UNIVERSIDAD DE CARABOBO

FACULTAD DE INGENIERIA

ESCUELA DE ELECTRICA

DPTO. SISTEMAS Y AUTOMATICA

PROYECTO DE GRADO

RECONOCIMIENTO DE ROSTROS MEDIANTE SISTEMA DE VISIÓN ARTIFICIAL Y TÉCNICAS DE SEGMENTACIÓN APLICADAS A TRAVÉS DE MATLAB

AUTORES;

LAMAS JONYRIS

MORENO ALEXANDER

Valencia, Junio de 2011

UNIVERSIDAD DE CARABOBO

FACULTAD DE INGENIERIA

ESCUELA DE ELECTRICA

DPTO. SISTEMAS Y AUTOMATICA

PROYECTO DE GRADO

RECONOCIMIENTO DE ROSTROS MEDIANTE SISTEMA DE VISIÓN ARTIFICIAL Y TÉCNICAS DE SEGMENTACIÓN APLICADAS A TRAVÉS DE MATLAB

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

AUTORES;

LAMAS JONYRIS

MORENO ALEXANDER

TUTOR

PROF. WILMER SANZ

Valencia, Junio de 2011

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERIA
ESCUELA DE ELÉCTRICA
DPTO. SISTEMAS Y AUTOMATICA

CERTIFICADO DE APROBACIÓN

Los abajo firmantes miembros del jurado asignado para evaluar el Trabajo Especial de Grado titulado **"RECONOCIMIENTO DE ROSTROS MEDIANTE SISTEMA DE VISIÓN ARTIFICIAL Y TÉCNICAS DE SEGMENTACIÓN APLICADAS A TRÁVES MATLAB",** realizado por los bachilleres **Moreno Aguiar, Alexander Javier,** C.I.: **18.060.172** y **Lamas Villegas, Jonyris Mercedes,** C.I.:16.801.642 **,** hacemos constar que hemos revisado y aprobado dicho trabajo.

_____

Ing. Wilmer Sanz
TUTOR

_____ _____

Ing. Demetrio Rey lago
JURADO

Ing. Liliana Villavicencio
JURADO

DEDICATORIA

Dedico este trabajo de investigación a mis padres **Elodia Aguiar y Nolberto Moreno,** que en cada amanecer cuando abro mis ojos y tengo el privilegio de volver a mirar, veo crecer mi deuda con estos dos tesoros que bajo sus servicios con Dios han dedicado sus vidas entera para poder crear luz en este mundo que está lleno de tantas oscuridad, con su amor eterno que será siempre alimento en mi vida que estas dos personas crearon, por tantas cosas gracias una vez más.

A mis hermanos por ser más que un apoyo y darme las fuerzas para lograr mis sueños y metas alcanzadas por mí.

A mi maravillosa familia, que por el gran cariño y afecto que les tengo a cada uno, se me hace complicado nómbralos a todos o mencionar a alguien en particular aunque no mencione sus nombres forman parte de mi vida, en mis éxitos obtenidos que sin lugar a dudas son especiales por su sinceridad y buenos deseos de siempre.

…. A todos mis más sinceros agradecimientos de todo corazón.

Dios los Bendiga.

DEDICATORIA

Todo lo puedo en Cristo que me Fortalece. *Filipenses 4:13.*

Primeramente deseo dedicarle este proyecto y toda mi carrera a mi Señor y salvador, todo sea para su honra, pues Él es digno de alabanza; reconociendo que sólo por Él estoy aquí y es Su amor, Su consuelo, Su brazo fuerte y Fiel lo que me sostiene en pie. Jamás tendré con que pagarte oh mi Señor.

A mi mamá mujer luchadora, virtuosa, amiga incondicional, le dedico éste logro por brindarme su ayuda, apoyo, comprensión, consejos, cuidados y oraciones; Por ser tan única conmigo, sin ella no sería lo que soy y su voz de aliento acompaño cada día de mi carrera. A mi papá por su ayuda, apoyo, consejos, cuidados y su esfuerzo. Le dedico este trabajo por estar conmigo en todo momento.

A mi hermanito por su ayuda, compresión por estar conmigo. Esperando ser una inpiración para que él también logre sus metas; A mis abuelos, por su ayuda y por ser tan especial conmigo. En especial a mi abuela porque sé que muchas de sus oraciones me ayudaron (el vivir es Cristo, más el morir es ganancia).

A mis familiares que me apoyaron y ayudaron. Quisiera aprovechar estas líneas para agradecerle a mi tío José Luis Villegas Salazar por su ayuda especial y mis hermanos en Cristo Manuel Lopez, Pastor Moncada, William Hernandez, Romer Sequera, Alexis Tallaferro, Flia Rojas y Francys Pinto por sus oraciones.

Nunca tendré con que pagar su ayuda, ruego a Dios que los guarde y les colme de sus ricas bendiciones. Amén.

Por tanto, os digo que todo pidiereis orando lo recibiréis… *S. Marcos 11:24.*

Jonyris Lamas

AGRADECIMIENTOS

Más gracias sean dadas a Dios, que nos da la victoria por medio de nuestro Señor Jesucristo.

1^{era} corintios 15:57.

Primeramente le damos las gracias a Dios, por esta victoria y por sus misericordias que son nuevas cada mañana; El nos quiso bendecir con este logro académico y con el don inmerecido de la salvación por medio de la bendita y preciosa persona del Señor Jesucristo, quien se ofreció por nosotros una sola vez y para siempre en propiciación por nuestros pecados.

A nuestros padres, por su apoyo, esfuerzo y ayuda incondicional. Especialmente a nuestras madres por sus oraciones constantes a nuestro favor.

A nuestros hermanos, familiares, amigos y compañeros de clase que contribuyeron con nosotros a lo largo de nuestra carrera universitaria y durante este proyecto.

A nuestros hermanos por la gracia de Cristo, por sus oraciones a nuestro favor y apoyo. A nuestro tutor Wilmer Sanz, por su valiosa colaboración, disposición, apoyo y conocimiento.

A todos mil gracias; esperando que Dios les guarde y bendiga grandemente.

Los Autores

ÍNDICE GENERAL

Página

ÍNDICE DE FIGURAS

Página

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la presente investigación se aborda el problema de reconocimiento de rostros, tema estudiado hoy en día por diversas empresas e instituciones que se dedican a la investigación de reconocimientos de formas. Siguiendo esa corriente de investigación, el objetivo general de este trabajo es diseñar e implementar un sistema computacional capaz de reconocer rostros a partir de imágenes faciales capturadas a través de una cámara web. El sistema compara la imagen adquirida con aquellas almacenadas en una base de datos. Para ello, se desarrollará una aplicación en MATLAB, utilizando la interfaz visual Guide. Para el reconocimiento es necesario aplicar técnicas de segmentación de imágenes para extraer las características de interés.

Este trabajo de investigación, está conformado básicamente por cinco capítulos, en los cuales se trata de llevar una secuencia en el desarrollo del proceso de investigación, de manera tal, que permita alcanzar todos los objetivos de la investigación

En este sentido, en el Capítulo I se presenta el planteamiento del problema de investigación, así como también el objetivo general y los objetivos específicos para alcanzar la solución de dicho problema, la justificación que le enfoca la importancia de este proyecto y alcances del trabajo realizado.

En el Capítulo II incluye los fundamentos teóricos utilizados para la investigación, los cuales permiten comprender el tema en estudio.

En el Capítulo III se señalan los pasos, métodos y procedimientos aplicados para la realización del estudio.

En el Capítulo IV se explican los criterios establecidos y los algoritmos utilizados para llevar a cabo el software, además se describen con detalle el contenido de las ventanas del

sistema (la interfaz gráfica de usuario); también se justifica la creación y componentes del prototipo.

Y finalmente en el Capítulo V, se presenta el análisis de resultados, las conclusiones y recomendaciones con respecto al trabajo especial de grado.

Además de los capítulos mencionados anteriormente posee dos suplementos que contienen la codificación del software y el manual de usuario respectivamente.

RECONOCIMIENTO DE ROSTROS MEDIANTE SISTEMA DE VISIÓN ARTIFICIAL Y TÉCNICAS DE SEGMENTACIÓN APLICADAS A TRAVÉS DE MATLAB

AUTORES: Lamas Jonyris, Moreno Alexander

TUTOR: Ing. Wilmer Sanz

Junio 2011

RESUMEN

En este proyecto especial de grado, se presenta un sistema de reconocimiento de rostros partiendo de la captura de la imagen del mismo, para aplicarle técnicas clásicas de segmentación. El algoritmo usado se basa en la extracción de dos puntos característicos del rostro y sus respectivas distancias a un tercer punto fijado externamente; luego estos puntos son comparados con las características almacenadas en una base de datos, para su posterior reconocimiento. El producto es un software capaz de identificar personas, haciendo uso de la herramienta GUIDE de Matlab y un prototipo para la toma de imagen mediante cámara web, donde se controlan parámetros tales como iluminación, distancia y posición del usuario.

Es importante destacar, que la interfaz gráfica es sencilla, amigable e interactiva. El administrador del sistema (usuario con contraseña) tiene acceso a la base de datos y es capaz de modificarla.

De acuerdo con los resultados obtenidos se puede destacar la eficiencia del sistema y su rápida respuesta.

CAPITULO I. EL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

El análisis del rostro a partir de estudios imagenológicos para obtener datos e información, ha sido clave en los últimos años en el área de investigación activa que abarca diversas disciplinas, como procesado de imágenes, reconocimiento de patrones, visión por computador entre otros. Actualmente la disciplina que ha crecido y ha cobrado gran interés es la visión por computador. Esta consiste en la capacidad de la máquina para “ver” el mundo que lo rodea, más precisamente para deducir las estructuras y las propiedades del mundo tridimensional a partir de una o más imágenes bidimensionales.

La complejidad de las imágenes y los procesos de visualización van a depender de las características de interés a extraer en la imagen estudiada, ya que cada una tiene un conjunto de características particulares. Por ende, es necesario hacer un procesamiento previo de la imagen tomada [1].

El procesamiento de imágenes es un conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad, facilitar la búsqueda de información, descartar algunas características, entre otros. Una de las técnicas para procesar una imagen es a través de la segmentación cuya finalidad es extraer el área o estructura de interés de la imagen, logrando el computador reconocer y dar el tratamiento adecuado para su aplicación.

Unas de las aplicaciones que ha venido cobrando gran utilidad en el campo de la visión por computador ha sido el reconocimiento de rostros. El reconocimiento de rostros consiste en la identificación de personas a partir de una imagen adquirida de su cara, unos de los métodos se basa en la similitud estadística de la geometría humana. Sin embargo; son muchos los problemas específicos que surgen en el procesamiento de imágenes con rostros, aunque existe una relación

muy estrecha entre todos ellos, pero la mayoría se pueden clasificar según la dificultad que se trate de resolver. Podemos identificar los siguientes grandes problemas [2]:

- **Condiciones de iluminación:** Podemos clasificar dentro de este grupo otros muchos problemas con rostros que han sido abordado por diferentes autores, y que tratan de responder preguntas sobre condiciones concretas de iluminación. Ya que el control de la misma nos proporciona una disminución significativa de los efectos indeseados como imágenes con un bajo contraste, brillos y sombras.

- **Estimación de pose:** En este caso se trata de resolver la posición y orientación de la cabeza del usuario en relación a la cámara (a esto llamamos la pose). Un problema relacionado es la estimación del punto de mira (en otras palabras, que el individuo tenga la mirada fija en un punto de referencia) que se centra en el análisis de los ojos.

- **Localización de puntos de interés:** El propósito de la localización es determinar las posiciones exactas que ocupan los ojos y otros elementos que puedan resultar de interés, aquí el objetivo esencial es la precisión de los resultados.

Recientemente se han hecho diferentes progresos debido a los avances en el modelado de rostros y técnicas de análisis. Los sistemas para reconocimiento han sido desarrollados para la detección y seguimiento; sin embargo el reconocimiento confiable de rostros sigue ofreciendo un gran reto para los investigadores debido al gran número de usos que está posee, asociado a esto los diversos problemas ya descritos. Entre las principales aplicaciones tenemos:

- **Seguridad:** El interés del reconocimiento de rostros en el ámbito de seguridad afecta tanto al sector público como al privado. Entre algunos usos específicos podemos encontrar:

  - control de aduanas, inmigración, pasaportes;

- documentos nacionales de identidad, seguridad social, permiso de conducir, etc., con información de rostros;
- sistemas de clave a través de rostros, domestico o empresarial;
- seguridad en aplicaciones como acceso a Internet, a bases de datos, registros médicos, etc.;
- entrada y control de acceso a edificios.

- **Video-vigilancia:** Existen otras aplicaciones relacionadas con la seguridad, que situamos en este grupo. En relación con la categoría anterior, la particularidad es que los usuarios pueden no ser conscientes de que están siendo controlados. Entre los ejemplos de este tipo tenemos:

  - sistemas de vigilancia en aeropuertos o en edificios públicos;
  - búsqueda de personas desaparecidas;
  - seguimiento automático de sospechosos de robo en centros comerciales;

- **Entrenamiento Computacional:** El procesamiento automático de los rostros permite añadir nuevos mecanismos de interacción hombre/máquina que, convenientemente diseñados, pueden resultar más intuitivos, sencillos y potentes que los sistemas tradicionales de teclado y ratón. Por ejemplo, tenemos:

  - interfaces para la navegación en entornos virtuales;
  - interacción natural con robots;
  - videojuegos basados en la percepción del rostro;
  - sistemas de aprendizaje a distancia.

Finalmente, en esta investigación se abordará el problema de reconocimiento de rostros a partir de las técnicas clásicas de segmentación de imágenes, la cual consiste en la extracción de puntos característicos de una imagen para su posterior descripción o reconocimiento, lo que

conduce a la creación de un software en Matlab que sea capaz de identificar a una persona. Todo esto involucrando la construcción de un prototipo para la toma de imagen y así poder limitar algunas variables como iluminación, distancia y posición del usuario.

1.2 JUSTIFICACIÓN DE LA INVESTIGACIÓN

Se realizará este estudio con la finalidad de ofrecer una herramienta computacional para el reconocimiento de rostros, todo esto debido al interés que ha ido creciendo a medida que van surgiendo nuevos avances tecnológicos como las cámaras Web, cámaras digitales, dispositivos móviles y aumentos de la demanda en el ámbito de seguridad: acceso de zonas restringidas, la identificación en entradas a aeropuertos y entes públicos (Universidades), etc.

Una de las principales características del reconocimiento de rostros es que se utilizan aspectos faciales correspondientes a cada persona, siendo este el método más usado de manera natural por el ser humano para reconocer a sus semejantes [3].

Aunque los progresos en el reconocimiento de rostros han sido animadores, el mismo ha resultado ser una tarea difícil, debido a que las imágenes varían considerablemente según el punto de vista, iluminación, expresión, posición, entre otros. Todo esto nos lleva a introducir ciertas premisas que mantendrán al margen estas variantes.

Este proyectó final de grado está enmarcado dentro del área de aporte a la investigación, desarrollada en el Departamento de Sistemas y Automática de la Universidad de Carabobo. Concretamente en la línea de Visión Artificial, que tiene además aplicaciones en otra línea de investigación, de los cuales la Robótica Industrial.

1.3 OBJETIVOS DE LA INVESTIGACIÓN

Los objetivos generales y específicos se describirán a continuación:

1.3.1OBJETIVO GENERAL

Desarrollar un sistema de reconocimiento de rostros mediante visión artificial y técnicas de segmentación aplicadas a través de Matlab.

1.3.2 OBJETIVOS ESPECÍFICOS

- ✓ Analizar los algoritmos de programación orientados a las técnicas de segmentación para calificar su posterior uso.
- ✓ Desarrollar un algoritmo basado en las técnicas de segmentación para la extracción de características faciales y su posterior reconocimiento.
- ✓ Seleccionar materiales y equipos que serán usados en el módulo físico de reconocimiento para fijar el rostro del usuario.
- ✓ Construir el escenario que será implementado para la captura de la imagen.
- ✓ Desarrollar la interfaz de usuario en Matlab para visualizar el reconocimiento de rostros.

1.4 ALCANCE DE LA INVESTIGACIÓN

El avance de reconocimiento de rostros ha sido motivante, incluso ha pasado a ser una labor difícil, especialmente debido a las variantes en las imágenes, tales como posición, iluminación y algunas otras variantes de consideración. La presente investigación se limitará a

estudiar el reconocimiento mediante la visión artificial a través de una cámara web y las técnicas de segmentación aplicadas a través de Matlab.

Además, se introducirán algunas premisas para no tener las variantes en cuanto a posición e iluminación; lo que concibe dentro del proyecto la construcción de elementos como: soporte de rostro, fijación de la distancia de captura de la imagen, iluminación controlada, adaptación y creación de algoritmos para procesamiento de imagen y elaboración de interfaz grafica de usuario (GUI).

CAPITULO II. MARCO TEÓRICO

2.1 ANTECEDENTES DEL PROBLEMA

Palacios S. [2004] SISTEMA DE RECONOCIMIENTO DE ROSTROS

El presente proyecto propone un sistema computacional capaz de reconocer rostros a partir de imágenes faciales capturadas a través de una cámara web. El sistema compara paramétricamente la imagen adquirida con aquellas almacenadas en una base de datos (usuarios registrados). Para ello, se desarrolló una aplicación en MATLAB, utilizando la interfaz visual Guide. En esta aplicación se pueden controlar los parámetros utilizados para el proceso de reconocimiento. El método de reconocimiento empleado es basado en el procesamiento de imágenes "eigenfaces". Este proyecto será usado como base teoría para armar el prototipo de captura de imágenes. [4]

Valvert J. [2004] MÉTODOS Y TÉCNICAS DE RECONOCIMIENTO DE ROSTROS EN IMÁGENES DIGITALES BIDIMENSIONALES

Este trabajo se basa en dar a conocer y exponer las técnicas más utilizadas para el procesamiento de una imagen digital que contiene un rostro humano, así como las técnicas de identificación de rasgos básicos de un rostro humano en una imagen digital.

La fase de detección, es la parte más crítica del proceso de reconocimiento de rostros, ya que, en ella, se debe almacenar información sobre el rostro, que después deberá ser usada para el reconocimiento. En esta etapa se procesan los rostros, haciéndolos invariantes al tamaño y a los cambios de contraste, utilizando los algoritmos de los momentos invariantes de Huy de Yan. Para el aprendizaje y reconocimiento de rostros, se utilizan clasificadores no lineales de patrones, los cuales pueden ser calculados por métodos de simulación o algún método no matemático, en este caso, las redes neuronales y los algoritmos genéticos. Este proyecto fue de ayuda para comprender las técnicas del reconocimiento de rostro en una imagen digital. [5]

De Ponte C. y Ojeda J. [2004] SISTEMA DE CONTROL DE ACCESO AUTOMÁTICO BASADO EN TÉCNICAS DE VISION ARTIFICIAL PARA EL RECONOCIMIENTO DE ROSTRO.

Presentado como trabajo especial de grado, para optar al título de licenciado en computación, Universidad de Carabobo; En su trabajo los autores realizaron la construcción de un sistema, que por medio de reconocimiento de rostros, permitiera o impidiera el acceso de personal a un área restringida, haciendo uso de dos técnicas "Modelo Escondido de Markov" y "Redes Neuronales". Este trabajo permitirá conocer las metodologías de reconocimiento de patrones faciales. [6]

López C. [2005] RECONOCIMINETO FACIAL MEDIANTE VISIÓN ARTIFICIAL EN ENTORNO DE PROGRAMACION OPENCV

En esta investigación se diseño un software para la detección de rostros mediante el uso de la descomposición en valores principales (PCA) y Eigenfaces**.** La elección de las técnicas, para la detección de rostros basada en el uso de eigenfaces ha venido determinada en gran medida por el uso de las librerías OpenCV (Intel® Open Source Computer Vision Library) de Intel®. En esta investigación se expone cuan determinante es la iluminación para el diseño de detección, los cuales fueron base para el desarrollo de este proyecto especial de grado. [7]

Rojas T. [2007] DISEÑO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA UN MANIPULADOR INDUSTRIAL ANTROPOMÓRFICO

En esta investigación se presenta el diseño de un sistema de visión artificial para posicionar un manipulador industrial antropomórfico disponible en el Laboratorio de Robótica y Visión Industrial de la Escuela de Ingeniería Eléctrica de la Universidad de Carabobo, donde la fuente principal parte de la información que puede proveer un sistema de visión artificial el cual puede ser particularmente aplicada en robótica para la ejecución de tareas autónomas por parte de los

manipuladores mecánicos, cuando éstos son adecuadamente programados. Además. Todo esto es desarrollado con base a la herramienta matemática de la Transformada Circular de Hough para la detección de objetos esféricos, sistema de visión asociado al robot dado con la finalidad de enriquecer el potencial didáctico y académico de dicho laboratorio.[8]

Sáez A. [2009] RECONOCIMIENTO DE IMÁGENES A TRAVÉS DE SU CONTENIDO

En este proyecto se estudia el desarrollo de algoritmos de tratamiento digital de imágenes que permitieran clasificar fotografías en base al contenido gráfico. La idea para el desarrollo, se basa en la creación de un conjunto de imágenes para posteriormente segmentar la imagen, y extraer de ella objetos y fondos conocidos, con el fin de clasificarlas de acuerdo a los elementos localizados. En dicho proyecto, se desarrollaron tres algoritmos de alto nivel para el reconocimiento de los siguientes elementos: imágenes con cielo, imágenes con césped, e imágenes con caras. El algoritmo localizador de caras, se basa en la identificación previa de los ojos, donde se aplican restricciones de tipo geométrico. Luego casarón a identificar una boca, en la parte inferior de los ojos centrado entre ambos. En caso de localizar afirmativamente los ojos y la boca, en la geometría apropiada, la respuesta del algoritmo será afirmativa. Este trabajo se utilizó de apoyo conceptual en la investigación debido a la metodología y procedimientos usados por el autor. [9]

Gámez C. [2009] DISEÑO Y DESARROLLO DE UN SISTEMA DE RECONOCIMIENTO DE CARAS MEDIANTE LA DESCOMPOSICIÓN DE LOS VALORES PRINCIPALES (PCA)

En este proyecto se presenta un sistema biométrico de reconocimiento que utiliza como característica biométrica una imagen digital estática del rostro humano. La fase de detección está basada principalmente, en la detección de píxeles de piel en la imagen. Posteriormente se realiza una selección de regiones de piel candidatas a ser caras y su validación mediante “mapas” de ojos y boca. Además se implementa un sistema alternativo de detección de la cara, por si con el

anterior método no se detecta ninguna. Éste método toma la mayor región de piel encontrada en la imagen y genera una elipse con sus características para devolver como cara la parte de la imagen que coincide con la elipse.

En la fase de reconocimiento se tienen las zonas de las imágenes detectadas como caras. Se utiliza PCA para extraer las características que representan a las imágenes. A continuación, estas características sirven para entrenar y simular unas redes neuronales. Con las salidas de las redes neuronales se seleccionan las imágenes de la base de datos que más se parecen a la cara de la imagen de prueba. Este trabajo será usado para el desarrollo del marco conceptual. [10]

Cortés M. [2009] RECONOCIMIENTO DE CARAS FRONTALES MEDIANTE LA EXTRACCIÓN DE PUNTOS CARACTERÍSTICOS

En esta investigación se realizó un sistema de reconocimiento facial en posición frontal (o cuasi-frontal) basado en las aproximaciones más "prometedoras" de las encontradas durante la fase de desarrollo teórico. La solución seleccionada se basa en la detección de un subconjunto de los puntos característicos de la cara (keypoints, KP) y definir una serie de características relativas a los mismos que posteriormente permitieran evaluar las capacidades discriminatorias entre las diversas personas a reconocer. En este trabajo se utilizaron diversos métodos de segmentación que forman parte del pre-procesamiento de la imagen, estos servirán de apoyo teórico para esta investigación. [11]

Cruz L. y Carranza F. [2009] RECONOCIMIENTO DEL IRIS

En este artículo se presenta el reconocimiento del iris, como uno de los modelos más efectivos para la identificación de una persona. Este método es estudiado por muchos investigadores. Uno de ellos es Daugman y propone una técnica que se basa en capturar la imagen del iris, y proceder a normalizarla a través de una conversión a coordenadas polares, y la aplicación contigua de ecuaciones diferenciales que permiten la obtención de un patrón

denominado código del iris. Para la comparación de dos patrones, se utiliza la distancia de Hamming. Donde los resultados que se obtuvieron son muy eficientes y con mucha precisión, y cada vez se amplía su uso. Esté trabajo fue usado para conocer las metodologías de reconocimiento de rostro. [12]

Hernández J. [2009] RECONOCIMIENTO DE ROSTROS UTILIZANDO HISTOGRAMAS SECUENCIALES DE IMAGEN

Este trabajo trata el problema de reconocimiento de rostros, tema muy abordado hoy en día por diversas instituciones que se dedican a la investigación de reconocimiento de patrones y formas. El reconocimiento de rostros es un problema en el que se tienen que analizar diversos factores que influyen en la capacidad de reconocimiento del sistema, tales como pose, iluminación, gesticulación, entre otros, existen métodos ya desarrollados, con una eficiencia alta; sin embargo el objetivo de este trabajo es diseñar e implementar un método que sea capaz de reconocer rostros mediante el uso de histogramas secuenciales de imagen. El histograma de una imagen es una estadística que muestra la frecuencia acumulada de pixeles de algún color en particular, o bien, de una combinación de diversos colores, en el caso particular del rostro el histograma presenta una distribución multimodal, debido a que las características de cada rostro son muy variantes.

Es necesario aplicar técnicas de segmentación de imágenes para poder aislar el rostro de la fotografía capturada y convertirlo a su respectivo en escala de grises, una vez realizado lo anterior, se procede a obtener el histograma de la imagen, el cual es comparado mediante un método estadístico conocido como distancias de Hamming, para poder hacer uso de histogramas secuenciales de imagen, es necesario comparar una imagen con una secuencia de diversas imágenes previamente almacenadas. El resultado es un método cuya eficiencia es bastante aceptable comparada con otros métodos ya existentes, además de que gracias a que el método diseñado e implementado tiene un bajo consumo de recursos, se ejecuta muy rápidamente, así mismo es muy económico y versátil, debido a que se puede implementar para diferentes sistemas

operativos. Este trabajo fue base para la compresión de las técnicas de segmentación aplicadas a una imagen desarrolladas a largo de esta investigación. [13]

Dragolici A. [2010] DETECCION, LOCALIZACIÓN E IDENTIFICACIÓN BIOMETRICA DE CARAS EN IMÁGENES: MÉTODOS Y EVALUACIÓN EN EL MARCO NIST- MBGG

A lo largo de este proyecto se estudia, implementa y evalúa un detector facial y un detector de ojos en imágenes. Como punto de partida se realiza un estudio y evaluación de los sistemas biométricos y un estudio para las técnicas de implementación de los sistemas de detección facial existentes hasta el momento, desarrollando finalmente el mecanismo seleccionado. El objetivo principal del proyecto es la implementación de un algoritmo capaz de detectar objetos (caras) sobre imágenes además de poder ser evaluado en diferentes condiciones y entornos de trabajo.

Para llevar a cabo dicha evaluación se han utilizado las imágenes de rostros de posición frontal de las bases de datos CMU (Universidad Carnegie Mellon) y NIST- MBGG (National Institute of Standards and Technology Multi Biometrics Grand Challenge) comparando el sistema implementado con sistemas de libre distribución que ya han mostrado un rendimiento competitivo y que son una referencia en el ámbito de la detección facial. Este trabajo fue usado como soporte conceptual, para el desarrollo de las bases teóricas. [14]

2.2 BASES TEÓRICAS

2.2.1 VISIÓN ARTIFICIAL

La visión es uno de los mecanismos sensoriales de percepción más importantes en el ser humano. En el presente proyecto se pretende desarrollar un sistema de reconocimiento de rostros mediante el tratamiento de imágenes digitales. El interés de los métodos de procesamiento de imágenes digitales se fundamenta en dos áreas principales de aplicación: a) mejora de la calidad

para la interpretación humana; b) procesamiento de los datos de la escena para la percepción de las máquinas de forma autónoma. En el intento por dotar a las máquinas de un sistema de visión aparece lo que se conoce como visión artificial.

Desde una perspectiva general, la visión artificial por computador es la capacidad de la máquina para ver el mundo que lo rodea, más precisamente para deducir las estructuras y las propiedades del mundo tridimensional a partir de una o más imágenes bidimensionales [1].

Por supuesto, no es del alcance de este trabajo de investigación, explicar la infinidad de dispositivos de captura de imágenes existentes, desde las simples cámaras analógicas de video hasta sofisticados sistemas de visión en espectros de longitudes de onda diferentes a la luz visible. El lector será capaz de imaginar lo amplio que es este campo tecnológico. En lo concerniente a esta investigación, se recurrirá a una modesta cámara tipo Web (o Webcam) para la captura de imágenes [7].

En general, existen aspectos comunes que afectan la captura de imágenes para el posterior reconocimiento, independientemente del dispositivo utilizado [3]:

- Fuente de iluminación.
- Posición del rostro.

Para el desarrollo de la aplicación en este proyecto de investigación, se establecieron las siguientes premisas relativas a los aspectos anteriormente listados [7]:

2.2.2 FUENTES DE ILUMINACIÓN

En todo proceso de visión artificial es fundamental la elección del más adecuado tipo de fuente de iluminación, ya que la obtención de resultados satisfactorios dependerá en gran medida

de ello. Puede disminuirse de manera significativa los efectos indeseados como imágenes con un bajo contraste, brillos, y sombras. Un sistema de luces apropiado simplifica la imagen del objeto a estudiar aportando información de mayor calidad para la detección y extracción del mismo, haciendo innecesaria la aplicación de algoritmos de corrección en el proceso de visión.

El trabajar en entornos poco estructurados dificulta la aplicación de los procesos y técnicas de visión, por lo que se intenta que las condiciones de iluminación permanezcan invariantes el mayor tiempo posible [3].

En un entorno de trabajo pueden diferenciarse dos tipos de fuentes de iluminación [3]:

- Fuente de iluminación directa
- Fuente de iluminación arbitraria

2.2.2.1 FUENTES DE ILUMINACIÓN DIRECTA

Si se tiene una fuente de iluminación directa, la luz incide directamente sobre la cara del usuario intentándose así reducir al máximo los efectos de sombra puesto que se mejora sensiblemente el contraste de la imagen. Sin embargo aparecen otros problemas como brillos y zonas oscuras indeseadas si el foco de luz no se encuentra en posición y orientación cercana a la cámara de captura o si el plano de la cara del usuario no está completamente perpendicular al haz de luz del foco emisor. Por tanto, puede falsearse el proceso de captura de imágenes. Además hay que tener en cuenta que un foco de luz apuntando directamente sobre la cara del usuario resulta molesto, disminuyendo el grado de aceptación del sistema de reconocimiento de rostro [3].

2.2.2.2 FUENTES DE ILUMINACIÓN ARBITRARIA

Generalmente, la iluminación arbitraria del entorno de trabajo no suele ser recomendable en procesos de visión por computador ya que las imágenes obtenidas presentan un bajo contraste

y un nivel de sombras inadecuado. Las imágenes obtenidas, más si se trabaja en exteriores, pueden variar enormemente de una a otra hora del día, dificultando tanto el proceso de captación como el de reconocimiento ya que influyen otras fuentes de iluminación no naturales [3].

2.2.2.3 FUENTE DE ILUMINACIÓN DEL ENTORNO DE TRABAJO

Vistas las particularidades de los diferentes tipos de iluminación y teniendo en cuenta que una de las características iníciales del proyecto es que se asumía que las condiciones de iluminación iban a ser constantes, para la iluminación del entorno de trabajo se ha elegido una mezcla de ambos tipos de iluminación, es decir, se cuenta con diferentes focos de luz directa, pero no se prescinde de la iluminación indirecta y arbitraria, ya sea natural o artificial, procurándose sin embargo que la zona donde el usuario se establezca para que se le capture una imagen de una cara tenga una variabilidad de iluminación prácticamente nula al igual que el fondo [3].

2.2.3 POSICION DEL ROSTRO

Para determinar la posición del rostro se tiene que tener presente la ubicación del dispositivo de formación de imágenes (cámara web). En este trabajo la cámara web estará colocada sobre un soporte, el cual a su vez tendrá una base donde la persona colocará el rostro para que la cámara web pueda capturar la imagen respectiva. El diseño consiste en un recinto, donde se encontrará la cámara web en un extremo y en el otro el soporte del rostro, se dispondrá la cámara del rostro a unos 15 cm de separación teniendo presente que todos los puntos de interés deberán encontrarse enfrente de la lente. El esquema de la base propuesta se muestra en la Fig. 2.1.

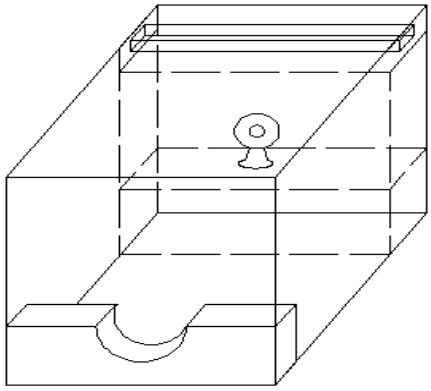


Figura 2.1 Base para la captura de imágenes [4]

El proceso general de la visión artificial se puede sintetizar en la figura 2.2 donde las cajas representan datos y las burbujas procesos. Se parte de la escena tridimensional y se termina con la aplicación de interés [1].

En la visión por computador, la escena tridimensional es vista por una, dos o más cámaras para producir imágenes a color. Las imágenes adquiridas pueden ser segmentadas para obtener características de interés tales como bordes o regiones. Posteriormente, de las características se obtienen propiedades subyacentes mediante el correspondiente proceso de descripción. Tras lo cual se consigue la estructura de la escena tridimensional requerida para la aplicación.

Las imágenes de intensidad están íntimamente ligadas al concepto de luminosidad. Esto es la relación de la cantidad de luz radiada sobre los puntos de las superficies de la escena y la cantidad de luz incidente en los correspondientes puntos de la imagen [1].

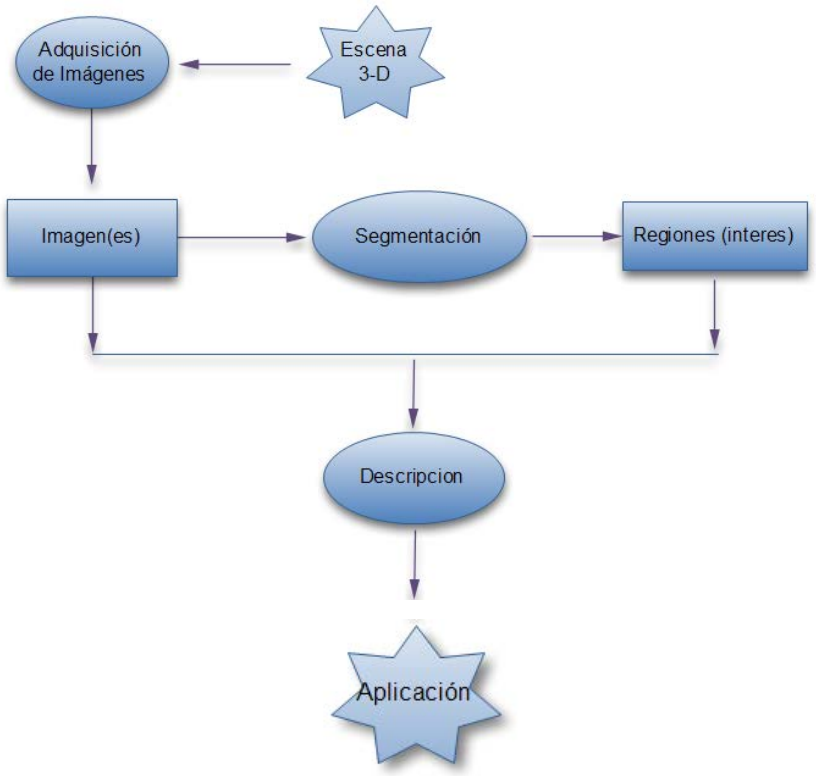


Figura 2.2 Diagrama de bloques mostrando el proceso completo de la Visión Artificial. [1]

En definitiva, la imagen que ha de ser tratada por el computador se presenta digitalizada espacialmente en forma de matriz con una resolución de MxN elementos. Cada elemento de la matriz denominado pixel (picture element) tendrá un valor asignado, que se corresponde con el nivel de luminosidad del punto correspondiente en el rostro capturado, dicho valor es el resultado de la cuantización de la intensidad o nivel de gris [1].

2.2.4 PIXEL

Es la abreviatura de las palabras inglesas “picture element”. Es el menor de los elementos de una imagen al que se puede aplicar individualmente un color o una intensidad o que se puede diferenciar de los otros mediante un determinado procedimiento. Ver figura 2.3 [9].

Figura 2.3 Representación de un píxel [10]

2.2.4.1 RELACIÓN ENTRE PIXEL

2.2.4.1.1 VECINDAD

Un píxel p de coordenadas (x, y) presenta un total de cuatro vecinos en el plano vertical y horizontal, siendo sus coordenadas las representadas en la figura 2.4.

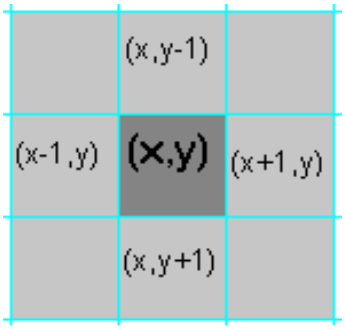


Figura 2.4. Píxel de interés y su conectividad tipo 4

Este conjunto de píxeles se denomina vecindad de tipo 4 del píxel p, se denota $N_4(p)$. Además se puede considerar la existencia de otros cuatro vecinos asociados a las diagonales, cuyas coordenadas se visualizan en la figura 2.5.

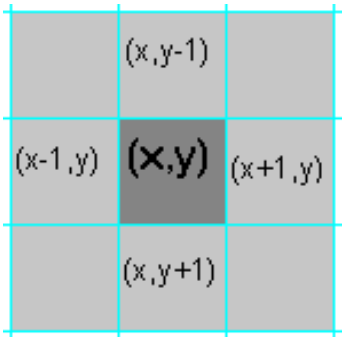


Figura 2.5 Píxel de interés y sus vecinos diagonales [14]

La suma de los anteriores define los ocho vecinos del píxel p, se denomina vecindad de tipo 8 y se denota N_8(p) [14].

2.2.4.1.2 CONECTIVIDAD

La conectividad entre píxeles es un concepto importante usado para establecer las fronteras de los objetos y las regiones componentes de una imagen. Para establecer si dos píxeles están conectados hemos de establecer si son adyacentes en algún sentido (por ejemplo si son 4-vecinos y si sus niveles de gris cumplen algún criterio de similitud) [14].

2.2.4.1.3 DISTANCIA

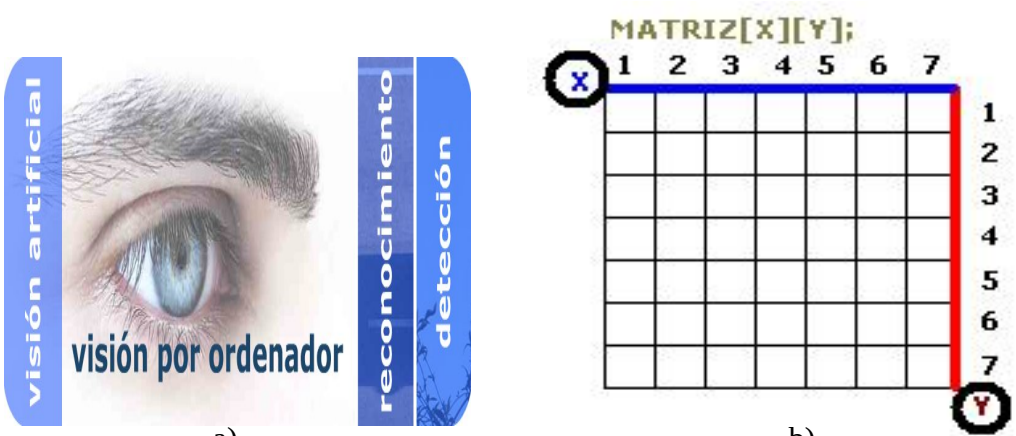
Con la distancia se quiere obtener el mínimo número de pasos elementales que se necesitan para ir de un punto a otro. Dados dos píxeles p y q con coordenadas (x,y), (s,t) respectivamente, se puede definir una función de distancia D como se muestra en la ecuación 1 [14]:

- La distancia euclídea entre p y q está dada como:

$$D_E(p,q) = \sqrt{(x-s)^2 + (y-t)^2} \quad (1)$$

2.2.5 IMAGEN DIGITAL

Una imagen digital se compone de una agrupación de píxeles, cada uno con un valor de intensidad o brillo asociado. Una imagen digital se representa mediante una matriz bidimensional, de forma que cada elemento de la matriz se corresponde con cada píxel en la imagen (ver figura 2.6) [9].



a) b)

Figura 2.6. a) Imagen Digital; b) Estructura matricial de una imagen [15]

2.2.5.1 PROCESAMIENTO

Es importante conocer el proceso de creación de una imagen digital a partir de datos adquiridos por cámaras, así como también saber cómo tales imágenes son representadas en un computador de forma que permita realizar su procesamiento.

Los elementos usuales de un sistema de procesamiento de imágenes son los siguientes: captura, almacenamiento, procesamiento, reconocimiento y presentación. En la fase de captura los rayos reflejados por los objetos deben ser capturados y convertidos en una señal eléctrica para poder ser procesados, esta tarea es realizada por la cámara. Si el dispositivo de captura es una

cámara analógica, entonces la señal eléctrica que produce generalmente está en uno de los dos estándares de video más comunes: NTSC[2] y PAL[3].

Esta señal análoga se debe digitalizar y decodificar para poder aplicar las distintas técnicas de procesamiento de imágenes y transformaciones morfológicas. Estas permiten su segmentación y la posible extracción de las características que llevan a la etapa de interpretación y reconocimiento [16].

2.2.5.2 CLASIFICACIÓN

Dependiendo del rango de los valores que pueda tomar cada píxel podemos distinguir los siguientes tipos de imágenes:

• **Imágenes binarias:** el rango está formado por los valores negro o blanco [0 1] únicamente. En la figura 2.7 se muestra una imagen de este tipo [9].

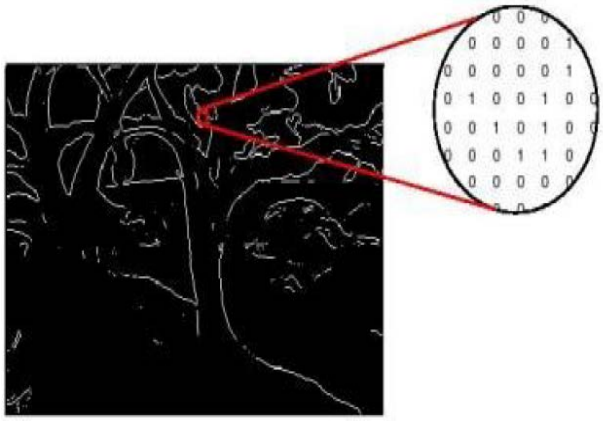


Figura 2.7 Imagen binaria [17]

NTSC[2] : Estándar americano de video, siglas de National Television System Committee

PAL[3]: Estándar europeo de video, siglas de Phase Alternation Line Ratio.

• **Imágenes de intensidad:** también conocidas como imágenes en escala de grises, existen hasta 256 niveles de grises, por lo que su rango se encuentra entre [0, 255]. En la figura 2.8 se muestra una imagen de este tipo [9].

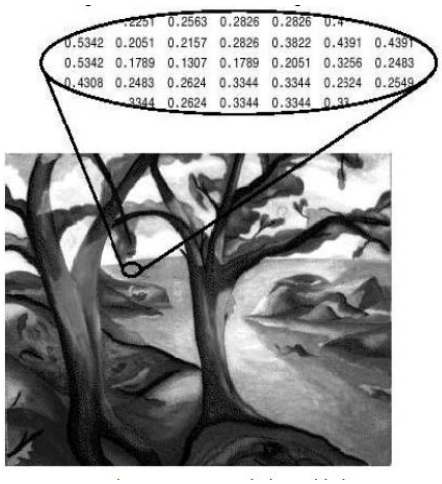


Figura 2.8 Imagen en escalas de grises [17]

• **Imágenes en color:** todo color se puede componer a partir de tres componentes básicas. El contenido de cada píxel de la imagen es una terna de valores, un valor por cada componente de color básico. En la figura 2.9 se muestra una imagen de este tipo [9].

Figura 2.9 Imagen en color [17]

2.2.5.3 ESPACIO DE COLOR

Para la implementación del proyecto se van a utilizar imágenes en color y sus características por lo que se va a dar una pequeña introducción sobre los distintos espacios de color que pueden ser utilizados en el procesamiento de imágenes. Un espacio de color es la forma por la que se puede especificar, crear y visualizar un color [9].

2.2.5.3.1 MODELO RGB

El modelo RGB (del inglés Red, Green, Blue) está basado en la síntesis aditiva de las intensidades de luz relativas al rojo, al verde y al azul para conseguir los distintos colores, incluyendo el negro y el blanco. La representación gráfica del modelo RGB (ver figura 2.10) se realiza mediante un cubo unitario con los ejes R, G y B.

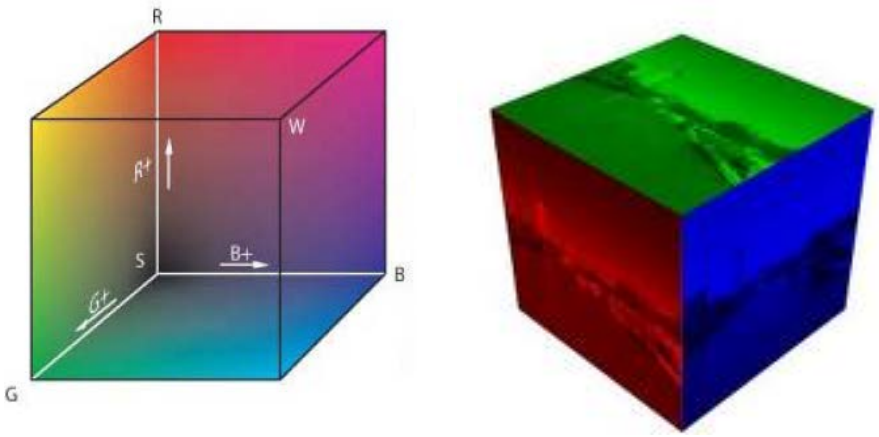


Figura 2.10 Representación del modelo de color RGB [18]

Las imágenes con modelo RGB contienen tres planos de imágenes independientes, uno para cada color primario. El procesamiento de imágenes en color, utilizando el modelo RGB, toma sentido cuando las imágenes se expresan naturalmente en términos de estos tres planos. Actualmente muchas cámaras a color utilizadas para adquirir imágenes digitales, utilizan el

formato RGB. Esto convierte al modelo RGB en un modelo de gran importancia para el procesamiento de imágenes [9].

2.2.5.3.2 MODELO HSV

Las siglas H, S y V corresponden a Tono (hue), Saturación (saturation) y valor (value) respectivamente. También se denomina HSB, siendo B el brillo (brighness). El sistema coordenado es cilíndrico, y el subconjunto de este espacio donde se define el color es una pirámide de base hexagonal, como se muestra en la figura 2.11.

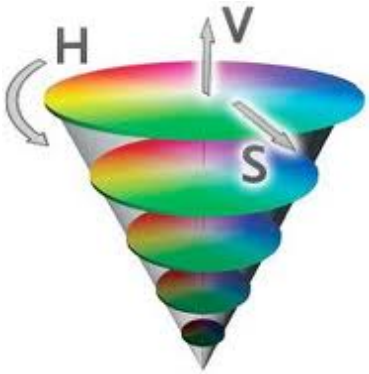


Figura 2.11 Representación gráfica del modelo de color HSV [19]

El área hexagonal corresponde a un valor de V = 1, conteniendo los colores más brillantes. El tono se mide como el ángulo alrededor del eje S. El rojo se sitúa a 0º, el verde a los 120º y el azul a los 240º. Los colores complementarios son aquellos que se encuentren a 180º del señalado. El rango de S se extiende desde 0 (coincidiendo con el eje de la pirámide) hasta 1, coincidiendo con el final del área hexagonal de la pirámide. La obtención de este espacio de color a partir del RGB se describe en las ecuaciones 2, 3 y 4 [9]:

$$H = arcos\ \frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{((R-G)^2+(R-B)(G-B))}} \quad (2)$$

$$S = 1 - 3\frac{\min(R,G,B)}{R+2G} \quad (3)$$

$$V = \frac{(R+G+B)}{3} \quad (4)$$

donde R, G y B son los valores del canal rojo, verde y azul respectivamente.

2.2.5.3.3 MODELO YCBCR

YCBCR es una codificación no lineal del espacio de color RGB, usado comúnmente en la compresión de imágenes. El color es representado por la luminancia (Y) y por dos valores diferentes de color (Cb y Cr) que son características colorimétricas del color.

El parámetro Y indica la luminosidad o la claridad del color (que se pueden ver como un tono de gris), los parámetros Cb y Cr indican el tono del color: Cb ubica el color en una escala entre el azul y el amarillo, Cr indica la ubicación del color entre el rojo y el verde.

La obtención de este espacio de color a partir del RGB se describe en las ecuaciones 5, 6 y 7:

$$\mathrm{Y} = 0.299\mathrm{R} + 0.587\mathrm{G} + 0.114\mathrm{B} \quad (6)$$

$$Cr = R - Y \quad (7)$$

$$Cb = B - Y \quad (8)$$

Siendo R, G y B son los valores del canal rojo, verde y azul respectivamente. La sencillez de la transformación y la separación explicita de las componentes de luminancia y de

crominancia del color, hacen a este espacio de color un método atractivo para la modelación del color de la piel [9].

2.2.6 SEGMENTACIÓN

La segmentación es el proceso que divide una imagen en regiones u objetos cuyos píxeles poseen atributos similares, considerando el atributo más básico de segmentación la amplitud, aunque los contornos de la imagen también han de ser considerados. La segmentación por ende es uno de los procesos más importantes de un sistema automatizado de visión ya que permite extraer los objetos de la imagen para su posterior descripción y reconocimiento [20].

Las técnicas de segmentación pueden encuadrarse en tres grupos fundamentales: técnicas basadas en la detección bordes, técnicas basadas en regiones y técnicas de umbralización.

2.2.6.1 SEGMENTACIÓN POR BORDES

Se centran en la detección de contornos. Delimitan el borde de un objeto y segmentan los píxeles dentro del contorno como pertenecientes a ese objeto. Su desventaja consiste en conectar contornos separados o incompletos, lo que los hace susceptibles a fallas [21].

2.2.6.2 SEGMENTACIÓN POR REGIONES

En esta aproximación todos los píxeles que correspondan a un objeto se agrupan juntos y son marcados para indicar que pertenecen a una región. Los píxeles son asignados a regiones según algún criterio que los distingue del resto de la imagen. Un criterio muy estricto puede provocar fragmentación mientras uno poco estricto ocasiona uniones indeseadas [22].

2.2.6.3 SEGMENTACIÓN POR UMBRAL

Esta técnica segmenta la imagen píxel por píxel, es decir, no toman en consideración el valor de los píxeles vecinos para el proceso. Si el valor de un píxel está dentro de un rango de valores especificado para un objeto el píxel es segmentado. Son efectivas cuando los objetos y el fondo de la imagen tienen rangos de valores diferentes y existe un contraste marcado entre ellos. Como la información de los píxeles vecinos es ignorada, las fronteras de regiones borrosas pueden ocasionar problemas [21].

2.2.6.4 THRESHOLDING

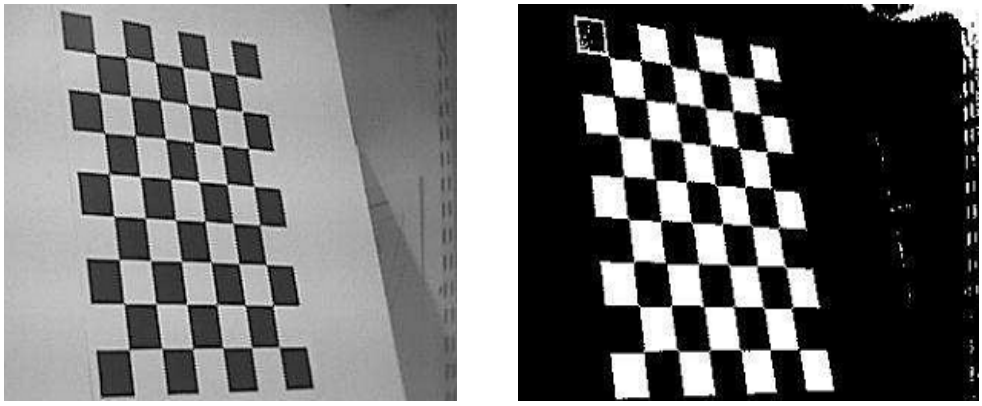
Uno de los acercamientos a segmentación de imágenes más importantes es thresholding. Tradicionalmente, se ha efectuado una forma simple de definir el rango de valores de nivel de gris en la imagen original que consiste en elegir los pixeles en este rango según pertenezcan o no al fondo: se toman los que sí pertenecen y se rechazan todos los demás. Una imagen de este tipo se muestra como una imagen binaria (de dos niveles) utilizando blanco y negro u otros colores para distinguir las regiones (no hay una convención estándar sobre cuáles son los rasgos de interés, si los blancos o los negros, así que la elección varía en cada caso). Este tipo de operación se denomina thresholding [20].

2.2.6.5 BINARIZACIÓN

La binarización de una imagen consiste en un proceso de reducción de la información de la misma, en la que sólo persisten dos valores: verdadero y falso. En una imagen digital, estos valores, verdadero y falso, pueden representarse por los valores 0 y 1 o, más frecuentemente, por los colores negro (valor de gris 0) y blanco (valor de gris 255).

En el procesamiento y análisis de imágenes, la binarización se emplea para separar las regiones u objetos de interés en una imagen del resto. Las imágenes binarias se usan en

operaciones booleanas o lógicas para identificar individualmente objetos de interés o para crear máscaras sobre regiones [23]. En la figura 2.12, se observa un ejemplo de la binarización de una imagen.



a) b)

Figura 2.12: a) Imagen a escala de grises, b) imagen binarizada

2.2.7 PROCESO MORFOLÓGICOS DE IMÁGENES

Para la definición de las operaciones básicas del procesamiento morfológico de imágenes binarias, se precisa repasar algunas nociones básicas de la teoría de conjuntos.

- **Inclusión**: Y es subconjunto de X si todos los elementos de Y pertenecen a X (ver ecuación 8, la cual es llamada condición de inclusión) :

$$Y \subseteq X \Leftrightarrow (p \in Y \Rightarrow p \in X) \quad (8)$$

La inclusión es reflexiva $(X \subseteq X)$, antisimétrica $(Y \subseteq X \text{ y } X \subseteq Y \Rightarrow X = Y)$y transitiva $(Y \subseteq X \text{ y } X \subseteq Z \Rightarrow Y \subseteq Z)$. Un conjunto que cumpla estas tres condiciones se dice que es un conjunto totalmente ordenado.

- **Intersección**: La intersección de dos conjuntos X e Y es el conjunto de los elementos que pertenecen a ambos conjuntos (ver ecuación 9, es conocida como condición de intersección):

$$X \cap Y = (p \mid p \in X \text{ y } p \in Y) \tag{9}$$

La intersección es conmutativa e asociativa. Esta última propiedad es importante en Morfología e indica que $X \cap X = X$.

- **Unión**: La unión de dos conjuntos se constituye por los elementos que pertenecen a uno o al otro (ver ecuación 10, también conocida como condición de unión):

$$X \cup Y = (p \mid p \in X \text{ o } p \in Y) \tag{10}$$

Al igual que la intersección, la unión de conjuntos es conmutativa e asociativa.

El objetivo de las transformaciones morfológicos es la extracción de estructuras geométricas en los conjuntos sobre los que se opera, mediante la utilización de otro conjunto de forma conocida, al que se le denomina elemento estructurante. El tamaño y forma del elemento estructurante se elige, a priori, de acuerdo con la morfología sobre la que va a interseccionar y en función de la obtención de formas que se desea extraer. En la figura adjunta aparecen algunos tipos de elementos estructurantes empleados en el procesamiento morfológico [24] [25].

Figura 2.13 Elementos estructurantes típicos [25]

2.2.7.1 OPERACIONES MORFOLÓGICAS ELEMENTALES

2.2.7.1.1 EROSIÓN

La erosión del conjunto A por el elemento estructurante B se define como el conjunto de todos los puntos z, pertenecientes a A, de forma que cuando el elemento estructurante B se traslada a ese punto, el elemento queda incluido en A (ecuación 11):

$$\varepsilon_B(A) = A\boldsymbol{\Theta}B = \{z|(B)_z \subseteq A\} \tag{11}$$

Por lo tanto, la transformación de la erosión es el resultado de comprobar si el conjunto B está completamente incluido en el conjunto A. Cuando no ocurre, el resultado de la erosión es el conjunto vacío.

La ecuación anterior puede reformularse en términos de una intersección de conjuntos trasladados, donde las traslaciones vienen definidas por el elemento estructurante de la siguiente manera (ecuación 12):

$$A\boldsymbol{\Theta}B = \bigcap_{b\epsilon B} A_{-b} \tag{12}$$

Es decir, el resultado de la erosión de A por B es la intersección de todas las traslaciones de A por los puntos –b, donde b∈ B.

El efecto de una operación de erosión puede observarse en la Fig. 2.14, en la que un elemento estructurante, en forma de disco circular, hace desaparecer los conjuntos de menor tamaño al elemento. El resto de objetos, los que quedan tras la transformación, son degradados [24] [25].

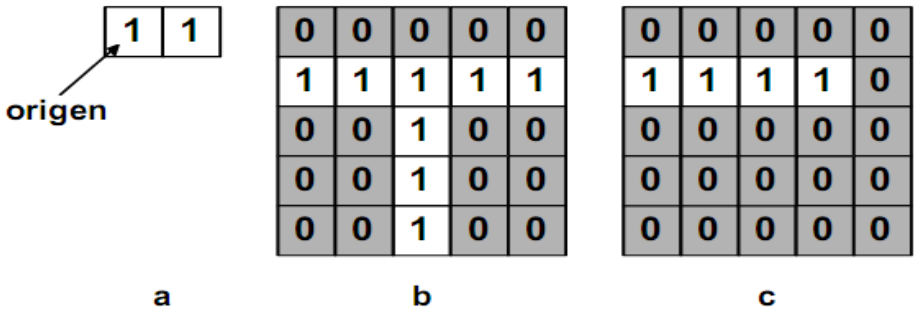


Figura 2.14 (a) Elemento estructural B. (b) Imagen original. (c) Resultado d la erosionada [26]

Se dice, por tanto, que la erosión es una degradación de la imagen. En términos de teoría de conjuntos, el conjunto erosionado queda contenido en el original.

2.2.7.1.2 DILATACIÓN

El resultado de la dilatación es el conjunto de puntos origen del elemento estructurante B tales que el elemento estructurante complementado contiene algún elemento del conjunto A, cuando el elemento se desplaza por el espacio que contiene a ambos conjuntos. (ver ecuación 13)

$$\delta_B(A) = A \oplus B = \{z \big| (\widehat{B})_z \cap A \neq \emptyset\} \tag{13}$$

Se puede rescribir la ecuación 6.13 como una unión de conjuntos trasladados donde las traslaciones vienen definidas por el elemento estructurante (ver ecuación 14):

$$A \oplus B = \cup_{b \epsilon B} A_b \tag{14}$$

Que en palabras establece que el resultado de la dilatación de A por B es la unión de todas las traslaciones de A por los puntos b, donde b∈ B.

En la Fig. 2.15 puede observarse el efecto de una operación de dilatación, donde un elemento estructurante B de forma circular aumenta la definición de los objetos de la imagen original [24][25].

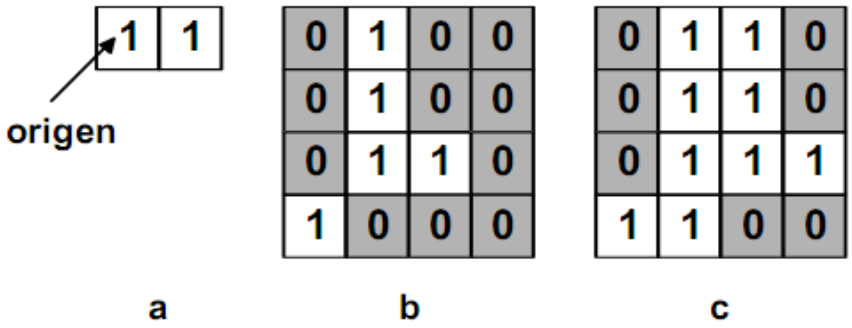


Figura 2.15 Ejemplo de dilatación (a) Elemento estructural B. (b) Imagen original. (c) Resultado de la dilatación [26]

La dilatación binaria opera engrandando los objetos, cerrando los agujeros y las grietas, en proporción al tamaño de elemento estructurante que se le aplique.

2.2.7.2 FILTROS MORFOLÓGICOS ELEMENTALES

En base a las operaciones morfológicas elementales, se pueden construir filtros morfológicos que pueden utilizarse en lugar de los filtros lineales estándar. Mientras que los filtros lineales suelen distorsionar la forma geométrica subyacente de la imagen, los morfológicos la dejan intacta [25].

2.2.7.2.1 APERTURA

Como se ha visto anteriormente, tras la aplicación de una erosión pueden eliminarse pequeños objetos. Sin embargo, tiene el inconveniente de disminuir a su vez el tamaño del resto de los objetos. Este efecto puede ser subsanado con una operación en cascada de erosión y

dilatación binaria con igual elemento estructurante. Esta operación recibe el nombre de apertura. La apertura de una imagen A por un elemento estructurante B se expresa:

$$\gamma_B(A) = A \circ B = (A \boldsymbol{\Theta} B) \oplus B = \delta_B(\varepsilon_B(A)) \tag{15}$$

Si A no cambia con la apertura con B, se dice que A es abierta respecto a B.

La apertura morfológica binaria elimina todos los objetos que no están completamente contenidos en el elemento estructurante, y además no disminuye el tamaño a los objetos que superen la erosión. Por esto, la operación puede resultar ideal para la eliminación de ruido en imágenes binarias [25].

2.2.7.2.2 CIERRE

De forma análoga al caso anterior, se define la operación de cierre de una imagen A por un elemento estructurante B como la dilatación de A por B, seguida de la erosión por el mismo elemento estructurante.

$$\varphi_B(A) = A \cdot B = (A \oplus B) \boldsymbol{\Theta} B = \varepsilon_B(\delta_B(A)) \tag{16}$$

Si A no cambia con el cierre por B se dice que A es cerrada respecto a B.

La aplicación de la dilatación, tal y como se ha visto anteriormente, produce una extensión de la imagen original. Gracias a ella pueden rellenarse huecos de los objetos que forman la imagen, pero a su vez se aumenta el tamaño de éstos. La aplicación de la erosión devuelve a los objetos el tamaño inicial que tenían. El cierre morfológico binario permite rellenar estructuras sin modificar el tamaño inicial de éstas [25].

2.2.8 MATLAB

MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. Además, integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían adicionalmente, sin necesidad de hacer uso de la programación tradicional.

Por otra parte, dispone en la actualidad de un amplio abanico de programas de apoyos especializados, denominados Toolboxes, que extienden significativamente el número de funciones incorporadas en el programa principal. Estos Toolboxes cubren prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación, destacando entre ellos el 'toolbox' de proceso de imágenes, señal, control robusto, estadística, análisis financiero, matemáticas simbólicas, redes neurales, lógica difusa, identificación de sistemas, simulación de sistemas dinámicos, etc.

Cabe destacar, es un entorno de cálculo técnico, que se ha convertido en estándar de la industria, con capacidades no superadas en computación y visualización numérica. De forma coherente y sin ningún tipo de fisuras, integra los requisitos claves de un sistema de computación técnico: cálculo numérico, gráficos, herramientas para aplicaciones específicas y capacidad de ejecución en múltiples plataformas. Esta familia de productos proporciona al estudiante un medio de carácter único, para resolver los problemas más complejos y difíciles [27].

2.2.8.1 LECTURA, ESCRITURA Y VISULIZACION DE IMÁGENES EN MATLAB.

Para leer imágenes contenidas en un archivo al ambiente de Matlab se utiliza la función “imread”, cuya sintaxis es:

imread('nombre del archivo')

Donde el nombre del archivo es una cadena de caracteres conteniendo el nombre completo de la imagen con su respectiva extensión, los formatos de imágenes que soporta Matlab son los mostrados en la tabla 1.1

Tabla 2.1 Formatos y extensiones de imágenes soportadas por Matlab.

Formato	Profundidad de Color
BMP (.bmp)	· 1 (Mapa de bits) · 4-8 bits (Escala grises) · 8 bits (Color Indexado) · 24 bits(RGB)
GIF *Graphics Interchange Format* (.gif)	· 8 bits (256 colores)
PICT (.pct; .pic)	· RGB: 16/32 bits · Escala Grises: 2,4 8 bits
JPEG *Joint Photographic Expert Group* (.jpg; .jpe)	24 bits
PHOTOSHOP (.psd)	32 bits
TARGA (.tga; .vda;.icb;.vst)	16, 24 y 32 bits
PNG *Portable Networks Graphics* (.png)	24 bits
TIF *Tag Image File Format* (.tif)	32 bits

Fuente: González S. Imagen Digital. [28]

Para introducir una imagen guardada en un archivo con alguno de los formatos especificados en la tabla anterior solo tiene que usarse la función "imread"y asignar su resultado a una variable que representará a la imagen. De tal forma que si se quisiera introducir la imagen contenida en el archivo data.jpg a una variable para su procesamiento en Matlab, entonces se tendría que escribir en línea de comandos [29]:

>>image=imread(’data.jpg’);

Con ello la imagen contenida en el archivo *data.jpg* quedará contenida en la variable “image”.Una vez que la imagen está contenida en una variable de búsqueda y sustitución de matlab, es posible utilizar las funciones para procesar la imagen. Por ejemplo, una función que permite encontrar el tamaño de la imagen es size(variable) [29]

>>[m, n]=size(image);

En donde *m* y *n* contendrán los valores de las dimensiones de la imagen.

Para grabar el contenido de una imagen en un archivo se utiliza la función imwrite (variable,’nombre del archivo’), en donde *variable* representa la variable que contiene a la imagen y *nombre del archivo* el nombre del archivo con su respectiva extensión. Suponiendo que la variable *image2* contiene la imagen que nos interesa grabar en el archivo *dato2.jpg* tendríamos que escribir [29]:

>>imwrite(image2, ’data2.jpg’) ;

Después que se realiza un procesamiento con la imagen, es necesario desplegar el resultado obtenido, la función imshow(variable) permite desplegar la imagen en una ventana en el ambiente de trabajo de Matlab. Si la variable a desplegar por ejemplo, es “face" al escribir en la línea de comandos [29]:

>>imshow(face);

En el caso de que se quisiera mostrar una imagen en Matlab, se puede visualizar lo antes mencionado con el ejemplo de la figura 2.16:

```
>>Imagen= imread('C:\Users\usuario\Desktop\Kath.jpg');
>>imshow(Imagen);
```

Figura 2.16 Visualización de una imagen en Matlab [Fuente propia]

Si se quiere convertir la imagen de RGB a escala de grises, esto se logra con el comando rgb2gray(imagen), se puede visualizar lo antes mencionado con el ejemplo de la figura 2.17.

```
>>Imagen= imread('C:\Users\usuario\Desktop\Kath.jpg');
>>gris=rgb2gray(Imagen),
>>imshow(gris);
```

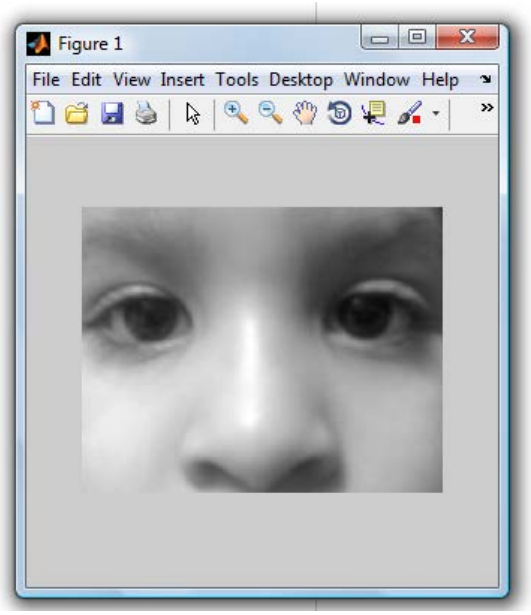


Figura 2.17 Visualización de una imagen en Matlab a escala de grises [Fuente propia]

2.2.8.2 ACCESO A PIXEL Y PLANOS EN LAS IMÁGENES DE MATLAB

El acceso a píxel de una imagen es una de las operaciones más comunes en visión computacional y en Matlab está sumamente simplificado; sólo bastará con indexar el píxel de interés en la estructura de la imagen. Si se considera que se tiene una imagen *image1* en escala de grises (Figura 2.4) y se desea obtener su valor de intensidad en el píxel especificado por *m*=100 y *n*=100; solo se tendría que escribir:

```
>>image1(100,100)
ans =
```

De igual forma si se desea cambiar el valor de este píxel a negro, es decir asignarle el valor de *0* lo que tendría que escribirse en línea de comandos es:

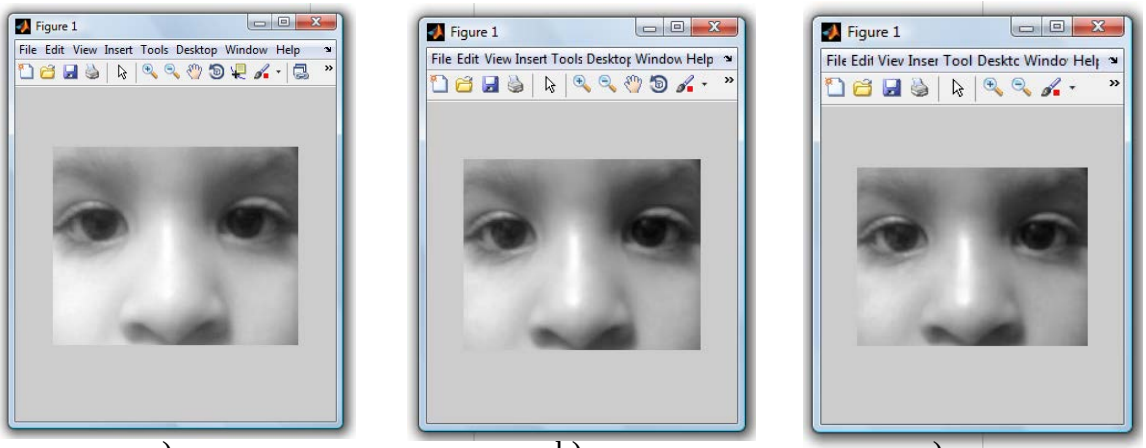
```
>>image1(100,100)=0 ;
```

En el caso de imágenes a escala de grises estas solo tienen un plano, constituido por la matriz *m* x *n* que contiene los valores de intensidad para cada índice.

Sin embargo, las imágenes de color cuentan con más de un plano. En el caso de imágenes RGB (tal como se explicó arriba) estas cuentan con 3 planos uno para cada color que representa. Se considera ahora que la imagen RGB está contenida en la variable image2, y se desea obtener cada uno de los planos que la componen. Entonces se escribiría [29]:

```
>>planeR=image2( :, :,1) ;
>>planeG=image2( :, :,2) ;
>>planeB=image2( :, :,3) ;
```

Los planos resultantes por los anteriores comandos son mostrados en la figura 2.18.



a) b) c)

Figura 2.18 Tres planos correspondientes a) PlanoR. b) PlanoG. c) PlanoB

[Fuente propia]

Si se deseara manipular un píxel de una imagen a color RGB este tendrá un valor para cada uno de sus planos correspondientes. Supóngase que se tiene la imagen RGB contenida en la

variable image2y deseamos obtener el valor del píxel *m*=100 y *n*=100 para cada uno de los diferentes planos R, G y B. Se tendría que escribir:

```
>>valueR=image2(100,100,1) ;
>>valueG=image2(100,100,2) ;
>>valueB=image2(100,100,3) ;
```

Lo cual dará como resultado una tripleta de valores. De igual forma que con el caso de escala de grises podemos modificar este píxel a otro color mediante el cambio de su valor en cada uno de sus respectivos planos; por ejemplo un cambio a color *blanco* mediante:

```
>>image2(100,100,1)=255;
>>image2(100,100,2)=255;
>>image2(100,100,3)=255;
```

En ocasiones resulta preferible saber el color o la intensidad de gris (el valor del píxel) de forma iterativa; Es decir, tener la posibilidad de seleccionar un píxel en una región y obtener el valor de este. Esta posibilidad es ofrecida por la función impixel, la cual iterativamente entrega el valor (uno o tres) del píxel seleccionado que aparezca en la ventana desplegada por la función imshow. El formato de esta función es:

```
value=impixel;
```

Donde value representa un escalar, en el caso de que la imagen sea a escala de grises o bien un vector de 1 x 3 con los valores correspondientes a cada uno de los planos RGB.

Para utilizar esta función es necesario antes, desplegar la imagen con la función imshow. Una vez desplegada se llama a la función y cuando el cursor del ratón este sobre la superficie de la imagen cambiará a una +. Cuando se presione el botón izquierdo del ratón se seleccionará el

píxel, el cual se puede seleccionar otra vez en caso de que se allá cometido un error a la hora de posicionar el ratón, ya que la función seguirá activada hasta que se presione la tecla de enter.

Una operación importante en visión computacional es el determinar un perfil de la imagen; es decir convertir un segmento de la imagen a una señal unidimensional para analizar sus cambios. Esto es de especial significado en la visión estereo en donde se analizan para los algoritmos segmentos epipolares de cada cámara. Matlab dispone de la función improfile que permite trazar el segmento interactivamente con el ratón, desplegando después el perfil de la imagen en una gráfica diferente. Esta función necesita que la imagen original sea previamente desplegada mediante la función imshow. Debe de considerarse que si la imagen es a escala de grises, el perfil mostrará solo una señal correspondiente a las fluctuaciones de las intensidades de la imagen, sin embargo, si la señal es de color RGB esta mostrará un segmento de señal para cada plano. Para la utilización de esta función solo es necesario escribir en línea de comandos:

>>improfile

Como es una función iterativa en cuanto el ratón se encuentra en la superficie de la imagen el puntero cambiará de símbolo a una +, de esta manera se puede, mediante el establecimiento de una línea en la imagen, configurar el perfil deseado [29].

2.2.8.3 IMÁGENES BINARIAS Y SEGMENTACIÓN POR UMBRAL

Una imagen binaria es una imagen en la cual cada píxel puede tener solo uno de dos valores posibles 1 o 0. Como es lógico suponer una imagen en esas condiciones es mucho más fácil encontrar y distinguir características estructurales. En la visión por computador, el trabajo con imágenes binarias es muy importante ya sea para realizar segmentación por intensidad de la imagen, para generar algoritmos de reconstrucción o reconocer estructuras.

La forma más común de generar imágenes binarias es mediante la utilización del valor umbral de una imagen a escala de grises; es decir se elige un valor limite (o bien un intervalo) a partir del cual todos los valores de intensidades mayores serán codificados como 1 mientras que los que estén por debajo serán codificados a cero. En Matlab este tipo de operaciones se realizan de forma bastante sencilla utilizando las propiedades de sobrecarga de los símbolos relaciónales [29].

2.2.8.4 OPERACIONES MORFOLÓGICAS

Una de las operaciones más utilizadas en visión sobre imágenes previamente binarizadas es las operaciones morfológicas. Las operaciones morfológicas son operaciones realizadas sobre imágenes binarias basadas en formas. Estas operaciones toman como entrada una imagen binaria regresando como resultado una imagen también binaria. El valor de cada píxel de la imagen binaria resultado es basado en el valor del correspondiente píxel de la imagen original binaria y de sus vecinos. Entonces eligiendo apropiadamente la forma de los vecinos a considerar, puede construirse operaciones morfológicas sensibles a una forma en particular.

Las principales operaciones morfológicas son la dilatación y la erosión. La operación de dilatación adiciona píxeles en las fronteras de los objetos, mientras la erosión los remueve. En ambas operaciones como se menciono se utiliza una rejilla que determina cuales vecinos del elemento central de la rejilla serán tomados en cuenta para la determinación del píxel resultado.

La rejilla es un arreglo cuadricular que contiene unos y ceros, en los lugares que contienen uno serán los vecinos de la imagen original con respecto al píxel central, los cuales serán tomados en consideración para determinar el píxel de la imagen resultado, mientras que los lugares que tengan ceros no serán tomados en cuenta.

Una vez determinado el tamaño de la rejilla y su configuración. Se aplica la operación morfológica. En el caso de la dilatación, si alguno de los píxeles de la rejilla configurados como

unos coincido con al menos uno de la imagen original el píxel resultado es uno. Por el contrario en la erosión todos los píxel de la rejilla configurado como unos deben coincidir con todos los de la imagen si esto no sucede el píxel es 0.

En Matlab las funciones utilizadas para realizar estas dos operaciones morfológicas son imerode e imdilate. El formato de ambas funciones es:

```
ImageR=imerode(ImageS,w);
ImageR=imdilate(ImageS,w);
```

Donde ImageR es la variable que recibe a la imagen resultado, ImageS es la imagen binaria origen a la que se desea aplicar la operación morfológica y w es una matriz de unos y ceros que determina el formato y estructura de la rejilla [29].

Se puede obtener la dilatación y erosión de una imagen en Matlab con los comandos imdilate e imerode, como se explica en el siguiente ejemplo de la dilatación de la figura 2.19 y la erosión de la figura 2.20:

```
>>Imagen= imread('C:\Users\usuario\Desktop\Kath.jpg');
>>se=strel('disk',10);
>>imd=imdilate(Imagen,se);
>>figure
>>subplot(1,2,1),imshow(Imagen);
>>subplot(1,2,2),imshow(imd);
```

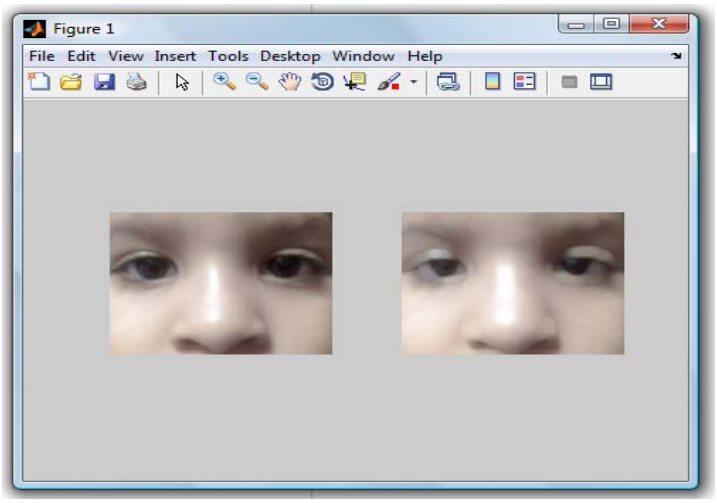


Figura 2.19 Dilatación de una imagen en Matlab con elemento estructurante disco [Fuente propia]

```
>>Imagen= imread('C:\Users\usuario\Desktop\Kath.jpg');
>>se=strel('disk',10);
>>imr=imerode(Imagen,se);
>>subplot(1,2,1),imshow(Imagen);
>>subplot(1,2,2),imshow(imr);
```

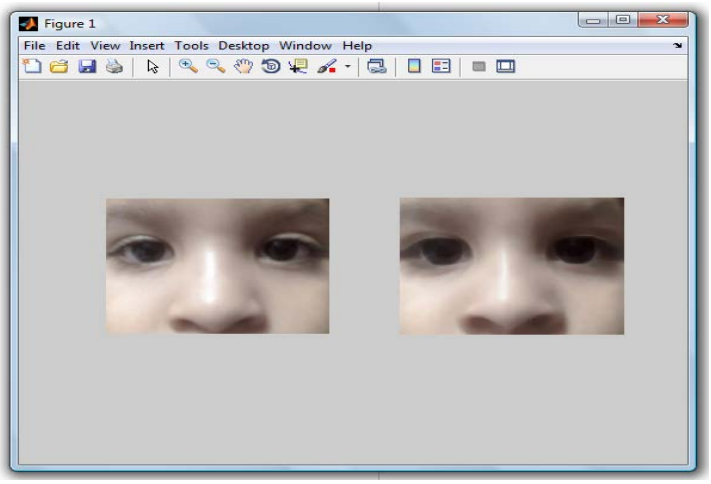


Figura 2.20 Erosión de una imagen en Matlab con elemento estructurante disco [Fuente propia]

Además de estas operaciones se puede obtener en Matlab la apertura de una imagen, que consiste en recorrer la imagen con un elemento estructurante, donde este elemento está totalmente incluido en uno de los objetos presentes y todos los puntos del elemento estructurante pertenecen al objeto. Esto se logra con el comando imopen(imagen), como se muestra en el ejemplo de la figura 2.21:

```
>>Imagen= imread('C:\Users\usuario\Desktop\Kath.jpg');
>>se=strel('disk',10);
>>imaper=imopen(Imagen,se);
>>figure
>>subplot(1,2,1),imshow(Imagen);
>>subplot(1,2,2),imshow(imaper);
```

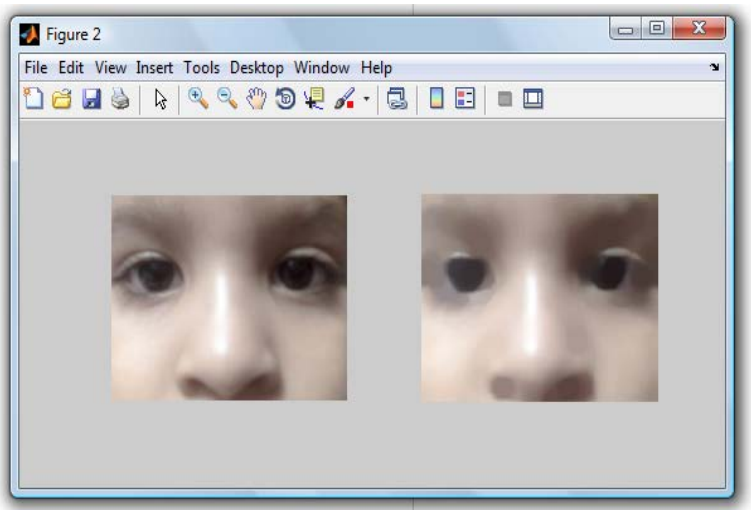


Figura 2.21 Apertura de una imagen en Matlab con el mismo elemento estructurante [Fuente propia]

Se puede obtener en Matlab el cierre de una imagen, que consiste en recorrer la imagen con un elemento estructurante, donde este elemento se encuentra totalmente fuera de los objetos

(fondo de la imagen), donde entonces todos los pixeles del elemento estructurante formaran parte del fondo de la imagen; esto se logra con el comando imclose(imagen) como se muestra en la figura 2.22:

```
>>Imagen= imread('C:\Users\usuario\Desktop\Kath.jpg');
>>se=strel('disk',20);
>>imcierre=imclose(Imagen,se);
>>figure,>>subplot(1,2,1),imshow(Imagen);
>>subplot(1,2,2),imshow(imcierre);
```

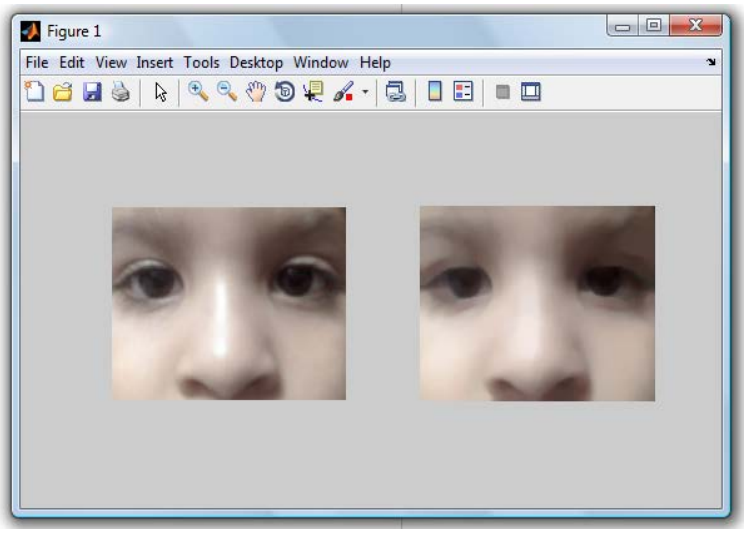


Figura 2.22 Cierre de una imagen en Matlab con el mismo elemento estructurante. [Fuente propia]

2.2.9 INTEFAZ GRÁFICA EN MATLAB (GUIDE)

GUIDE (Graphical User Interfaces Development Environment), se conoce como una aplicación de desarrollo de interfaces graficas de usuario, siendo este un entorno de programación visual disponible en MATLAB para realizar y ejecutar programas que necesiten ingreso continuo

de datos. Matlab permite desarrollar fácilmente un conjunto de pantallas (paneles) con botones, menús, ventanas, etc., que permiten utilizar de manera muy simple programas realizados dentro de este entorno. Este conjunto de herramientas se denomina interfaz gráfica de usuario (GUI). Tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++. Al seleccionar una GUIDE en Matlab se nos abrirá una venta de dialogo mostrada en la figura 2.23 [30]:

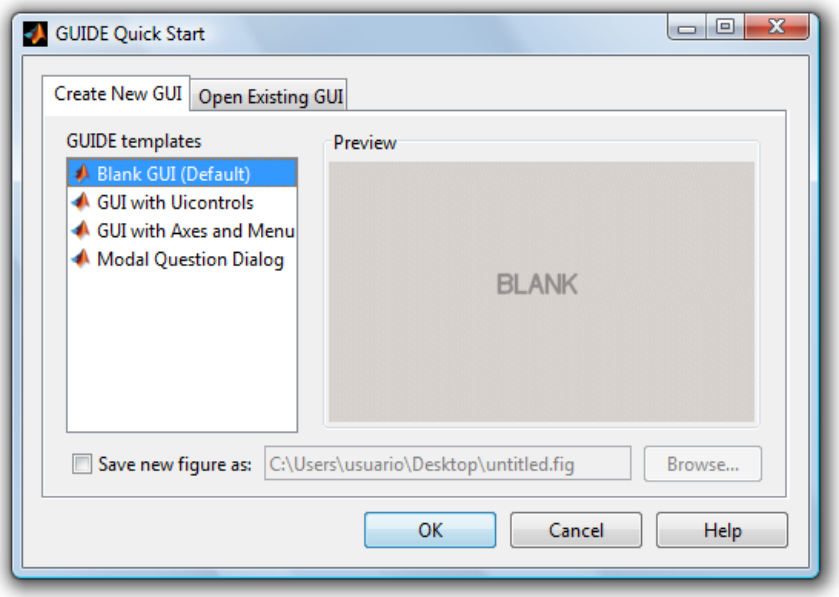


Figura 2.23 Ventana de Dialogo [24]

a) **Blank GUI (Default):** La opción de interfaz gráfica de usuario en blanco (viene predeterminada), nos presenta un formulario nuevo, en el cual podemos diseñar nuestro programa.

b) **GUI with Uicontrols:** Esta opción presenta un ejemplo en el cual se calcula la masa, dada la densidad y el volumen, en alguno de los dos sistemas de unidades. Podemos ejecutar este ejemplo y obtener resultados.

c) **GUI with Axes and Menu:** Esta opción es otro ejemplo el cual contiene el menú File con las opciones Open, Print y Close. En el formulario tiene un Pop-up Menu, un Push Button y un objeto Axes, podemos ejecutar el programa eligiendo alguna de las seis opciones que se encuentran en el menú despegable y haciendo click en el botón de comando.

d) **Modal Question Dialog**: Con esta opción se muestra en la pantalla un cuadro de diálogo común, el cual consta de una pequeña imagen, una etiqueta y dos botones Yes y No, dependiendo del botón que se presione, el GUI retorna el texto seleccionado (la cadena de caracteres 'Yes' o 'No') [30].

Al elegir la opción, Blank GUI, nos muestra la ventana del ambiente de desarrollo de interfaz gráfica de usuario (GUIDE), figura 2.24

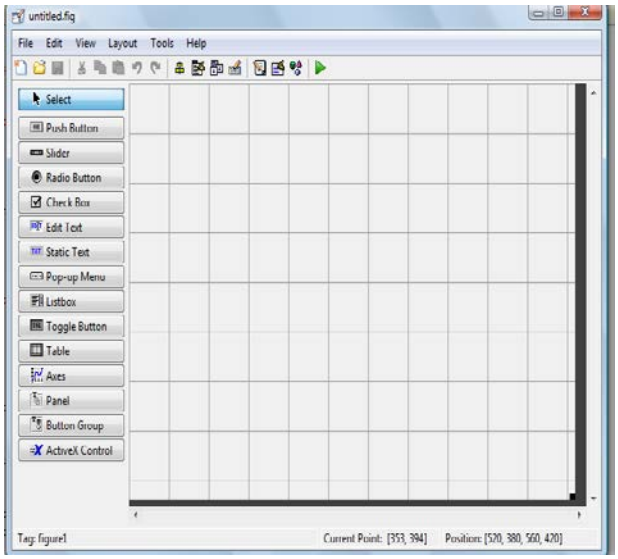


Figura 2.24 ventana principal de GUIDE [30]

En esta ventana se presenta el área de diseño el cual deben colocarse cada uno de los controles que conforman a la interfaz. Una vez que los controles están en posición, se editan las funciones de llamada (Callback) de cada uno de ellos, escribiendo el código de MATLAB que se ejecutará cuando el control sea utilizado. Siempre será difícil diseñar GUIs, pero no debería ser difícil implementarlas. GUIDE está diseñado para hacer menos complejo el proceso de aplicación de la interfaz gráfica [30].

2.2.9.1 CONTROLES DE INTERFAZ CON EL USUARIO

Los controles de la interfaz con el usuario en MATLAB se especifican con la orden **uicontrol**. Estos controles tienen mucho en común con los menús de la interfaz con el usuario, pero los primeros tienen muchos estilos. La sintaxis de **uicontrol** es:

k = **uicontrol** (‘Style’, ‘especificacion de estilo’, ……
‘String’, ‘cadena para exhibir’, …..
‘Value’, [valor], …..
‘BackgroundColor’, [r,g,b], …..
‘Max’ [valor], …..
‘Min’ [valor], …..
‘Position’, [izq, base, ancho, alto], …..
‘CallBack’, ‘cadena de invocacion’)

Donde ‘especificación de estilo’ es una de las cadenas pop-up, push, radio, check box, slider, edit (texto editable), text (texto estático), frame o axes [30].

- **Texto estático (text):** El texto estático puede exhibir símbolos, mensajes o incluso valores numéricos en una GUI y puede colocarse en el lugar apropiado. Este control no tiene cadena de invocación. A continuación se muestra un ejemplo de texto estático en la figura 2.25 [30].

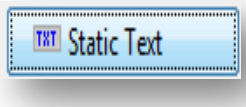


Figura 2.25 Static text de la GUIDE [30]

- **Menú desplegable (pop-up)**: Estos menús desplegables difieren de los menús de interfaz con el usuario en que pueden aparecer en cualquier punto del panel, mientras que los menús de interfaz con el usuario solo se encuentran en la parte superior de la ventana. Ejemplo del Popup menú en la figura 2.26 [29].

Figura 2.26 Pop-up menú de la GUIDE [30]

- **Botón (push)**: Los botones son pequeños objetos de la pantalla generalmente acompañados con texto. Al presionar el botón con el ratón, se producirá una acción que será ejecutada por Matlab. ejemplo del Push button en la figura 2.26 [30].

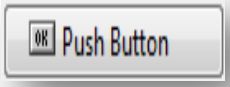


Figura 2.27 Push button de la GUIDE [30]

- **Casilla de Verificación (Check box)**: Las casillas de verificación están diseñadas para realizar operaciones de encendido/apagado. La casilla activa o desactiva, por ejemplo, la aparición de los ejes en una interfaz. Ejemplo del Check box en la figura 2.27

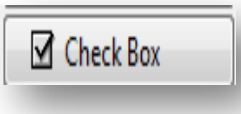


Figura 2.28 Checkbox de la GUIDE [30]

- **Botón de radio (radio):** Cuando se usa un solo botón de radio, es igual que la casilla ve verificación. Sin embargo, cuando se usan en grupo, estos son mutuamente exclusivos, es decir, si un botón de radio está encendido, los demás estarán apagados, mientras que las casillas de verificación son independientes entre sí. Ejemplo del Radio button en la figura 2.29.

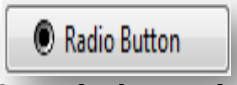


Figura 2.29 Radio button de la GUIDE [30]

- **Control deslizante (slider):** Es un dispositivo que permite modificar un parámetro de forma continua. Ejemplo del Edit text en la figura 2.30

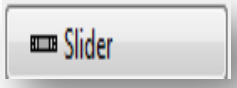


Figura 2.30 Slider de la GUIDE [30]

- **Texto editable (edit):** El dispositivo de texto editable le permite al usuario introducir una cadena. Puede aceptar valores numéricos en forma de vector o matriz como una cadena mediante el mismo dispositivo. La cadena de entrada puede convertirse a valores numéricos mediante la instrucción str2num. Ejemplo del Edit text en la figura 2.31.

Figura 2.31 Edit text de la GUIDE [30]

- **Uso de ejes para graficación (Axes)**: Generalmente es necesario desplegar varias gráficas dentro de una interfaz. El comando axes abre un eje (gráfica) en un punto específico dentro de un panel. Ejemplo del axes en la figura 2.32.

Figura 2.32 Axes de la GUIDE [30]

- **Cajas de lista (Listbox)**: El componente **Listbox** muestra una lista de artículos y permite al usuario, seleccionar una o más opciones de la lista. Al respecto puede observarse un ejemplo del Listbox en la figura 2.33.

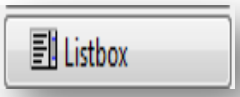


Figura 2.33 Listbox de la GUIDE [30]

CAPÍTULO III. MARCO METODOLÓGICO

3.1 TIPO DE LA INVESTIGACIÓN

En esta fase de la investigación se define la estrategia para el tratamiento metodológico, categorizando la investigación en la modalidad de proyecto factible ya que consiste en: “la investigación, elaboración y desarrollo de una propuesta de un modelo operativo viable, cuyo propósito es la búsqueda de solución de un problema. La propuesta debe tener apoyo en una investigación de tipo documental, de campo o un diseño que incluya ambas modalidades” [31].

No obstante cabe destacar debido a la relación interna que guarda los proyectos factibles en cuanto a otras investigaciones nombradas anteriormente. La investigación es de tipo documental ya que: “la investigación documental es un procedimiento científico, un proceso sistemático de indagación, recolección, organización, análisis e interpretación de información o datos en torno a un determinado tema”. Refiriéndose también la investigación de tipo documental por el hecho de disponer esencialmente de documentos que son el resultado de otras investigaciones, lo cual representa la base teórica del área objeto en estudio, el conocimiento de este se construye a partir de su lectura, análisis, reflexión e interpretación de dichos documentos.

Existen una serie de pasos para desarrollar la investigación documental que compone el proyecto factible, lo cual se mantendrán presente para el desarrollo del proyecto y hacer de ésta un proceso más eficiente. Expuesto esta aclaratoria se consideraron los siguientes pasos: Selección y delimitación del tema, acopio de información o de fuentes de información, Análisis de los datos, Redacción de la investigación y presentación final [32].

Aplicándose los lineamientos básicos propuestos para la investigación y a manera de poder organizar y plantear tanto los objetivos, problemas y soluciones se jerarquizara dos tipos de estudios, iniciándose con el estudio explorativo mediante el cual se recaudó la información necesaria para abordar el tema, aprender, examinar y establecer las variables de importancia para

ser consideradas en los resultados, y por ultimo un estudio comparativo ya que al obtener la imagen del rostro en estudio se hará una comparación de los resultados de las características de interés arrojas mediante la aplicación de técnicas de segmentación en Matlab.

3.2 TÉCNICA DE RECOLECCIÓN DE DATOS

Se utilizan una variedad de métodos a fin de recopilar los datos sobre una situación existente como la toma de base de datos (imágenes), revisión de documental y observación. Cada uno tiene ventajas y desventajas. Generalmente, se utilizan dos o tres para complementar el trabajo de cada una y ayudar a asegurar una investigación completa.

- **Base de datos:** Las imágenes están tomadas en un ambiente controlado, todas las personas se encuentran a una distancia fija de la cámara, las variaciones en la iluminación son muy pequeñas y todas las imágenes tienen el mismo tamaño. En cada fotografía aparece una sola cara que ocupa la mayor parte de la imagen y está centrada en ella, además las variaciones entre las imágenes de un mismo individuo son pequeñas, se reducen a variaciones en la expresión del rostro.

- **Revisión documental:** se refiere a los libros de texto, artículos, revistas científicas y a los antecedentes que serán de base para el análisis en el trabajo especial de grado. Lo primero es la preparación de la revisión documental definiendo cual es el material de interés, verificando que verdaderamente contenga material de interés provechoso que pueda ser utilizado para el desarrollo del trabajo especial de grado; luego se procede a una clasificación del material con el objeto de ubicarlo en una de las partes del proyecto con la finalidad de crear un esquema, después viene la eliminación de la información que no es de utilidad dejando solo lo que es realmente compatible para partir de datos concretos para así rechazar o confirmar hipótesis [32].

3.3 DISEÑO DE LA INVESTIGACIÓN

El sistema general de la investigación se iniciará con la búsqueda bibliográfica y antecedentes relacionados con el tema, posteriormente se hará la toma de imágenes, así como también a la revisión de comandos disponibles en la Toolbox de procesamientos de imágenes en matlab para la segmentación, luego se procederá a construir un algoritmo que permita extraer las características faciales de interés, lo que conduce finalmente a la etapa de comparación para aplicar una evaluación o emitir un criterio sobre la data acerca del rostro. A continuación se detallará lo antes expuesto en sus correspondientes fases**.**

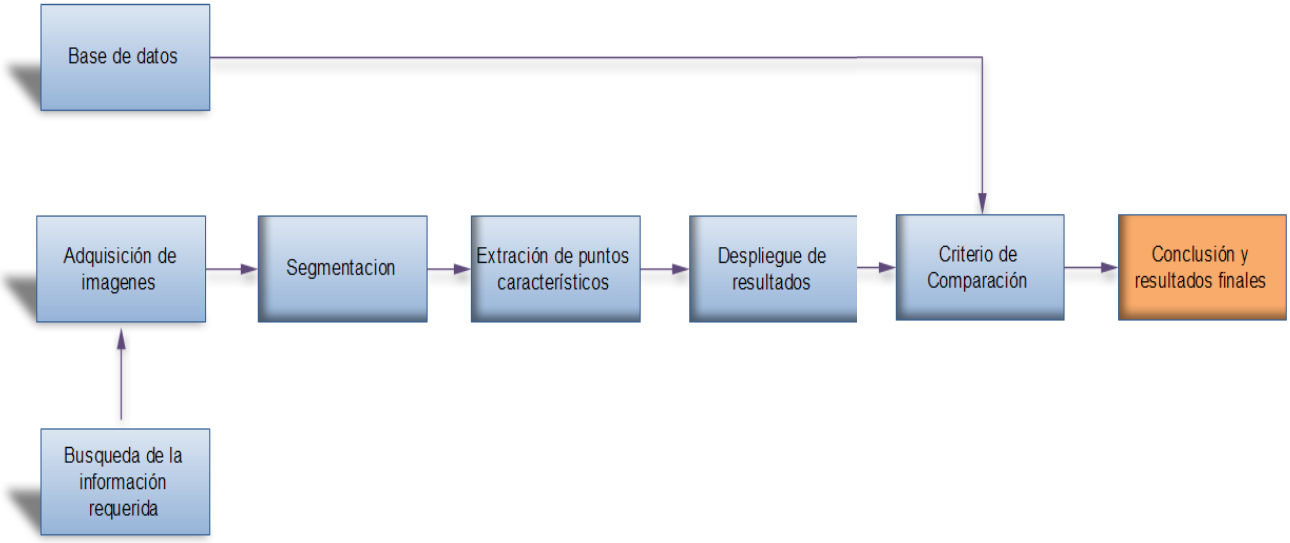


Figura 3.1 Diagrama en bloques del sistema de reconocimiento de rostro [6] [8] [11]

3.4 ETAPAS DE LA INVESTIGACIÓN

Esta investigación se realizará cumpliendo paso a paso los objetivos a través de las siguientes etapas:

Fase I: Recopilación y documentación de la visión artificial en los sistemas de reconocimiento de rostro

Se realizara el estudio referencial y documental para recaudar los fundamentos teóricos basados en bibliografía especializada, provenientes de los diferentes medio de información como son: trabajos de pre-grado, artículos científicos, informes técnicos, sitios web, entre otras investigaciones. Así como también se estudiaran los fundamentos básicos concernientes a la visión artificial y al reconocimiento de rostro.

Fase II: Determinación de las técnicas de Segmentación para el reconocimiento

En esta segunda fase se van a experimentar con las técnicas clásicas de segmentación de imágenes empleando “Image Procesing Toolbox” de Matlab. Durante esta fase, se estudiarán las diferentes metodologías que permitirán extraer información subyacente de una imagen para su posterior uso.

Fase III: Aplicación Determinar los requerimientos del software.

En esta fase se desarrollará una aplicación de software en Matlab a partir de las funciones necesarias que proporcionan las técnicas de segmentación. Esta aplicación tendrá como producto final el uso de las técnicas de umbralización combinadas con el proceso morfológico de los objetos ya segmentados. Todo esto trayendo consigo el desarrollo de una interfaz gráfica que permitirá evaluar la eficiencia de la aplicación.

Fase IV: Definición del escenario

Debido a que se está implementando un dispositivo basado en imágenes (Webcam), en esta fase se generó la necesidad de diseñar un ambiente apropiado para lograr el funcionamiento

del sistema; es decir, se realizara la base donde se apoyara el rostro y la cámara, además se elaborara un sistema de iluminación que cumpla las necesidades requeridas.

Fase V: Identificación del equipo

En esta fase se indagarán acerca de las características del equipo a ser implementado para el módulo de reconocimiento de rostro. Para lograr dicho objetivo se recurrirá a la revisión de manuales técnicos y especificaciones de los equipos, como también la disponibilidad del equipo a nivel nacional.

Fase VI: Interfaz gráfica y despliegue de resultados

En esta fase se determinarán las características que debe tener la interfaz para que cumpla con los requerimientos del reconocimiento para luego pasar a la construcción de la misma. Esta interfaz gráfica, nos permitirá visualizar los resultados obtenidos y su respectivo funcionamiento. Estos despliegues de resultados serán la base para las conclusiones de la investigación.

CAPÍTULO IV. MARCO OPERACIONAL

4.1 DESARROLLO DEL SISTEMA DE RECONOCIMIENTO

El desarrollo del sistema de reconocimiento se basa en la extracción del rostro en la imagen, mediante una serie de puntos que lo caracterizan. Para poder llevar a cabo esta extracción se describe a continuación el diagrama de flujo del sistema (figura 4.1); el cual, establece la pauta para desarrollar el presente capítulo de una forma muy minuciosa, y también representa la estructura básica para realizar cualquier reconocimiento de rostro.

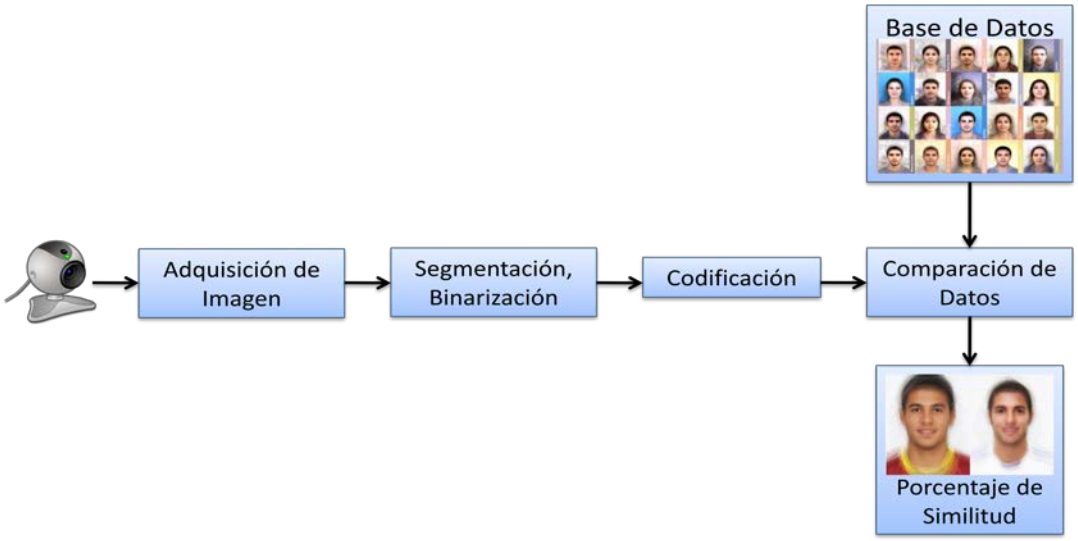


Figura 4.1 Diagrama de bloque general del reconocimiento de rostro.

- ✓ **Adquisición de la Imagen**: En este módulo se captura la imagen del rostro de la persona a ser reconocida por el sistema.
- ✓ **Segmentación y Binarización**: En esta fase se procesa la imagen capturada (ella es convertida a escala de grises, para luego segmentarla y binarizarla).
- ✓ **Codificación**: En esta etapa se realiza el código, con la finalidad de extraer los puntos característicos esenciales del rostro.

- ✓ **Comparación**: En esta fase se realiza una comparación entre la muestra y una base de datos que contiene características especificas de varios rostros, con el fin de identificar a la persona (porcentaje de similitud).

Es importante resaltar como se ha visto en el diagrama de bloque general, que el programa se divide en cuatro bloques principales, cada uno de los cuales consta de una o varias funciones. En este apartado se va a proceder a una descripción de cada uno de estos bloques y funciones que constituyen el programa.

4.2 ADQUISICIÓN DE LA IMAGEN

Para realizar el reconocimiento de rostro es necesario tener una imagen a la cual se le desea hacer el estudio o reconocimiento. En este caso, las imágenes fueron tomadas abriendo un canal de video en Matlab, y se capturaron desde una cámara web convencional. Además, se proporcionó un ambiente controlado; es decir, todas las personas se encuentran a una distancia fija de la cámara, con nivel de iluminación similar, en cada fotografía aparece una solo cara que ocupa toda la imagen y está centrada en ella. Para abrir el canal se implementó la siguiente función de Matlab.

Tabla 4.1 Función de Matlab utilizada para la adquisición de la imagen.

Nombre	Descripción	Entrada	Salida
videoinput	Representa la conexión entre Matlab y un objeto de adquisición de imagen	Parámetros del dispositivo: *Nombre del adaptador, dispositivo*	Canal de video

Es necesario resaltar, que para determinar la cantidad de adaptadores disponibles en el sistema se usa la siguiente función de Matlab imaqhwinfo.

Luego de la adquisición de Matlab se procesa la imagen, originalmente capturada en formato RGB.

4.3 PREPROCESAMIENTO DE LAS IMÁGENES

Las técnicas de preprocesado pretenden es simplificar o cambiar la representación de una imagen en otra más significativa y más fácil de analizar, entre los grandes grupos en la parte de preprocesado se encuentra la etapa de segmentación.

En este proyecto se expondrá las siguientes técnicas de preprocesado que han sido utilizadas:

4.3.1 COMPENSACIÓN DE ILUMINACIÓN

Esta es llevada a cabo debido al hecho de que la apariencia del color depende de las condiciones luminosas, y con este método se trata de normalizar este efecto. El algoritmo de compensación de iluminación que se utiliza es el propuesto en el articulo [Hsu et al. 2001]. Como se ha dicho esta técnica usa un “blanco de referencia” para normalizar la apariencia del color.

Según el artículo, se consideran pixeles que estén en el 5% del valor superior de la luminancia como “blanco de referencia”, solo si el numero de estos pixeles es suficientemente grande (>100). Las componentes R, G y B de la imagen se ajustan de forma que el valor medio de esos pixeles del “blanco de referencia” es escalado linealmente a 255. La imagen no se cambia si no se detecta un número suficiente de pixeles de “blanco de referencia”.

En este proyecto, igual que en articulo [Hsu et al. 2001], se toman como pixeles de "blanco de referencia" aquellos que estén en el 5% del valor superior de la luminancia, cuyo rango es [16,235].

Para la implementación de la compensación de iluminación se utilizan funciones básicas de Matlab. Cabe destacar el uso de la siguiente función:

Tabla 4.2 Función de Matlab utilizada en la compensación de iluminación.

Nombre	Descripción	Entrada	Salida
rgb2ycbcr	Convierte la imagen RGB en la imagen equivalente en el espacio de color YCbCr	Imagen RGB: matriz [M,N,3]	Imagen YCbCr: matriz [M,N,3]

4.3.2 SEGMENTACIÓN DE LA IMAGEN

En el proceso de segmentación se busca dividir la imagen en las partes u objetos que la forman. El nivel al que se realiza esta subdivisión depende de la aplicación en particular, es decir, en nuestro caso de reconocimiento de rostros la segmentación terminará cuando se hayan detectado todas las regiones de interés. En general, la segmentación es una de las tareas más complicadas dentro del procesado de la imagen.

Por tanto, la segmentación va a dar lugar en última instancia al éxito o fallo del análisis llevado a cabo en el tratamiento de la imagen. En la mayor parte de los casos, una buena segmentación dará lugar a una solución correcta, por lo que, se ha puesto en esta investigación todo el esfuerzo posible en la etapa de segmentación. Los algoritmos de segmentación de imagen generalmente se basan en la propiedad básica de las técnicas de umbrales, crecimiento de regiones, y técnicas de división y fusión.

Debido a todo esto, en los siguientes puntos del presente capítulo se describirá de forma muy rigurosa las técnicas de segmentación aplicada en esta investigación, el cual conduce al éxito de la misma.

4.3.2.1 BINARIZACIÓN (THRESHOLDING)

El objetivo de transformar la imagen en otra en blanco y negro es encontrar los rasgos que caracterizan una cara, como son los ojos. Para binarizar la imagen se implemento dos métodos, el primer método toma como valor de umbral el determinado a partir de la instrucción de Matlab graythresh, en cual proporciona un nivel que se puede utilizar para convertir una imagen en escala de grises, a una imagen binaria con im2bw y el segundo método se implemento para comprobar de forma experimental el valor del umbral mas optimo a partir del calculado por la instrucción graythresh.

El motivo de implementar este segundo método se debe a que el primer método no es capaz de realizar una binarización aceptable para la extracción de puntos característicos en una iluminación de espacio controlado.

4.3.2.1.1 Método 1

Este método se aplica para conseguir una imagen binaria (blanco y negro) a partir de un valor de umbral proporcionado por la graythresh. Como se observa en la Figura 4.2 que el valor umbral calculado con el primer método no es eficiente, Si se umbraliza la imagen con este valor, el resultado es tal que no permite identificar con certeza los ojos en la imagen y por lo tanto la extracción de la regiones de interés serian erróneas.

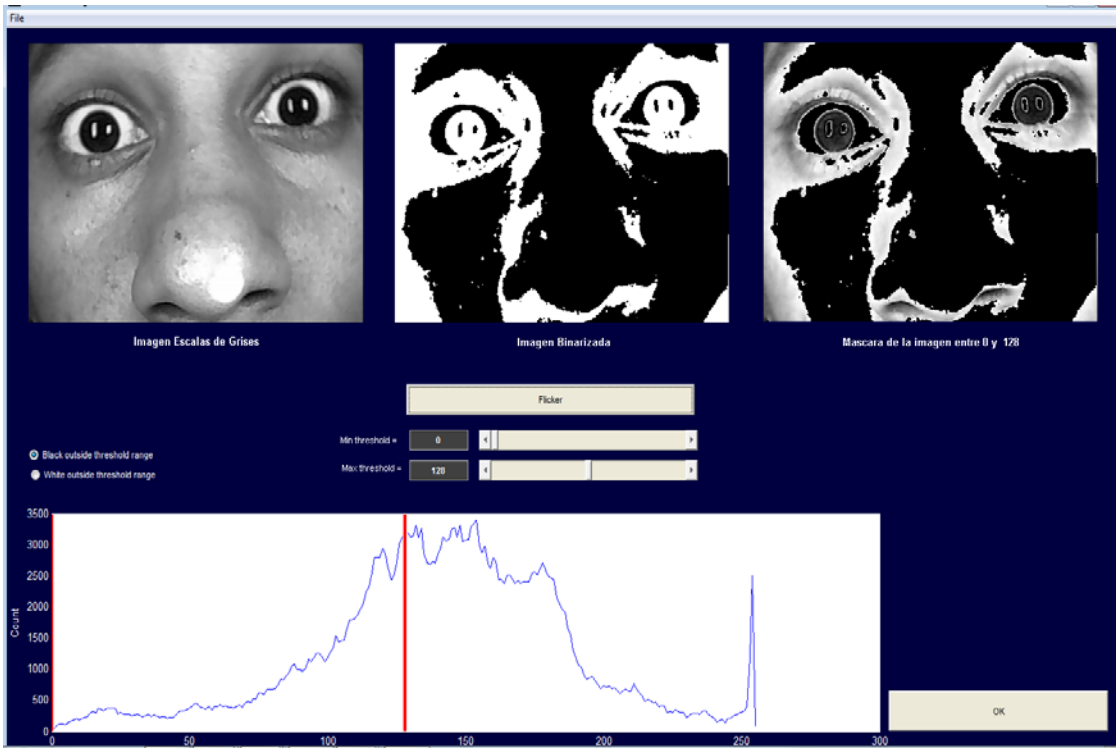


Figura 4.2. Imagen binarizada a partir del valor de umbral proporcionado por la instrucción graythresh de matlab.

4.3.2.1.2 Método 2

Este segundo método es implementado debido a que el primer método fue imposible de resaltar de forma concreta la localización de los ojos. Este se basa esencialmente en hacer un barrido a la largo del histograma de la imagen con valores de umbral por debajo del establecido con la instrucción graythresh, lo que comprueba de forma experimental el valor del umbral mas optimo para la ubicación de los ojos. En la figura 4.3 se muestra el valor de umbral determinado experimentalmente con resultados bastante satisfactorios.

Es importante resaltar que este nivel de umbralización se ha probado con un gran número de rostros en un espacio de iluminación controlado lo que demuestra la efectividad del mismo.

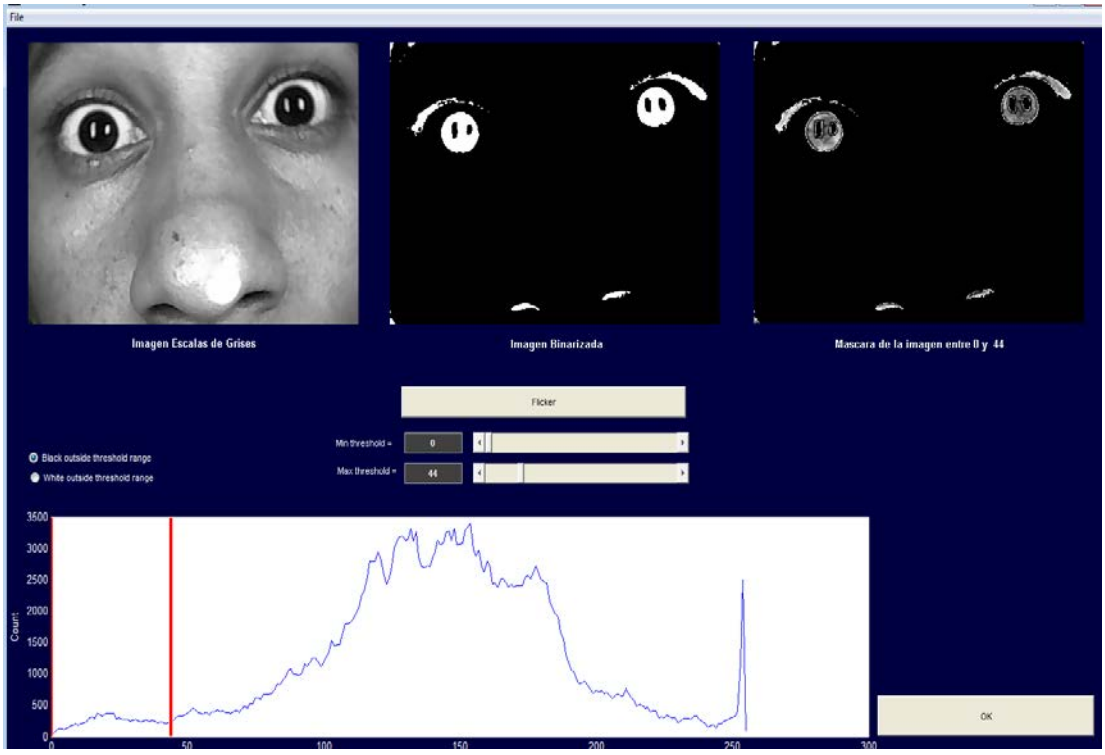


Figura 4.3 Imagen binarizada a partir del valor de umbral determinado de forma experimental a partir de graythresh de matlab.

Tabla 4.3 Función de Matlab utilizada en la binarización de la imagen.

Nombre	Descripción	Entrada	Salida
graythresh	Determinar el valor de umbral a una imagen, para así binarizarla	Imagen escala de grises	Valor de Umbral

4.3.2.2 FILTRADO MORFOLÓGICO

Una vez que se tiene la imagen binaria se procede a eliminar y agrupar regiones para crear las zonas que realmente serán de interés. Esto se realiza con operadores basados en morfología

binaria. Este tipo de operadores ya se vieron en capítulo 2 (marco teórico) del mismo. En primer lugar se realiza un cierre de la imagen. El cierre consiste en una dilatación seguida de una erosión. La dilatación es comúnmente conocida como relleno, expansión o crecimiento. Puede ser usado para rellenar huecos de tamaño igual o menor que el elemento estructurante con la que se opera la dilatación. La erosión es lo opuesto a la dilatación; Los efectos son de encogimiento, contracción, o reducción. Puede ser utilizado para eliminar islas menores en tamaño que el elemento estructurante.

Esta función de cierre se utiliza para rellenar pequeños huecos que haya quedado en zonas que tienen probabilidad de ser ojos. Posteriormente se finaliza realizando una erosión a la imagen.

Es importante resaltar la elección del elemento estructurante elegido para las operaciones morfológicas. En cuanto a su forma, se ha elegido un disco. En cuanto al tamaño del elemento estructurante seleccionado se ha comprobado experimentalmente un valor apropiado, para ello se realizaron pruebas con distintos tamaños de radio del disco del elemento estructurante, dicho valor se mantendrá presente para cualquier operación morfológica implementada en la presente investigación.

Tabla 4.4 Funciones de Matlab utilizada para la extracción de zonas de interés.

Nombre	Descripción	Entrada	Salida
imdilate	Dilata la imagen de entrada usando el elemento estructurante SE	Imagen binaria o en escala de grises, elemento estructurante SE	Imagen dilatada
imerode	Realiza la erosión de la imagen de entrada con el elemento estructurante SE	Imagen binaria o en escala de grises, elemento estructurante SE	Imagen erosionada

imclose	Ejecuta un cierre morfológico sobre una imagen en escala de gris o binaria	Imagen y elemento estructurante con el que se realiza el cierre	Imagen con el cierre

4.3.2.3 SELECCIÓN DE PUNTOS CARACTERÍSTICOS

Con las regiones de interés bien resaltadas se pasa a la parte de selección de los puntos característicos. Esta selección se va a realizar en función de unas características mínimas que deben tener las regiones a ser consideradas como ojos, dichas características están relacionadas con las dimensiones y la forma de la región, teniendo en cuenta que lo que se desea extraer son los ojos.

En este punto se implemento la instrucción de Matlab regionprops (ver tabla 4.5), el cual se encarga de determinar el centro de las todas las regiones de interés presente en la imagen.

Tabla 4.5 Función de Matlab utilizada para la selección de puntos característicos.

Nombre	Descripción	Entrada	Salida
regionprops	Mide un conjunto de propiedades para cada región etiquetada en la matriz.	Matriz etiquetada y lista de propiedades	Vector con las medidas realizadas en cada región

4.4 DESARROLLO DE ALGORITMOS Y FUNCIONES MEDIANTE LAS TÉCNICAS CLÁSICAS DE SEGMENTACIÓN PARA LA EXTRACCION DE PUNTOS CARACTERISTICOS (OJOS EN UNA IMAGEN)

Para la extracción de puntos característicos (ojos en una imagen) se desarrollo un algoritmo donde se implementan las técnicas clásicas de segmentación, dicho algoritmo hace el llamado a una función M-file, el cual permite el uso de estas técnicas, esta función es compensación_luz.m, Este código está disponible en el apéndice. Las especificaciones de entrada y salida de esta función son las siguientes:

img_compensada1= compensación_luz(RGB)

RGB: Una imagen en modelo RGB el cual contiene tres planos de imágenes independientes (rojo, verde y azul).

img_compensada1: Las componentes R, G y B de la imagen se ajustan de forma que el valor medio de esos pixeles del “blanco de referencia” es escalado linealmente a 255.

4.4.1 EJEMPLO DE EXTRACCION DE PUNTOS CARACTERISTICOS (OJOS)

Para la extracción de estos puntos característicos se elaboró el siguiente algoritmo en Matlab, el cual se visualiza la implementación de esta función y las técnicas de segmentación usada en ésta investigación:

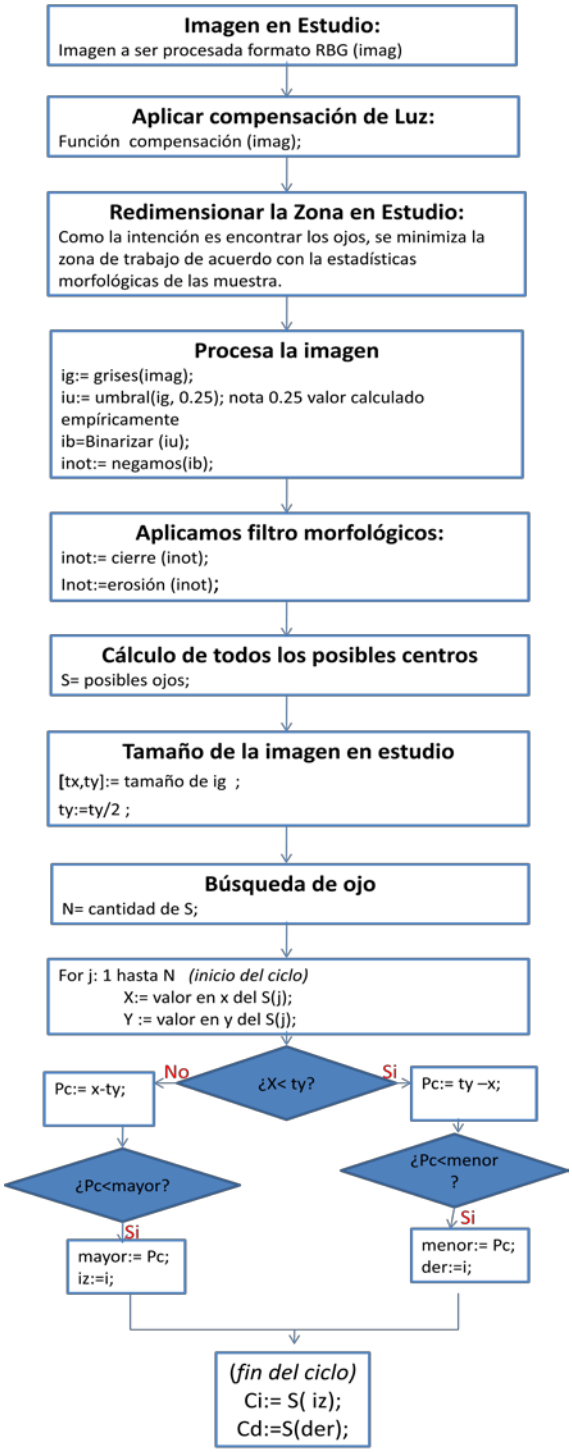


Figura 4.4 Diagrama de flujo de extracción de los ojos

Donde, las variables usadas en el diagrama de flujo de extracción de ojos se encuentran descriptas en la tabla 4.6:

Tabla 4.6 Especificaciones del diagrama de flujo extracción de ojos.

Variable	Significado de la variable
Imag	Imagen en formato RGB
Ig	Imagen a escala de grises
Iu	Imagen umbralizada
Ib	Imagen binarizada
Inot	Imagen negada que luego se le aplicaran los filtros
S	Posibles ojos
J	Contador del ciclo
Tx	Cantidad de filas de la imagen
Ty	Cantidad de columnas de la imagen
N	Cantidad de posibles ojos
X	Coordenada en X del posible ojo
Y	Coordenada en Y del posible ojo
Pc	Posible centro
Ci	Centro del ojo izquierdo; este posee dos coordenadas en Xci, Yci
Cd	Centro del ojo derecho; este posee dos coordenadas en Xcd, Ycd

En la figura 4.5 se muestra la Ejemplo.jpg a la cual se le implementó las técnicas de segmentación.

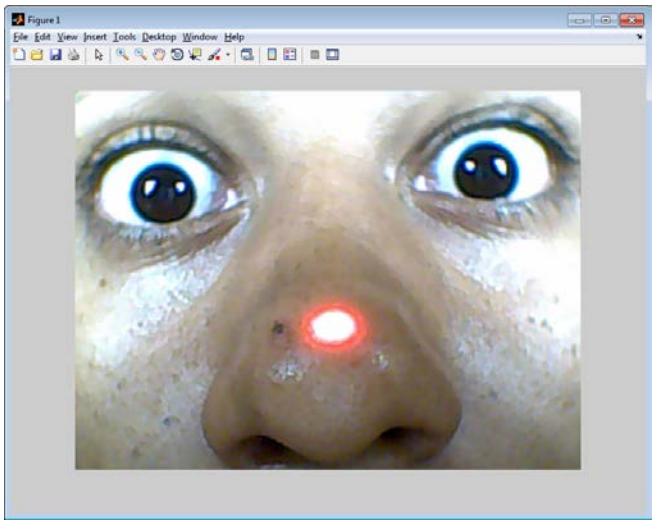


Figura 4.5 Imagen a la cual se le aplica la segmentación.

En la figura 4.6 se muestra la imagen una vez aplicadas las técnicas de segmentación.

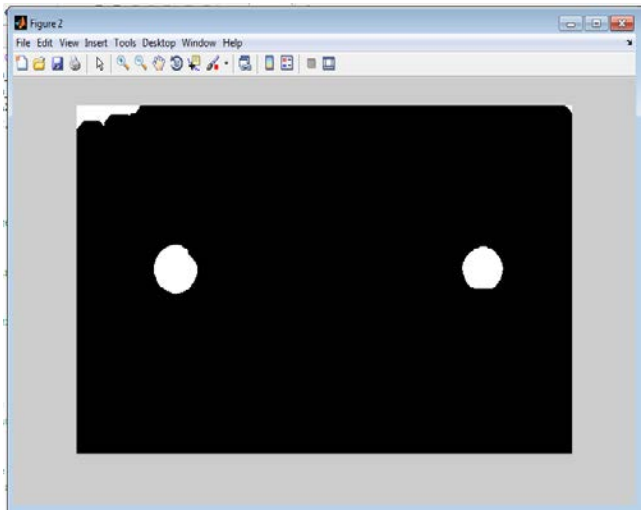


Figura 4.6 Resultado una vez aplicada la segmentación.

En la figura 4.7 se muestra la obtención del centro de las zonas resaltadas como ojo a partir de la instrucción regionprops.

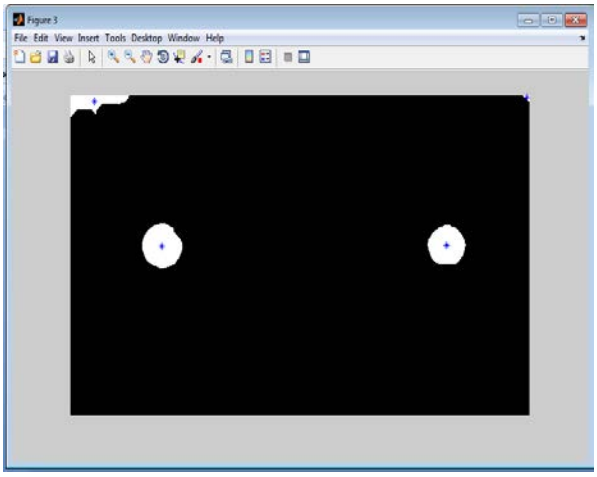


Figura 4.7 Centro de los posibles ojos.

En la figura 4.8 se muestra la extracción de los puntos característicos candidatos a ser ojos.

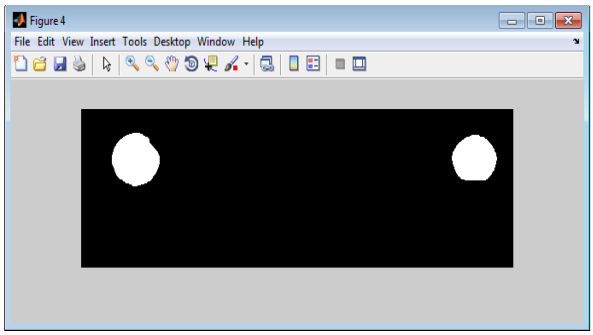


Figura 4.8 Extracción de puntos característicos.

4.5 DESARROLLO DE ALGORITMO MEDIANTE LAS TÉCNICAS CLÁSICAS DE SEGMENTACIÓN PARA LA EXTRACCION DEL PUNTO DE REFERENCIA (LASER)

Para la extracción del punto de referencia (laser), se procedió de forma similar como en el caso de los ojos, la diferencia radica al ser el punto definido por el laser blanco, lo que conduce en buscar en la imagen el punto más blanco, para ello se implemento la instrucción im2bw. Para el uso de esta instrucción se busco de forma experimental el nivel de umbral, el cual definirá de forma concreta la región en interés.

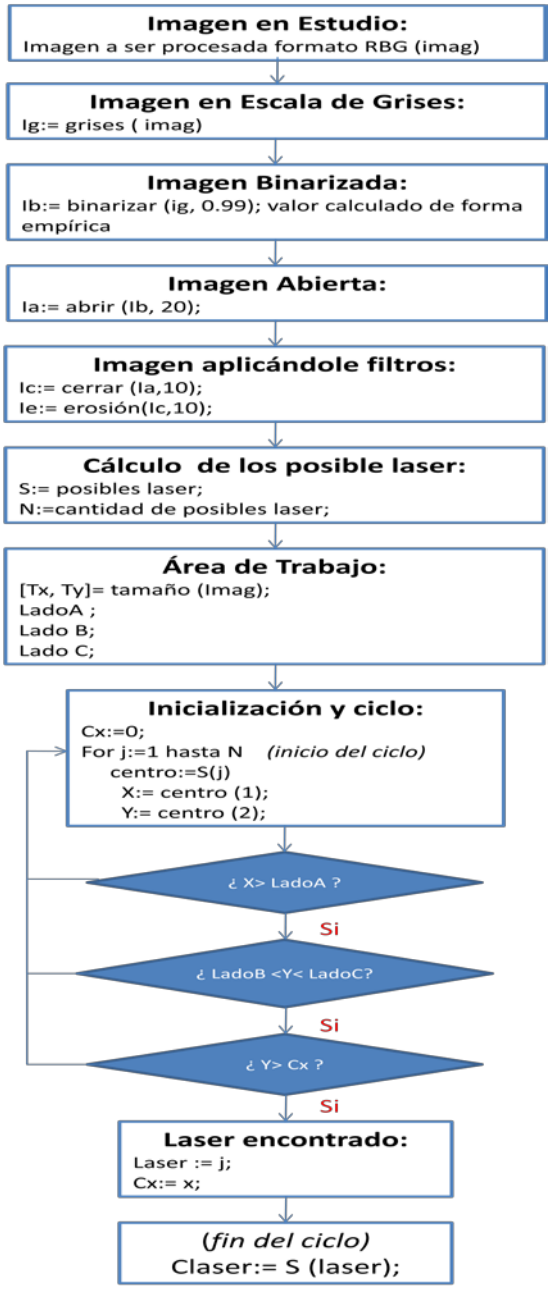


Figura 4.9 Diagrama de flujo para la extracción del punto de referencia

A continuación se presenta la tabla 4.7 la cual contiene una breve descripción de las variables utilizadas en el diagrama de flujo para la extracción del laser:

Tabla 4.7 Descripción de las variables del diagrama de flujo extracción del laser.

Variable	Significado de la variable
Imag	Imagen en formato RGB
Ig	Imagen a escala de grises
Ib	Imagen binarizada
Ia	Abrir los zonas de ruidos para así descartar
Ic	Imagen cerrada; uso de filtro morfológico
Ie	Imagen erosionada; uso filtro morfológico
S	Posibles laser
N	Cantidad de posibles laser
LadoA	Zona de trabajo, en la recta Y= LadoA
LadoB	Zona de trabajo, en la recta X= LadoB
LadoC	Zona de trabajo, en la recta X= LadoC
Tx	Cantidad de filas de la imagen
Ty	Cantidad de columnas de la imagen
Cx	Coordenada X del laser encontrada en el ciclo
J	Contador del ciclo
X	Coordenada en X del posible laser
Y	Coordenada en Y del posible laser
Laser	Laser encontrado en la zona de trabajo
Claser	Centro del laser encontrado

4.5.1 EJEMPLO DE EXTRANCCION DEL PUNTO DE REFERENCIA (LASER)

Para la extracción de este punto de referencia se elaboró el siguiente algoritmo en matlab, el cual se visualiza la implementación de las técnicas de segmentación usada en esta investigación:

```
RGB = imread('C:\Users\USUARIO\Desktop\Ejemplo.jpg');
I = rgb2gray(RGB);
figure
imshow(I);
threshold =0.99
j=im2bw(I,threshold);
bw = bwareaopen(j,20);
se = strel('disk',10);
bw = imclose(bw,se);
bw = imerode(bw,se);

s  = regionprops(bw, 'centroid');%centro de laser
centroids = cat(1, s.Centroid);
figure;
imshow(I)
hold on
plot(centroids(:,1), centroids(:,2), 'b*')
hold off
n=length(s);
mayor=0;
[tx,ty]= size (I);
a=0.85*tx/2;
b=4.5*ty/16;
c=9.5*ty/16;
for i=1:n
 centro = s(i).Centroid;
 x = centro(1);
 y = centro(2);
 if x>a
    if y>b && y<c
       if y>mayor
          laser=i;
          mayor=x;
       end
    end
 end
end
```

En la figura 4.10 se muestra la imagen una vez binarizada al nivel del umbral definido experimentalmente.

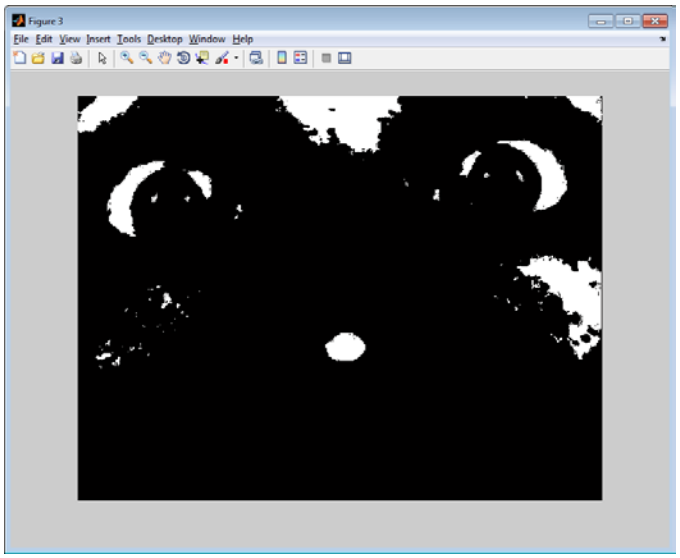


Figura 4.10 Extracción de puntos característicos.

En la figura 4.11 se muestra la imagen original donde se seleccionan las regiones de interés a partir de la binarización.

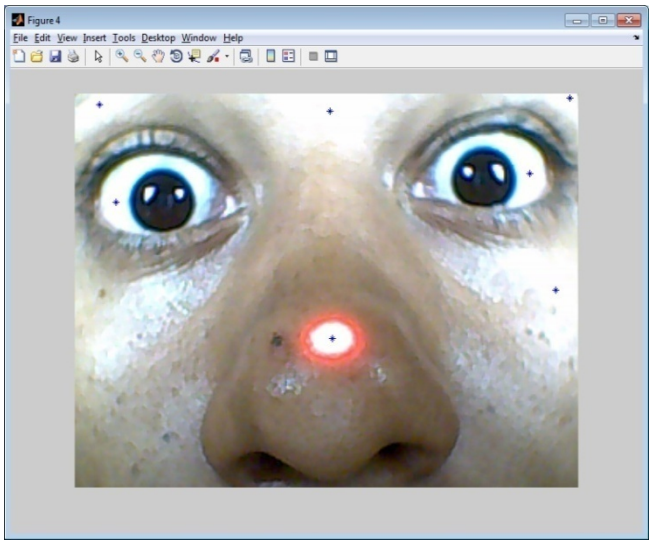


Figura 4.11 Extracción de puntos característicos.

En la figura 4.12 se muestra la extracción del punto de referencia (laser).

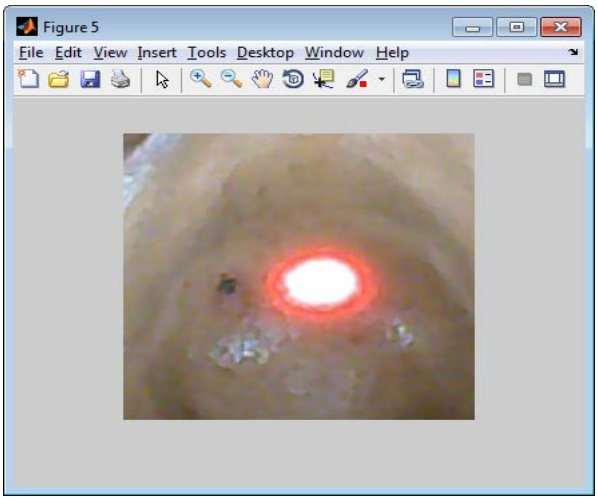


Figura 4.12 Extracción de puntos característicos.

4.6 DESARROLLO DE ALGORITMOS Y FUNCIONES PARA EL MANEJO DE LA BASE DE DATOS DEL SISTEMA

Luego de tener los puntos característicos de un rostro, para proceder a reconocerlo es necesario comprobar su existencia dentro de una base de datos, la cual contiene los datos del los usuarios registrado en el sistema; en este caso particular, se creó una base datos de veinte (20) personas, aunque el sistema cuenta una plataforma robusta; es decir, se puede añadir la cantidad de usuarios que el administrador precise. Para el manejo (llenado o borrado) de la misma es necesario poseer la contraseña o clave de acceso. En esta base se encuentran los datos (nombre, imagen, distancia del ojo derecho al laser, distancia ojo izquierdo al laser y distancias entre los ojos) de las personas inscriptas en el sistema. Esta base de datos está compuesta por un archivo y una carpeta de imágenes en formato RGB.

4.6.1 LLENADO DE LA BASE DE DATOS

Para añadir una persona al sistema el administrador tendrá que dar funcionamiento al programa; luego el usuario a ser registrado suministrará un nombre (por ejemplo: nombre y apellido), éste deberá contener un máximo de 25 caracteres, de los cuales será excluido el punto

(.); además es importante recordar, que para la capturar de la imagen se proporciono un ambiente controlado.

La fotografía capturada será procesada para extraer de ella los puntos característicos (explicados anteriormente) y con estos calcular las distancias entre ellos. Las cuales son almacenadas en un archivo junto con el nombre de ese usuario; por otra parte las imágenes serán almacenadas en una carpeta. Para mantener un orden y una correspondencia entre el usuario y su imagen, se decidió renombrar la imagen con el nombre suministrado por el usuario, el cual se muestra en la figura 4.13.

Figura 4.13 Imágenes almacenadas en la carpeta de la base de datos.

A continuación se presenta algunas funciones usadas en Matlab para almacenar la imagen como se muestra en la figura anterior:

Tabla 4.8 Funciones en Matlab usadas para almacenar imágenes con un nombre especifico.

Nombre	Descripción	Entrada	Salida
Strcat	Une la dirección de almacenamiento con el nuevo nombre (nombre y extensión)	Dirección/ubicación y nombre del usuario	Nombre de la imagen con ubicación

Además, en la base de datos no se encuentran dos registros con el mismo Nombre, ya que el sistema se encarga de verificar esto. Para que esto fuese posible, los diseñadores procedieron de la siguiente manera: crearon una matriz numérica que guarda los nombre de las persona registrada, donde la cantidad de filas son las persona registrada y numero de columna es un valor fijo, veintiocho (de los cuales los primeros veinticinco son para el nombre del usuario y el resto para guardar las distancias). Para crear esta matriz de número, convirtieron la cadena de caracteres (el nombre) en un vector de números, para realizar esta conversión en Matlab es muy sencillo debido a la función *uint8 ()*, la cual convierte una cadena de caracteres a numero según la tabla ASCII. A continuación se muestra en la figura 4.14 el diagrama de flujo para explicar la verificación; es decir, que no existan dos registros iguales.

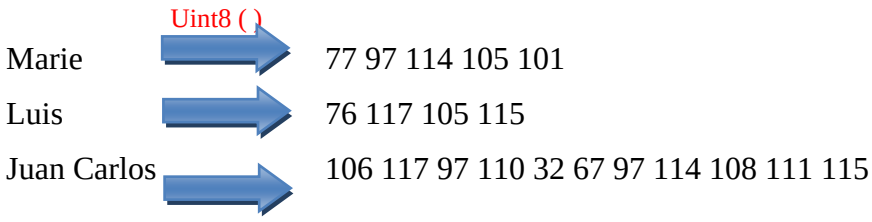
Figura 4.14 Diagrama de flujo de verificación de registro

Donde, las variables usadas en este diagrama de flujo se encuentran especificadas en la tabla 4.9

Tabla 4.9 Descripción de las variables usadas en el diagrama de verificación de registro.

Variable	Significado de la variable
M	Matriz Numérica
Filam	Cantidad de filas de la matriz M, contiene el número de usuarios inscriptos en el sistema
Colm	Cantidad de columnas de la matriz M, contiene los datos del usuario
I, J	Contadores del Ciclo.
Contador	Contador; cantidad de ceros que posee una fila.
Cont	Contador; cantidad de coincidencias entre el los datos del vector y los datos de la columna cuando está en una determinada fila.
Tr	Tamaño real de la columna matriz; valores distintos de ceros contenidos en una fila.

A continuación, se presenta un ejemplo sobre el almacenamiento del nombre de usuario en la matriz numérica:



Matriz numérica:

77	97	114	105	101	0	0	0	0	0	0
76	117	105	115	0	0	0	0	0	0	0
106	117	97	110	32	67	97	114	108	111	115

Tr para la fila uno=5; ya que contador= 6 y colm=11.

Tr para la fila dos=4; ya que contador=7 y colm=11.

Tr para la fila tres=11; ya que contador=0 y colm=11.

4.6.2 BORRADO DE LA BASE DE DATOS

Para eliminar un usuario del sistema, es necesario poseer la clave de acceso al sistema y el nombre con el cual se encuentra registrado, elaboraron una función para eliminar la imagen de dicho usuario, el nombre fue borrado usando una codificación similar a la mostrada en el diagrama de verificación de usuario (ver figura 4.15) la diferencia primordial radica en el hecho, que al final no se emite un mensaje sino que borran esa fila; Se pueden destacar, las siguientes instrucciones usadas en Matlab para borrar una imagen.

Tabla 4.10 Funciones en Matlab usadas para borrar imágenes con un nombre especifico.

Nombre	Descripción	Entrada	Salida
strtok	separa el nombre con el cual fue registrado el usuario de la extensión de la imagen	Nombre del registro y Punto (.)	Nombre y extensión
strcat	Une la dirección de almacenamiento con el nuevo nombre (nombre y extensión)	Dirección/ubicación y nombre del usuario	Nombre de la imagen con ubicación

4.6.2.1 EJEMPLO DEL BORRADO DE UNA IMAGEN CON UN NOMBRE EN PARTICULAR

A continuación se muestra la función implementada para el borrado de una imagen almacenada en la carpeta de la base de datos del sistema:

```
Function  borrar_fotos_almacenadas (w)
 files=dir('C:usuario \foto ');
   for i=3:length(files)
     n=files(i).name;
     [nombre,extension]=strtok (n,'.');
     if length(nombre)==length(w)
      con=0;
     for j=1:length(w)
        if nombre(j)== w(j)
            con=con+1;
        end
    end;
    if con== length(w)
         nombre_completo= strcat('C:usuario \foto\',n);
         delete(nombre_completo);
     return;
    end;
     end;
   end;
```

4.7 CONSTRUCCIÓN DE LA INTERFAZ GRÁFICA

Aprovechando la herramienta GUIDE de Matlab se construyó la interfaz gráfica de usuario (GUIs en ingles) con el fin de facilitar el manejo del programa. Para la ejecución de la aplicación de reconocimiento de rostro 2011, estando el cursor en la ventana de comandos de Matlab, se debe escribir el siguiente comando:

```
>> Presentacion
```

(Sin importar realmente la utilización de letras mayúsculas o minúsculas) con lo que se desplegará la primera ventana de la interfaz gráfica de usuario. Antes de explicar la estructura de cada ventana, se mostrara figura 4.15 el diagrama de bloques de las ventanas del programa con el fin de señalar el nivel jerarquía en la ejecución de las ventanas y el nombre de cada una de ellas.

- ❖ Presentación
  - ✓ Menú Principal.
    - Reconocimiento.
      - ♦ Configurar.
    - Nuevo Registro.
      - ♦ Configurar
    - Consultar.
      - ♦ Base de Datos.
    - Borrar Registro
  - ✓ Menú Ayuda.

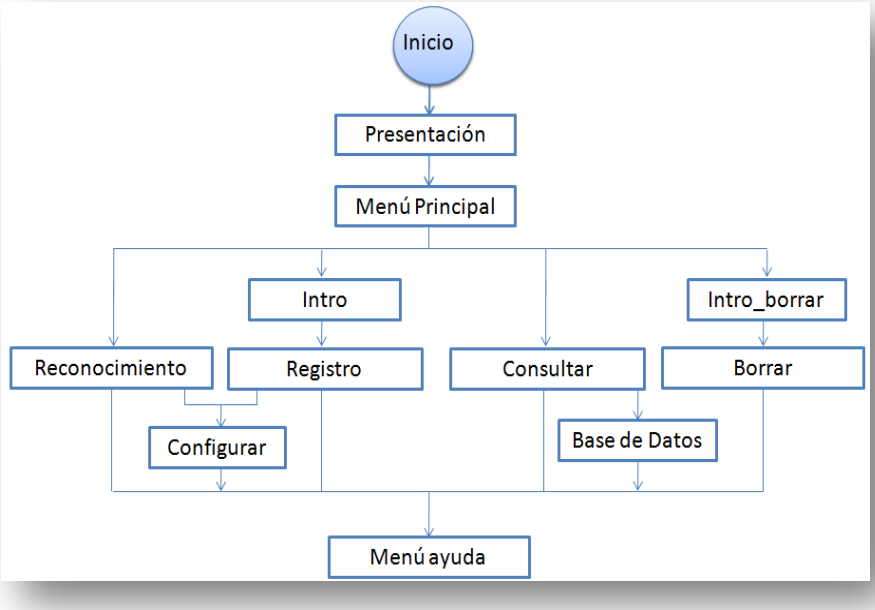


Figura 4.15 Diagrama de bloque de las ventanas del programa.

Como fue mencionado anteriormente, la primera ventana del programa es presentación, ésta contiene una breve descripción del software y su finalidad; además posee el titulo de la aplicación (Reconocimiento de Rostro 2011), nombre de los autores y el nombre del tutor; por lo tanto, es una ventana solamente informativa (ver figura 4.16). Ella también posee una opción (continuar) con el cual se ejecuta la siguiente ventana.

Figura 4.16 Ventana de Presentación.

A continuación se presenta la siguiente ventana (Menú Principal) está posee cinco botones; es básicamente un panel de tarea donde el usuario o administrador podrá elegir la opción que desee de acuerdo con lo que quiera realizar. Las primeras cuatro opciones que la compone son: Reconocimiento, Nuevo Registro, Consultar Registro, Borrar Registro y la última es Salir, cuya finalidad es salir del programa. La ventana menú principal se muestra en la figura 4.17.

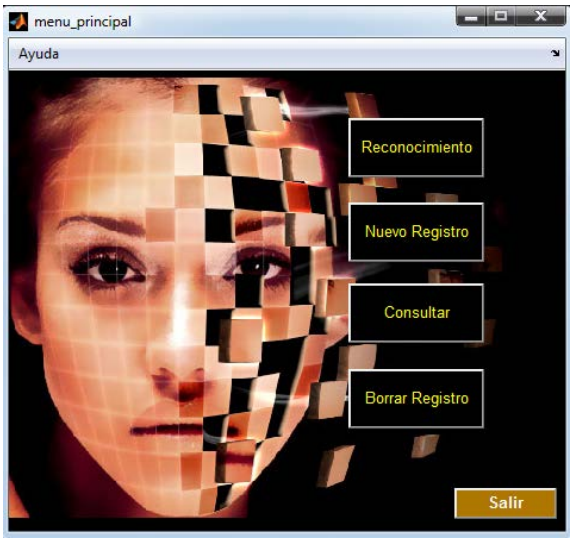


Figura 4.17 Ventana Menú Principal.

La primera opción de la ventana Menú principal es Reconocimiento. Al hacer clic en este botón, se ejecuta una nueva ventana (Reconocimiento de Rostro 2011); La cual tiene como finalidad reconocer el rostro del usuario, mediante la captura de imagen del rostro, de la cual se extraerán las características esenciales, para comparar estos datos con los guardados en la base del sistema y así emitir un posible candidato, del cual se mostrara su imagen y su nombre, de no existir su registro dentro de la base de datos se imprimirá por la ventana un mensaje no se encuentra y la imagen aparecerá con una inscripción “No Disponible”.

El usuario a ser reconocido debe efectuar una serie de pasos: configurar el dispositivo (webcam), activar el canal de video, capturar una imagen cumpliendo con los requisitos del sistema (nivel de iluminación y punto de referencia), confirmar la imagen y presionar el botón buscar.

Para explicar, lo anteriormente dicho se muestra en la figura 4.18 el diagrama de flujo para realizar el reconocimiento de un rostro.

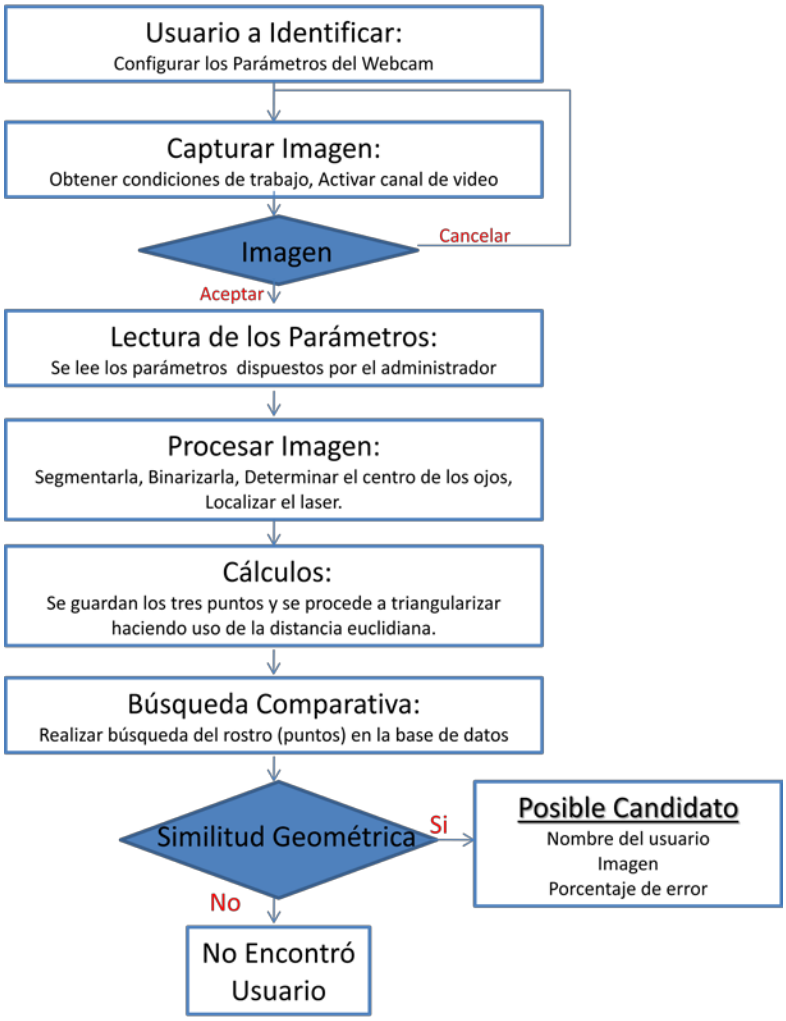


Figura 4.18 Diagrama de Flujo de Reconocimiento.

Para trabajar en la ventana se explicará las opciones que posee y como usarlas. Se muestra en la figura 4.19 y de ella se pueden destacar tres áreas de interés para el usuario del software.

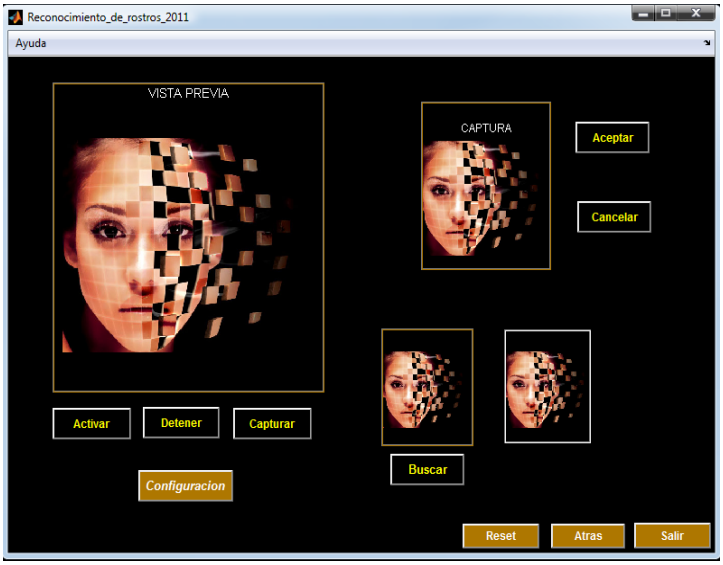


Figura 4.19 Ventana de Reconocimiento de Rostro 2011.

- ✓ Área de Vista Previa.
- ✓ Área de Captura.
- ✓ Área de Resultados.

El área de vista previa de la captura de la Webcam, su función es mostrar el entorno capturado por la cámara (webcam). Además se pueden asociar con esta área los siguientes botones: Configuración, Activar, Desactivar y Capturar; Como se muestra en la figura 4.20.

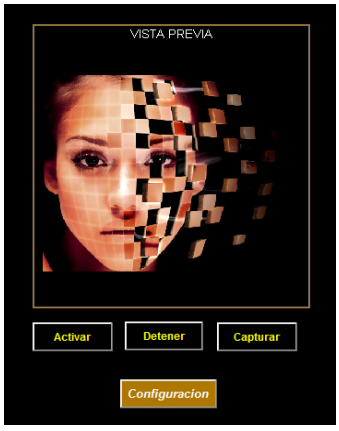


Figura 4.20 Vista Previa de la Imagen a Capturar.

En una PC, se pueden contar con diferentes dispositivos de captura, es por esto que consideraron necesario agregar el botón configurar, con el cual el usuario pueda seleccionar que dispositivo prefiere utilizar para realizar la captura de la imagen. En la figura 4.21 se muestra la ventana para configuración dispositivo a utilizar.

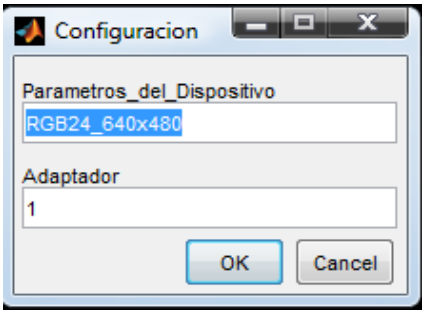


Figura 4.21 Ventana de Configuración del Dispositivo de Captura.

Es oportuno recordar, que para obtener información acerca de la cantidad de dispositivos teclee la siguiente instrucción en la ventana de trabajo de Matlab imaqhwinfo.

Después de haber tomado una foto con la webcam es necesario mostrar la imagen en la ventana, para determinar si la captura fue realizada correctamente, para esto se creó el área de captura (ubicada en la parte superior derecha de la ventana reconocimiento de rostro 2011), al hacer clic sobre el botón capturar, en esta sección se observará la fotografía capturada por cámara web, para que el usuario tenga la posibilidad de aceptar o cancelar la captura de dicha imagen. Una vez aceptada la foto, ésta aparecerá en la región inferior derecha de la ventana en estudio; luego al presionar el botón buscar, en el recuadro aparecerán los datos (imagen y nombre) del posible candidato como fue mencionado anteriormente. Es importante destacar, que dicha ventana posee además los siguientes botones: Reset, Atrás, Salir (ver figura 4.19). El botón reset reinicializa la interfaz, el botón Atrás, retorna a la ventana anterior (menú principal) y el botón Salir, sale del programa.

La segunda opción de la ventana menú principal es Nuevo Registro, al hacer clic sobre este botón aparece la ventana administrador, en la cual el administrador del sistema debe teclear la contraseña para así tener acceso a la opción señala, esta ventana previa de acceso se muestra en la figura 4.22.

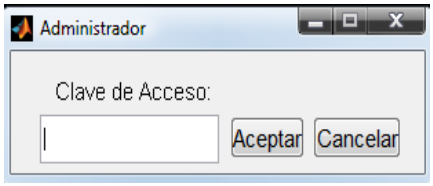


Figura 4.22 Ventana para introducir la clave de acceso del sistema.

Una vez introducida correctamente la clave del sistema, inmediatamente el administrador tendrá acceso a la ventana registro, la cual se muestra en la figura 4.23; en esta ventana el operador podrá añadir personas a la base de datos del sistema o crear una nueva base de datos. Además la ventana tiene botones y de que son de interés para el administrador, por cual se procede a explicar cada uno de ellos.

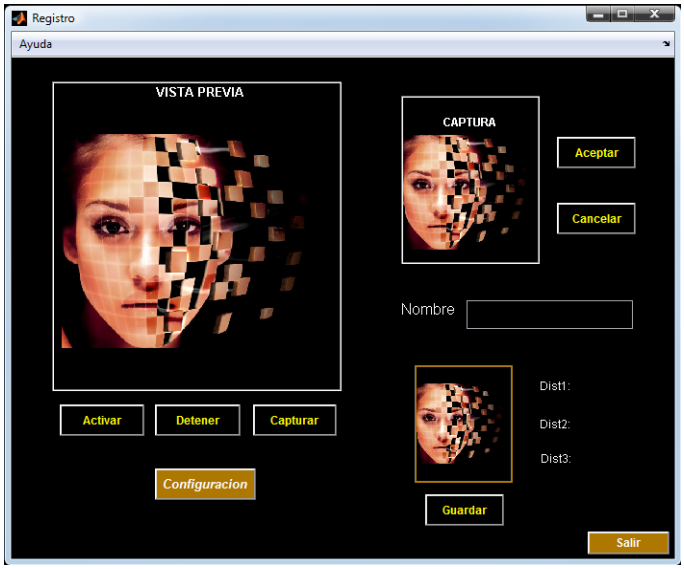


Figura 4.23 Ventana Registro.

De manera similar a la ventana reconocimiento de rostro 2011, la ventana registro posee un área de vista previa y una de captura las cuales conservan sus propiedades; también posee el campo Nombre, este debe ser llenado por el administrador y es un dato suministrado por el usuario (por ejemplo: Nombre y Apellido) no debe exceder los 25 caracteres, no debe poseer puntos, ni comenzar con espacio; este nombre debe ser guardado por el usuario para futuras consultas. A continuación se explica su funcionamiento mediante el recurso de un diagrama de flujo, el cual es mostrado en la figura 4.24.

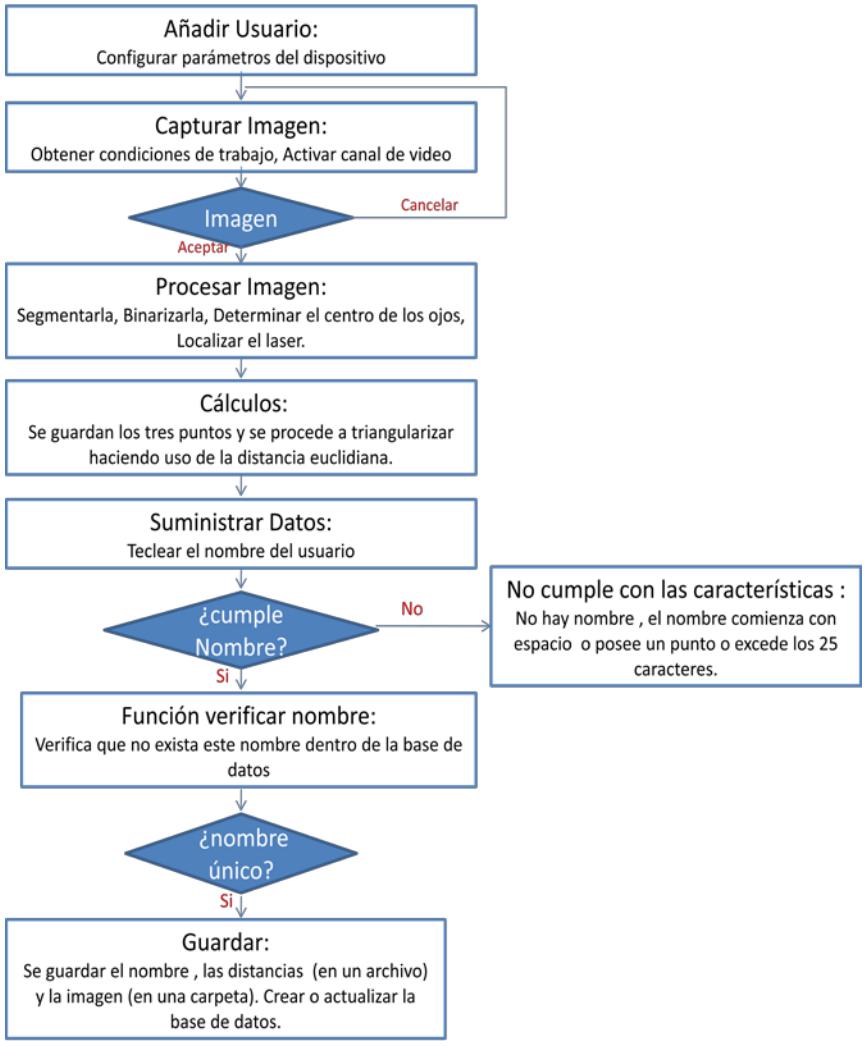


Figura 4.24 Diagrama de Flujo de Registro.

La tercera opción del menú principal es consultar, al hacer clic en este botón aparecerá una nueva ventana consultar registró, ésta tiene tres botones: Consultar, base de datos y atrás; además posee el campo nombre donde el usuario introducirá el nombre con el cual fue registrado en la base de datos.

Al pulsar consultar aparecerá la imagen con que fue registrado y el nombre del usuario, de no estar registrado en el sistema, en el recuadro imagen aparecerá “imagen no disponible”; el botón atrás retorna a la ventana menú principal. La ventana Consultar Registro se muestra a continuación en la figura 4.25.

Figura 4.25 Ventana Consultar Registro.

También, se muestra en la figura 4.26 el diagrama de flujo de la metodología usada para consultar registro.

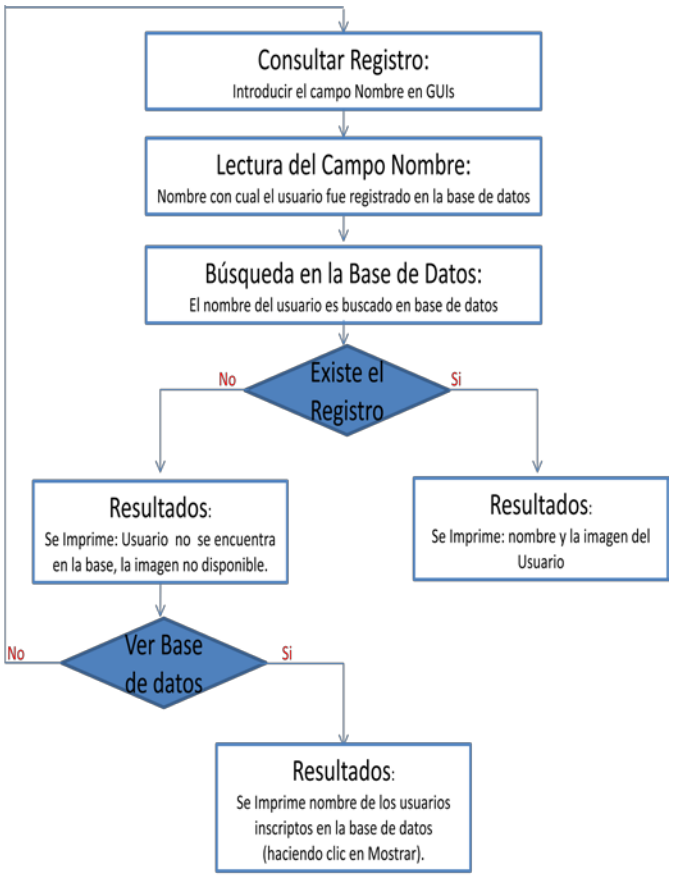


Figura 4.26 Diagrama de Consultar

Al hacer clic sobre el botón Base de datos se abrirá la ventana Base de datos (ver figura 4.27). Ésta posee tres botones: Mostrar, Siguiente y Atrás; al hacer clic sobre el botón Mostrar aparecerá la lista que contiene los usuarios registrado en la base de datos del sistema, para continuar leyendo la lista deberá hacer clic en el botón Siguiente (hasta observar todo el listado, si así lo desea) cuando termine se le informara al usuario a través de un mensaje de ayuda que ya no existe más usuarios inscriptos en la base de datos. El botón atrás retorna a la ventana consultar registro.

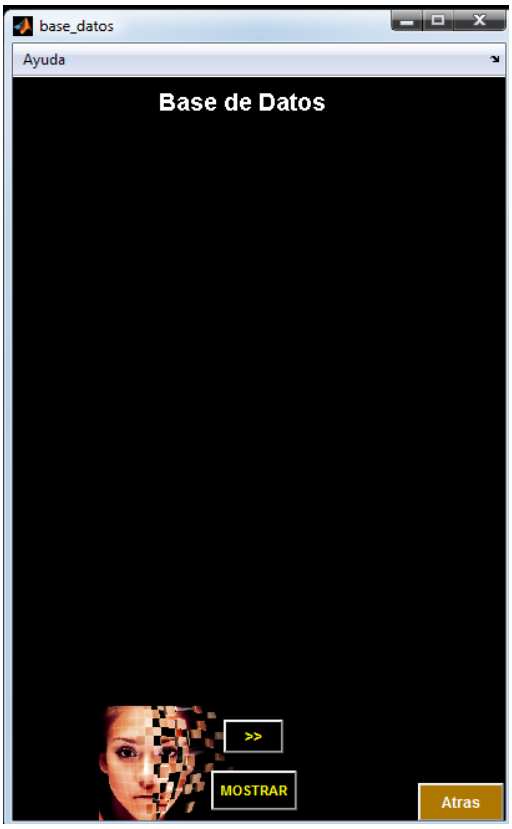


Figura 4.27 Ventana Base de Datos

La cuarta opción de la ventana menú principal es Borrar Registro, al hacer clic a esta opción se ejecuta la ventana administrador, para introducir la clave de acceso (ver figura 4.22). Una vez cumplido el requisito de acceso se ejecuta la ventana Borrar, en ella se modifica la cantidad de usuario inscriptos en la base de datos; es decir, en ella se borran registros de la base de datos, para lo cual el administrador debe tipiar el nombre del usuario que desea eliminar de la base de datos del sistema; además ella tiene una región donde se emitirá un mensaje indicando si dicho usuario fue eliminado exitosamente de la base de datos, también tiene el botón atrás que retorna al ventana anterior (menú principal); La ventana borrar registro se muestra en la figura 4.28.

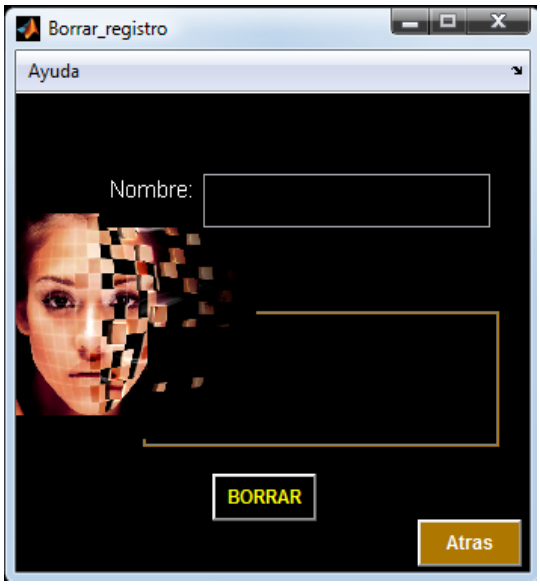


Figura 4.28 Ventana Borrar Registro.

Además, se muestra en la figura 4.29 el diagrama de flujo que indica la lógica usada en esta ventana para eliminar una persona de la base de datos del sistema.

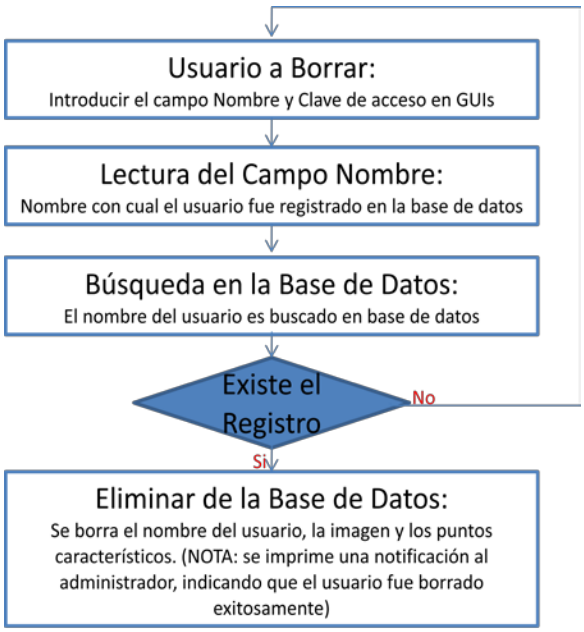


Figura 4.29 Diagrama de Flujo de Borrar registro.

Tambien, este software ofrece información de los parámetros y ventanas existentes en el programa, mediante la ventana ayuda; la cual puede ejecutarse desde cada una de las ventanas del sistema y aparece como pestaña (con el nombre de Ayuda) en la parte superior izquierda de cada interfaz. El manejo de esta ventana es muy sencillo, consiste en buscar en la lista la opción o ventana de la cual desea obtener información y hace clic en ella, inmediatamente se mostrará una breve descripción del comando. Se muestra en la figura 4.30.

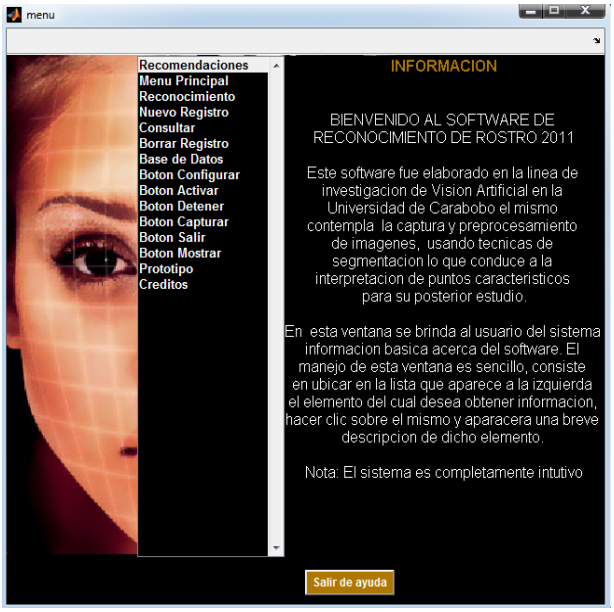


Figura 4.30 Ventana de Ayuda.

Es importante resaltar que el sistema es interactivo a lo largo de su ejecución se emite mensajes al usuario para ayudarle en el manejo del software.

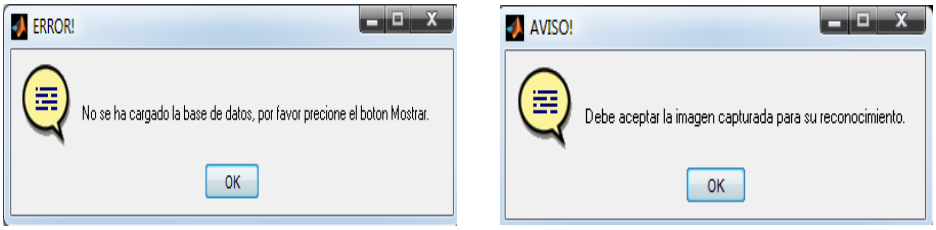


Figura 4.31 Ejemplo de los mensaje de ayuda emitidos al usuario

4.8 CONSTRUCCIÓN DEL PROTOTIPO

Debido a la necesidad de tener el rostro a una posición fija con respecto a la ubicación de la cámara web, un nivel de iluminación controlado (similar para cada usuario) y punto de referencia externo (laser), se diseñó una estructura en madera que cumpla con estos requerimientos.

Para explicar ésta estructura en forma clara para el lector y de acuerdo con su función, se decidió separarla en sus partes principales:

- ✓ Estructura para posicionar el rostro del usuario.
- ✓ Plataforma de soporte.
- ✓ Pieza para evitar deslumbramiento indeseado.
- ✓ Pieza para fijar el punto de referencia.

Además, de las piezas el prototipo posee dos lámparas recargables, una cámara convencional y un laser con pulsador.

Para garantizar que el usuario tenga el rostro en una posición fija con respecto a la webcam, se construyo una pieza tipo mentonera cuya forma permite la adaptación cómoda y correcta del usuario. Para mayor detalle de ella, se presenta la figura 4.32 donde se destacan sus características esenciales y sus dimensiones.

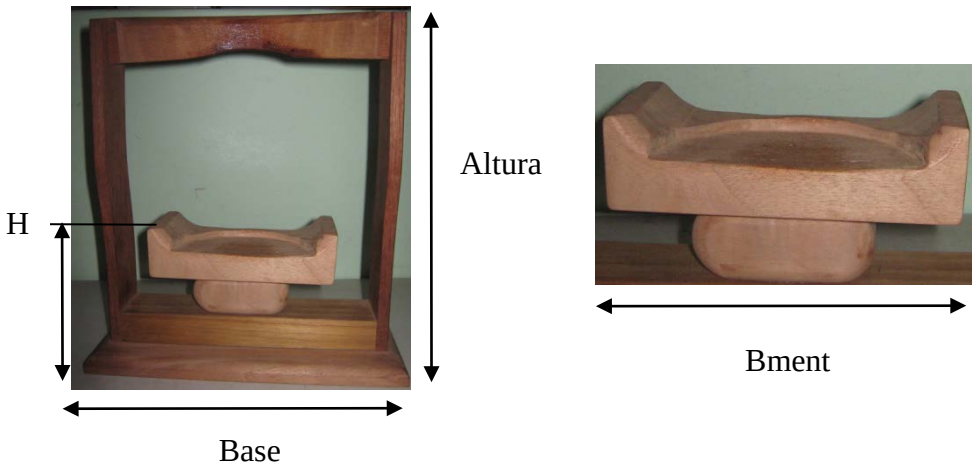


Figura 4.32 Pieza para colocar el rostro.

Donde:

Tabla 4.11 Dimensiones de la pieza para posicionar el rostro.

Variable	Dimensiones (cm)
Base	20.2
Altura	27.2
H	9.6
Bment	13.9

Para colocar con precisión todas las piezas del prototipo se diseño una plataforma de soporte, en la misma se ubicaron los siguientes elementos: pieza para posicionar el rostro (mentonera), dos piezas para evitar deslumbramiento, estructura del punto de referencia (laser), cámara web y dos lámparas recargables.

En vista de la necesidad de extraer los puntos característicos (ojos), se requirió tener una buena iluminación lo que condujo a usar dos lámparas, éstas generaron un inconveniente

produciendo un deslumbramiento en zonas no deseadas del rostro, por lo cual se diseñaron dos piezas que proporcionaron una solución práctica a dicho deslumbramiento. En la figura 4.33 se muestra las piezas utilizadas y las dimensiones de la misma.

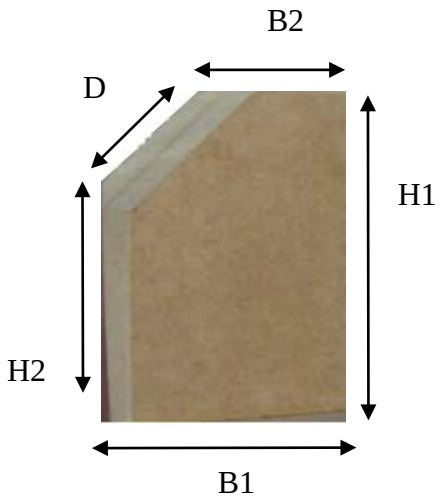


Figura 4.33 Pieza para evitar deslumbramiento.

Donde las dimensiones fueron reseñadas en la tabla 4.12

Tabla 4.12 Dimensiones de la pieza**.**

Variable	Dimensiones (cm)
B1	7.9
B2	3.5
H1	11.6
H2	7.6
D	6

Para colocar el punto de referencia (laser) en la ubicación deseada, se construyó una estructura la cual se muestra en la figura 4.34 y sus dimensiones.

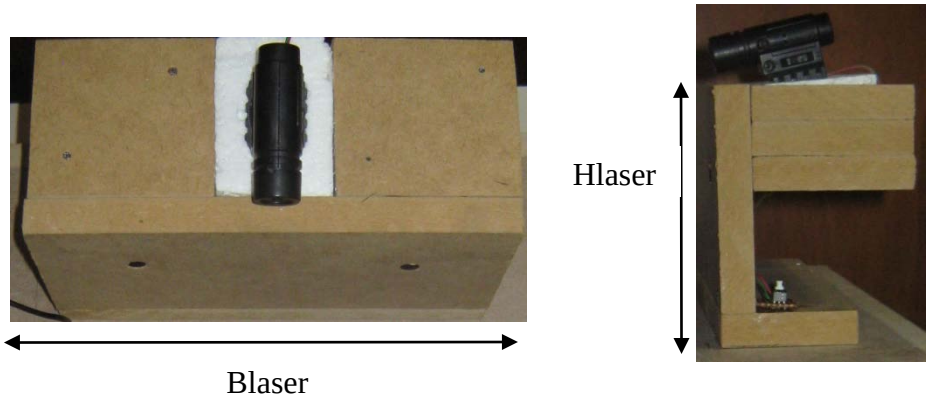


Figura 4.34 Pieza para colocar el laser.

Donde el tamaño de la pieza se encuentra en la tabla 4.13.

Tabla 4.13 Dimensiones de la pieza.

Variable	Dimensiones (cm)
Blaser	11.5
Hlaser	20

En la figura 4.35, se muestra el prototipo completo y las diferentes vistas para obtener percepción correcta del mismo.

Figura 4.35 Estructura en estudio y sus diferentes vistas.

A continuación se describen las características generales de cada uno de los dispositivos usados en el prototipo:

- ✓ **Cámara Web**:

WEBCAM I-LOOK 111 USB

Descripción

Genius *i*-Look 111 permite capturar imágenes de vídeo y fotografías con sólo pulsar un botón. Su base es ajustable giratoria 360º e incluye función zoom in/out.

Ficha Técnica

Características:

- Captura imágenes y vídeo animado.
- Base ajustable con 360º de rotación.
- Zoom in/out para una mayor claridad.
- Vídeo vigilancia permite el uso como sistema de seguridad.

- Convierte ficheros AVI a MPEG.
- Botón para captura de imágenes.

Especificaciones:

- Selección de resolución, **RGB24:** 640x480, 352x288, 320x240, 176x144, 160x120 píxeles.

I240: 640x480, 352x288, 320x240, 176x144, 160x120 Píxeles

- Lentes: Foco manual.
- Balance de blancos: Auto.
- Formatos de imagen:

* JPEG: 640 x 480 pixeles máx.
* AVI: hasta 30fps.

- Dimensiones: 84.8 x 76 x 92.7 mm.

Requisitos del Sistema:

- Entorno Windows® IBM PC Pentium II 400 compatible o superior.
- 64MB DRAM.
- 25MB de espacio libre en disco.
- Windows VISTA,XP/Me/2000/98SE.
- Puerto USB.
- CD-ROM para instalación de software.

- ✓ **Lámparas de emergencia:**

AMCO LIGHTING Modelo 1045C

-Linterna recargable

-Fija y portátil

-24x4pzs.x1500MCDde luces led luminosas

-Batería de ácido de plomo de 12 placas recargable

-Carga completa de batería en menos de 20 horas con voltaje de 110 voltios

Especificaciones:

-Alimentación: 110 Voltios ~ 60 Hertz

-Potencia: 3 Watt

-Batería (1 batería tipo GH642 de 6Vcc 4.2Ah).

5.1 ANÁLISIS DE RESULTADOS

Como se mencionó en el capítulo anterior, la base de datos consta de 20 personas, cada una de éstas posee en su registro los siguientes datos:

- ✓ Una imagen del rostro (en formato RGB) a una resolución de 480x640 pixeles.
- ✓ El nombre con el cual fue registrado, el cual cumple con las características del sistema (no excede de veinticinco caracteres de los cuales está excluido el punto y no puede comenzar con el carácter espacio).
- ✓ Las distancias entres los puntos característicos del rostro (distancia entre el ojo derecho y la referencia, distancia entre el ojo izquierdo y la referencia, distancia entre los ojos).

Además, dichas personas fueron tomadas de forma aleatoria entre la población estudiantil de la Universidad de Carabobo.

Para estudiar la eficiencia del software, se hicieron cuarentas pruebas a cuatro personas distintas registradas en la base de datos del sistema. Los resultados se clasificaron de la siguiente manera:

- ✓ **Acierto**: identifica correctamente al usuario en estudio.
- ✓ **Desacierto**: no encontró al usuario en el sistema.
- ✓ **Confusiones**: encontró un candidato, pero éste no es el usuario en estudio.

Los resultados obtenidos en cada prueba fueron reseñados en tablas, una para cada persona y el porcentaje de aciertos (desaciertos/equivocaciones) fue mostrado en un gráfico circular por cada tabla.

Tabla 5.1 Resultados obtenidos con la persona 1

Persona 1	Aciertos	Desaciertos	Confusiones
Prueba 1	x		
Prueba 2	x		
Prueba 3	x		
Prueba 4	x		
Prueba 5	x		
Prueba 6	x		
Prueba 7	x		
Prueba 8	x		
Prueba 9	x		
Prueba 10	x		
Prueba 11	x		
Prueba 12	x		
Prueba 13	x		
Prueba 14	x		
Prueba 15	x		
Prueba 16	x		
Prueba 17	x		
Prueba 18	x		
Prueba 19	x		
Prueba 20	x		
Prueba 21	x		
Prueba 22	x		
Prueba 23		x	
Prueba 24	x		
Prueba 25	x		
Prueba 26	x		
Prueba 27	x		
Prueba 28	x		
Prueba 29	x		
Prueba 30	x		
Prueba 31	x		
Prueba 32	x		
Prueba 33	x		
Prueba 34	x		
Prueba 35	x		
Prueba 36		x	
Prueba 37	x		
Prueba 38	x		
Prueba 39	x		
Prueba 40	x		
TOTAL	38	2	0

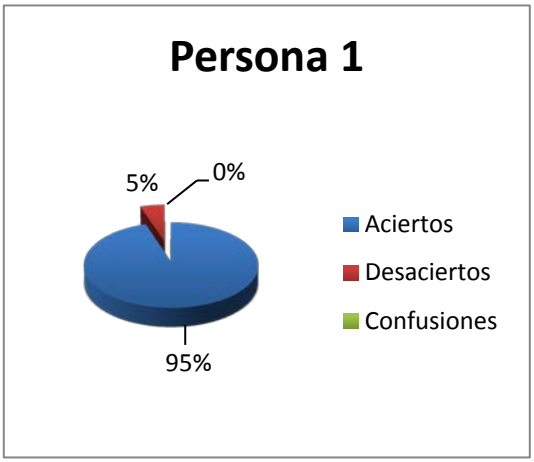


Figura 5.1 Porcentaje obtenido con la muestra 1.

Tabla 5.2 Resultados obtenidos con la persona 2.

Persona 2	Aciertos	Desaciertos	Confusiones
Prueba 1	x		
Prueba 2	x		
Prueba 3		x	
Prueba 4		x	
Prueba 5	x		
Prueba 6	x		
Prueba 7	x		
Prueba 8	x		
Prueba 9	x		
Prueba 10	x		
Prueba 11	x		
Prueba 12		x	
Prueba 13	x		
Prueba 14	x		
Prueba 15	x		
Prueba 16	x		
Prueba 17	x		
Prueba 18		x	
Prueba 19	x		
Prueba 20	x		
Prueba 21	x		
Prueba 22	x		
Prueba 23	x		
Prueba 24	x		
Prueba 25	x		
Prueba 26	x		
Prueba 27	x		
Prueba 28	x		
Prueba 29	x		
Prueba 30	x		
Prueba 31	x		
Prueba 32	x		
Prueba 33	x		
Prueba 34	x		
Prueba 35		x	
Prueba 36	x		
Prueba 37	x		
Prueba 38	x		
Prueba 39		x	
Prueba 40	x		
TOTAL	34	6	0

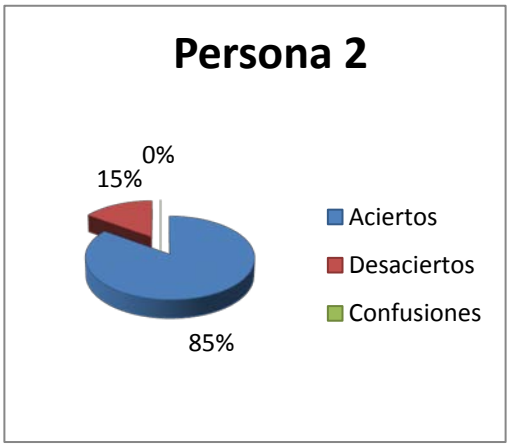


Figura 5.2 Porcentaje obtenido con la muestra 2.

Tabla 5.3 Resultados obtenidos con la persona 3.

Persona 3	Aciertos	Desaciertos	Confusiones
Prueba 1	x		
Prueba 2	x		
Prueba 3	x		
Prueba 4	x		
Prueba 5	x		
Prueba 6	x		
Prueba 7	x		
Prueba 8		x	
Prueba 9	x		
Prueba 10	x		
Prueba 11	x		
Prueba 12	x		
Prueba 13	x		
Prueba 14	x		
Prueba 15	x		
Prueba 16	x		
Prueba 17	x		
Prueba 18	x		
Prueba 19	x		
Prueba 20	x		
Prueba 21	x		
Prueba 22	x		
Prueba 23	x		
Prueba 24	x		
Prueba 25	x		
Prueba 26	x		
Prueba 27	x		
Prueba 28	x		
Prueba 29	x		
Prueba 30	x		
Prueba 31	x		
Prueba 32	x		
Prueba 33	x		
Prueba 34	x		
Prueba 35	x		
Prueba 36	x		
Prueba 37	x		
Prueba 38	x		
Prueba 39	x		
Prueba 40	x		
TOTAL	39	1	0

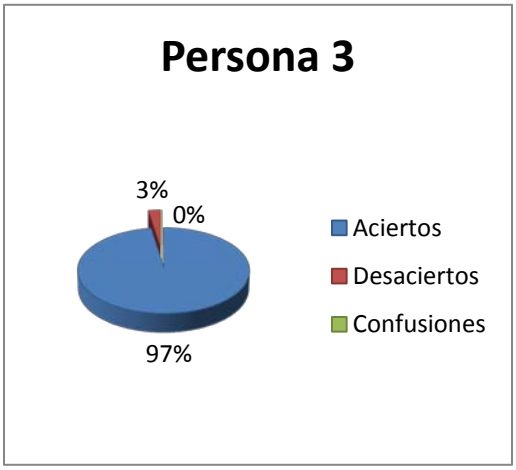


Figura 5.3 Porcentaje obtenido con la muestra 3.

Tabla 5.4 Resultados obtenidos con la persona 4.

Persona 4	Aciertos	Desaciertos	Confusiones
Prueba 1	x		
Prueba 2	x		
Prueba 3	x		
Prueba 4	x		
Prueba 5	x		
Prueba 6	x		
Prueba 7	x		
Prueba 8	x		
Prueba 9	x		
Prueba 10	x		
Prueba 11	x		
Prueba 12	x		
Prueba 13	x		
Prueba 14	x		
Prueba 15	x		
Prueba 16	x		
Prueba 17	x		
Prueba 18	x		
Prueba 19	x		
Prueba 20	x		
Prueba 21	x		
Prueba 22	x		
Prueba 23	x		
Prueba 24	x		
Prueba 25	x		
Prueba 26	x		
Prueba 27	x		
Prueba 28	x		
Prueba 29	x		
Prueba 30	x		
Prueba 31	x		
Prueba 32	x		
Prueba 33	x		
Prueba 34	x		
Prueba 35	x		
Prueba 36	x		
Prueba 37	x		
Prueba 38	x		
Prueba 39	x		
Prueba 40	x		
TOTAL	40	0	0

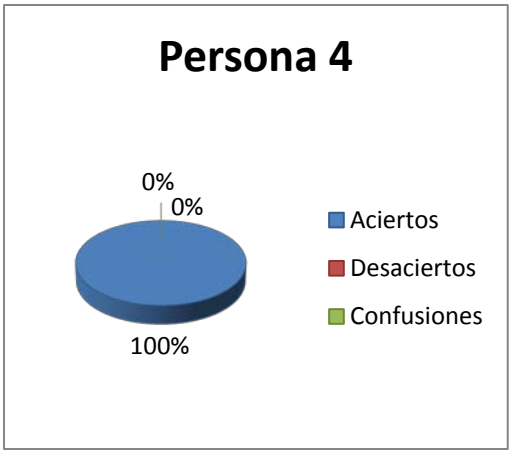


Figura 5.4 Porcentaje obtenido con la muestra 4.

Las cientos sesentas pruebas realizadas distribuidas entre cuatro personas (cuarenta cada una) mostradas en la tabla 5.1, 5.2, 5.3 y 5.4 fueron resumidas en la tabla 5.5. En ella se muestra la cantidad de acierto, desacierto; con el fin de determinar el porcentaje de eficiencia del software.

Tabla 5.5 Resultados totales obtenidos con las pruebas realizadas.

Respuesta	Cantidad/Porcentaje
Aciertos	151(94.25%)
Desaciertos	9 (5.75%)
Confusiones	0 (0%)

Además, se elaboró el gráfico representado en la figura 5.5 obtenido a partir de tabla 5.5 donde muestra de forma general el porcentaje de eficiencia del software (94%).

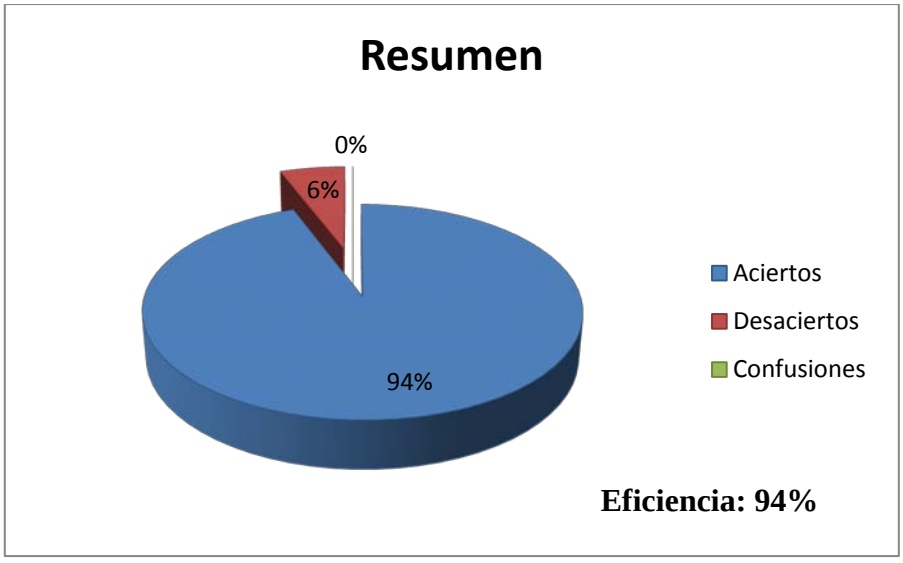


Figura 5.5 Resultados totales obtenidos con las pruebas realizadas.

Después de realizar y analizar las pruebas se puede decir que a partir de la eficiencia obtenida del sistema, a juicio personal se puede clasificar como muy bueno.

Es necesario destacar, que los resultados obtenidos dependen no solo del software, sino también de la utilización correcta del prototipo; es decir, deben tener un nivel de luminosidad óptimo (lámparas cargadas) y que la persona a ser reconocida conserve algunos parámetros, como por ejemplo: posicionar su rostro igual, esto se consigue colocando la cara de la forma más cómoda para el usuario (para así garantizar está posición sea la misma, cada vez que use el sistema) y mirar siempre a mismo lugar durante la captura de la imagen.

5.2 CONCLUSIONES

- ✓ De acuerdo con los resultados obtenidos en el capitulo cuatro se obtuvo en el sistema de reconocimiento de rostros un nivel de eficiencia de 94%, es evidente que los objetivos por los cuales fue desarrollado este proyecto fue cumplido satisfactoriamente.

- ✓ Se logró seleccionar con éxitos los equipos implementados para el sistema de reconocimiento de rostros.

- ✓ Se pudo determinar que la base de datos juega un papel importante e influyente en los resultados, ya que la posición del rostro, la dirección de la mirada y el nivel de iluminación es una limitante para la posibilidad de acierto en el software.

- ✓ Se logró realizar una interfaz grafica amigable e interactiva con el usuario, además se le brinda al administrador del sistema la posibilidad de acceder a la base de datos.

- ✓ El diseño de este algoritmo posee una sencillez comparado con los que se encuentran actualmente como lo son las redes neuronales, los vectores de componentes principales

entre otros; ya que éste se compone principalmente de las técnicas clásicas de segmentación de imágenes.

- ✓ Finalmente se puede decir que el sistema de reconocimiento es versátil, económico, eficiente en un nivel aceptable y el tiempo mínimo de respuesta es bastante rápido.

5.3 RECOMENDACIONES

- ✓ El no mirar siempre al mismo punto es una limitante para el software, por lo cual se le recomienda al usuario del sistema mirar un punto de referencia al momento de registrarse (preferiblemente el punto rojo dentro de la cámara web); con la finalidad de garantizar el porcentaje de eficiencia al momento de realizar el reconocimiento.

- ✓ La eficiencia del software depende de la intensidad lumínica de las lámparas, por lo tanto, se recomienda que las lámparas estén completamente cargada y después de cuatro horas consecutivas encendidas apagarlas y cargarlas completamente.

- ✓ Se recomienda cada cierto tiempo cambiar las baterías al laser, debido a que si este pierde su intensidad el software carecerá de un parámetro y el mismo no permitirá el registro o reconocimiento de un usuario.

- ✓ Se recomienda actualizar la base de datos periódicamente debido a los cambios en los rasgos faciales a través del tiempo.

Trabajo futuro:

- ✓ Colocar un segundo punto de referencia (laser), lo que generará una mayor efectividad al posible candidato.

- ✓ Se recomienda el almacenamiento de los usuarios a la base de datos a través de la cédula de identidad, ya que este patrón es único para cada persona.

- ✓ Colocar el incentro del triangulo en estudio (triangulación del rostro), como otro parámetro de comparación.

- ✓ Capturar varias imágenes de cada usuario con diferentes poses con la finalidad de obtener una distancia promedio.

REFERENCIAS BIBLIOGRÁFICAS

[1] Gonzalo P., Jesús M. de la cruz (2001) VISION POR COMPUTADOR, IMAGENES DIGITALES. Editorial Alfaomega.

[2] Garcia G. (2007). Procesamiento de caras humanas mediante integrales proyectivas. Universidad de Murcia.

[3] Reyes C. (2005) RECONOCIMINETO FACIAL MEDIANTE VISIÓN ARTIFICIAL EN ENTORNO DE PROGRAMACION OPENCV.

[4] Palacios S. (2004) SISTEMA DE RECONOCIMIENTO DE ROSTROS.

[5] Valvert J. (2004) MÉTODOS Y TÉCNICAS DE RECONOCIMIENTO DE ROSTROS EN IMÁGENES DIGITALES BIDIMENSIONALES.

[6] De Ponte C., Ojeda J. (2004) SISTEMA DE CONTROL DE ACCESO AUTOMÁTICO BASADO EN TÉCNICAS DE VISION ARTIFICIAL PARA EL RECONOCIMIENTO DE ROSTRO.

[7] Rojas T. (2007) DISEÑO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA UN MANIPULADOR INDUSTRIAL ANTROPOMÓRFICO. Universidad de Carabobo.

[8] Peralta A. (2009) RECONOCIMIENTO DE IMÁGENES A TRAVÉS DE SU CONTENIDO. Madrid, España.

[9] Gámez C. (2009) DISEÑO Y DESARROLLO DE UN SISTEMA DE RECONOCIMIENTO DE CARAS MEDIANTE PCA.

[10] Cortés M. (2009) RECONOCIMIENTO DE CARAS FRONTALES MEDIANTE LA EXTRACCIÓN DE PUNTOS CARACTERÍSTICOS.

[11] Florian L., Carranza F. (2009) RECONOCIMIENTO DEL IRIS.

[12] Hernández J. (2009) RECONOCIMIENTO DE ROSTROS UTILIZANDO HISTOGRAMAS SECUENCIALES DE IMAGEN**.**

[13] Dragolici A. (2010) DETECCION, LOCALIZACION E IDENTIFICACION BIOMETRICA DE CARAS EN IMÁGENES: METODOS Y EVALUACION EN EL MARCO NIST- MBGG.

[14] De la Escalera H., (2001) VISIÓN POR COMPUTADOR, FUNDAMENTOS Y MÉTODOS. Editorial Prentice Hall, Madrid, España.

[15] Bartolomé F. Imagen digital, [Documento en línea], http://blog.interdominios.com /2009/07/ [Consulta: 2010, Noviembre 5].

[16] Colmenares L. (2007). Estudio del seguimiento en tiempo real de agujas quirúrgicas durante procedimientos de entrenamiento en neurocirugía. Venezuela.

[17] ING. Tinoco R. (2009). SISTEMA DE RECONOCIMIENTO FACIAL POR MEDIO DE EIGENFACES Y REDES NEURONALES. MÉXICO, D. F.

[18] Academic Spanish. Esquema del modelo RGB, [Documento en línea], http://www.esacademic.com/ dic.nsf/eswiki/2910 [Consulta: 2010, Noviembre 5].

[19] Aguilar S. Esquema del modelo HSV. [Documento en línea], http://aprende.colorotate.org/color-models.html [Consulta: 2010, Noviembre 5].

[20] Técnicas de segmentación [Consulta en línea]. Disponible: http://catarina.udlap.mx/u_dl_a/ tales/documentos/lem/carcedo_y_a/capitulo1.pdf [Consulta: 2011, Enero 05].

[21] González R. (1993) Digital Image Processing. Addison Wesley. Publishing Company.

[22] Forsyth D. (2003) Computer vision: A modern approach. Prentice Hall.

[23] Monografias, Matlab [Documento en línea]. Disponible: http://www.monografias.com/trabajos5/matlab/matlab.shtml [Consulta: 2011, Enero 5].

[24] Tosina J. (2005) Diseño de un filtro morfológico para la reducción de fluorescencia.

[25] Platero C. (2009) APUNTES DE VISIÓN ARTIFICAL, PROCESAMIENTO MORFOLOGICO.

[26] Sobrado E. (2003) SISTEMA DE VISIÓN ARTIFICIAL PARA EL RECONOCIMIENTO Y MANIPULACIÓN DE OBJETOS UTILIZANDO UN BRAZO ROBOTVISIÓN POR COMPUTADORA.

[27] Tutorial de imágenes, [Documento en línea]binarización, http://www.dimages.es/Tutorial%20A.I/segmentacion/binaria.htm [Consulta: 2011, Enero 5].

[28] González S. Imagen Digital., [Documento en línea], http://www.slideshare.net/salgonsan/imagen-digital-2033585 [Consulta: 2010, Noviembre 5].

[29] Jiménez E. Zaldivar D. (2003) VISIÓN POR COMPUTADOR UTILIZANDO MATLAB Y EL TOOLBOX DE PROCESAMIENTO DIGITAL DE IMÁGENES.

[30] Cárdenas D., Espinosa J., Vargas J. (2002) Interfaces Gráficas en Matlab Usando GUIDE.

[31] Barrios, M. (1998). Manual de trabajos de grado, de especialización y maestría y tesis doctorales. Universidad Pedagógica Experimental Libertador. Caracas: FEDUPEL.

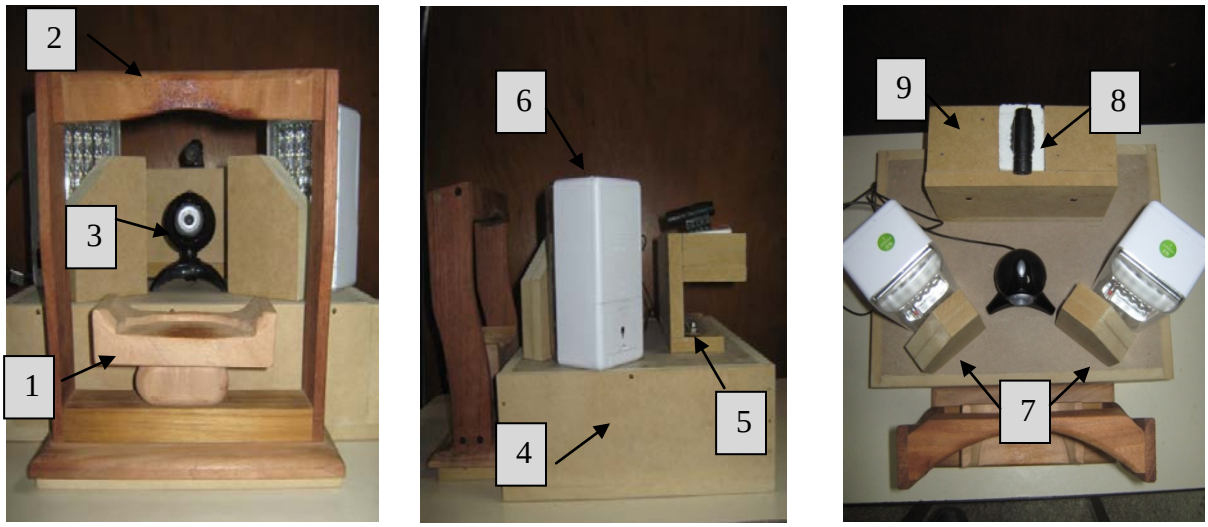
[32] Alfonzo I. (1994). TÉCNICAS DE INVESTIGACIÓN BIBLIOGRÁFICA**.** Caracas: Contexto Ediciones.

MANUAL DE USUARIO PARA EL PROTOTIPO DE RECONOCIMIENTO DE ROSTROS 2011

Esta sección tiene como principal función documentar las pautas concernientes al uso del prototipo de reconocimiento de rostros, entre los puntos que se detallan se encuentran:

- Precauciones.
- Descripción del prototipo.
- Funcionamiento del prototipo.
- Ejecución del software.
  - Registro.
  - Reconocimiento.
- Posibles equivocaciones.

Cabe destacar que el prototipo propuesto para el reconocimiento de rostros como se muestra en la Figura A-1, básicamente consiste en mantener un patrón de referencia de la posición del rostro de cada uno de los usuarios que tendrán parte en éste sistema. Dicha referencia responderá a los requerimientos necesarios para lograr un óptimo funcionamiento del mismo.



(a) (b) (c)

Figura A-1. Prototipo del reconocimiento: (a) Vista delantera, (b) Vista lateral, (c) vista aérea.

Fuente: Elaboración propia.

1 Pieza mentonera.

2 Apoyo frontal.

3 Cámara web.

4 Plataforma de soporte.

5 Pulsador (ON/OFF laser).

6 Lámparas recargables.

7 Piezas para evitar deslumbramiento.

8 Laser.

9 Pieza de soporte al laser.

PRECAUCIONES

Para garantizar el buen funcionamiento del sistema y el bienestar de quienes la operan es necesario tomar en cuenta las siguientes precauciones:

1. Conecte el cable de la cámara web al equipo (computadora) en el puerto USB que se encuentre en funcionamiento.
2. Se debe encender el laser una vez que el usuario se encuentre correctamente ubicado en la mentonera.
3. No dejar permanentemente encendido el laser para registrar varios usuarios, se recomienda encenderlo y apagarlo para cada uno de los registros.
4. No dejar que las lámparas recargables pierda toda su carga, si esto sucede apáguelas cárguelas completamente y vuelva a encenderlas.
5. No apoyarse ni mover los equipos que se encuentran dispuestos en la estructura.

DESCRIPCIÓN DEL PROROTIPO

- ✓ Pieza Mentonera

Esta pieza consta de una mentonera y el apoyo frontal, aquí el usuario dispondrá a colocar su mentón como lo muestra en la figura A-2, en la figura A-3 se puede observa una posición correcta que el usuario deberá mantener en dicha estructura.

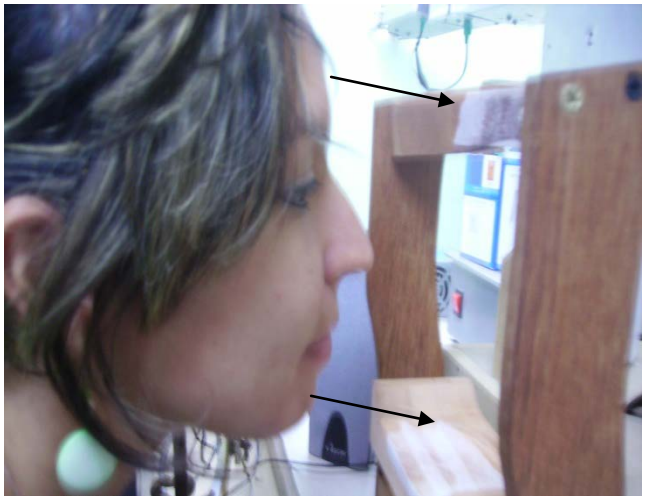


Figura A-2 Como colocarse en la mentonera.

Fuente: Elaboración propia.

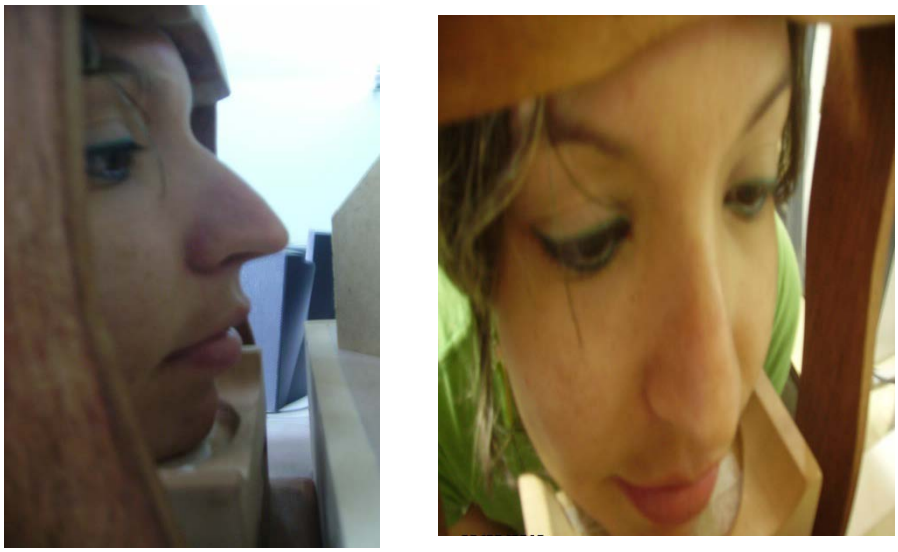


Figura A-3 Posición correcta.

Fuente: Elaboración propia.

FUNCIONAMIENTO DEL PROROTIPO

- ✓ Lámparas recargables

Una vez posicionado correctamente en la pieza mentonera el administrador dispondrá a encender cada una de las lámparas, como la muestra la figura A-4.

Figura A-4 Encendido de las lámparas.
Fuente: Elaboración propia.

- ✓ Laser

Ya estando las lámparas en funcionamiento el administrador deberá encender el laser activando el pulsador como se muestra en la figura A-5, en la figura A-6 se observa el escenario antes descrito y en la figura A-7 el mismo escenario con el usuario puesto en la mentonera.

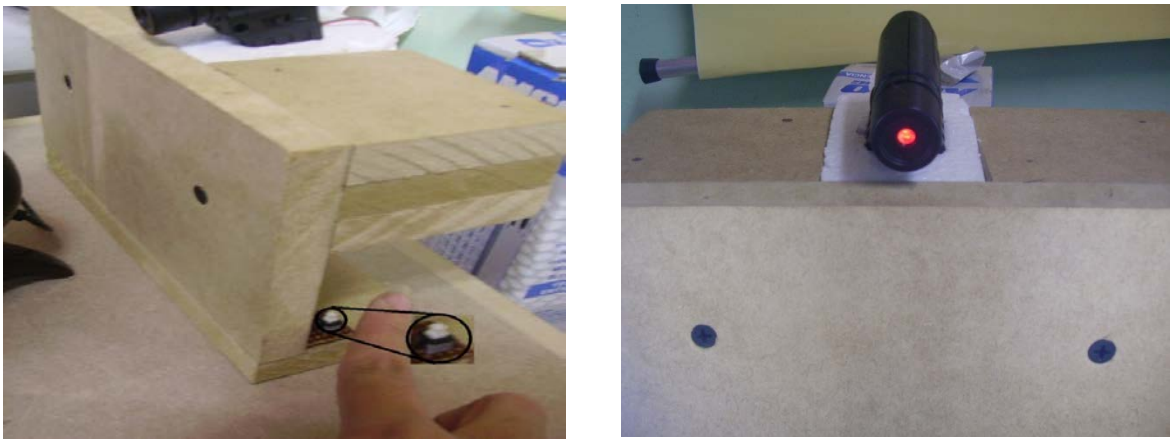


Figura A-5 Encendido del laser.

Fuente: Elaboración propia.

Figura A-6 Escenario obtenido.

Fuente: Elaboración propia.

Figura A-7 Usuario en condiciones para toma de imagen.

Fuente: Elaboración propia.

- ✓ Cámara web

Conecte el cable de la cámara web a un puerto usb de la computadora como se muestra en la figura A-8.

Figura A-8 Conexión de la cámara web a la computadora.

Fuente: Elaboración propia.

EJECUCIÓN DEL SOFTWARE

- ✓ Registro

Una vez posicionado el usuario y el escenario ya dispuesto, el administrador abrirá el menú principal del software de reconocimiento, donde procederá a registrar el usuario. Para entrar a la ventana de registro seleccione el botón nuevo registro del menú principal donde se desplegara una ventana de contraseña como se muestra en la figura A-9.

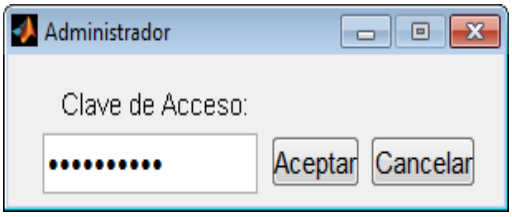


Figura A-9 Clave de acceso.

Una vez introducida la clave se abrirá la ventana de registro como se muestra en la figura A-10.

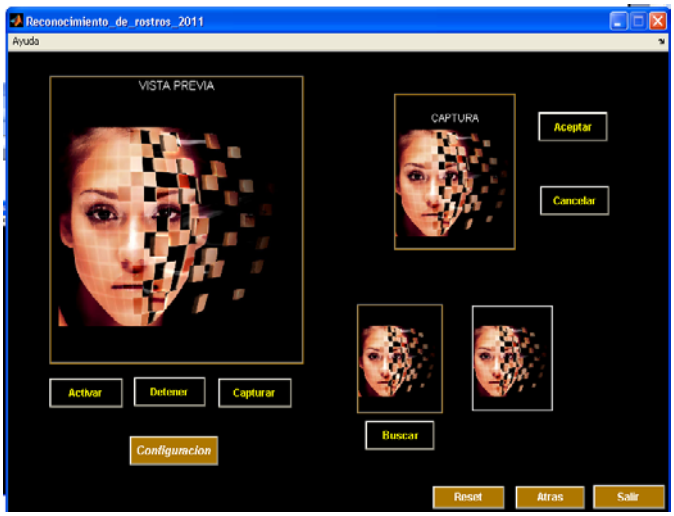


Figura A-10 Conexión de la cámara web a la computadora.

Configure los parámetros como adaptador y resolución de la cámara haciendo click en el botón configuración en la ventana de registro, lo que mostrará una ventana como se observa en la figura A-11.

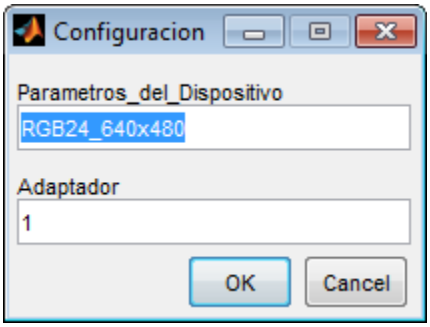


Figura A-11 Configuración de parámetros de la cámara.

Fuente: Elaboración propia.

Una vez hecha la configuración active el canal de video haciendo click en el botón activar de la ventana de registro, dicha apertura del canal se muestra en la figura A-12.

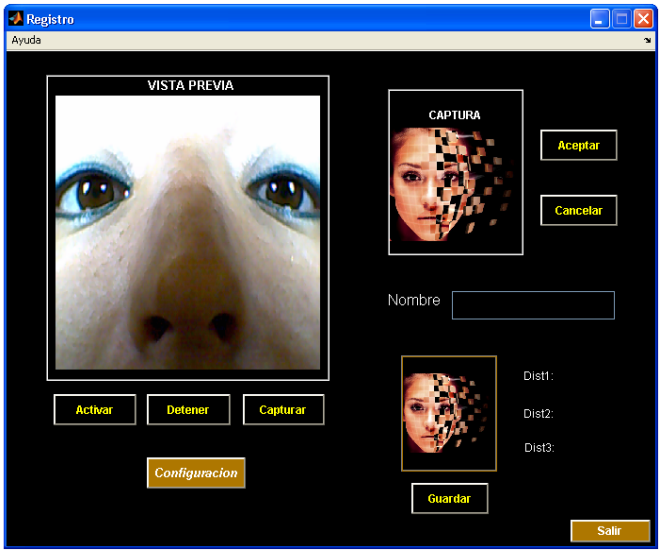


Figura A-12 Apertura del canal de video.

Fuente: Elaboración propia.

Luego encender el laser para así tener todo listo para la captura de la imagen. En la figura A-13 se observa una vez el laser encendido y en la figura A-14 muestra la captura de la imagen.

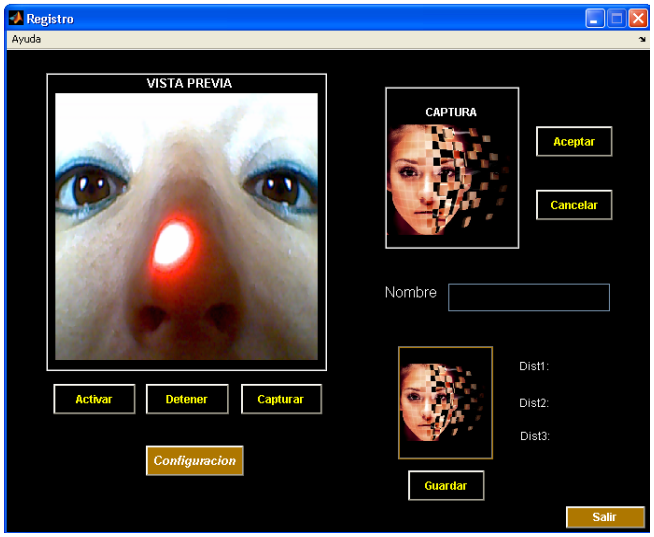


Figura A-13 Encendido del laser para captura de la imagen.

Fuente: Elaboración propia.

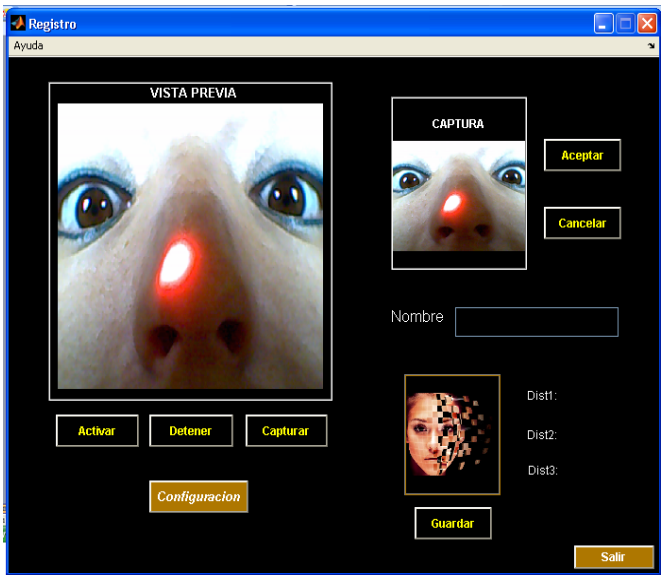


Figura A-14 Captura de la imagen.

Fuente: Elaboración propia.

Una vez captura de la imagen haga click en el botón aceptar para procesar la triangulación del rostro tomando lectura de la Dist1 (distancia del ojo derecho al punto de referencia), Dist2 (distancia del ojo izquierdo al punto de referencia) y Dist3 (distancia del ojo derecho y el ojo izquierdo. En la figura A-15 muestra la los parámetros Dist1, Dist2 y Dist3 leídos de la imagen, luego escriba el nombre del usuario y presione le botón guardar.

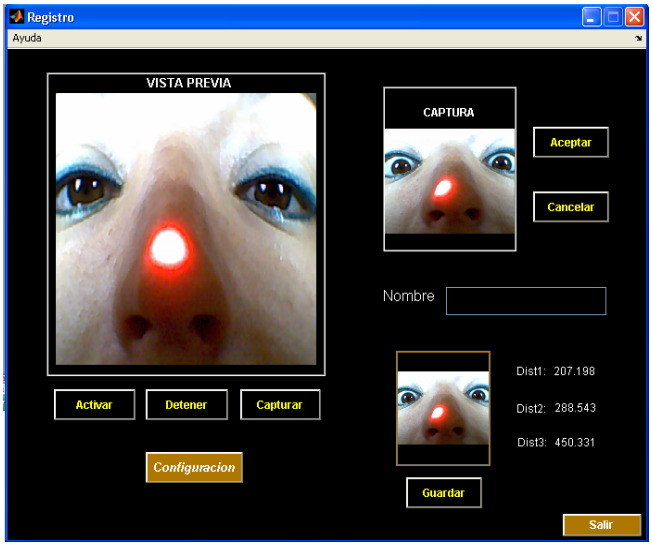


Figura A-15 Captura de la imagen.

Fuente: Elaboración propia.

Ya registrado el usuario presione salir>>ok, aquí le preguntará el sistema si ya existe alguna base de datos, si ese usuario solo lo quiere añadir a la base de datos presione el botón yes, si lo que quiere es borrar toda la base de datos y que quede solo ese nuevo usuario presione No, al presionar No le sistema le volverá a preguntar si está seguro en borrar toda la base de datos existe si presione yes procederá a borrarla y quedara solo el nuevo usuario registrado, si se arrepiente y no quiere borrar toda la base de datos haga click en No y el sistema mantendrá toda la base de datos y añadirá el nuevo usuario registrado, dichos mensajes se muestran en la figura A-16.

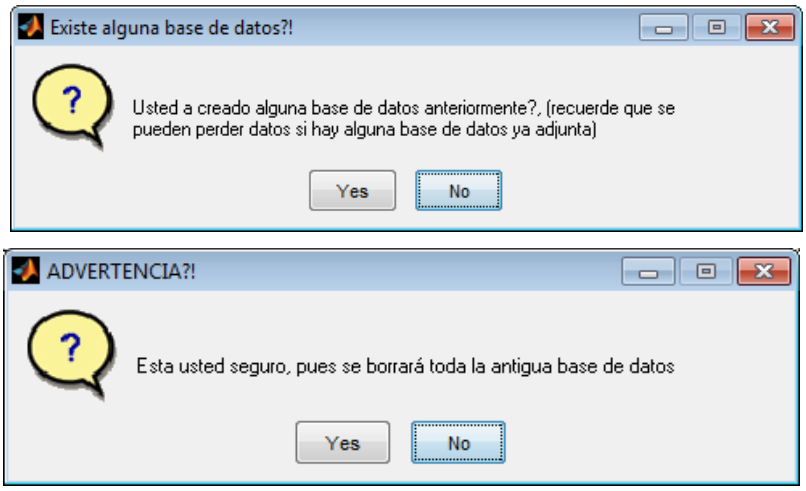


Figura A-16 Preguntas para registro de nuevo usuario.

Fuente: Elaboración propia.

Ya añadido el usuario al sistema haga click en el botón reconocimiento en el menú principal, se abrirá una ventana como se muestra en la figura A-17.

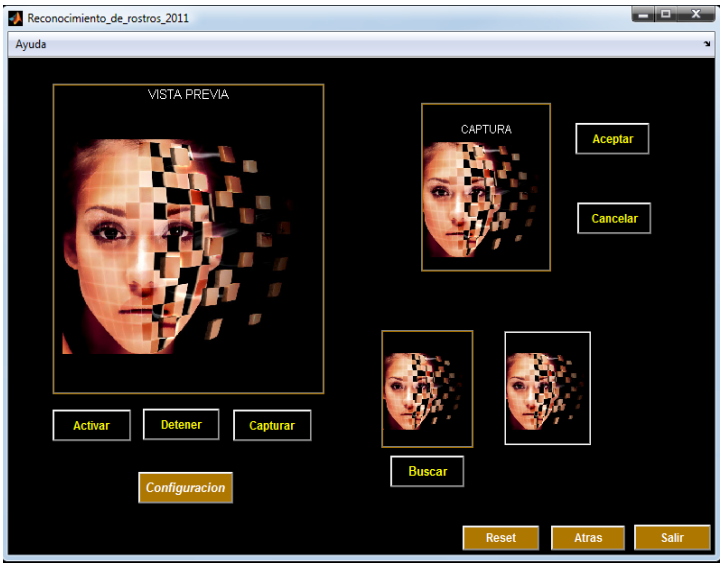


Figura A-17 Ventana de reconocimiento.

Fuente: Elaboración propia.

Luego proceda de la misma forma como en la ventana de registro, configure los parámetros de la cámara, active el canal, encienda el laser y capture la imagen como se observa en la figura A-18.

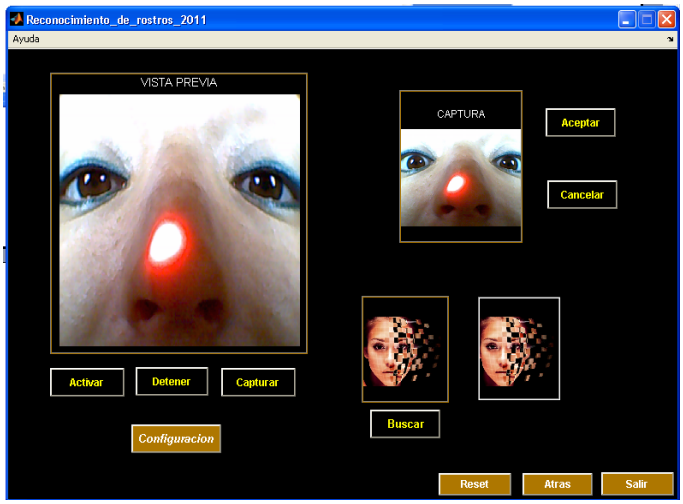


Figura A-18 Captura de la imagen de reconocimiento.

Fuente: Elaboración propia.

Haga click en botón aceptar para leer las distancias Dist1, Dist2 y Dist3. Proceda luego a presionar el botón Buscar el cual examinará el usuario en la base de datos y emitirá el posible candidato, si el usuario no se encuentra en la base datos se mostrará una imagen no disponle y candidato no encontrado. En la figura A-19 se muestra el candidato encontrado en el sistema de reconocimiento

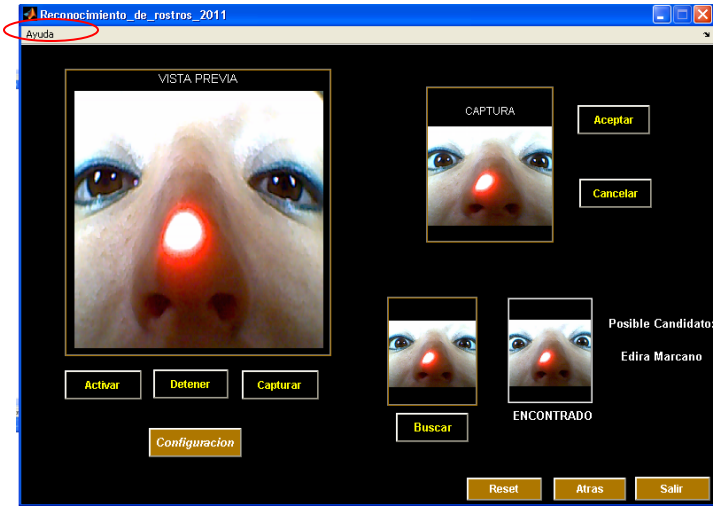


Figura A-19 Reconocimiento de rostro de un usuario en estudio.

Fuente: Elaboración propia.

Además, es importante señalar que cada ventana del programa contiene una pestaña en la parte superior izquierda con el nombre de ayuda.Al hacer clic sobre esta pestaña señala en rojo en la figura A-19, se ejecutara la ventana menú la cual contiene ofrece información básicas al operador del sistema, acerca de los elementos y parámetros del programa. El manejo de dicha ventana es muy sencillo, consiste en buscar en la lista la opción o ventana de la cual desea obtener información y hace clic en ella, inmediatamente se mostrará una breve descripción del comando.

Figura A-20 Ventana menú ayuda.

Fuente: Elaboración propio

APÉNDICE B

A continuación se muestra el código implementado en Matlab para este proyecto:

• Compensación_luz

```
function correct=compensacion_luz(im)
%% Se compensa la iluminacion usando el 5% de mayor luma si se
% encuentra cantidad suficiente de "blanco referencia" (mas del 1% del
% total).
% Argumentos de entrada:
%       - im: imagen en formato RGB
% Argumentos de salida:
%       - correct: imagen en formato RGB con la compensación de
% iluminación

%Transformación de la imagen al espacio YCbCr
imY=double(rgb2ycbcr(im));
[a,b,c]=size(im);
%Selección de la componente Y. Está en el rango [16 235]
luma=double(imY(:,:,1));
cont=0;
aux=0;
im=double(im);

% Corrección de gama (para calcular referencia)
for i=1:a,
    for j=1:b,
        if luma(i,j)>.95*(235-16),
                        cont=cont+1;
            aux=aux+luma(i,j);
        end
    end
end

ref=aux/cont;% valor medio píxeles blanco de referencia,nuevo valor de
% referencia

if cont>=.01*a*b,%reescalado de los canales.
    for i=1:a,
        for j=1:b,
            if imY(i,j,1)<ref,
                im(i,j,:)=im(i,j,:)*255/ref;
            else
```

```
                im(i,j,:)=255;
            end
        end
    end
end

correct=uint8(im);
```

• **Matriz de impresión de datos**

```
function arch = matriz(A)
global ban A
[M,sM]=size(A);
    cont=0;
    band=0;
for k=1:sM
    if A(1,k)==0
        cont= cont+1;
    end
end
if sM==cont;
      A=zeros(1,28);
      delete(gcf);
      band=1;
end;

    if band==0
   message = sprintf('Usted a creado alguna base de datos anteriormente?,
(recuerde que se pueden perder datos si hay alguna base de datos ya
adjunta)');
   reply = questdlg(message, 'Existe alguna base de datos?!','Yes','No','No');
if strcmpi(reply, 'No')
    message = sprintf('Esta usted seguro, pues se borrará toda la antigua base
de datos');
   reply = questdlg(message, 'ADVERTENCIA?!','Yes','No','No');
    if strcmpi(reply, 'Yes')
load('archivo.txt');
F=archivo;
borrar_fotos_carpetas (F);
f = fopen( 'archivo.txt', 'wt' );

% Empieza escritura de valores de la matriz
for m=1:M
    for o=1:sM
        fprintf( f, '%d\t ', A(m,o));
    end
    % Terminamos una fila, empieza nueva línea
    fprintf( f, '\n' );
```

```
end
% Termina escritura de valores de la matriz
ban=1;
fclose( f );
return
    end;
end
 if strcmpi(reply, 'No')
     f = fopen( 'archivo.txt', 'at' );
 for m=1:M
    for o=1:sM
        fprintf( f, '%d\t ', A(m,o));
    end
    % Terminamos una fila, empieza nueva línea
    fprintf( f, '\n' );
end
% Termina escritura de valores de la matriz
ban=1;
fclose( f );
   return;
 end
      if strcmpi(reply, 'Yes')
        f = fopen( 'archivo.txt', 'at' );
 for m=1:M
    for o=1:sM
        fprintf( f, '%d\t ', A(m,o));
    end
    % Terminamos una fila, empieza nueva línea
    fprintf( f, '\n' );
end
% Termina escritura de valores de la matriz
ban=1;
fclose( f );
      end;
    end;
```

• **Matriz de llenado de datos**

```
function matrizdatos (A)
global B c fil A
if fil~=1
    c=1;
else
 B=A;
 fil=1;
end
```

• **Verificación si un usuario existe en la base de datos**

```
function control (A,x)

[nfila,ncola]= size(A);
[nfilx,ncolx]= size(x);

for l=1:nfila
    cont=0;
for k=1:ncola
    if A(l,k)==0
        cont= cont+1;
    end
end
d= (ncola-cont);
if ncolx==d;
    con=0;
    for j=1:d
        if A(l,j)== x(j)
            con=con+1;
        end
    end;
    if con== d;
         message = sprintf('Ya existe en nuuestra base de datos un usuario con
ese nombre, Si no es es usted que ya ha sido registrado, por favor instroduzca
la inicial de su segundo nombre.');
      reply = questdlg(message, 'Nombre de usuario repetido!', 'OK', 'OK');
      if strcmpi(reply, 'OK')
    return;
      end
    end
end;
end;
end
```

• **Inicializa matriz una vez impresa**

```
function sra (nfil)
global c fil A
if fil~=1
    c=1;
else
 fil=nfil;
  A=zeros(1,28);
End
```

• **Buscar foto almacenada**

```
function buscar_foto (nombr)
files=dir('C:\Documents and
Settings\Administrador\Escritorio\tesis_2011\foto\');
   for i=3:length(files)
     n=files(i).name;
     [nombre,extension]=strtok(n,'.');
     if length(nombre)==length(nombr)
      con=0;
     for j=1:length(nombr)
        if nombre(j)== nombr(j)
            con=con+1;
        end
    end;
    if con== length(nombr)
         nombre_completo= strcat('C:\Documents and
Settings\Administrador\Escritorio\tesis_2011\foto\',n);
         axes(handles.axes5)
        axis off;
        imshow(nombre_completo);
     return;
    end;
     end;
   end;
```

• **Borra foto de un usuario registrado**

```
%UNIVERSISDAD DE CARABOBO
%FACULTAD DE INGENERÍA
%ESCUELA DE ELÉCTRICA
%DEPARTAMENTO DE SISTEMA Y AUTOMATICA
%PROYECTO DE GRADO
%AUTORES: ALEXANDER MORENO
%         JONYRIS LAMAS
% función creada para borrar foto almacenada en la carpeta de imagen
% dicha imagen se encuentra almacenada con el noombre mas la extencion

function borrar_fotos_almacenadas (w)
 files=dir('C:\Documents and
Settings\Administrador\Escritorio\tesis_2011\foto\'); %direccion donde se
encuentra la acrpeta
   for i=3:length(files)
     n=files(i).name;
     [nombre,extension]=strtok(n,'.'); % intrucción de matlab para separar el
nombre y la extención de la imagen

     if length(nombre)==length(w) % tamaño del noombre de la imagen comparado
con el nombre introducido en la GUIs
      con=0;
```

```
     for j=1:length(w)
         %Ciclo de comparacion
        if nombre(j)== w(j)
            con=con+1;% numeros de coincidencias entre el nombre de la imagen
y el nombre introducido desde la GUIs
        end
    end;
    if con== length(w)
        % si el numero de coincidencia y tamaño del nombre coinciden es
        % porque se encontro el usuario
        % se une el nombre con la direccion donde se encuentra
         nombre_completo= strcat('C:\Documents and
Settings\Administrador\Escritorio\tesis_2011\foto\',n);
         delete(nombre_completo); % borrar la imagen
     return;
    end;
     end;
   end;
```

• **Borra todas las fotos de los usuarios**

```
function borrar_fotos_carpetas(F)
[nfil,ncol]= size(F);
y=ncol-3;

for l=1:nfil
    cont=0;
    cont1=0;
    cont7=0;
for k=1:y
    if F(l,k)==0
        cont= cont+1;
    end
end
d= (y-cont);
for j=1:d
  R(l,j)= F(l,j);

end

for p=1:length(R)
    if R(l,p)==0
        cont1= cont1+1;
    end
end
h=length(R)-cont1;
for o=1:h
    R1(1,o)=R(l,o);
end
```

```
for p2=1:length(R1)
    if R1(1,p2)==0
        cont7= cont7+1;
    end
end
 gat1= length(R1);

if cont7~=0
    if gat1<=gat
z2=h+1;
    for z1=1:cont7
        R1(:,z2)= [];
    end;
    end
end
gat=length(R1);
w= char([R1(1,:)]);


files=dir('C:\Documents and
Settings\Administrador\Escritorio\tesis_2011\foto\');
   for i=3:length(files)
     n=files(i).name;
     [nombre,extension]=strtok(n,'.');
     if length(nombre)==length(w)
      con=0;
     for j=1:length(w)
        if nombre(j)== w(j)
            con=con+1;
        end
    end;
    if con== length(w)
         nombre_completo= strcat('C:\Documents and
Settings\Administrador\Escritorio\tesis_2011\foto\',n);
         delete(nombre_completo);
    end;
     end;
   end;
   R(l,:);
   R1(1,:)=[];
  end;
```

• Impresión en el archivo de datos borrado

```
function imprimir_borrado (u)
[M,sM]= size(u);
f = fopen( 'archivo.txt', 'wt' );

% Empieza escritura de valores de la matriz
```

```
for m=1:M
    for o=1:sM
        fprintf( f, '%d\t ', u(m,o));
    end
    % Terminamos una fila, empieza nueva línea
    fprintf( f, '\n' );
end
% Termina escritura de valores de la matriz

fclose( f );
```

• Intro clave de acceso para el registro

```
function [Password, UserName] = Contrasena(varargin)


%% Comprobamos que existe java
if ~usejava('swing')
   error('passwordEntryDialog: Java es requerido para que este programa se
ejecute.');
end

%% Argumentos de entrada
% Argumentos de entrada se analizan con la clase inputParer class

% Creamos objetos de entrada parser
Programopciones = inputParser;
% ProgramOptionsParser.KeepUnmatched = true;

Programopciones.addParamValue('Nombre', false, @(x) islogical(x) ||
isnumeric(x));
% Programopciones.addParamValue('DefaultUserName', '', @ischar);
Programopciones.addParamValue('ValidarContrasena', false, @(x) islogical(x) ||
isnumeric(x));
Programopciones.addParamValue('CheckPasswordLength', true, @(x) islogical(x)
|| isnumeric(x));
Programopciones.addParamValue('PasswordLengthMin', 1, @isnumeric);
% Programopciones.addParamValue('PasswordLengthMax', 8, @isnumeric);
Programopciones.addParamValue('WindowName', 'Administrador', @ischar);

Programopciones.parse(varargin{:});

Programopciones = Programopciones.Results;

%% Creamos taamaño y posicion de la interfaz grafica de Usuario
% Centramos la interfaz.
```

```
set(0,'Units','pixels')
Screen = get(0,'screensize');


PosicionGUI = [0 0 300 75];

if Programopciones.ValidarContrasena
    PosicionGUI = PosicionGUI + [0 0 0 50];
end;

PositionGUI = [1.6*Screen(3:4)/2-PosicionGUI(3:4)*4/2 PosicionGUI(3:4)];


%% CREAMOS LA INTERFAZ GRAFICA

BackgroundColor = get(0,'DefaultUicontrolBackgroundcolor');
handles.figure1 = figure('Menubar','none', ...
    'Units','Pixels', ...
    'Resize','off', ...
    'NumberTitle','off', ...
    'Name',Programopciones.WindowName, ...
    'Position',PositionGUI, ...
    'Color', BackgroundColor, ...
    'WindowStyle','modal');


% CREAR DIALOGO DE CONTRASEÑA
handles.java_Password = javax.swing.JPasswordField();
[handles.java_Password, handles.edit_Password] =
javacomponent(handles.java_Password, [22 7 130 30], handles.figure1);
handles.java_Password.setFocusable(true);
drawnow;

handles.text_ValidarContrasena = uicontrol('Parent',handles.figure1, ...
    'Tag', 'text_LabelPassword', ...
    'Style','Text', ...
    'Units','Pixels',...
    'Position',[32 45 200 16], ...
    'FontSize',10.5, ...
    'String','Clave de Acceso:',...
    'HorizontalAlignment', 'Left');

%  Crear Boton OK
handles.pushbutton_OK = uicontrol('Parent',handles.figure1, ...
    'Tag', 'pushbutton_OK', ...
    'Style','Pushbutton', ...
    'Units','Pixels',...
    'Position',[160 10 54 26], ...
    'FontSize',10.5, ...
    'String','Aceptar',...
```

```
    'HorizontalAlignment', 'Center');

% Crear Boton Cancelar
handles.pushbutton_Cancel = uicontrol('Parent',handles.figure1, ...
    'Tag', 'pushbutton_Cancel', ...
    'Style','Pushbutton', ...
    'Units','Pixels',...
    'Position',[220 10 64 26], ...
    'FontSize',10.5, ...
    'String','Cancelar',...
    'HorizontalAlignment', 'Center');


% % CUADRO DE ADICCION DE NOMBRE
if Programopciones.Nombre
    handles.java_UserName = javax.swing.JTextField();
    handles.java_UserName.setFocusable(true);
    [handles.java_UserName, handles.edit_UserName] =
javacomponent(handles.java_UserName, [], handles.figure1);

    handles.java_UserName.requestFocus;     % Get focus
    drawnow;
else
    handles.java_Password.requestFocus;     % Get focus
    drawnow;
end;
%% CONFIGURAMOS LLAMADAS PARA LOS OBJETOS

set(handles.pushbutton_OK,     'Callback', {@pushbutton_OK_Callback, handles,
Programopciones}, 'KeyPressFcn', {@pushbutton_KeyPressFcn, handles,
Programopciones});
set(handles.pushbutton_Cancel, 'Callback', {@pushbutton_Cancel_Callback,
handles, Programopciones}, 'KeyPressFcn', {@pushbutton_KeyPressFcn, handles,
Programopciones});
set(handles.java_Password, 'ActionPerformedCallback',
{@pushbutton_OK_Callback, handles, Programopciones});
setappdata(handles.figure1, 'isCanceled', false);

%% ESPERAMOS INFORMACION

%Espera a que el usuario complete la entrada
drawnow;
uiwait(handles.figure1);

%% OBTENEMOS INFORMACION AL PRESIONAL CANCEL O o.K

isCanceled = ~ishandle(handles.figure1) || getappdata(handles.figure1,
'isCanceled');
if isCanceled
    if ishandle(handles.figure1)
```

```
        delete(handles.figure1);
    end;
    UserName = '';
    menu_principal;
    return;
end;


if Programopciones.Nombre
    UserName = strtrim(get(handles.java_UserName, 'Text'));
else
    UserName = '';
end;
delete(handles.figure1);

%% DEDINIMOS LAS FUCIONES A USAR

function pushbutton_KeyPressFcn(hObject, eventdata, handles, Programopciones)

switch eventdata.Key
    case 'return'
        Callback = get(hObject, 'Callback');
        feval(Callback{1}, hObject, '', Callback{2:end});
end;

function pushbutton_OK_Callback(hObject, eventdata, handles, Programopciones)

%Configuramos lo que queremos obterner al pisar Aceptar
if Programopciones.CheckPasswordLength
    Password = handles.java_Password.Password';
    if length(Password) < Programopciones.PasswordLengthMin
        strMessage = sprintf('Instroduzca la clave, el campo Clave de Acceso
se encuentra vacio.');
        hError = errordlg(strMessage, 'Usuario');
        uiwait(hError);
        if Programopciones.ValidarContrasena
            set(handles.java_PasswordValidate,'Text', '');
        end;
        handles.java_Password.requestFocus
        return;
    end;
end;
Password = handles.java_Password.Password';
H=uint8(Password);
f=[76    97    98   114   111    98    50    48    49    49];

 [filcla,ncolcla]= size(f);
 [fille,ncolle]= size(H);
 contador=0;
 if ncolcla== ncolle
```

```
     for h=1:ncolcla
         if f(1,h)==H(1,h)
             contador= contador+1;
         end
     end
     if contador ~= ncolcla
         strMessage = sprintf('Clave incorrecta por favor introduzca la clave
para tener acceso al Registro.');
         hError = errordlg(strMessage, 'Usuario');
     return;
     end;
%      delete(Administrador);
     Registro;
 end;

  if ncolcla ~= ncolle
      strMessage = sprintf('Clave incorrecta por favor introduzca la clave
para tener acceso al Registro.');
         hError = errordlg(strMessage, 'Usuario');
     return;
   end;

uiresume(handles.figure1);

function pushbutton_Cancel_Callback(hObject, eventdata, handles,
Programopciones)
setappdata(handles.figure1, 'isCanceled', true);
uiresume(handles.figure1);
```

• Intro clave de acceso para el borrado

```
function [Password, UserName] = Contrasena(varargin)


%% Comprobamos que existe java
if ~usejava('swing')
   error('passwordEntryDialog: Java es requerido para que este programa se
ejecute.');
end

%% Argumentos de entrada
% Argumentos de entrada se analizan con la clase inputParer class

% Creamos objetos de entrada parser
Programopciones = inputParser;

Programopciones.addParamValue('Nombre', false, @(x) islogical(x) ||
isnumeric(x));
```

```
Programopciones.addParamValue('ValidarContrasena', false, @(x) islogical(x) ||
isnumeric(x));
Programopciones.addParamValue('CheckPasswordLength', true, @(x) islogical(x)
|| isnumeric(x));
Programopciones.addParamValue('PasswordLengthMin', 1, @isnumeric);
Programopciones.addParamValue('WindowName', 'Administrador', @ischar);

Programopciones.parse(varargin{:});
Programopciones = Programopciones.Results;

%% Creamos taamaño y posicion de la interfaz grafica de Usuario
% Centramos la interfaz.

set(0,'Units','pixels')
Screen = get(0,'screensize');


PosicionGUI = [0 0 300 75];

if Programopciones.ValidarContrasena
    PosicionGUI = PosicionGUI + [0 0 0 50];
end;

PositionGUI = [1.6*Screen(3:4)/2-PosicionGUI(3:4)*4/2 PosicionGUI(3:4)];


%% CREAMOS LA INTERFAZ GRAFICA

BackgroundColor = get(0,'DefaultUicontrolBackgroundcolor');
handles.figure1 = figure('Menubar','none', ...
    'Units','Pixels', ...
    'Resize','off', ...
    'NumberTitle','off', ...
    'Name',Programopciones.WindowName, ...
    'Position',PositionGUI, ...
    'Color', BackgroundColor, ...
    'WindowStyle','modal');


% CREAR DIALOGO DE CONTRASEÑA
handles.java_Password = javax.swing.JPasswordField();
[handles.java_Password, handles.edit_Password] =
javacomponent(handles.java_Password, [22 7 130 30], handles.figure1);
handles.java_Password.setFocusable(true);

drawnow;

handles.text_ValidarContrasena = uicontrol('Parent',handles.figure1, ...
    'Tag', 'text_LabelPassword', ...
```

```
    'Style','Text', ...
    'Units','Pixels',...
    'Position',[32 45 200 16], ...
    'FontSize',10.5, ...
    'String','Clave de Acceso:',...
    'HorizontalAlignment', 'Left');

%  Crear Boton OK
handles.pushbutton_OK = uicontrol('Parent',handles.figure1, ...
    'Tag', 'pushbutton_OK', ...
    'Style','Pushbutton', ...
    'Units','Pixels',...
    'Position',[160 10 54 26], ...
    'FontSize',10.5, ...
    'String','Aceptar',...
    'HorizontalAlignment', 'Center');

% Crear Boto Cancelar
handles.pushbutton_Cancel = uicontrol('Parent',handles.figure1, ...
    'Tag', 'pushbutton_Cancel', ...
    'Style','Pushbutton', ...
    'Units','Pixels',...
    'Position',[220 10 64 26], ...
    'FontSize',10.5, ...
    'String','Cancelar',...
    'HorizontalAlignment', 'Center');


% % CUADRO DE ADICCION DE NOMBRE DE USUARIO
if Programopciones.Nombre
    handles.java_UserName = javax.swing.JTextField();
    handles.java_UserName.setFocusable(true);
    [handles.java_UserName, handles.edit_UserName] =
javacomponent(handles.java_UserName, [], handles.figure1);

    handles.java_UserName.requestFocus;     %
else
    handles.java_Password.requestFocus;
    drawnow;
end;

%% CONFIGURAMOS LLAMADAS PARA LOS OBJETOS

set(handles.pushbutton_OK,     'Callback', {@pushbutton_OK_Callback, handles,
Programopciones}, 'KeyPressFcn', {@pushbutton_KeyPressFcn, handles,
Programopciones});
set(handles.pushbutton_Cancel, 'Callback', {@pushbutton_Cancel_Callback,
handles, Programopciones}, 'KeyPressFcn', {@pushbutton_KeyPressFcn, handles,
Programopciones});
```

```
set(handles.java_Password, 'ActionPerformedCallback',
{@pushbutton_OK_Callback, handles, Programopciones});
setappdata(handles.figure1, 'isCanceled', false);

%% ESPERAMOS INFORMACION

%Espera a que el usuario complete la entrada
drawnow;
uiwait(handles.figure1);

%% OBTENEMOS INFORMACION AL PRESIONAL CANCEL O o.K

isCanceled = ~ishandle(handles.figure1) || getappdata(handles.figure1,
'isCanceled');
if isCanceled
    if ishandle(handles.figure1)
        delete(handles.figure1);
    end;
    UserName = '';
    menu_principal;
    return;
end;


if Programopciones.Nombre
    UserName = strtrim(get(handles.java_UserName, 'Text'));
else
    UserName = '';
end;
delete(handles.figure1);

%% DEDINIMOS LAS FUCIONES A USAR


function pushbutton_KeyPressFcn(hObject, eventdata, handles, Programopciones)

switch eventdata.Key
    case 'return'
        Callback = get(hObject, 'Callback');
        feval(Callback{1}, hObject, '', Callback{2:end});
end;

function pushbutton_OK_Callback(hObject, eventdata, handles, Programopciones)

%Configuramos lo que queremos obterner al pisar Aceptar
if Programopciones.CheckPasswordLength

    Password = handles.java_Password.Password';
    if length(Password) < Programopciones.PasswordLengthMin
```

```
        strMessage = sprintf('Instroduzca la clave, el campo Clave de Acceso
se encuentra vacio.');
        hError = errordlg(strMessage, 'Usuario');
        uiwait(hError);
        handles.java_Password.requestFocus
        return;
    end;
end;
Password = handles.java_Password.Password';
H=uint8(Password);
f=[76    97    98   114   111    98    50    48    49    49];

 [filcla,ncolcla]= size(f);
 [fille,ncolle]= size(H);
 contador=0;
 if ncolcla== ncolle
     for h=1:ncolcla
         if f(1,h)==H(1,h)
             contador= contador+1;
         end
     end
     if contador ~= ncolcla
         strMessage = sprintf('Clave incorrecta por facor introduzca la clave
para tener acceso al Registro.');
         hError = errordlg(strMessage, 'Usuario');
     return;
     end;

     Borrar_registro;
 end;

  if ncolcla ~= ncolle
      strMessage = sprintf('Clave incorrecta por facor introduzca la clave
para tener acceso al Registro.');
         hError = errordlg(strMessage, 'Usuario');
     return;
   end;

uiresume(handles.figure1);

function pushbutton_Cancel_Callback(hObject, eventdata, handles,
Programopciones)
setappdata(handles.figure1, 'isCanceled', true);
uiresume(handles.figure1);
```

• **Interfaz grafica**

- **Presentación**

```
function varargout = Recono(varargin)
%PRESENTACION M-file for Presentacion.fig
%      PRESENTACION, by itself, creates a new PRESENTACION or raises the
existing
%      singleton*.
%
%      H = PRESENTACION returns the handle to a new PRESENTACION or the handle
to
%      the existing singleton*.
%
%      PRESENTACION('Property','Value',...) creates a new PRESENTACION using
the
%      given property value pairs. Unrecognized properties are passed via
%      varargin to Presentacion_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      PRESENTACION('CALLBACK') and PRESENTACION('CALLBACK',hObject,...) call
the
%      local function named CALLBACK in PRESENTACION.M with the given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Presentacion

% Last Modified by GUIDE v2.5 30-Mar-2011 23:59:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Presentacion_OpeningFcn, ...
                   'gui_OutputFcn',  @Presentacion_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


% --- Executes just before Presentacion is made visible.
function Presentacion_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

%-----Centramos la figura---------
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/16);
yr=scrsz(3) - pos_act(3);
yp=round(yr/50);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
% --------------------------------------
% Choose default command line output for Presentacion
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Presentacion wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Presentacion_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
%Incluir imagen
%Importamos imagen *.jpg,junto con su mapa de colores
varargout{1} = handles.output;
axes(handles.axes1)
background = imread('uc.jpg');
axis off;
imshow(background);

axes(handles.axes2)
background = imread('LOGO.jpg');
axis off;
```

```
imshow(background);

axes(handles.axes3)
background = imread('1.jpg');
axis off;
imshow(background);



%Título1
text(45,-240,'Reconocimiento de
Rostros','Fontname','Arial','Fontsize',22,'Fontangle','Italic','Fontweight','B
old','color','BLUE');

%Título2
text(260,-160,'Esta aplicacion fue creada por:
','Fontname','Arial','Fontsize',9,'Fontangle','Italic','Fontweight','Bold','co
lor','black');
text(250,-143,'Jonyris Lamas; Alexander Moreno
','Fontname','Arial','Fontsize',9,'Fontangle','Italic','Fontweight','Bold','co
lor','black');
text(240,-126,'cuya tutoria fue por: Prof. Wilmer
Sanz.','Fontname','Arial','Fontsize',9,'Fontangle','Italic','Fontweight','Bold
','color','black');

text(210,-85,'En el menu de reconocmiento se captura la
imagen','Fontname','Arial','Fontsize',9,'Fontangle','Italic','Fontweight','Bol
d','color','black');
text(210,-68,'original, pasando luego a una etapa de procesado
','Fontname','Arial','Fontsize',9,'Fontangle','Italic','Fontweight','Bold','co
lor','black');
text(230,-51,'donde se aplican tecnicas de
segmentación,','Fontname','Arial','Fontsize',9,'Fontangle','Italic','Fontweigh
t','Bold','color','black');
text(210,-34,'para finalmente comparar con una base de
datos','Fontname','Arial','Fontsize',9,'Fontangle','Italic','Fontweight','Bold
','color','black');
text(240,-17,'arrojando así el candidato
encontrado.','Fontname','Arial','Fontsize',9,'Fontangle','Italic','Fontweight'
,'Bold','color','black');
%trabajo especial
text(225,24,'TRABAJO ESPECIAL DE GRADO
PRESENTADO','Fontname','Arial','Fontsize',9,'Fontangle','Italic','Fontweight',
'Bold','color','black');
text(220,40,'ANTE LA ILUSTRE UNIVERSIDAD DE
CARABOBO','Fontname','Arial','Fontsize',9,'Fontangle','Italic','Fontweight','B
old','color','black');
text(210,56,'PARA OPTAR AL TITULO INGENIERO
ELECTRICISTA','Fontname','Arial','Fontsize',9,'Fontangle','Italic','Fontweight
','Bold','color','black');
```

```
%Título4
text(12,17,'Universidad de
Carabobo','Fontname','Arial','Fontsize',11,'Fontangle','Italic','Fontweight','
Bold','color','black');
text(19,33,'Facultad de
Ingeniería','Fontname','Arial','Fontsize',11,'Fontangle','Italic','Fontweight'
,'Bold','color','black');



%Botón Continuar
 botok=uicontrol('Style','pushbutton','Units','normalized','Position',[.44 .07
.12 .07],'String','CONTINUAR',...
 'Callback','clear all; close all;clc; menu_principal;'); %GUI es el nombre
del siguiente programa.
```

- **Menu_principal**

```
%UNIVERSISDAD DE CARABOBO
%FACULTAD DE INGENERÍA
%ESCUELA DE ELÉCTRICA
%DEPARTAMENTO DE SISTEMA Y AUTOMATICA
%PROYECTO DE GRADO
%AUTORES: ALEXANDER MORENO
%         JONYRIS LAMAS

%INTERFAZ GRÁFICA: MENÚ PRINCIPAL
% Esta interfaz es un panel de tareas


function varargout = menu_principal(varargin)
% MENU_PRINCIPAL M-file for menu_principal.fig
%      MENU_PRINCIPAL, by itself, creates a new MENU_PRINCIPAL or raises the
existing
%      singleton*.
%
%      H = MENU_PRINCIPAL returns the handle to a new MENU_PRINCIPAL or the
handle to
%      the existing singleton*.
%
%      MENU_PRINCIPAL('CALLBACK',hObject,eventData,handles,...) calls the
local
```

```
%      function named CALLBACK in MENU_PRINCIPAL.M with the given input
arguments.
%
%      MENU_PRINCIPAL('Property','Value',...) creates a new MENU_PRINCIPAL or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before menu_principal_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to menu_principal_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help menu_principal

% Last Modified by GUIDE v2.5 09-May-2011 16:26:39

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @menu_principal_OpeningFcn, ...
                   'gui_OutputFcn',  @menu_principal_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before menu_principal is made visible.
function menu_principal_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to menu_principal (see VARARGIN)
% Choose default command line output for menu_principal
%-----Centramos la figura----------------------------------
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
```

```
xp=round(xr/13);
yr=scrsz(3) - pos_act(3);
yp=round(yr/50);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
% -------Fondo de la interfaz -------------------------
axes1 = imread('carai.jpg');
axis off;
imshow(axes1);
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes menu_principal wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = menu_principal_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%----- se ejecuta cuando se presiona el botón reconocimiento
% se abre la siguiente interfaz gráfica reconocimiento de rostro 2011
% cierra la ventana actual.
function pushbutton4_Callback(hObject, eventdata, handles)
global centinela2 centinela1
centinela2=1;
centinela1=6;
delete(gcf);
Reconocimiento_de_rostros_2011;


%----- se ejecuta cuando se presiona el botón nuevo registro
% se abre la siguiente interfaz gráfica intro para colocar el contraseña
% cierra la ventana actual.
function pushbutton5_Callback(hObject, eventdata, handles)
global banderin centinela4 centinela3
banderin=2;
centinela4=1;
centinela3=6;
 delete(gcf);
intro;


%----- se ejecuta cuando se presiona el botón consultar
```

```
% se abre la siguiente interfaz gráfica consultar registro
%cierra la ventana actual.
function pushbutton6_Callback(hObject, eventdata, handles)
delete(gcf);
Consultar_registro;


%----- se ejecuta cuando se presiona el botón borrar registro
% se abre la siguiente interfaz gráfica intro_borrar para colocar la clave
% cierra la ventana menú principal
function pushbutton7_Callback(hObject, eventdata, handles)
delete(gcf);
intro_borrar


%----- se ejecuta cuando se presiona el botón salir
% cierra la ventana actual y sale del programa
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(gcf);

% fin de la programación de la ventana menú principal
```

- **Reconocimiento**

```
%UNIVERSISDAD DE CARABOBO
%FACULTAD DE INGENERÍA
%ESCUELA DE ELÉCTRICA
%DEPARTAMENTO DE SISTEMA Y AUTOMATICA
%PROYECTO DE GRADO
%AUTORES: ALEXANDER MORENO
%         JONYRIS LAMAS

%INTERFAZ GRÁFICA:RECONOCIMIENTO DE ROSTRO 2011
%  En ella se reconoce el rostro de una persona mediante captura de imagen
%  luego se extraen los puntos caracteristico de dicho rostro
%  para compararlos con los guardados en la base de datos del sistema



function varargout = Reconocimiento_de_rostros_2011(varargin)
% RECONOCIMIENTO_DE_ROSTROS_2011 M-file for Reconocimiento_de_rostros_2011.fig
%      RECONOCIMIENTO_DE_ROSTROS_2011, by itself, creates a new
RECONOCIMIENTO_DE_ROSTROS_2011 or raises the existing
```

```
%      singleton*.
%
%      H = RECONOCIMIENTO_DE_ROSTROS_2011 returns the handle to a new
RECONOCIMIENTO_DE_ROSTROS_2011 or the handle to
%      the existing singleton*.
%
%
RECONOCIMIENTO_DE_ROSTROS_2011('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in RECONOCIMIENTO_DE_ROSTROS_2011.M with the
given input arguments.
%
%      RECONOCIMIENTO_DE_ROSTROS_2011('Property','Value',...) creates a new
RECONOCIMIENTO_DE_ROSTROS_2011 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Reconocimiento_de_rostros_2011_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to
Reconocimiento_de_rostros_2011_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Reconocimiento_de_rostros_2011

% Last Modified by GUIDE v2.5 29-Mar-2011 17:29:42

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Reconocimiento_de_rostros_2011_OpeningFcn, ...
                   'gui_OutputFcn',
@Reconocimiento_de_rostros_2011_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
%% Frame ID


% --- Executes just before Reconocimiento_de_rostros_2011 is made visible.
function Reconocimiento_de_rostros_2011_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Reconocimiento_de_rostros_2011 (see
VARARGIN)

%-----Centramos la figura---------
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/19);
yr=scrsz(3) - pos_act(3);
yp=round(yr/75);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
% --------Fondo de la imagen ---------------------
axes2 = imread('carai.jpg');
axis off;
imshow(axes2);
pp = imread('carai.jpg');
axes(handles.axes3);
axis off;
imshow(pp);
pp1 = imread('carai.jpg');
axes(handles.axes4);
axis off;
imshow(pp1);
pp2 = imread('carai.jpg');
axes(handles.axes5);
axis off;
imshow(pp2);
% Choose default command line output for Reconocimiento_de_rostros_2011
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% UIWAIT makes Reconocimiento_de_rostros_2011 wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
```

```
function varargout = Reconocimiento_de_rostros_2011_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
varargout{1} = handles.output;



% --- Se ejecuta al presionar el botón Activar desde la interfaz ----
% se abre el canal de video
function Activar_Callback(hObject, eventdata, handles)
global q w
vid = videoinput('winvideo',w,q);
vidRes = get(vid, 'VideoResolution');
nBands = get(vid, 'NumberOfBands');
hImage = image(zeros(vidRes(2),vidRes(1),nBands), 'parent', handles.axes2);
preview(vid, hImage);
handles.video=vid;
handles.band=0;
guidata(hObject,handles);

% --- se ejecuta al presionar el botón Detener desde la interfaz ---
function Detener_Callback(hObject, eventdata, handles)
Ce=handles.video;
closepreview(Ce);
delete(Ce);


% --- se ejecuta al presionar el botón Capturar GUIs ---
% se captura una imagen (en formato RGB) del canal de video
function Capturar_Callback(hObject, eventdata, handles)
global centinela1 centinela2
% hObject    handle to Capturar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Cap= handles.video;
imgAdq = getsnapshot(Cap);
axes(handles.axes3)
% axis off;
centinela1= 1;
centinela2=2;
imshow(imgAdq);
handles.imagen=imgAdq;
handles.band=1;
guidata(hObject,handles)
```

```
% --- Executes on button press in Configuracion.
function Configuracion_Callback(hObject, eventdata, handles)
% hObject    handle to Configuracion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Configuracion



% --- Executes on button press in Cancelar.
function Cancelar_Callback(hObject, eventdata, handles)
global centinela1 centinela2
centinela2=1;
centinela1=0;
% hObject    handle to Cancelar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pp = imread('carai.jpg');
axes(handles.axes3);
axis off;
imshow(pp);


% --- Executes on button press in Aceptar.
function Aceptar_Callback(hObject, eventdata, handles)
global distm1 distm2 distm3 centinela1
if centinela1==1
    centinela1=0;

%busqueda de centro de los ojos
% RGB = imread('C:\Documents and Settings\Administrador\Escritorio\fafa.jpg')
RGB = handles.imagen;
img_compensada1= compensacion_luz(RGB);
I = rgb2gray(img_compensada1);
threshold = 0.225;
aw = im2bw(I,threshold);
bw= not(aw); %negamos la imagen binarizada

% % %remove all object containing fewer than 30 pixels
bw = bwareaopen(bw,20);
%
% %fill a gap in the pen's cap
se = strel('disk',7);
bw = imclose(bw,se);
bw = imerode(bw,se);

%Pasamos la parte superior de la imagen, alli se encuentran los ojos
```

```
% bw = bw(1:size(bw, 1)/2, 1:size(bw, 2))

   s  = regionprops(bw, 'centroid');%centro de todos los posibles ojos
centroids = cat(1, s.Centroid);
[tx,ty] = size(bw);
ty=ty/2;
n=length(s);
menor=ty;
mayor=ty;
desc=tx-100;
for i=1:n
 centro = s(i).Centroid; %centro del ojo i
 x = centro(1);
 y = centro(2);
 if desc>y
 if x<ty
     posiblecent= ty-x;
     if posiblecent<menor
         menor= posiblecent;
         c=i;
     end
 else
     posiblecent= x-ty;
      if posiblecent<mayor
         mayor= posiblecent;
         d=i;
      end
 end
 end
end
%
  centro = s(c).Centroid; %centro del ojo derecho
  xcd = centro(1);
  ycd = centro(2);

  centro = s(d).Centroid; %centro del ojo izquierdo
  xci = centro(1);
  yci = centro(2);


%Busqueda del centro del laser
I = rgb2gray(RGB);
j=im2bw(I,0.99);
bw = bwareaopen(j,20);
%
% %fill a gap in the pen's cap
se = strel('disk',10);
bw = imclose(bw,se);
bw = imerode(bw,se);

%Busqueda de centros
```

```
s  = regionprops(bw, 'centroid');%centro del posible laser
centroids = cat(1, s.Centroid);
n=length(s);
mayor=0;
laser=0;
for i=1:n
 centro = s(i).Centroid; %centro en estudio
 x = centro(1);
 y = centro(2);
  [tx,ty]= size (I);
  jo=0.85*tx/2;
jo1=4.5*ty/16;
jo2=9.5*ty/16;
 if y>jo
     if x>jo1 && x<jo2
        if y>mayor
            laser=i;
            mayor=y;
        end
     end

 end
end
if laser==0
    strMessage = sprintf('No se encuentra el laser, verifique que este
encendido y capture de nuevo la imagen.');
         hError = errordlg(strMessage, 'Usuario');
         return;
end

centro = s(laser).Centroid; %centro del laser
  xlaser = centro(1);
  ylaser = centro(2);

%distancia del centro del punto al ojo1
distm1=sqrt(((xcd-xlaser)^2)+((ycd-ylaser)^2));

%distancia del centro del punto al ojo2
distm2=sqrt(((xci-xlaser)^2)+((yci-ylaser)^2));
%distancia del centro del ojo1 al ojo2
distm3=sqrt(((xcd-xci)^2)+((ycd-yci)^2));

imag= handles.imagen;
axes(handles.axes4)
axis off;
imshow(imag);
else
     helpdlg('Debe capturar una imagen o preciono mas de una vez el boton
aceptar.','AVISO!');
     return;
end
```

```
% --- Executes on button press in Buscar.
function Buscar_Callback(hObject, eventdata, handles)
% hObject    handle to Buscar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global distm1 distm2 distm3 centinela1 centinela2
 if centinela1==0 && centinela2==2

load('archivo.txt');
F=archivo;
[n,m]=size(F);
candidato=0;
pcandidato=0;
 for j=1:n
     v26=F(j,26);
     v27=F(j,27);
      v28=F(j,28);
       if distm1>(v26-30) && distm1<(v26+30)
            if distm2>(v27-30) && distm2<(v27+30)
                 if distm3>(v28-20) && distm3<(v28+20)
                     pcandidato=j;
                     if abs(distm3-v28)<15
                         candidato=pcandidato
                     end

                 end
            end
       end
 end;
 if pcandidato==0
     enc='NO ENCONTRADO';
     nombr= 'Sin Resultados';
     posib= 'Posible Candidato:';
     set(handles.text8,'String',posib);
      set(handles.text1,'String',enc);
      nombre_complet= strcat('C:\Documents and
Settings\Administrador\Escritorio\tesis_2011\no_disponible.jpg');
    axes(handles.axes5);
    axis off;
    imshow(nombre_complet);
    set(handles.text4, 'string', nombr);
    return
 end;
 if candidato==0
     candidato=pcandidato;
 end
y=m-3;
cont=0;
for k=1:y
    if F(candidato,k)==0
```

```
        cont= cont+1;
    end
end
d= (y-cont);
    for ye=1:d
       vecnom(ye)= F(candidato,ye);
    end;
  nombr= char([vecnom]);
  files=dir('C:\Documents and
Settings\Administrador\Escritorio\tesis_2011\foto\');
   for i=3:length(files)
     n=files(i).name;
     [nombre,extension]=strtok(n,'.');
     if length(nombre)==length(nombr)
      con=0;
     for j=1:length(nombr)
        if nombre(j)== nombr(j)
            con=con+1;
        end
    end;
    if con== length(nombr)
         nombre_completo= strcat('C:\Documents and
Settings\Administrador\Escritorio\tesis_2011\foto\',n);
        axes(handles.axes5);
         axis off;
        imshow(nombre_completo);
    end;
     end;
   end;
  enc= 'ENCONTRADO';
  posib= 'Posible Candidato:';
  set(handles.text4, 'string', nombr);
  set(handles.text1,'String',enc);
  set(handles.text8,'String',posib);
 else
   helpdlg('Debe aceptar la imagen capturada para su
reconocimiento.','AVISO!');
     return;
end
```

```
% --- Executes on button press in Salir.
function Salir_Callback(hObject, eventdata, handles)
global centinela2 centinela1
centinela2=1;
centinela1=6;
message = sprintf('Desea salir del Sistema de Reconocimiento?');
reply = questdlg(message, 'Salir', 'OK','Cancel', 'OK');
if strcmpi(reply, 'Cancel')
```

```
    return;
end
delete(gcf);

function Configuracion

%% Parametros
%% Configuracion
global Frames Video Objects Settings

%% Settings
Settings.DeviceID       = 'RGB24_640x480';
Settings.Adaptor        = 1;

global Video Settings Objects

str(1) = {Settings.DeviceID};
str(2) = {num2str(Settings.Adaptor)};



%% marco config
answer =
inputdlg({'Parametros_del_Dispositivo','Adaptador'},'Configuracion',[1,32;
1,32],str);

% Actualizo configuracion
Config.Parametros_del_Dispositivo= char(answer(1));
Config.Adaptador= str2double(char(answer(2)));


%% Guardo mis nuevos parametros
global q w
 q= nombre(Config.Parametros_del_Dispositivo);
 w= nombre(Config.Adaptador);


% --- Executes on button press in Atras.
function Atras_Callback(hObject, eventdata, handles)
global centinela2
centinela2=1;
delete(gcf);
menu_principal;

function t= nombre(A)
t= A;

% --- Executes on button press in Reset.
function Reset_Callback(hObject, eventdata, handles)
```

```
global centinela2
centinela2=1;
delete(gcf);
Reconocimiento_de_rostros_2011;
```

- **Registro**

```
function varargout = Registro(varargin)
% REGISTRO M-file for Registro.fig
%      REGISTRO, by itself, creates a new REGISTRO or raises the existing
%      singleton*.
%
%      H = REGISTRO returns the handle to a new REGISTRO or the handle to
%      the existing singleton*.
%
%      REGISTRO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in REGISTRO.M with the given input arguments.
%
%      REGISTRO('Property','Value',...) creates a new REGISTRO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Registro_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Registro_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Registro

% Last Modified by GUIDE v2.5 08-Apr-2011 20:47:26

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Registro_OpeningFcn, ...
                   'gui_OutputFcn',  @Registro_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end
% End initialization code - DO NOT EDIT
%% Frame ID


% --- Executes just before Registro is made visible.
function Registro_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Registro (see VARARGIN)


%-----Centramos la figura---------
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/18);
yr=scrsz(3) - pos_act(3);
yp=round(yr/74);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
% --------------- fondo de la interfaz ---------------------

axes2 = imread('carai.jpg');
axis off;
imshow(axes2);
axes3 = imread('carai.jpg');
axes(handles.axes3);
axis off;
imshow(axes3);
axes4= imread('carai.jpg');
axes(handles.axes4);
imshow(axes4);
axis off;
% Choose default command line output for Registro
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% UIWAIT makes Registro wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Registro_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
varargout{1} = handles.output;
% passwordEntryDialog



% --- Executes on button press in Activar.
function Activar_Callback(hObject, eventdata, handles)
 global q w
vid = videoinput('winvideo',w,q);
vidRes = get(vid, 'VideoResolution');
nBands = get(vid, 'NumberOfBands');
hImage = image(zeros(vidRes(2),vidRes(1),nBands), 'parent', handles.axes2);
preview(vid, hImage);
handles.video=vid;
handles.band=0;
guidata(hObject,handles);


% --- Executes on button press in Detener.
function Detener_Callback(hObject, eventdata, handles)
% hObject    handle to Detener (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Ce=handles.video;
closepreview(Ce)
delete(Ce)


% --- Executes on button press in Capturar.
function Capturar_Callback(hObject, eventdata, handles)
global centinela3 centinela4
centinela3= 1;
centinela4=2;
% hObject    handle to Capturar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Cap= handles.video;
imgAdq = getsnapshot(Cap);
axes(handles.axes3)
% axis off;
imshow(imgAdq);
handles.imagen=imgAdq;
handles.band=1;
guidata(hObject,handles)
```

```
% --- Executes on button press in Configuracion.
function Configuracion_Callback(hObject, eventdata, handles)
% hObject    handle to Configuracion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Configuracion



% --- Executes on button press in Cancelar.
function Cancelar_Callback(hObject, eventdata, handles)
% hObject    handle to Cancelar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global centinela3 centinela4
centinela4=1;
centinela3=0;
axes3 = imread('carai.jpg');
axes(handles.axes3);
axis off;
imshow(axes3);



% --- Executes on button press in Aceptar.
function Aceptar_Callback(hObject, eventdata, handles)
global dist1 dist2 dist3 banderin centinela3
if centinela3==1
    centinela3=0;

% imag= handles.imagen;
% axes(handles.axes4)
% axis off;
% imshow(imag);

%busqueda de centro de los ojos
% RGB = imread('C:\Documents and Settings\Administrador\Escritorio\pf2.jpg')
RGB = handles.imagen;
img_compensada1= compensacion_luz(RGB);
I = rgb2gray(img_compensada1);
threshold = 0.225;
aw = im2bw(I,threshold);
bw= not(aw); %negamos la imagen binarizada

% % %remove all object containing fewer than 30 pixels
bw = bwareaopen(bw,20);
%
% %fill a gap in the pen's cap
se = strel('disk',7);
```

```
bw = imclose(bw,se);
bw = imerode(bw,se);

%Pasamos la parte superior de la imagen, alli se encuentran los ojos
% bw = bw(1:size(bw, 1)/2, 1:size(bw, 2))

   s  = regionprops(bw, 'centroid');%centro de todos los posibles ojos
centroids = cat(1, s.Centroid);
[tx,ty] = size(bw);
ty=ty/2;
n=length(s);
menor=ty;
mayor=ty;
desc=tx-100;
for i=1:n
 centro = s(i).Centroid; %centro del ojo i
 x = centro(1);
 y = centro(2);
 if desc>y
 if x<ty
     posiblecent= ty-x;
     if posiblecent<menor
         menor= posiblecent;
         c=i;
     end
 else
     posiblecent= x-ty;
      if posiblecent<mayor
         mayor= posiblecent;
         d=i;
      end
 end
 end
end

  centro = s(c).Centroid; %centro del ojo derecho
  xcd = centro(1);
  ycd = centro(2);

  centro = s(d).Centroid; %centro del ojo izquierdo
  xci = centro(1);
  yci = centro(2);


%Busqueda del centro del laser
I = rgb2gray(RGB);
j=im2bw(I,0.99);
bw = bwareaopen(j,20);
%
% %fill a gap in the pen's cap
se = strel('disk',10);
```

```
bw = imclose(bw,se);
bw = imerode(bw,se);

%Busqueda de centros
s  = regionprops(bw, 'centroid');%centro del posible laser
centroids = cat(1, s.Centroid);
n=length(s);
mayor=0;
laser=0;
for i=1:n
 centro = s(i).Centroid; %centro en estudio
 x = centro(1);
 y = centro(2);
  [tx,ty]= size (I);
  jo=0.85*tx/2;
jo1=4.5*ty/16;
jo2=9.5*ty/16;
 if y>jo
     if x>jo1 && x<jo2
        if y>mayor
            laser=i;
            mayor=y;
        end
     end

 end
end
if laser==0
    strMessage = sprintf('No se encuentra el laser, verifique que este
encendido y capture de nuevo la imagen.');
         hError = errordlg(strMessage, 'Usuario');
         return;
end;

centro = s(laser).Centroid; %centro del laser
  xlaser = centro(1);
  ylaser = centro(2);

%distancia del centro del punto al ojo1
dist1=sqrt(((xcd-xlaser)^2)+((ycd-ylaser)^2));

%distancia del centro del punto al ojo2
dist2=sqrt(((xci-xlaser)^2)+((yci-ylaser)^2));
%distancia del centro del ojo1 al ojo2
dist3=sqrt(((xcd-xci)^2)+((ycd-yci)^2));
banderin=0;
set(handles.text8, 'string', dist1);
set(handles.text9, 'string', dist2);
set(handles.text10, 'string', dist3);

imag= handles.imagen;
```

```
axes(handles.axes4)
axis off;
imshow(imag);
else
     helpdlg('Debe capturar una imagen o preciono mas de una vez el boton
aceptar.','AVISO!');
     return;
end



% handles.imagen=imgAdq;
% handles.band=1;
% guidata(hObject,handles)
% hObject    handle to Aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in Guardar.
function Guardar_Callback(hObject, eventdata, handles)
global centinela3 centinela4
if centinela3==0 && centinela4==2
% lee nombre de usuario
 nomb=get(handles.entrada,'String');
 w= nombre(nomb);
 x= uint8(w);
 [nfil,ncol]= size(x);

 if ncol==0
      helpdlg('Instroduzca su nombre, el campo Nombre se encuentra
vacio.','ERROR!');
     return;
 end

 if x(1)==32
      helpdlg('Verifique porque su nombre empieza con espacio, en el campo
Nombre.','ERROR!');
     return;
 end
 for po=1:ncol
 if x(po)==46
      errordlg('Su nombre No puede incluir el caracter punto (.) ','ERROR!');
     return;
 end
 end

 if ncol>25
      helpdlg('Su nombre excede la cantidad de caracteres requerido, El campo
Nombre acepta (25 caracteres maximo).','ERROR!');
```

```
     return;
 end
  sra (nfil);
  global c fil A ban dist1 dist2 dist3 banderin
    if banderin~= 0
    helpdlg('No se han podido cargar las distancias Dist1, Dist2 y Dist3. Por
favor verifique si acepto la captura de la imagen.','ERROR!');
    return
    end
  if fil~=1
      [nfila,ncola]= size(A);
      [nfilx,ncolx]= size(x);

for l=1:nfila
    cont=0;
for k=1:(ncola-3)
    if A(l,k)==0
        cont= cont+1;
    end
end
d= (ncola-cont-3);
if ncolx==d;
    con=0;
    for j=1:d
        if A(l,j)== x(j)
            con=con+1;
        end
    end;
    if con== d;
         helpdlg('Ya existe en nuestra base de datos un usuario con ese
nombre, Si no es usted que ya fue registrado?, por favor instroduzca la
inicial de su segundo nombre.','Nombre de usuario repetido!');
     return;

    end
end;
end;
  end;
  if ban==1
      load('archivo.txt');
B= archivo;

      [nfila,ncola]= size(B);
      [nfilx,ncolx]= size(x);

for l=1:nfila
    cont=0;
for k=1:(ncola-3)
    if B(l,k)==0
        cont= cont+1;
    end
```

```
end
d= (ncola-cont-3);
if ncolx==d;
    con=0;
    for j=1:d
        if B(l,j)== x(j)
            con=con+1;
        end
    end;
    if con== d;
         helpdlg('Ya existe en nuestra base de datos un usuario con ese
nombre, Si no es usted que ya fue registrado?, por favor instroduzca la
inicial de su segundo nombre.','Nombre de usuario repetido!');
     return;

    end
end;
end;
  end;
  for i=1:ncol
   A(fil,i)= x(i);
  end
  A(fil,26)=dist1;
  A(fil,27)=dist2;
  A(fil,28)=dist3;
  helpdlg('Usted ha sido registrado exitosamente, Recuerde su nombre de
registro para futuras consultas.','Dialogo!');
  fil= fil+1;
  im=handles.imagen;
  guardar_imagen(w,im);

   else
   helpdlg('Debe aceptar la imagen capturada para su
reconocimiento.','AVISO!');
     return;
end
```

```
% --- Executes on button press in Salir.
function Salir_Callback(hObject, eventdata, handles)
global centinela4 centinela3
centinela4=1;
centinela3=6;
message = sprintf('Desea salir del Sistema de Registro?');
reply = questdlg(message, 'Salir', 'OK','Cancel', 'OK');
if strcmpi(reply, 'Cancel')
    return;
```

```
end
global c fil A
c=0;
fil=1;
matriz(A);
A=zeros(1,28);
delete(gcf);
menu_principal;

function Configuracion

%% Parametros
%% Configuracion
global Frames Video Objects Settings

%% Settings
Settings.DeviceID       = 'RGB24_640x480';
Settings.Adaptor        = 1;

global Video Settings Objects

str(1) = {Settings.DeviceID};
str(2) = {num2str(Settings.Adaptor)};



%% marco config
answer =
inputdlg({'Parametros_del_Dispositivo','Adaptador'},'Configuracion',[1,32;
1,32],str);

% Actualizo configuracion
Config.Parametros_del_Dispositivo= char(answer(1));
Config.Adaptador= str2double(char(answer(2)));


%% Guardo mis nuevos parametros
global q w
 q= nombre(Config.Parametros_del_Dispositivo);
 w= nombre(Config.Adaptador);

function t= nombre(A)
t= A;

function guardar_imagen (w,im)
total=strcat('C:\Documents and
Settings\Administrador\Escritorio\tesis_2011\foto\',w);
total1=strcat('.jpg');
direccion= strcat(total,total1);
```

```
imwrite(im, direccion);

function entrada_Callback(hObject, eventdata, handles)
% hObject    handle to entrada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of entrada as text
%        str2double(get(hObject,'String')) returns contents of entrada as a
double


% --- Executes during object creation, after setting all properties.
function entrada_CreateFcn(hObject, eventdata, handles)
% hObject    handle to entrada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function clave_Callback(hObject,data, eventdata, handles)
% hObject    handle to clave (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of clave as text
%        str2double(get(hObject,'String')) returns contents of clave as a
double

% --- Executes during object creation, after setting all properties.
function clave_CreateFcn(hObject, eventdata, handles)
% hObject    handle to clave (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

• **Consultar registro**

```
function varargout = Consultar_registro(varargin)
% CONSULTAR_REGISTRO M-file for Consultar_registro.fig
%      CONSULTAR_REGISTRO, by itself, creates a new CONSULTAR_REGISTRO or
raises the existing
%      singleton*.
%
%      H = CONSULTAR_REGISTRO returns the handle to a new CONSULTAR_REGISTRO
or the handle to
%      the existing singleton*.
%
%      CONSULTAR_REGISTRO('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in CONSULTAR_REGISTRO.M with the given input
arguments.
%
%      CONSULTAR_REGISTRO('Property','Value',...) creates a new
CONSULTAR_REGISTRO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Consultar_registro_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Consultar_registro_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Consultar_registro

% Last Modified by GUIDE v2.5 31-Mar-2011 17:25:12

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Consultar_registro_OpeningFcn, ...
                   'gui_OutputFcn',  @Consultar_registro_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Consultar_registro is made visible.
function Consultar_registro_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Consultar_registro (see VARARGIN)
%-----Centramos la figura---------
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/13);
yr=scrsz(3) - pos_act(3);
yp=round(yr/52);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
% ------------------fondo de la imagen--------------------
axes1 = imread('carai.jpg');
axis off;
imshow(axes1);

% Choose default command line output for Consultar_registro
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Consultar_registro wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Consultar_registro_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function entrada_Callback(hObject, eventdata, handles)
% hObject    handle to entrada (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of entrada as text
%        str2double(get(hObject,'String')) returns contents of entrada as a
double


% --- Executes during object creation, after setting all properties.
function entrada_CreateFcn(hObject, eventdata, handles)
% hObject    handle to entrada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in consulta.
function consulta_Callback(hObject, eventdata, handles)
nombret=get(handles.entrada,'String');
w= nombre(nombret);
x= uint8(w);
[nfilx,ncolx]= size(x);

if ncolx==0
      helpdlg('Instroduzca su nombre, el campo Nombre se encuentra
vacio.','ERROR!');
     return;
 end

 if x(1)==32
      helpdlg('Verifique porque su nombre empieza con espacio, en el campo
Nombre','ERROR!');
     return;
 end

 if ncolx>25
      helpdlg('Su nombre excede la cantidad de caracteres requerido, El campo
Nombre acepta (25 caracteres maximo)','ERROR!');
     return;
 end
load('archivo.txt');
C=archivo;
[nfil,ncol]= size(archivo);
if nfil==0
```

```
      helpdlg('La base de datos se encuentra vacia.','Información');
     return;
end;
band=0;
y=ncol-3;
for l=1:nfil
    cont=0;
for k=1:y
    if C(l,k)==0
        cont= cont+1;
    end
end
d= (y-cont);
if ncolx==d;
    con=0;
    for j=1:d
        if C(l,j)== x(j)
            con=con+1;
        end
    end;
    if con== d;
        files=dir('C:\Documents and
Settings\Administrador\Escritorio\tesis_2011\foto\');
        for i=3:length(files)
        n=files(i).name;
        [nombre,extension]=strtok(n,'.');
        if length(nombre)==length(w)
        con=0;
        for j=1:length(w)
        if nombre(j)== w(j)
            con=con+1;
        end
        end;
         if con== length(w)
         nombre_completo= strcat('C:\Documents and
Settings\Administrador\Escritorio\tesis_2011\foto\',n);
         axes(handles.axes1);
         axis off;
        imshow(nombre_completo);
        end;
         end;
        end;

        t='Usuario registrado en Nuestra base de datos';
        set(handles.text3,'String',w);
        set(handles.text4,'String',t);
        band=1;
%          helpdlg('Usuario registrado en Nuestra base de datos');
    end;
end;
end;
    if band==0
```

```
    nombre_complet= strcat('C:\Documents and
Settings\Administrador\Escritorio\tesis_2011\no_disponible.jpg');
    axes(handles.axes1);
    axis off;
    imshow(nombre_complet);
    t='Usted no se encuentra registrado. Contacte al Administrador';
    set(handles.text3,'String',w);
     set(handles.text4,'String',t);
    end;

function t= nombre(A)
t= A;
% --- Executes on button press in atras.
function atras_Callback(hObject, eventdata, handles)
delete(gcf);
menu_principal;


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
global bandera
bandera=2;
delete(gcf);
base_datos;
```

- **Base de datos**

```
%UNIVERSISDAD DE CARABOBO
%FACULTAD DE INGENERÍA
%ESCUELA DE ELÉCTRICA
%DEPARTAMENTO DE SISTEMA Y AUTOMATICA
%PROYECTO DE GRADO
%AUTORES: ALEXANDER MORENO
%         JONYRIS LAMAS

%INTERFAZ GRÁFICA: BASE DE DATOS
%   Muestra los usuarios incriptos en la base de datos
%   Para esto, debe el uusrio presionar  el botón mostrar, y luego >>
%   hasta que observe todos los usurios inscriptos en el sistema.
%   esta informacion  se muestra en bloque de 26 personas o menos.

function varargout = base_datos(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @base_datos_OpeningFcn, ...
```

```
                   'gui_OutputFcn',  @base_datos_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before base_datos is made visible.
function base_datos_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to base_datos (see VARARGIN)

%-----Centramos la figura---------
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/13);
yr=scrsz(3) - pos_act(3);
yp=round(yr/90);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
% --------------------------------------
axes1 = imread('carai.jpg');
axis off;
imshow(axes1);
% Choose default command line output for base_datos
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes base_datos wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = base_datos_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- se ejecuta al presinar el boton mostrar en la interfaz grafica
function mostrar_Callback(hObject, eventdata, handles)
global D E fil q band bandera
load('archivo.txt');  % Cargar el archivo
D= archivo;           %Se le asigna lo que esta en el archivo a la variable D
E=D;
fil=1;
[nfil,ncol]= size(D); %tamaño del archivo. Donde: nfil= # de filas, ncol= # de
columnas

if nfil==0
    % numeros de fila = 0 es porque no existe usuarios incriptos en la base de
datos
    % se imprime un mensaje al usuario para informarle que no existe datos
      helpdlg('La base de datos se encuentra vacia.','Información');
     return;
end;
q= nfil;
band=0;
bandera=0;
% si la base de datos esta compuesta por menos de 26 usuarios
% estos se mostraran en la ventana, de los cuales  se mostrara el nombre y la
posicion que ocupa.
% el nombre: es el nombre con el cual el usuario fue inscripto.
% la posicion: depende de la posicion en la base del sistema.
 if nfil<26
     for j=1:q
         D(1,:)=[];
     end;
    w=1;
for i=1:q
    v(1,w)= i; % v: vector de numeros, va desde p hastael numero filas.
    w=w+1;
end
 v=v'; % se le aplica trapuesto, para hacer que sea vertical en vez de
horizontal
t= char([E]); % el archivo es completamente numerico, ya que el nombre fue
convertido con Unit8
              % se trasforma estos numeros a letras según la tabla ascii
              % con la intrucion char.

set(handles.salida, 'string', t); %se imprime/muestra el nombre del usuario en
la interfaz
set(handles.salida1, 'string', v);%se imprime/muestra el vector vertical, en
la interfaz
```

```
band=1;
 end
 if band==0
for j=1:26
         D(1,:)=[];
end;
for i=1:nfil
    v(1,i)= i;
end
v=v';
t= char([E]);
set(handles.salida, 'string', t);
set(handles.salida1, 'string', v);
 end


% --- Ejecuta al presionar el boton atras----
% retorna a la ventana consultar registro y cierra la ventana actual
function atras_Callback(hObject, eventdata, handles)
global bandera
bandera=2;
delete(gcf);
Consultar_registro;


%----- se ejecuta al presionar el botón siguiente (>>)-----
function pushbutton3_Callback(hObject, eventdata, handles)
global E fil D q band bandera
% si bandera es difente de cero quiere decir que no presionaron el botón
mostrar
% por lo cual no han visto los primeros 26 registros;
%para evitar esto, le enviaron un msj al usuario indicandole debe presionar
Mostrar
if bandera==0
 if band==1
  % si la cantidad de usuarios es menor a 26
  %ya todas las personas de la base de datos estan siendo mostradas
  %por lo cual le enviaron un mensaje al usuario informandole esto.
     helpdlg('No existe mas usuarios en la Base de Datos.','ERROR!');
return
 end;
 % si el numero de filas es mayor a 26
[nfila,ncola]= size(D); % tamaño del archivo borrado
[nfilb,ncolb]= size(E); % tamaño del archivo original
if nfila>26
    for j=1:26
         D(1,:)=[];
    end;
    for j=1:26
         E(1,:)=[];
    end;
```

```
 w=1;
 p= (26*fil)+1; % p: contiene el numero apartir del cual se va leer el archivo
original
                % fil: contador; aumenta con la cantidad de veces que se
presiona>>
for i=p:q
    v(1,w)= i;  % v:vector numerico, contiene las nuvas posiciones apartir de
p
    w=w+1;
end
v=v';% trapuesta del vector; vector vertical
t= char([E]);
set(handles.salida, 'string', t);
set(handles.salida1, 'string', v);
fil= fil+1;% contador del numero de veces que se presiona el botón sig. en la
GUIs
end
 % si el numero de filas es cero es porque ya se mostraron todos los usuarios
 % le enviaron un mensaje al usuario para que este al tanto
if nfila==0
     helpdlg('No existe mas usuarios en la Base de Datos.','ERROR!');
     load('archivo.txt');
      A= archivo;
      E=D;
      fil=1;
return
 end;

 if nfila<26
     for j=1:nfila
         D(1,:)=[];
    end;
     for j=1:26
         E(1,:)=[];
    end;
    w=1;
  p= (26*fil)+1;

for i=p:q
    v(1,w)= i;
    w=w+1;
end
v=v';
t= char([E]);
set(handles.salida, 'string', t);
set(handles.salida1, 'string', v);
 end
 bandera=0;

end
```

```
if bandera~= 0
    %No han presionado el boton mostrar
    %por lo cual no se a cargado el archivo
 helpdlg('No se ha cargado la base de datos, por favor precione el boton
Mostrar.','ERROR!');
return
end

% fin de la programacion de la ventana Base de Datos
```

- **Borrar registro**

```
%UNIVERSISDAD DE CARABOBO
%FACULTAD DE INGENERÍA
%ESCUELA DE ELÉCTRICA
%DEPARTAMENTO DE SISTEMA Y AUTOMATICA
%PROYECTO DE GRADO
%AUTORES: ALEXANDER MORENO
%         JONYRIS LAMAS

%INTERFAZ GRÁFICA: BORRAR REGISTRO
%   elimina un registro de la base de datos.
%   para esto el administrador debe poseer el nombre del registro

function varargout = Borrar_registro(varargin)
% BORRAR_REGISTRO M-file for Borrar_registro.fig
%      BORRAR_REGISTRO, by itself, creates a new BORRAR_REGISTRO or raises the
existing
%      singleton*.
%
%      H = BORRAR_REGISTRO returns the handle to a new BORRAR_REGISTRO or the
handle to
%      the existing singleton*.
%
%      BORRAR_REGISTRO('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in BORRAR_REGISTRO.M with the given input
arguments.
%
%      BORRAR_REGISTRO('Property','Value',...) creates a new BORRAR_REGISTRO
or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Borrar_registro_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Borrar_registro_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
```

```
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Borrar_registro

% Last Modified by GUIDE v2.5 05-May-2011 23:42:03

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Borrar_registro_OpeningFcn, ...
                   'gui_OutputFcn',  @Borrar_registro_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Borrar_registro is made visible.
function Borrar_registro_OpeningFcn(hObject, eventdata, handles, varargin)

%-----Centramos la figura---------
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/13);
yr=scrsz(3) - pos_act(3);
yp=round(yr/40);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
% -------Fondo de la interfaz -------------------
axes1 = imread('carai.jpg');
axes(handles.axes1);
axis off;
imshow(axes1);


% Choose default command line output for Borrar_registro
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes Borrar_registro wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Borrar_registro_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function entrada_Callback(hObject, eventdata, handles)
% hObject    handle to entrada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of entrada as text
%        str2double(get(hObject,'String')) returns contents of entrada as a
double


% --- Executes during object creation, after setting all properties.
function entrada_CreateFcn(hObject, eventdata, handles)
% hObject    handle to entrada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function otra_Callback(hObject, eventdata, handles)
% hObject    handle to otra (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of otra as text
```

```
%        str2double(get(hObject,'String')) returns contents of otra as a
double


% --- Executes during object creation, after setting all properties.
function otra_CreateFcn(hObject, eventdata, handles)
% hObject    handle to otra (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- se ejecuta al presionar el boton borrar en la base de datos
function borra_Callback(hObject, eventdata, handles)
% lee nombre de usuario del campo nombre
 nomb=get(handles.entrada,'String');
  w= nombre(nomb);%asignan lo leido a la variable nomb
 x= uint8(w); %convierten el nombre en un vector numerico
[nfilx,ncolx]= size(x); % tamaño del vector numerico
 if ncolx==0
     % si el numero de columnas del nombre es cero
     % es porque no han llenado el campo nombre y presionaron borrar por error
     % le enviaron un mensaje al administrador que corrija el error
      helpdlg('Instroduzca su nombre, el campo Nombre se encuentra
vacio.','ERROR!');
     return;
 end
 % con numero de columna difernte de cero, es decir llenaron el campo nombre
 if x(1)==32 % el campo nombre comienza con espacio, o esta lleno de espacio
              % le enviaron un mensaje al adminstrador de que verifique este
error
      helpdlg('Verifique porque su nombre empieza con espacio, en el campo
Nombre','ERROR!');
     return;
 end

 if ncolx>25
     %el numero de columna del nombre mayor a 25 caracteres
     % excede la cantidad de caracteres
     % y no encuentra nadie con esta condiicion en la bse de datos
      helpdlg('Su nombre excede la cantidad de caracteres requerido, El campo
Nombre acepta (25 caracteres maximo)','ERROR!');
     return;
 end
load('archivo.txt');%  cargar el archivo de datos
```

```
A=archivo;% A: contiene lo que esta en el archivo
[nfil,ncol]= size(A);% tamaño del archivo
if nfil==0
    %si numero de filas es cero, no existen usuarios en la base de datos
    % se le envian un msj al administrador del sistema
      helpdlg('La base de datos se encuentra vacia.','Información');
     return;
end;
band=0;
y=ncol-3; %las ultimas tres columnas del archivos son las distancia de interes
         % y: contiene los venticincos carcteres del posible nombre del
archivo
%------ ciclo de busqueda de cero dentro del archivo-----
for l=1:nfil
    cont=0;
for k=1:y
    if A(l,k)==0
        cont= cont+1;
    end
end
d= (y-cont);% si el nombre del usuario en el archivo, es menor a 25 caracteres
           % se llena de cero en los espacio restantes
%----- ciclo de busqueda del nombre que desea borrar----
if ncolx==d;
    con=0;
    for j=1:d
        % numero de coincidencia entre el nombre guardado en el archivo y
        % el vector nombre introducido por el administrador
        if A(l,j)== x(j)
            con=con+1;% contador del numero de coincidencias
        end
    end;
    if con== d
        % si el numero de coincidencia es igual al nombre del archivo
        q= l;% fila del archivo a sert elimada archivo
        archivo(q,:)=[]; %borraron toda la fila
        u= archivo;% nuevo archivo, sin la fila borrada
        imprimir_borrado(u);% función imprimir archivo
        borrar_fotos_almacenadas(w);
        t='Fue borrado exitosamente de la Base de Datos';
        set(handles.text3,'String',w); % nombre del usuario borrado
        set(handles.text4,'String',t); % mensaje al administrador:
                                       % le garantiza la correcta ejecución
        band=1;
    end;
end;
end;
    if band==0
        %si band es igual a cero es porque no existe el usuario en la base
    t='Usted NO se encuentra registrado. verifique su nombre o consulte su
registro';
```

```
    set(handles.text3,'String',w); % imprime el nombre del usuario que desea
borrar
     set(handles.text4,'String',t);% mensaje de que no se encuentra ese nombre
en la base de datos
    end;


function t= nombre(A)
t= A;

% --- Ejecuta al presionar el boton atras----
% retorna a la ventana menu principal y cierra la ventana actual
function Atras_Callback(hObject, eventdata, handles)
delete(gcf);
menu_principal;
% fin de la programacion de la ventana borrar registro
```

- **Ayuda**

```
%UNIVERSISDAD DE CARABOBO
%FACULTAD DE INGENERÍA
%ESCUELA DE ELÉCTRICA
%DEPARTAMENTO DE SISTEMA Y AUTOMATICA
%PROYECTO DE GRADO
%AUTORES: ALEXANDER MORENO
%         JONYRIS LAMAS

%INTERFAZ GRÁFICA: MENÚ AYUDA
%   Brinda una breve información acerca de los elementos y ventanas del
software
%   Comienza con una informacion por defecto acerca del sistema
%   La interfaz está compuesta por: un panel de varible (listbox)
%                                   un panel de informacion (text)
%   el usuario dara clic sobre el elemento(en el panel de variable) de la cual
desee obtener información

function varargout = menu(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @menu_OpeningFcn, ...
                   'gui_OutputFcn',  @menu_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

%----se ejecuta al abrir la interfaz gráfica-----
function menu_OpeningFcn(hObject, eventdata, handles, varargin)

%--------Centramos la figura----------------------------------
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/19);
yr=scrsz(3) - pos_act(3);
yp=round(yr/75);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

% -------Fondo de la interfaz--------------------------------
background = imread('carai.jpg'); % imagen elegida
axes(handles.background); %Carga la imagen en background
axis off;
imshow(background);
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


function varargout = menu_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;


% --- Se ejcuta al presionar una de las varibles en el panel de control de la
GUIs------
% el panel de variable posee 15 opciones, que pueden ser elegida por el
usuario desde la interfaz gráfica (GUIs)
% las opciones se encuentran dentro del panel de variable en la GUIs
function listbox2_Callback(hObject, eventdata, handles)
inf=get(hObject,'Value');%leyendo opción elegida por el usuario del sofware
desde la interfaz gráfica
gos=get(hObject,'String');
switch inf
    case 1
    % opcion uno elegida por el usuario del software en la interfaz:
Recomendaciones
    % se muestra los querimientos minimimos que deben poseer para trabajar
        set(handles.text1,'String',...
```

```
            [{'Estos son los Requerimientos minimos recomendables que debe
poseer : ',' Pentium 4 ',' 1Gb ram ',' 2.6 KHz ', ' Camara Web ', ' Matlab 7.9
(R2009b) o superior',' Prototipo'}]);

    case 2
   %opcion dos elegida por el usuario es: Menu principal
   %se explica el funcionamiento de la ventana y las opciones que ella posee
        set(handles.text1,'String',...
            [{'Ventana de Menu Principal :','',' Esta consiste basicamente en
panel de tareas, nos permite acceder a las demas ventanas del software a
traves de la seleccion deseada por el usuario'...
            '','Posee los siguientes botones : Reconocimiento , Consultar
Registro , Nuevo Registro y  Borrar Registro. '}]);
    case 3
    %opcion tres elegida por el usuario es: Reconocimiento
   %se explica el funcionamiento de la ventana y las opciones que ella posee
         set(handles.text1,'String',...
            [{'Ventana de Reconocimiento:','',' Esta nos permite reconocer un
usuario. Para hacer posible esto , la persona a ser reconocida debe:
Configurar los parametros del dispositivo (webcam), activar el canal de dicha
camara, tomarse una foto, una vez confirmada la captura; es decir, una vez
aceptada la foto, debera presionar el boton buscar.',...
            '',' Nota: el usuario debe estar registrado en la base de datos
para ser reconocido'}]);

    case 4
   %opcion cuatro, elegida por el usuario es: Nuevo registro
   %se explica el funcionamiento de la ventana y las opciones que ella posee
        set(handles.text1,'String',...
            [{'Ventana de Nuevo Registro:','',' En esta ventana se puede
añadir un usuario a la base de datos; para tener acceso a ella el usuario debe
poseer la contraseña del sistema.',...
            'luego de teclear la contraseña correctamente, el administrador
debera configurar el dispositivo (camara), activar el canal de camara,
configurar el prototipo, capturar la imagen, confirmar la imagen, ',...
            ' tambien, tipiar el nombre del usuario (No debe exceder de 25
caracteres y debe excluir el punto), finalmente guardar el registro.'}]);


    case 5
   %opcion cinco, elegida por el usuario es: Consultar Registro
   %se explica el funcionamiento de la ventana y las opciones que ella posee
        set(handles.text1,'String',...
             [{'Ventana de Consultar Registro: ','',' Esta ventana nos permite
verificar si un usuario esta inscripto en la base de datos, esto se logra
colocando el nombre con que fue registrado el usuario y presionando el boton
Consultar; ademas posee un boton Base de Datos (se muestra todas las personas
registradas en la bases de datos). '}]);


    case 6
```

```
   %opcion seis, elegida por el usuario es: Borrar Registro
   %se explica el funcionamiento de la ventana y las opciones que ella posee
        set(handles.text1,'String',...
          [{'Ventana Borrar Registro: ','',' Esta ventana nos permite borrar a
cualquier usuario de la base de datos, para esto se requiere la clave de
acceso del sistema y el nombre con cual el fue registrado el usuario '}]);


    case 7
   %opcion siete, elegida por el usuario es: Base de Datos
   %se explica el funcionamiento de la ventana y las opciones que ella tiene
        set(handles.text1,'String',...
            [{'Ventana Base de Datos: ','',' Esta nos permite conocer los
usuarios que existe en la base de datos, para esto se debe presionar el boton
Mostrar, luego presione >> para continuar observando los usuarios
añdidos.',...
            'Repita este proceso hasta que no exitan mas usuario, si desea
retornar a la primera pagina presione nuevamente mostrar. ', ' NOTA : Cada vez
que se presione mostrar volvera al primer listado). '}]);


    case 8
   %opcion ocho, elegida por el usuario es: botón Configurar
   %se explica el funcionamiento de este botón,él se encuentra en dos ventanas
del programa
        set(handles.text1,'String',...
            [{'Botón Configurar Parametros:','', 'Ya que una PC puede contar
con diferentes dispositivos de captura, se considero necesario este botón, con
el cual el usuario podra seleccionar que dispositivo prefiere utilizar para
realizar la captura de la imagen. '...
            '','Para obtener información escriba en la ventana de trabajo de
Matlab la siguiente instrucción imaqhwinfo para conocer la cantidad de
dispositivo y las resoluciones con el cual trabaja webcam. '}]);


    case 9
   %opcion nueve, elegida por el usuario es: botón activar canal de video
   %se explica el funcionamiento de este botón,él se encuentra en dos ventanas
del programa
        set(handles.text1,'String',...
            [{'Botón Activar Camara:','',' En este botón se abre el canal de
video con el cual trabaja el sistema de reconocimiento, para su apertura
dedera por lo menos una vez configurar el dispositivo y resolucion con el cual
se va a trabajar. '}]);
    case 10
   %opcion diez, elegida por el usuario es: botón detener canal de video
   %se explica el funcionamiento de este botón,él se encuentra en dos ventanas
del programa
        set(handles.text1,'String',...
            [{'Botón Detener Camara: ','','',' Permite el cierre del canal de
video. '}]);
    case 11
   %opcion once, elegida por el usuario es: botón capturar
```

```
   %se explica el funcionamiento de este botón,él se encuentra en dos ventanas
del programa
        set(handles.text1,'String',...
           [{'Botón Capturar:','','Realiza captura de imagen apartir del canal
de video.','',' La captura de esta imagen es formato RGB'}]);

    case 12
   %opcion doce, elegida por el usuario es: botón Salir
   %se explica el funcionamiento de este botón,él se encuentra en todas las
ventanas del programa
        set(handles.text1,'String',...
            [{'Botón Salir: ','',' Sale del Sistema. '}]);
    case 13
  %opcion trece, elegida por el usuario es: mostrar
   %se explica el funcionamiento de este botón y su ubicación
        set(handles.text1,'String',...
              [{'Botón Mostrar: ','',' Este boton se encuentra ubicado en la
ventana de base de datos, con el cual el usuario puede observar la lista de
persona que se encuentran registrada en el sistema. '}]);

    case 14
   %opcion catorce, elegida por el usuario es: Prototipo
   %se explica como usuarlo y que lo compone
        set(handles.text1,'String',...
            [{' Prototipo: ','',' Consiste en una estructura de madera donde
el usuario colocara su rostro; Contiene dos lampara, una camara web
convencional y un laser, los cuales se encuentra debidamente ubicados para
generar la captura de la imagen adecuada. '}]);
    case 15
    %ultima opcion, elegida por el usuario es: Credito del software
        set(handles.text1,'String',...
            [{' Creditos del Programa : ','',' Reconocimiento de Rostro fue
elaborado en el software Matlab 7.9 por los Bachilleres: Alexander Moreno y
Jonyris Lamas, ',...
            ' para optar por el titulo de Ingeniero Electricista en la
Universidad de Carabobo Facultad de Ingenieria, bajo la tutoria del Ing.
Wilmer Sanz.  Junio 2011.  '}]);

end
guidata(hObject,handles);


function listbox2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function background_DeleteFcn(hObject, eventdata, handles)
```

```
% se ejecuta al presionar el boton salir en la la interfaz
% tiene como funcion salir del menu ayuda, cierra esta ventana
function pushbutton1_Callback(hObject, eventdata, handles)
delete(gcf)
%fin de la programacion de la ventana ayuda.
```