UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Actualización y mejora de un sistema automatizado de lectura y corrección de pruebas de evaluación del alumnado.

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Collados Llovera, Pablo

Tutor/a: Rodrigo Peñarrocha, Vicent Miquel

CURSO ACADÉMICO: 2022/2023

Resumen

El proyecto actual es la continuación del proyecto ”Diseño e implementación de un sistema automatizado de lectura y corrección de pruebas de evaluación del alumnado”. Esta continuación consiste en la revisión, actualización, corrección de errores y desarrollo de nuevas herramientas con el objetivo de mejorar la funcionalidad del proyecto, su robustez frente a errores y facilitar su uso por parte del profesorado. El problema principal del proyecto anterior era que el programa no daba los resultados esperados al utilizarse en las nuevas impresoras del departamento debido a la fiabilidad de digitalización de las nuevas impresoras. Además, el programa no contaba con un sistema de corrección de los ficheros erróneos de forma que se tenía que hacer manualmente. Por otro lado, la migración del proyecto en Matlab a octave para que pueda implementarse en software libre no ha sido exitosa debido a la lentitud de octave, que hace que el programa sea inviable.

Resum

El projecte actual és la continuació del projecte ”Disseny i implementació d’un sistema automatitzat de lectura i correcció de proves d’avaluació de l’alumnat”. Aquesta continuació consisteix en la revisió, actualització, correcció d’errors i desenvolupament de noves eines amb l’objectiu de millorar la funcionalitat del projecte, la seua robustesa enfront d’errors i facilitar el seu ús per part del professorat. El problema principal del projecte anterior era que el programa no donava els resultats esperats amb les noves impressores del departament degut a la fiabilitat de digitalització de les noves impressores. A més, el programa no comptava amb un sistema de correcció dels fitxers erronis de manera que s’havia de fer manualment. D’altra banda, la migració del projecte en Matlab a octave perquè puga implementar-se en programari lliure no ha sigut reeixida a causa de la lentitud de octave, que fa que el programa siga inviable.

Abstract

The current project is the continuation of the project ”Design and implementation of an automated system for reading and correcting student evaluation tests.”This continuation consists of the review, updating, error correction and development of new tools with the aim of improving the functionality of the project, its robustness against errors and facilitating its use by teachers. The main problem with the previous project was that the program did not give the expected results when used on the department’s new printers due to the digitization reliability of the new printers. Furthermore, the program did not have a system for correcting erroneous files so it had to be done manually. On the other hand, the migration of the Matlab project to octave so that it can be implemented in free software has not been successful due to the slowness of octave, which makes the program unviable.

A mis padres y a mi tutor Vicent Miquel.

Índice general

Índice de figuras

Parte I

Memoria

Capítulo 1

Introducción

1.1. Resumen del Proyecto de Final de Carrera anterior

Con el paso del tiempo, surgen nuevas formas de procesamiento, distintas mejoras y actualizaciones que hacen necesaria la supervisión y actualización de todos los programas para garantizar su correcto funcionamiento. En el caso del Proyecto Final de Carrera ”Diseño e implementación de un sistema automatizado de lectura y corrección de pruebas de evaluación del alumnado”que desarrolló el alumno Tomás Escudero Montoya para el departamento de comunicaciones de la ETSIT hace ya más de 10 años y por lo tanto, este proyecto es una continuación natural de dicho proyecto.

El proyecto consiste en desarrollar un sistema de corrección de exámenes, no solo de tipo test sino también de desarrollo, y mediante la digitalización y procesado de unas plantillas diseñadas por el departamento y el alumno en conjunto, poder extraer la información necesaria para evaluar a los alumnos de forma automática.

El proyecto consta de 2 partes, la primera parte abarca todo el procesado, el reconocimiento de imágenes y la obtención de las notas. Toda la primera parte se realiza en lenguaje matlab. Por otro lado, la segunda parte consiste en el proceso de copia de los ficheros corregidos en formato tiff multipágina al espacio compartido de cada alumno para que así puedan visualizarlos los alumnos de manera sencilla en poliformat. Esta parte se realiza en el entorno de linux shell.

1.1.1. Proceso de corrección de exámenes en Matlab

El programa desarrollado en matlab consta de una GUI(Graphical User Interface), para comunicarse con el programa. En ella el usuario le proporciona al programa un fichero tiff multipágina con los exámenes digitalizados, le indica si el examen es de respuesta múltiple o de desarrollo, y por último la carpeta destino donde irán todos los archivos generados por el programa. Una vez el usuario ha proporcionado todo lo que el programa necesita, este se pone el marcha.

El programa procesa todas las páginas de una en una. Cada página puede estar en blanco y negro, en escala de grises o en color. El programa se encarga de convertir esa imagen en una imagen lógica o binaria, es decir, una imagen de unos y ceros, en caso de que esté en escala de grises o en color.

Una vez se tiene la imagen binaria es necesario encuadrarla, esto es debido a que a la hora de digitalizar cada hoja se puede escanear de forma ligeramente distinta. El proceso de encuadrado

se realiza con los 4 cuadrados situados en las cuatro esquinas de cada hoja. Se obtiene el centro de cada uno de los cuadrados y se compara con los centros que deberían tener cada uno en una imagen perfectamente escaneada, denominados puntos de control. Este tipo de comparación es un tipo de transformación geométrica que se denomina *Image Registration*, donde se busca alinear 2 imágenes distintas.

Una vez se tiene la imagen bien encuadrada se realiza el procesado por zonas. Una zona es la del modelo/problema del examen, otra es la de identificación del alumno, y la última es que contiene las respuestas. Cada zona está previamente establecida y abarca un rango 'XY' de píxeles, es por eso que es necesario el encuadrado.

Para extraer la información de valor de cada zona se realizan operaciones morfológicas para poder degradar toda la información no deseada y solo quedarse con lo rellenado(pintado) en la plantilla por cada alumno. El proceso de encuadrado permite la resta entre la plantilla rellenada por el alumno y la plantilla original de forma que solo se quede las marcas rellenadas por el alumno. Posteriormente se realiza un proceso de erosionado sobre la imagen, que al haber restado se encuentra invertida. El erosionado consiste en erosionar los colores, que en el caso de la imagen, al ser en blanco y negro, sería como eliminar el blanco. El proceso de erosionado permite eliminar así las áreas blancas que sean pequeñas y mantener las áreas blancas más grandes. De esta forma se consigue eliminar cualquier información no deseada que no se haya podido eliminar con la resta.

Una vez se ha limpiado la imagen, la zona se divide en filas y columnas, como si se tratase de una tabla, y a partir de ahí según en que posición de esa tabla se encuentre la marca pintada por el alumno el programa es capaz de obtener la información que desea. Un ejemplo de esto sería, en el apartado de identificación, el programa detecta que en la fila 1 y columna 7 hay una marca, automáticamente, el programa está programado para saber que el primer dígito del documento de identidad del alumno es el 6. De la misma forma se obtiene la información en la zona de modelo y de respuestas.

Como se ha mencionado anteriormente, el programa permite corregir exámenes tipo test, y exámenes de desarrollo. En los exámenes de tipo test el alumno indica el modelo, su identificación y las respuestas. Por otro lado, en los exámenes de desarrollo, el alumno sólo indica el número de problema y su identificación, y es el profesor el que posteriormente rellenará el apartado de nota. Para la corrección de los exámenes tipo test, es necesario que el profesor proporcione, ya sea mediante un archivo csv o tiff, las respuestas correctas de cada modelo del tipo test.

Para el caso de los exámenes tipo test, mediante una serie de operaciones matriciales entre las marcas de respuesta del alumno y las plantillas con las soluciones proporcionadas por el profesor se obtiene la nota de cada alumno. Para el caso de los exámenes de desarrollo, el programa simplemente identifica la nota que ha rellenado el profesor.

Una vez el programa ya tiene todo lo que necesita, por un lado crea un fichero tiff, que contiene la plantilla corregida, para cada alumno, y por otro lado va registrando la nota e identificación que obtiene en un fichero csv.

Posteriormente, el profesor se encargará de subir el fichero csv con las notas obtenidas a padrino, y el alumno podrá acceder a la plantilla corregida en su espacio compartido.

1.1.2. Copia de ficheros al espacio compartido

Una vez se han generado correctamente el fichero tif de cada alumno, es necesaria la copia de cada fichero al espacio compartido de la asignatura para cada alumno, de forma que se pueda subir el fichero tif a ella para que el alumno pueda revisar su examen de forma fácil.

Esto proceso se realiza de forma automática desde la línea de comandos de Linux o Bash mediante un script que se ejecutará.

1.2. Situación actual

Todo el proyecto anterior se realizó para que funcionara en el la fotocopiadora multifunción RICOH MPC 2500 del Departamento de Comunicaciones de la ETSIT. Durante los últimos años el departamento, ha adquirido nuevos dispositivos y el programa no funciona de forma correcta en ellos. Asimismo, nuevas actualizaciones de matlab han provocado también errores en el programa, como por ejemplo cambios en funciones integradas de matlab, o cambios en el entorno GUI design(guide) de matlab.

Matlab ya ha anunciado que no va a seguir desarrollando el entorno guide, y pretende sustituirlo por el entorno appdesigner que considera más intuitivo y mejor.

Por otro lado, el programa está diseñado para detectar errores, ya sean de máquina o humanos, pero existen una serie de errores humanos que el programa no puede detectar y que pasan desapercibidos por el profesorado hasta que llega el momento de subir las notas a padrino. Un ejemplo de esto es que, un alumno rellene bien el apartado de identificación, por lo que el programa no generará ningún error, pero que se equivoque escribiendo su identificación. Esto generará un archivo correcto pero con una identificación incorrecta, de manera que a la hora de copiar los ficheros al espacio compartido, el programa no podrá encontrar ningún alumno con esa identificación.

1.3. Objetivos

Los objetivos principales de este proyecto son:

- Proporcionar robustez al proceso de tratamiento y procesado de la imagen para que funcione no sólo con los nuevos dispositivos del departamento, sino con cualquier dispositivo futuro.
- Actualizar el código de Matlab para que funcione en las nuevas versiones.
- Migrar la GUI a entorno appdesigner de Matlab.
- Migrar el programa desarrollado en matlab a octave para que pueda funcionar en un entorno de software libre.

Los objetivos secundarios de este proyecto son:

- Facilitar al profesor la gestión de exámenes con errores.
- Mejorar la transferencia de ficheros a poliformat en dos aspectos:
    1. unificar en una rutina la transferencia y comprobación de que se ha copiado correctamente un archivo.
    2. Realizar en Windows dicho proceso para evitar al usuario tener que usar una máquina virtual de Linux que puede ser un inconveniente que rechace el uso del programa por nuevos usuarios.
- Adaptar el programa para mejorar el procesado de imágenes a color.

Capítulo 2

Actualización y ampliación del código

2.1. Compatibilidad con nuevas digitalizadoras

2.1.1. Metodología de la revisión

En el momento de introducción de dos nuevas impresoras en el departamento, se esperaba que con su uso, se facilitara el procesado y el uso del programa, sin embargo no fue así. Uno de los objetivos principales del proyecto actual ha sido encontrar la causa del problema y solucionarlo.

El programa de matlab consta de unas 40 funciones. A la hora de revisar el proyecto es importante construir una visión global del proyecto, tener clara la dependencia entre las distintas funciones que lo conforman para así, en caso de querer analizar un error o una parte concreta del programa, saber aislar las funciones necesarias para que no haga falta ejecutar el programa entero cada vez.

Para un análisis eficiente del código se ha hecho uso de la herramienta de matlab *pause execution* que permite poder parar la ejecución del programa en una línea concreta.

Una vez se entendió el código y el papel que desempeña cada función, lo que se hizo fue ejecutar el programa de forma general y después de ejecutarlo, se observó que no existía un error que provocara que el programa fallase sino que el programa no hacía lo que debía y se determinó que lo que causaba que no funcionase en las nuevas impresoras se debía encontrar en la parte de tratamiento de la imagen, en concreto, en los apartados de preprocesado y procesado, que son los que engloban el encuadrado, y todo el proceso de operaciones morfológicas para extraer la información.

Mediante representaciones gráficas de los distintos cambios que se iban realizando sobre de las imágenes se encontró que el motivo por el que el programa no funcionaba bien era, porque la resta no se hacía de forma correcta.

Como se ha mencionado en el capítulo anterior, se aplica una resta de la plantilla original sin rellenar con la plantilla rellenada por el alumno para así facilitar el proceso de erosión. Al superponer la plantilla base con la rellenada se observó que las imágenes no coincidían, lo que provocaba que la resta no fuera correcta. Como se ha mencionado, la razón por la que se realizaba la resta es para facilitar la erosión.

El reconocimiento de las marcas funciona por tamaño de áreas. Se analizaron miles de exámenes

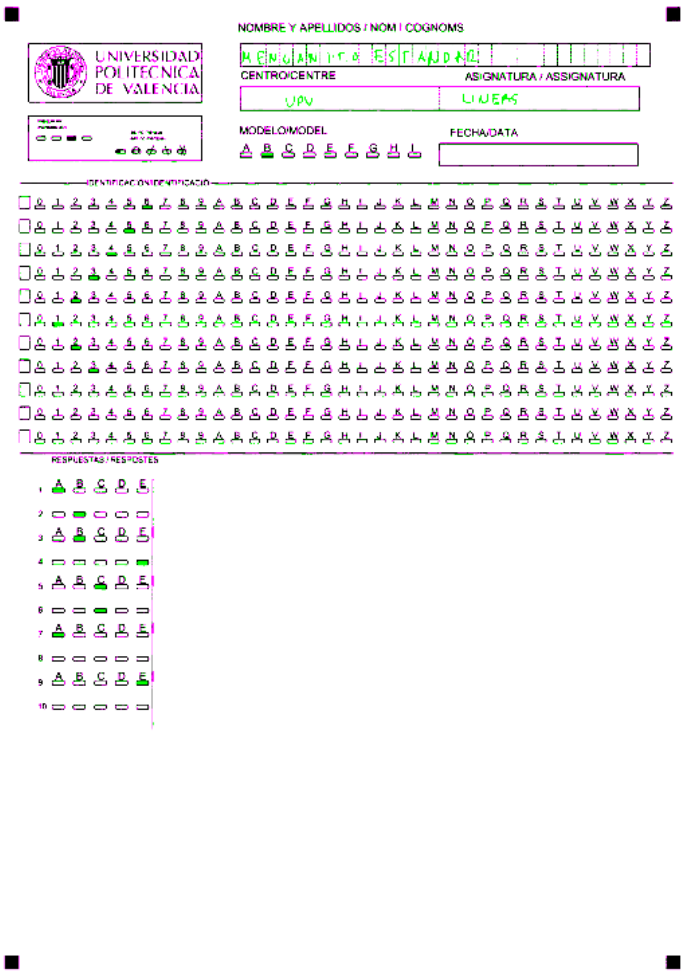

Figura 2.1: Plantilla con transformación geométrica óptima.

en el proyecto anterior y se determinó los rangos de tamaño de áreas más adecuados para el reconocimiento de cada área, de forma que incluso en exámenes donde hay marcas mal pintadas se puedan detectar las marcas. Por ejemplo, para la zona de identificación, se seleccionan áreas de entre 50 y 300. De forma que todas las marcas con esas áreas serán detectadas por el programa y procesadas. En un programa donde se realiza la resta de forma correcta solo quedan las marcas deseadas. Sin embargo, al realizar mal la resta, no se consiguen eliminar bien las marcas no deseadas (Refiriéndose a marcas no deseadas, a todo lo impreso en la hoja que no es rellenado por el alumno dentro de las casillas) y por lo tanto, todo el proceso de erosión preparado para una resta de forma correcta resulta insuficiente.

Una vez realizada la comparación se hizo evidente que el problema se debía encontrar en el encuadrado de la imagen, que originaba que no coincidiese con la plantilla original.

Como se puede observar en la figura 2.2 , donde se ven superpuestas la plantilla original con la rellenada, la desviación es de aproximadamente unos +(15-20) píxeles en el eje Y, es decir, una desviación de 15-20 píxeles hacía abajo y esto se observa de forma constante en todas las plantillas procesadas de varios ficheros tiff procesados con la misma digitalizadora.

En el capítulo anterior se ha mencionado, que para procesar las diferentes zonas que conforman

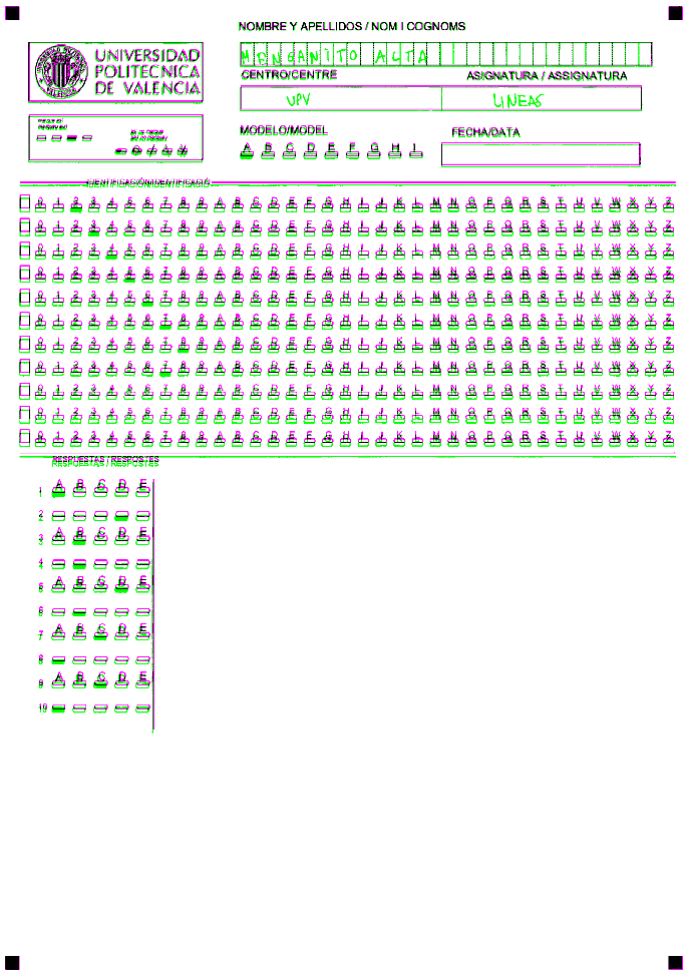

Figura 2.2: Plantilla con transformación geométrica errónea

una hoja como se ve en la figura 2.3, una vez se tiene acotada la zona se divide en una especie de tabla Figura 2.4. Estas zonas fueron debidamente acotadas en el proyecto anterior y para el caso de las zonas de identificación y respuestas/nota cada columna tiene una separación de unos 80 píxeles, por lo que una desviación de 15-20 no supone una diferencia significativa.

Dentro de cada celda de la tabla no se procesa todo el contenido. Como se parte de un proceso de encuadrado casi perfecto, es posible definir dentro de cada celda una zona que se puede llamar de reconocimiento para así poder minimizar al máximo cualquier posible marca no deseada que pudiera estar en la misma celda y que no se corresponda a la misma casilla marcada. Se decidió ampliar por lo tanto, la zona de reconocimiento de cada celda para que ocupe toda la celda, de esta forma se puede asegurar que tanto para las nuevas digitalizadoras del departamento como para futuras, si existe una ligera desviación, pueda detectarse igualmente, sin tener que alterar los rangos de creación de zonas que se establecieron en el proyecto anterior.

Una vez obtenida esta información se optó por dos posibles soluciones. Una primera más específica para las nuevas 2 digitalizadoras pero quizás menos robusta para posible futuras digitalizadoras, que consistía en eliminar la resta. Y otra más robusta que consistía en solucionar la raíz del problema que en su momento parecía encontrarse en el encuadrado.

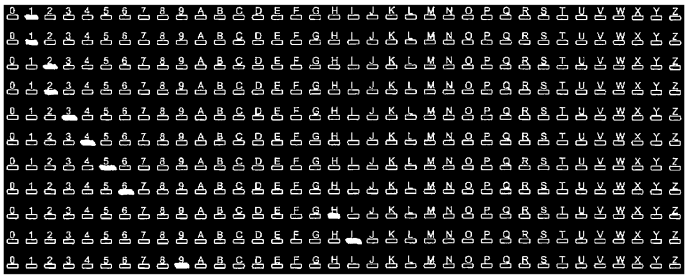

Figura 2.3: Ejemplo de la zona de Identificación de la plantilla.

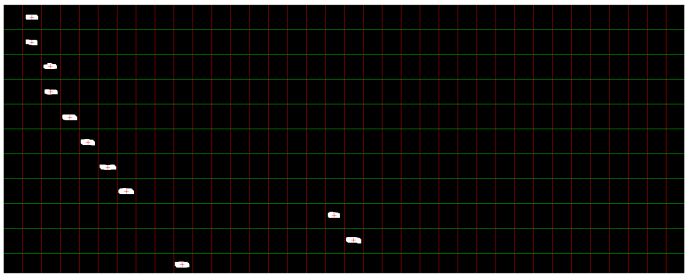

Figura 2.4: División en celdas de la zona de Identificación.

2.1.2. Primera opción: Eliminar la plantilla

La resta entre la plantilla original y la rellenada por el alumno no es más que un método que facilita el posterior proceso de aplicación de operaciones morfológicas. En esta opción se decidió eliminar la resta y por lo tanto era necesario una realización de operaciones morfológicas mucho más agresiva que con la resta ya que muchas marcas no deseadas ya no se eliminaban.

Figura 2.5: Erosionado de una plantilla con transformación errónea.

Como la imagen a procesar es una imagen binaria invertida, es decir, una imagen donde los blancos son negros y viceversa, es necesario hacer uno de la operación morfológica de erosión, que como se ha explicado brevemente en el capítulo anterior, consiste en degradar los blancos. Por otro lado, la dilatación es la operación morfológica que potencia los blancos

Una erosión más agresiva es inconsistente ya que provocaba que fuera necesario cambiar los rangos de área mencionados anteriormente. Era necesario encontrar la combinación correcta de dilatación y erosión. Aquí es donde entran el juego las operaciones de cerrado y apertura, que consisten en la alternancia de dilatación y erosión. En la apertura se realiza una erosión seguida de una dilatación, mientras que en el cerrado se empieza con una dilatación seguida de una erosión. La combinación

de una apertura, seguida de un cerrado, que sería realmente una serie erosión-dilatación-dilatación-erosión, es muy efectiva a la hora de eliminar ruido [1].

Para la plantilla rellenada se decidió usar un strel de tipo cuadrado. Un strel es el elemento con el que se realizan las operaciones morfológicas, y se fue variando el tamaño hasta encontrar el resultado que se consideraba más óptimo como se puede observar en la figura 2.6.

Figura 2.6: Nuevo proceso de erosionado de plantilla sin usar la resta de plantillas

Por lo tanto, como se confirmó que la desviación no era lo suficientemente grande para que resultara un problema de invasión de otras celdas, se determinó eliminar resta y realizar unas operaciones morfológicas más agresivas, y se consiguió obtener unos resultados idénticos a los obtenidos con la resta.

2.1.3. Segunda opción: Mantener la plantilla

La realización de la segunda opción permitía atajar la raíz del problema y prometía proveer una solución más robusta frente a problemas que surgiesen en cualquier tipo de digitalizadora futura.

Es necesario entender como funciona el proceso de transformación geométrica mediante puntos de control. Como se ha explicado en la introducción, consiste en obtener los centros de los 4 puntos de las esquinas y compararlos con unos puntos base que se corresponden con los de la plantilla original. Comparando las coordenadas de ambos puntos de manera afín, la función fitgeotrans de matlab devuelve la matriz 3x3 que contiene la transformación geométrica, que posteriormente se tendrá que aplicar sobre la imagen deseada. Esta matriz contiene información sobre el escalado, rotación, cizallamiento y traslación que se tiene que aplicar a la imagen para que se encuadre a la plantilla original y se pueda hacer la resta de manera correcta.

Primeramente se probó con diferentes tipos de transformaciones que permite la función fitgeotrans de matlab, pero rápidamente se descartó debido a que se obtenían resultados casi idénticos. Es provocó un cambio en la forma de abordar el problema ya que dejaba claro que aparentemente el problema no estaba en el encuadrado, y todo parecía apuntar a que parecía ser problema de la digitalizadora.

Con la ayuda del tutor se procedió a digitalizar de distintas maneras con la finalidad de descartar la posibilidad de que fuera un error a la hora de establecer los parámetros de la digitalización, como por ejemplo en la selección de la resolución, o tamaño de las imágenes escaneadas.

Una vez descartado que fuese un error en la parametrización se probó a digitalizar con diferentes orientaciones. El programa está preparado para que en caso de que una hoja esté girada 90 o 180 grados, se pueda orientar de forma automática.

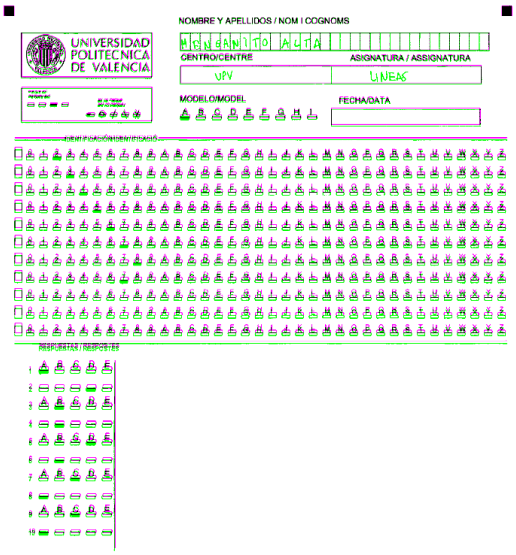

Figura 2.7: Imagen digitalizada con orientación normal

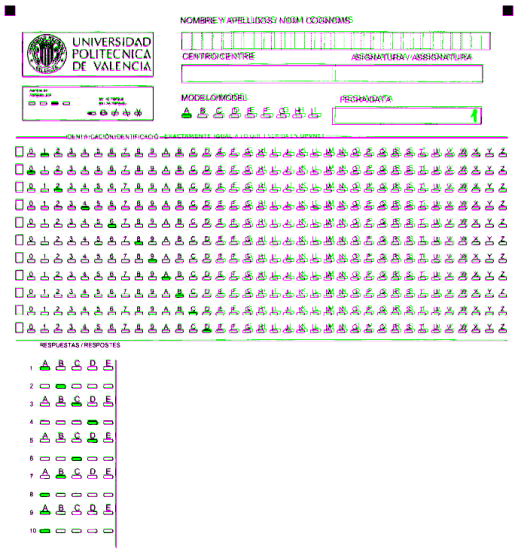

Figura 2.8: Imagen digitalizada con orientación 180 grados

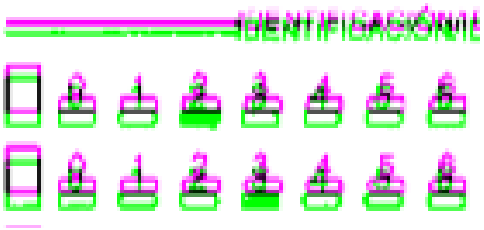

Figura 2.9: Ampliación Imagen digitalizada con orientación normal

Figura 2.10: Ampliación Imagen digitalizada con orientación 180 grados

Al probar con las hojas giradas 180 grados se descubrió que ahora la desviación en una zona concreta era de -(15-20) píxeles, es decir que el píxel estaba 15 píxeles arriba que su posición deseada, contrariamente a la imagen que no estaba girada 180 grados. Esto parecía indicar que quizás el problema radicaba en la velocidad de digitalización de la impresora, que provocaba que hubiese digitalización irregular de la hoja de forma que unas zonas no se digitalizaban de forma perfecta. Esto se podía comprobar ya que había algunas zonas donde la diferencia es de 15-20 píxeles y otras de un poco menos. Esta irregularidades también aparecían de manera más leve al digitalizar las hojas de manera horizontal.

Como el problema parecía residir en que la impresora no representaba de manera perfectamente fiel la imagen escaneada, y por lo tanto deformaba ligeramente la posición de las zonas. Por ello se decidió implementar la primera opción: Eliminar Plantilla.

2.2. Actualización del código Matlab

2.2.1. Revisión del código con herramientas de Matlab

La constante actualización del lenguaje matlab hace necesaria la revisión del código que tiene más de 10 años de existencia.

A lo largo del proyecto se ha hecho uso de varias apps integradas en matlab como *Code Analyzer*, *Dependency Analyzer* y *Profiler*.

Code Analyzer permite visualizar de forma sencilla todos las advertencias, errores o recomendaciones que hace matlab de todas las funciones que conforman el proyecto. Se ha ido sustituyendo y adaptando el código para que funcione con las funciones que han sustituido a las antiguas. Ejemplo de esto, es la sustitución de funciones como *im2bw*, *cp2tform*, *imtransform* entre otras.

Si bien no son cambios significativos de código, ya que solo generan *warning* o recomendaciones, se ha considerado la adaptación para evitar problemas en el futuro.

La app *Profiler* o perfilado en castellano, permite cronometrar todo el tiempo de ejecución del código y genera un informe. De esta forma se puede ver que funciones consumen más tiempo que otras. Hay funciones que son fundamentales, como *imread*, que permite leer los archivos tif o encuadrar, que se encarga del proceso de alineación con los puntos de control establecidos.

La función que más tiempo consumía en todo el tiempo de ejecución es la función *imfilter*, que permite eliminar ruido en imágenes en escala de grises. En este proceso, se tiene que ir analizando los píxeles y su vecindario, lo que requiere tiempo. Sin embargo en imágenes binarias de unos y ceros aplicar un filtro no tiene mucho sentido, ya que obtener valores intermedios en una matriz binaria no es conveniente.

Como se ha mencionado en el primer capítulo, el programa acepta imágenes en escala de grises, binarias y en color, sin embargo el tipo de imagen más común es la imagen binaria, es por eso que se decidió cambiar ligeramente la forma de preprocesar las imágenes en función de su tipo. De esta forma se consiguió reducir significativamente el tiempo de ejecución para imágenes binarias.

2.2.2. Error en la escritura de imágenes

Un cambio importante que esta vez si que originaba un error en las últimas versiones de matlab es el de escritura de imágenes lógicas. La función imwrite de matlab permite escribir un archivo imagen de la extensión indicada.

En el proyecto, a la hora de escribir la imagen corregida en un archivo tiff es necesario el uso de la función *imwrite* y de la compresión jpeg. La compresión jpeg de imágenes binarias no es óptima en términos de fiabilidad. Sin embargo en versiones más antiguas de matlab se permitía esta compresión sin que saltara ningún error. Aunque se desconoce a partir de que versión matlab se implementó, matlab decidió implementar un error específico para que ya no se pueda realizar la compresión jpeg sobre una imagen binaria. Si bien, el profesorado del Departamento de Comunicaciones no ha experimentado ningún problema al ejecutar el programa en versiones más antiguas, es necesaria la actualización del código para evitar sorpresas al instalar versiones más recientes.

Existen por lo tanto dos opciones principales para solucionar el problema. Por una parte se puede cambiar el tipo de compresión para que sea específica en blanco y negro o modificar la imagen

para que se pueda usar la compresión jpeg.

La compresión jpeg se usa para imágenes a color y en escala de grises, esto posibilita en un futuro se pueda trabajar con imágenes a color además de en blanco y negro. Por otro lado, existen compresiones como 'CCITT' o 'fax4' que son específicas para imágenes bitonales pero son menos eficientes. Es por eso que se ha establecido la compresión jpeg como la más adecuada ya que cambiar el tipo de compresión a una que permita binarias limitaría su uso futuro.

Las imágenes en escala de grises trabajan con valores de 0 a 255, donde el 0 es el negro puro y el blanco puro es el 255. Es tan sencillo como convertir la imagen binaria en una imagen *uint8* y multiplicar por 255, de forma que los negros se mantienen negros y los blancos que en una imagen binaria son unos, se mantienen blancos. Una vez se tiene la imagen en escala de grises, se puede realizar la compresión y escritura de manera eficiente.

2.2.3. Adición de fichero csv con la identificación del Alumnado

Un problema que se ha descrito en el apartado de situación actual del primer capítulo es el de la falta de preparación del programa para detectar algunos errores humanos. No son pocos los alumnos que se equivocan a la hora de rellenar su identificación.

Con el fin de adaptar el programa para evitar este tipo de situación el tutor propuso la opción de permitir introducir un tercer archivo de entrada que sería un archivo csv de dos columnas con la identificación y nombre de todos los alumnos matriculados de la asignatura. De esta forma, el programa durante su ejecución, aunque obtenga una cadena válida de identificación comprobará con el archivo csv si existe tal identificación y si no es así mandará el archivo a la carpeta de errores.

2.2.4. Migración del entorno GUIDE al entorno APPDESIGNER

Mathworks anunció que a partir de 2019 dejaría de actualizar el entorno guide para así centrarse en crear y desarrollar el entorno *App designer* [2], cuyo objetivo es ser más fácil e intuitivo para el usuario, además de ser más robusto. La sintaxis de appdesigner es similar a guide a nivel intuitivo pero requiere cambios en el código que pueden resultar tediosos y largos. Es por eso que Matlab permite la migración de forma fácil de guide a appdesigner mediante unas funciones que el propio matlab implementa.

En un principio, en vez de cambiar el código de forma completamente manual se decidió usar la herramienta proporcionada por matlab para fomentar que los usuarios de matlab se cambien a *app designer* de forma relativamente fácil. La herramienta revisa el código e indica al usuario que partes son compatibles y que partes no lo son. A partir de ahí, convierte el código compatible de forma automática mediante funciones como *convertToGUIDECallbackArguments*.

La GUI es una función que contiene funciones *child*. A parte de las funciones de inicialización y cierre, a cada elemento se puede asignar, lo que se denomina un *callback*. Un *callback* son una serie de acciones indicadas por el usuario que se realizarán como consecuencia de su activación, por ejemplo pulsar un botón. Cada *callback* de un objeto es una función distinta de forma que la información que se encuentra dentro de cada función es independiente.

La comunicación entre los diferentes objetos es indispensable para el funcionamiento de la GUI, es por eso que el entorno guide de matlab utiliza *handles* y *eventdata* para comunicar información

de utilidad entre los distintos objetos que forman la GUI. Por otro lado el entorno *App Designer* permite pasar información entre los diferentes objetos asignando propiedades de forma que todos los objetos puedan acceder a ellas o bien utilizando el prefijo app en lugar de *handles*.

Gracias a la herramienta de conversión de matlab, se pudo conseguir un código funcional compatible con *App Designer*, sin embargo, tanto la implementación de nuevas funcionalidades como la creación de otras aplicaciones se hizo directamente en el lenguaje que usa *App Designer*. Al realizar estas últimas partes de forma manual se determinó que el lenguaje que utiliza *App Designer* es más sencillo e intuitivo que el lenguaje que usaba GUIDE, por lo que se decidió convertir el código antiguo de forma manual para así homogeneizar todo el código y para que a su vez no se dependiera de tener que usar la herramienta de conversión.

2.3. Nuevas aplicaciones en APPDESIGNER

Además de la migración del entorno guide al entorno *app designer*, se han añadido nuevas funcionalidades a la interfaz gráfica. Una unidad gráfica, en entorno guide es realmente una figura o una ventana, y es posible comunicar varias ventanas entre sí llamándose entre ellas como si de funciones se tratasen. En entorno *app designer* por otro lado las figuras o ventanas pasan a ser una aplicación.

La aplicación de procesado, es decir, la interfaz gráfica migrada a *app designer*, se encarga de realizar todo el proceso de matlab a partir de varios archivos proporcionados por el usuario. La aplicación consta de varias secciones, cada una especializada en una funcionalidad concreta. Y puede comunicarse o llamar a otras aplicaciones para interactuar con ellas.

En este proyecto se han creado 2 nuevas aplicaciones, una aplicación de corrección de exámenes que han dado error a partir de la aplicación de Visualización anterior y otra aplicación que permite ver histogramas con las notas organizadas por modelo.

2.3.1. Aplicación de visualización de estadísticas

El profesor encargado del programa realizó durante años anotaciones sobre posibles mejoras que se pudieran implementar. Una de las mejoras trataba sobre el análisis de las notas, como por ejemplo, análisis de notas por modelo.

Se decide implementar por lo tanto, una nueva aplicación a la que se pueda acceder desde la aplicación principal de procesado. La aplicación consta de un desplegable donde se encuentran todos archivos csv con los modelos. Una vez se selecciona uno, es posible mostrar mediante un histograma las calificaciones para cada modelo.

Para poder disponer de las notas por modelo, se decidió añadir una nueva función al programa parecida a la existente de *generar csv*. La función *generar csv* crea un archivo csv donde va escribiendo las notas y la identificación del alumno que ha obtenido esa nota, pero no indica nada acerca del modelo de cada alumno. Es por eso que para implementar esta sección se creó una nueva función que generase un archivo csv por cada modelo que exista y vaya añadiendo las notas de los alumnos pertenecientes a ese modelo.

Una vez se tiene acceso a todos los ficheros csv en el desplegable, es necesario representar mediante un histograma la información de cada fichero. En un principio se decidió representar esta

información mediante un histograma pero debido a ciertas limitaciones que se experimentaron en entorno *app designer* se optó por representar la información en un gráfico de barras.

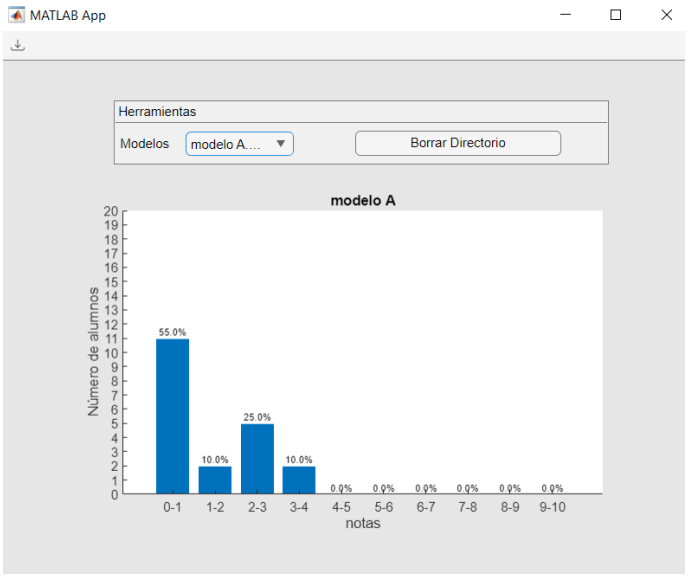

Figura 2.11: Ventana de la aplicación de Estadísticas

Un histograma muestra en el eje Y la frecuencia o el número de veces que se repite un valor dentro de un rango del eje X que en el caso de la aplicación era los diferentes rangos de notas de 1-10. Para hacer la información más visual se diseñó el histograma de forma que además de representar la frecuencia de cada intervalo, también mostrara el porcentaje con respecto al número total de notas de ese modelo. Las limitaciones surgen a la hora de manipular el eje Y, ya que así como matlab permite manipular el eje Y, el entorno *app designer* no lo permite. De esta forma se decidió utilizar un diagrama de barras para la representación. Utilizando las propiedades del histograma, se puede plasmar manualmente la información que se quiera representar en un diagrama de barras.

La generación de los ficheros csv con las notas de cada modelo, es de uso exclusivo para esta aplicación y por lo tanto para evitar que ocupe espacio en el ordenador del usuario, se ha implementado una sección que permite borrar la carpeta donde se encuentran los archivos csv de forma que si el profesor ya ha obtenido la información necesaria acerca de los exámenes tipo test, pueda borrar de forma rápida desde la propia aplicación los ficheros.

La función añadida para generar los csv con las notas por modelo solo es necesaria en los exámenes de tipo test, en los exámenes de desarrollo la nota final automáticamente se guarda en ficheros según problemas por lo que no hacía falta crear ninguna función. Es recomendable por lo tanto solo usar la opción de borrado en el caso de los exámenes de tipo test.

2.3.1.1. Programación de la aplicación

El programa se puede abrir desde la aplicación principal de procesado, o desde la propia aplicación. Para que funcione la aplicación, necesita la ruta donde se encuentran los archivos csv que tiene que mostrar, para ello, si se llama desde la aplicación principal, si se indica una carpeta de salida en la sección de carpeta de salida, la ruta de dicha carpeta será la que utilizará la aplicación de forma predeterminada. En caso de no introducir una carpeta o abrir la aplicación directamente, se

solicitará al usuario que indique la carpeta. Una vez se tiene la ruta de la carpeta se mostrarán todos los ficheros csv que existan en esa carpeta en una lista desplegable.

Los contenidos de cada csv se leen y se muestran en forma de diagrama de barras. Se utiliza la propiedad histcounts de un histograma para poder obtener los datos que se mostrarán en el diagrama de barras. Como se ha mencionado, se decidió usar un diagrama de barras en lugar de un histograma ya que era necesario modificar de forma activa los parámetros del histograma para obtener mejores resultados.

Una vez se han visualizado todos los archivos se permite al usuario descargarlos en formato png y tif. En el proceso, se van leyendo los archivos csv que hay en la carpeta mediante un bucle *for* y se van mostrando los diagramas de barras en los ejes de la aplicación. A este proceso se añade la exportación de la gráfica a formato png. La exportación de la gráfica se puede realizar de dos maneras, o bien escribiendo la gráfica como una imagen, en un archivo, o bien exportándola.

Al escribir la imagen en el archivo, cierta información como los ejes y sus títulos, que es información que está fuera de la gráfica, no se muestra y aunque de esta forma se puede ir escribiendo directamente en formato tif las diferentes gráficas que se van mostrando, las gráficas carecen de la información necesaria para interpretarlas.

Por otro lado, el exportar la gráfica directamente, tiene el inconveniente de que no permite guardar directamente en a formato tif, sin embargo, con este método se puede exportar toda la información de la gráfica, incluidos los ejes y sus títulos. Por lo tanto se decidió crear un paso intermedio de forma que se exporta la imagen a formato png, y a continuación se lee y escribe o adjunta en un archivo tif. La razón de uso del formato tif multipágina es que de esta manera el profesor podrá disponer de todas las gráficas en un mismo archivo en lugar de tener un archivo png por modelo.

Con el uso de esta aplicación, el profesorado podrá visualizar y descargar histogramas de las notas para cada modelo, que le servirán para detectar discrepancias o inconsistencias a la hora de crear los diferentes modelos de examen.

2.3.2. Aplicación de corrección de exámenes erróneos

En la aplicación principal existe una sección de visualización cuyo objeto de uso reside en poder cargar los ficheros erróneos y para así ver el error de forma rápida dentro de la aplicación. Al presionar el botón de visualización una vez se ha seleccionado el fichero a visualizar, la aplicación lleva a otra aplicación donde se puede ver el fichero de forma rápida. Si bien es una implementación que es útil, realmente solo se puede utilizar de forma efectiva si se desea corregir y visualizar los exámenes erróneos al finalizar el procesado de los exámenes, ya que si se desea hacer en otro momento, cuesta lo mismo abrir el programa y cargar los ficheros que ir a la carpeta del ordenador y abrirlos de forma manual. Es por eso que la sección de visualización ha sido muy poco utilizada por el profesorado a lo largo de la existencia del programa.

Por otro lado, la función de visualización permitía ver los ficheros erróneos pero el proceso de corrección de estos ficheros se seguía realizando de forma manual por parte del profesor.

Con el objetivo de dar uso a la sección de visualización y además implementar una herramienta de corrección automática se remodeló la aplicación visualización para así además de visualizar los ficheros erróneos, poder corregirlos de forma lo más automática posible. Por ejemplo, en caso de que sea un fallo del programa, o que debido a que los alumnos no han rellenado lo suficientemente

bien las marcas, no se registren bien los exámenes, siendo este último el origen más común de los errores.

En la versión antigua, en la aplicación de procesado existía una sección donde se seleccionaba la carpeta con los ficheros erróneos, se cargaba una lista y a la hora de visualizar, se seleccionaba un elemento de la lista y se abría la aplicación de visualización, donde se mostraba el elemento seleccionado. Esto significa que cada vez que se quiere ver un archivo, se tiene que cerrar la aplicación y volverla a llamarla presionando el botón de Visualizar en la aplicación de procesado. Es por eso que para la realización de la renovación de la aplicación de corrección de visualización que, de ahora en adelante pasará a llamarse aplicación de corrección de exámenes erróneos, se decidió eliminar de la aplicación de procesado la sección de visualización, y se añadió una versión similar dentro de la aplicación de forma que cada vez que se corrija un examen no haga falta salirse de la aplicación y volver a entrar.

La aplicación de corrección de exámenes consta de varias secciones, las secciones de corrección (marcadas en azul cian), 2 listas(marcadas en rojo) y una tabla y unos ejes como se puede ver en la figura 2.12. Una sección está destinada a la corrección de los exámenes de tipo desarrollo y la otra a la corrección de exámenes de tipo test. Por otro lado, una lista sirve para cargar los ficheros de la carpeta que se ha seleccionado, de la misma forma que se hacía desde la aplicación principal y otra muestra las páginas del archivo seleccionado. La tabla sirve para poder cargar la lista con los nombres de los alumnos que como se ha comentado anteriormente, servía para poder comparar la identificación y comprobar si existía un alumno con dicha identificación. Por último, el fichero seleccionado se mostrará en los ejes.

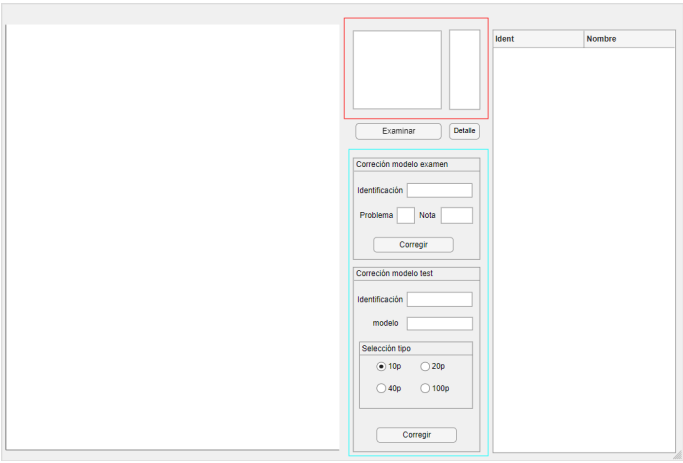

Figura 2.12: Ventana de la aplicación de Corrección de Exámenes erróneos

2.3.2.1. Secciones

Existe una sección para cada tipo de examen. Los procesos de corrección cambian ligeramente según el tipo de error y el tipo de examen, pero por lo general se utiliza una metodología similar, que se basa en la identificación de carpetas, en el traslado de archivos y en la escritura de archivos.

Para la identificación de las carpetas, normalmente se solicita al usuario que seleccione la carpeta. A lo largo de todo el proceso se necesita constantemente que el usuario seleccione archivos y carpetas lo cual puede resultar tedioso. Es por eso que mediante técnicas de manipulación de rutas

de archivo se puede conseguir identificar los mismos archivos sin necesidad de que el usuario tenga que estar indicándolo todo el rato. La manipulación se realiza aprovechando que todos los archivos, aunque se encuentren en subcarpetas distintas, se encuentran en la carpeta del examen, de forma que utilizando funciones de reemplazamiento de cadena de caracteres es posible alterar la ruta indicada.

2.3.2.2. Sección corrección de exámenes tipo test

Existen 4 tipos de errores a la hora de generación de los ficheros tiff corregidos:

- Error de identificación. Figura 2.13
- Error de modelo/problema. Figura 2.14
- Error de detección de los puntos de control, que ocurre cuando no se consigue encuadrar la imagen porque no detecta los puntos de control. Figura 2.15
- Error de nota que solo ocurre en los exámenes de desarrollo. Figura 2.16

Los errores de identificación y de modelo en exámenes de tipo test son 2 errores muy distintos y por lo tanto necesitan un proceso de corrección distinto.

Por un lado cuando se genera solo un error de identificación, se genera un archivo tiff en la carpeta de errores con el nombre incorrecto, pero como la nota se calcula de forma independiente a la identificación, si el modelo se ha introducido correctamente, la nota se guarda de forma correcta junto con la identificación incorrecta en el fichero csv de notas erróneas. El proceso de corrección consiste que el profesor indique en la casilla de texto editable la identificación correcta del alumno. El programa se encarga de identificar el fichero seleccionado, lo renombra con la información proporcionada y lo mueve a la carpeta de correctos. Por otro lado, identifica y extrae la nota correcta del fichero csv con las notas de ficheros erróneos y se copia junto a la identificación correcta en el fichero csv de la carpeta de correctos.

El proceso de corrección de errores de modelo es más complicado, ya que conlleva volver a ejecutar la parte del programa que permite calcular nota. Para evitar esto se optó por crear una función independiente que recoge los fragmentos de código necesarios del código para así obtener la nota una vez el profesor haya indicado el modelo correcto.

A diferencia del caso de error de identificación, cuando ocurre solo un error de modelo, se genera en la carpeta de correctos la plantilla correctamente identificada y en la carpeta de errores se guarda la plantilla mal corregida en un fichero tiff multipágina que contiene todos los ficheros con error de modelo. El programa se encarga de generar la nota para el modelo especificado por el profesor y de añadir esa página correctamente corregida al fichero tiff multipágina identificado en la carpeta de correctos.

Teniendo en cuenta las limitaciones de cada tipo de error, se desarrolló la aplicación de forma que si solo se introduce la identificación o si se introducen tanto la identificación como el modelo, se sigan procesos distintos.

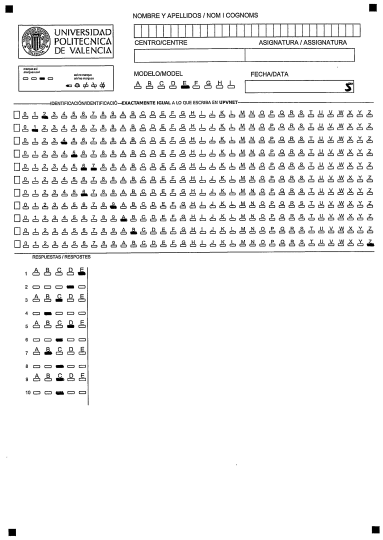

Figura 2.13: Error de Ident

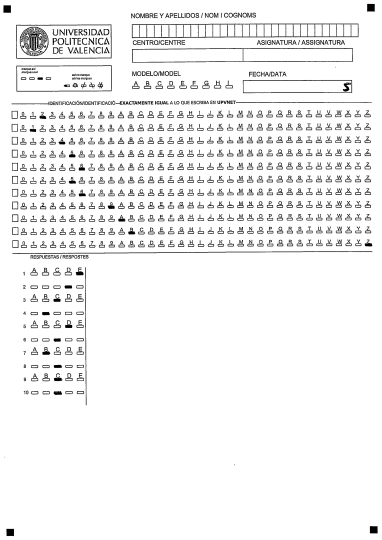

Figura 2.14: Error de Modelo

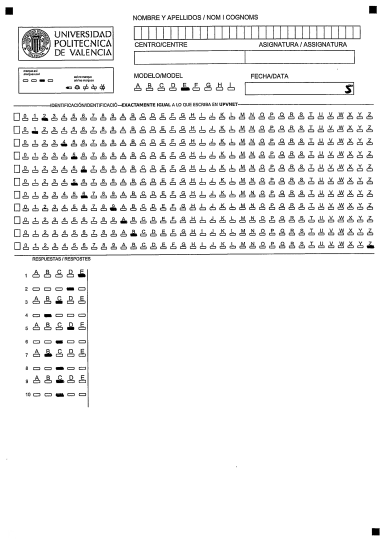

Figura 2.15: Error No detectado

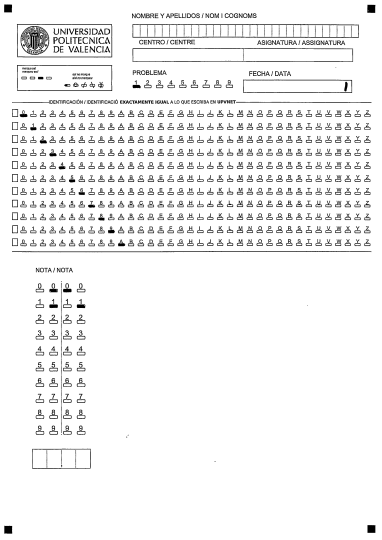

Figura 2.16: Error Nota

2.3.2.3. Sección corrección de exámenes tipo desarrollo

En el caso de los exámenes de desarrollo, se generan distintos tipos de ficheros csv con las notas. Por un lado se genera un csv por problema en la carpeta de correctos donde irán todos los alumnos y sus notas correspondientes en caso de que se hayan leído correctamente los 3 campos. En la carpeta de errores habrán ficheros csv por problema, para los casos de errores en la identificación o la nota y un fichero para los errores de problema.

Para este tipo de exámenes, la corrección en caso de error de problema no es tan difícil, en comparación con los exámenes de tipo test ya que no hay que calcular la nota en función del problema,

simplemente hay que leerla.

El programa en un principio estaba diseñado de forma que solo se tuviera que introducir el campo que es incorrecto, pero para evitar limitaciones y confusiones a la hora de tener que introducir los datos se decidió diseñar el programa para que funcionase en las 8 posibles combinaciones, resultantes de tener 3 campos editables.

2.3.2.4. Sección Lista de ficheros

Como se ha explicado anteriormente, el usuario ahora puede elegir la carpeta dentro de la aplicación de corrección. Una vez se ha seleccionado la carpeta, se cargan en una lista los nombres de todos los ficheros .tif que existen en la carpeta seleccionada, que deberá ser la carpeta de errores. En un principio se diseñó la lista de manera similar a la lista que existía en la aplicación de procesado, de forma que haciendo doble clic en cualquier archivo de la lista, se mostrara el fichero en los ejes de la aplicación. De esta forma el profesor puede ver el contenido de la página y escribirlo en cualquiera de las secciones para poder corregir el archivo. Este proceso requiere tener que ver e introducir a mano la identificación completa del alumno cada vez que se quiere corregir un fichero.

Con el fin de facilitar la escritura del fichero se programó la lista de forma que, al hacer doble clic en el elemento, además de mostrar en la gráfica el contenido del fichero imagen, se extraiga el título, como cadena de caracteres y se escriba automáticamente en los campos editables de identificación como se puede ver en la figura 2.18. De esta forma el profesor solo tiene que modificar el elemento de identificación que haya dado error.

En los exámenes tipo test. En la lista, además de los ficheros tif de la carpeta de errores también aparecen los ficheros de la carpeta de correctos que tienen solo 1 página, debido a que estos se corresponden con los ficheros con error de modelo. Estos ficheros aparecerán en la lista con la identificación correcta como se puede ver en la figura 2.17

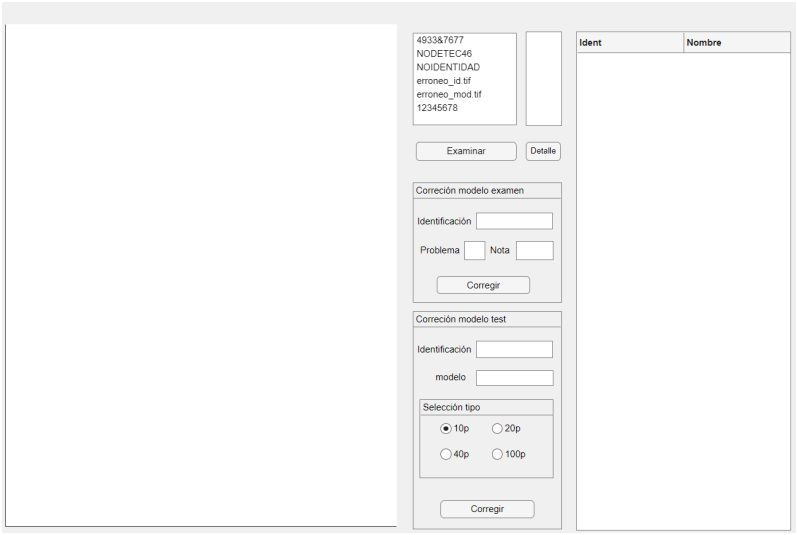

Figura 2.17: Ventana de la aplicación con ficheros cargados

2.3.2.5. Sección Lista de páginas

Si bien, en la primera página de cada fichero tif, se encuentra toda la información esencial, se ha diseñado la aplicación de forma que se pueda acceder a cada una de las páginas del fichero. De esta forma, en caso de que a un alumno se le olvide rellenar el número de problema en la plantilla de un examen de desarrollo, el profesor puede ver el contenido de las páginas y saber que número es.

Resulta también muy conveniente para la corrección de los errores de modelo en exámenes de tipo test, ya que todos los errores de modelo se añade a un único fichero.

2.3.2.6. Sección Lista de alumnos

Para facilitar al máximo el trabajo de corrección de errores, el profesor, puede facilitar un listado de alumnos con su identificador. De esta forma, se puede obtener una tabla con los nombres de los alumnos y sus identificadores. De esta forma, el profesor puede buscar en la lista de manera sencilla el nombre del alumno, o su identificación y así compararla de forma fácil con el nombre del fichero erróneo. En la tabla, haciendo doble click también se puede copiar el contenido del elemento seleccionado a el campo editable de identificación.

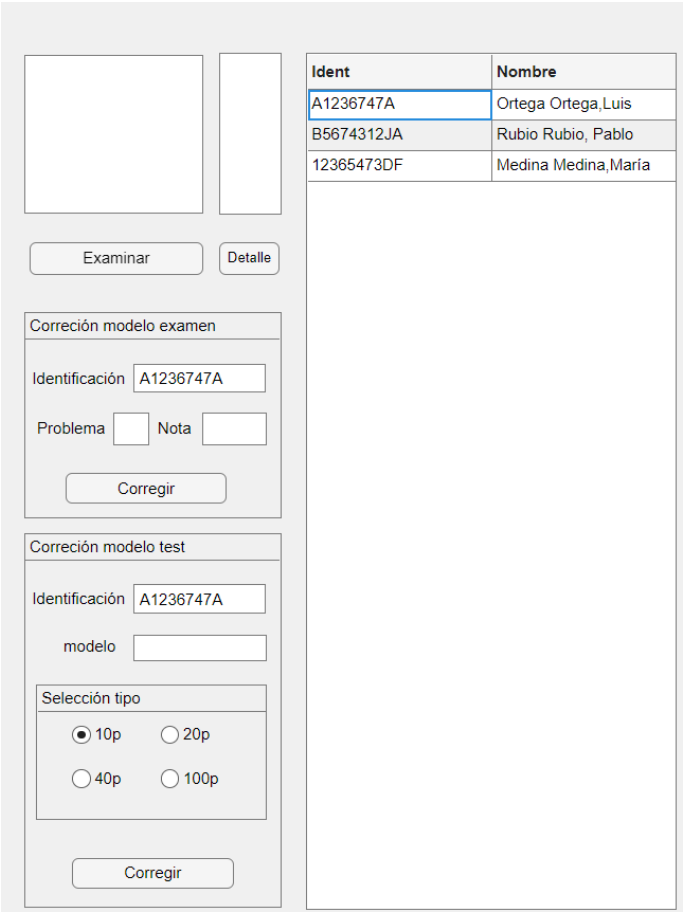

Figura 2.18: Ventana con la lista de nombres cargados

2.3.2.7. Programación de la aplicación

Una vez se carga la aplicación, lo primero que hay que hacer es examinar la carpeta de errores. Después de seleccionarla, se cargan los nombres de los ficheros tif en la lista de ficheros. El nombre de cada fichero pasa por una serie de modificaciones para eliminar el formato y solo quedarse con la cadena de caracteres que se debería corresponder con la identificación. El programa está diseñado para que cuando el usuario seleccione un archivo tif que no es individual, es decir los tif que recopilan todos los errores de identificación o de modelo, no se escriba nada en los campos editables figura 2.20.

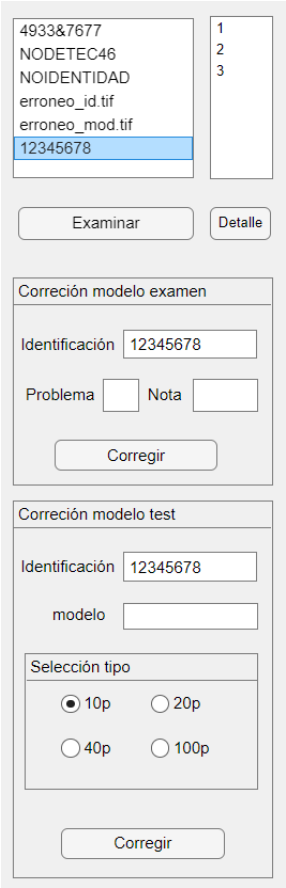

Figura 2.19: Campos editables rellenados automáticamente

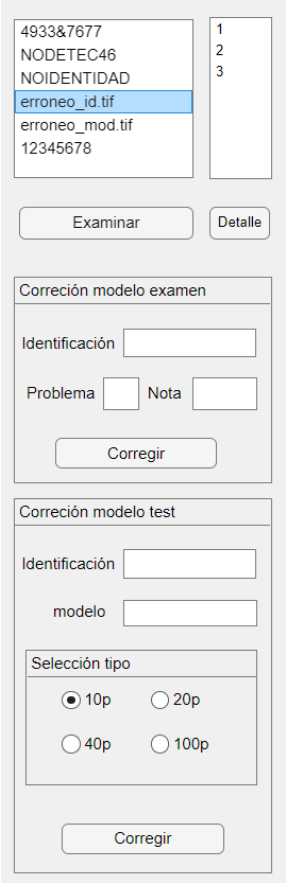

Figura 2.20: Campos editables no rellenados

En caso de que se seleccione un fichero erróneo de un alumno, la ruta del fichero que se deberá editar coincide con el fichero a modificar, sin embargo si se quiere trabajar desde un fichero tif de recopilación de errores, el usuario deberá indicar el fichero que quiere editar como se ve en el diagrama 2.21.

En caso de que el profesor quiera cargar también la lista con los nombres de los alumnos, está se cargará a la vez que la lista de ficheros. El programa está diseñado para que busque automáticamente el fichero csv dentro de la carpeta del examen. Como en teoría en la carpeta del examen solo existen las carpetas de Correctos, Errores, Modelos y el fichero csv, es fácil encontrar el fichero

Figura 2.21

csv que contiene el nombre de los alumnos. En caso de que esté en otra carpeta, el programa pedirá al usuario que le indique el fichero.

Una vez se ha seleccionado el fichero y se ha mostrado en los ejes, el profesor deberá introducir los campos que considere necesarios para su corrección y ejecutará el proceso de corrección.

El proceso de corrección funciona de la siguiente manera, el programa tiene guardada la ruta de la carpeta y la ruta del fichero a corregir. Con esta información se modifica la ruta del fichero cambiando en la cadena de la ruta la carpeta de Errores por la de Correctos y el nuevo nombre que se construirá a partir de la información introducida por el alumno, y posteriormente mediante la función *movefile* de matlab se enviará a la carpeta de correctos.

Una vez el archivo ya se ha renombrado y se ha movido de carpeta de correctos, es necesario obtener la nota para así escribirla en el fichero csv de la carpeta de correctos. Para obtener la nota del examen erróneo se utiliza el nombre del archivo original, se lee el fichero csv de la carpeta de errores y mediante comparación, se busca la nota para la identificación del fichero, ya que al igual que el fichero csv de correctos, el fichero csv de erróneos contiene la nota y la identificación incorrecta que coincidirá título del fichero tif incorrecto. De esta forma se extrae la nota y se escribe de forma automática en el fichero csv de la carpeta de correctos junto con esta vez, la identificación correcta proporcionada por el usuario.

Como se ha mencionado anteriormente, cuando se produce un error en el modelo en un tipo test, pero la identificación es correcta, el archivo plantilla se crea bien en la carpeta de correctos, y el archivo mal corregido se añade al fichero tif de errores de modelo en la carpeta de errores. Es por eso que los tif con errores de modelo tienen la peculiaridad de que solo tienen una página mientras que los ficheros correctos tienen 2, una con las marcas de corrección y la plantilla original. De esta forma, cuando se cargan inicialmente los ficheros en la lista al examinar la carpeta de errores, el programa, analiza todos los archivos de la carpeta de Correctos, aísla los que solo tienen una página y los añade a la lista de ficheros de la aplicación. De forma que el profesor puede tener también el nombre de los ficheros con error de modelo aunque solo pueda editarlos desde el fichero tif de recopilación de errores de modelo.

Para la corrección de errores de modelo en exámenes tipo test, la nota se calcula de manera automática y se escribe en el fichero csv de la carpeta de correctos.

Por otro lado, el proceso de corrección de exámenes de desarrollo, al ser diseñado para que funcione con las 8 combinaciones de argumentos de entrada, tiene diferentes implementaciones.

Lo más costoso de revisar y escribir es la identificación, lo cual no supone ningún problema debido a los mecanismos que se han desarrollado para que se copie de forma automática en el campo deseado, de forma contraria, es muy rápido para el profesor indicar el número de problema y la nota, por lo que la forma más eficiente y rápida de ejecución es introduciendo los 3 campos. Esto se debe a que como se ha mencionado anteriormente, para un error de identificación la nota se

genera de manera correcta en el fichero csv de errores y hay que extraerla luego, pero en este tipo de exámenes, si se especifica el problema y la nota directamente, se puede escribir directamente la nota en el fichero de correctos sin necesidad de buscarla en ningún fichero. De todas formas, el programa está preparado para que en caso de que no se especifique el problema o la nota, se pueda buscar en dichos ficheros. Este proceso se muestra en el diagrama de la figura 2.22.

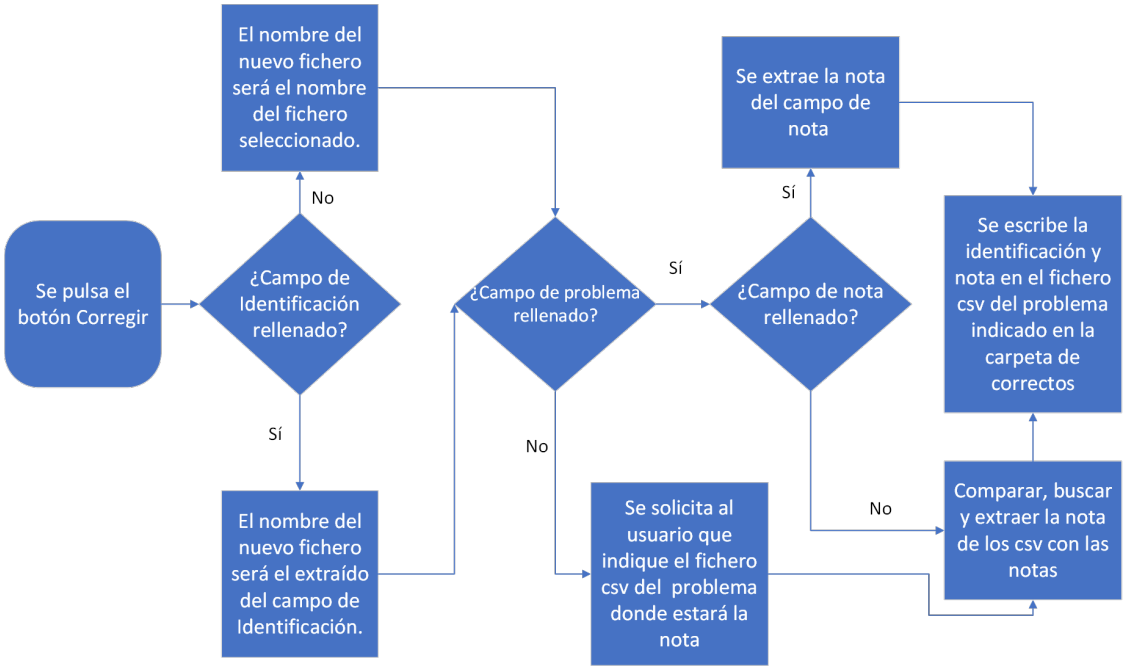

Figura 2.22: Proceso de decisión de Corrección de exámenes de desarrollo

Para terminar, es importante entender el funcionamiento de la tabla con los nombres de los ficheros. Del listado de alumnos realmente solo interesan aquellos que no tienen un fichero tif generado. Es por eso, que una vez se lee el fichero csv con la lista, antes convertirla en una tabla, se realizan una serie de operaciones sobre ella. Para ello se buscan en la carpeta de correctos, todos los ficheros tif generados y se guardan en una lista. Una vez se tiene la lista con los nombres de los ficheros tif generados y una lista con la identificación de todos los alumnos, utilizando la negada de la función ismember de matlab se obtienen los índices de todos los elementos de la lista de alumnos que no aparecen en la lista de tif generados. De esta forma se pueden aislar todos los exámenes que no se han corregido bien y generar una nueva lista. Y por lo tanto, aplicando los índices obtenidos a la lista de alumnos, una vez se cargue la lista solo aparecerá la identificación y el nombre de los alumnos que no tienen fichero corregido. Estos alumnos pueden ser, o bien alumnos cuyo fichero se encuentra en la carpeta de errores o bien alumnos que no se han presentado. De esta manera, se pueden filtrar de la lista todos los datos no deseados.

2.4. Conversión de formato TIF a formato PDF

Una de las principales anotaciones que realizó el tutor a tener en cuenta en la actualización, fue la posibilidad de convertir los ficheros tif a pdf para ser subidos al espacio compartido poliformat. El problema del formato tif, es que muchos dispositivos móviles no tienen un lector de imágenes predeterminado capaz de leer en formato tif multipágina, y los alumnos desconocedores de esto, envían correos al profesorado diciendo que los archivos están mal subidos. Por lo tanto, convertir los archivos a PDF solucionaría el problema.

Se barajaron 2 alternativas de conversión. La primera consistía en utilizar varias librerías de java existentes en matlab para poder aplicar la conversión directamente en matlab. La segunda consistía en utilizar librerías desde la línea de comandos.

2.4.1. Conversión a PDF en Matlab

Para poder convertir de formato tif a una imagen son necesarias las librerías apache PDFBOX de oracle y javax, ambas integradas en la ruta estática de clases de matlab, por lo que se pueden importar directamente.

La implementación de esta alternativa fue descartada rápidamente por varias razones La primera se debe a que el método *getImageReaders* de la clase javax no tiene un conversor específico para imágenes de formato tif. Por lo que se hace imposible leer una imagen de un fichero tif y obtener una *buffered image*. Es posible que exista una manera alternativa de generar una *buffered Image* que es un objeto de java, pero probablemente se requieran conceptos muy avanzados de programación en Matlab y java. Otra razón, es que al implementar la conversión dentro del proceso, hace más complicada la corrección de los exámenes en la aplicación diseñada al crearse los ficheros en PDF en lugar de tif.

La última razón, que se descubrió posteriormente al utilizar la herramienta de matlab *Code Analyzer*, es que matlab ya ha anunciado que las librerías de java integradas desaparecerán en las futuras versiones y será necesario añadir manualmente los paquetes o clases a la ruta estática.

2.4.2. Conversión a PDF en línea de comandos

La otra alternativa era el uso de la línea de comandos. A la hora de ejecutar el script de comandos se pueden utilizar librerías que realizan la conversión de forma automática. En el Cuarto capítulo se explicará en profundidad todo el proceso de conversión de PDF en línea de comandos de Ubuntu.

El profesor Mariano Baquero ya ideó una forma de convertir los ficheros tif a PDF en línea de comandos utilizando la función *tiff2pdf* de la librería tiff-tools de Linux. Sin embargo a la hora de probar esa función, aparecía un error de tamaño máximo excedido. Esto es un bug que aparece a veces y que no se ha corregido principalmente debido a la falta de mantenimiento de la librería, ya que es una librería vieja y en desuso.

Se decidió probar, por lo tanto otras alternativas, como por ejemplo la librería *imagemagick* o *graphicsmagick*. Después de probar ambas se determinó usar la librería *graphicsmagick* ya que daba menos errores, aunque se puede usar perfectamente *imagemagick*.

Capítulo 3

Migración a Octave

3.1. Situación

Otra de las anotaciones mencionadas por el profesorado fue la de trasladar todo el proyecto escrito en lenguaje matlab a un lenguaje alternativo que fuese de software libre. Dentro de los posibles candidatos se encontraban lenguajes como C++, python u octave.

Matlab es un software de pago, y aunque la universidad proporciona una licencia de forma gratuita para su uso por parte de alumnado y profesorado, es un factor limitante a la hora de querer continuar el trabajo en casa o desarrollar en el futuro una versión que pueda ser utilizada no solo por el departamento sino, a un nivel más amplio.

Después de revisar todas las opciones se optó por octave debido a su alta compatibilidad, lo que permitía una migración más sencilla y a la vez más intuitiva en contraposición a tener que usar un lenguaje completamente distinto como python o C++.

Octave [3] al ser un lenguaje gratuito no cuenta con muchas de las herramientas de matlab, ya sea paquetes o distintas funcionalidades, por lo tanto, antes de realizar la migración fue necesario cerciorarse de que además de poder adaptar el código de matlab a octave, también tuviera acceso a librerías alternativas que hicieran posible que funcionase.

3.2. Proceso de migración

El proyecto es un programa de procesado de imágenes que utiliza *image processing toolbox* un *add-on* de matlab para realizar las operaciones morfológicas y demás funciones integradas de matlab que forman todo el tratamiento y procesado de la imagen. El equivalente a IPT de matlab en octave es el paquete *image*. Este paquete tiene ciertas limitaciones como por ejemplo, la optimización, o el número de formas de strel.

En el proceso de erosión del proyecto anterior, se utilizaba strel de tipo disco y línea. En el paquete *image* de octave no existe la forma de tipo disco, es por eso que se decidió modificar el proceso de erosionado en octave. Otra limitación que se experimentó en octave, fue que la función que realiza la transformación geométrica (imtransform) no acepta imágenes binarias , lo que provoca que se tenga que convertir la imagen a tipo double para poder ser encuadrada.

En este caso, se fue ejecutando código con la misma metodología que se explica en el capítulo 2, y se fue adaptando para que funcionase en octave.

3.3. GUI

En general, todo el proceso de migración de matlab a octave fue relativamente cómodo debido a la alta compatibilidad que existe entre ambos lenguajes de programación, sin embargo, una limitación muy importante que marcó la diferencia a la hora de migrar el código, fue que octave no dispone de un entorno guide o *app designer* como matlab.

Como se explicó en el capítulo 2, el entorno guide o appdesigner es una herramienta que permite al usuario poder construir una ventana de diálogo o figura con la que comunicarse con el programa. Esto se puede realizar de forma manual en matlab construyendo las funciones, y parametrizando todos los objetos manualmente. Para una ventana sencilla, es algo que se puede realizar, pero para construir aplicaciones más complejas resulta muy largo y tedioso. El uso del entorno guide o *app designer* resulta la mejor opción cuando se quiere construir una aplicación ya que simplifica enormemente la cantidad de código y el tiempo que el usuario tiene que emplear para conseguir una aplicación funcional.

La inexistencia de una alternativa a guide en octave dejaba como única alternativa para construir la aplicación, su realización de forma manual. Sin embargo, se encontró una prometedora solución, una herramienta creada por Sergio Burgos, llamada GuiEditor, desarrollada como un software de terceros compatible con octave que permitiría construir una aplicación de forma similar al entorno guide de matlab.

GuiEditor[4] sería el equivalente al entorno guide de matlab. Presenta una interfaz más sencilla y con menos funcionalidades, pero dispone de todos los elementos esenciales para la aplicación a desarrollar. A diferencia de guide, y aunque se puede sincronizar con octave, no está integrado en octave de forma directa por lo que la aplicación o ventana de diálogo se crea como un proyecto. Una vez se ha acabado el proyecto, es necesario exportarlo como paquete que se tendrá que instalar en octave de la misma manera que cualquier otro paquete como image. A la hora de construir la

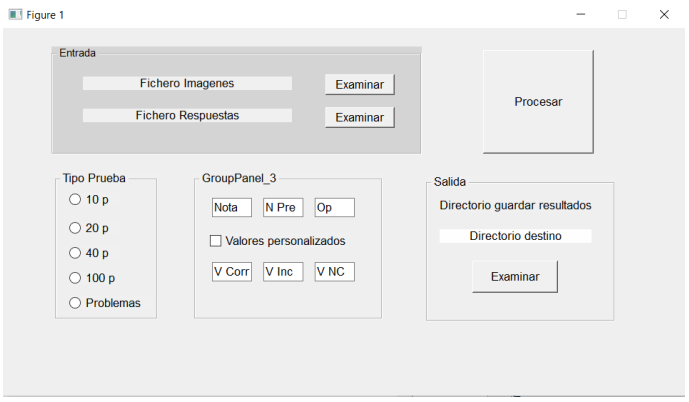

Figura 3.1: Ventana de GUI de octave.

primera versión de la ventana de diálogo se decidió solo mantener las secciones esenciales.

Al ser un software poco utilizado y extendido, no hay gran cantidad de documentación sobre su

uso. Existen tutoriales creados por el propio desarrollador, donde se explican sus funcionalidades de forma superficial, pero que resultan insuficientes a la hora de desarrollar una aplicación compleja.

Como se explicó en el capítulo 2, la comunicación de diferentes objetos que conforman la ventana se realiza a través de *handles*. Los *handles* son un *struct* que almacena la información para que pueda ser utilizada por otros objetos. Sin embargo, en guiEditor, no existe una alternativa. Existe el parámetro de entrada llamado data que permite comunicar información, pero no puede almacenarla en *struct* de forma, que no permite guardar diferente información de distintos objetos.

Para solucionar este problema se decidió hacer uso de variables globales. Al ser globales, todas las funciones de una misma clase, pueden acceder a ellas. Las variables más importantes que se establecieron como globales, fueron las rutas de acceso a los diferentes archivos de entrada y la carpeta de salida. Y de esta forma se consiguió implementar el entorno en octave de manera muy similar al entorno GUIDE de Matlab.

3.4. Pruebas del programa

Una vez se terminó el diseño e implementación de la GUI y se adaptó todo el código, comenzaron las pruebas para comprobar la fiabilidad de octave.

Los resultados fueron decepcionantes, unos 9-10 segundos por imagen para los exámenes de tipo test y 4-6 segundos por imagen para los exámenes de desarrollo. Además durante el proceso de generación de marcas se redimensiona la imagen a la mitad para poder introducir bien las marcas, y aunque en Matlab no hay problema, en octave se perdía mucha calidad.

Para poder localizar las funciones que consumían más tiempo, se utilizó la herramienta *profiler* de octave, muy similar pero peor que la de matlab.

La herramienta *profiler* cronometra el tiempo que tarda cada función en ejecutarse y genera un informe en formato html. De esta forma se puede ver la jerarquía de cada función e investigar de forma profunda cuales son las que consumen más tiempo.

Function	Time (s)	Time (%)	Calls
imresize>conv_interp_vec	22.561	13.56	104
convn	18.498	11.12	225
__magick_read__	17.222	10.35	25
binary .*	14.565	8.75	4338
__magick_write__	13.344	8.02	52
lookup	11.405	6.85	250
imdilate	11.405	6.85	150
interp2	10.837	6.51	25
binary -	4.456	2.68	15775
binary *	4.131	2.48	15187
generar_marcas	3.625	2.18	25
binary +	2.504	1.50	30251
conv2	2.340	1.41	25
binary >=	1.927	1.16	1144
repmat	1.893	1.14	452
imtransform	1.882	1.13	25
maketform>inv_affine	1.586	0.95	25
zeros	1.516	0.91	1334

Figura 3.2: Perfilado del proceso de digitalización de exámenes de tipo test en octave.

Los resultados en los exámenes de respuesta múltiple fueron los siguientes:

En el proceso de corrección de exámenes tipo test, las funciones que emplean mayor tiempo son preprocesado, la función de redimensionamiento y la de escritura. Dentro de la función *preprocesado*, las funciones de octave que más tiempo consumen son *imtranform*, que se encarga de la transformación geométrica, *imread*, que se encarga de leer el fichero tif y *bwmorph* que es la función que aplica operaciones morfológicas sobre la imagen.

Si comparamos con los resultados del perfilado obtenidos en matlab, Figura 3.3, la diferencia es notable. Si bien en octave tarda casi 3 minutos en procesar 25 hojas lo que resulta en 6,4 segundos por hoja en matlab tan sólo tarda 15,38 segundos lo que se traduce en 0,4 segundos por hoja.

Code	Calls	Total Time (s)
`[im_pre,centros]=preprocesado(fichero_imagen,info,k,tipo);`	25	4.008
`revision=postprocesado(identity,m_res,m_mod,valor,im_pre,tipo,0);`	24	2.893
`imwrite(imresize(revision,escala),destino,'tif','Compression','jpeg','RowsP…`	24	2.514
`[ident,m_res,m_mod,erro_m]=procesado_test(im_pre,k);%cambiada`	25	2.434
`imwrite(imresize(imagenGrises,escala),destino,'tif','Compression','jpeg','R…`	24	1.989
		1.542
		15.380

Figura 3.3: Perfilado de funciones más utilizadas en el proceso de digitalización de exámenes de tipo test en matlab.

Por otro lado, en el proceso de corrección de exámenes de desarrollo se obtuvieron los siguientes resultados:

Top > **preprocesado**

Function	Total (s)	Self (s)	Calls
imread	340.749	0.008	116
bwmorph	101.028	0.700	348
encuadrar	34.534	0.322	23
busca_FPF	4.952	0.206	116
not	0.615	0.615	139
orientacion	0.091	0.004	23
binary >	0.000	0.000	116
islogical	0.000	0.000	116
binary <=	0.000	0.000	116
binary !=	0.000	0.000	232
binary ==	0.000	0.000	116

Figura 3.4: Funciones más utilizadas dentro de la función preprocesado en octave.

Function	Time (s)	Time (%)	Calls
__magick_read__	675.468	75.04	232
convn	69.145	7.68	782
__magick_write__	32.727	3.64	116
imresize>conv_interp_vec	17.483	1.94	232
binary .*	11.090	1.23	9396
conv2	10.359	1.15	116
lookup	10.263	1.14	1720
interp2	7.280	0.81	23
binary >=	6.656	0.74	8421
binary *	6.070	0.67	28139
imdilate	4.846	0.54	138
binary +	4.283	0.48	56654
binary ==	4.230	0.47	47478
applylut	4.191	0.47	116
binary -	3.464	0.38	29561
__magick_finfo__	2.761	0.31	1
uint8	1.847	0.21	348
zeros	1.784	0.20	2695

Figura 3.5: Perfilado de funciones más utilizadas en el proceso de digitalización de exámenes de tipo desarrollo en octave.

Code	Calls	Total Time (s)
`[im_pre,centros,Localizado]=preprocesado(fichero,info,k,tipo);`	116	12.679
`imwrite(imresize(a,escala),[dir_correcto,filesep,nombre_fich],'tif','Compre…`	116	3.606
`a=imread(fichero,'tif',m);`	116	2.539
`ident(:,:,n)=identificar_ex2(not(im_pre));`	25	1.008
`a= uint8(a) * 255;`	116	0.899
		2.119
		22.851

Figura 3.6: Perfilado del proceso de digitalización de exámenes de tipo desarrollo en matlab.

Vuelven a aparecer las funciones de transformación geométrica, redimensionado, escritura y operaciones morfológicas dentro de las primeras, siendo las funciones de escritura y lectura las que más tiempo consumen.

Al comparar los resultados de octave, Figura 3.5 con los de matlab, Figura 3.6 se puede observar como en ambas las funciones de lectura y escritura además de la contenida dentro de la función preprocesado son las que mayor tiempo consumen sin embargo, el tiempo en octave es 39 veces mayor al de matlab.

3.5. Conclusiones

Quedan de esta manera, comprobadas las limitaciones del lenguaje de software libre de octave. Que pasó de ser una prometedora alternativa a Matlab a quedarse relegado para casos de máxima emergencia donde no se disponga de Matlab.

La lentitud del proceso es muy probable que se deba a que las funciones de octave estén mucho menos optimizadas que las de Matlab, lo que genera una diferencia de velocidad abismal a la hora de comparar ambas alternativas.

Capítulo 4

Conclusiones y líneas futuras

4.1. Conclusiones

Durante esta redacción, se ha intentado introducir al lector al trabajo de fin de grado realizado por el alumno Tomás escudero que consistía en el diseño completo de un sistema de corrección de exámenes y explicar todas las mejoras, ampliaciones y modificaciones que se habían propuesto por parte del profesorado y que se han realizado en este trabajo de fin de grado.

A la hora de proponer el trabajo de fin de grado por parte del profesor a alumno, la tarea principal de este era encontrar el foco del error que provocaba que no funcionase de manera correcta en las nuevas digitalizadoras del departamento.

Una vez solucionado el problema anterior, el trabajo de alumno ha sido el de facilitar todo el proceso de corrección de los profesores ya sea, mediante la creación de nuevas herramientas que permiten agilizar la corrección de los archivos con errores, la optimización del proceso de subir los archivos al espacio compartido o la optimización del sistema previamente diseñado.

La creación de la aplicación de corrección de exámenes erróneos y la implementación de la lista de alumnos, permitirán al profesor ahorrar tiempo a la hora de tener corregir los exámenes, y además proporcionarán robustez a la hora de detectar exámenes incorrectos.

Un objetivo que no ha sido cumplido, ha sido el de crear una alternativa viable a Matlab, que permita hacer uso del sistema diseñado en software libre. Octave ha demostrado no serlo, y por lo tanto, a no ser que se realicen mejoras sustanciales en la optimizaciones de las funciones descritas en el capítulo 3, va a seguir sin ser una buena alternativa.

Por otro lado, debido a las limitaciones de las digitalizadoras del departamento, explicadas en el capítulo 2, y aunque se consiguió que funcionase para las digitalizadoras del departamento, se tuvo que optar por cambiar todo el proceso de preprocesado y procesado para que se hiciesen todas las operaciones morfológicas sin la resta.

Repasando todo lo que se ha realizado en este trabajo de fin de grado, se puede afirmar que se ha conseguido cumplir con la mayoría de objetivos impuestos al principio de la realización del proyecto, a excepción de los comentados en los párrafos anteriores, y que por lo tanto dejan la puerta abierta a posibles futuras mejoras.

4.2. Líneas futuras

La tarea principal de este trabajo era que funcionase para las digitalizadoras del departamento, cuyo problema está explicado, por este motivo, no se ha experimentado con otras digitalizadoras, y por lo tanto no se tiene información suficiente para saber si funcionará en otras digitalizadoras, de manera que si se quisiera ampliar en un futuro para ser utilizado en otros departamentos, se deberá hacer la debida diligencia en el preprocesado en general.

Esto se debe a que tanto las zonas establecidas como los puntos de control están acotados para la plantilla creada por el departamento de comunicaciones. Por lo tanto en caso de querer extender su uso a otros departamentos, será necesario crear diferentes versiones del programa o diferentes versiones de la función de crear zonas en función de las plantillas que se quieran procesar.

Por otro lado, debido a que octave ha resultado ser una opción no viable, otra posible tarea sería la de intentar migrar el código de lenguaje matlab a un lenguaje como python, java o C que cuentan con librerías como imageio u openCV, que son mejores y más optimizadas que las utilizadas por octave, y que además cuentan con una comunidad más amplia.

Otra posible actualización sería la de sustituir el procesado digital de la señal por un modelo de inteligencia artificial. Actualmente se desconoce el tiempo y la dificultad de dicha transformación pero debido al auge de la inteligencia artificial en el reconocimiento de imágenes es una alternativa que hay que considerar.

Existen otro tipo de implementaciones que ya fueron mencionadas por el creador del sistema de corrección anterior. Que no han sido realizadas en este trabajo de fin de grado, y que por lo tanto quedan abiertas al futuro para que en caso de que los profesores así lo decidan puedan ser implementadas.

Bibliografía

[1] Rafael C. González, Richard E. Woods y Steven L. Eddins. *Digital Image Processing using MATLAB*. Upper Saddle River: Pearson Prentice Hall, 2004.

[2] *Matlab App Designer*. URL: `https://www.mathworks.com/help/matlab/app-designer.html` (visitado 28-07-2023).

[3] *GNU Octave Documentation*. URL: `https://docs.octave.org/latest/` (visitado 20-07-2023).

[4] *GitLab - Laboratorio de Informática*. URL: `https://gitlab.com/labinformatica/guieditor` (visitado 16-06-2023).

Parte II

Anexos

Capítulo 1

Manuales Usuario

1.1. Manual Aplicación Procesado

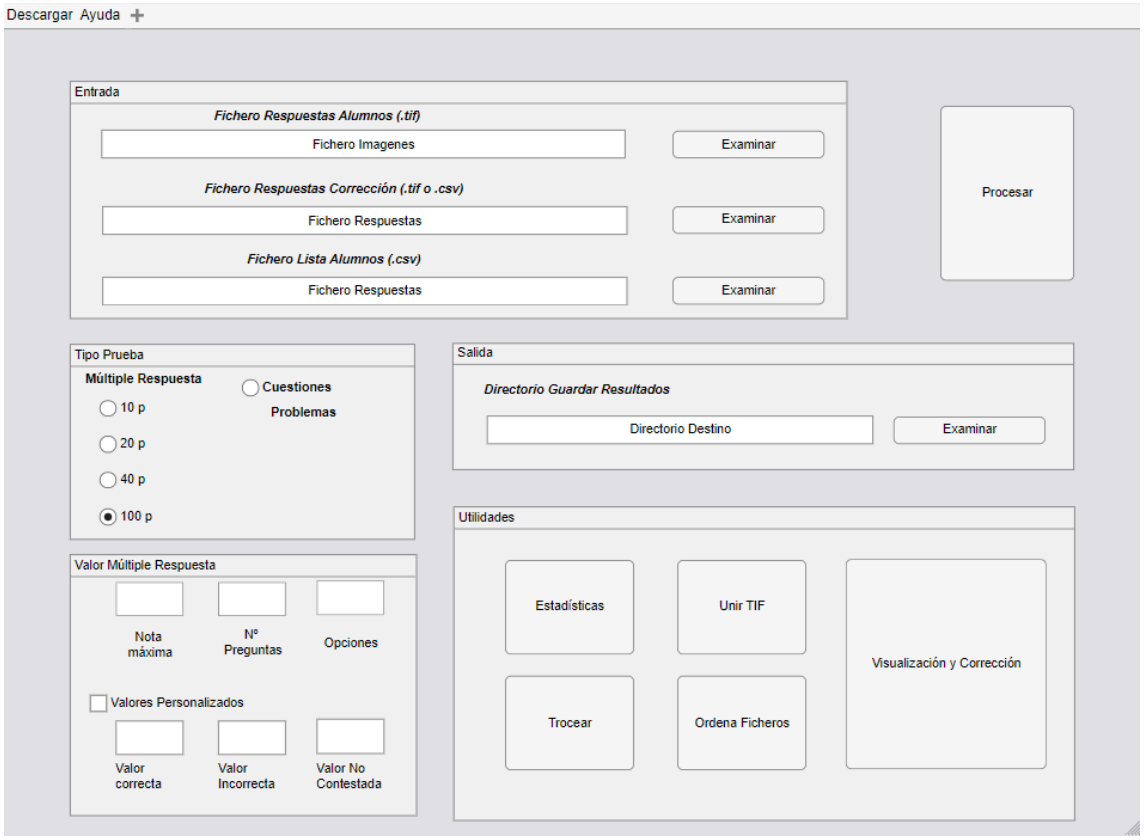

Figura 1.1: Ventana de la aplicación de Procesado.

La aplicación se compone de 5 secciones principales:

- Sección Entrada
- Sección Tipo Prueba
- Sección Valor Múltiple Respuesta
- Sección Salida
- Sección Utilidades

1.1.1. Sección Entrada

Figura 1.2: Captura de la sección de Entrada.

En esta sección se examinan los ficheros de entrada. Existen 3 tipos de ficheros de entrada:

- Fichero Respuestas alumnos: Fichero tipo tiff multipágina que contiene todas las plantillas de examen rellenadas por alumnos digitalizadas.
- Fichero Respuestas Corrección: Fichero de tipo csv o tiff que contiene la solución de cada modelo para las pruebas de respuesta múltiple.
- Fichero Lista Alumnos: Fichero de tipo csv que contiene una lista con la identificación y el nombre de todos los alumnos que cursan la asignatura.

Fichero Lista Alumnos permite ofrecer robustez durante la corrección comparando la identificación indicada por el alumno con la lista proporcionada para así descartar exámenes que están bien rellenados pero mal identificados.

1.1.2. Sección Tipo Prueba

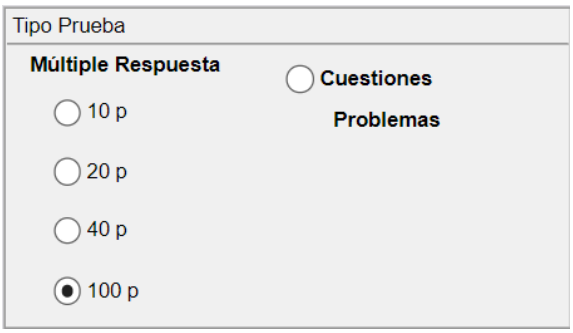

Figura 1.3: Captura de sección de Tipo Prueba.

En esta sección se indica el tipo de prueba que se va a corregir. Existen 5 tipos de pruebas:

- Prueba de respuesta múltiple o tipo test
    1. 10p: La plantilla es de 10 preguntas
    2. 20p: La plantilla es de 20 preguntas
    3. 40p: La plantilla es de 40 preguntas
    4. 100p: La plantilla es de 100 preguntas
- Prueba de desarrollo

1.1.3. Sección Valor Múltiple Respuesta

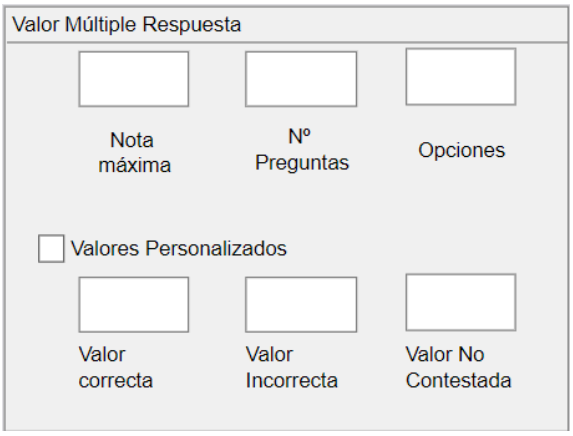

Figura 1.4: Captura de Sección Valor Múltiple Respuesta.

Esta sección es exclusiva para los exámenes de respuesta múltiple. En ella se indican:

- En caso de que se quieran valores por defecto.
    1. Nota máxima.
    2. Número de preguntas
    3. Opciones de cada pregunta
- En caso de que se quieran valores personalizados.
    1. Valor Correcta
    2. Valor Incorrecta
    3. Valor No Contestada

1.1.4. Sección Salida

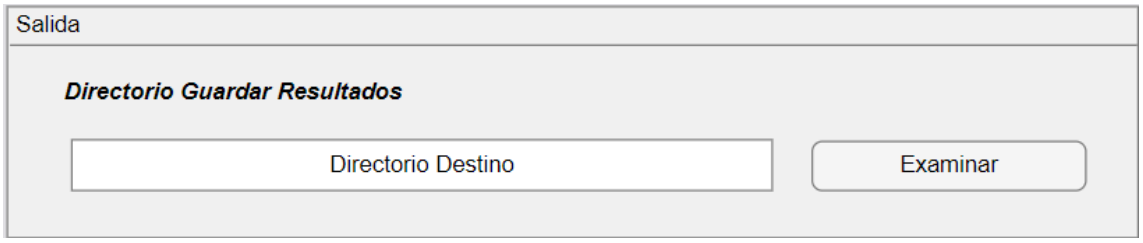

Figura 1.5: Captura de Sección Salida.

En esta sección se examina el directorio donde se cargarán todos los archivos creados.

Para las pruebas de respuesta múltiple, el directorio constará de las siguientes carpetas y archivos:

- Carpeta de Correctos
  1. Ficheros '@ID.tif'.
  2. Fichero csv con las notas e identificación de cada alumno.
- Carpeta de Errores
  1. Fichero tif recopilación de todos los errores de identificación
  2. Fichero tif recopilación de todos los errores de modelo
  3. Fichero csv con las notas e identificación de cada alumno mal identificado
  4. Ficheros tif con errores de identificación y de puntos de control
  5. Fichero log que contiene información de todo el proceso.
- Carpeta de Modelos
  1. Hay tantos ficheros csv como modelos. Cada fichero contiene las notas de su modelo correspondiente.

Para las pruebas de desarrollo:

- Carpeta de Correctos
  1. Ficheros '@ID.tif'.
  2. Ficheros csv con las notas e identificación de cada alumno para cada problema.
- Carpeta de Errores
  1. Ficheros csv con las notas e identificación de cada alumno mal identificado para cada problema
  2. Ficheros tif con errores de identificación, problema, nota o puntos de control.

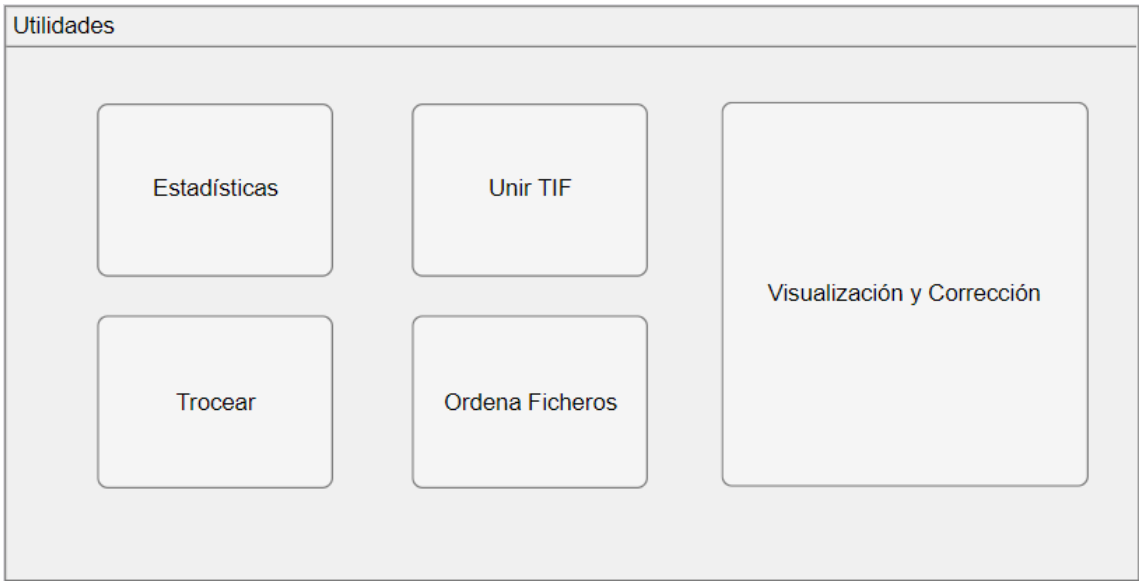

Figura 1.6: Captura de Sección Utilidades.

1.1.5. Sección Utilidades

Esta sección contiene las herramientas que no forman parte del procesado de exámenes pero que pueden ser útiles antes o después.

- Trocear
- Estadísticas
- Unir TIF
- Ordena Ficheros
- Visualización y Corrección

1.2. Manual Aplicación Visualización Estadísticas

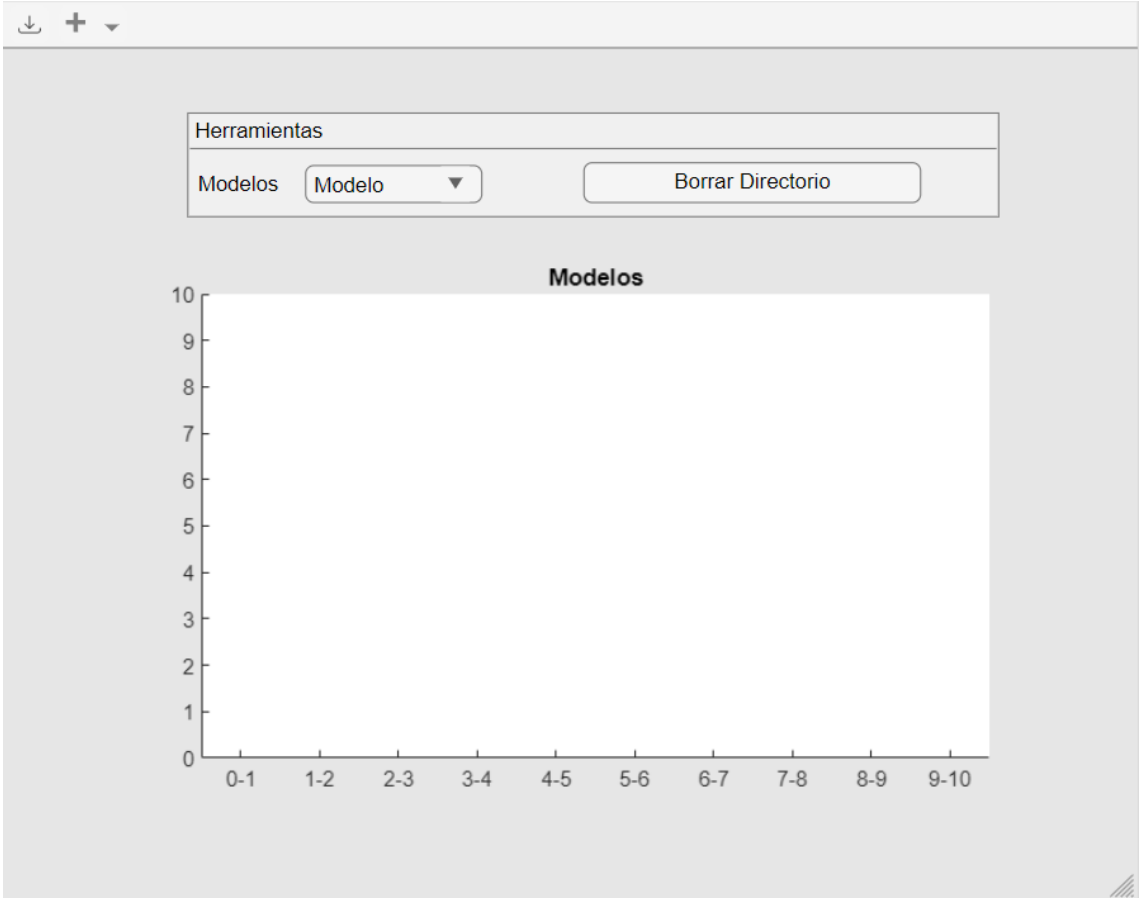

Figura 1.7: ventana aplicación Visualización Estadísticas.

La aplicación Visualización Estadísticas permite al usuario visualizar mediante un histograma, las notas por modelo o problema de una determinada prueba.

Se puede acceder directamente desde la aplicación de procesado indicando o sin indicar la carpeta de salida. En caso de no estar indicada, al clicar en el desplegable se solicitará al usuario que indique la carpeta.

Una vez se carguen correctamente los ficheros csv, aparecerán sus nombres en el desplegable y se podrá visualizar su contenido.

1.2.1. Borrar directorio

Para las pruebas de respuesta múltiple, se crea una carpeta específica llamada modelos. En caso de que el profesor quiera borrarla una vez ha obtenido la información necesaria, lo podrá hacer pulsando el botón. No usar en caso de que se estén visualizando notas de exámenes de desarrollo, ya que los ficheros csv de problemas se encuentran en la carpeta de Correctos, no en una específica.

1.2.2. Descargar

Opción de descargar los histogramas en un fichero TIF multipágina. Recomendable crear una carpeta específica para guardarlos.

1.3. Manual Aplicación Corrección Exámenes Erróneos

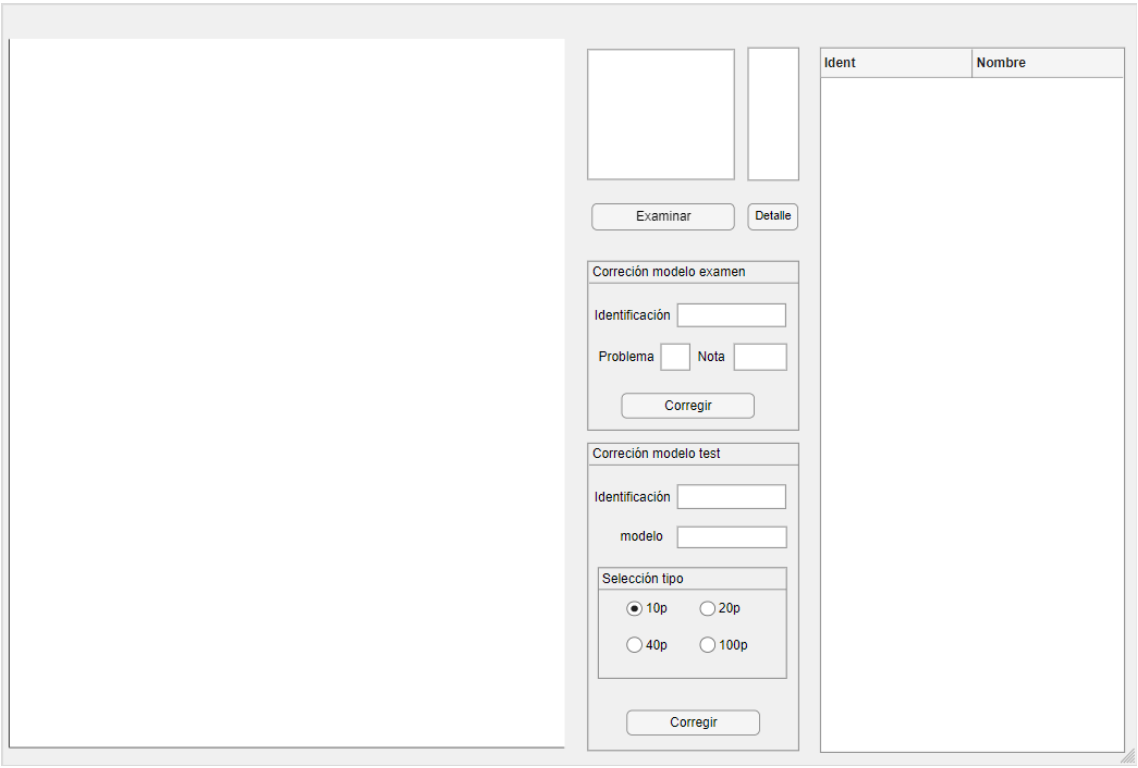

Figura 1.8: Ventana aplicación de Corrección de exámenes erróneos.

La aplicación contiene:

- Lista con las páginas enumeradas para el fichero seleccionado
- Lista con los ficheros tif de la carpeta seleccionada
- Tabla con los datos del fichero csv Lista Alumnos
- Sección Corrección modelo examen
- Sección Corrección modelo test

1.3.1. Formato del fichero CSV con lista de alumnos

Importante asegurarse de que la columna que contiene la identificación de la intranet sea de tipo Caracter, si no es así todos los números de identificación que empiecen por cero, el cero se omitirá al ser un cero a la izquierda, en caso de que se exporte como una columna numérica.

1.3.2. Primer paso: Seleccionar la carpeta de trabajo

Dentro de la aplicación, el primer paso a realizar es seleccionar la carpeta de Errores del examen que se va a corregir.Para ello, el usuario deberá pulsar el botón de examinar. Una vez esté seleccionada,

se cargará en la lista, los ficheros tif de esa carpeta. En caso de que el usuario quiera usar Lista de Alumnos, podrá cargarla de forma manual o en caso de que se encuentre dentro de la carpeta del examen, se cargará de forma automática.

Toda la aplicación está programada para poder encontrar todos los ficheros necesarios a partir de la ubicación de la carpeta y la ubicación del fichero seleccionado. En caso de que no pueda encontrar el fichero se solicitará al usuario seleccionarlo manualmente.

1.3.3. Segundo Paso: Visualizar los exámenes

Al hacer doble clic sobre cualquier elemento de la lista, por un lado se mostrará en los ejes de la izquierda la primera página del fichero tif seleccionado, y por otro lado se copiará el nombre del fichero seleccionado a las casillas de identificación de ambas secciones como se puede ver en la figura 1.9.

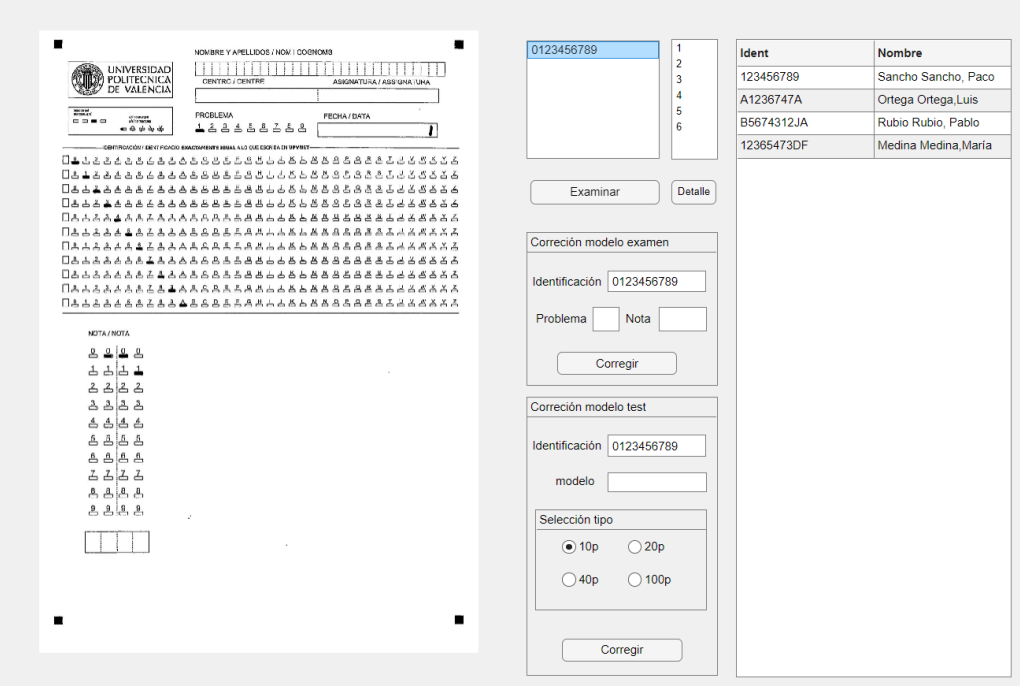

Figura 1.9: Ejemplo de uso de la Aplicación con lista de ficheros y de alumnos cargadas.

1.3.4. Tercer Paso: Corrección de exámenes

1.3.4.1. Sección Corrección modelo examen

La sección consta de 3 campos editables:

- Identificación del alumno
- Número del problema
- Nota

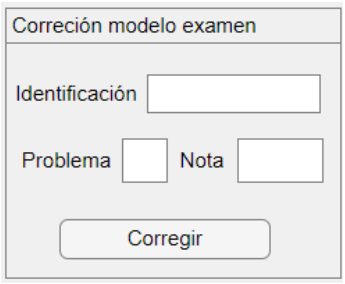

Figura 1.10: Captura de Sección Corrección modelo examen.

Según los parámetros de entrada introducidos el programa lleva un proceso u otro. Es recomendable para el caso de exámenes de desarrollo, que se rellenen las 3 casillas 1.10. Una vez se hayan introducido los datos necesarios, se pulsará el botón de corregir, y el proceso comenzará.

1.3.4.2. Sección Corrección modelo test

Figura 1.11: Captura de Sección Corrección modelo test.

- Identificación del alumno
- Modelo del examen
- Selección tipo de examen

En caso de que sea un error de identificación, solo hará falta rellenar la casilla de identificación ya que la nota estará guardada en el fichero csv de la carpeta de Errores. Una vez se hayan introducido los datos necesarios, se pulsará el botón de corregir, y el proceso comenzará.

Capítulo 2

Manual Programador

Fichero .m (función matlab)	Funciones usadas (Children)	Funciones que hacen uso (Parents)
busca **Descripción:** Calcula los centroides de las marcas de posición de la plantilla **Entrada:** Matriz binaria que representa la imagen preprocesada **Salida:** Matriz con coordenadas x e y de los centroides. Si la matriz 4x2 todos los centroides calculados. Si es 3x2, 2x2 o 1x2. Faltan puntos, no se podrá realizar el registro de la imagen	toolbox: \images\images\bwlabel.m toolbox: \images\images\regionprops.m current dir: rel2abs	preprocesado
calcular_nota **Descripción:** Extrae la información de la nota marcada por el profesor **Entrada:** Matriz que representa la imagen resta. **Salida:** error>Error en la introducción de la nota nota_n>valor de la nota decimal nota_s>valor nota string	toolbox: \images\images\bwlabel.m toolbox: \images\images\imerode.m toolbox: \images\images\regionprops.m toolbox: \images\images\strel.m current dir: crear_zonasHD current dir: rel2abs	tipo_examen
completo **Descripción:** Punto de enlace entre la GUI y los procesos de obtención de información. **Entrada:** fichero_imagen> nombre del fichero de imágenes de respuesta de los alumnos tipo>tipo de prueba seleccionada imagen_res>nombre del fichero de respuesta, tipo TIF o CSV. lista_alumnos>tabla con DNI y nombre alumnos **Salida:** NO	current dir: insert_respuestas current dir: tipo_examen current dir: tipo_test	Procesado_App
crear_zonasHD **Descripción:** Crea el conjunto de matrices de decisión. **Entrada:** NO **Salida:** x>matriz decisión identificación (eje x) y>matriz decisión identificación (eje y) xr>matriz decisión respuesta test (eje x) yr>matriz decisión respuesta test (eje y) xm> matriz decisión modelo (eje x) xp>matriz decisión nota profesor (eje x) yp>matriz decisión nota profesor (eje y)	current dir: valor.mat	calcular_nota getmodelo identificar_ex procesado_completo_zonas procesado_test procesado_zonas

encuadrar **Descripción:** Realiza el registro de la imagen de entrada con base la plantilla original. **Entrada:** im>Matriz de imagen preprocesada coor>Centroides calculados con *busca* **Puntos base:** *base=[59,63;2424,63;59,3452;2423,3451];* **Salida:** Imagen registrada		preprocesado
escribir_log **Descripción:** Escribe información en el fichero de log para el tipo de prueba de desarrollo **Entrada:** fid_log>descriptor del fichero errorn> variable error en nota errori>variable error identificación errorm>variable error problema nombrep> string con la identificación del alumno nota_n>string con la nota calculada k>número de página **Salida:** Los resultados se escriben en el fichero		tipo_examen
fichero_escribir **Descripción:** Inicialización del fichero CSV de notas **Entrada:** fichero>Nombre del fichero a inicializar **Salida:** Se inicializa el fichero con la información de la primera fila. DNI;NOTA		genera_csv genera_csv_mod
genera_csv **Descripción:** Introduce el valor de identificación y nota de un alumno, en el fichero correspondiente, en el tipo de prueba test. **Entrada:** nota> string con la nota del alumno ident> string con la identificación del alumno error> variable que indica el fichero donde escribir. *Si error = =1 > se escribe en fichero notas_error.csv* *Si error = = 0 > se escribe en fichero notas_ok.csv*	current dir: fichero_escribir	postprocesado

Salida: Fichero CSV con la información de identificación y nota añadida.		
genera_csv_mod **Descripción:** Introduce el valor de identificación y nota de un alumno, en el fichero correspondiente del problema pasado como entrada. **Entrada:** nota_n> nota del alumno tipo número ,nota_s>nota del alumno tipo cadena ident> string identificación alumno mod> número de problema errorn> error en la nota errori> error en la identidad errorm> error en el número de problema k> número de página **Salida:** Fichero CSV con la información de identificación y nota añadida.	current dir: fichero_escribir	tipo_examen
generar_marcas **Descripción:** Introduce las marcas de las respuestas correctas, erroneas y solución en la imagen del fichero @ID.tif de las pruebas tipo test. **Entrada:** solucion> matriz de solucion del profesor respuestas>matriz de respuestas del alumno imagen> Matriz de imagen preprocesada La forma de las marcas se cargan desde la variable marcas incluida en el programa **Salida:** revision>matriz que representa la imagen a color de la imagen preprocesada con las marcas.	toolbox: \images\images\imresize.m	Postprocesado postvisualizacion
genera_fich_modelos **Descripción:** Genera y escribe la nota en el fichero csv del modelo correspondiente **Entrada:** nota>string con la nota calculada m_mod>matriz de mapeo del modelo ruta>Ruta de la carpeta donde se guardarán los modelos **Salida:** No		postvisualizacion postprocesado

getmodelo **Descripción:** Extrae la información del número de problema marcado por el alumno, para las pruebas tipo desarrollo. **Entrada:** ud>matriz binaria imagen resta **Salida:** modelo> matriz de mapeo del problema marcado erro_m > indica si ha habido error en la introducción del número de problema	toolbox: \images\images\bwlabel.m toolbox: \images\images\imerode.m toolbox: \images\images\regionprops.m toolbox: \images\images\strel.m current dir: crear zonasHD current dir: rel2abs	tipo_examen
Histograma App **Descripción:** Aplicación donde se visualizan los histogramas de notas por modelo		Procesado_App

ident2string **Descripción:** Convierte la información de la matriz de mapeo de identificación en string **Entrada:** ident>matriz de mapeo de identificación lista_alumnos>tabla con DNI y nombre alumnos **Salida:** identidad> string con la identificación extraida de la matriz de mapeo erro > indica si ha habido error en la introducción de la identificación		tipo_examen tipo_test
identificar_ex **Descripción:** Extrae la información de identificación del alumno para el tipo de pruebas test. **Entrada:** Matriz binaria representa imagen resta **Salida:** identidad> string con la identidad extraida de la matriz de mapeo	toolbox: \images\images\bwlabel.m toolbox: \images\images\imerode.m toolbox: \images\images\regionprops.m toolbox: \images\images\strel.m current dir: crear_zonasHD current dir: rel2abs	tipo_examen
insert_respuestas **Descripción:** Extrae la información de la solución aportada por el profesor **Entrada:** imagen_res> Fichero con solución de los modelos. Tipo TIF o CSV tipo> entero que indica tipo de prueba, *1-10p* *2- 20p* *3- 40p* *4- 100p* **Salida:** m_mod_res> Estrucutura de matrices por modelo	subfunction: generar_csv_res subfunction: leer_csv current dir: preprocesado current dir: procesado_zonas	completo

localiza respuesta **Descripción:** Extrae la información de la solución aportada por el profesor para un modelo pasado como entrada **Entrada:** m_mod> Matriz de mapeo del modelo. **Salida:** respuestas> Matriz de respuestas del modelo indicado.		Postprocesado postvisualizacion
orientacion **Descripción:** Corrige el giro de 180 grados de la imagen preprocesada **Entrada:** im>imagen preprocesada **Salida:** imgirada> imagen con la corrección de giro realizado, si es necesario	toolbox: \images\images\bwlabel.m toolbox: \images\images\imrotate.m toolbox: \images\images\regionprops.m	preprocesado
postprocesado **Descripción:** Realiza las tareas de generación de ficheros CSV, introduce las marcas y localiza la respuesta del modelo marcado y calcula la nota **Entrada:** ident>matriz de mapeo de identificación m_res>matriz de mapeo de respuestas m_mod>matriz de mapeo del modelo valor>valores de los tipos de respuestas, correcta, incorrecta, en blanco im_pre>imagen preprocesada tipo>tipo de prueba tipo test error> indica si hay error en la extracción de información de identificación o modelo **Salida:** revision>imagen con marcas generadas	current dir: genera_csv current dir: generar_marcas current dir: localiza_respuesta current dir: genera_fich_modelos	tipo_test

postvisualizacion **Descripción:** Calcula la nota y genera tif corregido **Entrada:** tipo>tipo de prueba seleccionada m_mod>matriz de mapeo del modelo newdestino>nombre de fichero de imágenes digitalizadas **Salida:** nota>string con la nota calculada	current dir: preprocesado current dir: generar marcas current dir: localiza respuesta current dir:genera fich modelos current dir: procesado test	Visualizacion App
preprocesado **Descripción:** Realiza las tareas de preprocesado de la imagen de entrada **Entrada:** fichero>nombre de fichero de imágenes digitalizadas info>estructura de información del fichero deimagen k> número de página a preprocesar tipo> tipo de prueba tipo test. **Salida:** im_pre>imagen preprocesada centros> centroides marcas de posición localizado>dato booleano que indica si la página es una plantilla o no.	toolbox: \images\images\bwmorph.m toolbox: \images\images\im2bw.m toolbox: \images\images\imfilter.m toolbox: \images\images\imresize.m toolbox: \images\images\imrotate.m current dir: busca current dir: encuadrar current dir: orientacion	insert_respuestas tipo_test tipo_examen postvisualizacion
Procesado App **Descripción:** Fichero donde se encuentran las funciones relacionadas con la app.		
procesado test **Descripción:** Reune los procesos de extracción de información para las pruebas tipo test **Entrada:** im_pre> imagen preprocesada k> número de página a procesar **Salida:** identificacion> matriz de mapeo de identificación m_res>matriz de mapeo de respuestas alumno m_mod>matriz de mapeo modelo marcado erro_m>error en modelo	toolbox: \images\images\bwlabel.m toolbox: \images\images\graythresh.m toolbox: \images\images\im2bw.m toolbox: \images\images\imerode.m toolbox: \images\images\regionprops.m toolbox: \images\images\strel.m current dir: crear zonasHD current dir: identificar ex current dir: rel2abs	tipo_test postvisualizacion

procesado_zonas **Descripción:** Extrae la información del la solución del modelo marcada por el profesor **Entrada:** im_pre> imagen preprocesada de la solución del profesor tipo> tipo de prueba tipo test. **Salida:** m_res>matriz de mapeo de solución profesor m_mod>matriz de mapeo modelo marcado por el profesor	toolbox: \images\images\bwlabel.m toolbox: \images\images\graythresh.m toolbox: \images\images\im2bw.m toolbox: \images\images\imerode.m toolbox: \images\images\regionprops.m toolbox: \images\images\strel.m current dir: crear_zonasHD current dir: rel2abs	insert_respuestas
rel2abs **Descripción:** Convierte el valor relativo de las componentes x e y de los centroides calculados a valores aboslutos de la imagen. **Entrada:** centroides>matriz con los valores de las componentes x e y de los centroides xx1> valor de la componente x de la posición absoluta con respecto a la que se quiere convertir el valor yx1>valor de la componente y de la posición absoluta con respecto a la que se quiere convertir el valor **Salida:** centros> matriz con los valores de los centroides con valores absolutos de la imagen		busca calcular_nota getmodelo identificar_ex procesado_test procesado_zonas
seleccion_modelo **Descripción:** Genera y escribe la nota en el fichero csv del modelo correspondiente **Entrada:** letra> string con la letra del modelo **Salida:** m_mod>matriz de mapeo del modelo		Visualizacion_app
tipo_examen **Descripción:** Función principal que inicia los procesos de extracción de información para la prueba de desarrollo **Entrada:** fichero> nombre del fichero de imágenes a procesar. lista_alumnos>tabla con DNI y nombre alumnos tipo> tipo de prueba tipo test. **Salida:** NO. La información se genera en los ficheros de notas e imágenes correspondientes.	toolbox: \images\images\bwmorph.m toolbox: \images\images\graythresh.m toolbox: \images\images\im2bw.m toolbox: \images\images\imerode.m toolbox: \images\images\imfilter.m toolbox: \images\images\imresize.m toolbox: \images\images\imrotate.m toolbox: \images\images\strel.m current dir: calcular_nota current dir: preprocesado current dir: escribir_log current dir: genera_csv_mod current dir: getmodelo current dir: ident2string current dir: identificar_ex	completo

tipo_test **Descripción:** Función principal que inicia los procesos de extracción de información para la prueba de tipo test **Entrada:** fichero> nombre del fichero de imágenes a procesar. tipo> tipo de prueba tipo test seleccionada lista_alumnos>tabla con DNI y nombre alumnos **Salida:** NO. La información se genera en los ficheros de notas e imágenes correspondientes.	current dir: ident2string current dir: postprocesado current dir: preprocesado current dir: procesado_test	completo
visualizacion_app **Descripción:** Aplicación donde se realizan funciones de corrección de exámenes erróneos.		Procesado App