App móvil para reconocer texto en imágenes

Víctor Manuel Montes Llorente

Resumen —. Los lectores OCR son una herramienta muy útil para poder escanear imágenes que disponen de texto y poder obtener posteriormente este texto. Esta es la idea principal de este proyecto, pero con el añadido de que este integrado en una aplicación Android que permita a cualquier usuario, a través de una imagen obtenida a partir de la cámara o el almacenamiento interno del dispositivo, que pueda obtener el texto de la imagen. Para realizar esta tarea, ha sido proporcionado el núcleo de procesamiento del OCR en el lenguaje C++, este núcleo se procesa en dos partes, la primera parte es procesada dentro del dispositivo Android, y la segunda es procesada a través de un servidor Apache, que posteriormente será el encargado de devolver el resultado al dispositivo, esto permitirá al usuario poder utilizar un traductor también integrado en la propia aplicación.

Palabras clave — OCR, C++, nucleo, Android, Apache, dispositivo, traductor, aplicación, L-OCR

Abstract — OCR readers are really a useful tool for scanning text-containing images and retrieve the text within them. That's the main target of this project, but adding an Android integration that allow the users to take a picture, either from the camera or internal storage, and get the text contained in it. To achieve that, an OCR core code developed in C++ has been provided. This core runs in two parts: the first one is handled by the Android device itself, and the other one is processed through an Apache server that will later provide the result to the device. That will allow the user to have, indeed, a real-time translator built-in in the app itself.

Index Terms — OCR, C++, core, Android, Apache, device, translator, aplication, L-OCR

1 Introducción

Actualmente existe un claro auge en el desarrollo de aplicaciones para móviles, dado que cada vez es más sencillo desarrollar para dispositivos y lo más importante, es más fácil que estas aplicaciones lleguen a los usuarios. Esta facilidad de acceso es posible gracias a sistemas operativos móviles como Android o IOS, que integran sus propios markets en los que los desarrolladores pueden subir sus aplicaciones, y posteriormente, los usuarios pueden descargarlas.

Existen infinidad de aplicaciones para dispositivos móviles como podrían ser aplicaciones de edición de imágenes, de grabación de video, lectores QR y muchas más.

En este proyecto nos vamos a enfocar concretamente a los denominados lectores OCRs. Un lector de OCR es un tipo de software que es capaz de detectar texto mecanoscrito en una imagen, y posteriormente devuelve este texto en un formato editable para el usuario. Es importante no confundir OCRs con ICRs, puesto que un ICR es un software que se encarga de detectar texto manuscrito y no mecanoscrito.

Los lectores OCRs son cada vez más conocidos por la gente, dado que dispositivos como algunas impresoras, disponen de ellos, cosa que permite a las personas poder transformar fácilmente un documento en papel a un documento de texto, como por ejemplo a un documento Word. Pero este tipo de lectores también pueden tener otro tipo de aplicaciones, como podría ser la obtención de texto a partir de una fotografía realizada con un dispositivo móvil. Este tipo de aplicación es la que se ha realizado para este proyecto, concretamente ha sido creada para ser ejecutada en dispositivos con un sistema operativo Android.
En las siguientes secciones se explica con más detalle el objetivo de la aplicación la que será nombrada durante el documento como L-OCR, como se ha desarrollado, problemas encontrados y diferentes puntos para entender el proceso llevado a cabo en su desarrollo.

1.1 Objetivos y subobjetivos

El objetivo principal de este proyecto ha sido la creación de una aplicación Android que permitiera al usuario obtener el texto de una imagen a través de una captura realizada por la cámara del dispositivo, u obtenida desde el almacenaje interno de este. Una vez se dispone de la imagen, se le permite al usuario seleccionar que área de la imagen que desea que sea procesada. Posteriormente se obtiene el resultado de la imagen procesada con texto encontrado en

- *E-mail de contacto: VictorManuel.Montes@e-campus.uab.cat*
- *Mención realitzada: Ingeniería del Software*
- *Trabajo tutoritzado por: Ernest Valveny (Departamento Ciencias de Computación)*
- *Curso 2014/15*

esta, este texto puede ser copiado en el portapapeles del dispositivo Android, o puede ser traducido por un traductor simultáneo que incorpora la propia aplicación. Para cumplir el objetivo expuesto se ha dividido el trabajo en varios objetivos y subobjetivos que a continuación serán enumerados.

1.2 Objetivos

- Creación de una interfaz Android que permita al usuario realizar las siguientes tareas:
    - Permitir al usuario obtener una imagen a través de la cámara, o a través del almacenaje interno del dispositivo.
    - Permitir el recorte y rotación de la imagen obtenida.
- Añadir un procesador de OCRs en C++ en la aplicación. Esta integración se divide en dos fases:
    - Fase 1: Integración de una primera parte del procesador OCR en dentro de la propia aplicación Android, teniendo en cuenta que el código a integrar esta creado en C++ y en Android se programa en JAVA, por lo tanto se debe buscar la forma de comunicar estos dos lenguajes.

      Esta primera parte es la encargada de obtener el localizamiento de las palabras dentro de una imagen, para así posteriormente recortar estas imágenes y extraerlas. En la figura 1 podemos ver un ejemplo de cómo funciona esta primera fase.

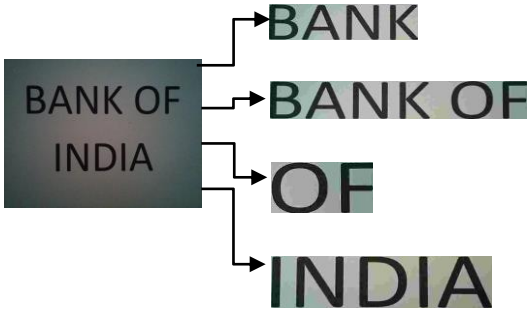


Figura 1: resultado de extracción de texto

    - Fase 2: En esta fase, se busca que a partir de las imágenes extraídas en la primera fase de localización del texto, que se procese cada una de las imágenes obtenidas y se intenta reconocer que palabra es la contenida en la imagen. Una vez reconocida se devuelve el resultado encontrado. Este proceso se debe hacer con cada una de las imágenes para finalmente obtener el resultado conjunto obtenido del procesamiento de todas ellas.

1.3 Subobjetivos

- Permitir al usuario copiar el resultado final de la aplicación en el portapapeles del dispositivo Android.
- Integrar un traductor simultáneo.

1.4 Restricciones

Para este proyecto se cuentan con diversas restricciones, estas están pensadas para que el dispositivo deba soportar la mínima carga posible, por lo que para realizar esto se deben tener en cuenta las siguientes restricciones:

- Las imágenes que sean procesadas desde el dispositivo deben ser lo más pequeñas posibles, por esto se le da la opción al usuario de recortar que parte de la imagen desea que sea procesada.
- Se debe disponer de un terminal con acceso a internet.
- Las imágenes que sean enviadas al servidor deben pesar lo mínimo posible, en magnitud de Kilobytes. Esto debe ser así, ya que para subir las imágenes al servidor y que estas sean procesadas, se deben enviar a través de una conexión a internet, y muchas compañías telefónicas restringen la cantidad de datos que puedes enviar y recibir mensualmente.
- La aplicación solo puede ser ejecutada desde un terminal Android.
- El traductor de la aplicación está limitado a 2000 palabras/mes, esto es debido a que la implementación de este servicio se realiza con la API de BING, y tan solo permite realizar una traducción de 2000 palabras mensuales de forma gratuita, si se desea poder traducir más palabras, se debe pagar por el servicio.

1.5 Estado del arte

Para la realización de este proyecto se cuenta con código C++ proporcionados por Ernest Valveny del departamento de ciencias de la computación. Este código es el núcleo de la aplicación L-OCR, y se encarga de procesar una imagen con el objetivo de encontrar texto en esta, y posteriormente retornan una cadena de texto con el resultado encontrado.

Este núcleo se divide en dos partes, la primera es la encargada de realizar una localización de texto dentro de una imagen generando diversas imágenes por cada palabra localizada, y la segunda parte, es la encargada de procesar todas las imágenes obtenidas por la primera parte del núcleo, realizando así un reconocimiento de la palabra contenida en cada una de las imágenes. El código de estos núcleos está basado en las librerías de OpenCV y VLfeat de

C++, ambas son OpenSource. OpenCV es una librería para la realización de proyectos de machine learning, en cambio, VLfeat es una librería que implementa algoritmos de visión por computador.

Para poder dar uso a estos núcleos se debe disponer de una aplicación Android.

2 METODOLOGÍA

Existen diversos marcos de desarrollo agiles, pero para la realización de este proyecto se ha escogido SCRUM.

2.1 SCRUM

SCRUM es un marco de desarrollo ágil que se basa en ciclos de trabajo que son denominados sprints, en el que al final de cada sprint se dispone de un entregable, por lo tanto, es importante marcarse un objetivo para cada sprint cuando se planifica el proyecto.

En SCRUM existen tres tipos de roles, que son:

- Product Owner: Es el cliente de la aplicación.
- SCRUM Master: Es el encargado de guiar durante el proceso de desarrollo según los principios de la propia metodología.
- SCRUM Team: El conjunto de personas que forman el equipo de trabajo.

Como es de esperar, al ser un proyecto desarrollado por una única persona, todos los roles recaen sobre un único individuo, excepto el Product Owner, que se podría asignar al tutor del proyecto.

El ciclo de un sprint de scrum se puede ver en la siguiente imagen:

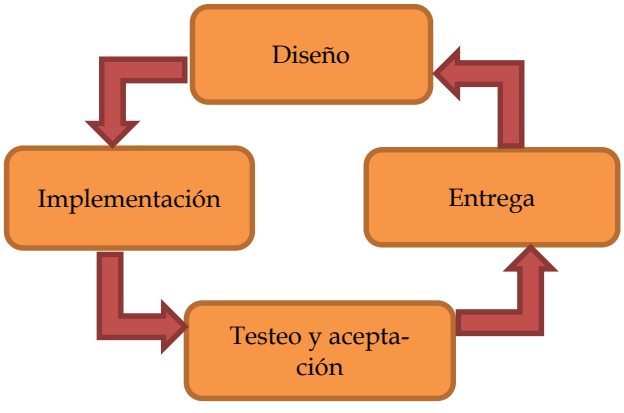


Figura 2: Ciclo de vida Sprint

Tal y como se puede apreciar en la figura X los puntos a llevar a cabo en cada sprint son los siguientes:

- Diseño: A partir de los datos obtenidos se intenta plasmar como se debe proceder para la realización del proyecto.
- Implementación: Se genera el código necesario para llevar a cabo el objetivo del sprint.
- Testeo y aceptación: Se realiza pruebas de la implementación para verificar que funciona correctamente, y si es así se acepta.
- Entrega: Se realiza un entregable de un sprint concreto al cliente.

En Scrum, se pueden tener tantos sprints como sean necesarios, definiendo objetivos y requisitos para cada uno de ellos, que podrán ser modificados antes de la realización de un sprint, pero nunca durante. Y contra mas sprints se realicen, mas pulida podrá estar la aplicación.

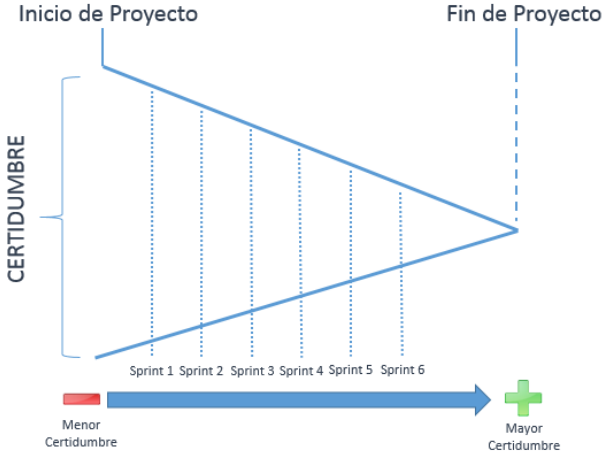


Figura 3: Certidumbre del proyecto según los sprints

Esta metodología ha sido la más idónea para la realización de este proyecto, ya que cada mes se debía realizar un entregable al tutor del TFG, y las entregas a realizar eran cuatro, por lo que este proyecto se ha compuesto de 4 sprints de 1 mes cada uno.
Los objetivos y entregables a realizar en cada uno de los sprints han sido los siguientes:

- Sprint 1:
  - Entregable: Informe Inicial
  - Objetivo: Implementación de la primera parte del núcleo en Android mediante NDK.
- Sprint 2:
  - Entregable: Informe de progreso I
  - Objetivo: Implementación de la interfaz Android.

- Sprint 3:
  - Entregable: Informe de progreso II
  - Objetivo: Creación de un WebService para la ejecución de la segunda parte del núcleo del OCR, e implementación del cliente en Android.
- Sprint 4:
  - Entregable: Propuesta del articulo
  - Objetivo: Implementar un traductor y permitir al usuario copiar el texto obtenido del L-OCR.

3 DESARROLLO

3.1 Interfaz

En toda aplicación debe haber un diseño previo de la interfaz de usuario, esta interfaz puede ser modificada si fuera necesario durante el transcurso de desarrollo, pero es importante disponer de una idea inicial y sólida de la que partir. Inicialmente para el proyecto se diseñó la siguiente interfaz:

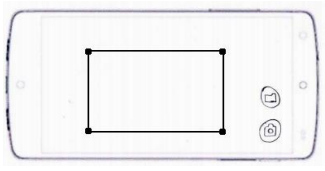


Figura 4: Pantalla inicial

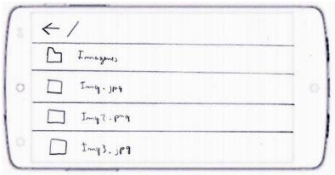


Figura 5: Galería

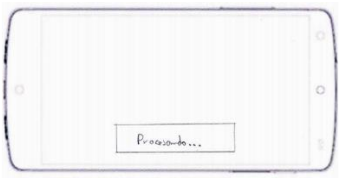


Figura 6: Procesamiento imagen

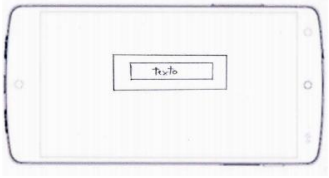


Figura 7: Resultado

Como podemos observar, en la pantalla inicial, se cuenta con un selector de imagen, para capturar con la cámara, tan solo lo que hay dentro del recuadro. También disponemos de dos botones, el que contiene la imagen de una cámara, permite realizar la foto, y el botón con una carpeta, nos permite ir hacia una galería (Figura 5), que nos permite seleccionar una imagen del dispositivo para procesar. Posteriormente se muestra al usuario que se está procesando, para que finalmente devuelva el resultado obtenido de la imagen.

En la interfaz final, el selector de imagen ha sido separado de la pantalla inicial, permitiendo al usuario recortar también una imagen obtenida directamente del dispositivo, añadiendo también la posibilidad de rotar la imagen si fuera necesario. También se ha añadido una pantalla más, que contiene el traductor simultáneo de la aplicación.

A continuación se muestra el diseño final de la interfaz ya implementada en la aplicación L-OCR:

Figura 8: Pantalla Inicial

Figura 9: Selector imagen

Figura 10: Procesando

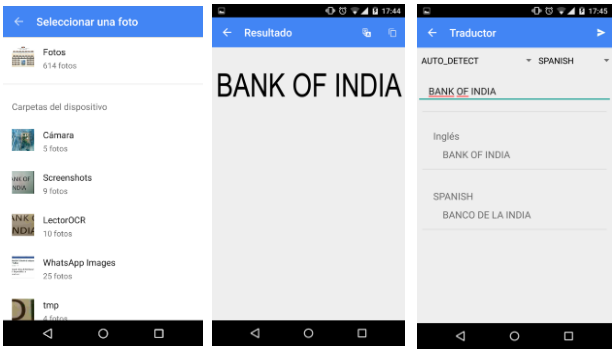


Figura 11: Galería Figura 12: Resultado Figura 13: Traductor

Nota: El resultado obtenido en la figura 12 es un resultado esperado, pero no tiene por qué corresponder con el resultado real actual del L-OCR.

Tanto desde la figura 8 haciendo una foto, como desde la figura 11, obteniendo la imagen de la galería, llegaríamos a la pantalla que se muestra en la figura 9. En esta pantalla se puede seleccionar que parte de la imagen se desea procesar o si se desea rotar la imagen, para posteriormente procesarla, y el resultado obtenido, es mostrado en la pantalla de la figura 12, desde la que se puede copiar el texto, o si se desea, ir al traductor del L-OCR.

Para el diseño de esta interfaz han sido usados componentes de Android L, como el uso de una toolbar en cada activity, en la que se han incorporado todos los botones necesarios para gestionar el funcionamiento de la aplicación. Como se puede apreciar en las imágenes de la figura 8 a la 13, no se ha añadido el logo de la aplicación como se hacía en versiones anteriores de Android en los ActionBar, esto es así, debido que a partir de la versión 5.0 (Android L), no es recomendable hacer uso de los logos dentro de la aplicación.

3.2 Decisiones de desarrollo

Inicialmente se deseaba implementar ambas partes del núcleo del L-OCR dentro de la propia aplicación Android. Por lo que así se empezó y se logró implementar la primera parte del núcleo dentro de la aplicación Android sin problema utilizando el NDK de Android, que permite comunicar código C++ o C con código JAVA.

Posteriormente se intentó integrar de la misma manera la segunda parte del núcleo, pero dado que esta parte del utiliza la librería VLfeat que no tiene soporte nativo para NDK, no se logró incorporar esta librería al proyecto y por lo tanto el código de la segunda parte del núcleo no podía ser integrado. Dada esta situación, se decidió que esta parte fuera integrada dentro de un WebService que sería el encargado de ejecutar esta parte del núcleo empaquetado en un .exe, la ejecución seria realizada al recibir una petición desde el dispositivo Android, y posteriormente el WebService recogería los resultados obtenidos por parte de la segunda parte del núcleo y se los enviaría al dispositivo.

3.3 Esquemas de funcionamiento

En este subapartado se procede a mostrar el funcionamiento interno de las dos partes del núcleo del L-OCR y del traductor de la aplicación.

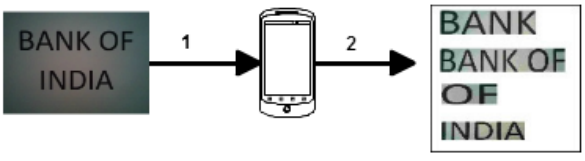


Figura 14: Procesamiento desde el dispositivo

La figura 14 representa el esquema de funcionamiento de la primera parte de procesamiento del L-OCR, este procesamiento se realiza íntegramente en el dispositivo Android. En esta primera parte del núcleo, el dispositivo se encarga de procesar una imagen de manera que realiza una búsqueda dentro de la imagen para localizar texto, si se localiza el texto, el procesador devuelve rectángulos que representan el posicionamiento de este texto dentro de la imagen, y mediante estos recuadros de posicionamiento se extrae diferentes imágenes de cada una de las palabras localizadas. Estas imágenes serán procesadas en el servidor como veremos a continuación.

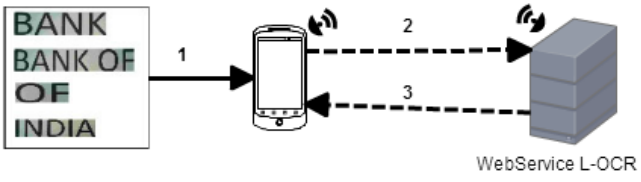


Figura 15: Procesamiento desde el servidor

La figura 15, representa el funcionamiento de la segunda parte del procesamiento del L-OCR. En esta parte, el dispositivo obtiene una a una todas las imágenes que se han extraído de la primera parte del procesador, cada una de estas imágenes es subida individualmente al servidor, este se encarga de vectorizar las imágenes, para que posteriormente, a través de un diccionario de palabras, que contiene también sus palabras vectorizadas, compare los vectores del diccionario junto al extraído de la imagen, y aquel vector con una mayor similitud al vector de la imagen, será el que representara a la palabra que estaba contenida en la imagen, y se obtendrá una cadena de texto representativa de la imagen. Esta palabra se devuelve al dispositivo en forma de cadena de texto, y una vez el dispositivo deja de enviar imágenes al servidor para que este le devuelva la palabra resultante, entonces, se muestra el resultado conjunto de todas las imágenes procesadas por pantalla. Este resultado puede ser utilizado luego para ser llevado al traductor integrado, el esquema de funcionamiento del traductor es el siguiente.

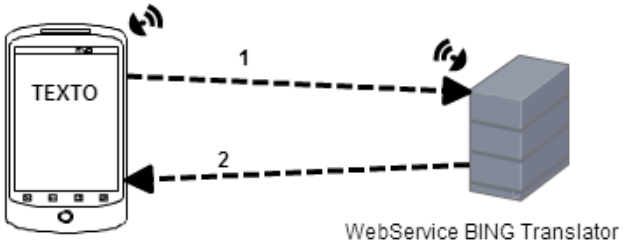


Figura 16: BING Translator

Como podemos ver el esquema del traductor es muy similar al del procesamiento desde el servidor (Figura 15), estos es debido a que ambos utilizan un WebService para funcionar, pero en el caso del traductor, el WebService ha sido implementado por Microsoft, y ellos proporcionan la API de Android que te permite establecer una conexión con este servicio. El funcionamiento es tan simple como indicarle al servicio de BING en que idioma está el texto y a que idioma se quiere traducir, se realiza la petición junto al texto a traducir y este te devuelve la respuesta de la traducción. Es importante tener en cuenta que para que este servicio funcione, es necesario tener una cuenta en cuenta, que para poder utilizar este servicio, se necesita una cuenta en Microsoft Azure Marketplace, y posteriormente solicitar unos datos para que se permita utilizar el servicio.

3.4 Implementación del núcleo

Como bien ha sido explicado en el esquema de funcionamiento, el núcleo se divide en la parte de procesamiento de Android y la parte de procesamiento por parte del servidor.

Para realizar la implementación de la primera parte del núcleo de procesamiento del L-OCR en Android se ha utilizado el denominado NDK (Native Development Kit). NDK es una interfaz de programación para desarrollo de aplicaciones nativas, es decir, NDK proporciona una interfaz que permite comunicar código C o C++ con Java, esta interfaz es denominada JNI (Java Native Interface). NDK,

a parte de contener JNI, también dispone de todo lo necesario para que se pueda compilar un proyecto de forma que pueda ser ejecutado desde cualquier dispositivo Android, para lograr esto, incorpora su propio compilador que permite compilar en arquitecturas como armeabi, armeabi-v7 y x86, para realizar la compilación en diferentes arquitecturas, NDK necesita de dos archivos de configuración nombrados como Android.mk y Application.mk, en el caso de Android.mk se deben especificar parámetros como donde se encuentra situado el compilador, que archivos .c o .cpp queremos que sean compilados junto a nuestro proyecto Android, o el nombre de la librería nativa que llamaremos desde JAVA. En el caso de Application.ndk, se especifica parámetros como las arquitecturas de compilación a las que será compiladas el proyecto o la plataforma para la que será compilada, en el caso de la aplicación L-OCR, las arquitecturas escogidas son armeabi-v7 y x86, esto es asi ya que la gran mayoría de dispositivos Android que se utilizan hoy en día utilizan estas arquitecturas. También es importante conocer que la estructura típica de los proyectos Android es modificada ligeramente al incorporar NDK en nuestro proyecto, dado que para poder incorporar nuestro código C o C++, se necesita de un directorio nombrado como jni que contiene todas las cabeceras .h y todos los archivos .c o .cpp junto a los dos .mk de configuración. En la siguiente figura podemos apreciar el esquema de la interfaz JNI.

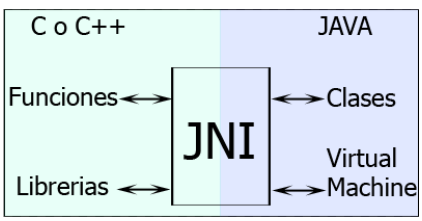


Figura 17: Esquema de JNI

Por otro lado, la implementación de la segunda parte del procesador del L-OCR ha sido realizada mediante el lenguaje de programación PHP procesado en un servidor Apache. En este caso, nuestro código C++ es procesado desde el servidor, y para realizar esto se necesita un .exe de la segunda parte del núcleo junto con las .dll pertinentes para su correcta ejecución, estas .dll son de las librerías de OpenCV y Vlfeat. La ejecución del .exe es lenta, por lo que se ha tenido que ampliar el tiempo que el servidor Apache permite para que un proceso este activo en espera de respuesta, este tiempo ha sido aumentado de 30s a 120s por petición.

4 TESTS

Para el desarrollo de la aplicación se han realizado algunos tests que a continuación se expondrán.

Se ha realizado un test para comprobar cuál es la experiencia de uso de la aplicación, teniendo en cuenta cosas como la facilidad para detectar funcionalidades de la aplicación por parte del usuario o como de agradable le resultaba a la vista el contenido que veía en la propia aplicación. Para realizar este test, se pidió a diferentes compañeros de la universidad, a amigos y familiares, si podían utilizar la aplicación sin que se les explicara cómo funcionaba, tan solo se exponía cual era el objetivo de aplicación. Y a partir del posterior uso de la aplicación se obtuvieron diferentes opiniones como las que se ven a continuación:

- Algunos botones no quedan totalmente claros para que sirven.

- Los colores de la aplicación no parecen los más adecuados.

- El selector de imagen no queda claro cómo funciona.

Estos tres comentarios nombrados anteriormente fueron los más comunes, por lo tanto son los que se escogieron como referencia y a partir de aquí se realizaron diversos cambios en la aplicación, hasta que finalmente el resultado de la interfaz es el que se puede apreciar de la figura 9 a la figura 13.
Por otro lado, durante la implementación del código, se fueron realizando pruebas unitarias de cada una de las funciones que fueron creadas para aplicación, una vez se tenían testeadas las diferentes funciones individualmente se realizaba un test de conjunto en las que se verificaba que dado un INPUT, al interactuar las funciones entre sí, acababan devolviendo el OUTPUT esperado. Los test de conjunto solo se ejecutaron en el código Android, puesto que el código del servidor tan solo dispone de una función Main, por lo que el código del servidor tan solo se testeo con un test unitario.

5 INCIDENCIAS EN EL DESARROLLO DEL PROYECTO

Durante el transcurso de este proyecto han ido apareciendo diferentes problemas.

Inicialmente se quería crear una aplicación en la que el núcleo del procesador del L-OCR estuviera todo implementado dentro del dispositivo Android. Inicialmente esto se logró con lo que hemos denominado durante todo el articulo como la primera parte del núcleo del procesador, esta parte se logró integrar perfectamente dentro de la propia aplicación mediante NDK, esto se pudo hacer puesto que las únicas librerías diferentes a las estándares de C++ que incorporaba el proyecto, eran las librerías de OpenCV, estas librerías están preparadas para podes ser importadas dentro de un proyecto Android con NDK. El problema surgió en la segunda parte del procesador del L-OCR, esta parte también incorpora las librerías VLfeat a parte de las OpenCV, y estas nuevas librerías que se utilizan hizo que no fuera posible que siguiéramos implementando código dentro de la aplicación, así que se tuvo que buscar una alternativa viable que nos permitiera seguir utilizando el código C++ que ya estaba creado para procesar las imágenes con texto. Esto se solucionó creando un WebService que ejecutara este código desde el servidor, y esta solución es sobre la que el artículo se ha basado para así explicar el

funcionamiento de la segunda parte del procesador del L-OCR.

Otro de los problemas que surgieron durante el transcurso del proyecto era el formato en el que la cámara de OpenCV obtenía las imágenes. OpenCV utiliza por defecto la definición de colores BGR para una imagen, en cambio, Android utiliza RGB, esto inicialmente era un problema, puesto que los colores que mostrabas en la aplicación no eran los colores reales. En la figura 18 se puede apreciar en la imagen de la izquierda el resultado que mostraría la aplicación y en la derecha el resultado que realmente debería mostrar.

Figura 18: Imagen Mostrada y real

Para solucionar esto, tras revisar las funciones propias de openCV, se encontró una que permitía hacer la conversión a RGB permitiendo así obtener la imagen con el color real.

También hubo problemas iniciales con el posicionamiento de la imagen al obtener la foto. El problema era el siguiente, según la posición en la que tomaras la imagen con la cámara de OpenCV, la imagen que retornaba podía estar girada 90º, 180º o 270º, y esto era un problema, ya que si la imagen esta rotada, el procesador del L-OCR no es capaz de detectar texto de una imagen. Para solucionar esto se optó por añadir un botón de rotación de imagen tal y como se aprecia en la figura 9 del selector de imagen, con este botón, el usuario de la aplicación puede rotar sin problema la imagen hasta colocarla en la posición correcta. También se valoró intentar que la propia aplicación corrigiera el posicionamiento de la imagen de forma automática a partir de los sensores del dispositivo, pero estos sensores tan solo devuelve 0 y 1 para indicar si el móvil está en posición horizontal o vertical, por lo que no podemos saber realmente la orientación exacta del dispositivo, ya que podemos poner el dispositivo horizontal o vertical de dos maneras, para que se vea más claro, si cogemos el dispositivo móvil de la forma estándar, es decir, con el altavoz que se utiliza para oír las llamadas hacia arriba, es un tipo de posición vertical y si rotamos el teléfono 180º, lo seguiríamos teniendo en vertical pero bocabajo. Por el motivo explicado anteriormente se acabó descartando esta posible solución.

Y finalmente, uno de los últimos problemas con los que el desarrollo del proyecto se encontró, fue que se inició el desarrollo de la aplicación con la versión de Android 5.1.0, y al actualizarse el dispositivo a la versión 5.1.1, la galería que podemos ver en la figura 11 que nos permite seleccionar una imagen almacenada en el dispositivo para que posteriormente sea procesada, dejo de funcionar. Finalmente, tras indagar, se descubrió que con la nueva versión, Android no devolvía los paths virtuales de la misma forma, por lo tanto no se podía obtener la imagen de la forma que estaba programada en el código, así que se optó por obtener la imagen directamente y almacenarla en el buffer, en vez de obtener su path virtual, para así posteriormente almacenarla en el dispositivo con un path ya conocido y poder pasarle este path al selector de imagen de la figura 9.

6 REQUERIMIENTOS DE LA APLICACIÓN

Para el correcto funcionamiento de la aplicación se necesitan las siguientes características:

- Dispositivo con Android 5.0 o superior.
- Conexión a internet desde el dispositivo.
- Darle los siguientes permisos a la aplicación:
  - Acceso a internet
  - Acceso a la cámara
  - Acceso a la memoria externa del dispositivo.
- También se necesita disponer de un servidor que permita la utilización de código PHP y la ejecución de .exe

La aplicación ha sido testeada en un dispositivo Nexus 5 con Android 5.1.1.

El servidor ha sido montado en local en un portátil Asus X550L, con un procesador I7 a 3,1Ghz y 8GB de RAM y una conexión de red de 100Mbps.

7 POSIBLES MEJORAS FUTURAS

Actualmente el tiempo de procesamiento del código que se ejecuta en el servidor es lento, tarda aproximadamente 50 segundos en procesar cada imagen que se envía, por lo que para aumentar esta velocidad de ejecución se podría optar por una mejor optimización del código, o utilizar un servidor con más capacidad de procesamiento. También se podría implementar un sistema de mutithreading dentro de la aplicación Android que realizara varias peticiones con cada una de las imágenes a procesar, de manera que se reduciría considerablemente el tiempo obtención del resultado final del procesamiento de la imagen.

El código del servidor también podría ser implementado dentro de la propia aplicación Android en un futuro, siempre y cuando el NDK sea lo suficientemente mejorado como para facilitar la incorporación de librerías externas que no tengan soporte nativo, esto haría que el tiempo de ejecución de esta parte del código mejorara, ya que durante el proyecto se ha podido comprobar que el código C++ ejecutado desde un terminal Android, en nuestro caso un Nexus 5, es más rápido que ejecutar el código desde un PC con S.O Windows 8.1 y mejor Hardware que el del terminal Android.

8 Conclusión

Una vez finalizado el desarrollo de la aplicación se puede concluir que se han logrado cumplir los objetivos de desarrollo de una interfaz que permite obtener y seleccionar una parte concreta de una imagen, también se ha logrado añadir las dos fases del L-OCR. En cuanto a los subobjetivos de la aplicación también han realizados con éxito, por lo que la aplicación permite copiar el texto de los resultados, y también incorpora un traductor a diversos idiomas del resultado de la aplicación. Así que claramente se han logrado cumplir con éxito todos los objetivos y subobjetivos planteados al inicio de la planificación del proyecto L-OCR.

Por la parte de integración del código proporcionado por el CVC se concluye que, el NDK de Android es una herramienta muy potente para la implementación de código C++ y C en aplicaciones Android, pero, solo se debe escoger esta vía si realmente es imprescindible que se deba incorporar el código en C++ o C dentro de la aplicación. Esto es así ya que la configuración del NDK es bastante compleja y engorrosa, y si además se quiere usar actualmente un IDE diferente a Eclipse, la documentación existente es bastante escueta, además, hasta el momento intentar usar el debbuger es una tarea casi imposible, al menos actualmente, probablemente, dentro de un tiempo el uso de NDK sea más sencillo, puesto que IDEs como AndroidStudio está incorporando soporte para esta herramienta, y además permitirá de una forma sencilla el uso de la opción de debbuger.

Como bien he puesto anteriormente se debería evitar en medida de lo posible el uso del NDK, y para hacer esto existen soluciones viables, como es la que se ha acabado incorporando en este proyecto en la segunda parte del procesador del L-OCR, la creación de un WebService que permita la ejecución del código C++ o C a través de una simple petición desde el dispositivo Android. La parte mala de esta solución es que se debe disponer de un servidor, y esto requiere de un mantenimiento por la parte del desarrollador, a no ser que se prefiera contratar un servidor a una empresa proveedora de servicios, y en esta caso la utilización del servidor evidentemente tendría un coste económico. Pero a pesar de esto, en mi opinión, para evitar frustraciones innecesarias, hasta que la incorporación del NDK en proyectos Android sea mejorada, creo que la mejor opción es utilizar un servidor externo.

A nivel de desarrollo del proyecto la conclusión es que por mucha planificación que un proyecto pueda tener, si el proyecto es diseñado teniendo diferentes carencias de conocimientos que se necesitan para su correcto desarrollo, puede pasar que, durante el transcurso del proyecto se deban tomar decisiones que impliquen cambios para así poder lograr el objetivo fijado inicialmente. Estas decisiones se deben tomar evidentemente teniendo en cuenta la viabilidad que suponen para el proyecto y el tiempo que se dispone para llevarlas a cabo posteriormente.

9 Referencias

[1] OpenCV, (7/03/2015) OpenCV para Android
URL: http://opencv.org/platforms/android.html

[2] Android, (7/03/2015) Android NDK
URL: https://developer.android.com/tools/sdk/ndk/index.html

[3] Android, (1/04/2015) Custom Drawing
URL: http://developer.android.com/training/custom-views/custom-drawing.html

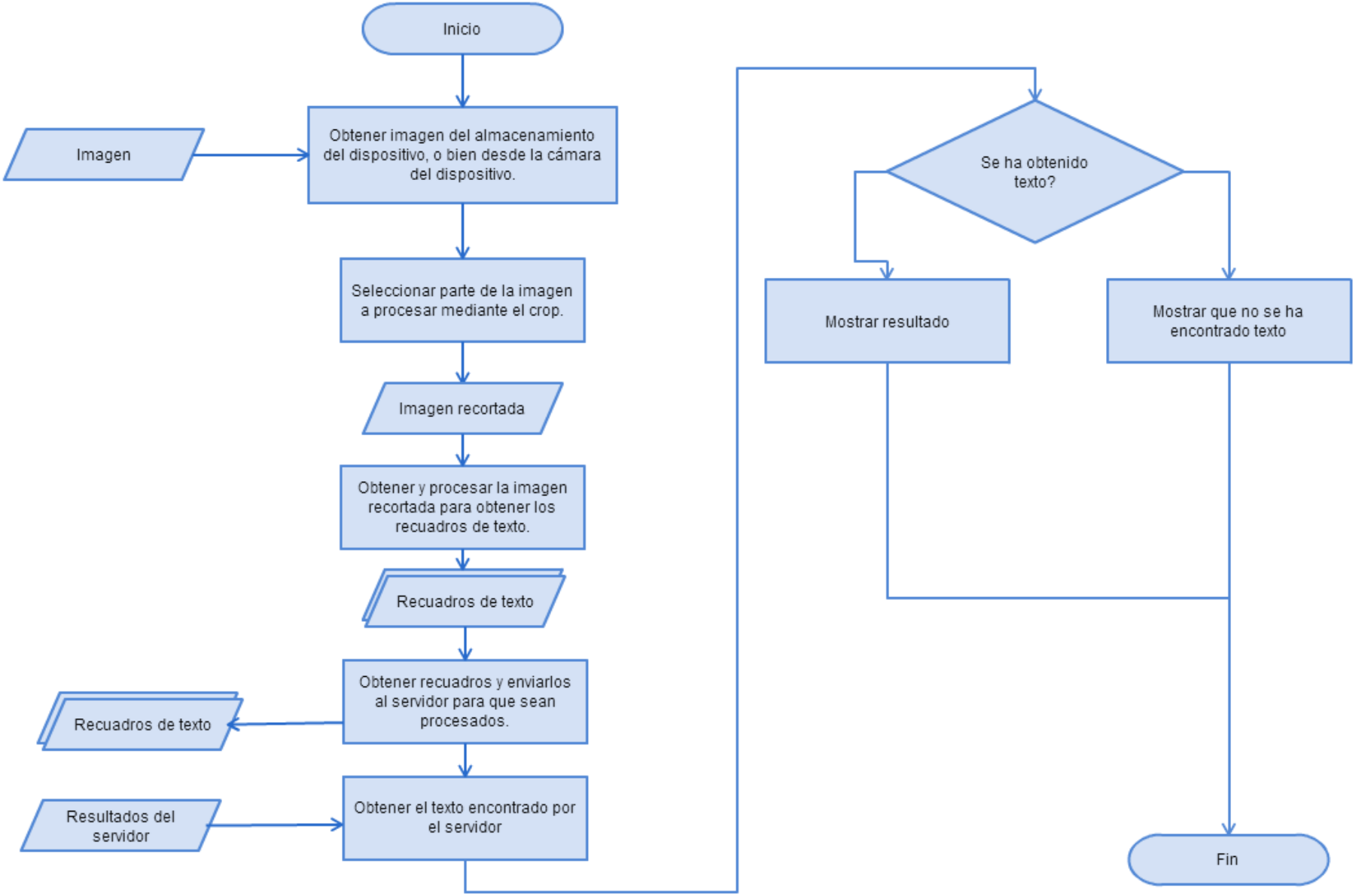
[4] Android, (6/04/2015) AsyncTaks
URL: http://developer.android.com/reference/android/os/AsyncTask.html

[5] A Java wrapper for the Microsoft Translator API (1/05/2015) Microsoft Translator Java API.
URL: https://code.google.com/p/microsoft-translator-java-api/

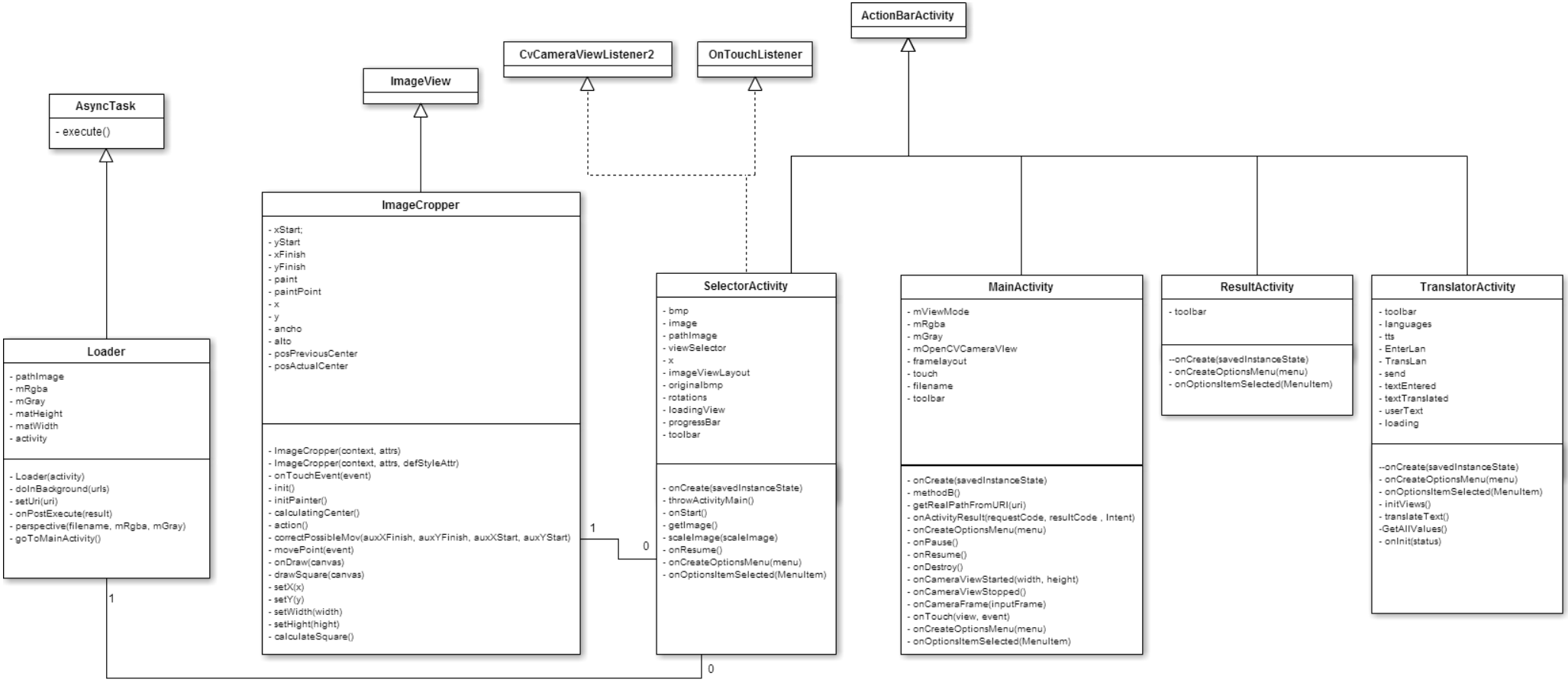
[6] PHP, (30/05/2015), Manual PHP
URL: http://php.net/manual/es/index.php

APENDICE

A1. DIAGRAMA DE FLUJO



A2. DIAGRAMA DE CLASES



A3. diagrama de casos de uso