execute("SELECT nombre1, apellido1, tipopersonal FROM telecom.personal_telecom
where cedula = '"+str (self.cedula)+"'")
# Se guardan los datos en variables para registrar la asistencia
for i in self.cursor:

self.nombre = str(i[0]) #Registra el Nombre de la persona para ingresarlo como dato a la
#tabla de asistencia
self.apellido = str(i[1])#Registra el Apellido de la Persona para ingresarlo como dato a la
#tabla de asistencia
self.tipopersonal = str(i[2])
self.lugar = 'Esc. Telecom' #Lugar Por defecto que se asigna al control de asistencia

# Inserta los datos en la tabla campos
self.cursor.execute("insert into telecom.asistencia_telecom (cedula, nombre, apellido,
tipopersonal, fecha, fechahoy, hora_e, lugar) values ('"+str (self.cedula)+"',
'"+str (self.nombre)+"' , '"+str (self.apellido)+"' , '"+str (self.tipopersonal)+"' ,
'"+str (self.fecha)+"' , '"+str (self.fechabusqueda)+"',
 '"+str (self.hora_e)+"' , '"+str (self.lugar)+"')")
# Quedan los campos vacios al registrar la asistencia
self.lineEdit.setText("")
vc.release() # apaga la camara

QtGui.QMessageBox.information(self, 'Correcto', 'Asistencia Registrada Satisfactoriamente')
# mensaje para avisar que se registro satisfactoiamente la asistencia
label_Foto = QtGui.QLabel(self)  #QLabel crea una etiqueta para almacenar texto o imagenes
label_Foto.setGeometry(80, 300, 160, 160)#setGeometry() inserta la poscion y el
#tamano de la etiqueta
pixmap3 = QtGui.QPixmap('basededatos/subject'+str(self.id)+'.jpg')#QPixmap permite ubicar
#una imagen para ser insertada dentro de la ventana
pixmap3 = pixmap3.scaled(160, 160, QtCore.Qt.KeepAspectRatio) # escala la imagen en
#100 x 100 pixeles, para su insercion en la etiqueta
label_Foto.setPixmap(pixmap3)#relaciona el objeto con la etiqueta
label_Foto.show() # muestra la imagen con las dimensiones antes se aladas
self.fechanot='adadfsf'#valores por defecto para las notificaciones
self.tiponot ='asdsa' #valores por defecto para las notificaciones
self.cursor2.execute("SELECT fecha, tipo, notificacion FROM telecom.notificaciones_telecom
 where cedula = '"+str (self.cedula)+"'")
for i in self.cursor2:
self.fechanot = str(i[0]) #si la persona posee norificaciones se regitra en esta variabe,
self.tiponot = str(i[1]) #el tipo de notificacion puede ser grupal "G" o individual "I"
```

```
self.notificacion = str(i[2]) #Registra las notificaciones
if self.fechahoy==self.fechanot: #Compara las fechas, sin son igual ala actual y hay
#notificaciones pasa a la sigueinte etapa
if self.tiponot=='G': #Si la notificacion es grupal se imprime
#en pantalla la notificacion.
self.notificaciones.addItem(self.notificacion)

else: # de lo contrario significa que la notificacion es individual, por tanto no se
#imprime en pantalla, sino que se envia en forma de subventana
QtGui.QMessageBox.information(self, 'Notificaci n', ''+str(self.notificacion)+'')

self.con.commit()

else: #No fue reconocido con exito
# Quedan los campos vacios al terminar el proceso
self.lineEdit.setText("")
vc.release() #apaga la camara
QtGui.QMessageBox.critical(self, 'Error', 'NO FUE RECONOCIDO, INTENTE DE NUEVO')
# Mensaje de error especificando que no se registro la asistencia

cv2.imshow("Reconocimiento Facial", predict_image)
cv2.waitKey(1000)
cv2.destroyAllWindows()

else: # este else es para registrar la salida, si la cedula es encontrada, pero las fechas
#son iguales, significa que debe registrar la salida, mismo procedimiento
self.notificaciones.clear() #borrar la barra de notificaciones

vc = cv2.VideoCapture(0) #Activa la camara web
while (True):  #Inicia el programa

ret,frame = vc.read() #Se lee y guarda un frame
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #Convirtiendo la imagen a blanco y negro
faces = face_cascade.detectMultiScale(gray, 1.2, 3) #Si se encuentran rostros se buscan
#sus coordenadas y se guarda su posicion
#Se dibuja un rectangulo en las coordenas del rostro
for (x,y,w,h) in faces:

cv2.putText(frame,'Quedese QUIETO!!', (x,y), cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,250),3,1)
cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
```

```
cv2.imshow('Posicionarse en el Centro',frame) #Se muestra la imagen
if cv2.waitKey(1000): #if cv2.waitKey(10) == ord('s'):
break

cv2.destroyAllWindows()

#Comienza la captura de imagenes
start = time.time()
count = 0 #Contador
while int(time.time()-start) <= 2:
ret,frame = vc.read() #Se lee y guarda un frame
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #Para cambiar la imagen a escala de grises
faces = face_cascade.detectMultiScale(gray, 1.2, 3,
flags = cv2.CASCADE_SCALE_IMAGE, minSize=(150,150))
#Si se encuentran rostros se buscan sus
#coordenadas y se guarda su posicion
#Se dibuja un rectangulo en las coordenas del rostro
for (x,y,w,h) in faces:
cv2.putText(frame,'Capturando!', (x,y), cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,250),3,1) #Puntero
#que indica que se estan capturando las imagenes
count +=1
cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
resized_image = cv2.resize(frame[y:y+h,x:x+w], (273, 273))
if count%5 == 0:
cv2.imwrite(pathdir+'/'+'subject'+str(self.id)+'.jpg', resized_image ); #se guardan las
#fotos en la carpeta pathdir con el respectivo nombre indica

cv2.imshow('Captura de Fotos',frame) #Se muestra la imagen
cv2.waitKey(10)

cv2.destroyAllWindows()

image_paths = [os.path.join(pathdir, f) for f in os.listdir(pathdir) if f.endswith('.jpg')]
# A ade las im genes con la extensi n .jpg en image_paths
for image_path in image_paths:
predict_image_pil = Image.open(image_path).convert('L') #Se lee la imagen y se convierte
#a escala grises
predict_image = np.array(predict_image_pil, 'uint8') #Se convierte el formato de la imagen
#en una matriz numpy
```

```
faces = face_cascade.detectMultiScale(predict_image) # Detectar la cara en la imagen
for (x, y, w, h) in faces:
nbr_predicted, conf = recognizer.predict(predict_image) #Etiquetas usada para
#el reconocimiento
nbr_actual = int(os.path.split(image_path)[1].split(".")[0].replace("subject", ""))
#Se comprueba si el reconocimiento es correcto mediante la comparaci n de las
#etiquetas nbr_actual
#y nbr_predicted, y su umbral de confianza
if nbr_actual == nbr_predicted and conf<100: #Mientras mayor sea el valor de la variable
#de confianza, menos el reconocedor tiene confianza en el reconocimiento.

self.cursor.execute("SELECT hora_e FROM telecom.asistencia_telecom where cedula =
'"+str (self.id)+"' and fecha= '"+str (self.fechahoy)+"'")
# Se guardan los datos en variables para registrar la asistencia
for i in self.cursor:

self.hora_e = str(i[0]) #Registra el Nombre de la persona para ingresarlo como dato
#a la tabla de asistencia

self.hora_s1 = time.strftime("%X")
# Inserta los datos en la tabla campos
self.formato = "%H:%M:%S"
self.h1 = datetime.strptime(self.hora_s1, self.formato)
self.h2 = datetime.strptime(self.hora_e, self.formato)
self.resultado = self.h1 - self.h2
self.cursor2.execute("update telecom.asistencia_telecom set hora_s='"+str (self.hora_s1)+"' ,
horatotal='"+str (self.resultado)+"' where cedula= '"+str (self.id)+"' and fecha=
'"+str (self.fechahoy)+"' ")

# Quedan los campos vacios al guardar
self.lineEdit.setText("")
vc.release()
QtGui.QMessageBox.information(self, 'Correcto', 'Registr  exitosamente su salida.') #mensaje
#para avisar que se registro satisfactoiamente la asistencia
label_Foto = QtGui.QLabel(self)  #QLabel crea una etiqueta para almacenar texto o imagenes
label_Foto.setGeometry(80, 300, 160, 160)#setGeometry() #inserta la posicion y el tamano
#de la etiqueta
pixmap3=QtGui.QPixmap('basededatos/subject'+str(self.id)+'.jpg')#QPixmap permite
#ubicar una imagen para ser insertada dentro de la ventana
pixmap3 = pixmap3.scaled(160, 160, QtCore.Qt.KeepAspectRatio) # escala la imagen en
```

```
#100 x 100 pixeles, para su insercion en la etiqueta
label_Foto.setPixmap(pixmap3)#relaciona el objeto con la etiqueta
label_Foto.show() # muestra la imagen con las dimensiones antes se aladas

else: #No fue reconocido con exito
# Quedan los campos vacios al terminar el proceso
self.lineEdit.setText("")
vc.release() #apaga la camara
QtGui.QMessageBox.critical(self, 'Error', 'NO FUE RECONOCIDO, INTENTE DE NUEVO')
# Mensaje de error especificando que no se registro la asistencia

cv2.imshow("Reconocimiento Facial", predict_image)
cv2.waitKey(1000)

cv2.destroyAllWindows()

else:  #este else es si se encuentra no registrado como personal activo
QtGui.QMessageBox.critical(self, 'Error', 'Usted no se encuentra registrado en la base
de datos')
#Mensaje de error especificando que no esta en la base de datos
self.lineEdit.setText("")

else:   #Si la casilla esta en blanco y presiona el boton de asitencia envia
este mensaje
QtGui.QMessageBox.critical(self, 'Error', 'Debe ingresar un numero de cedula')
# Mensaje de error especificando que ya se registro
self.lineEdit.setText("")

# Se cargan los datos indicados de la tabla
self.fecha1='sdds'
self.cursor.execute("SELECT * FROM telecom.asistencia_telecom")
# Al presionar el boton lo primero es borrar todos los datos
self.lista.clear()
# Se agregan los elementos al QListWidget
for i in self.cursor:
self.id1 = str(i[0])
self.cedula1 = str(i[1])
self.nombre1 = str(i[2])
self.apellido1 = str(i[3])
self.tipopersonal = str(i[4])
```

```
self.fecha1 = str(i[5])
self.fecha2 = str(i[6])
self.hora_e1 = str(i[7])
self.hora_s1 = str(i[8])
self.horatotal = str(i[9])
self.lugar1 = str(i[10])
self.observacion = str(i[11])
if self.fechahoy==self.fecha1:
self.lista.addItem(self.nombre1 + "  " + self.apellido1 + " - " + self.fecha2 + " - "
 + self.hora_e1 + " - " + self.hora_s1 + " - " + self.horatotal
 + " - " + self.lugar1)
self.con.commit()
self.con.close()

app = QtGui.QApplication(sys.argv)
MyWindow = MyWindowClass(None)
MyWindow.show()
app.exec_()
```

Apéndice B

Algoritmo de Reconocimiento Facial

```
vc = cv2.VideoCapture(0) #Activa la camara web

while (True):  #Inicia el programa
ret,frame = vc.read() #Se lee y guarda un frame
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #Convirtiendo la imagen
#a escala grises
faces = face_cascade.detectMultiScale(gray, 1.2, 3,
flags = cv2.CASCADE_SCALE_IMAGE,
 minSize=(150,150))
#Si se encuentran rostros
#se buscan sus coordenadas y se guarda su posicion
#Se dibuja un rectangulo en las coordenas del rostro
for (x,y,w,h) in faces:
cv2.putText(frame,'Quedese QUIETO!!', (x,y), cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,250),3,1)
cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
cv2.imshow('Posicionarse en el Centro',frame) #Se muestra la imagen
if cv2.waitKey(1000):
break

cv2.destroyAllWindows()

pathdir='./fotoscapturadas' #Carpeta de captura de imagenes
```

```
os.mkdir(pathdir) #Crea la carpeta donde se encuentran las imagenes capturadas
#por la camara web

#Comienza la captura de imagenes
start = time.time()
count = 0 #Contador
while int(time.time()-start) <= 2:
ret,frame = vc.read() #Se lee y guarda un frame
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #Para cambiar la imagen a escala de grises

faces = face_cascade.detectMultiScale(gray, 1.2, 3,flags = cv2.CASCADE_SCALE_IMAGE,
minSize=(150,150))
#Si se encuentran rostros se buscan
#sus coordenadas y se guarda su posicion
#Se dibuja un rectangulo en las coordenas del rostro
for (x,y,w,h) in faces:
cv2.putText(frame,'Capturando!', (x,y), cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,250),3,1)
#Puntero que indica que se estan capturando las imagenes
count +=1
cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
resized_image = cv2.resize(frame[y:y+h,x:x+w], (273, 273))
if count%5 == 0:
cv2.imwrite(pathdir+'/'+'subject'+str(self.id)+'.jpg', resized_image ); #se guardan
#las fotos en la carpeta pathdir con el respectivo nombre i
cv2.imshow('Captura de Fotos',frame) #Se muestra la imagen
cv2.waitKey(10)

cv2.destroyAllWindows()

#Comparacion de imagenes
image_paths = [os.path.join(pathdir, f) for f in os.listdir(pathdir) if f.endswith('.jpg')]
#Anade las imagenes con la extension .jpg en image_paths
for image_path in image_paths:
predict_image_pil = Image.open(image_path).convert('L') #Se lee la imagen y se
#convierte a escala grises
predict_image = np.array(predict_image_pil, 'uint8') #Se convierte el formato de
#la imagen en una matriz numpy
faces = face_cascade.detectMultiScale(predict_image) # Detectar la cara en la imagen

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

```
nbr_predicted, conf = recognizer.predict(predict_image) #Etiquetas usada para el
#reconocimiento
nbr_actual = int(os.path.split(image_path)[1].split(".")[0].replace("subject", ""))

#Se comprueba si el reconocimiento es correcto mediante la comparacion de las etiquetas
#nbr_actual y nbr_predicted, y su umbral de confianza
if nbr_actual == nbr_predicted and conf<100:

Registra la asistencia

else:

No fue reconocido
```

Apéndice C

Código para Importar la Base de Datos de Entrenamiento

```
#Codigo para el entrenamiento de las imagenes en la base de datos
#Modulo para importar la base de datos al codigo principal
import numpy as np
from PIL import Image
import shutil

face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
recognizer = cv2.createLBPHFaceRecognizer() #Reconocedor de cara
path = './basededatos'
def images_and_labels (path):

# Se a a d e n todas las rutas de imagenes absolutas en una lista image_paths
# Se leeran las imagenes con la extension .jpg en el conjunto de entrenamiento
image_paths = [os.path.join(path, f) for f in os.listdir(path) if f.endswith('.jpg')]
images = [] # Las images que contiene la cara
labels = [] # Etiqueta que se le asigna a la imagen
for image_path in image_paths:

image_pil = Image.open(image_path).convert('L')   # Leer la imagen y
#convertir a escala de grises
image = np.array(image_pil, 'uint8')  # Convertir el formato de imagen en una matriz
#numpy
```

```
nbr = int(os.path.split(image_path)[1].split(".")[0].replace("subject", ""))
# Etiqueta de la imagen
faces = face_cascade.detectMultiScale(image)  # Detectar la cara en la imagen
# Si se detecta la cara, anexar a las imagenes con su etiqueta
for (x, y, w, h) in faces:
images.append(image)
labels.append(nbr)

cv2.waitKey(50)

return images, labels

cv2.destroyAllWindows()
```

Apéndice D

Código para el Registro de Asistencia de Entrada

```
#Se toman los datos para registrar la asistencia.
self.cedula = str(self.lineEdit.text())
self.fecha = time.strftime("%x")
self.hora_e = time.strftime("%X")
# Se cargan los datos indicados de la tabla, dicha tabla se encuentran los datos
#del personal registrado en la base de datos
self.cursor.execute("SELECT nombre1, apellido1, tipopersonal FROM telecom.personal_telecom
where cedula = '"+str (self.cedula)+"'")
# Se guardan los datos en variables para registrar la asistencia
for i in self.cursor:

self.nombre = str(i[0]) #Registra el Nombre de la persona para ingresarlo como dato
#a la tabla de asistencia
self.apellido = str(i[1])#Registra el Apellido de la Persona para ingresarlo como dato
#a la tabla de asistencia
self.tipopersonal = str(i[2])
self.lugar = 'Esc. Telecom' #Lugar Por defecto que se asigna al control de asistencia

# Inserta los datos en la tabla campos
self.cursor.execute("insert into telecom.asistencia_telecom (cedula, nombre, apellido,
tipopersonal, fecha, fechahoy, hora_e, lugar) values ('"+str (self.cedula)
+"', '"+str (self.nombre)+"' , '"+str (self.apellido)+"' , '"+str (self.tipopersonal)+"'
```

```
, '"+str (self.fecha)+"' , '"+str (self.fechabusqueda)
+"', '"+str (self.hora_e)+"' , '"+str (self.lugar)+"')")
# Quedan los campos vacios al registrar la asistencia
self.lineEdit.setText("")
vc.release() # apaga la camara

QtGui.QMessageBox.information(self, 'Correcto', 'Asistencia Registrada Satisfactoriamente')
# mensaje para avisar que se registro satisfactoriamente la asistencia
label_Foto = QtGui.QLabel(self)  #QLabel crea una etiqueta para almacenar texto o imagenes
label_Foto.setGeometry(80, 300, 160, 160)#setGeometry()
#inserta la posicion y el tama o de la etiqueta
pixmap3 = QtGui.QPixmap('basededatos/subject'+str(self.id)+'.jpg')#QPixmap permite
#ubicar una imagen para ser insertada dentro de la ventana
pixmap3 = pixmap3.scaled(160, 160, QtCore.Qt.KeepAspectRatio) # scala la imagen en
#100 x 100 pixeles, para su insercion en la etiqueta
label_Foto.setPixmap(pixmap3)#relaciona el objeto con la etiqueta
label_Foto.show() # muestra la imagen con las dimensiones antes se aladas
self.fechanot='adadfsf'#valores por defecto para las notificaciones
self.tiponot ='asdsa' #valores por defecto para las notificaciones
self.cursor2.execute("SELECT fecha, tipo, notificacion FROM telecom.notificaciones_telecom
where cedula = '"+str (self.cedula)+"'")
for i in self.cursor2:
self.fechanot = str(i[0]) #si la persona posee notificaciones se regitra en esta variabe,
self.tiponot = str(i[1]) #el tipo de notificacion puede ser grupal "G" o individual "I"
self.notificacion = str(i[2]) #Registra las notificaciones
if self.fechahoy==self.fechanot: #Compara las fechas, sin son igual ala actual y
#hay notificaciones pasa a la sigueinte etapa
if self.tiponot=='G': #Si la notificacion es grupal se imprime
#en pantalla la notificacion.
self.notificaciones.addItem(self.notificacion)

else: # de lo contrario significa que la notificacion es individual, por tanto
#no se impreme en pantalla, sino que se envia en forma de subventana
QtGui.QMessageBox.information(self, 'Notificaci n', ''+str(self.notificacion)+'')
```

Apéndice E

Código para el Registro de Asistencia de Salida

```
self.cursor.execute("SELECT hora_e FROM telecom.asistencia_telecom where cedula =
'"+str (self.id)+"' and fecha= '"+str (self.fechahoy)+"'")
# Se guardan los datos en variables para registrar la asistencia
for i in self.cursor:

self.hora_e = str(i[0]) #Registra el Nombre de la persona para ingresarlo
#como dato a la tabla de asistencia

self.hora_s1 = time.strftime("%X")
# Inserta los datos en la tabla campos
self.formato = "%H:%M:%S"
self.h1 = datetime.strptime(self.hora_s1, self.formato)
self.h2 = datetime.strptime(self.hora_e, self.formato)
self.resultado = self.h1 - self.h2
self.cursor2.execute("update telecom.asistencia_telecom set hora_s='"+str (self.hora_s1)
+"' , horatotal='"+str (self.resultado)+"' where cedula= '"+str (self.id)+"'
and fecha= '"+str (self.fechahoy)+"' ")

# Quedan los campos vacios al guardar
self.lineEdit.setText("")
vc.release()
QtGui.QMessageBox.information(self, 'Correcto', 'Registr  exitosamente su salida.')
```

```
# mensaje para avisar que se registro satisfactoriamente la asistencia

label_Foto = QtGui.QLabel(self)  #QLabel crea una etiqueta para almacenar texto o imagenes
label_Foto.setGeometry(80, 300, 160, 160)#setGeometry() inserta la posicion
#y el tamano de la etiqueta
pixmap3 = QtGui.QPixmap('basededatos/subject'+str(self.id)+'.jpg')#QPixmap permite
#ubicar una imagen para ser insertada dentro de la ventana
pixmap3 = pixmap3.scaled(160, 160, QtCore.Qt.KeepAspectRatio) # escala la imagen
#en 100 x 100 pixeles, para su insercion en la etiqueta
label_Foto.setPixmap(pixmap3)#relaciona el objeto con la etiqueta
label_Foto.show() # muestra la imagen con las dimensiones antes se aladas
```

Apéndice F

Código para Mostrar la Asistencia

```
#Mostrar la asistencia en la tabla que se encuentra en el menu principal
 # Se cargan los datos indicados de la tabla
 self.fecha1='sdds'
 self.cursor.execute("SELECT * FROM telecom.asistencia_telecom")
 # Al presionar el boton lo primero es borrar todos los datos
 self.lista.clear()
 # Se agregan los elementos al QListWidget
 for i in self.cursor:
 self.id1 = str(i[0])
 self.cedula1 = str(i[1])
 self.nombre1 = str(i[2])
 self.apellido1 = str(i[3])
 self.tipopersonal = str(i[4])
 self.fecha1 = str(i[5])
 self.fecha2 = str(i[6])
 self.hora_e1 = str(i[7])
 self.hora_s1 = str(i[8])
 self.horatotal = str(i[9])
 self.lugar1 = str(i[10])
 self.observacion = str(i[11])
 if self.fechahoy==self.fecha1:
 self.lista.addItem(self.nombre1 + "  " + self.apellido1 + " - " + self.fecha2+ " - " +
 self.hora_e1 + " - " + self.hora_s1 + " - " +self.horatotal + " - " + self.lugar1)
 self.con.commit()
 self.con.close()
```

Apéndice G

Código para Conectar la Base de Datos

```
#Conexi n de Python con la PostgreSQL (base de datos)
import psycopg2

cadenaConexion="host='localhost' dbname='basededatos' user='postgres' password
='contrase a'" #parametros para establecer la conexi n
self.con =psycopg2.connect(cadenaConexion)  #utilizando la libreria psycopg2
#se establece la conexi n
self.cursor = self.con.cursor()  # se crea un objeto con el proposito de
#manipular las variables de PostgreSQL
self.con.commit() #Se otorga el permiso para establecer la conexion
```

Apéndice H

Código en PostgreSQL

```
REATE DATABASE telecom
WITH OWNER = postgres
ENCODING = 'UTF8'
TABLESPACE = pg_default
LC_COLLATE = 'Spanish_Venezuela.1252'
LC_CTYPE = 'Spanish_Venezuela.1252'
CONNECTION LIMIT = -1;

CREATE SCHEMA telecom
AUTHORIZATION postgres;

CREATE TABLE telecom.asistencia_telecom
(
id serial NOT NULL,
cedula character varying(11) NOT NULL,
nombre character varying(50) NOT NULL,
apellido character varying(50) NOT NULL,
tipopersonal character varying(50),
fecha character varying(50) NOT NULL,
fechahoy date NOT NULL,
hora_e character varying(50) NOT NULL,
hora_s character varying(50),
horatotal character varying(50),
lugar character varying(50) NOT NULL,
observacion character varying(250),
```

```
CONSTRAINT asistencia_telecom_pkey PRIMARY KEY (id)
)
WITH (
OIDS=FALSE
);
ALTER TABLE telecom.asistencia_telecom
OWNER TO postgres;

CREATE TABLE telecom.personal_telecom
(
id serial NOT NULL,
cedula character varying(11) NOT NULL,
nombre1 character varying(50) NOT NULL,
nombre2 character varying(50),
apellido1 character varying(50) NOT NULL,
apellido2 character varying(50),
fechanac character varying(50),
nacionalidad character varying(50) NOT NULL,
telefono character varying(50) NOT NULL,
correo character varying(50) NOT NULL,
tipopersonal character varying(50) NOT NULL,
condicion character varying(50),
dedicacion character varying(50),
escalafon character varying(50),
fechareg character varying(50) NOT NULL,
CONSTRAINT personal_telecom_pkey PRIMARY KEY (cedula)
)
WITH (
OIDS=FALSE
);
ALTER TABLE telecom.personal_telecom
OWNER TO postgres;

CREATE TABLE telecom.notificaciones_telecom
(
id serial NOT NULL,
cedula character varying(11) NOT NULL,
fecha character varying(20) NOT NULL,
tipo character varying(2) NOT NULL,
notificacion character varying(450),
```

```
CONSTRAINT notificacines_telecom_pkey PRIMARY KEY (id)
)
WITH (
OIDS=FALSE
);
ALTER TABLE telecom.notificaciones_telecom
OWNER TO postgres;
```

Apéndice I

Manual de Usuario

Bienvenidos

Bienvenidos al manual de usuario del software piloto de reconocimiento facial para el control de asistencia de la Escuela de Telecomunicaciones de la Universidad de Carabobo, el cual será de ayuda para el correcto uso y funcionamiento del programa.

Descripción de la Interfaz Gráfica

Figura 9.1: Interfaz Gráfica del Software Piloto

1. Cédula: Campo en el cual el usuario debe ingresar su número de cédula.

2. Asistencia: Botón para registrar la asistencia.

3. Etiqueta de Foto: Luego de ser reconocido satisfactoriamente, el software tomará la foto de la base de datos del sujeto en cuestión, y la insertará en la etiqueta de fotos.

4. Lista de Asistencia: Al registrar satisfactoriamente la asistencia, se cargan los datos respectivos a la tabla de asistencia, en pantalla se mostrará; Nombre, Apellido, Fecha, Hora de entrada, Hora de salida, Cantidad de horas laboradas, lugar donde se registró la asistencia.

5. Lista de Notificaciones: En dicha lista se muestran las notificaciones del día, es importante recalcar que, se muestran las notificaciones grupales, es decir, dirigido a un público en común, ya sea profesores, administrativo o estudiantes y solo se mostrará a la hora de entrada del personal.

6. Administración: Al seleccionar esta opción, le pedirá al usuario que se autentique debido que el acceso está restringido para el administrador.

7. Ayuda: En esta opción, el usuario podrá obtener documentación respecto al uso eficiente del software.

8. Acerca de: Se mostrará una ventana mostrando datos de los responsables en la elaboración del software.

Funcionamiento

Registro de Asistencia con Detección y Reconocimiento Facial

El usuario debe ingresar su número de cédula en el campo número 1 y hacer clic en el botón asistencia;

Si el usuario presiona el botón asistencia sin haber ingresado un número de cédula aparecerá un mensaje de error en pantalla, el cual se observa en la figura 9.2.

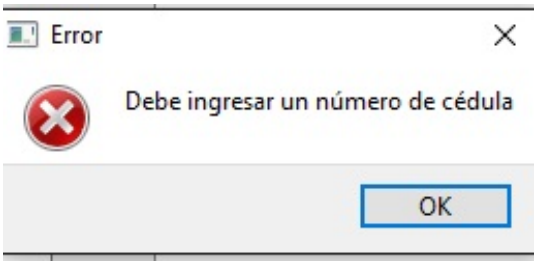


Figura 9.2: Ventana de error al no ingresar número cédula

Si el usuario ingresa un número diferente a su cédula de identidad, y dicho número no se encuentra registrado en la base de datos aparecerá el mensaje que observa en la figura 9.3.

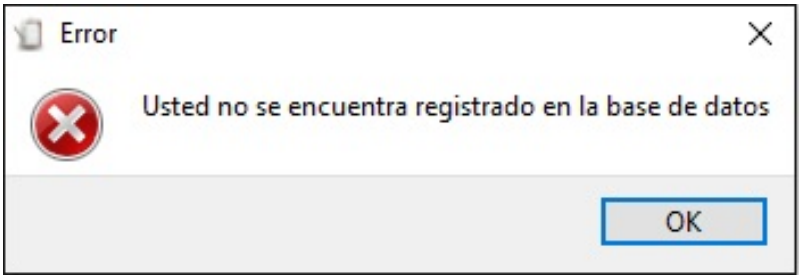


Figura 9.3: Ventana de error si la cédula no se encuentra en la base de datos.

Si el usuario ingresa un número de cédula que se encuentre registrado en la base de datos, al presionar el botón asistencia iniciará el proceso de captura de foto, el cual se observa en la figura 9.4.

Figura 9.4: Ventana para capturar foto

Consideraciones para la captura de foto:

• Al presionar el botón asistencia automáticamente se inicia el proceso de captura de foto, para dicha captura tomará un lapso de 4s.

• Dentro de la ventana de captura tiene que aparecer un recuadro azul alrededor de la cara, con un mensaje que indica que se está capturando la foto tal como se muestra en la figura anterior, pasado los 4 segundos terminara el proceso.

• Si no aparece el recuadro azul alrededor de la cara, espere que finalice el proceso e intente nuevamente el proceso de captura.

• Debe mantenerse quieto durante el proceso de captura, para que el reconocimiento sea óptimo.

• Finalizado el proceso de captura aparecerá uno de los siguientes mensajes:

Si el usuario fue perfectamente reconocido y es la primera asistencia del día, el sistema lo toma como asistencia de entrada para el día en cuestión. Ver figura 9.5.

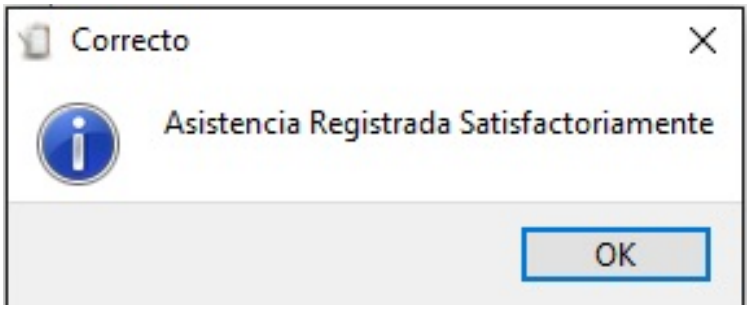


Figura 9.5: Ventana de asistencia registrada correctamente

Si el usuario fue perfectamente reconocido y ya ha registrado la asistencia de entrada, el sistema lo toma como asistencia de salida para el día en cuestión. Es importante señalar que, se puede registrar la asistencia de salida tantas veces el usuario lo desee, siempre y cuando sea el mismo día. Ver figura 9.6.

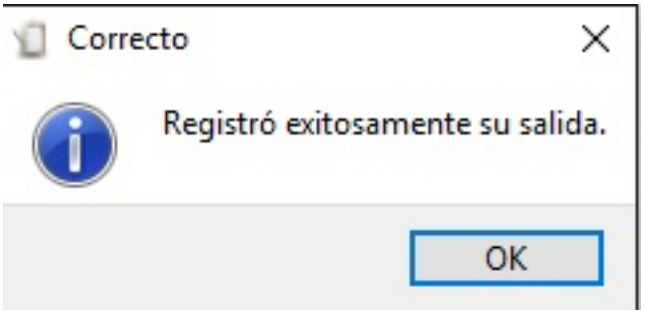


Figura 9.6: Ventana para registrar asistencia de salida

Si el usuario no fue perfectamente reconocido aparecerá el siguiente mensaje que se observa en la figura 9.7.

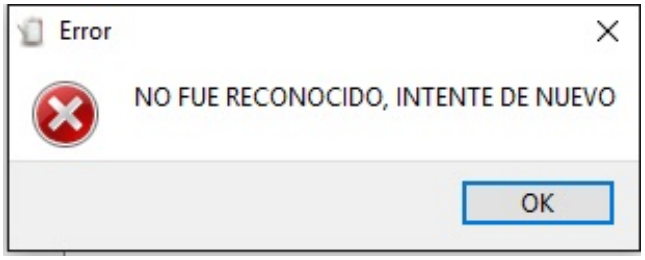
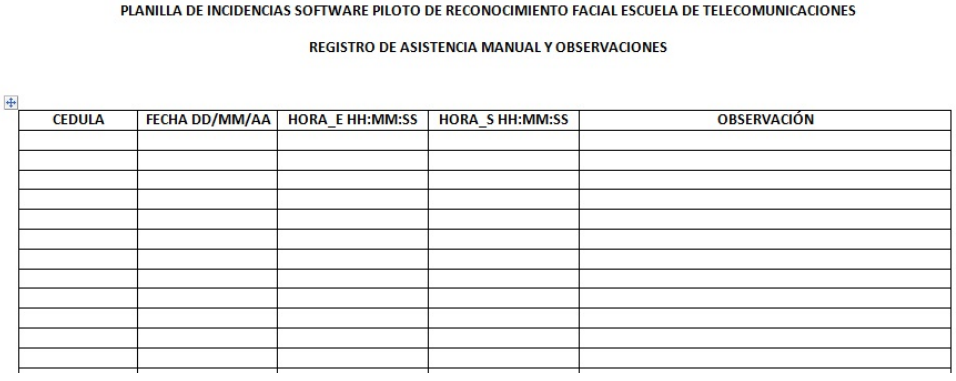


Figura 9.7: Ventana de error al no reconocer a la persona

Por tanto el usuario debe repetir el proceso indicado anteriormente, si persiste el problema con el reconocimiento, el usuario debe hablar con el administrador de sistema, y llenar la planilla de incidencias, para que su asistencia pueda ser llenada manualmente. Ver figura 9.8 y apéndice L.


PLANILLA DE INCIDENCIAS SOFTWARE PILOTO DE RECONOCIMIENTO FACIAL ESCUELA DE TELECOMUNICACIONES

REGISTRO DE ASISTENCIA MANUAL Y OBSERVACIONES

CEDULA	FECHA DD/MM/AA	HORA_E HH:MM:SS	HORA_S HH:MM:SS	OBSERVACIÓN

Figura 9.8: Planilla de incidencias

Si el registro se hizo de forma manual, debe culminar de forma manual para ese día.

Del mismo modo, el administrador de sistema actualizará su foto de la base de datos para solventar dicho inconveniente.

Registro de Asistencia

Finalizado el proceso de autenticación de usuarios por reconocimiento facial, y si efectivamente fue reconocido, la asistencia se registra de forma automática, además se muestra en pantalla dicha acción, donde el usuario puede verificar que realmente fue registrada su asistencia.

Si la asistencia es de entrada, aparecerá en pantalla lo siguiente que se observa en la figura 9.9:

Figura 9.9: Ventana de asistencia de entrada

Donde se puede constatar, datos personales, fecha de registro de asistencia y la hora de entrada.

Por otro lado, si registró la salida, aparecerá de la siguiente manera. Ver figura 9.10.

Figura 9.10: Ventana de asistencia de salida

Donde se puede verificar que aparece la hora de salida y la cantidad de horas laboradas en ese día.

Visualizar Notificaciones

Hay dos tipos de notificaciones, una llamada individual y la otra grupal. Cabe destacar que, dichas notificaciones solo se podrán visualizar al momento de registrar la asistencia de entrada.

Las notificaciones individuales (figura 9.11) van dirigidas como su palabra lo indica a un individuo en particular y se mostrará como una ventana emergente posterior a la ventana con el mensaje de registro de asistencia satisfactorio.

Figura 9.11: Ventana de notificaciones individuales

Por otro lado, las notificaciones grupales (figura 9.12) van dirigidas a todo el personal, al igual que las individuales solo se mostrará al registrar la asistencia de entrada y se mostrará en pantalla, tal como se especificó al comienzo.

Figura 9.12: Ventanas de notificaciones grupales

Administrador del Sistema

Al presionar el botón administrar aparecerá la siguiente ventana que se observa en la figura 9.13:

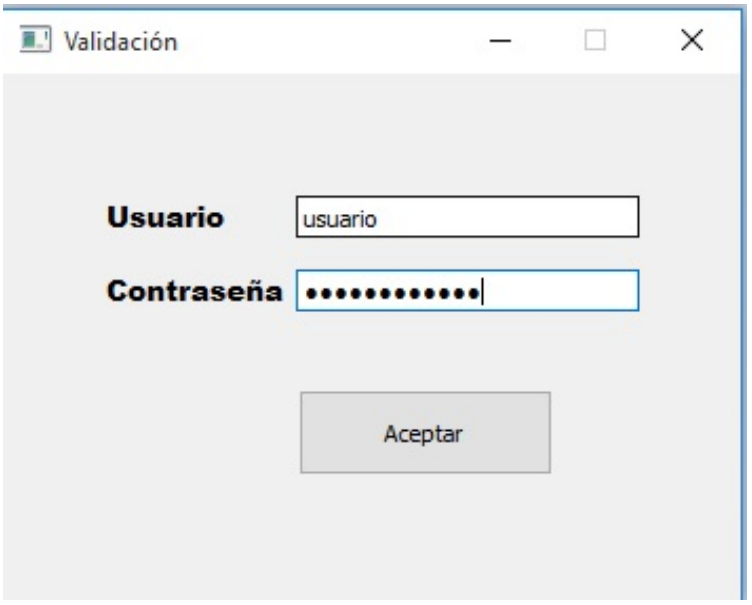


Figura 9.13: Ventana del botón administrar

Apéndice J

Manual de Administrador

Bienvenidos

Bienvenidos al manual de administrador del software piloto de reconocimiento facial para el control de asistencia de la Escuela de Telecomunicaciones de la Universidad de Carabobo, el cual será de ayuda para el correcto manejo y gestión del programa.

Descripción de la Interfaz Gráfica

Figura 10.1: Ventana de administración

1. Botón Agregar Personal: Lleva a una ventana donde el administrador puede agregar nuevo personal a la base de datos.

2. Botón Modificar Personal: Lleva a una ventana donde se puede modificar los datos del personal incluyendo la foto en la base de datos.

3. Botón Asistencia Manual: El cual direcciona a una nueva ventana, donde el administrador puede agregar la asistencia de forma manual sin pasar por el reconocimiento facial.

4. Botón Generar Data: El cual lleva a la ventana donde se pueden descargar las asistencia del personal.

5. Botón Notificaciones: Direcciona a la ventana donde se podrá interactuar con la ventana principal, enviando mensajes al personal.

6. Manual de Usuario: El cual abrirá un archivo con la documentación necesaria para la administración del personal.

7. Botón Salir: Cierra la ventana de administración.

Agregar Personal

Figura 10.2: Ventana para agregar personal

Como se muestra en la figura 10.2, se tiene un menú tipo formulario, donde el administrador rellenara hasta completar todo los campos, cuenta con 9 líneas de texto que deben ser llenadas en algunos casos de forma obligatoria, y con 4 combo-box (opciones desplegables), donde se podrá seleccionar la opción correspondiente a cada persona.

Evento del Botón Registrar

Al presionar el botón registrar, mostrará un mensaje según sea el caso que a continuación se detalla.

- Si no rellena los campos obligatorios mandara un mensaje advirtiendo que se deben llenar los campos obligatorios. Ver figura 10.3.

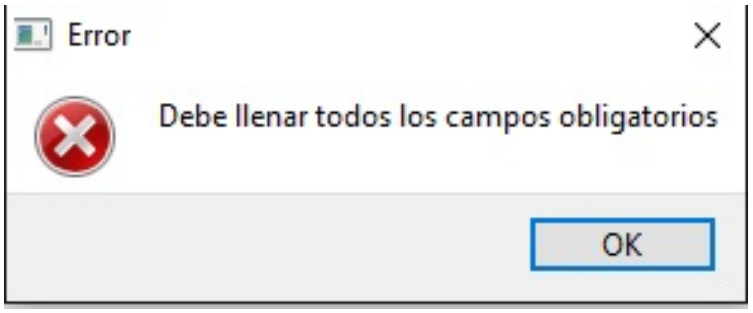


Figura 10.3: Ventana de error al no llenar los campos obligatorios

- Rellena todo los campos, pero la cédula ya se encuentra registrada en la base de datos. Ver figura 10.4.

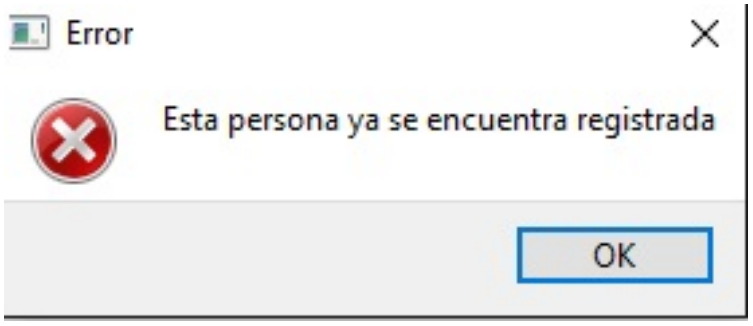


Figura 10.4: Ventana de error si la persona ya se encuentra registrada

- Rellena todos los campos exitosamente, además la cédula no se encuentra registrada, en tal caso se registra exitosamente los datos. Ver figura 10.5.

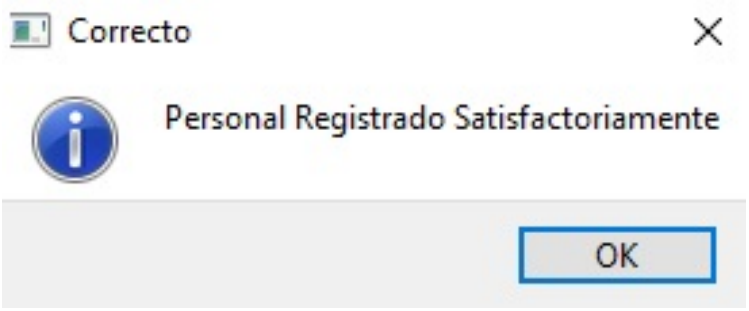


Figura 10.5: Ventana de personal registrado con éxito

Si el registro de datos fue exitoso, al presionar el botón OK, se activara la cámara inmediatamente para hacer el registro fotográfico de la persona. Ver figura 10.6.

Figura 10.6: Ventana para presionar s cuando se este listo

Cuando aparezca el recuadro azul alrededor de la cara el usuario se debe presionar la tecla s para hacer la captura de la foto.

Es importante destacar que, para este proceso es importante que el usuario se mantenga en una posición estable, para evitar que la captura de foto sea defectuosa. Ver figura 10.7

Figura 10.7: Ventana para la captura de fotos

Luego de presionar la tecla s, se hace la captura de foto y debe aparecer el recuadro azul con el mensaje capturando. Al terminar el proceso aparecerá un mensaje anunciando que el registro fue exitoso como se observa en la figura 10.8.

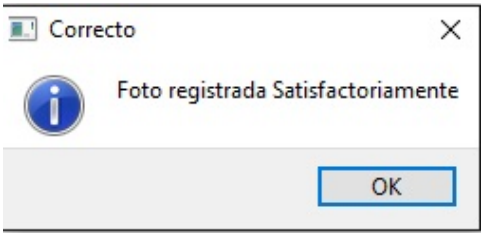


Figura 10.8: Ventana de la foto registrada con éxito

Modificar Personal

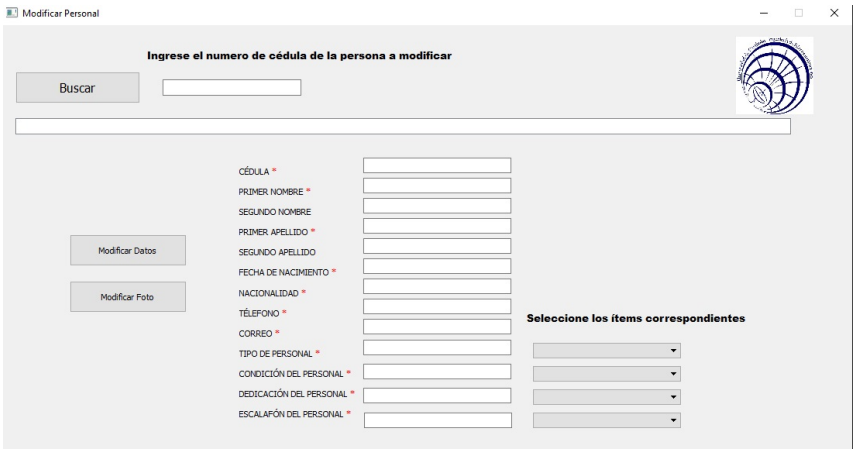


Figura 10.9: Ventana de modificar personal

El menú de modificar (figura 10.9), está diseñado para cambiar parte de los datos del personal, además de actualizar el registro fotográfico de este último y está estructurado en dos partes; una para la búsqueda de personal y la otra para modificar.

Con respecto a la búsqueda, se debe ingresar la cédula de la persona en cuestión en el cuadro de texto ubicado en la parte superior izquierda. Al presionar el botón buscar, si la cédula es encontrada en la base de datos mostrara los datos en una lista, además de rellenar automáticamente las casillas ubicadas en la parte media e inferior, de no encontrar un registro asociado a la cédula mandara el siguiente mensaje que se observa en la figura 10.10.

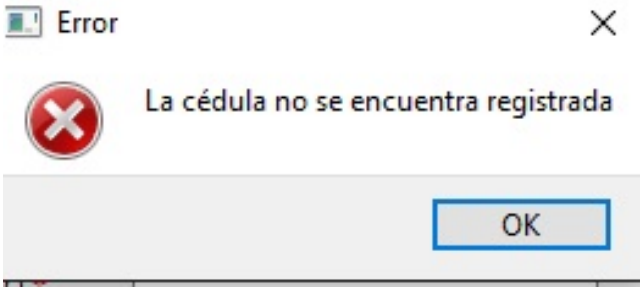


Figura 10.10: Ventana de error cédula no registrada

Evento del Botón Modificar Datos

Luego de haber realizado la búsqueda con éxito, el administrador podrá ir cambiando los datos, después de terminar el proceso de modificación debe presionar el botón modificar datos, si no ha dejado campos obligatorios vacíos se modificara correctamente los datos. Ver figura 10.11.

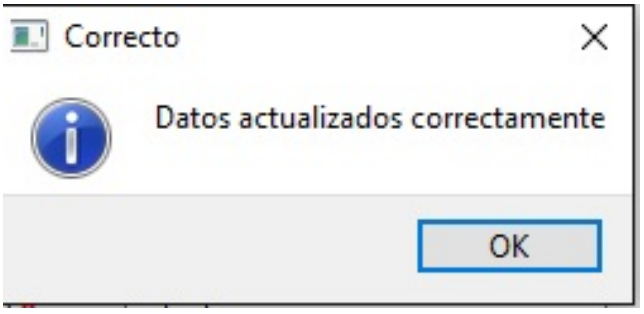


Figura 10.11: Ventana de datos actualizados con éxito

Por su lado, si dejo algún campo obligatorio vacío aparecerá el siguiente mensaje que se observa en la figura 10.12.

Figura 10.12: Ventana de error al no llenar campos obligatorios

Evento del Botón Modificar Foto

Al igual que el botón de modificar datos, primero debe hacer la búsqueda de la persona a la cual se le quiere actualizar la foto, luego de que se encuentre satisfactoriamente al presionar el botón modificar foto, se abrirá una ventana con la cámara activa. Ver figura 10.13.

Figura 10.13: Ventana para prepararse para la captura de fotos

Cuando aparezca el recuadro azul alrededor de la cara el usuario se debe presionar la tecla s para hacer la captura de la foto.

Es importante destacar que, para este proceso es importante que el usuario se mantenga en una posición estable, para evitar que la captura de foto sea defectuosa. Ver figura 10.14.

Figura 10.14: Ventana que indica la captura de fotos

Luego de presionar la tecla s, se hace la captura de foto y debe aparecer el recuadro azul con el mensaje capturando. Al terminar el proceso aparecerá un mensaje anunciando que fue modificada exitosamente la foto. Ver figura 10.15.

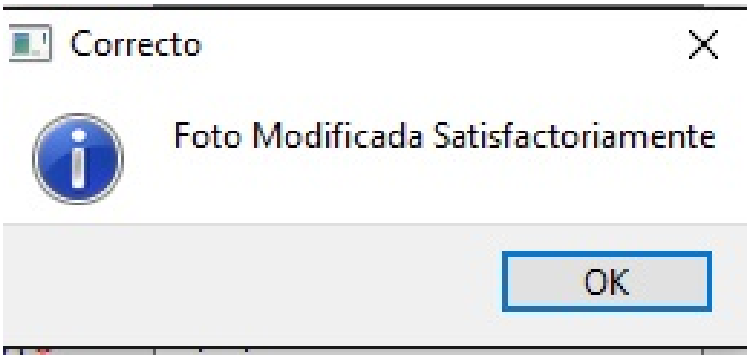


Figura 10.15: Ventana de foto modificada con éxito

Al contrario, si no ha realizado la búsqueda de una persona y presiona el botón modificar foto aparecerá el siguiente mensaje que se observa en la figura 10.16.

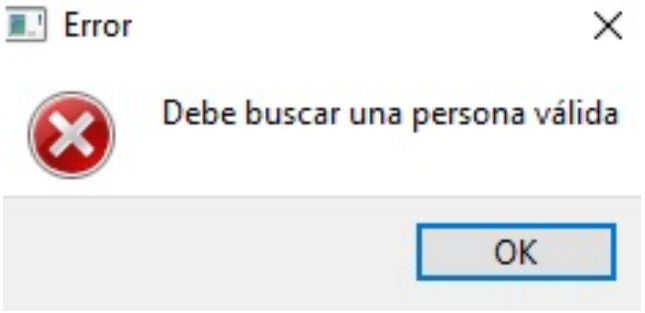


Figura 10.16: Ventana de buscar una persona válida

Asistencia Manual

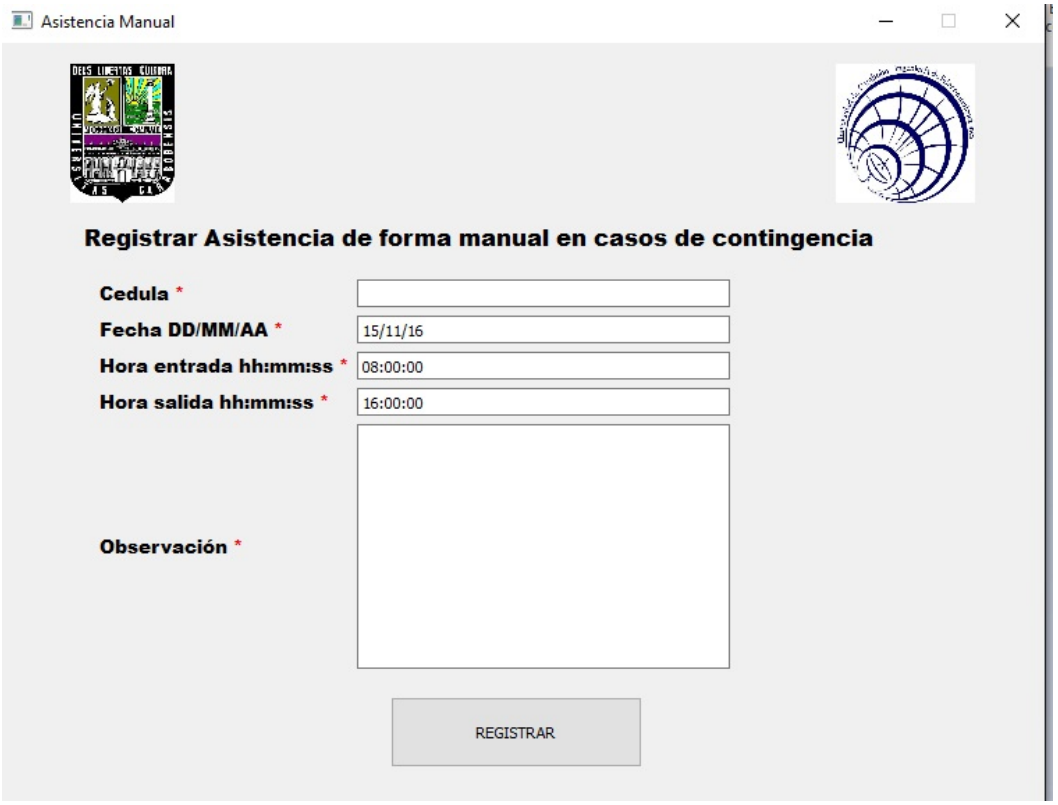


Figura 10.17: Ventana para el registro de asistencia manual

Para realizar el proceso de asistencia manual (figura 10.17) se requiere trabajar con la hoja de incidencia, para extraer datos como la hora de entrada, hora de salida, la fecha de asistencia y la observación asociada a la incidencia.

Evento del Botón Registrar

Se deben llenar todos los campos obligatorios, si presiona el botón registrar sin haber ingresado la cédula arrojara el siguiente mensaje que se observa en la figura 10.18.

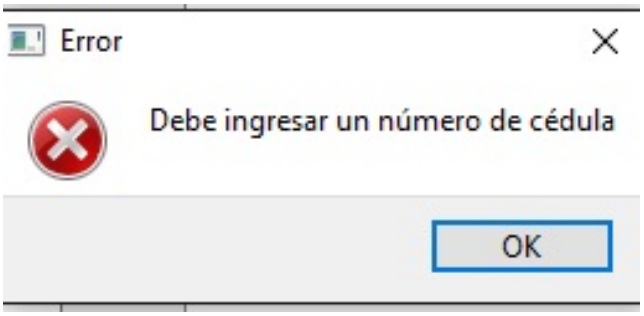


Figura 10.18: Ventana de error al no ingresar cédula

Si ingresa un numero de cédula y no se encuentra registrada, se mostrará el siguiente mensaje que se observa en la figura 10.19.

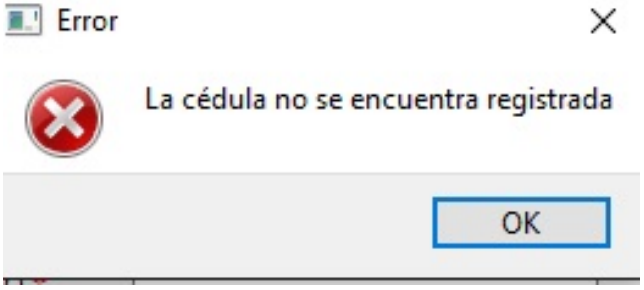


Figura 10.19: Ventana de error si la cédula no está registrada

Si la cédula es correcta, la asistencia manual será registrada exitosamente que se observa en la figura 10.20.

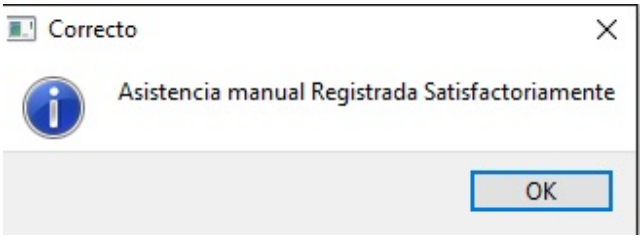


Figura 10.20: Ventana de asistencia manual registrada con éxito

En el cuadro de observaciones (figura 10.21), por defecto se registra la fecha y hora en que se realizó la asistencia manual.

Nota: Debe tener sumo cuidado en el manejo de la fecha, dado que si no ingresa la fecha correcta, registrara la asistencia en una fecha diferente.

fechahoy date	hora_e character varying(50)	hora_s character varying(50)	horatotal character varying(50)	lugar character varying(50)	observacion character varying(250)
2016-11-19	07:57:58	19:51:41	11:53:43	Esc. Telecom	
2016-11-20	20:03:22	20:50:21	0:46:59	Esc. Telecom	
2016-11-21	16:20:48	16:23:05	0:02:17	Esc. Telecom	
2016-11-22	11:18:35	11:22:36	0:04:01	Esc. Telecom	
2016-11-23	12:01:22			Esc. Telecom	
2016-11-15	08:00:00	16:00:00	8:00:00	Esc. Telecom	11/23/16 17:05:05 ,
2016-11-15	08:00:00	16:00:00	8:00:00	Esc. Telecom	11/23/16 17:19:25 ,
2016-11-15	08:00:00	16:00:00	8:00:00	Esc. Telecom	11/23/16 17:22:49 ,AQUI SE MUESTA LA NOTIFICACION

Figura 10.21: Tabla extraída de la base de datos para visualizar las observaciones

Generar Data

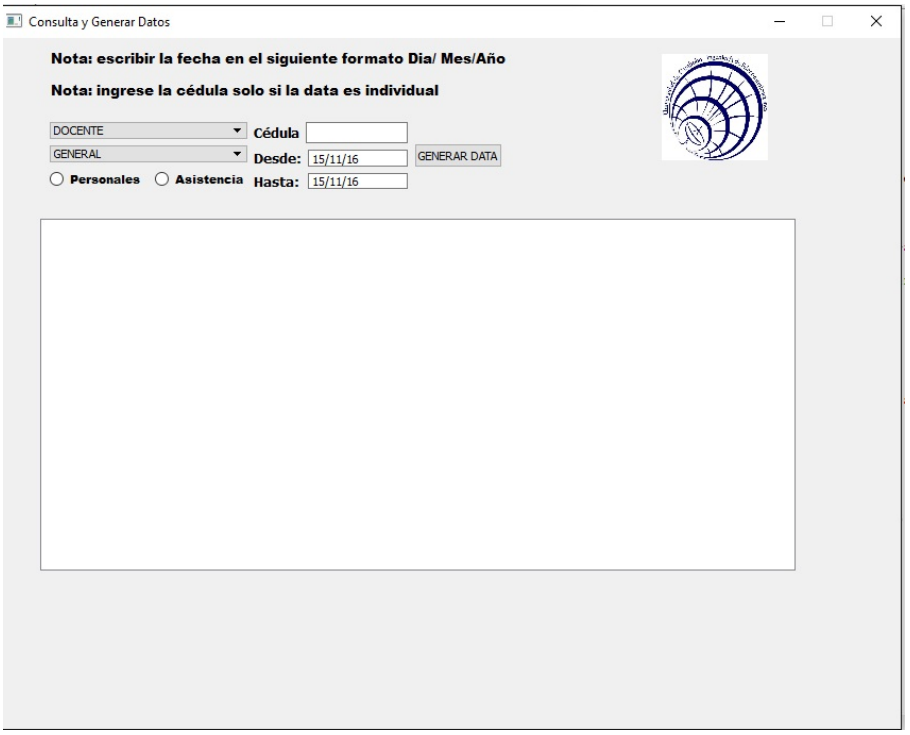


Figura 10.22: Ventana de generar data

En el menú de generar data (figura 10.22), se tienen varias opciones disponibles para la data, se puede elegir entre datos del personal grupal o individual según se desee, y por otro lado la data de asistencia, ya sea de forma grupal o individual según convenga.

Además, para mayor comodidad, esta filtrado por tipo de personal, ya sea Docente, Administrativo o Estudiantes.

Se cuenta con dos combobox (opciones desplegables), con el primero se selecciona el tipo de personal, Docente, Administrativo, Estudiante. Con el segundo se selecciona si se desea la data general o individual.

Luego se selecciona el radioboton correspondiente a la acción a ejecutar, si se desea los datos del personal se selecciona personal, en cambio si desea la data de asistencia se selecciona el botón asistencia.

Para el caso que requiera los datos de un usuario en particular, debe ingresar el número de cédula en la casilla correspondiente, de lo contrario mostrara el siguiente mensaje (figura 10.23).

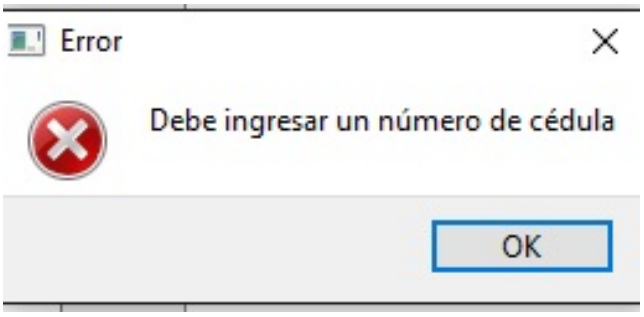


Figura 10.23: Ventana de error al no ingresar cédula

Para el caso que se quiera la data de asistencia en cualquiera de sus combinaciones, debe seleccionar un rango de fecha.

Al presionar el botón generar, si encontró un resultado satisfactorio retornara el siguiente mensaje que se observa en la figura 10.24.

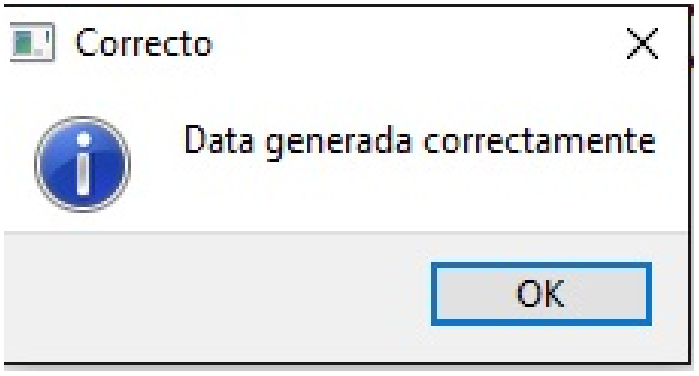


Figura 10.24: Ventana de data generada con éxito

La data se genera en un archivo de tipo XML (Excel) de fácil manejo y se guardará en la carpeta llamada reporte, ubicada junto al ejecutable del programa. Ver figura 10.25.

B	C	D	E	F	G	H	I	J	K	L
cedula	nombre	apellido	tipo personal	fecha	fechahoy	hora e	hora s	horatotal	lugar	observaciones
21136637	Elvis	Romero	DOCENTE	11/19/16	2016-11-19	07:57:58	19:51:41	11:53:43	Esc. Telecom	
21136637	Elvis	Romero	DOCENTE	11/20/16	2016-11-20	20:03:22	20:50:21	0:46:59	Esc. Telecom	
21136637	Elvis	Romero	DOCENTE	Asis manual	2016-11-15	08:00:00	16:00:00	8:00:00	Esc. Telecom	11/23/16 17:05:05 ,
14	aplicacion	vaplicacion	DOCENTE	Asis manual	2016-11-15	08:00:00	16:00:00	8:00:00	Esc. Telecom	11/23/16 17:19:25 ,
14	aplicacion	vaplicacion	DOCENTE	Asis manual	2016-11-15	08:00:00	16:00:00	8:00:00	Esc. Telecom	11/23/16 17:22:49 ,AQUI SE MUESTA LA NOTIFICACION
14	aplicacion	vaplicacion	DOCENTE	Asis manual	2016-11-15	08:00:00	16:00:00	8:00:00	Esc. Telecom	11/23/16 17:23:13 ,

Figura 10.25: Data generada

Además en pantalla también se mostrarán los datos, para que el administrador a priori los verifique. Ver figura 10.26.

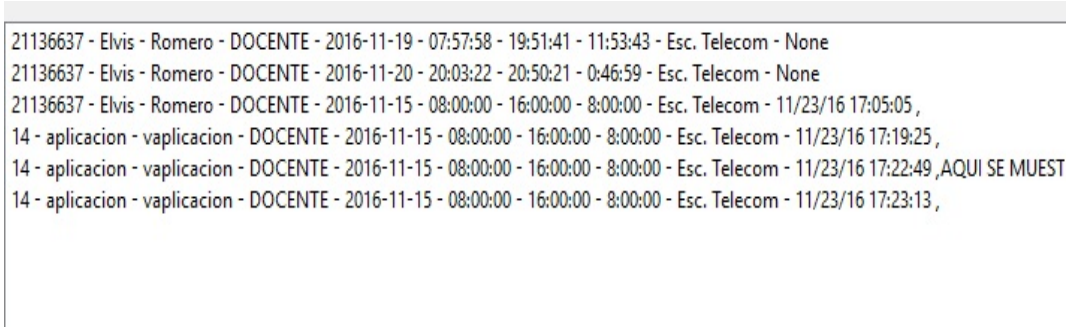


Figura 10.26: Datos mostrados en pantalla

Si no se encontraron datos en el rango de fecha se mostrará el siguiente mensaje de la figura 10.27:

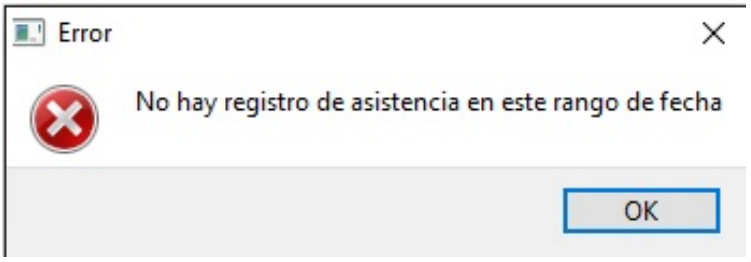


Figura 10.27: Ventana de error si no hay registro de asistencia en ese rango

Notificaciones

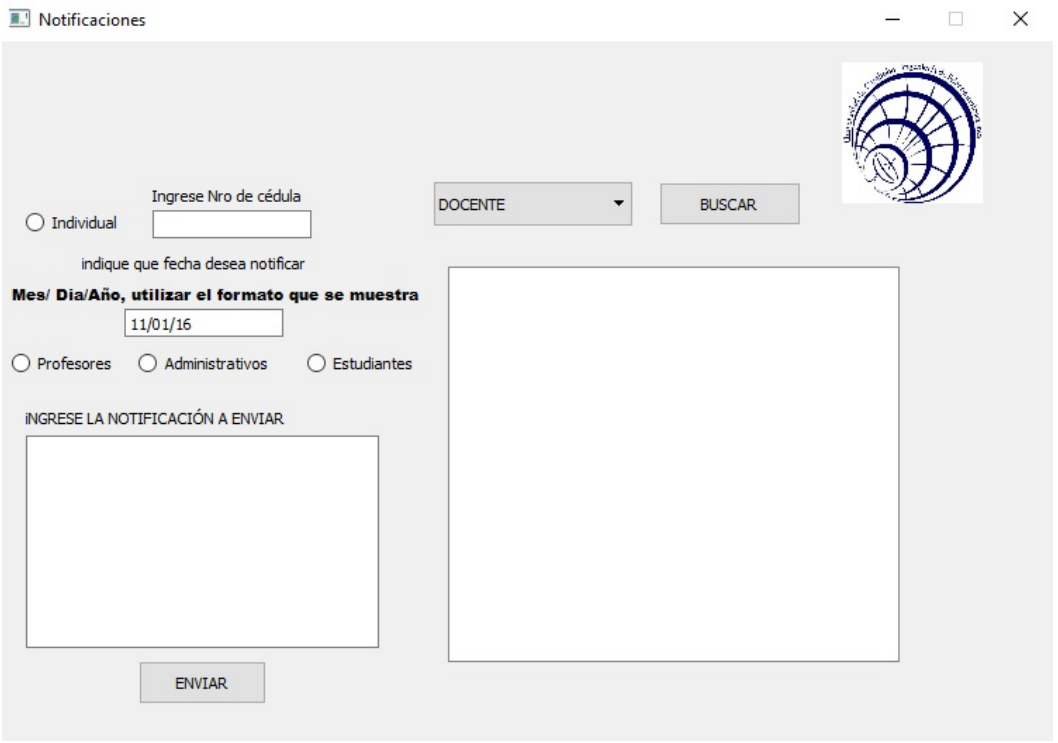


Figura 10.28: Ventana para las notificaciones

El menú notificaciones figura 10.28, interactúa directamente con la pantalla principal, está compuesta principalmente por 4 botones independientes entre sí, en el cual se puede seleccionar para enviar notificaciones de forma individual, o de forma grupal según convenga.

Y se debe ingresar la fecha que se quiere mostrar en la pantalla principal. Es muy importante que respete el formato tal cual está para que pueda aparecer la notificación.

Para las notificaciones individuales, se debe ingresar un número de cédula correcto, para ello se dispone de un botón buscar ubicado en la parte superior derecha de la pantalla, el cual podrá extraer la cédula del personal al que se le desea enviar la notificación, si seleccionó el botón individual y no ingresa un número de cédula aparecerá el siguiente mensaje que se observa en la figura 10.29:

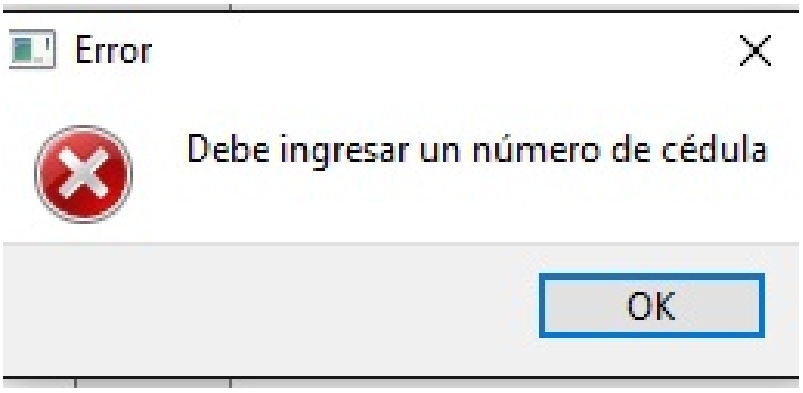


Figura 10.29: Ventana de error, debe ingresar cédula

Si ingresó el número de cedula, pero es incorrecto se mostrará el siguiente mensaje (figura 10.30).

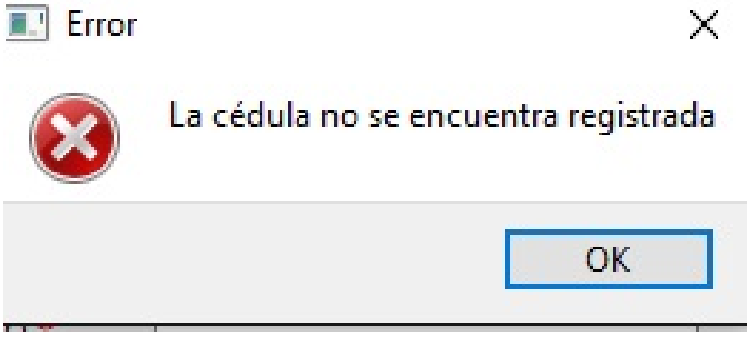


Figura 10.30: Ventana de error cédula no registrada

Si el número de cédula es correcto pero el campo de notificación está vacío, se mostraré el siguiente mensaje (figura 10.31):

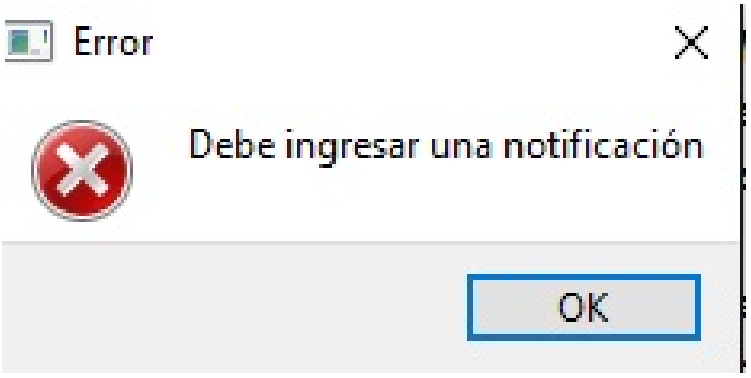


Figura 10.31: Ventana de error para notificaciones

Si el número de cédula es correcto, y el campo de notificaciones no está vacío, la notificación será enviada exitosamente. Ver figura 10.32. Es importante que la fecha sea escrita en el formato tal cual se muestra en pantalla, de lo contrario se puede producir errores en el envío de la notificación.

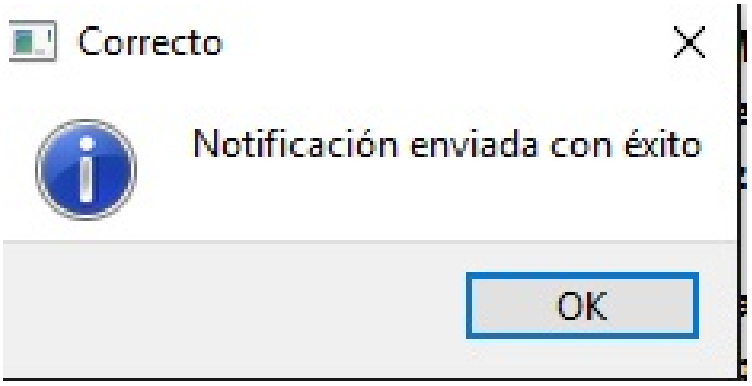


Figura 10.32: Ventana de notificación enviada con éxito

Para el caso de notificaciones generales, se debe seleccionar una opción, ya sea docente, administrativo o estudiante, además de la fecha en la cual se quiere que aparezca la notificación. Importante: se debe respetar el formato de la fecha, de lo contrario se puede producir errores en el envío de la notificación si tiene seleccionada una de las tres opciones correspondiente a notificación grupal y presiona el botón enviar.

Si el cuadro de notificación está vacío aparece el siguiente mensaje figura 10.33.

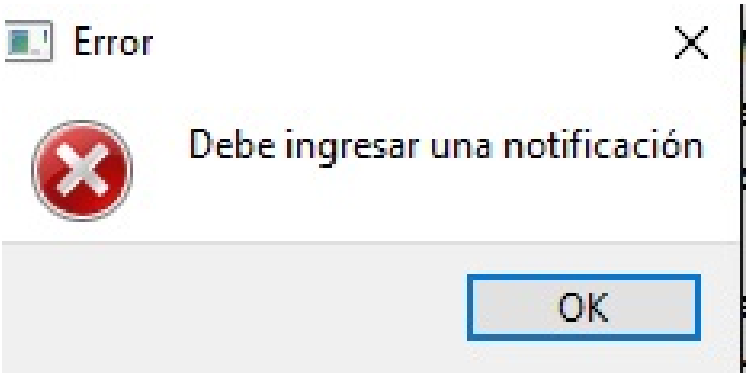


Figura 10.33: Ventana de error, ingresar una notificación

De lo contrario significa que el campo notificaciones no está vacío, lo que implica que la notificación será enviada con éxito. Ver figura 10.34.

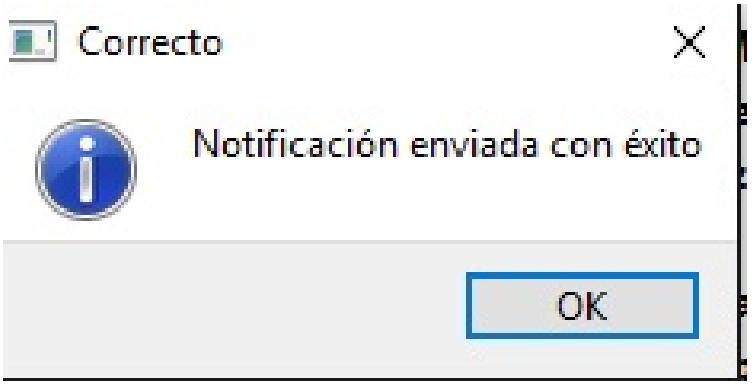


Figura 10.34: Ventana de notificación enviada con éxito

Manual de Usuario

Al presionar dicho botón se abrirá el documento manual de usuario, donde se especifica el uso de cada ventana.

Salir

Cierra la ventana de administración.

Apéndice K

Manual de Instalación

Bienvenidos

Bienvenidos al manual de instalación del software piloto de reconocimiento facial para el control de asistencia de la Escuela de Telecomunicaciones de la Universidad de Carabobo, el cual será de ayuda para la correcta ejecución del programa.

Requerimientos para la Ejecución

- Manejador de Base de datos PostgreSQL
- Cámara Web
- Software de reconocimiento facial
- PC con Sistema Operativo Windows XP, 7 o superior (32 bits o 64 bits).

Instalación del Manejador de Base de Datos

Paso 1: Descargar desde la página oficial de PostgreSQL:

http://www.enterprisedb.com/products-services-training/pgdownload#windows.

En la carpeta que contiene el software desarrollado se proporciona la versión 9.5 para Windows de 64bits, si el usuario requiere otra versión la puede descargar.

Paso 2: Ejecutar como administrador el ejecutable de PostgreSQL, le aparecerá una ventana como la que se muestra a continuación. Ver figura 11.1.

Figura 11.1: Ventana del instalador de PostgreSQL

Presione el botón siguiente, seleccione la ubicación para la instalación (figura 11.2). Puede dejar la que tiene por defecto y presione siguiente.

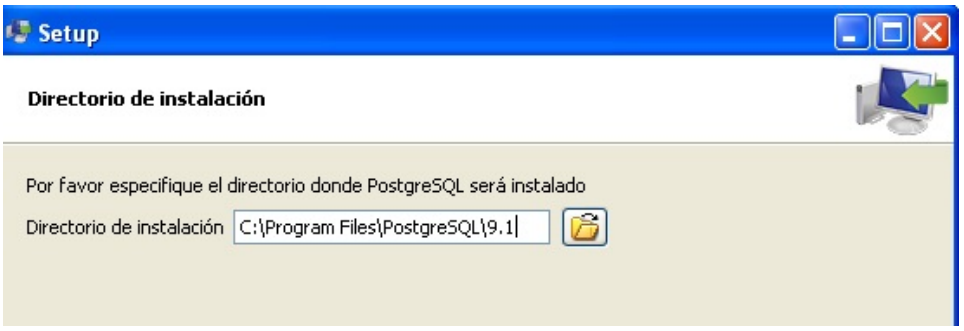


Figura 11.2: Elección del directorio de instalación

Paso 3: Seleccione la ubicación de las tablas generadas (figura 11.3), puede dejar la que trae por defecto, presione siguiente.

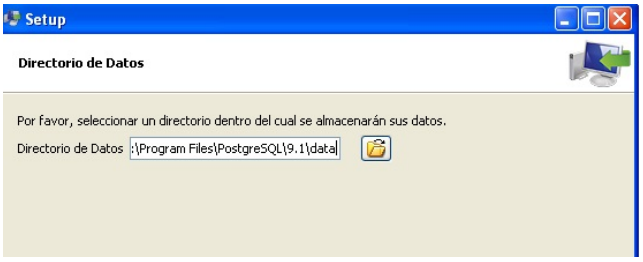


Figura 11.3: Selección del directorio de datos

Paso 4: Seleccione una contraseña de superusuario (figura 11.4). Nota: este valor es de suma importancia al momento de establecer la conexión con el software de reconocimiento facial, deben coincidir ambas contraseñas. Luego de establecida la contraseña presione siguiente.

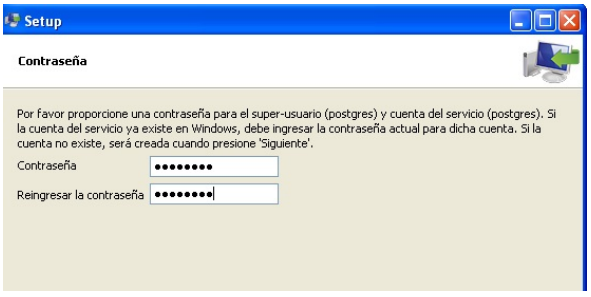


Figura 11.4: Configuración de contraseña

Paso 5: Seleccione el número de puerto (figura 11.5), puede dejar el que trae por defecto y presione siguiente.

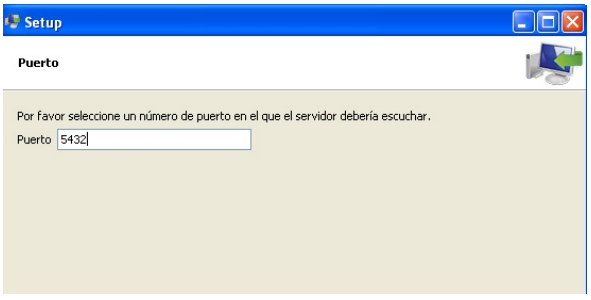


Figura 11.5: Selección de número de puerto

Paso 6: Seleccione la región correspondiente al país. Presione siguiente. Ver figura 11.6.

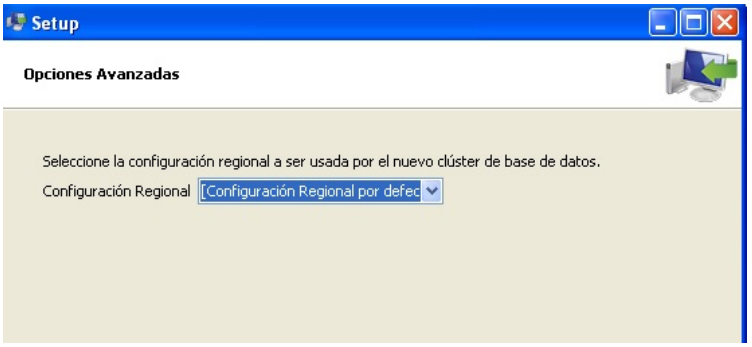


Figura 11.6: Configuración regional

Paso 7: Listo para instalar. Presione siguiente para que inicie la instalación. Ver figura 11.7.

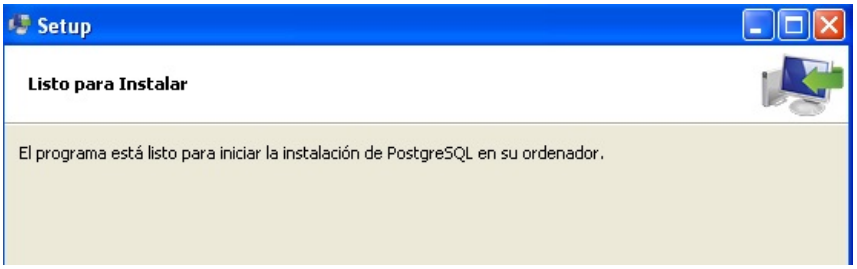


Figura 11.7: Inicio de instalación

Paso 8: Espere que finalice la instalación y presione terminar. Ver figura 11.8.

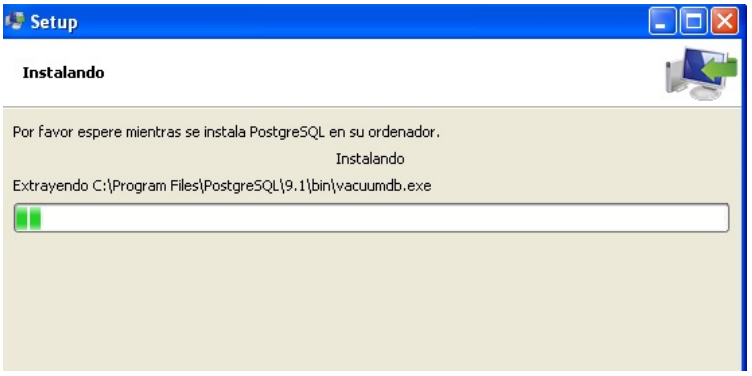


Figura 11.8: Cargando instalación

Luego de haber instalado satisfactoriamente el manejador de bases de datos, se deben crear los elementos necesarios para el funcionamiento del software.

Para ello se debe abrir el programa PgAdmin III con la intención de crear las tablas. Ver figura 11.9.

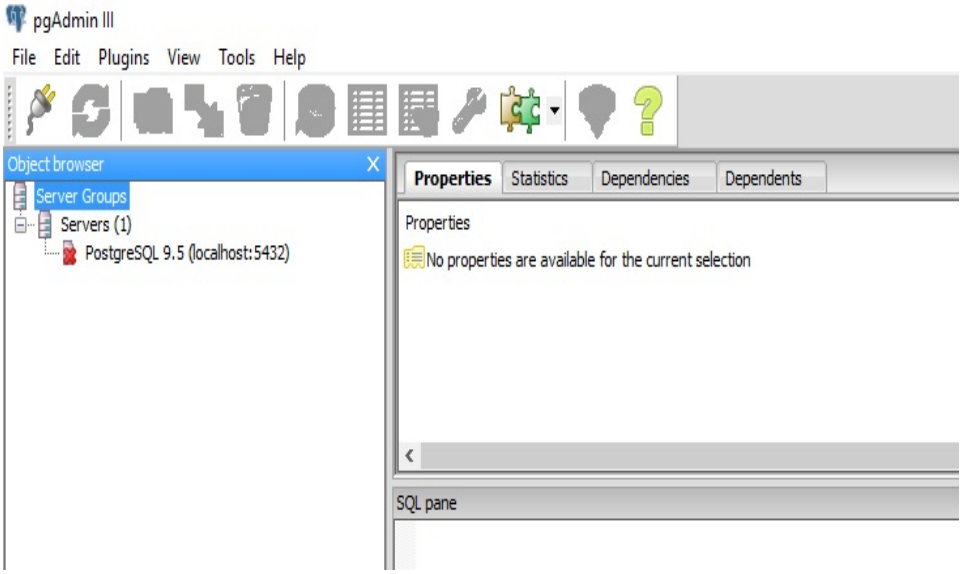


Figura 11.9: Programa PgAdmin III

Se da doble clic en el servidor llamado Postgres SQL, el cual le solicitará la contraseña preestablecida en la instalación figura 11.10.

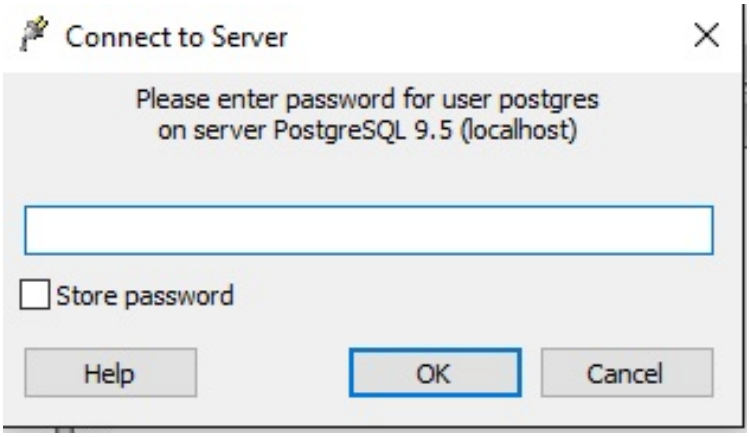


Figura 11.10: Conexión con servidor

Abrimos un Query para crear los elementos necesarios.

1. Creamos una nueva base de datos con el siguiente código:

```
CREATE DATABASE telecom
WITH OWNER = postgres
ENCODING = 'UTF8'
TABLESPACE = pg_default
LC_COLLATE = 'Spanish_Venezuela.1252'
LC_CTYPE = 'Spanish_Venezuela.1252'
CONNECTION LIMIT = -1;
```

Ejecutando este código se creará la base de datos llamada 'telecom'.

Luego ejecutamos el siguiente código:

```
CREATE SCHEMA telecom
AUTHORIZATION postgres;

CREATE TABLE telecom.asistencia_telecom
(
id serial NOT NULL,
cedula character varying(11) NOT NULL,
nombre character varying(50) NOT NULL,
apellido character varying(50) NOT NULL,
tipopersonal character varying(50),
fecha character varying(50) NOT NULL,
fechahoy date NOT NULL,
hora_e character varying(50) NOT NULL,
hora_s character varying(50),
horatotal character varying(50),
lugar character varying(50) NOT NULL,
observacion character varying(250),
CONSTRAINT asistencia_telecom_pkey PRIMARY KEY (id)
)
WITH (
OIDS=FALSE
);
ALTER TABLE telecom.asistencia_telecom
OWNER TO postgres;
```

```
CREATE TABLE telecom.personal_telecom
(
id serial NOT NULL,
cedula character varying(11) NOT NULL,
nombre1 character varying(50) NOT NULL,
nombre2 character varying(50),
apellido1 character varying(50) NOT NULL,
apellido2 character varying(50),
fechanac character varying(50),
nacionalidad character varying(50) NOT NULL,
telefono character varying(50) NOT NULL,
correo character varying(50) NOT NULL,
tipopersonal character varying(50) NOT NULL,
condicion character varying(50),
dedicacion character varying(50),
escalafon character varying(50),
fechareg character varying(50) NOT NULL,
CONSTRAINT personal_telecom_pkey PRIMARY KEY (cedula)
)
WITH (
OIDS=FALSE
);
ALTER TABLE telecom.personal_telecom
OWNER TO postgres;

CREATE TABLE telecom.notificaciones_telecom
(
id serial NOT NULL,
cedula character varying(11) NOT NULL,
fecha character varying(20) NOT NULL,
tipo character varying(2) NOT NULL,
notificacion character varying(450),
CONSTRAINT notificacines_telecom_pkey PRIMARY KEY (id)
)
WITH (
OIDS=FALSE
);
ALTER TABLE telecom.notificaciones_telecom
OWNER TO postgres;
```

Configuración de PostgreSQL para la Administración Remota

El siguiente procedimiento solo se realizará en la base de datos local, es decir; donde se será implementado el programa principal. Luego de haber culminado la instalación satisfactoriamente se procede a configurar los siguientes parámetros:

1) Para Windows permitir el acceso en el firewall o las medidas de seguridad en capas anteriores al servidor.

2) Abrir la carpeta data, ubicada dentro de la carpeta de PostgreSQL, C:Files.5.

Editar el fichero postgresql.conf, verificando que la siguiente línea de código este con * y no se encuentre comentada listen_addresses = '*'

Con esto se logra que el manejador pueda escuchar diferentes IP y por ende conectarse remotamente.

Editar el fichero pg_hba.conf para agregar el conjunto de IP que el servidor podrá leer, es importante resaltar que, la IP asignada a la PC donde estará el servidor tiene que ser estática, si es dinámica la configuración se pierde cada vez que se inicia la PC.

TYPE DATABASE USER ADDRESS METHOD

IPv4 local connections:

host all all xxx.xxx.x.2/24 md5

host all all xxx.xxx.x.5/24 md5

Si se quiere que toda la red se pueda conectar se debe establecer el rango completo de la red por ejemplo xxx.xxx.x.0/24, con esto especificamos que toda la red podrá conectarse al servidor.

Para los servidores remotos, solo es necesario instalar PgAdminIII y seguir el siguiente procedimiento:

1) Abrir el administrador de PgAdminIII, ya dentro de la aplicación se debe presionar el botón nueva conexión. Ver figura 11.11.

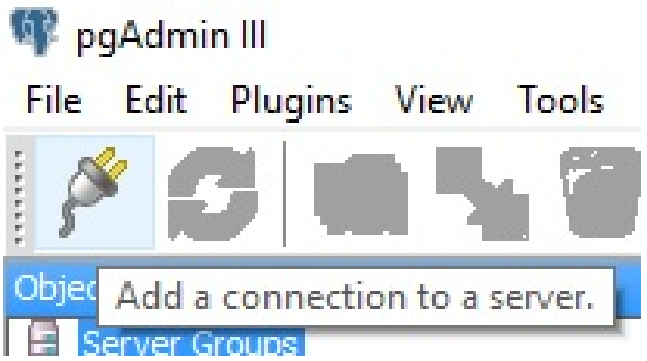


Figura 11.11: Administrador PgAdminIII

2) Configurar la nueva conexión remota como se observa en la figura 11.12.

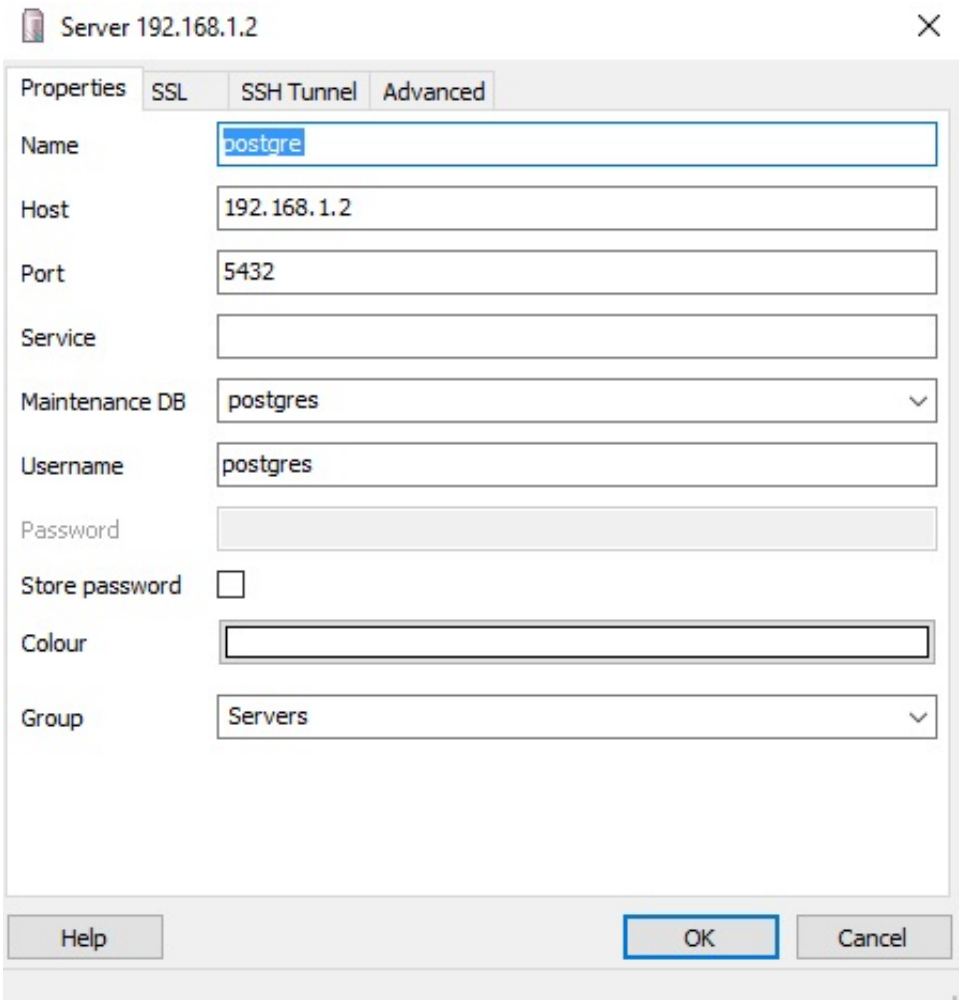


Figura 11.12: Configuración de la conexión remota

En la parte de Name se puede insertar el nombre que desee. En Host se debe insertar la IP donde está el servidor local, caso contrario no se establecerá la conexión. En Port se debe poner el mismo puerto. En Password se pondrá la contraseña usada en el servidor principal.

Realizado este procedimiento queda correctamente establecida la conexión remota.

Con esto estará culminado el manejo con el software de PostgreSQL, desde este punto, todo se manejará desde el software de reconocimiento facial.

Instalación del Software de Reconocimiento Facial

Con el propósito de hacer más cómodo la instalación del software, se hizo un ejecutable del programa en cuestión, con dicho trabajo, se evita las molestias de estar instalando Python en la PC, además de las librearías utilizadas para tal fin. Debido a que el archivo ejecutable convierte toda la información en un código binario que puede ser interpretado desde otras consolas, para este caso se hizo el ejecutable para el sistema operativo de Windows.

Para tal fin se utilizó la librería de Python llamada pyinstaller.

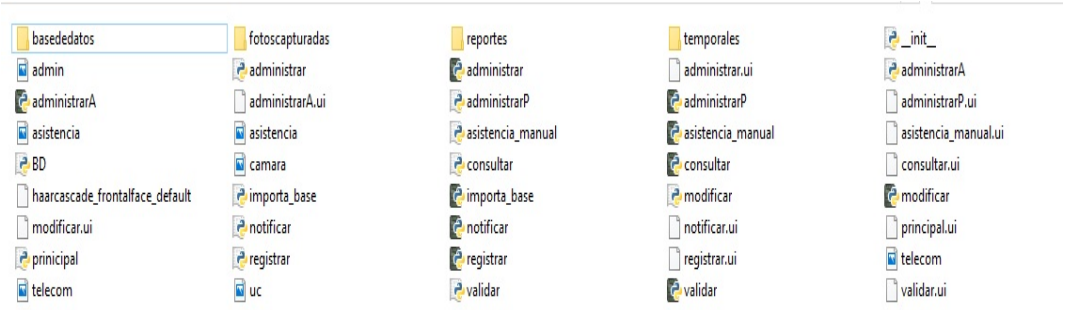


Figura 11.13: Archivos utilizados para crear el ejecutable

Luego que se tengan todos los archivos y módulos relacionados con el programa principal, se ejecuta de la cmd el siguiente comando:

Pyinstaller –onedir –onefile –noconsole principal.py (figura 11.14).

Figura 11.14: Ejecución por consola

Al terminal este proceso se tendrá el software ejecutable en otras máquinas con sistema operativo Windows, con la premisa de que no se tienen que instalar librerías adicionales para su funcionamiento.

Funcionamiento

Cuando se tengan los elementos que intervienen en el proceso de funcionamiento, ya se podrá utilizar la herramienta, es decir PostgreSQL debidamente configurada, software de reconocimiento facial y la cámara web. Para ver el funcionamiento del mismo debe referirse al manual de usuario (Apéndice I) y administración (Apéndice J) .

Apéndice L

Planilla de Incidencias

Registro de Asistencia Manual y Observaciones

CEDULA	FECHA DD/MM/AA	HORA_E HH:MM:SS	HORA_S HH:MM:SS	OBSERVACIÓN

Figura 12.1: Ejemplo de la planilla de incidencias

Cédula	Fecha DD/MM/AA	Hora$_E$ HH : MM : SS	Hora$_S$ HH : MM : SS	Observación

Apéndice M

Características de las Cámaras

Cámara 1: Cámara Web Havit Usb Hv-n632

En la figura 13.1 se observa la cámara adquirida para la implementación del software piloto de reconocimiento facial para el control de asistencias en la Escuela de Telecomunicaciones de la Universidad de Carabobo.

Figura 13.1: Cámara 1: Havit Usb Hv-n632

Características:

- Sensor de imagen: CMOS

- Píxeles: 3 Mega Max.

- Resolución: 640 × 480

- Velocidad de fotogramas: Max.30 fps en formato VGA

- Formato de salida: YUY2 sin comprimir USB2.0

- Conecte: USB 2.0 de alta velocidad (isócrono)

- Control del parpadeo: 50HZ/60HZ

- Formato de almacenamiento de captura de imágenes: BMP / JPG

- Formato de almacenamiento de imágenes: LAVI

- Tipo de interfaz: USB

- Pixels: 300K Hardware; Max.16MEGA Software

- Con Micrófono.

Cámara 2: Cámara Web Markvision M-350K-MV

Características:

- Resolución: 0.3 Megapixels

- Tamaño de la imagen: 648 x 480 Pixels

- Ajustes: Inclinación, Rotación

- Recursos: Microfone Integrado

Figura 13.2: Cámara 2: Markvision M-350K-MV

Apéndice N

Presupuesto

En la tabla 14.1 se observa un pequeño presupuesto de la inversión de los recursos que se necesitan para implementar el software piloto de reconocimiento facial para el control de asistencias en la Escuela de Telecomunicaciones de la Universidad de Carabobo, debido a que ella ya cuenta con computadoras solo se requirió la compra de la cámara web la cual puede ver en el apéndice M, en cambio si no se cuenta con ella se requiere una inversión total de 415000Bs.

Tabla 14.1: Presupuesto

Recursos	Precios (Bs)
Cámara Web	15.000
Computadora de Escritorio	400.000
Manejador de base de datos (PostgreSQL)	0
Software de reconocimiento facial	0
Total	415.000

NOTA: No se requirió inversión alguna para la realización e implementación del programa ya que todos los software utilizados poseen una licencia de código abierto tales como: Python, PostgreSQL, OpenCV y Qt.

Referencias Bibliográficas

[1] Jacqueline Richter. *El Decreto Ley Orgánica del Trabajo, de los Trabajadores y Trabajadoras*. Venezuela, 2013.

[2] Maritza Bracho. *Sistema de Reconocimiento de Rostros para Maggie*. Universidad Centroccidental Lisandro Alvarado, 2008.

[3] Gustavo Malpica y Nelson Mogollón. *Desarrollo de un software para la detección de cadenas nacionales mediante la identificación de patrones de imágenes*. Universidad de Carabobo, 2015.

[4] Germán Scrael. «Sistema de Reconocimiento Facial». Universidad Nacional del Litoral de Santa Fé, 2010.

[5] L Larcher, E Biasoni, C Cattaneo, A Ruggeri y AC Herrera. «Algoritmo para detección de bordes y ulterior determinación de objetos en imágenes digitales». En: *Mecánica Computacional* 30 (2011), págs. 2841-2852.

[6] Rubén Medina y Jesús Bellera. «Bases del Procesamiento de Imágenes Médicas». En: *Universidad de Los Andes, Facultad de Ingeniería, Grupo de Ingeniería Biomédica de la ULA. Venezuela* (1997).

[7] Rafael C Gonzalez y Richard E Woods. *Digital image processing*. 2.ª ed. Prentice hall Upper Saddle River, 2002.

[8] Steve. Rossius. «Reconocimiento de objetos mediante WebCam en tiempo real». Universidad Politécnica de Valencia, 2013.

[9] Jesús Ureña Cabello y Juan J Garcia. *Fusión de algoritmos para detectar objetos en movimiento*. Universidad de Oriente, Cuba. Universidad de Alcalá, España. 2002.

[10] Paul Viola y Michael J Jones. «Robust real-time face detection». En: *International journal of computer vision* 57.2 (2004), págs. 137-154.

[11] Jim R Parker. *Algorithms for image processing and computer vision*. John Wiley & Sons, 2010.

[12] Paul Viola y Michael Jones. «Rapid object detection using a boosted cascade of simple features». En: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2001, págs. I-511.

[13] Miguel Delgado Rodríguez. *Extracción automática de caras en imágenes captadas con móviles Android*. Universidad Politécnica de Catalunya, 2012.

[14] A Dragolici. *Detección, localización e identificación biométrica de caras en imágenes: métodos y evaluación en el marco NISTMBGG*. Universidad de Madrid, 2010.

[15] Marta Lucía Guevara, Julián David Echeverry Correa, William Ardila Urueña y col. «Detección de rostros en imágenes digitales usando clasificadores en cascada». En: *Scientia et Technica, Universidad Tecnológica de Pereira* 14.38 (2008), págs. 1-5.

[16] Edwar Jacinto Gómez. *Reconocimiento de Imágenes Faciales orientado a Controles de Acceso y Sistemas de Seguridad*. Universidad Distrital Francisco José de Caldas, Colombia. 2015.

[17] Yoav Freund y Robert E Schapire. «A desicion-theoretic generalization of on-line learning and an application to boosting». En: *European conference on computational learning theory*. SS971504. 1997, págs. 119-139.

[18] José Alfredo Poblete Castro. *Análisis de sentimiento y clasificación de texto mediante Adaboost Concurrente*. Universidad Católica de Valparaíso, 2016.

[19] Esdras E Sanabria Cuevas y MSc Daniel A Velazco Capacho. «Reconocimiento Facial Basado en Eigenfaces, LBHP Y Fisherfaces en la Beagleboard-xM». En: *Universidad de Pamplona, Colombia*. Vol. 10. 20. 2015, págs. 145-151.

[20] Javier Eslava Ríos. *Reconocimiento facial en tiempo real*. Universidad Autónoma de Madrid, 2013.

[21] Henry Paúl Espinosa Peralta. «Diseño e implementación de un sistema de seguridad y alerta para vehículos, basado en reconocimiento facial y localización GPS, en una Raspberry Pi B plus». Tesis doct. Quito, 2016.

[22] Patrascu Viorica Andreea. *Aplicacion para deteccion y reconocimiento facial en interiores*. Universidad de Sevilla, 2016.

[23] Javier Espinosa Montoro, Baldrich i Caselles y col. *Herramienta de gestión y búsqueda de fichas personales por reconocimiento facial*. Universidad Autónoma de Barcelona, 2015.

[24] Pablo Arturo Álvarez Corrales. *Prototipo de sistema piloto para control de acceso basado en reconocimiento de rostros*. Universidad Militar Nueva Granada, 2013.

Anexo A

Código XML. Detector de Caras Frontales AdaBoost, OpenCV. Véase en el CD adjunto, "haarcascade".

Anexo B

Algoritmo de Histogramas de Patrones Binarios Locales, OpenCV. Véase en el CD adjunto, "lbph".