Corrector automático de plantillas test con teléfono móvil, tableta o cámara web.

Jordi Prados Camargo

Resumen— Actualmente es muy común el uso de plantillas tipo test para evaluar conocimientos. Estas plantillas suelen corregirse de forma manual o automática, pero en ambos casos el tiempo para notificar los resultados a los interesados no es inmediato. El objetivo principal de este proyecto es reducir el tiempo de notificación creando un sistema que permita corregir estas plantillas de forma inmediata. Para hacer este proceso más fácil, cada plantilla es identificada por un código QR único, el cual aporta información útil para la detección. Como resultado, se ha diseñado un sistema compuesto por una cadena de procesado de plantillas para obtener los resultados, un servidor que gestiona esta cadena y ofrece los mecanismos necesarios para administrar y recibir las imágenes enviadas, y una aplicación Android, la cual permite capturar imágenes de las plantillas y enviarlas al servidor para obtener los resultados.

Palabras clave— Android, binarización de imágenes, Código QR, detección de esquinas, Django, histogramas, iluminación, OpenCV, Python, REST, transformación geométrica.

Abstract— Today it is very common to use templates to assess knowledge. These templates are usually corrected manually or automatically, but in both instances the timing for reporting results to stakeholders is not immediate. The main objective of this project is to reduce this notification time creating a system that corrects these templates immediately. To make this process easier each template is identified by a unique QR code which provides useful information for detection. As a result, a system capable of meeting the needs and objectives has been created. This system is composed of a processing chain to get the results of the templates, a server that manages the processing chain and provides the tools for receiving the images, and an Android application to capture images and display the results.

Index Terms— Android, image binarization, QR code, corner detection, Django, histogram, illumination, OpenCV, Python, REST, geometric transform.

1 Introducción

ESTE proyecto nace de la necesidad de crear un sistema que sea capaz de detectar la información de un examen tipo test y devolver un resultado de la manera más rápida posible. Ya existen algunas alternativas que realizan este proceso, pero con este proyecto se intenta dar una respuesta inmediata y así reducir y optimizar el tiempo necesario para dar los resultados de las pruebas.

La propuesta de este proyecto es diseñar una plantilla modelo para los exámenes tipo test y un sistema compuesto por tres elementos principales. El primer elemento de este sistema, y el más importante, será el encargado de la de detección de los resultados de una plantilla. El segundo será un servidor que se encargará de la gestión del sistema de detección y de ofrecer los mecanismos necesarios para crear nuevos test y recibir las imágenes tomadas de las plantillas. Por ultimo, el tercer elemento será la aplicación Android, que tendrá como principal objetivo tomar las fotografías, enviarlas al servidor y mostrar los resultados. Estos tres elementos que forman el sistema deben trabajar conjuntamente para conseguir que el sistema sea lo más robusto y preciso posible.

Cada componente de este sistema será detallado en los siguientes puntos así como los resultados y conclusiones obtenidas.

- *E-mail de contacte: jordiprados90@gmail.com*
- *Mención realizada: Computación*
- *Trabajo tutoritzado por: Javier Sánchez (departamento de computación)*
- *Curso 2014/15*

2 Estado del arte

Actualmente existen aplicaciones que permiten la corrección de exámenes tipo test a partir de imágenes. En la gran mayoría de casos estas aplicaciones están adaptadas a las necesidades de cada organización, por lo que las características de cada plantilla siguen un patrón fijo. Un ejemplo es el software utilizado en la Universidad Autónoma de Barcelona [1], donde existe un servicio para el profesorado que ofrece la posibilidad de corregir exámenes test o encuestas con unas determinadas características. Para hacer uso de este servicio es necesario reservar una hora y día, por lo que hace de este servicio poco flexible. También existen diferentes proyectos que ofrecen

herramientas para este fin [2], [3], las cuales una plantilla limitada y con unas características muy especificas para hacer uso del detector.

Todas estas herramientas y servicios encontrados son procesos *offline*, es decir, la obtención de resultados no se realiza al momento de entrega, sino, que se obtienen cuando el examen ha acabado y todas las plantillas han sido procesadas por una herramienta determinada. Esto hace que el tiempo de respuesta de los resultados no sea inmediato y se necesiten varios días.

3 OBJETIVOS

El objetivo principal esta compuesto por tres partes: cadena de procesado y detección, servidor y aplicación Android.

3.1 Cadena de procesado y detección

La cadena de procesado es la parte más importante del sistema. Su correcto funcionamiento es crítico para tener un sistema fiable. Sus objetivos principales son los siguientes:

- Tratamiento inicial de la imagen para eliminar posible ruido y homogeneizar la iluminación.
- Identificación de la plantilla detectando los cuadrados que la forman.
- Lectura del código QR de la plantilla para obtener información útil para la detección.
- Detectar los contornos de las tablas que forman la plantilla.
- Obtener los resultados analizando si las celdas de cada tabla están marcadas o no.

3.2 Servidor

El servidor es la parte central del sistema. Es el encargado de la interacción entre la aplicación Android y la cadena de procesado y detección. Este sistema tiene que cumplir los siguientes objetivos para poder ofrecer un buen servicio:

- Mecanismos para recibir imágenes.
- Utilizar la cadena de procesado para obtener los resultados de las imágenes.
- Devolver al usuario la respuesta de la detección.
- Interfaz web para crear nuevos exámenes con sus códigos QR correspondientes.
- Posibilidad de visualizar los resultados de las imágenes enviadas.

3.3 Aplicación Android

La aplicación Android es la parte que se encarga de visualizar los resultados obtenidos por la detección. Este sistema tiene que cumplir los siguientes objetivos para ofrecer un buen servicio:

- Herramienta para capturar fotografías de las plantillas.
- Enviar la fotografía tomada al servidor.
- Mostrar los resultados de la detección.
- Posibilidad de cambiar alguna respuesta mal detectada antes de enviar la confirmación final al servidor.

4 TRABAJO REALIZADO

En este apartado se explican todas las tareas realizadas durante el desarrollo del sistema de detección, servidor y la aplicación Android. Previamente a esto, se detalla cómo es la plantilla utilizada para los test.

4.1 Plantilla utilizada

La definición de un modelo de plantilla permite al sistema detectar la información necesaria lo más rápido posible. En los inicios del proyecto se creó una plantilla para empezar a trabajar, pero ésta ha ido evolucionado según las necesidades de los diferentes módulos de la detección. Por ejemplo un cambio significativo fue la introducción de códigos QR para identificar las plantillas, el cual supuso un reajuste en el diseño.

Las plantillas están compuestas por tres elementos principales: la tabla de la identificación del alumno, la zona donde esta el código QR y la zona de las tablas de las preguntas. Un ejemplo es el que se puede ver en la imagen del apéndice A1.

Todas las posibles combinaciones de plantillas tienen que estar basadas en la misma estructura. Lo único que esta permitido cambiar es incrementar o decrementar el número de preguntas y de respuestas. También, la distribución de las preguntas se puede hacer en una o dos tablas, lo que hace que el sistema permita detectar plantillas con una gran variedad de combinaciones.

4.2 Cadena de procesado.

La cadena de procesado es la parte más importante. Está compuesta por un conjunto de módulos donde cada uno tiene una función específica dentro de la detección. En concreto hay cinco módulos: preprocesamiento de la imagen, detección de cuadrados, detección del código QR e identificación de la plantilla, detección de los contornos de las tablas y detección de respuestas. Todos ellos trabajan como si fueran una caja negra, es decir, esperan unos parámetros y devuelven unos resultados. Estos módulos son secuenciales, esto quiere decir que los parámetros que pueden esperar son los valores de salida de otro módulo de la cadena.

Se ha utilizado Python para el desarrollo con librerías como NumPy [4] y OpenCV [5] entre otras. Estas librerías aportan funciones útiles para la detección, en especial la librería OpenCV tiene funciones de tratamiento de imagen que son esenciales para el sistema.

Todos los detalles de implementación de estos módulos son detallados en los siguientes puntos.

4.2.1 Preprocesamiento de la imagen.

Este módulo se encarga de filtrar la imagen para prepararla para el resto de módulos. Principalmente elimina el ruido, homogeneiza la iluminación y pasa la imagen a una escala de grises con la que será más fácil trabajar.

Primero de todo se utiliza una técnica de *smothing*, concretamente se utiliza *Gaussian smothing* [6] para reducir el ruido que puede haber en la imagen. Para este propósito se ha utilizado la función *GaussianBlur* de OpenCV.

Después de aplicarle este filtro a la imagen para eliminar el ruido, se realiza una normalización con el objetivo de homogeneizar la iluminación. El objetivo es resaltar los colores negros de las tablas que forman la plantilla, y para conseguirlo, primero se utiliza la operación *closing* de morfología matemática [7] con un kernel circular. El resultado de esta operación se puede ver en la imagen de la Figura 1.

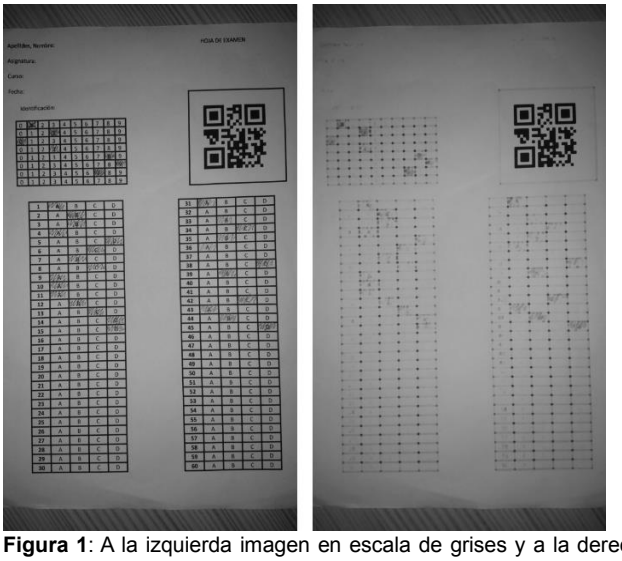


Figura 1: A la izquierda imagen en escala de grises y a la derecha imagen con la operación closing.

Como se puede observar en la Figura 1, la operación de closing reduce el color negro de las tablas. La parte del código QR no se ve reducida ya que el kernel utilizado no es lo suficientemente grande.

Una vez realizada esta operación de morfología matemática, se divide la imagen en escala de grises con la imagen resultante de la operación del *closing*. El resultado es el mostrado en la Figura 2, en la cual se puede observar como ahora las tablas de la plantilla quedan remarcadas por encima del resto y la iluminación ha sido homogeneizada. A esta operación se la conoce como *Bottom Hat* [7]. Esto será útil para el siguiente módulo, el cual intenta encontrar los cuadrados que forman la plantilla.

4.2.2 Detección de cuadrados.

En este módulo se realiza la tarea de identificar los cuadrados que forman la plantilla. Por cuadrados se entiende las zonas de mayor área que contienen otros cuadrados más pequeños.

Primero de todo se intentan resaltar las tablas de la

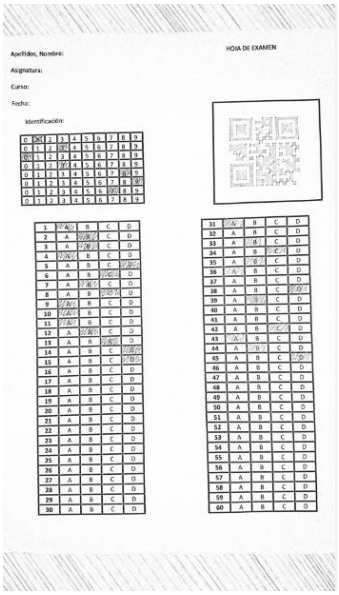


Figura 2: Imagen resultado de la operación Bottom hat.

plantilla por encima de cualquier elemento de la imagen. Se utiliza la imagen procesada y se binariza utilizando el mecanismo de binarización Otsu [8]. Esta técnica es especialmente útil cuando hay una clara diferencia entre los objetos a extraer y el fondo. El principio de binarización de una imagen se basa en las siguientes funciones:

- En el caso de que los objetos a extraer sean claros respecto al fondo:

$$g(x,y) = \begin{cases} 1, & f(x,y) > T \\ 0, & f(x,y) \leq T \end{cases} \quad (1)$$

- En el caso de que los objetos a extraer sean más oscuros que el fondo:

$$g(x,y) = \begin{cases} 1, & f(x,y) < \mathrm{T} \\ 0, & f(x,y) \geq \mathrm{T} \end{cases} \quad (2)$$

Donde f(x, y) es el valor de un punto de la imagen en escala de grises y T es el umbral. Para obtener este umbral, Otsu maximiza la intervarianza entre clases utilizando los datos de la imagen. Un ejemplo es el que se puede observar en la Figura 3. Esto tiene sus ventajas y sus desventajas, ya que por un lado, las zonas significativas quedan resaltadas, pero por el otro, existirán zonas importantes de la plantilla que no superen el umbral y no serán binarizadas correctamente.

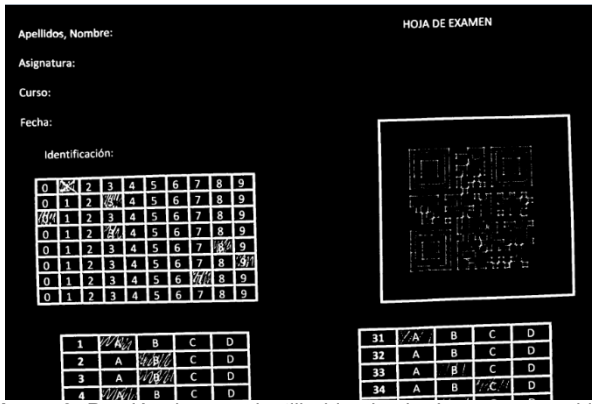


Figura 3: Porción de una plantilla binarizada. Las zonas en blanco son las tablas e información de la plantilla.

Esta binarización se realiza para detectar los contornos de las tablas y así poder utilizar la función *findContours* de OpenCV. Despues de estos pasos, se procede a utilizar las funciones *findContours, contourArea* y *approxPolyDP* de OpenCV utilizando la imagen binarizada previamente. La primera función nos devuelve las posiciones (x, y) de los contornos encontrados y la segunda estima el área que ocupa un contorno. La ultima función busca una forma aproximada de una figura geométrica, intentando reducir los puntos que representan el contorno. Ésta ultima, se basa en el algoritmo de "Douglas-Peucker Line-Simplification Algorithm" [9], el cual busca reducir el número de puntos utilizado para representar una curva. Esta función utiliza un parámetro llamado *épsilon* el cual es la distancia máxima entre el valor aproximado y el valor del contorno real.

Con estas funciones se intenta conseguir representar los contornos con 4 puntos, los cuales delimitaran las esquinas de los cuadrados de la plantilla.

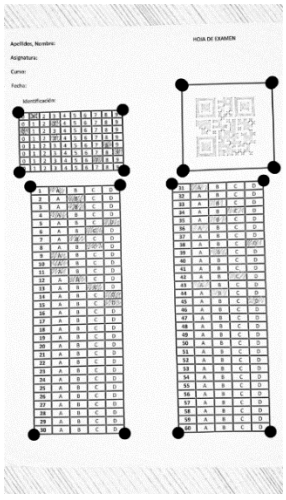


Figura 4: Esquinas que forman cada tabla.

Básicamente, el algoritmo para encontrar los cuadrados que forman la plantilla sigue los siguientes pasos:

- Encontrar los contornos de la imagen.
- Calcular el área de cada contorno con la función *contourArea*.
- Los contornos con mayor área y que no estén dentro de otros contornos serán los candidatos elegidos.
- Para los contornos elegidos, utilizar la función *approxPolyDP* para obtener la figura geométrica que los representa. En el caso de tener cuatro puntos, serán considerados parte de la plantilla.

En la Figura 4 se puede ver los puntos que forman cada cuadrado. Un problema de este método es que las esquinas que forman cada tabla no están ordenadas de la misma forma. Este aspecto tiene que ser corregido para poder detectar las celdas de cada tabla.

4.2.3 Detección del código QR e identificación de la plantilla.

Uno de los pasos más importantes es la detección del código QR de la plantilla. Como se ha podido ver en los puntos anteriores, el código QR se ve afectado por el preprocesado de la imagen. Esto no supone un problema ya que se dispone de los puntos que forman los cuadrados de la plantilla, por lo que solo se tiene que analizar el contenido de cada cuadrado de la imagen original en busca de QR.

Para la detección del código, se utiliza la librería en C llamada *zlib* [10] y la librería QRTools [11] de Python. Esta última utiliza la librería *zlib* para detectar un código QR en una imagen. Cuando un código QR es detectado se puede extraer su información, como por ejemplo, el valor del código. Tambíen, con una pequeña modificación en el código de la librería es posible extraer la localización de los puntos que forman el código QR. Estos puntos siempre están en el mismo orden, lo que nos permite saber la orientación global de la plantilla dentro de la imagen, tal y como se muestra en la Figura 5.

Para reordenar los puntos de cada cuadrado, se utilizan los cuatro puntos del código QR para aprovechar que estos están ordenados como se ha mostrado en la Figura 5.

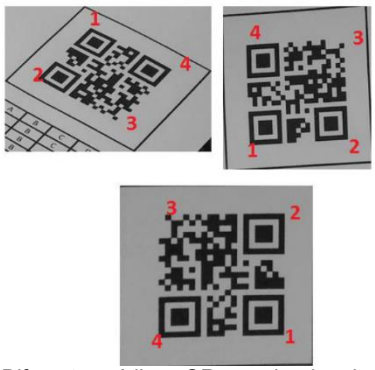


Figura 5: Diferentes códigos QR con el orden de cada esquina.

El objetivo ahora es encontrar una similitud entre la posición de estos puntos con la posición de los cuatro puntos de cada tabla detectada. El problema es que cada tabla trabaja con un rango de coordenadas (x, y) diferente y esto dificulta la comparación con los puntos del código QR. Para resolver este problema, las esquinas de cada tabla son normalizadas con valores entre el 0 y 1 utilizando la siguiente formula:

$$x_i' = \frac{x_i - \min x}{\max x - \min x} \qquad (3)$$

La formula anterior sirve también para normalizar valores de y. Este proceso se realiza también para los puntos que delimitan el código QR. De esta manera se consigue tener los puntos de cada tabla en el mismo rango de valores que los puntos del código QR. El siguiente paso es crear una métrica para determinar qué punto del código QR puede ser emparejado con otro punto de una tabla. La métrica utilizada es la distancia euclidiana entre dos puntos. De esta manera, dada una tabla detectada en la ima-

gen, el procedimiento para realizar la ordenación sigue los siguientes pasos:

- Identificar cada punto de la tabla con un número. Estos identificadores sirven para obtener el valor real de un punto antes de ser normalizado.
- Normalizar cada punto de la tabla y los del código QR.
- Por cada punto normalizado del código QR, emparejarlo con el punto de la tabla más cercano.
- Recorrer los emparejamientos en orden para obtener los números de identificación de cada punto de la tabla y así poder obtener el valor del punto real de la imagen.

Un ejemplo de esta ordenación es el mostrado en la Figura 6.

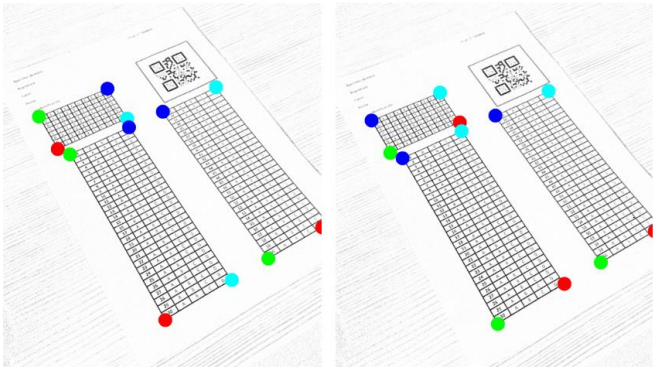


Figura 6: A la izquierda las esquinas de las tablas estan desordenadas, a la derecha estan ordenadas utilizando el código QR.

Una vez ordenados los puntos de cada tabla detectada se procede a identificar cual es cada tabla. Primero se intenta buscar la tabla de la identificación. Esta tabla es la más cercana al código QR. Una vez detectada esta tabla, el resto de tablas detectadas se cuentan como tablas de test.

El servidor, utilizando la información del código QR detectado, proporciona detalles de la plantilla tales como el número de tablas o de preguntas. Esta información es útil para poder descartar las plantillas en el caso de no poder detectar el número de tablas que estipula este test introducido en la base de datos del servidor.

4.2.4 Detección de los contornos de las tablas.

Utilizando la información obtenida en los módulos anteriores, este módulo centra su esfuerzo en encontrar las esquinas que delimitan cada celda de cada tabla.

Para detectar las celdas de una tabla, se utiliza la imagen procesada, como la de la Figura 2, y se utilizan las esquinas de cada tabla, ordenadas en el módulo anterior. Se aplica una transformación geométrica con el objetivo de facilitar la detección de las celdas. Para este caso se intenta calcular la transformación de perspectiva de una imagen 2D [12]. Este proceso es necesario ya que el sistema es sensible a la posición de la tabla, es decir, es necesario tener las tablas lo mas rectas posible. Se ha utilizado la función *getPerspectiveTransform* de OpenCV para realizar esta tarea. Un ejemplo de este proceso es el que se puede observar en la Figura 7.

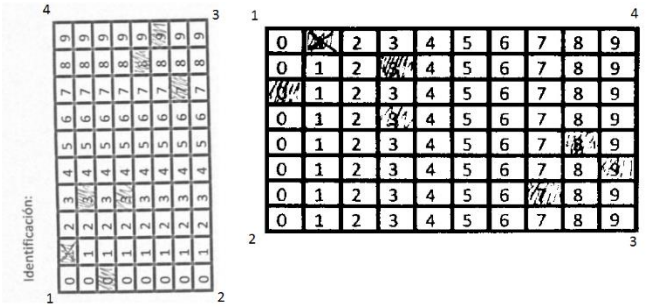


Figura 7: La tabla de la izquierda es la imagen de la plantilla, mientras que la tabla de la derecha es la misma tabla binarizada y aplicandole la transformación de perspectiva.

Se distinguen dos tipos de líneas dentro de la tabla, las líneas horizontales y las líneas verticales. El proceso trata de remarcar las líneas horizontales y las líneas verticales para después obtener el punto en el que convergen. Para este propósito se utiliza morfología matemática, en partícular la operación *closing*¸ con dos kernels diferentes. Un kernel con forma rectangular horizontal para remarcar las líneas horizontales, y un kernel rectangular vertical para remarcar las verticales. La medida del kernel es un aspecto crítico porque dependiendo de su tamaño puede remarcar más o menos zonas. Para establecer una medida óptima se utiliza una de las esquinas de la tabla a detectar y se va desplazando el punto en diagonal mientras el valor del punto sea de color negro. Cuando detecta que el valor del punto es de color blanco, se cuentan los puntos negros y ese valor se utiliza como número para el grosor de la línea del kernel. Un ejemplo de la detección de las líneas horizontales y verticales es el que se puede ver en la imagen de la Figura 8.

Una vez se han resaltado las líneas horizontales y verticales, se buscan los puntos de convergencia entre las dos. Para realizar este paso se utiliza la función *bitwise_and* [13] de OpenCV, la cual realiza la operación lógica AND entre los valores de las dos imágenes. Seguidamente, se procede a detectar el centro de masas de cada zona de convergencia calculada previamente. Para esto, primero se utiliza la función *findContours*, la cual nos devuelve los puntos de los diferentes contornos, y después se utiliza la función *moments* [14], [15] de OpenCV.

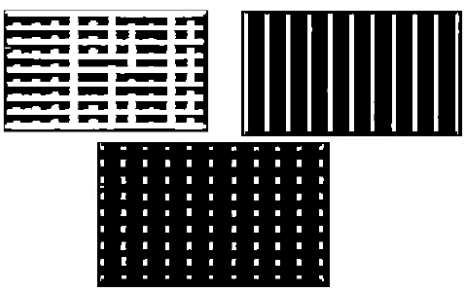


Figura 8: La primera imagen muestra las lineas horizontales de la tabla, la segunda las lineas verticales y la última muestra el punto de convergencia entre las dos primeras.

En el ámbito de visión por computador, los momentos de una imagen sirven para describir objetos después de la segmentación. También, usándolos se pueden extraer propiedades de la imagen tales como intensidad total, centro de masas e información sobre su orientación.

Como resultado, obtenemos los puntos centrales de cada esquina que definen las celdas. Un ejemplo es el mostrado en la Figura 9.

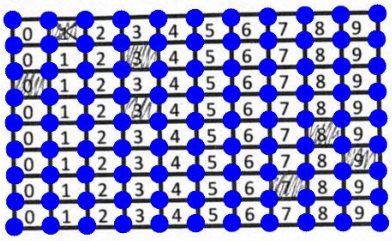


Figura 9: Esquinas de cada celda detectadas.

Una vez detectados las esquinas de cada celda, se reordenan por filas de izquierda a derecha y de arriba abajo formando una matriz de puntos. Las dimensiones de esta matriz dependen del tipo de tabla. En el caso de la tabla de identificación, la matriz será de 11 columnas por 9 filas. De esta manera, los puntos intermedios que no estén en una esquina pueden ser eliminados fácilmente determinando la distancia entre cada punto. También, dado que se conoce el número de preguntas y respuestas de la plantilla, los contornos de las tablas de test forman matrices dependiendo de las características de cada tabla. Esto facilita el trabajo para el módulo de detección de respuestas.

Llegados a este punto, se ha realizado una comparación entre 7 imágenes tomadas por diferentes dispositivos para ver como se comporta el detector de contornos de las tablas. Para esta prueba se han utilizado los siguientes dispositivos móviles: LG G3, Samsung Galaxy Young y Sony Xperia S. Esta prueba se ha realizado sobre una plantilla que tiene 471 esquinas, por lo que se intenta ver cuál de estos dispositivos alcanza este número. En el gráfico comparativo de la Figura 10 se muestra el número de esquinas detectadas correctamente por el sistema.
Cada teléfono móvil ha tomado siete imágenes, donde cada imagen es similar en ángulo e iluminación a otra imagen tomada por los otros dispositivos móviles. Se han intentado comparar estas imágenes similares para determinar el funcionamiento del modelo según la imagen de cada dispositivo.

Como se puede observar claramente en el gráfico de la Figura 10, las imágenes tomadas por el teléfono móvil LG G3 ofrecen los mejores resultados. En muchos casos consigue detectar el número máximo de esquinas de la plantilla. Por el contrario, al tener más resolución, el proceso necesita más tiempo para obtener resultados. En cambio, el teléfono móvil Sony Xperia S obtiene resultados algo mejores que los obtenidos por el Samsung Galaxy Young. Por último, los resultados obtenidos por el Samsung Galaxy Young no llegan a ser del todo satisfactorios ya que la lente de la cámara es de menor calidad. Las imágenes no son del todo nítidas y hay que enfocar muy bien para obtener una imagen clara.

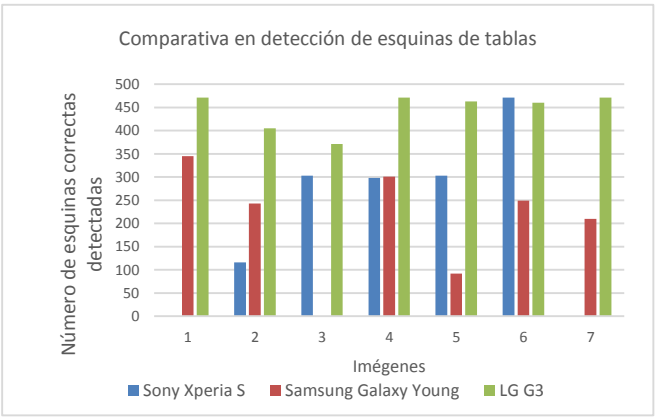


Figura 10: Gráfico comparativo para la detección de esquinas.

En este módulo existe el problema de que no se consiga detectar el número correcto de contornos. De hecho en el gráfico de la Figura 10 se muestra este problema en algunos dispositivos. Si no se detectan los contornos exactos, la plantilla no puede ser evaluada correctamente. Para solucionarlo se se ha creado una plantilla con el máximo de preguntas permitido y el máximo número de respuestas permitido por hoja, un ejemplo lo podemos ver en el apéndice A2. Esta plantilla se utilizará como patrón para el resto de plantillas entrantes, lo que quiere decir que la única plantilla a la que se le detectarán los contornos será a ésta. La información de los contornos de esta plantilla es guardada en un fichero *csv*, así como la disposición de los diferentes cuadrados que la forman. De esta manera, cuando una imagen de una plantilla entra en el sistema, se realizan los siguientes pasos para poner en correspondencia los puntos:

Para el cuadrado de identificación:

- Obtener la matriz de la transformación geométrica entre los cuatro puntos que forman el cuadrado de identificación y los cuatro puntos del cuadrado de identifiacion de la plantilla guardada en el fichero *csv*. Utilizar esta matriz para pasar los puntos del fichero *csv* a la imagen de la plantilla entrante.

Para los cuadrados de las tablas de las preguntas:

- Dado que pueden existir plantillas de exámenes con diferentes combinaciones de preguntas y respuestas, es necesario conocer estos datos de la plantilla entrante antes de continuar con la detección. Esta información se obtiene al enviar el código QR al servidor, el cual nos identifica la plantilla. Aprovechando que los datos del fichero del *csv* están guardados por filas y columnas formando una matriz, dependiendo del número de preguntas y respuestas de la plantilla se puede deliminar la zona de contornos a escoger dentro de esta matriz. Seguidamente, se calculan las cuatro esquinas que delimitan esta zona y se obtiene

la matriz de transformación geométrica entre estos cuatro puntos y los cuatro puntos de la tabla a detectar de la imagen entrante. Una vez obtenida la matriz se puede usar para pasar los puntos de la zona del *csv* seleccionada a la tabla de la imagen entrante.

Con este proceso se asegura que siempre se detectarán el número correcto de contornos, independientemente de la calidad de la imagen. Por el contrario, los puntos no siempre son posicionados exactamente pero las respuestas sí que podrán ser evaluadas para la detección. Un ejemplo de este desplazamiento es el mostrado en la Figura 11.

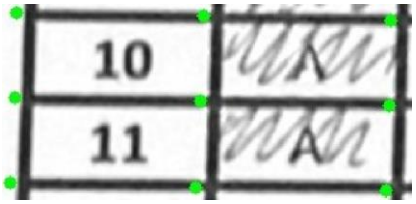


Figura 11: Ejemplo de desplazamiento en los puntos.

Este desplazamiento se debe a que las tablas detectadas de las diferentes imágenes no son del todo rectas. En muchos casos tienen una pequeña curva debido a un aspecto físico de la cámara que toma la fotografía. Esta curva hace que todos los puntos sean desplazados respecto al centro de masas de cada celda de la tabla.

4.2.5 Detección de respuestas.

Este módulo es la fase final de la cadena de procesado. Utiliza los contornos de cada tabla detectados para poder delimitar cada celda por 4 puntos. Con estos 4 puntos se obtiene la zona de cada casilla para poder procesarla por separado al resto de la imagen. Cada fila de cada tabla corresponde a una pregunta, por lo tanto, el objetivo es buscar la celda marcada dentro de una fila. Para esto se han creado dos detectores básicos: uno utiliza la comparación de histogramas, para determinar si una celda esta marcada o no, y el otro hace un recuento de las zonas marcadas de la celda.

El detector basado en la comparación de histogramas compara cada celda con una celda vacía. Primero se calculan los histogramas de cada celda y luego se comparan. Se utiliza la función *calcHist* de OpenCV y la función *compareHist* para esta tarea. Esta última función tiene la opción de utilizar 4 tipos de métricas para la comparación: Correlation, Chi-Square, Intersection y Bhattacharyya distance. Cada métrica ofrece un comportamiento distinto, por lo que se ha realizado un estudio para determinar cuál de ellas ofrece mejores detecciones de respuestas. Los resultados de este estudio se pueden ver en la tabla y la gráfica del apéndice A3.
Para determinar si una celda esta marcada o no, el valor obtenido de la comparación de histogramas tiene que ser superior al valor medio obtenido al comparar histogramas por cada celda de la fila.

Figura 12: Diferentes celdas binarizadas.

El segundo detector cuenta las zonas marcadas dentro de una celda. En este caso cada celda es binarizada tal y como aparece en los ejemplos de la Figura 12. De esta manera, por cada celda se hace un recuento del número de zonas en con color blanco marcadas utilizando la librería Numpy para trabajar con *arrays*. Este número debe de ser superior a un umbral para contarlo como respuesta marcada o no marcada. El umbral fijado depende del valor medio de cada fila.

Estos dos detectores son básicos pero tienen un buen comportamiento en general. No distinguen entre símbolos, por lo que hay que marcar bien la zona para obtener buenos resultados. Estos resultados son evaluados en el apartado 5 (Resultados) de este documento.

4.3 Servidor.

El servidor hace de intermediario entre la cadena de procesado y la aplicación Android. En un principio no existía interfaz web y solo se disponía de una API REST para atender a las peticiones que llegaban. Debido a la introducción del código QR se creó una interfaz para administrar los exámenes. Por lo tanto tenemos dos partes: la interfaz web y la parte de la API REST, que controla la base de datos, y la cadena de procesado. Para la creación de la API REST se ha utilizado Django REST framework [16] basado en Python. Éste ofrece muchas facilidades para crear este tipo de servicios. La estructura de la base de datos viene definida por el esquema Entidad - Relación de la Figura 13.

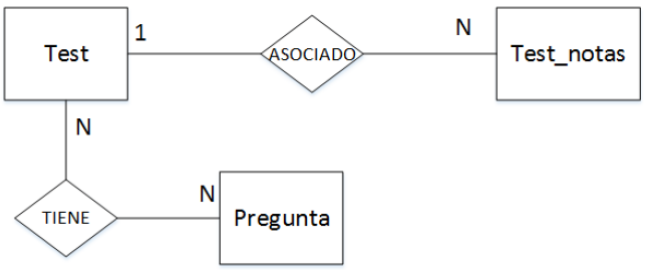


Figura 13: Diagrama Entidad - Relación de la base de datos.

La API se divide en diferentes peticiones a direcciones HTTP. Las direcciones que forman la API son las siguientes:

- /api/exams/: en el caso de hacer una petición GET, devuelve un JSON con la lista de todos los exámenes introducidos en el sistema. Los campos que contiene un examen son los siguientes: nombre, fecha, número de preguntas, número de respuestas por pregunta, número de tablas en la plantilla, código QR y preguntas de correción. En el caso de hacer una petición POST, se necesitan todos estos campos pasados en formato JSON, además se genera el contenido del código QR. Este código es único dentro de

la base de datos y está formado por 50 carácteres aleatorios. En realidad se crean 2 codigos QR, uno para la plantilla de corrección y otro para la plantilla de los alumnos. Por cada examen creado se inserta un proceso en un sistema de colas del servidor. Este proceso es una función para enviar emails a los alumnos notificándoles los resultados de las pruebas. Los emails son enviados cuando el reloj del sistema llega a la fecha de notificación estipulada en la creación del examen. Se ha utilizado el sistema *crontab* de Django para este propósito.

- /api/scores/: igual que en la dirección anterior, si se realiza una petición GET se obtiene el listado de todas las notas de los exámenes. Usando una petición POST, se insertan las respuestas utilizando un objeto en formato JSON y se devuelve el número de respuestas correctas, incorrectas y no contestadas.

- /api/checker/: Esta dirección solo acepta peticiones POST, además, el contenido de la petición tiene que contener una imagen. El objetivo es utilizar la cadena de procesado para detectar las respuestas de la plantilla y, después, devolver un objeto JSON con el contenido detectado. Antes de guardar los resultados en la base de datos, la detección debe ser confirmada por el usuario y enviada a /api/scores/ para registrar estos resultados.

La interfaz web tiene como objetivo ofrecer una herramienta para el profesado que sea capaz de administrar los exámenes y poder ver los resultados de los alumnos. Algunos ejemplos de esta interfaz son los mostrados en el apéndice A4.

4.4 Aplicación Android.

Esta última parte del sistema es la herramienta que se utiliza para capturar las imágenes de las plantillas. Es compatible con versiones de Android superiores a 4.1 Jelly Bean. También se utilizan los nuevos patrones de diseño de Material Design introducidos en la versión 5.0 Lollipop de Android [17]. Esto hace que esta aplicación sea compatible en la gran mayoría de dispositivos Android utilizando los nuevos formatos.

Se han creado dos aplicaciones diferentes para diferenciar las funciones que puede tener un profesor de las que puede tener un alumno. La diferencia entre estas dos aplicaciones reside en el hecho de que la aplicación del profesor permite saber el resultado de un examen al momento de enviar la foto. Por el contrario, el alumno no tendrá este privilegio, con esto se evitan filtraciones de respuestas en el tiempo del examen. Como se ha comentado antes, el alumno será notificado al finalizar el examen vía email.

El funcionamiento de esta aplicación es el mostrado en la Figura 14 y Figura 15 y sigue los siguientes pasos:

- Primero se realiza la fotografía de la plantilla. En este paso hay la posibilidad de cambiar el modo flash para mejorar la calidad de las imágenes.

- Una vez capturada la imagen, se realiza una vista previa para aceptar o descartar la imagen. Si la imagen es aceptada se enviará al servidor para su detección. Un ejemplo es el mostrado en la Figura 14.

- Cuando el servidor ha recibido la imagen y la ha procesado devuelve información relacionada con la plantilla y también los resultados de la detección. El usuario tiene que validar los datos detectados. Cuando estos resultados son aceptados, esta información se envía al servidor para que sea insertada en el sistema. Un ejemplo es la lista de respuestas mostrada en la Figura 15.

- En el caso de que la plantilla sea una plantilla de corrección, no se mostrará ningún tipo de información, simplemente será registrada en el servidor. Si se trata de una plantilla de estudiante, y se utiliza la aplicación del profesor, se mostraran los resultados de la detección. En caso de utilizar la aplicación del alumno, no se mostraran los resultados.

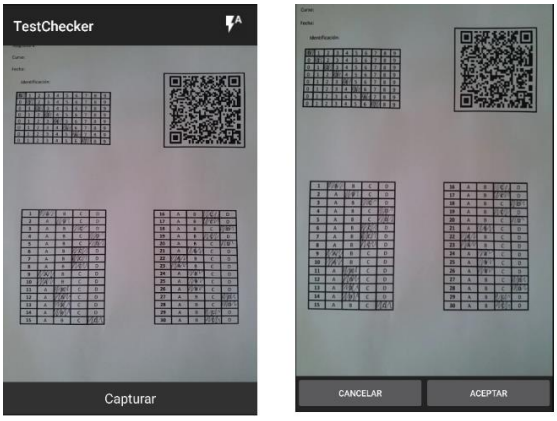


Figura 14: A la izquierda se muestra el modo cámara para capturar imágenes y a la derecha se muestra la confirmación de esta imagen.

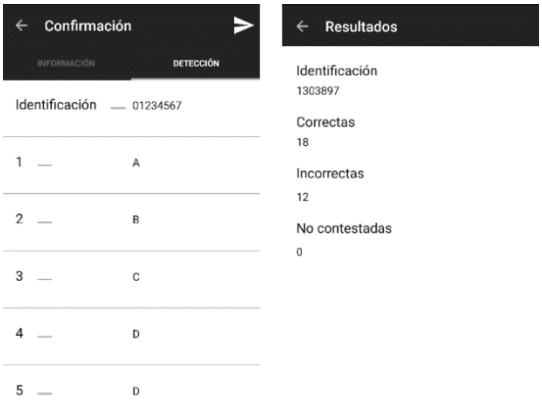


Figura 15: La imagen de la izquierda muestra los resultados de la detección. En ésta se pueden modificar los valores antes de ser enviados. En la imagen de la derecha, se muestran los resultados del test después de ser enviados al servidor para recibir la corrección. Estos resultados solo se muestran en la aplicación del profesor.

4.5 Funcionamiento general del sistema.

Todos los diferentes módulos explicados en los puntos anteriores funcionan de forma conjunta para obtener los resultados deseados. En este punto se detalla como se crea un examen tipo test en el sistema y también se ofrece una visión general de las etapas por las que pasa una imagen hasta que es corregida.

Para crear un examen en el sistema se tienen que seguir los siguientes pasos:

- Entrar en la sección para crear un nuevo examen y rellenar todos los campos que se piden del formulario.
- Entrar en el detalle del examen creado y obtener los dos códigos QR generados.
- Abrir una plantilla en un editor de texto y, en la tabla o tablas de preguntas, poner el número de preguntas y respuestas estipulado en la creación del examen.
- Copiar el código QR de correción en esta plantilla. También copiar el código QR del alumno en la misma plantilla pero en un fichero diferente.
- Guardar las plantillas e imprimirlas para detectarlas por el sistema.
- En la plantilla de corrección, marcar las respuestas correctas y utilizar la aplicación Android para guardar los resultados en el sistema.

En el esquema de la Figura 16 se pueden observar las diferentes etapas por las que pasa una imagen cuando entra en el servidor hasta que se consiguen evaluar los resultados. De esta manera se tiene una visión global de su funcionamiento.

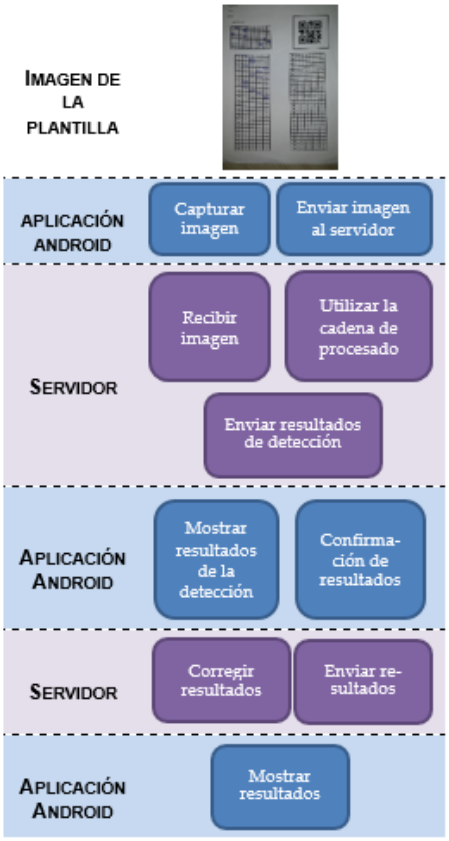


Figura 16: Etapas del funcionamiento general del sistema cuando se recibe una imagen.

5 RESULTADOS

El resultado del proyecto ha sido la creación de tres sistemas: una cadena de procesado con la creacion con diferentes módulos, un servidor web y una aplicación Android. Estos tres elementos han sido comentados a lo largo del documento pero no se ha evaluado como funcionan los tres conjuntamente. En este apartado se ofrecen los resultados obtenidos cuando el sistema trabaja con los tres elementos descritos. También se detallan cuáles son sus principales problemas.

Primero de todo, hay que decir que el detector funcionará mejor o peor en función de la calidad de la imagen. Se ha detectado que el detector de respuestas es sensible a la iluminación, por este motivo, se ofrece la posibilidad de utilizar el flash cuando se realizan capturas de plantillas en la aplicación Android.

Por lo general, el tiempo que tarda una plantilla en ser corregida una vez ha llegado al servidor no suele sobrepasar los 2 segundos. El tiempo total desde que se captura la imagen hasta que se recibe una respuesta depende del tiempo que se tarde en enviar la fotografía al servidor.

En el gráfico de la Figura 17 se muestra el número de respuestas detectadas correcta e incorrectamente por los diferentes detectores creados. Se basa en 8 imágenes con un total de 60 preguntas cada una. Los datos son valores promedios entre los dos detectores.

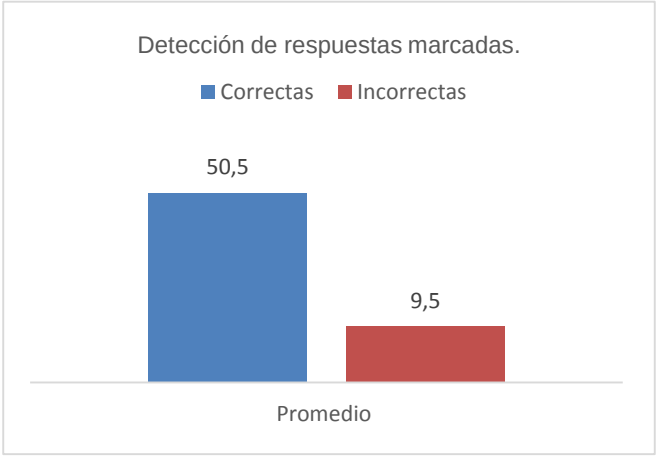


Figura 17: Valores promedios de los dos detectores.

Existe un problema con las respuestas que son detectadas de forma incorrecta. Este problema tiene que ver con diferentes factores. Uno de estos factores es el comentado anteriormente sobre el pequeño desplazamoniento que sufren los puntos de las tablas respecto al centro de masas de cada celda. Esto hace que los diferentes detectores cuenten para la detección los bordes que delimitan cada celda distorsionando así la detección. Otro factor importante es el preprocesado que se realiza a la imagen al inicio de la cadena de procesado. Dependiendo del color escogido para marcar las respuestas, con el preprocesado de la imagen se puede perder información de las respuestas marcadas. También, la resolución de las imágenes y las cualidades de la cámara son factores críticos. Si tenemos una resolución alta, el detector tendrá más detalle en cada celda. También la calidad de imagen hace

que las respuestas marcadas no sean tan sensibles al procesado de la imagen.

6 Líneas de trabajo futuras

Este proyecto inicia una base para la creación de un futuro sistema detector de plantillas tipo test. Como líneas de trabajo futura se podrían plantear las siguientes:

- Utilizar el servidor para recopilar datos de las detecciones y así tener un *Ground Thruth* con el que validar un futuro detector.
- Crear un detector más robusto con la capacidad de poder detectar diferentes símbolos a la hora de marcar las respuestas.
- Implementar medidas en la aplicación Android para determinar si una imagen está bien enfocada o no.
- Mejorar la interfaz web insertando filtros para las búsquedas de los exámenes.

7 Conclusiones

En la realización de este proyecto se ha podido ver todo el proceso de creación de un sistema complejo. El objetivo principal se ha cumplido ya que el sistema es funcional y los resultados que ofrece, en tiempo de respuesta y de detección, son satisfactorios. Los sistemas de servidor y aplicación Android se han cumplido con un resultado mejor de lo esperado. En un principio no se esperaba crear una página web para administrar el sistema, pero debido a diferentes reajustes en la planificación, se dispuso del suficiente tiempo para realizar esta tarea. También, la aplicación Android ha respondido a las expectativas creadas y ha quedado una aplicación acabada y funcional.

Otro aspecto muy importante ha sido la inclusión de los códigos QR en el sistema, ya que gracias a esta inserción, se simplificó la cadena y el sistema pasó a ser más robusto. Este cambio se implementó rápidamente y no supuso grandes cambios en la planificación. También ha sido necesario documentarse suficiente para poder entender y utilizar correctamente los diferentes algoritmos utilizados en la creación de la cadena de procesado.

Como conclusión final de este trabajo se ha conseguido analizar correctamente las imágenes de las plantillas, detectar las características de éstas con el objetivo de procesarlas y detectar las respuestas marcadas para después devolver un resultado. Para todo ello se ha utilizado el sistema creado: cadena de procesado, servidor y aplicación Android.

Agradecimientos

Quiero mostrar mi agradecimiento a Javier Sánchez Pujadas, tutor de este trabajo de final de grado. He recibido su soporte y, en muchos casos, ha aportado grandes ideas para el proyecto. También agradecer el apoyo y la ayuda recibida por mis compañeros de Starlab.

Referencias

[1] Servicio de informática UAB, Corrección de exámenes. Disponible en https://www.uab.cat/web/correccio-d-examens-1096478501685.html, Junio 2015

[2] Francisco de Assis Zampirolli, Jose Artur Quilici Gonzalez and Rogério Perino de Oliveira Neves, "Automatic Correction of Multiple-Choice Tests using Digital Cameras and Image Processing". Universidade Federal do ABC, 2010

[3] Darío Álvarez Gutierrez, Lucas Díaz, Sanzo. "COETEST: Corrección óptica de exámenes de test en papel automática, rápida y económica". Universidad de Oviedo, Julio 2011.

[4] NumPy package Python disponible en http://www.numpy.org/ 2013

[5] OpenCV library. 2015

[6] Raquel Urtasun, Computer Vision: Filtering. TTI Chicago, 2013 http://www.cs.toronto.edu/~urtasun/courses/CV/lecture02.pdf

[7] Jean Serra, "Image Analysis and Mathematical Morphology". January 1983.

[8] Bryan S. Morse, "Thresholding". Brigham Young University 1998 – 2000.

[9] John Hersberg, Jack Snoeyink. "Speeding Up the Douglas-Peucker Line-Simplification Algorithm". Palo Alto and Universitiy of British Columbia, 1992.

[10] Sitio official de la librería zlib. 2014. http://www.zlib.net/

[11] Sitio oficial de la librería QRTools, 2014. https://launchpad.net/qr-tools

[12] Denis Zorin, "2D transformations, homogeneous coordinates, hierarchical transformations". New York University 2001. http://mrl.nyu.edu/~dzorin/ig04/lecture05/lecture05.pdf

[13] OpenCV library, "función bitwise_and". 2015 http://docs.opencv.org/modules/core/doc/operations_on_arrays.html#bitwise_and

[14] OpenCV library, "función moments". 2015 http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=moments#moments

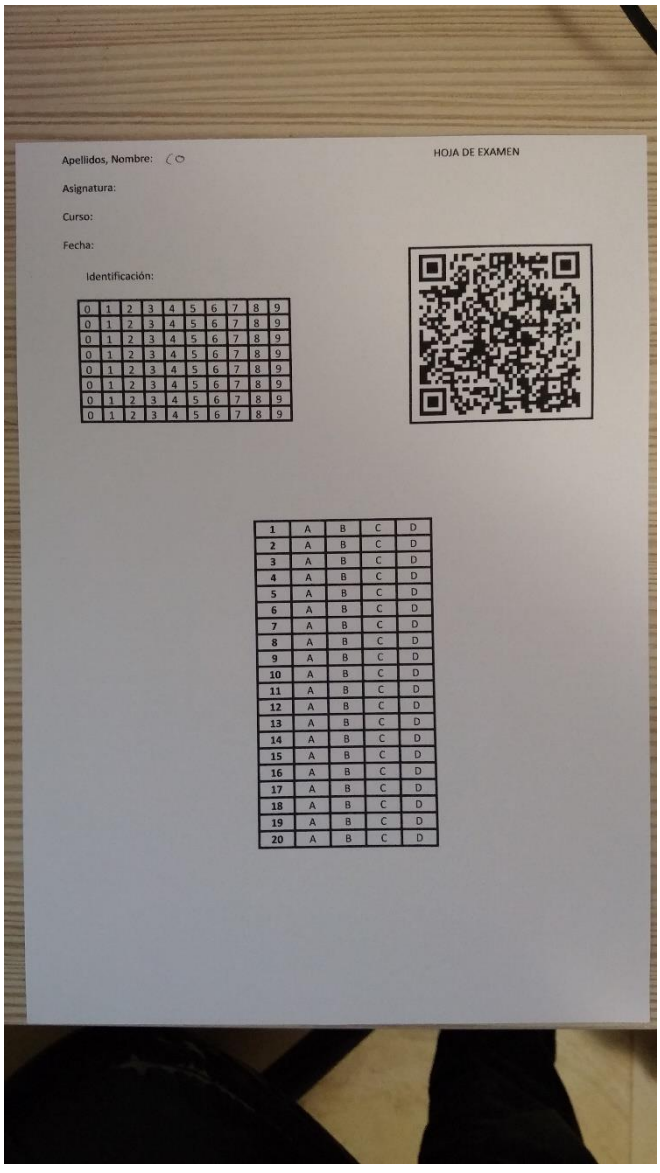
[15] Chapter 1, "Introduction to Moments". http://zoi.utia.cas.cz/files/chapter_moments_color1.pdf

[16] Django REST Framework, 2015. http://www.django-rest-framework.org/

[17] Ayuda Android developers, 2015. http://developer.android.com/index.html

APÉNDICES

A1. EJEMPLO DE PLANTILLA



A2. PLANTILLA BASE.

Apellidos, Nombre: HOJA DE EXAMEN

Asignatura:

Curso:

Fecha:

Identificación:

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

1	A	B	C	D	E	F	G	H	I	J	K	L
2	A	B	C	D	E	F	G	H	I	J	K	L
3	A	B	C	D	E	F	G	H	I	J	K	L
4	A	B	C	D	E	F	G	H	I	J	K	L
5	A	B	C	D	E	F	G	H	I	J	K	L
6	A	B	C	D	E	F	G	H	I	J	K	L
7	A	B	C	D	E	F	G	H	I	J	K	L
8	A	B	C	D	E	F	G	H	I	J	K	L
9	A	B	C	D	E	F	G	H	I	J	K	L
10	A	B	C	D	E	F	G	H	I	J	K	L
11	A	B	C	D	E	F	G	H	I	J	K	L
12	A	B	C	D	E	F	G	H	I	J	K	L
13	A	B	C	D	E	F	G	H	I	J	K	L
14	A	B	C	D	E	F	G	H	I	J	K	L
15	A	B	C	D	E	F	G	H	I	J	K	L
16	A	B	C	D	E	F	G	H	I	J	K	L
17	A	B	C	D	E	F	G	H	I	J	K	L
18	A	B	C	D	E	F	G	H	I	J	K	L
19	A	B	C	D	E	F	G	H	I	J	K	L
20	A	B	C	D	E	F	G	H	I	J	K	L
21	A	B	C	D	E	F	G	H	I	J	K	L
22	A	B	C	D	E	F	G	H	I	J	K	L
23	A	B	C	D	E	F	G	H	I	J	K	L
24	A	B	C	D	E	F	G	H	I	J	K	L
25	A	B	C	D	E	F	G	H	I	J	K	L
26	A	B	C	D	E	F	G	H	I	J	K	L
27	A	B	C	D	E	F	G	H	I	J	K	L
28	A	B	C	D	E	F	G	H	I	J	K	L
29	A	B	C	D	E	F	G	H	I	J	K	L
30	A	B	C	D	E	F	G	H	I	J	K	L

A3. COMPARATIVA DIFERENTES METRICAS DE COMPARACION DE HISTOGRAMAS.

Este apendice muestra una tabla comparativa entre las diferentes métricas utilizadas en el detector de comparación de histrogramas. Estos resultados evalúan el número de respuestas detectadas correctamente y el número de respuestas que han sido detectadas con un valor diferente al que en realidad se ha marcado en la hoja del test. Los resultados obtenidos con este método son parecidos en las diferentes métricas. Estos resultados no nos aportan una decisión definitiva y se deberían realizar muchas más pruebas para poder tener una decisión más concluyente sobre cuál de estas métricas es mejor. También, para evaluar estas métricas se tiene que definir un umbral que permita determinar si una respuesta está marcada o no. Este umbral es un punto crítico ya que dependiendo de su valor los resultados pueden variar.

Tabla 1: Comparativa de métricas.

	Métrica							
	Correlation		Chi-Square		Intersection		Bhattacharyya distance	
Imagen	Correc-tas	Incorrectas	Correctas	Incorrectas	Correctas	Incorrectas	Correctas	Incorrectas
1	57	11	56	12	57	11	51	17
2	59	9	57	11	59	9	47	21
3	59	9	56	12	58	11	55	13
4	59	9	57	11	35	9	55	13
5	46	22	56	23	47	21	44	24
6	49	19	57	22	30	38	46	22
7	65	3	62	6	65	3	56	12
8	66	2	63	5	66	2	60	8º
PROMEDIO	57,5	10,5	55	12,75	52,125	15,87	51,75	16,25

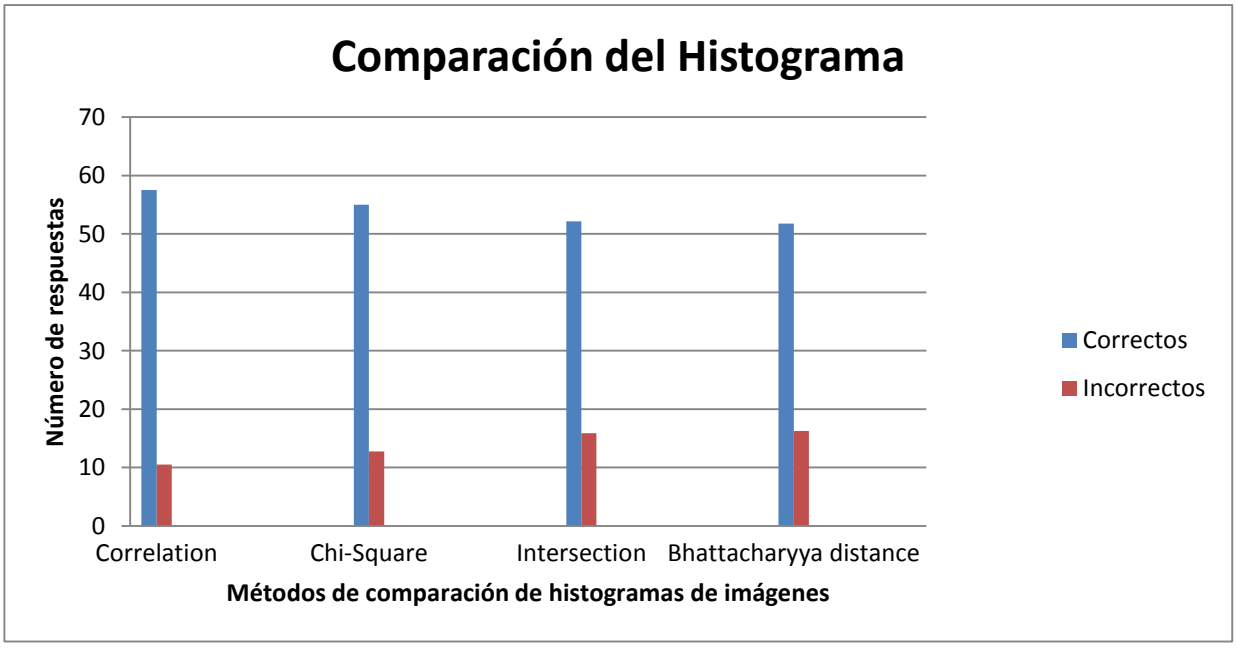


Figura 18: Gráfico comparativo de métricas.

A4. EJEMPLOS DE LA INTERFAZ DEL SERVIDOR.

En este apencice se muestran algunos ejemplos de la interzar web del servidor.

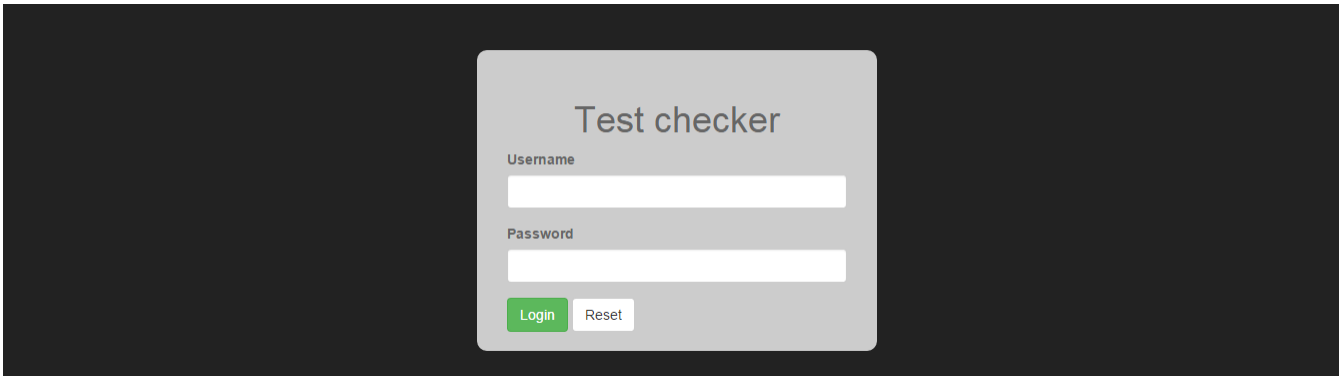


Figura 19: Formulario de inicio de sesión en la web del servidor.

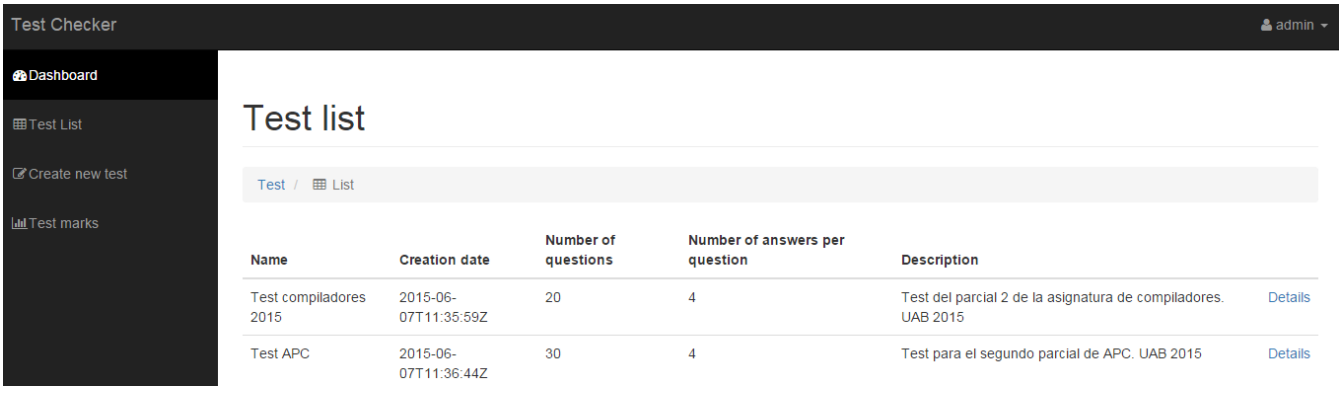


Figura 20: Listado de exámenes.

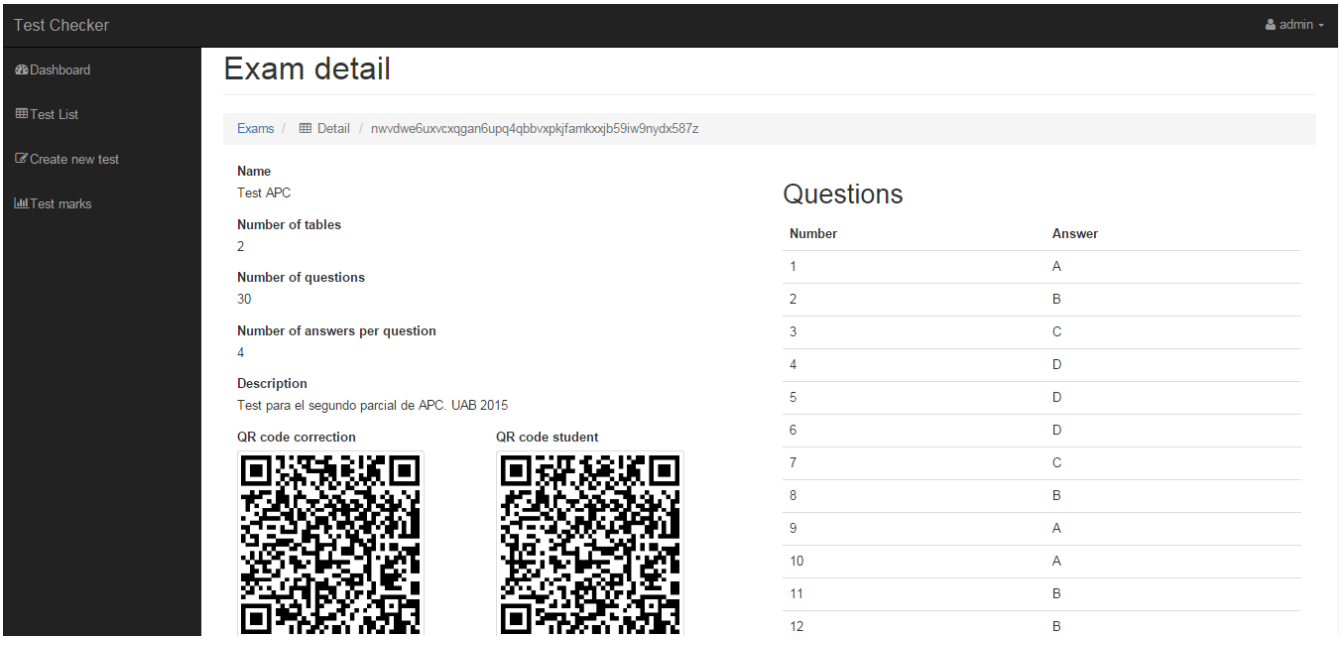


Figura 21: Detalles de una exámen.

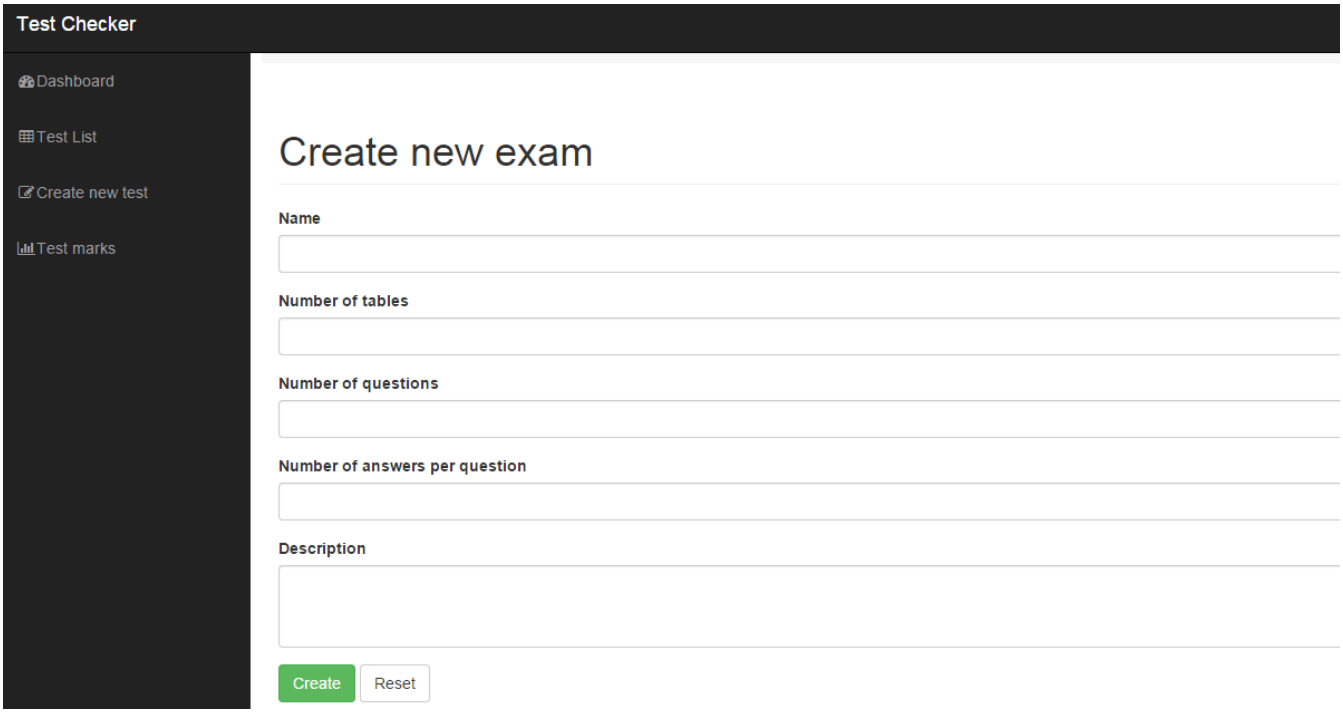


Figura 22: Formulario de creación de un nuevo exámen.

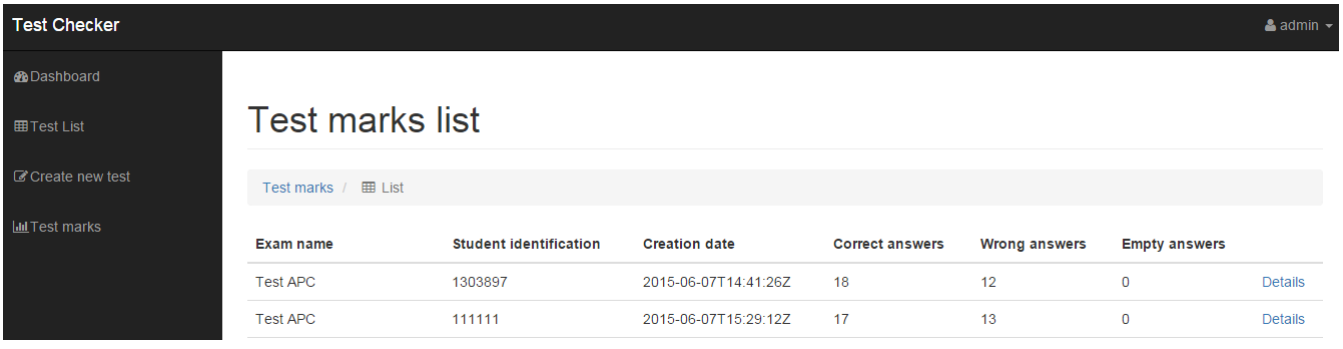


Figura 3: Listado de resultados de los exámenes.

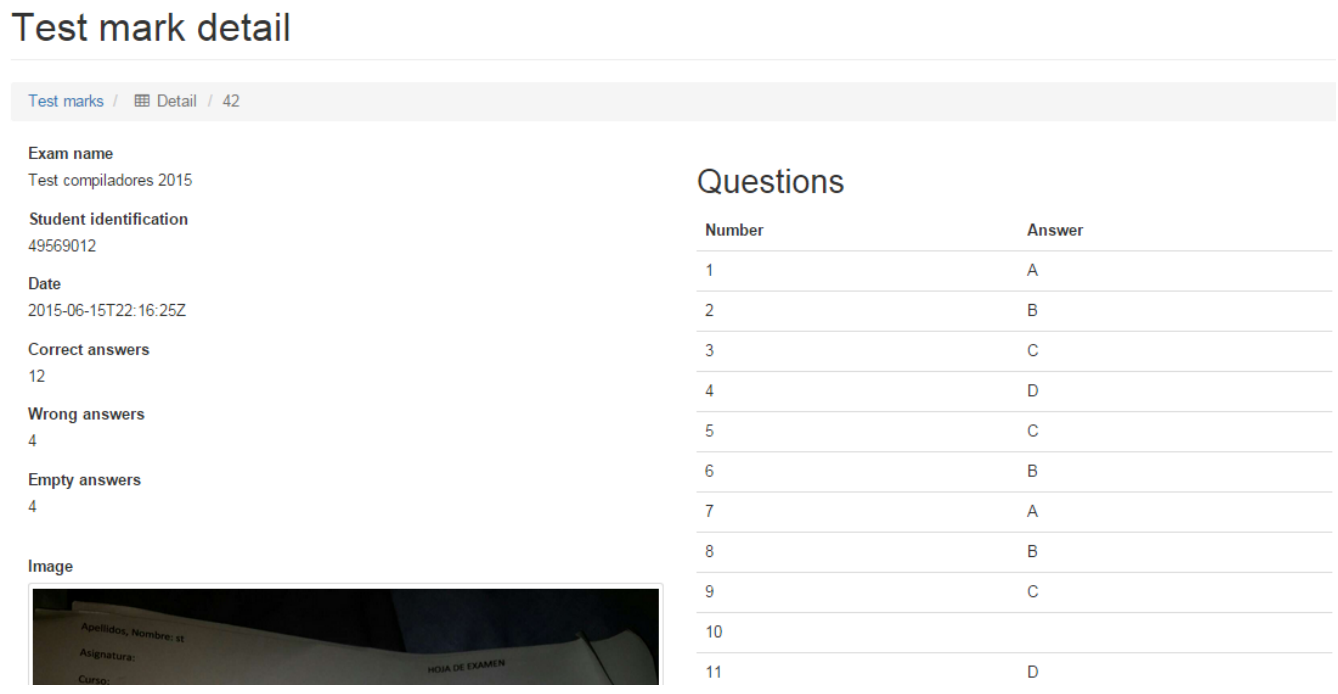


Figura 2: Detalles de un resultado añadido.