ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACIÓN

"Diseño e Implementación de un sistema de calificación para exámenes de opción múltiple mediante la herramienta MATLABR2008B"

TESINA DE SEMINARIO

Previo a la obtención del Título de:

INGENIERO EN ELECTRONICA

Y TELECOMUNICACIONES

Presentado por:

Mario Felipe Cotrina Escandón

Daniel de Jesús Peña Álvarez

GUAYAQUIL – ECUADOR

AÑO: 2011

DECLARACIÓN EXPRESA

La responsabilidad por los hechos, ideas y doctrinas expuestos en esta trabajo de Graduación, nos corresponden exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Graduación de la ESPOL).

_____

Mario Felipe Cotrina Escandón

_____

Daniel de Jesús Peña Álvarez

AGRADECIMIENTO

A las personas que dieron su apoyo para la realización del mismo, especialmente a la Ing Patricia Chávez por su ayuda y guía para la culminación de nuestro proyecto.

DEDICATORIA

A mis padres y hermanos que me ayudaron

A lo largo de mi carrera universitaria, a mi esposa

Que me alentó en la culminación y obtención

del título.

Mario Felipe Cotrina Escandón

A Johanna, mi esposa, por su amor y apoyo incondicional siempre

A David y Matias, mis hijos, que representan el futuro

A mis padres por todo lo que me han dado

Daniel de Jesús Peña Álvarez

TRIBUNAL DE SUSTENTACION

_____	_____
Mcs. Patricia Chavez B.	Ing. Daniel Ochoa D.
Profesora de la Materia	Delegado del Decano

RESUMEN

El presente proyecto presenta un sistema de de calificación de exámenes de opción múltiple, siendo las entradas las imágenes de los exámenes previamente escaneados. Muestra como salida la calificación de los mismos y tiene opciones de guardar las notas del evaluado y sus datos en un registro tipo texto.

En el capítulo 1 se trata de los antecedentes para el planteamiento y desarrollo de este software.

En el capítulo 2 se detallan los conceptos básicos de imágenes digitales en MATLAB, además se detallan los conceptos del tratamiento morfológico de imágenes que fueron aplicados para el desarrollo del proyecto.

En el capítulo 3 se detalla el proceso de implementación del proyecto, los pasos que se siguieron para llegar a la consecución del mismo.

En el capítulo 4 se presentan los resultados de las pruebas, de un número determinado de exámenes a corregir que fueron ejecutados por el programa, se determinará si existen falsos positivos o falsos negativos, en qué proporción se presentan, además si el programa tiene problemas en leer algunos exámenes.

En la parte final del documento y como anexo se presenta el manual de usuario que detalla los componentes del programa que es ejecutado en una interfaz grafica, para qué sirven y los pasos a seguir para correr el programa con éxito, también se adjunta como anexo el código en MatLAB de la programación.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCION

Este proyecto es un caso de cómo aplicar el procesamiento digital de imágenes, básicamente se procesa una imagen para poder extraer información o características especificas que el ojo humano no pueda captar o distinguir. Este proyecto analiza y procesa las características digitales de la imagen de una hoja de respuestas de un examen de opciones múltiples, se enfoca principalmente en reconocer la posición del casillero marcado por pregunta según el numero de pixeles en ella y determinar que numero de pregunta fue seleccionada para poder determinar la nota.

El siguiente documento detalla el manejo y las acciones que realiza el software de calificación de exámenes o pruebas de opciones múltiples, los exámenes son ingresados como imágenes escaneadas en formato jpg que fueron previamente almacenados en una base de datos. El software fue desarrollado con la herramienta MatLAB R2008b, la interfaz grafica nos muestra como entradas las respuestas del examen que servirá como patrón para la calificación de los exámenes a evaluar, tendremos como salida la calificación de los mismos y las opciones de mostrar respuestas fallidas, nulas o en blanco en un cuadro de mensaje, así mismo la opción de poder guardar esos datos en un archivo tipo txt que se almacenara como respaldo o registro en la carpeta de trabajo del programa. En nuestra facultad se están realizando proyectos con esta herramienta alentando de esta forma la

investigación, desarrollo e implementación de aplicaciones en las diversas áreas con esta plataforma.

CAPITULO 1

1.- Antecedentes Del Proyecto

Hoy en día casi no hay áreas de enfoque técnico que no hayan sido impactadas de alguna manera por el procesamiento digital de imágenes. En la actualidad, muchas instituciones educacionales, así como las de servicios públicos y privados diversos, evalúan a sus estudiantes, colaboradores y potenciales colaboradores respectivamente con pruebas de múltiple opción las cuales son realizadas masivamente y calificadas individualmente por personal humano. Esto toma demasiado tiempo en realizar debido a la cantidad de opciones y de exámenes, estas pruebas están siempre sujetas a errores involuntarios de parte del personal calificador.

Existen varias empresas desarrolladoras de software que han creado aplicaciones para automatizar el proceso mencionado anteriormente. Estas aplicaciones necesitan de licencia y muchas veces son difíciles de adquirir debido a sus altos costos.

La aplicación creada en el presente proyecto demuestra que con investigación, conocimientos sólidos respecto al programa en el que fue

desarrollado el software, y mucha práctica se puede lograr crear un producto diferente, útil y práctico.

1.1 Objetivos generales

- Automatizar procesos de calificación de exámenes múltiple opción de manera rápida y sin complicaciones para el usuario que usará la herramienta.
- Calificar mediante la comparación de respuestas, se hará con las diversas herramientas que posee matlab en sus herramientas de procesamiento de imágenes.
- Generar opciones de almacenamiento de datos para consultas posteriores o inmediatas.

1.2 Objetivos Específicos

- Elaborar un software que permita leer las respuestas del examen patrón y compararlas con las respuestas de los exámenes de los evaluados, y como salida proporcionar la nota final.
- Mostrar como salida en pantalla la calificación del exámen.
- Generar un reporte visual una vez calificado el examen donde se muestre un resumen de los resultados del examen ya calificado.

- Guardar en un archivo de texto, las calificaciones de cada estudiante, indicando los aciertos, errores, opciones nulas o anuladas y las opciones en blanco.
- Ingresar los datos personales del candidato o estudiante dentro del archivo de texto (.txt)

CAPITULO 2

2.- MARCO TEORICO

El siguiente marco teórico comprende los conceptos necesarios del procesamiento de imágenes digitales que se aplico en el proyecto.

2.1 Concepto de imágenes en MATLAB

MATLAB almacena las imágenes en escala de grises como vectores bidimensionales o matrices, en el que cada elemento de la matriz corresponde a un pixel. Por lo tanto una imagen se define como una función de dos dimensiones f(m,n) donde m y n son coordenadas espaciales **(Fig. 2.1).** La imagen de color RGB, es representada por una matriz tridimensional *m* x *n* x *p*, donde *m* y *n* tienen la misma significación que para el caso de las imágenes de escala de grises mientras *p* representa el plano, que para RGB que puede ser 1 para el rojo, 2 para el verde y 3 para el azul **(Fig. 2.2).**

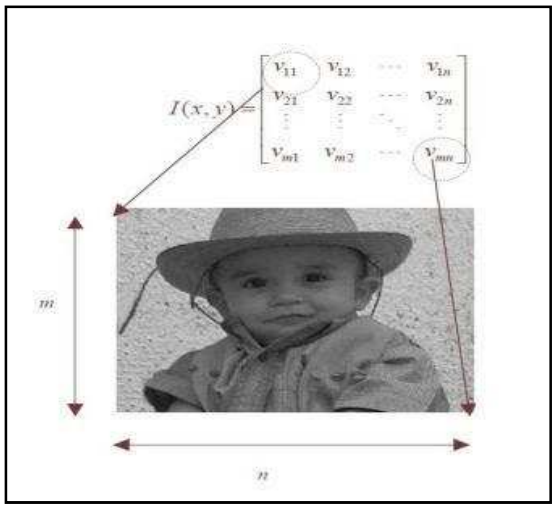

Fig. 2.1 Representación en matlab de una imagen en escala de grises [1]

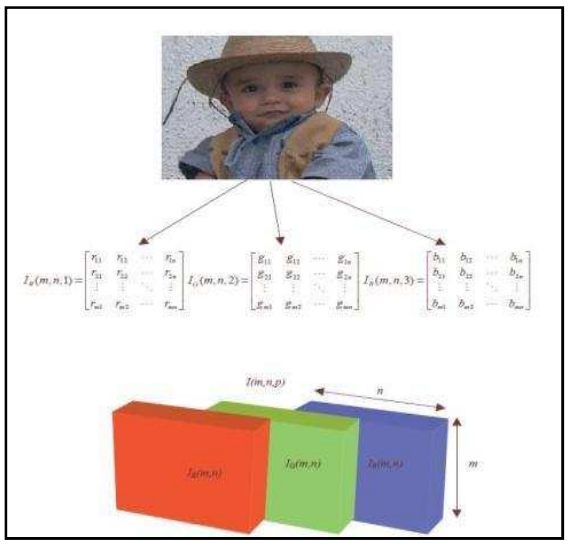

Fig. 2.2 Representación en matlab de una imagen a color. [1]

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 procesamiento de imágenes 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.

2.2 Binarización

La binarización es obtener una imagen que sólo sea representada por dos tonos de color, generalmente, blanco y negro. Para ello sólo debemos decidir cuál de los dos tonos de color dar a cada píxel de la imagen de entrada. Esta tarea se lleva a cabo especificando un valor umbral o límite también llamado umbral de binarización.

Para hacerlo se debe primero pasar la imagen a escala de grises, se recorre cada pixel, se compara su nivel de gris con el umbral de binarización, Si el valor del pixel es mayor o igual al umbral la salida será blanco o 1 y si valor del pixel es menor que el umbral será negro o 0 **(Fig. 2.3).**

Fig. 2.3 Ejemplo de la binarización de una imagen en escala de grises [2]

2.3 Etiquetamiento

La función bwlabel realiza un etiquetado de los componentes existentes en la imagen binaria, la cual puede ser considerada como una forma de averiguar cuántos elementos están presentes en la imagen. La función tiene el siguiente formato:

[L ne] = bwlabel(BW)

Donde L es la imagen resultado que contiene los elementos etiquetados con el número ne correspondiente al objeto, BW es la imagen binaria que se desea encontrar el número de objetos. Resumiendo esta función nos ayuda a determinar el número de elementos presentes en la imagen

2.4 Propiedades de los objetos en la imagen

El comando regionprops mide propiedades de objetos o de regiones en una imagen y las retorna como un arreglo. Cuando se aplica a una imagen con componentes rotulados o etiquetados, crea un elemento para cada componente, la función tiene el siguiente formato:

```
im= regionprops(L,'basic')
```

Donde im será una estructura con el número de elementos etiquetados o conectados en la imagen con sus respectivas propiedades calculadas, para este caso basic significa el cálculo básico de propiedades que son el área, centro de masa y las dimensiones de caja del elemento, L es la matriz de elementos etiquetados y es la única entrada a la función. Este comando nos ayuda a distinguir formas u aéreas por lo que se podrían analizar elementos específicos de acuerdo a una de estas características.

2.5 Operaciones Morfológicas

Una de las operaciones más utilizadas en visión sobre imágenes previamente binarizadas son las operaciones morfológicas. Estas aplican un elemento estructural a la imagen de entrada, sin cambiar

el tamaño de la imagen de salida. Las operaciones morfológicas más comunes son la dilatación y la erosión. En una operación morfológica, el valor de cada píxel en la imagen de salida depende del valor de ese píxel en la imagen de entrada y su relación con la vecindad. Seleccionando el tamaño y forma de la vecindad (definido a través de un elemento estructural) se puede crear una operación morfológica, que altera el valor del píxel en la imagen de salida.

2.5.1 Elemento estructural.

Define la forma y el tamaño de la vecindad del píxel que será analizado, para posteriormente alterar su valor. Formada por ceros y unos de forma y tamaño arbitrario en la cual las posiciones donde está el uno define la vecindad. La Matriz que define el elemento estructural tiene un tamaño muy inferior al tamaño de la imagen a la que modificará. El elemento estructural define el tamaño y la forma de la vecindad en la que se aplicará la operación morfológica. Dentro de las formas del elemento estructural tenemos: cuadrado, diamante, disco, línea, círculo, rectángulo, octágono **(Fig. 2.4).**

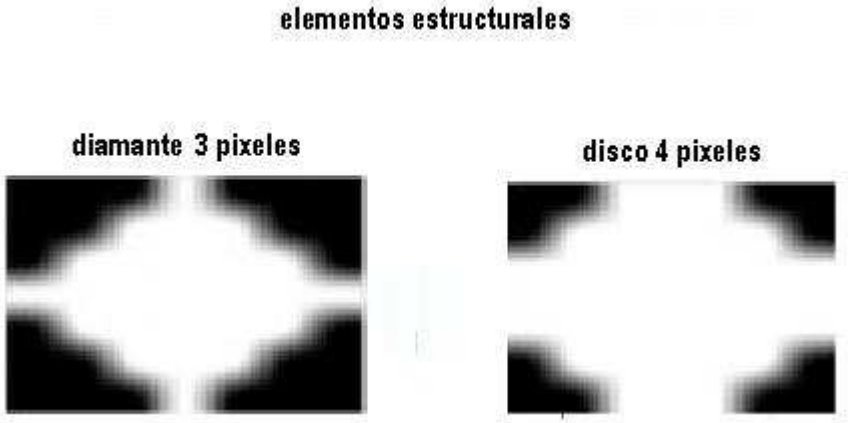

Fig. 2.4 Ejemplos de elementos estructurales [2]

2.5.2 Dilatación

La dilatación se basa en aumentar el nivel de los valores de los píxeles en el entorno de los objetos presentes en la imagen. Para calcular la dilatación se superpone el píxel central del elemento estructural a cada píxel de la imagen de entrada, entonces el píxel de la imagen de entrada se altera en función de los valores de los píxeles del entorno, definidos por el elemento estructural. El valor del píxel de salida será el máximo entre todos los píxeles presentes en la vecindad. La dilatación sirve para: ampliar bordes, unir objetos próximos, elimina detalles negros pequeños (**Fig. 2.5).**

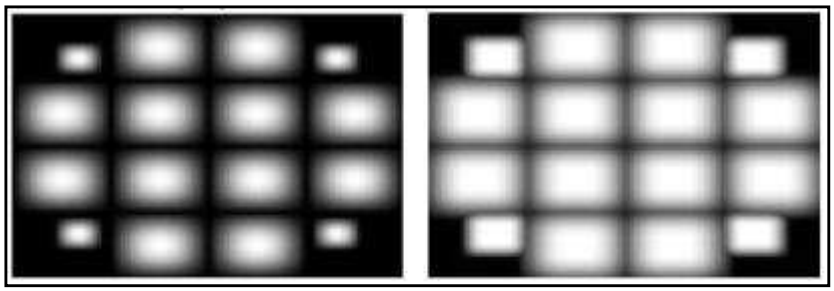

Fig. 2.5 Ejemplo de dilatación en una imagen binaria [2]

2.5.3 Erosión

La erosión se basa en reducir el nivel de los píxeles del entorno de un objeto presentes en una imagen. El pixel de salida será el mínimo de los niveles presentes en la vecindad definida por el elemento estructural. La erosión sirve para: reducir bordes, separar objetos próximos, eliminar puntos blancos separados, amplía detalles negros pequeños (**Fig. 2.6)**.

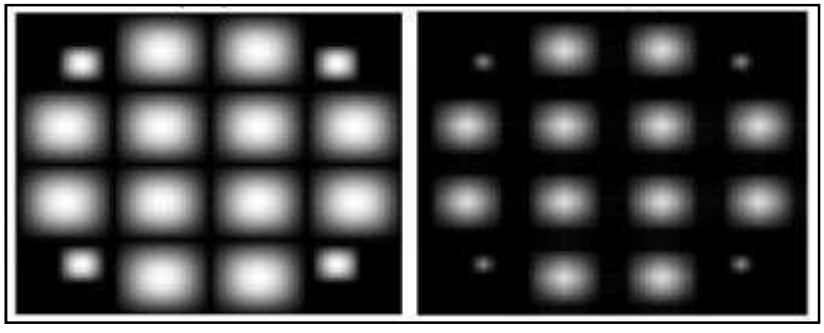

Fig. 2.6 Ejemplo de de erosión en una imagen binaria [2]

2.5.4 Apertura

Es la realización de una erosión seguida de una dilatación, utilizando el mismo elemento estructural en ambas operaciones. La apertura sirve para: suavizar contornos de los objetos, elimina pequeñas protuberancias, rompe conexiones débiles (**Fig. 2.7)**.

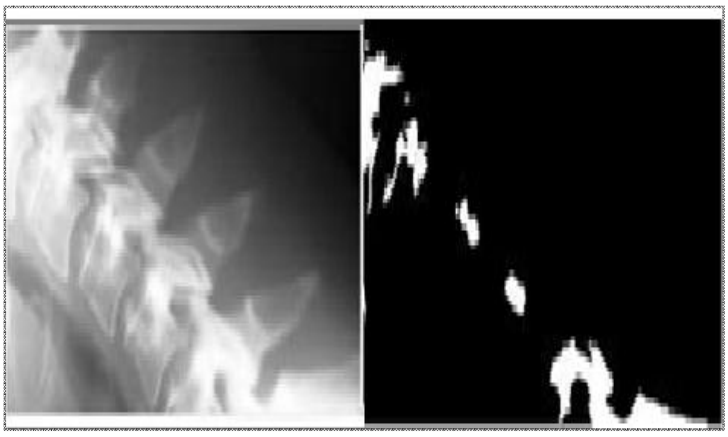

Fig. 2.7 Ejemplo de apertura [2]

2.5.5 Cerradura o cierre

Es la realización de una dilatación seguida de una erosión, utilizando el mismo elemento estructural en ambas operaciones. La cerradura sirve para: rellenar detalles conectando objetos que están próximos entre sí, suavizar los contornos, rellenar vacíos en el contorno, eliminar pequeños huecos (**Fig. 2.8)**

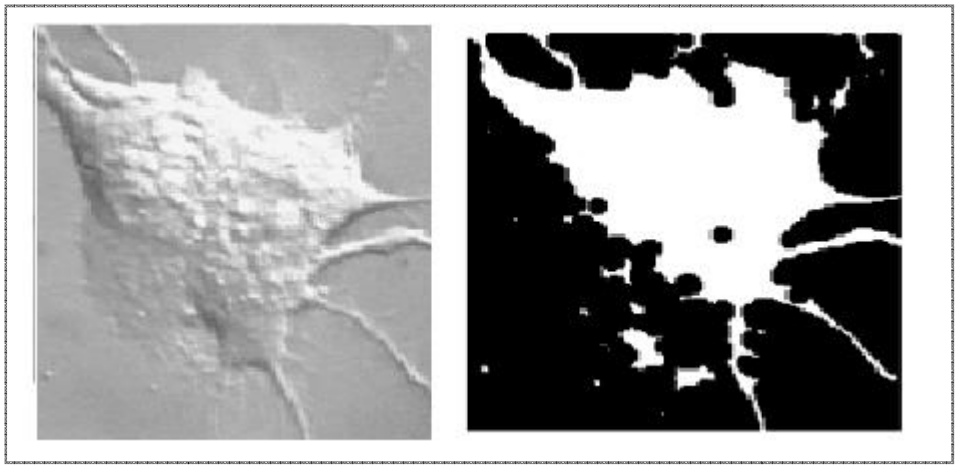

Fig. 2.8 Ejemplo de cerradura. [2]

CAPITULO 3

3.- Implementación

Para la implementación de este proyecto es importante contar con todos los recursos de hardware y software para garantizar el funcionamiento correcto del programa. Primero se detallara los requisitos de hardware mínimos necesarios para correr maltab y luego los pasos que se siguieron para el desarrollo del proyecto.

3.1 Requerimientos de hardware.

Matlab corre en varios sistemas operativos por lo que es necesario especificar cuáles son los requerimientos mínimos de hardware para cada uno de ellos.

Los requerimientos mínimos para una PC con Windows son:

- Procesador: Intel (Pentium IV o superior, Celeron, Xenón, Core) o AMD (Athlon)

- SO: Windows XP SP2 o 3, Windows 2000 (con Service Pack 1 o R2), Windows Vista SP1 y Windows Server 2008, windows 7.
- RAM: mínima: 512 RAM. Se recomendable: 2 G para tener mejor velocidad en los procesos.
- Espacio en disco: 1 GB de espacio en disco tras la instalación de todos los productos necesarios y librerías.
- Tarjeta gráfica: Adaptador gráfico compatible con OpenGL de 16, 24 o 32 bits
- Unidad: unidad de CD o dvdwr.
- Tarjeta de sonido: tarjeta de sonido estéreo compatible con Windows y altavoces

Los requerimientos mínimos para una PC con Linux son:

- Procesador: Intel (Pentium IV o superior, Celeron*, Xenón, Core) y AMD (Athlon 64, Opteron o Sempron*) (*el procesador debe ser compatible con el paquete de instrucciones SSE2)
- SO: Debian 4.0 y superior, Red Hat Enterprise Linux v4 y superior, OpenSuSE 9.3 y superior, Ubuntu 8 y superior
- RAM: mínima: 512 RAM. Muy recomendable: 1.024 MB
- Espacio en disco: 1 GB de espacio en disco tras la instalación de todos los productos necesarios.

- Tarjeta gráfica: Adaptador gráfico compatible con OpenGL de 16, 24 o 32 bits.
- Unidad: unidad de CD o acceso a la red para la instalación.

Los requerimientos mínimos para una Macintosh:

- Procesador: todos los Mac basados en Intel
- SO: Mac OS® X 10.5.5 y superior
- RAM: mínima: 512 RAM. Muy recomendable: 1.024 MB
- Espacio en disco: 1 GB de espacio en disco tras la instalación de todos los productos necesarios para el curso
- Unidad de CD o acceso a la red para la instalación

3.2 Desarrollo del proyecto

A continuación se detallará la secuencia de las partes en la implementación del proyecto.

3.2.1 Lectura del examen patrón y del bloque de respuestas

Se tiene previamente la base de datos de las imágenes en formato jpg de la hoja de respuestas de los exámenes de los evaluados en una carpeta y como adicional la imagen de la hoja de respuestas del exámen patrón.

El primer paso es cargar el examen patrón **Fig. 3.1**, si no se

selecciona no se seguirá con el resto del programa, se carga la imagen en una vista previa, se binariza la imagen, se dilata con un elemento estructural lineal, internamente se procede a separar el bloque de casillas de respuestas de la numeración, se analiza que casillero ha sido marcado fila por fila con su respectiva opción, para la imagen seleccionada de trabajo el área de la casilla marcada tiene un área aproximada de 480 pixeles, hay casos en el que evaluado no marca totalmente la casilla o lo hace con un lápiz que no es el adecuado para este tipo de exámenes, para este caso el programa tiene una verificación adicional del área para que sea tomada como valida mediante un promedio de áreas y si es mayor a este promedio se la da como válida, una vez identificado el casillero seleccionado se procede a almacenar esa información en una matriz para la posterior comparación con las demás hojas de respuestas.

Resumiendo la acción examen patrón es obtener una matriz donde se encuentren las respuestas en el caso de que se haya respondido y un indicador si no se ha respondido o si la respuesta es nula. Para ver los comandos y códigos utilizados ver el ANEXO B (código fuente).

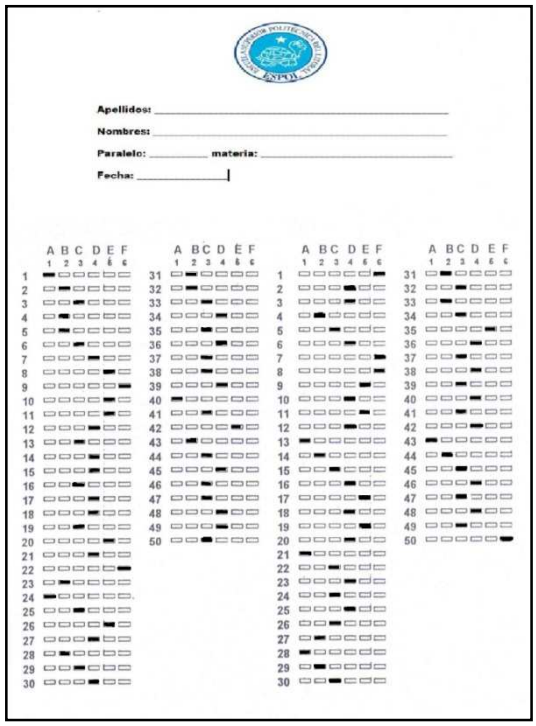

Fig. 3.1 Imagen del examen patrón

3.2.2 Lectura del examen a calificar.

En esta parte se hace un proceso similar al de el exámen patrón pero más sencillo, se lee la imagen y se la carga en la vista previa y se recorta el bloque de respuestas**.** La imagen recortada es almacenada en una matriz para luego ser llamada en la opción de calificar.

3.2.3 Calificar.

Se llama a la imagen recortada del exámen a calificar y se repite el proceso como se lo hace en el examen patrón, se aplica la binarizacion, dilatación de un elemento estructural de la imagen y la limpieza de posibles impurezas de la imagen, se separa el bloque de respuestas **Fig. 3.2**, se identifican las respuestas seleccionadas fila por fila y se las almacenan en una matriz con su respectiva opción.

Esta parte del programa tiene el algoritmo de calificación, comenzando con restar las respuestas del examen patrón de las respuestas del examen en particular, en el caso de que esta resta sea igual cero significa que hay una coincidencia que corresponderá una pregunta contestada correctamente ,contamos el numero de respuestas correctas y las establecemos en la salida de la interfaz grafica casilla correctas, ya con el numero de respuestas correctas de acuerdo al valor de cada una se establece la nota multiplicando el numero de respuestas correctas con el valor, la nota es visualizada en la interfaz grafica casilla nota. Además se guarda el valor de la nota y numero de respuestas correctas para ser exportadas en otra función.

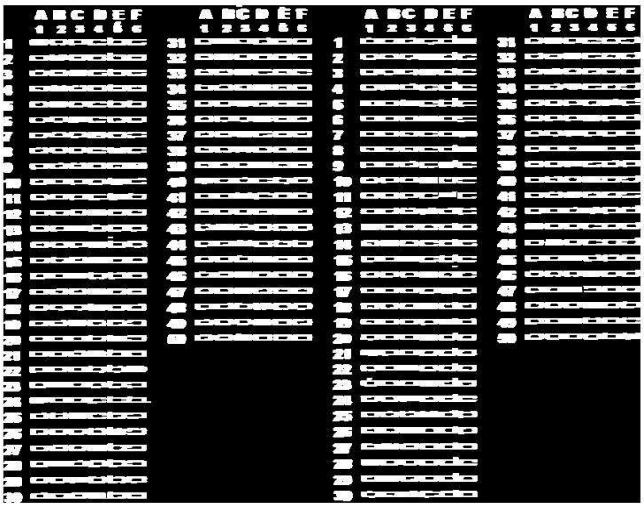

Fig. 3.2 Imagen del bloque de respuestas recortado, binarizadas y dilatadas

3.2.4 Mostrar correctas e incorrectas.

En esta parte se llama las respuestas correctas del examen patrón con las respuestas del examen ingresado, del bloque de respuestas identifico el numero de la pregunta que ha sido anulada y de igual forma las blancas, así mismo con las incorrectas y son almacenadas para ser presentadas en un cuadro de mensaje donde se mostraran al usuario **Fig.3.3**

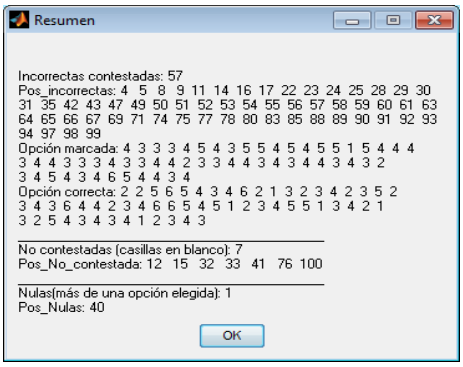

Fig. 3.3 Imagen del cuadro de mensaje que presenta las respuestas

3.2.5 Guardar datos.

Presenta una interfaz grafica donde pueden ser digitados los datos del evaluado, nombres, apellidos, cédula o número de matrícula, materia, paralelo, se ingresan los datos y se importa la información de las preguntas contestadas incorrectamente, nulas y blancas a un archivo txt creado con los atributos de escritura y lectura, se guarda toda esta información y este archivo txt se guarda en la carpeta de trabajo con el nombre del evaluado, esta interfaz fue desarrollada separada del programa principal pero están unidas por variables globales que son los datos que se ingresan del evaluado. Se puede visualizar más fácilmente los procesos, entradas y salidas en la **Fig. 3.4** que es el flujograma del programa.

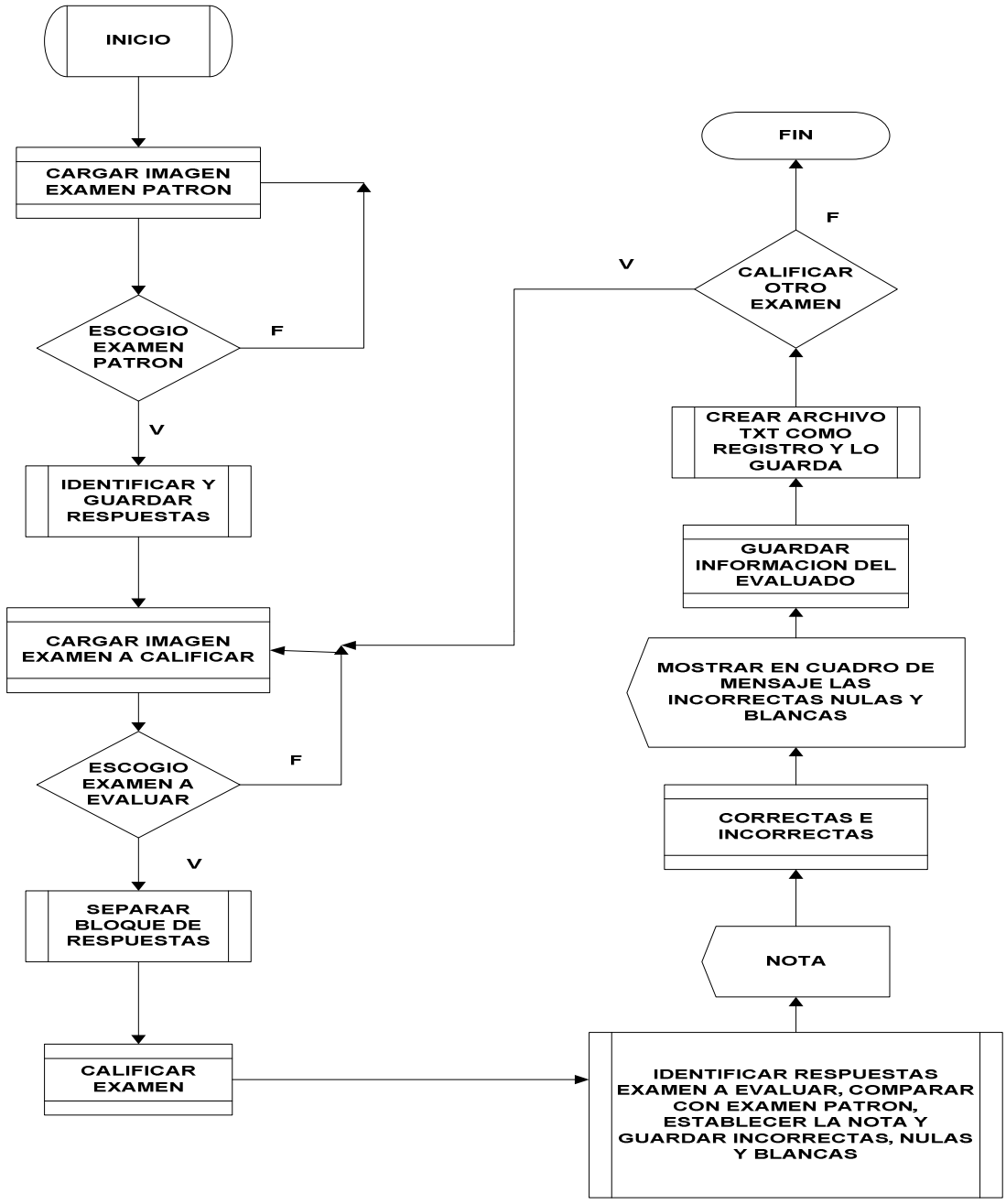

Fig. 3.4 Flujograma del programa

CAPITULO 4

4. PRUEBAS Y RESULTADOS

En este capítulo detallaremos cuales fueron las pruebas realizadas con el proyecto y los resultados de las mismas. El tipo de formato de las imágenes utilizadas para las pruebas fue jpg, el programa solo lee ese tipo de imágenes porque así fue establecido en los objetivos específicos que se detallan al principio de este documento, la dimensión en pixeles para todas fue 1533x1966 y una resolución por pulgada de 200, que es el valor por defecto que tenia la maquina que las escaneó.

Para que la comparación de los exámenes sea exitosa se permite una variación máxima de 1º con respecto a la horizontal caso contrario se presentan problemas con la ejecución del mismo. **Fig. 4.1**

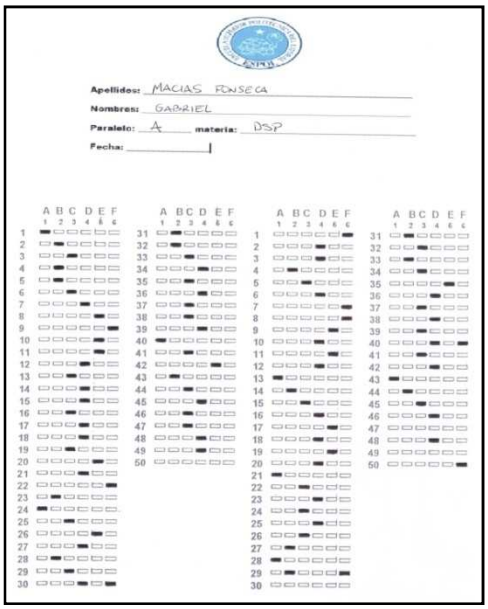

Fig. 4.1. Imagen desalineada 1º.

Se procedió a probar 35 exámenes, de este número se presentaron problemas con dos, se presento un mensaje de error en la ventana de comandos de matlab **Fig. 4.2.**

Fig. 4.2 Mensaje de error en MatLAB

Matlab es muy explicito en sus mensajes de error, el mensaje muestra que las dimensiones de las matrices de las imágenes no están concordando, se reviso la imagen y se observo que estaba inclinada **Fig. 4.3**. Se volvió a hacer el escaneo respectivo cuidando que este bien alineado resultando exitoso esta vez.

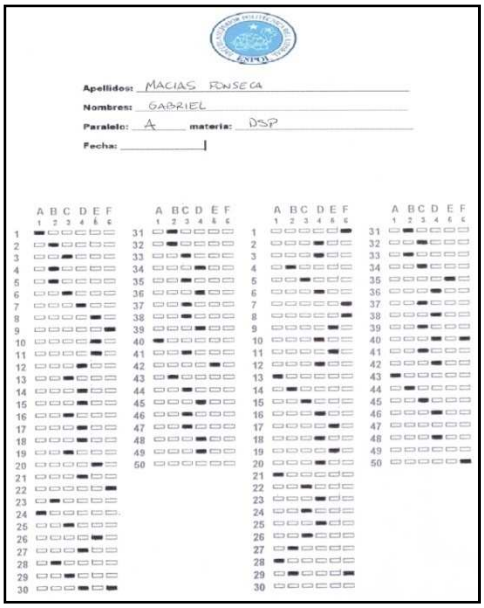

Fig. 4.3 Imagen alineada correctamente

De las pruebas realizadas se reviso que entre los resultados existiese la presencia de falsos positivos o falsos negativos, para esto se sabía que preguntas eran las correctas, incorrectas, nulas o blancas y su posición, estas eran marcadas personalmente y el programa tenía que corroborar lo hecho, el programa no se equivoco y mostro los resultados sin error **Tabla 4.1**.

Exámenes calificados	**35**
Falsos positivos	**0**
Falsos negativos	**0**
Nulas asignadas como positivas	**0**
Nulas asignadas como negativas	**0**
Problemas de margen en imágenes	**2**

Tabla 4.1 Tabla de resultados.

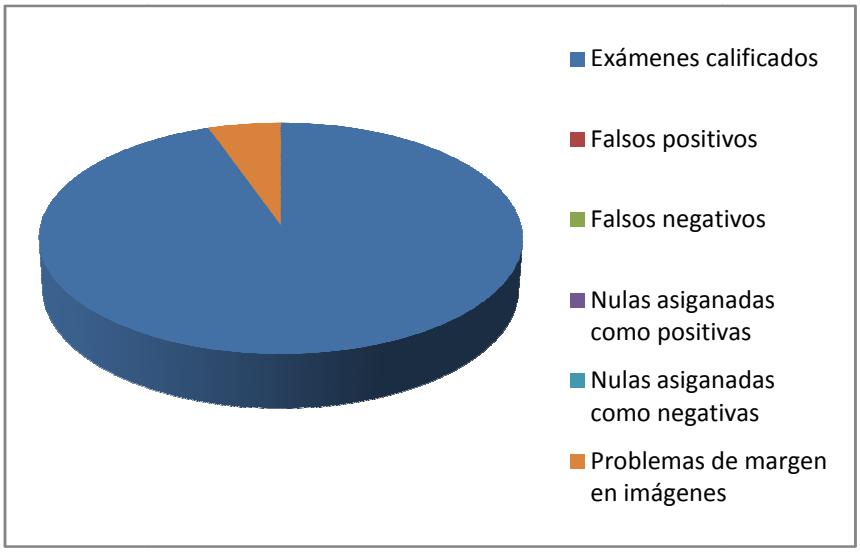

Fig.4.4 Resultado de las pruebas

Se obtuvo un porcentaje de eficiencia del 94% **Fig. 4.4** para las 35 imágenes escaneadas analizadas.

Las pruebas fueron corridas en una PC clon con sistema operativo Windows XP con procesador doble núcleo intel de 3GHZ, memoria RAM de 2gigas, el tiempo tomado por la PC para procesar la nota del primer exámen fue de 5

segundos, para el resto de exámenes tomo 3 segundos debido a que las respuestas del exámen patrón ya están almacenadas al ingresar el exámen patrón.

Se realizaron también pruebas con una PC clon con sistema operativo Windows xp con procesador Pentium 4 de 1.7GHZ, memoria RAM de 1giga, el tiempo tomado por la PC para procesar la nota del primer examen fue 25 segundos, para el resto de exámenes tomó 21 segundos en calificarlos.

Para determinar si había errores en los registros de los datos de los evaluados así como de su calificación o de errores en los nombres o archivos reemplazados al ser guardados, se procedió a revisar cada uno de los exámenes en los datos que se ingresaron y su creación en la respectiva carpeta de trabajo del programa en matlab, el error que se puede dar es ingresar equívocamente por el usuario algún nombre o identificación, los demás datos son importables del mismo programa**.**

Todas las pruebas realizadas se guardaron correctamente, en este caso no se presentaron problemas, el archivo *.txt conserva la información de cada uno de los exámenes sin alteraciones todos los exámenes fueron comparados con el patrón mostrado en la **Fig. 3.1.**

En la **Fig. 4.5** mostramos los resultados obtenidos al presionar Correctas e Incorrectas en nuestro programa, en la misma se detalla todos los aciertos y errores.

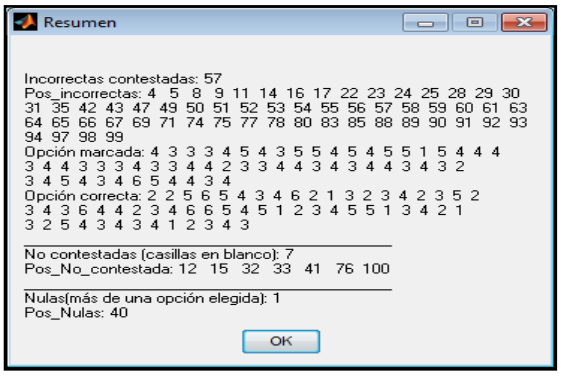

Fig. 4.5 Respuestas Correctas e Incorrectas

A continuación el programa permite guardar en un archivo con formato txt lo visualizado en la **Fig. 4.6** solo tenemos que presionar la tecla GUARDAR y nos muestra lo siguiente:

Fig. 4.6 Asignación de DATOS al examen calificado

Los archivos son editables ya que en la programación se estableció que sean editables en sus atributos, además se guardan por apellido y luego el nombre para ser más fácil de ordenar alfabéticamente.

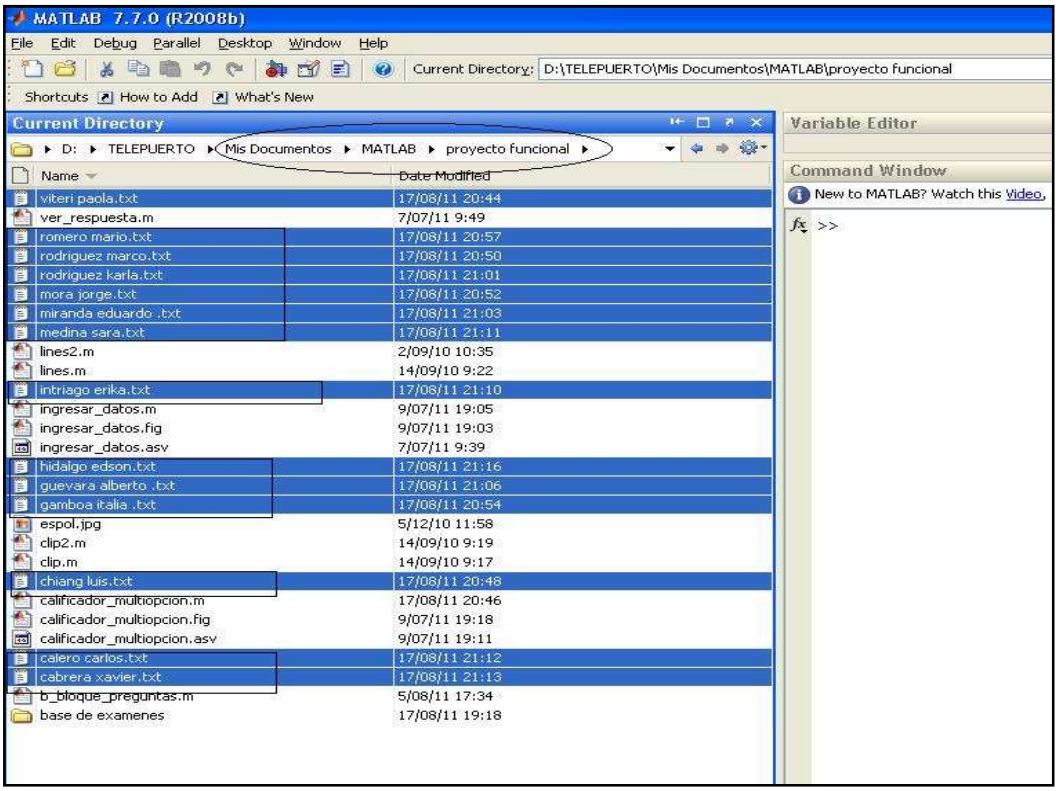

Fig.4.7 Carpeta de trabajo que almacena los archivos txt.

CONCLUSIONES

1) De las pruebas realizadas con el programa se observo que para poder identificar respuestas correctas, incorrectas, nulas y en blanco se depende de que las imágenes de los exámenes escaneados estén correctamente alineadas, caso contrario no correrá el programa ya que hace comparaciones de mascaras y si las mascaras no concuerdan matlab envía un mensaje de error que no hay concordancia de matrices.
2) Si las dimensiones en pixeles de la imagen así como los pixeles por pulgada o por centímetro son diferentes a las de la imagen con la cual se hicieron las pruebas se deben cambiar varios parámetros en la programación principal como en las funciones que son llamadas en el mismo, el programa fue diseñado para analizar la imagen que se tomo como muestra si hay alguna variación habrá que ajustar la programación.
3) La aplicación de elementos estructurales, dilatación y erosión así como apertura y cierre fueron de gran ayuda para identificar formas en este caso particular rectangular que es la forma de la casilla de la hoja de respuestas.
4) El archivo txt creado para guardar la información del evaluado no

presenta alteraciones o reemplazos con otros archivos existentes, además por su formato ocupa poco espacio de disco y la mejor opción para almacenar por su simplicidad, además no se guarda en otra ubicación distinta a la carpeta de trabajo del programa.

RECOMENDACIONES

1) En futuro trabajo se recomienda que la imagen escaneada sea leída o cargada directamente desde el escáner.
2) Se recomienda que el alineamiento de la imagen escaneada sea automatizado, ahorrando trabajo al usuario y aumentando la efectividad del programa.
3) Se recomienda que el programa pueda guardar los datos del evaluado de forma automática por medio del reconocimiento óptico de caracteres así se ahorraría tiempo en ingresar los datos del evaluado.
4) Se debe de tener en claro que el programa debe ser modificado en algunos parámetros si quiere que funcione con otro tipo de exámenes múltiple opción como son los que tienen casilleros en forma de círculo, también se debe tener en cuenta el tamaño de las casillas y las dimensiones en pixeles de la imagen escaneada.
5) Se necesita de un programa de edición de imágenes para determinar el tamaño en pixeles de una imagen y para determinar el tamaño de cierta área en particular.

ANEXOS

ANEXO A

MANUAL DE USUARIO

El siguiente manual de usuario detalla el manejo y funciones de un software desarrollado con la herramienta matlab 2008b, se lo ejecuta mediante una amigable y fácil interfaz grafica de usuario.

El programa está diseñado para calificar una hoja de respuestas de exámenes de opción múltiple a partir de un examen patrón del cual se guardaran sus respuestas y se compararan con los demás exámenes a calificar, a continuación detallaremos paso a paso cual es su funcionamiento

Para cargar el programa hay varias opciones, una vez abierto o cargado matlab la primera forma es digitar el nombre del software en la ventana de comandos.

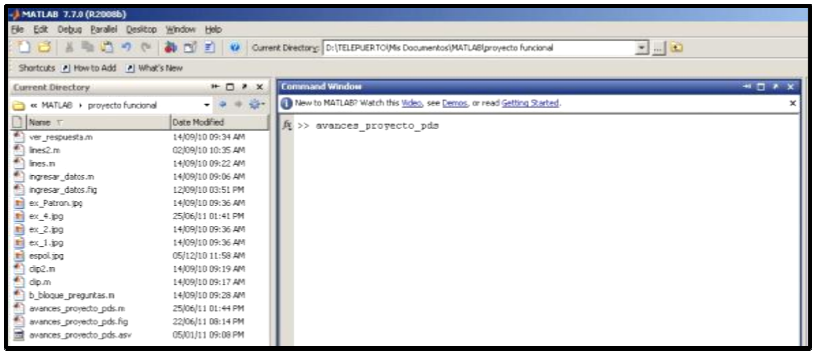

Fig. A-1. Ventana del command window en matlab.

Otra forma de correr el software es dar click derecho sobre el archivo de extensión .m y dar click en run file.

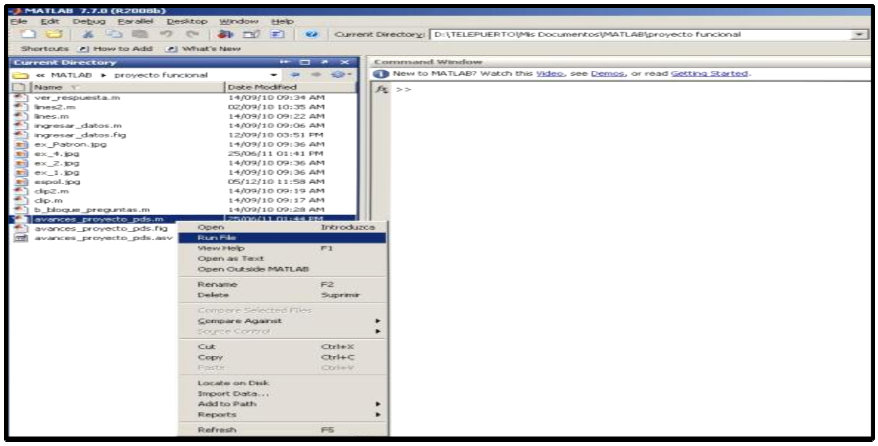

Fig. A-2. Ventana del current directory en matlab donde está la carpeta del programa

Una vez abierto el software aparecerá una interfaz grafica de usuario de fácil uso, a continuación se describirá la función de los botones que aparece en el panel de opciones:

Fig. A-3. Menú principal del programa

Descripción de botones

Examen patrón: este botón nos permite seleccionar la imagen del examen patrón sobre el cual se van a calificar todos los exámenes y lo carga en la vista previa examen patrón.

Examen a calificar: este botón nos permite seleccionar el examen que se desea calificar y cargarlo en la vista previa examen a calificar.

Calificar desde: Se coloca el primer examen que se quiera proceder a calificar

Calificar hasta: Colocamos el examen final hasta donde queremos que califique, el programa automáticamente empieza con el proceso de calificación según el rango que le ponemos, cada vez que el programa culmina de calificar un examen automáticamente pide guardarlo y lo hace en extensión *.txt. En la parte inferior del programa se podrá visualizar cual es el examen que está siendo calificado

Paso 1: cargamos el examen patrón presionando el botón examen patrón del panel de opciones.

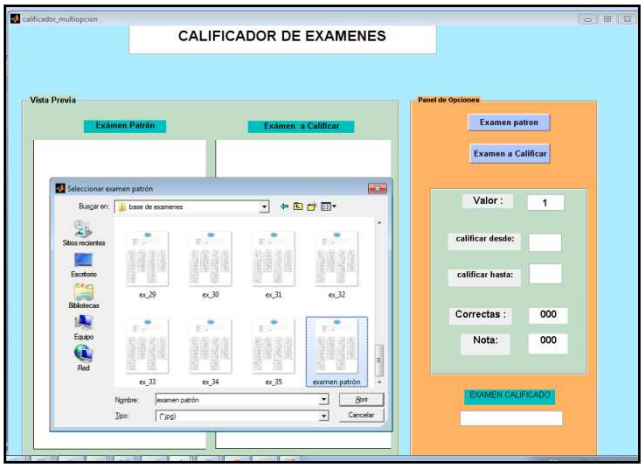

Fig. A-4.- Ventana de Selección de imagen del examen patrón

Fig. A-5.- Vista el exámen patrón cargado en la interfaz gráfica

Paso 2.- Se coloca la ponderación para cada pregunta antes de proceder a seleccionar el examen

Paso 3.- Seleccionamos el rango de exámenes en que queremos que empiece y termine nuestro proceso, luego se da en el botón examen a calificar y seleccionamos el examen.

Fig. A-6.- Vista el exámen a calificar cargado en la interfaz grafica

Paso 4.- Se proceden a guardar cada uno de los exámenes que se califican automáticamente, según el rango indicado. Se finaliza cada uno presionando ACEPTAR y automáticamente se procede con el siguiente

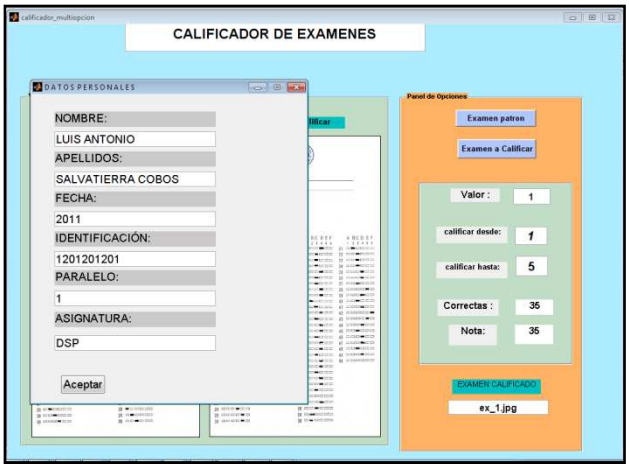

Fig. A-7.- Vista en donde se registra información de cada examen

Paso 5.- Al dar clic en el último examen se termina el proceso y aparece un mensaje que así lo indica.

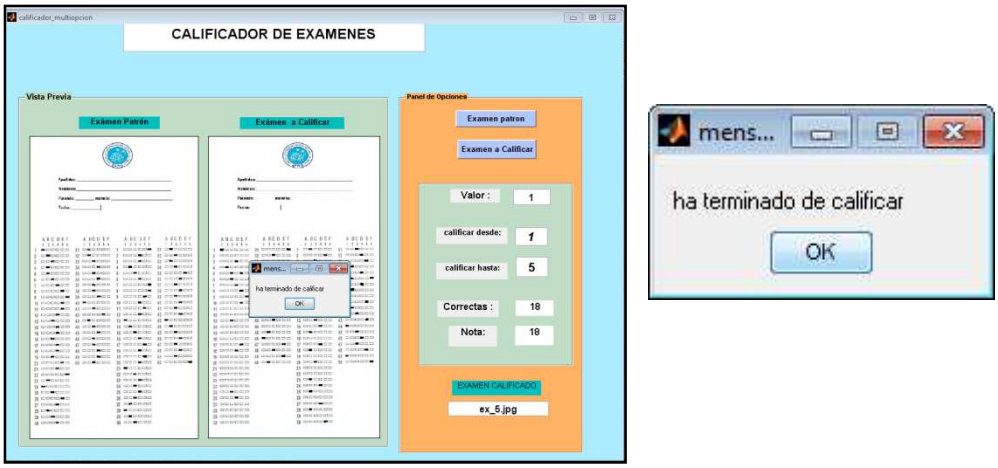

Fig. A-8.- Fin del proceso

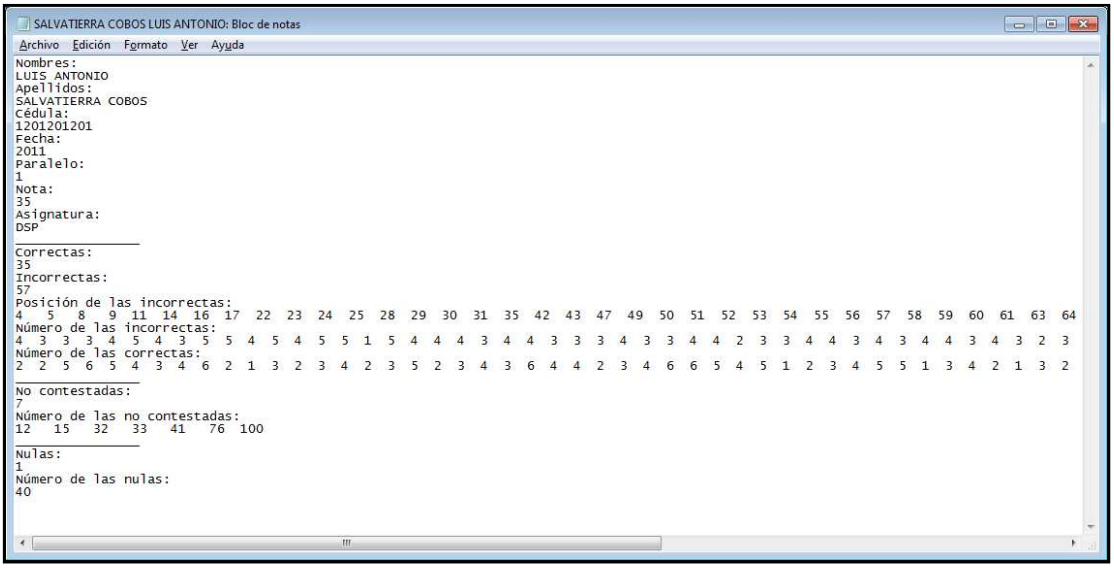

Fig. A-9. Vista del archivo txt donde se guarda la información del evaluado

Se realiza el mismo procedimiento cuando se quiera realizar una nueva evaluación a otro grupo de exámenes.

ANEXO B

CODIGO DE MATLAB

```
function varargout = calificador_multiopcion(varargin)
gui_Singleton = 1;clc
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @calificador_multiopcion_OpeningFcn, ...
                   'gui_OutputFcn',  @calificador_multiopcion_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

function calificador_multiopcion_OpeningFcn(hObject, eventdata, handles, varargin)
movegui(hObject, 'center')%Centrar GUI
set(handles.axes1,'XTick',[],'YTick',[])% Quitar etiqueta de los ejes del axes
set(handles.axes4,'XTick',[],'YTick',[])% Quitar etiqueta de los ejes del axes
handles.output = hObject;% Choose default command line output for calificador_multiopcion
guidata(hObject, handles);% Update handles structure

function varargout = calificador_multiopcion_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function CALIFICADOR_DE_EXAMENES_Callback(hObject, eventdata, handles)
function CALIFICADOR_DE_EXAMENES_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),...
   get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function cuadro_verde_de_correctas_valor_Nota_ResizeFcn(hObject, eventdata, handles)

function Valor_de_peso_de_nota_Callback(hObject, eventdata, handles)
% valor: 1
function Valor_de_peso_de_nota_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- INGRESA EL EXAMEN PATRON .
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function examen_patron_Callback(hObject, eventdata, handles)
% Seleccionar imagen con el examen (de formato JPG)
[nombre ruta]=uigetfile('*.jpg','Seleccionar examen patrón');
if nombre==0, return, end % Si se preciona el botón "Cancelar"
examen=imread(fullfile(ruta,nombre));% Leer la imagen como matriz
image(examen,'parent',handles.axes1)% Mostrar la vista previa de la imagen
set(handles.axes1,'XTick',[],'YTick',[])% Quitar ejes de axes
patron=examen(667:end,:,1:3);% Recortar la parte de nombre, fecha, etc
%% Binarización
binaria=~im2bw(patron,0.81);
% Encuadre (recortar la parte de los bordes)
% Encontrar los pixeles blancos
[f c]=find(binaria);
lmaxc=max(c);lminc=min(c);
lmaxf=max(f);lminf=min(f);
recortada=binaria(lminf:lmaxf,lminc:lmaxc);
% Filtrado (eliminar áreas menores a 10 pxl)
filtrada=bwareaopen(recortada,10);
% Dilatación
% Se usa una línea de 10 pxl de largo y 0º
se=strel('line',10,0);
% Dilatamos cada objeto de la imagen
juntar=imdilate(filtrada,se);
% En esta parte, segmentamos por columnas de forma se pueda eliminar las
% columnas con los números y dejar solo las respuestas.
re=juntar;
rec2=recortada;
n=0;
%% Segmentación
todas=[ ];
while 1
   % Recortar por filas
   [fl re rec1 rec2]=lines(re,rec2);
   % "n" es para eliminar las columnas impares que contienen a los números
   n=n+1;
   % Si es impar, realiza la siguiente iteración
   if mod(n,2)==1, continue, end
   % Rec1 es todo un bloque de respuestas
   % Enviamos a la función que retorna las respuestas seleccionadas
   las_resp=b_bloque_preguntas(rec1);
   todas=[todas las_resp];%Estas son las respuestas
   % El loop termina al terminarse los bloques de preguntas
   if isempty(re)
      break
   end
end
% Exportar a otra función las respuestas del examen patrón
handles.respuestas=todas;
% Actualizar la estructura handles de la GUI
```

```
guidata(hObject,handles)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- INGRESA Y EL EXAMEN A CALIFICAR .
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Examen_a_Calificar_Callback(hObject, eventdata, handles)

m=str2double(get(handles.desde,'String')); % asigno a m el numero de examen desde el cual
desea empezar a calificar
n=str2double(get(handles.hasta,'String'));% asigno a n el numero de examen hasta el cual se
califica
[nombre , ruta]=uigetfile('*.jpg','Seleccionar imagen');
for i=m:n %   para iniciar la calificacion desde el rango determinado.
nombre=['ex_',num2str(i),'.jpg'];
if nombre==0, return; end % Si se preciona "Cancelar"
examen=imread(fullfile(ruta,nombre)); % Leer imagen como matriz
image(examen,'parent',handles.axes4) % Presentar imagen en vista previa
set(handles.axes4,'XTick',[],'YTick',[])% Quitar ejes de la imagen
examen=examen(667:end,:,1:3);
set(handles.name_ex,'String', nombre);% Recortar parte del nombre, fecha, etc.

%%%%%%%%%%%%%%%
% Llamar imagen leída en la función anterior

binaria=~im2bw(examen,0.81);
%% Encuadre
[f c]=find(binaria);
lmaxc=max(c);lminc=min(c);
lmaxf=max(f);lminf=min(f);
recortada=binaria(lminf:lmaxf,lminc:lmaxc);
%% Filtrado
filtrada=bwareaopen(recortada,10);
%% Dilatación
se=strel('line',10,0);
juntar=imdilate(filtrada,se);
% En esta parte, segmentamos por columnas de forma se pueda eliminar las
% columnas con los números y dejar solo las respuestas.
re=juntar;
rec2=recortada;
n=0;
%% Segmentación
todas=[ ];
while 1
   % Recortar por filas
   [fl re rec1 rec2]=lines(re,rec2);
   % "n" es para eliminar las columnas impares que contienen a los números
   n=n+1;
   % Si es impar, realiza la siguiente iteración
   if mod(n,2)==1, continue, end
   % Rec1 es todo un bloque de respuestas
   % Enviamos a la función que retorna las respuestas seleccionadas
```

```
    las_resp=b_bloque_preguntas(rec1);
    todas=[todas las_resp];%Estas son las respuestas
    % El loop termina al terminarse los bloques de preguntas
    if isempty(re)
        break
    end
end
% Exportar a otra función
handles.contestadas=todas;
respuestas=handles.respuestas;

% Verificación de respuestas correctas
% Se resta, para si en el caso que sea 0, haya coincidencia
verificar=abs(respuestas-todas);
% Las iguales a 0 son correctas
[fila columna]=find(verificar==0);
% Exportar a otra función
handles.verificar=verificar;
% Contar número de respuetas correctas
num_resp_corre=length(columna);
% Escribir el número de respuestas correctas
set(handles.text6,'String',num_resp_corre)
% Obtener el Valor_de_peso_de_nota de cada pregunta
peso=str2double(get(handles.Valor_de_peso_de_nota,'String'));
% Caluclar y mostrar la nota de examen
nota=peso*num_resp_corre;
set(handles.text7,'String',nota)
% Exportar a otra función
handles.nota=nota;
handles.correctas=num_resp_corre;
% Actualizar estructura handles de la GUI
guidata(hObject,handles)
la_correcta=handles.respuestas;
respuesta=handles.contestadas;
% Estas son las no contestadas (en blanco)
[f_no_res c_no_res]=find(respuesta==-1);
num_no_contestada=length(c_no_res);
% Estas son las nulas (más de una opción)
[f_res_nu c_res_nu]=find(respuesta==0);
num_resp_nulas=length(c_res_nu);
% Número de incorrectas
verificar=handles.verificar;
% Estas son las incorrectas
[f_inc c_inc]=find(verificar~=0);
% num_incorrectas=length(c_inc);
% Ver no respondidas
opciones_incorrectas=respuesta(c_inc);
[f,c]=find(opciones_incorrectas==-1);%No ha respondido
% Las iguales a -1 (en blanco) se eliminan de las respuestas incorrectas
c_inc(c)=[ ];
opciones_incorrectas(c)=[ ];
% Ver nulas (son las iguales a 0)
[f,c]=find(opciones_incorrectas==0);%No ha respondido
```

```
% Las iguales a 0 (nulas) se eliminan de las respuestas incorrectas
c_inc(c)=[ ];
opciones_incorrectas(c)=[ ];
num_incorrectas=length(opciones_incorrectas);
% Mostrar un resumen del examen
%Exportar a la función que guarda en el archivo de texto
handles.incorrectas=num_incorrectas;
handles.pos_incorrectas=c_inc;
handles.opcion_marcada=respuesta(c_inc);
handles.opcion_correcta=la_correcta(c_inc);
%
handles.no_contestadas=num_no_contestada;
handles.pos_no_contestadas=c_no_res;
%
handles.nulas=num_resp_nulas;
handles.pos_nulas=c_res_nu;

% Actualizar estructura handles de la GUI
guidata(hObject,handles)
la_correcta=handles.respuestas;
respuesta=handles.contestadas;
% Estas son las no contestadas (en blanco)
[f_no_res c_no_res]=find(respuesta==-1);
num_no_contestada=length(c_no_res);
% Estas son las nulas (más de una opción)
[f_res_nu c_res_nu]=find(respuesta==0);
num_resp_nulas=length(c_res_nu);
% Número de incorrectas
verificar=handles.verificar;
% Estas son las incorrectas
[f_inc c_inc]=find(verificar~=0);
% num_incorrectas=length(c_inc);
% Ver no respondidas
opciones_incorrectas=respuesta(c_inc);
[f,c]=find(opciones_incorrectas==-1);%No ha respondido
% Las iguales a -1 (en blanco) se eliminan de las respuestas incorrectas
c_inc(c)=[ ];
opciones_incorrectas(c)=[ ];
% Ver nulas (son las iguales a 0)
[f,c]=find(opciones_incorrectas==0);%No ha respondido
% Las iguales a 0 (nulas) se eliminan de las respuestas incorrectas
c_inc(c)=[ ];
opciones_incorrectas(c)=[ ];
num_incorrectas=length(opciones_incorrectas);
%Mostrar un resumen del examen
% Exportar a la función que guarda en el archivo de texto
handles.incorrectas=num_incorrectas;
handles.pos_incorrectas=c_inc;
handles.opcion_marcada=respuesta(c_inc);
handles.opcion_correcta=la_correcta(c_inc);
%
handles.no_contestadas=num_no_contestada;
handles.pos_no_contestadas=c_no_res;
```

```
%
handles.nulas=num_resp_nulas;
handles.pos_nulas=c_res_nu;

% Actualizar estructura handles de la GUI
guidata(hObject,handles)
%=========================.....................
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
ingresar_datos; % Llamar GUI para ingresar datos del estudiantes
uiwait % Esperar hasta que se ingresen los datos
% Variables globales de los datos del estudiante (vienen de la otra GUI)
global nombre apellido fecha id paralelo asignatura %Variables globales
nota=handles.nota;% Llamar datos de la función anterior
correctas=handles.correctas;
%
num_incorrectas=handles.incorrectas;
c_inc=handles.pos_incorrectas;
respuesta=handles.opcion_marcada;
la_correcta=handles.opcion_correcta;
%
num_no_contestada=handles.no_contestadas;
c_no_res=handles.pos_no_contestadas;
%
num_resp_nulas=handles.nulas;
c_res_nu=handles.pos_nulas;
 % Guardar el resumen del examen en un archivo de texto
% Crear archivo de texto con el nombre del estudiante
fid = fopen([[apellido,' ',nombre],'.txt'], 'wt');
% Escribir en el archivo de texto
fprintf(fid,'%s\n','Nombres:');
fprintf(fid,'%s\n',num2str(nombre));
fprintf(fid,'%s\n','Apellidos: ');
fprintf(fid,'%s\n',num2str(apellido));
fprintf(fid,'%s\n','Cédula: ');
fprintf(fid,'%s\n',num2str(id));
fprintf(fid,'%s\n','Fecha: ');
fprintf(fid,'%s\n',num2str(fecha));
fprintf(fid,'%s\n','Paralelo: ');
fprintf(fid,'%s\n',num2str(paralelo));
fprintf(fid,'%s\n','Nota: ');
fprintf(fid,'%s\n',num2str(nota));
fprintf(fid,'%s\n','Asignatura: ');
fprintf(fid,'%s\n',num2str(asignatura));
fprintf(fid,'%s\n','_____________');
fprintf(fid,'%s\n','Correctas: ');
fprintf(fid,'%s\n',num2str(correctas));
fprintf(fid,'%s\n','Incorrectas: ');
fprintf(fid,'%s\n',num2str(num_incorrectas));
fprintf(fid,'%s\n','Posición de las incorrectas: ');
fprintf(fid,'%s\n',num2str(c_inc));
fprintf(fid,'%s\n','Número de las incorrectas: ');
fprintf(fid,'%s\n',num2str(respuesta));
fprintf(fid,'%s\n','Número de las correctas: ');
```

```
fprintf(fid,'%s\n',num2str(la_correcta));
fprintf(fid,'%s\n','____________________');
fprintf(fid,'%s\n','No contestadas: ');
fprintf(fid,'%s\n',num2str(num_no_contestada));
fprintf(fid,'%s\n','Número de las no contestadas: ');
fprintf(fid,'%s\n',num2str(c_no_res));
fprintf(fid,'%s\n','____________________');
fprintf(fid,'%s\n','Nulas: ');
fprintf(fid,'%s\n',num2str(num_resp_nulas));
fprintf(fid,'%s\n','Número de las nulas: ');
fprintf(fid,'%s\n',num2str(c_res_nu));
fclose(fid);% Cerrar escritura del archivo
pause(0.05)
end
pause(1);
msgbox('ha terminado de calificar','mensaje');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- CALIFICA EL EXAMEN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function calificar_Callback(hObject, eventdata, handles)

function figure1_CloseRequestFcn(hObject, eventdata, handles)
beep %emite un sonido de beeb
opc=questdlg('¿ Desea salir del programa ?', 'SALIR','Si','No','Si');
if strcmp (opc, 'No')
    return;
end
delete(hObject);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function opcion=ver_respuesta(fila_respuestas,mascara)
% Etiquetar objetos y contarlos
[L ne]=bwlabel(mascara);
% Obtener propiedades de cada objeto: área, caja y centro de masa.
prop=regionprops(L);
% medio=[ ];%Eliminar
medio=zeros(1,ne);%Eliminar
% Loop para ir midiendo el área de cada casilla
verificador=0;
opcion=-1;
for n=1:ne
   % Recortar de cada fila de respuestas solo la casilla
   casilla=imcrop(fila_respuestas,prop(n).BoundingBox);
   % Normalizar la casilla
   casilla=imresize(casilla, [20 13]);
   % Contar el número de pixeles en cada casilla
   area=sum(sum(casilla));
   % Almacenar en un matriz el valor anterior
   medio(n)=area;
   % Si el número de pixeles es mayor a 480 la casilla está llena
   if area>130
```

```
        % Suma 1 al verificador para analizar si hay solo una opción
        % seleccionada
        verificador=verificador+1;
        % Si "verificador" es mayor a 2, escribir 0 en "opción"
        if verificador>=2
            opcion=0;
            % Cerrar loop
            break
        end
        % Caso contrario, escribir el valor de la opción elegida
        opcion=n;
    end
    %  imshow(casilla)
    %  pause(0.5)
end

% medio, max(medio)
% figure(2),imshow(fila_respuestas)
% beep

% Verificar si mismo no ha respondido
if opcion==-1 % Valor de no haber respondido
    % Cálculo estadístico
    indicador=max(medio)-median(medio);
    % Si hay un valor significativos (hay respuesta)
    if indicador>100
        % Asignar la posición del valor máximo como la respuesta elegida
        [maximo opcion]=max(medio);
    end
end

function [fl re rec_1 rec_2]=lines2(mascara, recortada)
mascara=clip(mascara);
% recortada=clip(recortada);
[r]=size(mascara,1);
for s=1:r
    if sum(mascara(s,:))==0
        nm=mascara(1:s-1,1:end);%First line matrix
        rm=mascara(s:end,1:end);%Remain line matrix
        rec_1=recortada(1:s-1,1:end);%First line matrix
        rec_2=recortada(s:end,1:end);%Remain line matrix
        [fl rec_1]=clip2(nm,rec_1);
        [re rec_2]=clip2(rm,rec_2);
        %*-*-*Uncomment lines below to see the result*-*-*-*-
        %         subplot(2,1,1);imshow(fl);
        %         subplot(2,1,2);imshow(re);
        break
    else
        fl=mascara;%Only one line.
        re=[ ];
        rec_1=recortada;
        rec_2=[ ];
```

```
    end
end


function [fl re rec_1 rec_2]=lines(mascara, recortada)
% Divide el texto (en imagen) en líneas separadas.
% Recortar máscara
mascara=clip(mascara);
% Contar número de columnas
[r]=size(mascara,2);
for s=1:r
    if sum(mascara(:,s))==0 % La suma==0 da la separación entre columnas del texto
        nm=mascara(1:end,1:s-1);% Primer línea de la máscara
        rm=mascara(1:end,s:end);% Resto de la máscara
        rec_1=recortada(1:end,1:s-1);% Primera línea de la imagen sin dilatar
        rec_2=recortada(1:end,s:end);% Resto de la imagen sin dilatar
        % Recortar imagen (quitar bordes)
        [fl rec_1]  = clip2(nm,rec_1);
        [re rec_2] = clip2(rm,rec_2);
        % Termina el loop cuando se ha separado una línea
        break
    else
        fl=mascara;%Solo una línea.
        re=[ ];
        rec_1=recortada;
        rec_2=[ ];
    end
end

function las_resp=b_bloque_preguntas(rec1)
% ESTA función recibe todo un bloque de preguntas y retorna las respuestas
% seleccioandas
% Estructura para realizar apertura de la imagen
se=strel('line',6,0);
% Apertura de la imagen (eliminar pixeles espurios)
v=imopen(rec1,se);
% Estructura para realizar dilatación de la imagen
se=strel('rectangle',[9 1]);
d=imdilate(v,se);
% Loop para separar filas de respuestas y optener la respuesta seleccionada
n=0;
re=d;
rec2=rec1;
opcion=[ ];
while 1
    % Separar cada fila de respuesta y dejar el resto
    [fl re rec1 rec2]=lines2(re,rec2);
    n=n+1;
    if n<=2, continue, end
    % REC1: Aqui tengo cada línea de respuesta
    % Aquí verifico cual casilla está llena o vacía
    opcion=[opcion ver_respuesta(rec1,fl)];
```

```
    % Al terminar las filas de respuestas, sale del loop y retorna
    if isempty(re)
        break
    end
end
% Cambio de variable
las_resp=opcion;

function [imgnD]=clip(bn)
% Recortar bordes de la imagen de entrada
% _
% Encontrar pixeles en blanco
[f c]=find(bn);
% Encontar máximos y mínimos de los pixeles blancos
lmaxc=max(c);lminc=min(c);
lmaxf=max(f);lminf=min(f);
% Recortar la imagen
imgnD=bn(lminf:lmaxf,lminc:lmaxc);

function [imgnD,imgnO]=clip2(mascara,original)
% Recortar bordes de la imagen de entrada
% _
% Encontrar pixeles en blanco de la máscara (imagen dilatada)
[f c]=find(mascara);
% Encontar máximos y mínimos de los pixeles blancos
lmaxc=max(c);lminc=min(c);
lmaxf=max(f);lminf=min(f);
% Recortar imagen en base a la máscara
imgnD=mascara(lminf:lmaxf,lminc:lmaxc);
imgnO=original(lminf:lmaxf,lminc:lmaxc);
%%%%%%%%%%%%%%%%%%%%
function desde_Callback(hObject, eventdata, handles)
function desde_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function hasta_Callback(hObject, eventdata, handles)

function hasta_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function name_ex_Callback(hObject, eventdata, handles)

function name_ex_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

REFERENCIAS

[1] Gonzales Rafael & Good Richard, Digital Image Processing, Editorial Prentice Hall (2da edición), Enero 15 del 2002.

[2] Gutiérrez Alegre Enrique, Procesamiento Digital de Imagen: Fundamentos y Prácticas con MatLAB, Editado por Universidad de León Secretariado de Publicaciones y Medios Audiovisuales, 2004.

[3] Rodriguez Morales Roberto & Sossa Azuela Juan Humberto, Procesamiento y Análisis Digital de Imágenes, Editorial Rama, 2011.

[4] Cuevas Erick & Zaldívar Daniel & Pérez Marco, digital de imágenes con MatLAB y Simulink, AlfaOmega Grupo Editor, Septiembre del 2010

[5] MATLAB Mathworks Inc., “Image Processing Toolbox User’s Guide”

[6] Cuevas & Zaldivar, Visión por Computador utilizando MatLAB y el Toolbox de Procesamiento Digital de Imágenes, http://proton.ucting.udg.mx/tutorial/vision/cursovision.pdf, consultada mayo 2011 creada en 2009.

[7] http://www.slideshare.net/omarspp/imagen-morfologicas?src=embed

[8] García Víctor, Introducción al Procesamiento Digital de Imágenes, http://www.slideshare.net/IDVicMan/introduccion-al-procesamiento-digital-de-imagenes, creada en el 2009 consulta en enero 2011

[9] Anónimo, Procesamiento de Imágenes con MatLAB, http://www.slideshare.net/lonely113/procesamiento-digital-de-imgenes-con-matlab, consultada en marzo del 2011 creado en febrero 2011.

[10] Sanchez Omar, Operaciones Morfológicas y Conectividad, http://omarsanchez.net/opemorf.aspx, consultada en mayo 2011