sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Universidad Nacional del Litoral

Facultad de Ingeniería y Ciencias Hídricas

Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo

Autor: Sosa, Pablo Andrés
Director: Albornoz, Enrique Marcelo
Co-Director: Martínez, César Ernesto

4 de Septiembre de 2013

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

"Todos los triunfos nacen cuando nos atrevemos a comenzar..."

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Agradecimientos

A la hora de hacer el balance de las cosas que han pasado durante mi carrera universitaria quedan muchas cosas positivas, sobre todo, por la gente con la que he ido compartiendo momentos.

Quiero agradecer en primer lugar a mi *Familia*, por su paciencia a largo de estos años, y por permitirme lograr el objetivo de graduarme.

A mis compañeros de grupo en los trabajos a lo largo de la carrera, que el día de hoy ya son amigos, *Gaby G*, *Juampi*, *Fede* y *Nico U*.

Al *Enano*, con quien he compartido días de estudio, aunque nos hayamos abocado a carreras diferentes.

A aquellos compañeros de la facultad, que son amigos de peña, *Maxi*, *Julio*, *Guido*, *Osva*, *Gaby*, *Polen*, *Mati* y *Nico V*.

También, quiero agradecer enormemente a mis directores, *Marcelo* y *César*, por aceptar dirigirme y darme una mano durante todo el desarrollo del proyecto.

Mi más sincera gratitud,

Pablo A. Sosa.
Santa Fe, 4 de Septiempre de 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Resumen

La evolución tecnológica de los dispositivos móviles ha permitido que puedan ejecutarse algoritmos exigentes como aquellos utilizados en procesamiento de imágenes. Este contexto ha impulsado un nuevo campo de investigación y genera un nicho de mercado a nivel regional.

Existen aplicaciones comerciales que realizan procesamiento de imágenes y a partir de la identificación de objetos de interés retornan información de utilidad. Entre las más conocidas se pueden citar *Google Goggles*, *Wikitude* y *Layar*.

En este trabajo se presenta una aplicación para dispositivos móviles que incorpora técnicas de visión computacional. Se trabajó sobre el circuito turístico del "Camino de la Constitución" de la ciudad de Santa Fe. El sistema realiza el reconocimiento automático de imágenes de los diferentes mojones informativos y provee al usuario una vasta cantidad de información. Cabe destacar que este proyecto ha permitido realizar las siguientes publicaciones:

- *Aplicación turística para dispositivos móviles basada en técnicas de visión computacional*. Aceptado en el XIX Congreso Argentino de Ciencias de la Computación, CACIC 2013, Mar del Plata, del 21 al 25 de Octubre de 2013.
- *Aplicación turística para Android basada en técnicas de procesamiento de imágenes*. Presentado en el XVII Encuentro de Jóvenes Investigadores de la UNL, 4 y 5 de setiembre de 2013, Santa Fe.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Prefacio

Casi 5 millones de smartphones actualmente en nuestro país sumado a la gran cantidad de destinos turísticos que existen en Argentina hacen que un desarrollo para dispositivos móviles con aplicaciones en turismo sea mirado como un potencial caso de éxito. Por otra parte, el uso extendido de estos aparatos ha ocasionado no sólo la necesidad de crear nuevas aplicaciones, sino también que programas ya existentes para computadoras de escritorio (PC) debieran migrar y adaptarse a dichos dispositivos [2]. La velocidad de los procesadores de los primeros smartphones era de 200, 300 y 400 Mhz aproximadamente. En cambio, los teléfonos móviles actuales ya poseen microprocesadores de doble núcleo, provocando que se acerquen año a año al rendimiento necesario para que puedan ejecutarse los algoritmos más exigentes en Procesamiento de Imágenes (detección de objetos, seguimiento visual, etc.) [3].

En este trabajo se presenta el diseño y desarrollo de una aplicación para dispositivos móviles con sistema operativo Android [4] basada en técnicas de visión computacional. La aplicación pretende ser una herramienta útil para los visitantes de la ciudad de Santa Fe, que visiten el circuito turístico del "Camino de la Constitución". Este es un recorrido museológico que integra 18 sitios y edificios de valor simbólico y arquitectónico, con el objetivo de reconstruir y narrar la historia de la relación de la ciudad de Santa Fe con la Constitución Nacional.

Para llevar a cabo el proceso de reconocimiento de los mojones se definió y realizó un importante relevamiento fotográfico utilizando diversos dispositivos móviles y condiciones de iluminación naturales, para generar una base de datos de imágenes de los mojones informativos. Luego se exploraron técnicas

de procesamiento, análisis y clasificación de imágenes, orientadas a la clasificación de los mojones. En cada etapa se priorizó la obtención del mejor desempeño manteniendo la simplicidad y una carga computacional baja, a fin de lograr una aplicación veloz. Finalmente, se diseñó una interfaz gráfica que permite al usuario tomar una fotografía del mojón y obtener información turística relacionada.

Se decidió desarrollar la aplicación para Android, dado que es una plataforma que ofrece muchas ventajas frente a otros sistemas operativos. Es una plataforma abierta, lo que significa que no está sujeto a ningún fabricante de hardware y su código fuente se encuentra disponible en el sitio oficial de Android [1]. Ésto permite a los fabricantes de dispositivos móviles y desarrolladores, añadir aplicaciones a los dispositivos. Es el sistema operativo con mayor cuota de mercado a nivel mundial, impulsada por Google (una de las empresas más grandes e innovadoras en cuanto a Tecnología de la información) que junto a muchas otras empresas mundialmente reconocidas, apoyan su desarrollo [5]. Por otra parte, para la realización del procesamiento de imágenes se decidió utilizar la biblioteca OpenCV dado que es completamente Open Source, multiplataforma, tiene 15 años de desarrollo y por lo tanto tiene una base sólida, es estable y provee una gran cantidad de funcionalidades, suficientes para el presente proyecto. Además, provee una batería de funciones para Procesamiento y Análisis de imágenes, permite trabajar con video y tiene una versión disponible para Android [6].

El proyecto se encuentra organizado en cinco capítulos, como se explica a continuación.

En el Capítulo 1, se expone la motivación que da lugar a este proyecto, seguido por una explicación detallada del estado del arte del procesamiento de imágenes en dispositivos móviles, indicando dónde se encuentra el mismo en la actualidad. Por último, se exponen los objetivos generales y específicos del trabajo.

En el Capítulo 2, se explica el marco teórico de cada uno de los métodos que se utilizó en las etapas de normalización de las imágenes, extracción de características, clasificación, así como también el contexto que permite desarrollar la interfaz de usuario en Android.

En el Capítulo 3, se describe el diseño del sistema y se explican detalladamente las etapas del método de resolución propuesto. Por un lado, aquellas correspondientes al procesamiento, análisis y clasificación de imágenes y, por otro, las que corresponden al desarrollo de la interfaz de usuario.

En el Capítulo 4, se presentan algunos experimentos y resultados orien-

[1] http://source.android.com

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

tados a comprobar el desempeño del sistema final. Se consideran diferentes condiciones de iluminación, diferentes dispositivos y resoluciones.

En el Capítulo 5, se exponen las conclusiones surgidas de este proyecto y se proponen algunos trabajos futuros para continuar evolucionando el sistema.

Sosa, Pablo Andrés.
Santa Fe, Argentina.
4 de Septiembre de 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Índice general

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Índice de figuras

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

CAPÍTULO 1

Introducción

En este capítulo se presenta la motivación del trabajo y a continuación, se realiza una revisión del estado del arte del procesamiento de imágenes en dispositivos móviles. Luego, se exponen los objetivos generales y específicos de este proyecto final, y se detallan los alcances del mismo.

1.1 Motivación

La carrera universitaria que he realizado abarca diferentes áreas de estudio y en cada una existe la posibilidad de realizar el proyecto final de carrera. Sin embargo, el auge del desarrollo de software para dispositivos móviles junto con los avances del procesamiento de imágenes en éstos, fundamentalmente en aplicaciones de realidad aumentada, han influido en gran medida en la decisión de desarrollar una aplicación para dispositivos móviles utilizando visión computacional. La evolución de los teléfonos inteligentes y su uso masivo hicieron que las empresas de desarrollo de software adapten la forma en que realizan sus productos u ofrecen sus servicios, para que puedan correr sobre dichos dispositivos. El área que comprende la visión computacional no escapa a esta ola tecnológica, por lo que bibliotecas como las que permiten el procesamiento de imágenes o el reconocimiento óptico de caracteres, entre otras, han comenzado a estar disponibles para tablets o smartphones [7].

Si bien este último hecho es bastante reciente, y las primeras versiones aún sufren de algunos errores, empresas importantes a nivel internacional ya han lanzado sus primeras versiones de aplicaciones que incluyen procesamiento de imágenes en dispositivos móviles, *Goggles*, *Wikitude GmbH* y *Layar* son algunas de ellas.

En el contexto local existen diferentes iniciativas orientadas a fomentar el turismo en la ciudad de Santa Fe, como lo son la recuperación y restauración de inmuebles históricos y culturales, entre los que podemos destacar la recuperación del complejo "La redonda", "El molino Marconetti" y "La estación Belgrano". También se han generado espacios para eventos importantes como el TC 2000, el "Música en el río", y se ha impulsado un plan estratégico proyectado para el 2020, que incluye la restauración del puerto de la ciudad con la creación de edificios inteligentes y la instalación del hotel Hilton Garden Inn. Todo ésto motiva en gran medida el desarrollo de software con el objetivo de ofrecer servicios a la gran cantidad de turistas que se espera recibir en los próximos años.

En vista de los argumentos mencionados anteriormente, en este trabajo se presenta el diseño y desarrollo de una aplicación para dispositivos móviles con sistema operativo Android [4] orientada al turismo basada en técnicas de visión computacional. La aplicación pretende ser una herramienta útil para los visitantes de la ciudad de Santa Fe, que recorran el circuito turístico del "Camino de la Constitución". Este es un recorrido museológico que integra 18 sitios y edificios de valor simbólico y arquitectónico, con el objetivo de reconstruir y narrar la historia de la relación de la ciudad de Santa Fe con la Constitución Nacional. La propuesta integra diversas funciones: histórica, cultural, educativa y turística, y pone de relieve nuestra vida política pasada y presente relacionada con los diferentes procesos y acontecimientos vinculados con la Carta Magna[1].

Implementar este tipo de aplicaciones en un contexto local podría ser algo muy provechoso tanto para la Universidad, como para la Subsecretaría de Turismo de la ciudad de Santa Fe. Además, realizar la aplicación para el sistema operativo Android permite abarcar un espectro muy grande de usuarios de smartphones [8], ya que el mismo ha alcanzado el 75 % de la cuota de mercado a nivel mundial el presente año y lidera ampliamente el mercado en Latinoamérica. Por otra parte, se propone desarrollar una aplicación que solo requiera una cámara de fotos y se pretende avanzar en el reconocimiento automático de imágenes sin marcadores, es decir, sin alterar el ambiente. Para finalizar, se puede decir que un proyecto de esta naturaleza trae aparejada

[1] Más información en http://santafeciudad.gov.ar/cunadelaconstitucion/

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

la realización de una biblioteca de módulos básicos que podrían ser fácilmente reutilizables en trabajos futuros del Centro de Investigación en Señales, Sistemas e INteligencia Computacional (SINC(i))[2].

1.2 Estado del arte

Existen aplicaciones comerciales que realizan reconocimiento de objetos de interés en imágenes y retornan información turística útil a partir de la misma, como por ejemplo Wikitude, Layar y Google Goggles [10]. Éstas dos últimas, son aplicaciones muy similares entre sí. El funcionamiento es muy sencillo: se elige el tipo de información que se quiere visualizar y con la cámara se direcciona e identifica, con la ayuda del GPS y la brújula, a los lugares o sitios de interés que están cerca. A continuación se describe detalladamente como funciona cada una de ellas.

Wikitude es un software de realidad aumentada (RA)[3] para dispositivos móviles que fue desarrollado por la compañía austríaca Wikitude GmbH y publicado en octubre de 2008 como software gratuito. Para estimar la posición de los objetos en la pantalla del dispositivo móvil se utiliza la posición del usuario obtenida con el GPS, mientras que la dirección en la que mira el usuario es obtenida mediante el uso de la brújula y acelerómetro. Esta aplicación depende de la precisión del GPS, la cual no siempre es la requerida [11].

Layar es un software que utiliza el GPS y la brújula de los dispositivos Android para ubicar la posición del usuario y su orientación. El usuario del dispositivo realiza una captura del entorno que se reproduce en la pantalla y al mismo tiempo, el software superpone información relacionada. Este software dispone de varias capas de datos que muestran diferentes contenidos al usuario. La capa por defecto se llama Layar Local Search y obtiene información de Google para indicarnos lugares tales como cafeterías, restaurantes, cines, etc. La ubicación de los lugares se puede ver en una imagen en tiempo real o sobre un mapa [12].

Google Goggles es un servicio de Google disponible para Android e iPhone que permite reconocer objetos mediante fotos capturadas con un dispositivo

[2]http:// fich.unl.edu.ar/sinc/

[3]RA es el término que se usa para definir una visión directa o indirecta del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

móvil y devolver resultados de búsqueda e información relacionada. Este sistema reconoce: lugares, monumentos y textos, entre otras. Su funcionamiento consiste en apuntar con la cámara del dispositivo móvil a un lugar conocido, un código de barras o QR, o un producto. Si Google lo encuentra en su base de datos, ofrecerá información útil [13].

Una vez que se estudiaron las tres aplicaciones más populares para el reconocimiento de objetos mediante la utilización de dispositivos móviles, se decidió que la aplicación utilice procesamiento de imágenes para el reconocimiento de mojones. Uno de los condicionantes es que el sistema de GPS puede tener una localización imprecisa, de unas decenas de metros, lo que dificulta la distinción entre mojones próximos. Además, se pretende que la aplicación pueda ser utilizada en dispositivos móviles sencillos (sin GPS) cuyo requisito sea solamente el de tener la cámara de fotos. Otra de las condiciones que se establece es que los mojones deben permanecer sin alteraciones tal como se diseñaron, por lo tanto no se incorporarán marcas o códigos extras (como ser códigos QR) para identificarlos.

Entre las tecnologías que se utilizarán en el presente trabajo es necesario destacar al sistema operativo de los dispositivos móviles, los entornos de desarrollo integrado (IDE)y las bibliotecas de procesamiento de imágenes.

1.2.1 Sistema operativo móvil

Se decidió desarrollar la aplicación para Android. Se tomó esta determinación, dado que es una plataforma que ofrece muchas ventajas frente a otros sistemas operativos:

- El sistema operativo Android es una plataforma abierta, lo que significa que no está sujeto a ningún fabricante de hardware. El código fuente de Android se encuentra disponible en la web [4], esto permite a los fabricantes de dispositivos móviles y desarrolladores, añadir aplicaciones a los dispositivos.

- Permite que los desarrolladores publiquen sus aplicaciones en el Android Market o Google Play de forma gratuita.
- Por último, Google es una de las empresas más grandes e innovadoras en cuanto a Tecnologías de la Información y la Comunicación, y está detrás

[4] http://source.android.com

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Tabla 1.1: Comparación de entornos de desarrollo integrados.

Características	**Eclipse**	**Netbeans**
Velocidad	Lento	Rápido
Visor de interfaz	Posee	No Posee
Detección de errores XML	Posee	No Posee
Conectividad de los dispositivos	Buena	Buena

de Android junto a muchas otras empresas con marcas mundialmente reconocidas, apoyando su desarrollo.

1.2.2 Entornos de desarrollo integrado (IDE)

Se realizó una comparación entre Eclipse [5] [14] y Netbeans [6] [15], dos de los editores más populares para desarrollar aplicaciones Android. Para este proyecto se ha seleccionado el primero, dado que en la página oficial de Android, en la sección de instalación del conjunto de herramientas de desarrollo de software (SDK), se puede encontrar soporte para este IDE.

Respecto a los Emuladores, el de Netbeans parece ser más rápido que el de Eclipse, incluso parece consumir menos recursos. Eclipse cuenta con más herramientas y facilidades a la hora de realizar una aplicación, por ejemplo, para desarrollar interfaces gráficas usando XML se cuenta con un "visor de interfaz" donde se puede ver la vista previa de la aplicación, sin tener que iniciar el teléfono o emulador. Otra característica importante es que Eclipse detecta errores de código en los archivos XML, mientras que el plugin de Netbeans no lo hace. Respecto a la conectividad con los dispositivos, no se detectaron problemas con los IDEs. Los dos dispositivos (Smartphone Samsung y Tablet Asus) son visibles en ambos IDEs cuando se conectan por medio del cable USB.

1.2.3 Biblioteca de procesamiento de imágenes

Para la realización del procesamiento de imágenes se decidió utilizar la biblioteca OpenCV dado que es completamente Open Source, multiplataforma, tiene 15 años de desarrollo y por lo tanto tiene una base sólida, es

[5] http://www.eclipse.org/
[6] http://netbeans.org/

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

estable y provee una gran cantidad de funcionalidades, suficientes para el presente proyecto. Además, provee una batería de funciones para Procesamiento y Análisis de imágenes, permite trabajar con video y tiene una versión disponible para Android [6].

1.3 Objetivos del proyecto final

Una vez introducida la tarea de interés y el marco general de trabajo, se enuncian a continuación los objetivos generales y particulares del presente Proyecto Final.

Los objetivos generales de este trabajo son los siguientes:

- Desarrollar una aplicación para dispositivos móviles que brinde información turística inmediata al usuario, utilizando procesamiento y análisis de imágenes.

- Aplicar los conocimientos adquiridos en la carrera y realizar un aporte ingenieril a la comunidad en general.

Los objetivos específicos son los siguientes:

- Investigar y determinar las tecnologías a utilizar para el desarrollo de la aplicación en base al rendimiento de las mismas.

- Explorar distintas estrategias para la captura, el procesamiento, análisis y clasificación de las imágenes.

- Definir la arquitectura óptima para el sistema en base a análisis de desempeño y tiempo de cómputo.

- Diseñar e implementar un módulo para la presentación de la información turística.

- Desarrollar un prototipo de la aplicación y evaluar su comportamiento.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

1.4 Alcances del proyecto final

En el presente trabajo se pretende diseñar y desarrollar un prototipo de una aplicación para dispositivos móviles con sistema operativo Android. Debe realizar la captura, el procesamiento, análisis y clasificación de imágenes y se pretende que la misma retorne información turística útil al usuario. La aplicación se limitará a reconocer los mojones instalados por la Subsecretaría de Turismo de la ciudad de Santa Fe en el Camino de la Constitución y a partir de ésto deberá retornar información sobre el monumento o lugar referido. La aplicación debe funcionar para cualquier dispositivo con sistema Android 2.3 o posterior.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

CAPÍTULO 2

Marco teórico

En este capítulo se presenta el sistema operativo Android, la biblioteca OpenCV y los fundamentos teóricos de las técnicas de procesamiento de imágenes utilizadas.

2.1 Programación en dispositivos móviles

2.1.1 Plataforma Android

Es un sistema operativo para dispositivos móviles que fue desarrollado inicialmente por Android Inc., una empresa comprada por Google en 2005. Actualmente, ya se han superado las 700.000 aplicaciones. El lanzamiento se realizó el 5 de noviembre de 2007 y Google liberó la mayoría del código de Android bajo la licencia Apache, una licencia libre y de código abierto [16]. A continuación se mencionan algunas de las características mas destacables de la plataforma de desarrollo Android:

- Java es el lenguaje de programación en el que están escritas todas las aplicaciones Android.
- El sistema operativo no hace diferencia entre las aplicaciones instaladas

por defecto en el dispositivo y las implementadas por otros desarrolladores a la hora de establecer prioridades entre las aplicaciones. Esto hace posible que se puedan crear aplicaciones que tengan tanta importancia para el sistema operativo del teléfono como las que vienen de fábrica [17].

- Provee un framework que permite la reutilización de código fuente y gran cantidad de servicios que ayudan a desarrollar aplicaciones.

- Tiene una Máquina virtual llamada Dalvik, optimizada para dispositivos móviles, que permite gestionar las aplicaciones y la memoria de manera muy eficiente.

Máquina virtual Dalvik

Dalvik es una máquina virtual diseñada para Android. Desde sus inicios, Dalvik fue pensada para los dispositivos móviles con el fin de optimizar la duración de la batería y la potencia de procesamiento. Cada aplicación Android se ejecuta en un proceso propio y genera una instancia de la máquina virtual Dalvik, que fue diseñada para poder ejecutar múltiples máquinas virtuales en un mismo dispositivo.

Las aplicaciones Android tienen su código fuente escrito en Java pero Dalvik, a diferencia de Java VM (Virtual Machine), no trabaja sobre el bytecode de Java, sino que lo transforma en un código más eficiente que el original pensado para procesadores con poca capacidad de procesamiento. En la Figura 2.1 se observa como las dos máquinas virtuales toman el código fuente y lo compilan para obtener el bytecode de Java. En el caso de la JavaVM será ejecutado mientras que en el caso de la DalvikVM, será transformado al formato .dex con una herramienta incluida en el sistema dx(DX, Dalvik Executable) para conseguir el bytecode de Dalvik.

Componentes básicos de una aplicación Android

Hay cuatro tipos de componentes básicos que puede contener una aplicación. Estos son Activities, Services, Content providers y Broadcast receivers [1] [18]. Cada uno es una pieza única que ayuda, junto al resto de los bloques, a definir el comportamiento global de la aplicación. A continuación se describe cada uno de ellos.

[1] http://source.android.com/

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

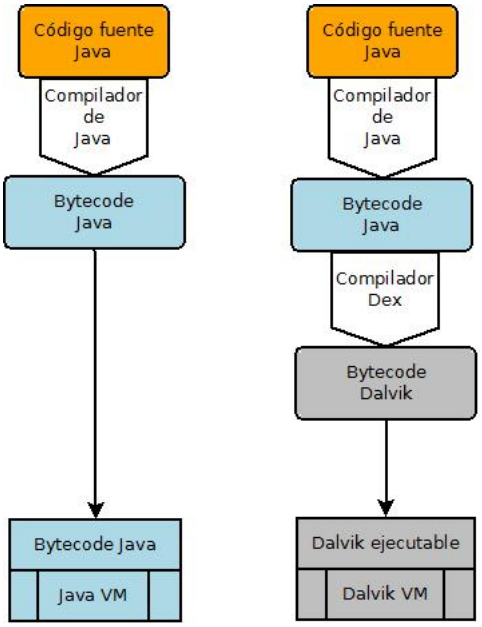

Figura 2.1: Compilación con JavaVM y DalvikVM.

- Activity: es una interfaz en el dispositivo con la que el usuario puede interactuar. Una aplicación normalmente está compuesta de muchas actividades que están en cierta medida relacionadas entre sí, donde el usuario puede ir pasando de una a otra. Una de las actividades de la aplicación se denomina "main activity" o "actividad principal", y es la que se ejecuta cuando se inicia la aplicación. El cambio de una pantalla a otra significa el comienzo de una nueva actividad. Para implementar una nueva actividad se debe crear una nueva clase que heredará los atributos y funcionalidades de la clase Activity, definiendo su propia interfaz de usuario (mediante una jerarquía de vistas) y definiendo la nueva funcionalidad de la actividad. Un ejemplo es una aplicación de correo electrónico que tenga una actividad que muestre en la pantalla la lista con los nuevos emails, otra actividad con la pantalla para poder escribir un nuevo correo y otra más para leer los mensajes.

- Service: es un componente que se ejecuta en el background para realizar procesos que deben continuar aun cuando las actividades de la aplicación no estén activas. Un componente de la aplicación puede inicializar un service y éste continuará ejecutándose en background incluso cuando el usuario cambie a otra aplicación. Un ejemplo es una aplicación para reproducir música o escuchar la radio, que sigue funcionando mientras el usuario se mueve de una aplicación a otra.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

- Content Provider: permite compartir datos entre aplicaciones. Por defecto, cada aplicación Android se ejecuta de manera independiente, almacenando sus datos en una base de datos o cualquier otra forma de almacenamiento. Cuando surge la necesidad de intercambiar información entre aplicaciones se realiza a través de los content providers que, dependiendo de los permisos que tenga asignada la aplicación, permitirán el acceso a datos de otra aplicación con permisos de lectura y/o escritura. Un ejemplo de proveedor de contenidos es el que proporciona Android para acceder a la información multimedia del usuario, como imágenes, videos y música. Cualquier aplicación que tenga los permisos adecuados puede, por tanto, acceder a la información.

- Broadcast Receiver: un broadcast en informática es básicamente el envío de información desde un nodo emisor hacia muchos nodos receptores en forma simultánea. En Android los broadcast pueden ser lanzados por el sistema operativo o por alguna aplicación y el componente Broadcast Receiver es aquel que detecta y reacciona a los broadcast. Ésto se puede informar al usuario a través de la barra de notificaciones en la parte superior del dispositivo. Ejemplos de un broadcast son las notificaciones de actualizaciones, activación del modo de red 3G, descarga de datos en el dispositivo, etc.

Ciclo de vida de aplicaciones y actividades en Android

En Android, el tiempo de vida de las aplicaciones está generalmente controlado por el sistema operativo y no por el usuario como es usual. Cada aplicación se ejecuta en un proceso y si el sistema operativo lo cree necesario lo elimina, por ejemplo: para liberar memoria [19].

A su vez, cada uno de los componentes básicos de las aplicaciones Android tiene su propio ciclo de vida. Sin embargo, la parte más importante de la aplicación que permite un correcto funcionamiento de la misma es aquella compuesta por actividades.

Las actividades se gestionan mediante métodos implementados en la clase Activity y utilizarlos de manera correcta hará que la aplicación sea estable. Cuando se inicia una actividad se coloca en el tope de una pila que contiene todas las actividades de la aplicación, cabe destacar que la actividad anterior se detiene, pero el sistema la mantiene almacenada en la pila de actividades. Esta estructura de datos hace posible que el usuario pueda regresar a la pantalla previa, para ésto se destruye la actividad actual y se reanuda la anterior [20]. El estado de cada actividad se puede determinar mediante la

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

posición de ésta en la pila que contiene todas las actividades que actualmente se están ejecutando. Una actividad puede estar en los siguientes estados:

- Activa (Resumed): se produce cuando la actividad se encuentra en la pantalla, está visible y el usuario puede interactuar con ella. En este estado la activity está en el tope de la pila de actividades en ejecución.

- En pausa (Paused): se produce cuando la actividad se ve en la pantalla pero no puede interactuar con el usuario. Otra actividad es la principal y se encuentra por encima de esta en la pila de actividades.

- Parada (Stopped): se produce cuando la actividad no se ve en el dispositivo.

Tanto el ciclo de vida de una actividad, como los estados por los que esta puede pasar se pueden observar en la Figura 2.2.

Métodos del ciclo de vida de una actividad

- onCreate(): se llama cuando se crea la actividad por primera vez. El próximo método es onStart().

- onRestart(): se llama después de que se ha parado la actividad y antes de volver a comenzarla. El próximo método es onStart().

- onStart(): se llama cuando se quiere que la actividad sea visible para el usuario. El próximo método es onResume() u onStop().

- onResume(): se llama antes de que la actividad comience a interactuar con el usuario. El próximo método es onPause().

- onPause(): se llama cuando el sistema quiere ejecutar otra actividad como principal. El próximo método es onResume() u onStop().

- onStop(): se llama a este método cuando la actividad deja de estar visible para el usuario. El próximo método es onRestart() u onDestroy().

- onDestroy(): se llama cuando se desea destruir la actividad. Es la última llamada que recibe la actividad. No tiene próximo método.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

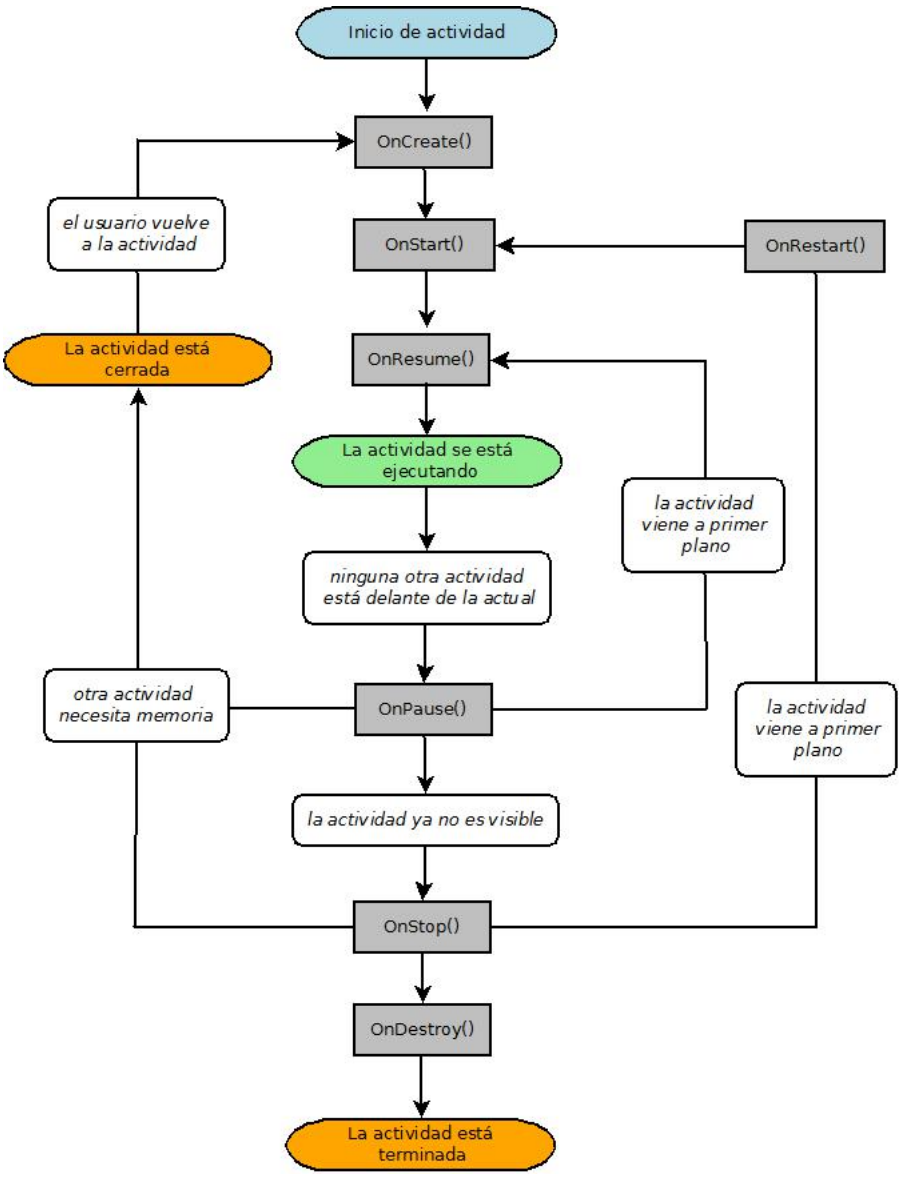

Figura 2.2: Ciclo de vida de una actividad en Android.

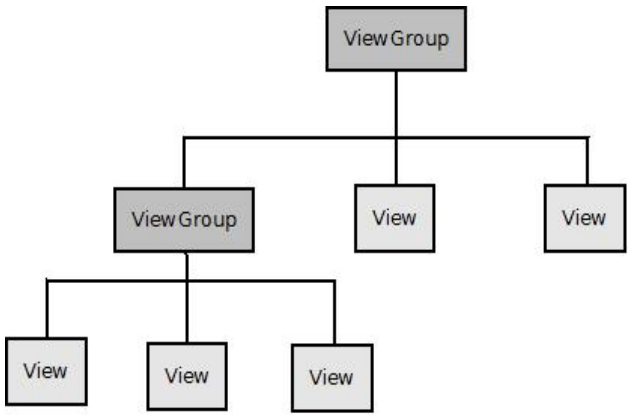

Figura 2.3: Jerarquía de views en Android.

Interfaz de usuario

Android soporta XML para el diseño de pantallas, que permiten implementar la interfaz de usuario de la aplicación. Para llevar a cabo dicha implementación Android nos proporciona una clase llamada View. Al instanciar esta clase podemos utilizar todos los métodos de un objeto de tipo View. Android utiliza esta clase como base para definir subclases y así ofrece objetos implementados para el control visual, estos se llaman widgets y algunos ejemplos son los Text, InputMethod, Button, RadioButton, CheckBox, etc [16].

View: un layout es un elemento no visible que determina la organización de una pantalla y un objeto View permite establecer el contenido de una zona rectangular de dicho layout. Un objeto View es el elemento más pequeño de la interfaz de usuario y al ser un elemento de dicha interfaz, también es un punto de interacción para los usuarios.

ViewGroup: es un objeto particular, que contiene a otros objetos de tipo View. En los objetos ViewGroup se definen las normas sobre cómo se muestran los objetos View hijos. La jerarquía que define Android para estos tipos de objetos se puede observar en la Figura 2.3, donde claramente se puede ver que los objetos ViewGroup son padres de los objetos View.

Layout: es una instancia de un objeto ViewGroup. Como se mencionó brevemente más arriba, la interfaz de usuario de una aplicación Android se define mediante archivos XML. Estos archivos contienen objetos de tipo View y

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

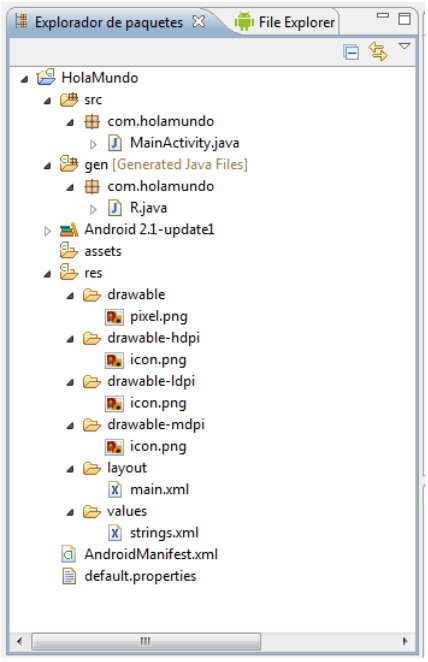

Figura 2.4: Estructura de archivos de un proyecto Android

ViewGroup que establecen la jerarquía entre los mismos. Así como Android proporciona objetos de tipo View predefinidos llamados widgets, también lo hace con los objetos de tipo ViewGroup que permiten definir la posición y características de los objetos View que se encuentran dentro. Algunos ejemplos de estos son LinearLayout, RelativeLayout, TableLayout y GridLayout. El archivo XML principal o raíz dentro de la jerarquía de View de la Figura 2.4 se llama main.xml, y a partir de este se generaran el resto de los layouts [18].

Widget: es un objeto view que actúa como interfaz para interactuar con el usuario. La plataforma Android proporciona un amplio conjunto de widgets ya implementados, como es el caso de botones, marcadores o campos de texto, con los que se puede construir rápida y fácilmente una interfaz de usuario. Algunos de los widgets que facilita el sistema son algo más complejos, como es el caso del reloj o del selector de fechas.

Archivo AndroidManifest.xml: es un archivo que esta presente en cada proyecto Android, y en este se define la estructura de la aplicación. Los elementos más importantes que contiene este archivo son:

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

- uses-permission: con esta sentencia se declaran los permisos que requiere la aplicación para su funcionamiento, por ejemplo: GPS e internet entre otros.

- permission: Este elemento declara los permisos que las actividades o servicios necesitan con el fin de mantener el buen uso de los datos de la aplicación o mantener la lógica de la misma.

- application: Este elemento contiene el núcleo de la aplicación (nombre, actividad principal, icono, etc.).

- Además se declara el nivel mínimo del API de Android que se necesita para ejecutar la aplicación en el dispositivo.

Estructura de archivos de un proyecto Android

Una aplicación Android, se puede ver desde el IDE como un conjunto de archivos y directorios, tal como lo muestra la Figura 2.4.

El directorio raíz contiene el nombre del proyecto y el archivo más importante dentro de este es el AndroidManifest.xml. También hay cuatro directorios relevantes para el proyecto: src, gen, Android Library y res. A continuación se explican los directorios más importantes de una aplicación Android.

- src: contiene todos los archivos con código Java.

- gen: contiene un archivo que se genera automáticamente y se llama R.java. Este contiene punteros a los recursos alojados en la carpeta res y es la unión entre los archivos Java y los recursos de la aplicación Android.

- Android Library: este directorio guarda archivos que la aplicación necesita para su funcionamiento. En el caso del presente proyecto contiene archivos que hacen referencia al SDK de Android y a la biblioteca OpenCV.

- res: contiene los recursos de la aplicación, todo lo que no es código Java y que pertenezca a la aplicación. Esta carpeta contiene imágenes, y archivos XML como main.xml y otros.

Cuando se compila un proyecto Android se genera un archivo con extensión .apk que contiene tanto el código fuente como todos los recursos de la aplicación.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

2.1.2 Biblioteca OpenCV para Android

OpenCV (Open Source Computer Vision Library) es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Se distribuye bajo licencia de software libre BSD (Berkeley Software Distribution), que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas [6]. Incluye una gran cantidad de algoritmos de procesamiento de imágenes. Además, tiene interfaces para C++, C, Python, Java, y es compatible con sistemas operativos como Windows, Linux, Android y Mac OS, lo que la hace totalmente multiplataforma, otro punto clave a tener en cuenta para utilizarla en este proyecto.

OpenCV proporciona un entorno de desarrollo fácil de utilizar y contiene cientos de algoritmos que abarcan diversas áreas del procesamiento y análisis de imágenes. En el primer grupo se encuentran la transformación entre espacios de colores y filtrado, entre otras. En el segundo, el reconocimiento de rostros, segmentación, análisis del movimiento, seguimiento de objetos en movimiento, etc. OpenCV está compuesta por diferentes módulos, dentro de los cuales se destacan los siguientes:

- core: en este módulo se encuentran las estructuras básicas de OpenCV como Mat (estructura para representar matrices) y funciones básicas que utilizan el resto de los módulos.
- imgproc: este módulo se encarga del procesamiento de imágenes mediante operaciones como filtrado, transformaciones geométricas, conversiones de color, histogramas, etc.
- highgui: este módulo se encarga de la gestión de entrada y salida y lo referente a la interfaz de usuario. A su vez, provee codecs para imagen y vídeo.
- calib3d: este módulo posee múltiples algoritmos de visión, geometría, calibración de cámara, estimación de posición de objetos, algoritmos de correspondencia estéreo y reconstrucción 3D.
- video: módulo para el análisis de vídeos que incluye estimación de movimiento, sustracción de fondo y seguimiento de objetos.
- objdetect: módulo de detección de objetos e instancias de los tipos predefinidos como por ejemplo caras, ojos, tazas, personas, autos, entre otros.

- ml: módulo de aprendizaje automático.
- gpu: módulo que implementa los algoritmos mencionados anteriormente utilizando GPU.

2.2 Procesamiento de imágenes

En esta sección se explican los fundamentos teóricos de las técnicas de procesamiento de imágenes que se utilizaron durante el proyecto.

2.2.1 Representación de imágenes

La resolución de una cámara fotográfica digital está limitada por el sensor que responde a las señales de luz. El sensor se compone de millones de "cubos" que se cargan en respuesta a la luz. Generalmente, estos cubos responden solamente a una gama limitada de longitudes de onda, correspondiente al color Rojo, Verde o Azul. Cada uno de estos cubos se llama píxel, y se utiliza un algoritmo de interpolación para unir la imagen de cada gama de longitud de onda por píxel en una imagen RGB, con el objetivo de representar un color completo. Las cámaras digitales realizan las capturas en RGB, y en ocasiones es necesario trabajar con imágenes que contienen un solo canal. Para esto, se deben realizar ciertas operaciones que permitan dicho tratamiento, entre estas se encuentran la conversión a escala de grises y la umbralización.

Conversión a escala de grises

Consiste en obtener un canal con las intensidades de grises a partir de una imagen color. Una buena forma de lograrlo es utilizando la fórmula perceptualmente ponderada definida en la ecuación 2.1.

$$I(x,y) = 0{,}299R(x,y) + 0{,}587G(x,y) + 0{,}114B(x,y) \tag{2.1}$$

donde I es la intensidad resultante en el píxel (x,y), y R, G, B, son las componentes en color de la imagen.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

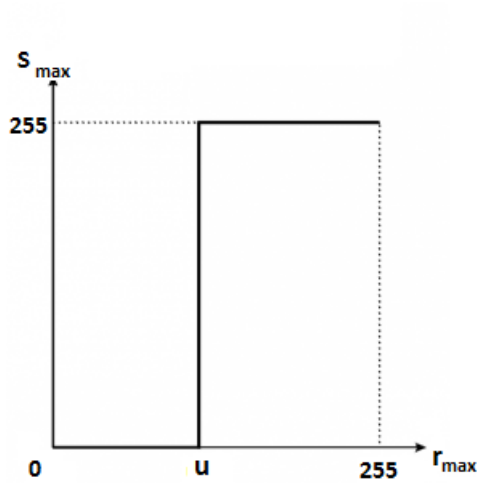

Figura 2.5: Umbral binario en imágenes.

Umbralización

Podemos definir una función de umbralización como una transformación con la forma de la expresión (2.2) donde r_{max} y s_{max} son los valores máximos de intensidad de la imagen de entrada y de salida respectivamente y u una constante que define el valor de umbral de la transformación. La función de umbralización (Figura 2.5) produce una imagen binaria asignando el valor s_{max} a los valores que superan u y 0 a los demás.

$$s = \begin{cases} 0 \text{ para } \ r < u \\ s_{max} \text{ para } \ r >= u \end{cases} \tag{2.2}$$

con $0 \leq r \leq r_{max}$ y $0 \leq u \leq r_{max}$.

2.2.2 Ecualización de histograma

Un histograma es una representación gráfica de una variable en forma de barras, donde la superficie de cada barra es proporcional a la frecuencia de los valores representados. En procesamiento de imágenes sirve para obtener un panorama de la distribución de los niveles de grises de una imagen.

La ecualización del histograma es una transformación que pretende obtener para una imagen un histograma con una distribución uniforme. Como resultado de esta transformación, se obtiene una expansión en el rango dinámico de la imagen [22]. Si se tienen imágenes digitales (valores discretos) con intensidades entre 0 y 1, y siendo:

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

- n la cantidad de píxeles de la imagen,
- p el número de bits de resolución de la imagen,
- L el máximo nivel de gris posible, dado por el valor de p,
- k un nivel de gris de la imagen,
- r_k un nivel de gris de la imagen normalizado a 1, con valores $r_k = 0, \frac{1}{2^p-1}, \frac{2}{2^p-1}, ..., 1$.
- $p_r(r_k)$ la probabilidad de un nivel de gris en la imagen calculado como:

$$p_r(r_k) = \frac{n_k}{n}, \text{ con } \begin{cases} 0 \leq r_k < 1 \\ k = 0, 1, ..., L-1 \end{cases} \tag{2.3}$$

la función de transformación utilizada es la Función de Distribución Acumulada:

$$s_k(r_k) = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} = \sum_{j=0}^{k} p_r(r_j), \text{ con } \begin{cases} 0 \leq r_k < 1 \\ k = 0, 1, ..., L-1 \end{cases} \tag{2.4}$$

donde s_k es el valor de la transformación sobre los k niveles de la imagen.

2.2.3 Operaciones morfológicas

Las dos operaciones básicas de la morfología matemática son erosión y dilatación y para explicarlas es necesario definir algunos conceptos de la teoría de conjuntos. Se hace referencia a A como un conjunto en $\mathbb{Z}^2$, $a = (a1, a2)$ un elemento de A, $\varnothing$ el conjunto vacío, $A \subseteq B$ para denotar que A es un subconjunto de B, $A \cap B$ para indicar la intersección entre A y B, $\hat{B}\{w|w = -b, \text{ para } b \in B\}$ para indicar la reflexión del conjunto B, $(A)_z = \{c|c = a + z, \text{ para } a \in A\}$ para indicar la traslación del conjunto A por $z = (z1, z2)$ y $\{.\}$ la notación de un conjunto [23].

Dilatación

Sean A y B conjuntos en $\mathbb{Z}^2$, la dilatación de A por B denotada como $A \oplus B$ queda definida como

$$A \oplus B = \{z|(\hat{B})_z \cap A \neq \varnothing\}, \tag{2.5}$$

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

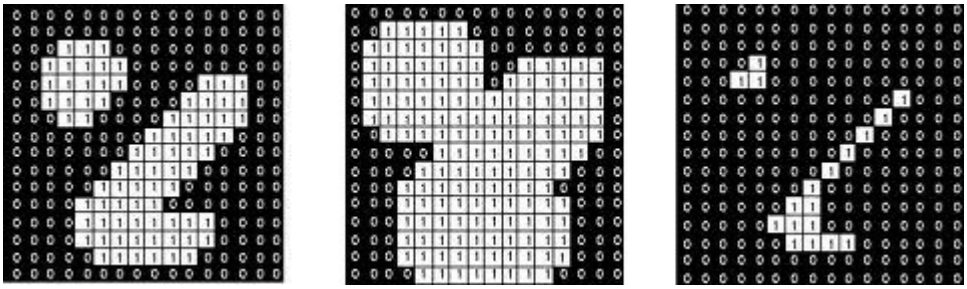

Figura 2.6: Imagen original, dilatada y erosionada.

donde B es conocido como máscara o kernel. Se considera $\mathbb{Z}^2$ como una imagen binaria y A como un objeto que tiene sus bits puestos a uno. Esta operación permite aumentar la cantidad de píxeles alrededor de la periferia de A y así, agrandar los objetos de la imagen o rellenar agujeros en dichos objetos [24].

Erosión

Sea un conjunto A y B de $\mathbb{Z}^2$, la erosión de A por B, denotada como $A \ominus B$ queda definida como

$$A \ominus B = \{z|(B)_z \subseteq A\} \tag{2.6}$$

Esta operación permite remover píxeles de la frontera de los objetos de una imagen, y por lo general su utilidad es la de remover ruido u objetos irrelevantes en la imagen [24]. En la Figura 2.6 se pueden ver los procesos de erosión y dilatación utilizando una máscara de 3x3 con todos los valores iguales a 1.

2.2.4 Detección de líneas principales

El proceso de detección de líneas principales o bordes consiste en identificar los puntos de la imagen en donde los valores de intensidades de grises cambian en forma abrupta o tiene discontinuidades. En este proyecto se utilizó el método de Canny, ya que su comportamiento es más robusto que el de Sobel ante la presencia de ruido de tipo gaussiano [25].

Método de Canny

Para que un detector de bordes pueda ser considerado óptimo debe cumplir los siguientes puntos:

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figura 2.7: A la izquierda el gradiente en x, a la derecha el gradiente en y.

- Buena detección: el algoritmo debe marcar la mayor cantidad de bordes reales de la imagen como sea posible.
- Buena localización: los bordes marcados deben estar lo más cerca posible del borde real en la imagen.
- Respuesta mínima: el borde de una imagen sólo debe ser marcado una vez, y si es posible, el ruido de la imagen no debe crear bordes falsos.

Esta técnica, que se caracteriza por estar optimizada para la detección de bordes diferenciales, consta de cuatro etapas: reducción del ruido, cálculo de gradientes, supresión no máxima al resultado del gradiente, e histéresis de umbral a la supresión no máxima.

Reducción del ruido. Consiste en realizar la convolución de la imagen con un filtro gaussiano. Para calcular el kernel gaussiano G se utiliza la ecuación (2.7).

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.7}$$

Calcular gradientes. El algoritmo de Canny encuentra básicamente bordes, donde la escala de grises de la imagen cambia más. Estas áreas se encuentran mediante la determinación de los gradientes de la imagen con el método de Sobel [22], donde las componentes del gradiente se aproximan mediante:

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{2.8}$$

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{2.9}$$

y las máscaras que implementan dichos gradientes se muestran en la Figura 2.7.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

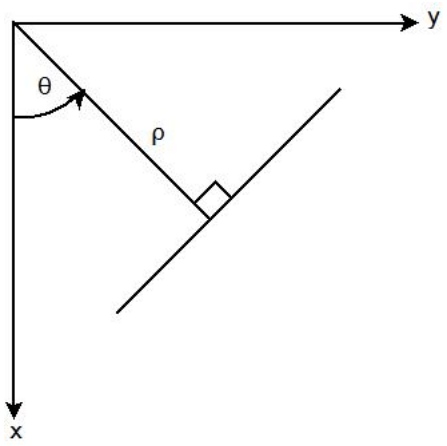

Figura 2.8: Representación normal de una línea.

Supresión no máxima al resultado del gradiente. Este paso consiste en obtener bordes de 1 píxel de grosor, considerando únicamente píxeles cuya magnitud es máxima en bordes gruesos y descartar aquellos cuyas magnitudes no alcancen ese máximo.

Histéresis de umbral a la supresión no máxima. El algoritmo de detección de bordes de Canny utiliza dos umbrales: los bordes con píxeles que superan el umbral mayor se marcan de forma fuerte en la nueva imagen, los píxeles de bordes más débiles que el umbral inferior son eliminados, y los píxeles que se encuentran entre los dos umbrales se marcan de forma débil en la imagen resultante.

Transformada de Hough

Este método considera las relaciones globales entre píxeles permitiendo encontrar ciertos patrones en la imagen como líneas y círculos [27]. El método para encontrar líneas propone un mapeo de los parámetros de las rectas presentes en la imagen en un espacio de parámetros de Hough. Este nuevo espacio de parámetros posee celdas acumuladoras que, para cada ángulo y ubicación de las rectas, almacenan el número de puntos colineales. Este método utiliza la ecuación de la recta en coordenadas polares $\rho = xcos\theta + ysen\theta$ (Figura 2.8) y el espacio de Hough tiene coordenadas (ρ, θ). Ésto se hace porque la ecuación tradicional de la recta $y = ax + b$ no puede manejar las líneas verticales, es decir las pendientes que tienden a infinito.

El atractivo del cálculo de la transformada de Hough surge de la subdivi-

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

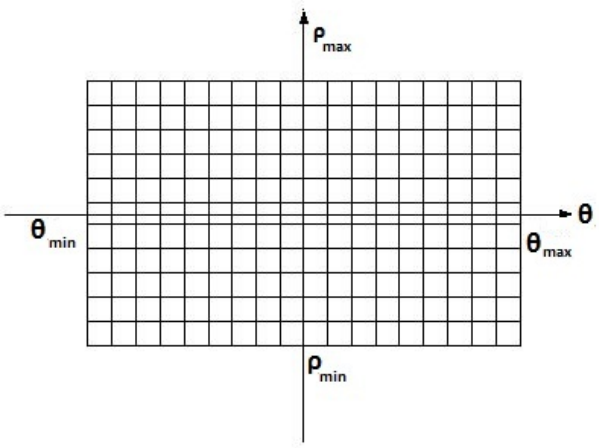

Figura 2.9: Cuantificación del plano $\rho\theta$ en células.

sión del espacio de parámetros en las denominadas *celdas acumuladoras*, que se muestran en la Figura 2.9, donde (ρ_{max}, ρ_{min}) y $(\theta_{max}, \theta_{min})$ son los rangos esperados de ρ y θ. La célula de coordenadas (i, j), con valor de acumulador $A(i, j)$, corresponde al cuadrado asociado con las coordenadas del espacio de parámetro (ρ_i, θ_j). Inicialmente estas células están puestas a cero. Después, para cada punto (ρ_k, θ_k), se fija el parámetro ρ igual a cada uno de los valores permitidos de subdivisión y se resuelve para el θ correspondiente. Las θ resultantes se redondean después al valor más próximo permitido del eje θ. Si una elección de ρ_p resulta ser la solución θ_q, se fija $A(p, q) = A(p, q) + 1$. Al final de este procedimiento, un valor de M en $A(i, j)$ corresponde a M puntos del plano xy situados en la línea $xcos\theta_j + ysen\theta_j = \rho_i$. La precisión de la colinealidad de estos puntos está determinada por el número de subdivisiones del plano $\rho\theta$. En la Figura 2.9 se puede ver la subdivisión del espacio de parámetros $\rho\theta$.

El rango para el ángulo θ es $\pm 90°$, medido con respecto al eje x. Analizando la Figura 2.8, una línea horizontal tiene un $\theta = 0°$, siendo ρ igual a la x positiva. De manera similar, una línea vertical tiene un $\theta = 90°$, siendo ρ igual a la y positiva, o $\theta = -90°$, siendo ρ igual a la y negativa [22].

2.2.5 Componentes conectadas

Este método consiste en asignar una misma etiqueta a los píxeles que componen un mismo objeto [28]. Si consideramos la imagen binaria de la Figura 2.10, el método se aplica en dos pasos:

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	1	1	0
0	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0

Figura 2.10: Ejemplo de imagen binaria que contiene tres objetos.

- En el primer paso, se debe recorrer la imagen fila por fila y asignar una etiqueta con valor $v \neq 0$ a cada píxel $P(i, j) \neq 0$. El valor v depende de los vecinos del píxel P(i,j) como se indica a continuación:

  - Si todos los vecinos pertenecen al fondo $P(i, j) = 0$, el píxel se etiqueta con un valor v no usado aún.

  - Si existe un solo vecino con una etiqueta v, entonces $P(i, j) = v$.

  - Si el píxel tiene más de un vecino con etiquetas distintas de cero, se le asigna a $P(i, j)$ cualquiera de las etiquetas. Además, si las etiquetas de los vecinos son distintas entre sí, se almacena dicha correspondencia.

  En la Figura 2.11 se muestra el resultado del método luego de la primer pasada, aplicado a la imagen de la Figura 2.10. Se almacena la correspondencia: 2-4.

- En el segundo paso, se recorre nuevamente la imagen re-etiquetando los píxeles según la tabla de correspondencias. Este paso tiene como objetivo eliminar etiquetas diferentes asignadas a un mismo objeto. En la Figura 2.12 se muestra el resultado del método luego de la segunda pasada, aplicado a la imagen de la Figura 2.11.

2.3 Métodos de extracción de características

En esta sección se presentan los métodos de extracción de características utilizados en el proyecto. Se pretende que éstos generen datos útiles que

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

0	0	0	0	0	0	0	0	0	0
0	0	0	2	2	0	0	3	3	0
0	4	4	4	4	0	0	0	0	0
0	0	0	0	0	0	0	5	5	0
0	0	0	0	0	0	0	5	5	0
0	0	0	0	0	0	0	0	0	0

Figura 2.11: Resultado del primer paso del algoritmo de componentes conectadas.

0	0	0	0	0	0	0	0	0	0
0	0	0	2	2	0	0	3	3	0
0	2	2	2	2	0	0	0	0	0
0	0	0	0	0	0	0	5	5	0
0	0	0	0	0	0	0	5	5	0
0	0	0	0	0	0	0	0	0	0

Figura 2.12: Resultado del segundo paso del algoritmo de componentes conectadas.

puedan ser usados en el proceso de clasificación.

Se decidió utilizar el método de los momentos invariantes de Hu que a priori permite obtener siete valores como características de la imagen, lo que resulta muy eficiente en cuanto a velocidad de procesamiento para correr en dispositivos móviles. La decisión de implementar el método de vector de distancias al primer píxel no nulo de la imagen se fundamento en la sencillez del mismo y que se ha utilizado en muchos trabajos de manera exitosa como método de extracción de características dentro del reconocimiento óptico de caracteres. Por último, decir que no se exploraron otros métodos de extracción de características para poder cumplir con los tiempos establecidos para el proyecto final de carrera, se deja como trabajos futuros la exploración de otras alternativas.

2.3.1 Momentos invariantes de Hu

Los momentos invariantes de Hu permiten extraer una serie de características de los objetos que no varían respecto de la posición, tamaño o rotación

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

de los mismos [29]. Hu obtuvo sus invariantes a través de los momentos geométricos y para una imagen discreta se definen como:

$$u_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - \hat{x})^p (y - \hat{y})^q f(x, y) \tag{2.10}$$

donde u_{pq} es el momento geométrico de orden $(p + q)$, $f(x, y)$ es el valor del píxel en la posición (x, y), $(\hat{x}, \hat{y})$ son las componentes del centroide del objeto y M y N el ancho y alto de la imagen respectivamente. Partiendo de los momentos definidos anteriormente podemos obtener n_{pq}, un momento de orden $p+q$ que es invariante al escalamiento, y su expresión viene dada por:

$$n_{pq} = \frac{u_{pq}}{u_{00}^{1+\frac{p+q}{2}}} \tag{2.11}$$

Utilizando n_{pq}, Hu logró un conjunto de momentos invariantes, todos con un grado menor o igual a tres. Las expresiones matemáticas que dan lugar a los siete momentos invariantes son [29]:

$$\phi_1 = n_{20} + n_{02} \tag{2.12}$$

$$\phi_2 = (n_{20} - n_{02})^2 + 4n_{11}^2 \tag{2.13}$$

$$\phi_3 = (n_{30} - 3n_{12})^2 + (3n_{21} - n_{03})^2 \tag{2.14}$$

$$\phi_4 = (n_{30} + n_{12})^2 + (n_{21} + n_{03})^2 \tag{2.15}$$

$$\begin{aligned} \phi_5 = (n_{30} - 3n_{12})(n_{30} + n_{12})((n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2) + \\ (3n_{21} - n_{03})(n_{21} + n_{03}) \times (3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2) \end{aligned} \tag{2.16}$$

$$\phi_6 = (n_{20} - n_{02})((n_{30} + n_{12})^2 - (n_{21} + n_{03})^2) + 4n_{11}(n_{30} + n_{12})(n_{21} + n_{03}) \tag{2.17}$$

$$\begin{aligned} \phi_7 = (3n_{21} - n_{03})(n_{30} + n_{12})((n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2) - \\ (n_{30} - 3n_{12})(n_{21} + n_{03}) \times (3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2) \end{aligned} \tag{2.18}$$

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

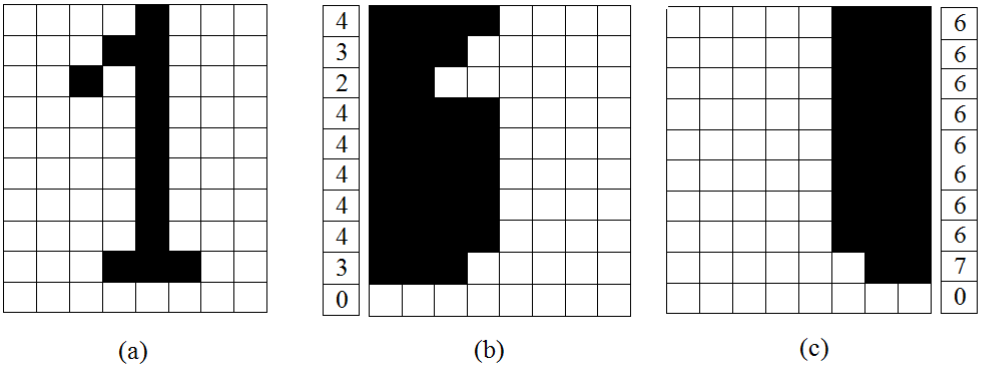

Figura 2.13: Información almacenada de los contornos. (a) Imagen original; (b) Vector distancia Izq-Der; (c) Vector distancia Der-Izq.

2.3.2 Vector de distancias al primer píxel no nulo

Este método consiste en calcular un vector que contiene, para cada fila, las distancias al primer píxel no nulo de la imagen. Ésto se calcula tanto de izquierda a derecha como de derecha a izquierda, obteniendo información sobre el contorno de la imagen. El método considera una distancia igual a uno cuando se encuentra un píxel no nulo en el primer píxel de la línea. Las distancias para las filas restantes se obtienen de forma similar al caso anterior. Para el caso especial en que no existe un píxel en la línea, la distancia pasará a valer cero. En la Figura 2.13 se puede ver el tipo de información que se almacena para cada contorno del objeto al utilizar la distancia como característica. También se puede apreciar que utilizando este método se reduce la dimensionalidad de los datos a la cantidad de píxeles en los laterales de la imagen.

2.4 Métodos de clasificación

Una vez definidas las características que serán extraídas de los diferentes objetos, es necesario proponer algún método para clasificar los patrones. En esta sección se presentará un método que permite clasificar patrones en base a las características presentadas anteriormente y además se introducirá un método alternativo basado en reconocimiento óptico de caracteres (OCR: del inglés Optical Character Recognition).

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

2.4.1 Clasificador basado en los vecinos más cercanos

La idea básica del método de clasificación de los k vecinos más cercanos (k-NN: del inglés k- Nearest Neighbours) es asignar la etiqueta de clase a un patrón dado según la clase a la que pertenezca la mayoría de sus k prototipos más cercanos en el espacio de características. El paradigma se fundamenta en una idea muy simple, intuitiva y de fácil implementación, lo que hace que sea una técnica de clasificación muy utilizada.

Este método utiliza clasificación supervisada estimando la distancia de la muestra que se pretende clasificar a cierto número de muestras (k vecinos), determinando su pertenencia a la clase de la que encuentre más vecinos etiquetados, basándonos en un criterio de mínima distancia [30]. Es una técnica válida solo para datos numéricos y en función del tipo de datos que se quiere evaluar, a veces conviene utilizar más o menos vecinos. Generalmente más vecinos reducen el ruido de la clasificación, sin embargo al incrementar el número de clases se dificulta la decisión de la clase a la que pertenece la muestra de entrada. Ante este problema se aplican reglas que determinan qué hacer en caso de empate entre dos clases respecto a la pertenencia de una muestra.

El algoritmo más sencillo para la búsqueda del vecino más cercano es el conocido como fuerza bruta, que calcula todas las distancias de las muestras del entrenamiento respecto a una muestra concreta, y asigna como conjunto de vecinos más cercanos aquél cuya distancia sea mínima. Los requisitos que deben cumplir las funciones de distancias son:

1. $d(m, l) \geqq 0$.
2. $d(m, m) = 0$.
3. $d(m, l) = d(l, m)$.
4. $d(m, l) \leqq d(m, j) + d(j, l)$.

Dado un conjunto de muestras $X = x_1, x_2, ..., x_N \in R^p$, el índice de similitud entre individuos más utilizado es la distancia euclídea:

$$d(m, l) = \sqrt{(x_{m1} - x_{l1})^2 + (x_{m2} - x_{l2})^2 + ... + (x_{mp} - x_{lp})^2} \tag{2.19}$$

La Figura 2.14 muestra un ejemplo que intenta clasificar el punto verde. Si con el clasificador k-NN se utilizan los 3 vecinos más cercanos, la muestra pertenecerá a la clase formada por triángulos porque 2 de los elementos que están más cerca de la muestra evaluada son triángulos y uno cuadrado. Sin embargo, si se utiliza el clasificador con los 5 vecinos más cercanos (5-NN), el elemento evaluado pertenecerá a la clase formada por cuadrados ya que hay 3 cuadrados y dos triángulos entre las cinco muestras más próximas al patrón de entrada.

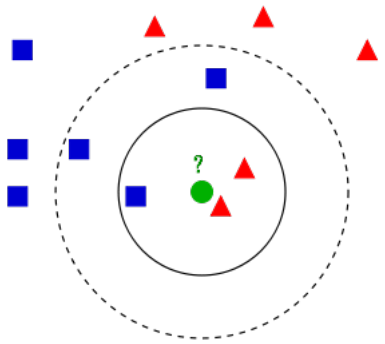

Figura 2.14: Ejemplo de clasificación con k-NN (tomado de Wikipedia).

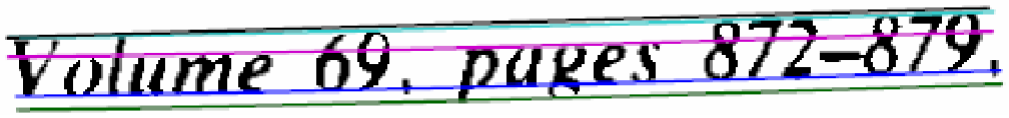

Figura 2.15: Ejemplo de una frase con líneas de tendencia.

2.4.2 Reconocimiento óptico de caracteres

Tesseract es una herramienta que permite hacer OCR y fue desarrollada por Hewlett Packard como software propietario a principio de los 90' [31]. Tras diez años sin desarrollo, se liberó como código abierto en el año 2005. Tesseract es desarrollado actualmente por Google y distribuido bajo la licencia Apache, versión 2.0. Además, es unos de los OCR más utilizados y con mayor precisión actualmente. Tesseract puede procesar texto en diferentes idiomas: inglés, francés, italiano, alemán, español, portugués, entre otros [31]. A continuación se presentan los pasos más importantes para que este OCR logre el reconocimiento óptico de caracteres.

- Reconocimiento de líneas: se encuentra cada línea de texto mediante la construcción de blobs (Binary Large OBjects, objetos binarios grandes), asignando a cada línea de texto un blob. La altura media del blob aproxima el tamaño del texto en la región, lo que es seguro para filtrar blobs que son menores que dicha altura, como por ejemplo: signos de puntuación o el ruido.

- Ajuste de líneas de tendencia: una vez que se encuentran las líneas de texto, se ajustan líneas de tendencia con aproximación cuadrática para seguir la trayectoria de la línea. Esto permite a Tesseract analizar páginas con líneas de texto curvas. La Figura 2.15 muestra el ajuste, y se puede ver que la línea de tendencia cian es levemente curva.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

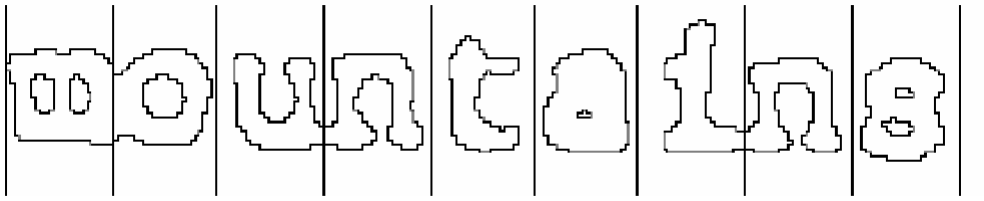

Figura 2.16: Ejemplo de segmentación de caracteres de longitud fija.

- Detección y cortado de caracteres de longitud fija: cuando se encuentra texto con caracteres de longitud fija, Tesseract corta la palabra en caracteres usando la longitud determinada como se muestra en la Figura 2.16.
- Reconocimiento de palabras: se realiza un proceso de dos pasadas. En la primera pasada, se intenta reconocer cada palabra. Cada palabra que se detecta correctamente pasa a un clasificador adaptativo como dato de entrenamiento y así, este clasificador hace un reconocimiento más preciso del texto. Dado que el clasificador puede haber aprendido algo útil después de la primer pasada, se recorre una segunda vez.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

CAPÍTULO 3

Método propuesto

En este capítulo se describen las etapas que conforman el método propuesto. En primer lugar, se presenta el diseño del sistema. Luego, se define la forma de realizar el relevamiento fotográfico utilizando diversos dispositivos móviles y condiciones de iluminación, para generar una base de datos de imágenes de los mojones informativos. A continuación, se presentan las técnicas de preprocesamiento que demostraron tener el mejor desempeño y se evalúan métodos de análisis y clasificación de imágenes, orientados a obtener el mejor rendimiento con el fin de lograr una aplicación veloz. Finalmente, se diseña una interfaz gráfica que permite al usuario capturar una fotografía del mojón (Figura 3.1)y obtener información multimedia turística relacionada.

3.1 Diseño del sistema

El trabajo cuenta con tres etapas bien diferenciadas, procesamiento de la imagen, extracción de características y clasificación, e, interfaz. El diseño del método propuesto se puede ver en la Figura 3.2. A continuación se explica brevemente cada una de las tres etapas:

- preprocesamiento: esta etapa tiene dos fases. La primera, consiste en aplicar a la imagen una serie de técnicas de procesamiento de imágenes de manera que la misma quede en condiciones adecuadas para la posterior extracción de características. La segunda, se basa en el recorte de los dígitos que posee

Figura 3.1: Mojón informativo. (Imagen tomada de http://santafeciudad.gov.ar)

cada mojón y que permite identificar cada uno de ellos de manera unívoca.

- extracción de características y clasificación: también consta de dos fases bien definidas. Por un lado, se realiza la extracción de características sobre los identificadores únicos de cada mojón, y, por otro, se clasifica en base a dichas características.

- interfaz de usuario y presentación de información multimedia: se evalúa qué información puede ser de relevancia para el usuario en base a hechos históricos y culturales relacionados con el mojón reconocido. Luego se selecciona la forma adecuada para alojar la información (fotos, videos, textos, audios, etc). Por último, se implementan los métodos de la interfaz de usuario para la visualización de la misma.

3.2 Generación de una base de datos

Se procedió a realizar la captura de fotografías de los 17 mojones del Camino de la Constitución de la ciudad de Santa Fe. Se desarrolló un protocolo que permitió tener una variedad suficiente de fotografías para poder trabajar durante todo el proyecto.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

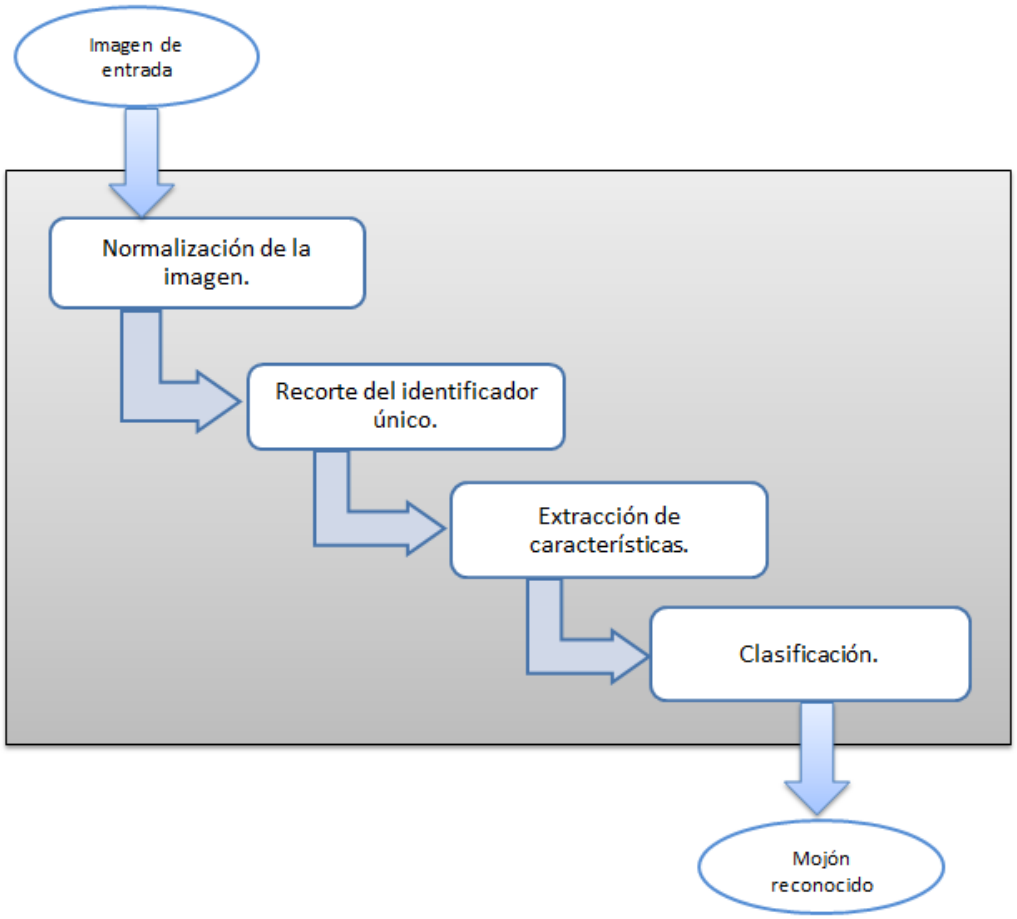

Figura 3.2: Proceso general del método propuesto.

3.2.1 Definición del protocolo para la adquisición de imágenes

El primer paso fue diseñar un protocolo para capturar las fotografías y generar una base de datos de imágenes. Se tomó un gran número de fotografías de los 17 mojones del Camino de la Constitución de la ciudad de Santa Fe, considerando siempre la mayor naturalidad posible en la captura para poder lograr un sistema más robusto. Se consideraron las siguientes condiciones:

1. Rotación: las fotos se capturaron con rotación hacia la izquierda o derecha.
2. Orientación del dispositivo: estuvo ubicado en forma vertical u horizontal, indistintamente.
3. Independencia del dispositivo: las capturas se realizaron para distintos dis-

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

positivos móviles:

a) Smartphone Samsung Galaxy i5550, sistema operativo Android 2.3.

b) Tablet ASUS Transformer TF101.CPU NVIDIA Tegra 2 1.0GHz. Android 3.2 Honeycomb O.S.

4. Múltiples resoluciones: las fotografías fueron capturadas con 1.3, 3.0 y 5 Megapíxeles.

5. Condiciones de iluminación: se consideraron condiciones naturales sin iluminación artificial y se capturaron de mañana, de tarde y de noche.

La base de datos conformada consiste en un conjunto de 693 fotografías. Éstas se utilizaron tanto para el procesamiento como para la clasificación de las imágenes.

3.2.2 Captura de fotografías según protocolo

En esta sección se muestran algunas imágenes capturadas durante el trabajo de campo según el protocolo establecido en la sección anterior.

En la Figura 3.3 se pueden ver ejemplos de capturas realizadas sobre diferentes mojones en condiciones de iluminación diurna con rotaciones tanto a izquierda como derecha. Como se puede observar los ángulos de inclinación permiten contemplar aquellos casos en los que el usuario no realiza una captura con un ángulo de inclinación cero.

La aplicación debe responder de forma correcta a las capturas realizadas por el usuario tanto cuando este coloca el dispositivo en forma vertical u horizontal, por lo tanto se realizaron capturas con ambas disposiciones, tal como se muestra en las Figuras 3.4 y 3.5.

Otra de las condiciones que se definieron en el protocolo de captura es la independencia del dispositivo. En la Figura 3.6 se muestran algunas capturas realizadas con una Tablet ASUS Transformer TF101 y en la Figura 3.7 se muestran las realizadas con un Smartphone Samsung Galaxy i5550.

Por último, se realizaron capturas nocturnas con los diferentes dispositivos para contemplar aquellos casos en donde el usuario decide utilizar la aplicación en dicha condición. En la Figura 3.8 se pueden ver algunos ejemplos de capturas realizadas en condiciones de iluminación nocturna con ambos dispositivos.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Figura 3.3: Mojones capturados con rotación hacia izquierda y derecha.

Figura 3.4: Mojones capturados con orientación vertical del dispositivo.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Figura 3.5: Mojones capturados con orientación horizontal del dispositivo.

Figura 3.6: Capturas diurnas de mojones, con Tablet Asus.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Figura 3.7: Capturas diurnas de mojones, con Samsung Galaxy.

Figura 3.8: Capturas nocturnas de mojones, ambos dispositivos móviles.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

3.3 Preprocesamiento de imágenes

Esta etapa se puede dividir en dos partes, en la primera se busca normalizar las imágenes capturadas desde el dispositivo móvil y en la segunda se realiza la extracción de la información distintiva del mojón.

3.3.1 Normalización de imágenes

Luego de evaluar varias alternativas, se obtuvo una secuencia de operaciones que permite preprocesar cualquier imagen para normalizarla (como por ejemplo la que se muestra en la Figura 3.9):

- Escalar la imagen: considerando el compromiso entre la velocidad de cómputo y los detalles en la imagen, el tamaño elegido es 800x600 píxeles.
- Convertir a escala de grises: se realiza porque los procesos posteriores no precisan información del color y así se manipula menos información (ver Figura 3.10).
- Ecualizar el histograma: se realiza para mejorar el contraste (ver Figura 3.11).
- Binarizar: es útil para la posterior detección de bordes.
- Detectar bordes: se obtienen los bordes de la imagen aplicando el método de Canny (ver Figura 3.13).
- Encontrar líneas principales: se utiliza la *transformada de Hough* para encontrar píxeles colineales que permitan determinar el ángulo de rotación de la imagen capturada utilizando características particulares presentes en todas las imágenes.
- Rotar: se rota la imagen utilizando el ángulo obtenido previamente (ver Figura 3.14).

  Cabe destacar que se deja fuera del alcance de este proyecto el análisis de imágenes con perspectiva, considerando que las capturas se realizan de frente al mojón. De todos modos, la interfaz gráfica será desarrollada para inducir al usuario a tomar la fotografía de frente al mojón.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

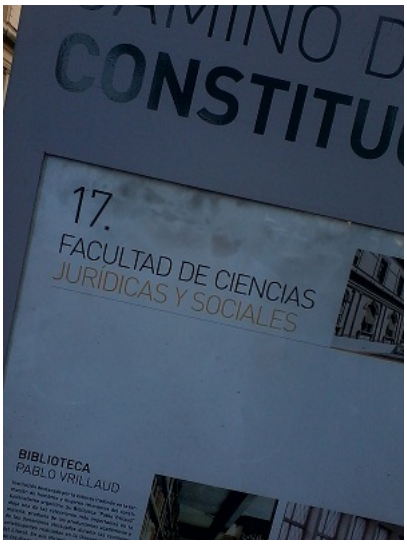

Figura 3.9: Imagen de entrada en el proceso de normalización.

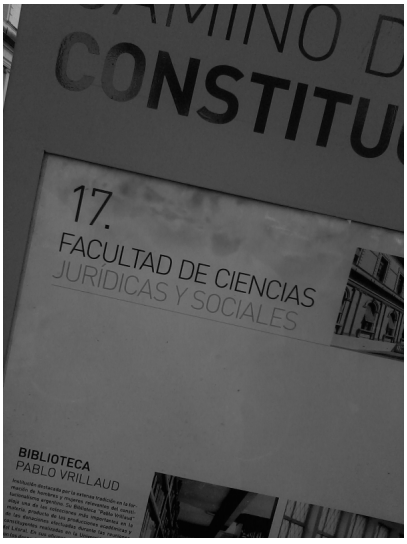

Figura 3.10: Imagen en escala de grises.

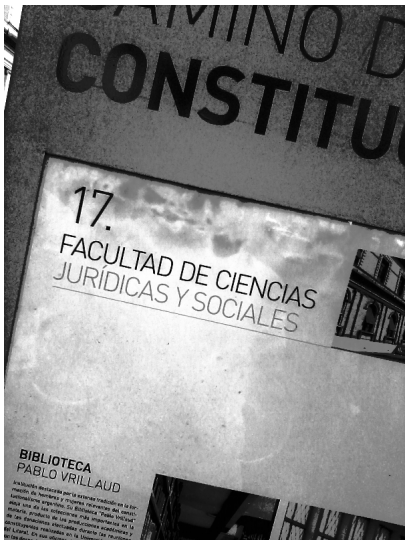

Figura 3.11: Imagen ecualizada.

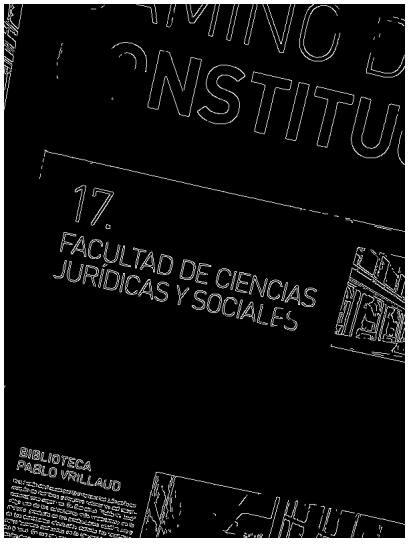

Figura 3.12: Binarización y Canny aplicado a una imagen.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

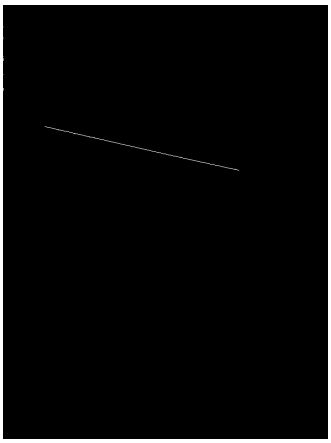

Figura 3.13: Línea más extensa encontrada con la transformada de Hough.

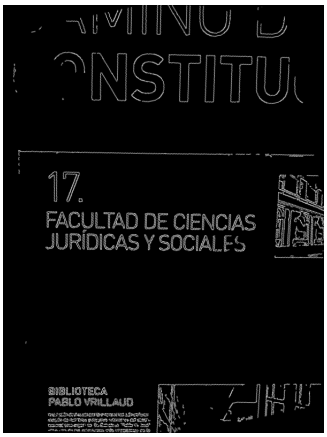

Figura 3.14: Imagen rotada.

Figura 3.15: Cuadrante superior izquierdo.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Figura 3.16: Corte horizontal y vertical

Figura 3.17: Imagen de sólo números.

3.3.2 Extracción del identificador único

Luego de realizar la normalización, se procede a extraer el número de mojón, identificador único escogido como patrón. Para ésto se realiza el siguiente procedimiento:

1. Localizar: se extrae el cuadrante superior izquierdo de la imagen (ver Figura 3.15).

2. Dilatar: se realiza un proceso de dilatación con una máscara de 2x2, para que las líneas de los marcos de los mojones queden bien definidas, y así, facilitar el recorte del identificador.

3. Recortar zona de interés: se buscan las líneas horizontal y vertical correspondientes a los marcos de los mojones para hacer el recorte. En primer lugar se utilizan acumuladores de píxels no nulos a lo largo de las filas para encontrar la línea más extensa, luego se aplica el mismo proceso sobre las columnas. Es importante el orden de estas operaciones para salvar casos de mojones con perspectiva (ver Figura 3.16).

4. Erosionar: se realiza una erosión para que los dígitos estén bien separados.

5. Extraer número: se utiliza el acumulador de píxeles no nulos sobre filas para obtener la imagen del número basado en los espacios anterior y posterior (ver Figura 3.17).

6. Recortar los dígitos: se realiza utilizando el método de componentes conectadas [28]. Se seleccionan las dos componentes más grandes y se evalúa la relación de aspecto de cada una para descartar ruido. En la Figura 3.18 se observa un ejemplo del resultado final del proceso de extracción del identificador único.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Figura 3.18: Imágenes de la extracción del identificador.

N	Mom.1	Mom.2	Mom.3	Mom.4	Mom.5	Mom.6	Mom.7
0	0.387	0.024	0.000	0.002	0.016	0.115	0.954
1	0.000	0.405	0.093	0.073	0.021	0.158	0.946
2	0.633	0.173	0.040	0.024	0.016	0.101	0.956
3	1.000	1.000	0.139	0.126	0.000	0.000	0.962
4	0.037	0.082	0.343	0.298	0.120	0.284	1.000
5	0.443	0.157	0.002	0.064	0.016	0.154	0.955
6	0.035	0.000	0.114	0.059	0.020	0.141	0.948
7	0.828	0.839	1.000	1.000	1.000	1.000	0.000
8	0.626	0.283	0.001	0.000	0.016	0.114	0.954
9	0.129	0.594	9.24E-4	0.013	0.018	0.142	0.956

Tabla 3.1: Momentos de Hu de referencia normalizados.

3.4 Extracción de características

En esta etapa se extraen características relevantes de las imágenes de los dígitos. A continuación se presentan las alternativas consideradas.

3.4.1 Momentos invariantes de Hu

Se almacenaron dígitos obtenidos en el proceso de extracción del identificador único para tener patrones de referencia que permitan trabajar en esta etapa. A los dígitos almacenados se le calculó los momentos invariantes de Hu. La implementación del código fuente del método se realizó con las funciones que provee la biblioteca OpenCV.

Dado que el orden de los momentos de Hu difieren demasiado, y al calcular las distancias en el método de clasificación, unos pesan más que otros, se decidió normalizar cada momento sobre los 10 patrones. En la Tabla 3.1 se pueden ver los datos normalizados.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

3.4.2 Vector de distancias al primer píxel no nulo

Este método utiliza la imagen binaria del dígito obtenida previamente. Se crea un vector columna de longitud igual al alto de la imagen cuyos valores, para cada fila, representan las distancias al primer píxel no nulo de la imagen, en el sentido izquierda a derecha. Luego, se crea otro vector considerando las distancias de derecha a izquierda. Esta información es representativa del contorno del número en la imagen[1].

3.5 Métodos de clasificación

A partir de imágenes limpias y manualmente ajustadas se calcularon los patrones de referencia para los momentos invariantes de Hu y vectores de distancias al primer píxel no nulos. Los mismos, son un promedio de los valores almacenados para cada dígito.

El paso siguiente es comparar las características calculadas para una nueva captura con las características de los patrones de referencia. Para realizar ésto, y considerando la velocidad de cómputo, se decidió clasificar utilizando una variación del método de vecinos mas cercanos k-NN.

3.5.1 k-NN con distancia mínima

En el k-NN con distancia mínima se comienza seleccionando una muestra por clase, en nuestro caso cada dígito se corresponde con una clase, normalmente el caso más cercano al baricentro de todos los elementos de dicha clase. Este paso reduce la cantidad de casos a almacenar. A continuación se asigna la nueva muestra a la clase cuyo representante esté más cerca. El procedimiento anterior puede verse como un 1-NN aplicado a un conjunto de 9 casos (uno por cada clase), y matemáticamente se reduce a la ecuación (3.1).

$$\text{Clase}_{\text{Y}} = \min_{j} \left\{ \sqrt{\sum_{i=1}^{n} (\mathbf{X}_i^j - Y_i)^2} \right\} \tag{3.1}$$

donde $\mathbf{X}^j$ son los vectores de referencia, Y el patrón a clasificar y n la dimensión de las características.

[1]Para esta etapa, las imágenes están normalizadas en 125x65 píxeles.

El coste computacional de este procedimiento es inferior al k-NN genérico, si bien su efectividad está condicionada a la homogeneidad dentro de las clases (cuanto mayor sea dicha homogeneidad, se espera que el procedimiento funcione mejor).

3.5.2 Método de clasificación alternativo con OCR

Como alternativa de clasificación se consideró la utilización del reconocimiento óptico de caracteres. Se investigaron varias alternativas de OCR y finalmente se seleccionó Tesseract OCR [31]. Ésta es una biblioteca muy eficiente, ampliamente utilizada[2] y esta implementada en C++. Sin embargo, se puede compilar como un proyecto Android que contiene código nativo utilizando el Native Development Kit (NDK) y luego incluir este proyecto como una biblioteca dentro de la aplicación Android. Tesseract OCR posee una API con una serie de funcionalidades que permiten el reconocimiento de caracteres.

Aquí también se realiza el preprocesamiento completo de la imagen para obtener los dígitos, sin embargo, se recortan los dígitos de la imagen original para ser utilizados en el OCR, dado que se comprobó experimentalmente que de esta forma se obtienen mejores resultados. El uso de las imágenes de sólo dígitos mejora el rendimiento del OCR.

3.6 Diseño de la interfaz de usuario

3.6.1 Selección de información de interés y métodos de acceso

De un análisis de las diferentes alternativas de información que podrían proveerse al usuario, se decidió que la aplicación deba brindar información en múltiples formatos multimediales. Entre éstos podemos mencionar la presentación de una galería de imágenes, la reproducción de videos, la visualización de documentos y páginas web, etc. Una vez definidos los contenidos a los que podrá acceder el usuario, se determinó que la forma más adecuada de acceso a éstos era vía internet, ya sea mediante Wi-Fi o 3G presente en el dispositivo con sistema operativo Android. Ésto se fundamenta en primer lugar en que el almacenamiento de toda la información multimedia implica un consumo excesivo e injustificable de recursos del dispositivo, y por otra parte, este diseño permite la actualización del contenido

[2]Disponible en https://code.google.com/p/tesseract-ocr/.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

multimedia de forma independiente de los dispositivos en los cuales está instalada la aplicación.

3.6.2 Desarrollo de la interfaz de usuario

En el diseño de la aplicación se priorizó la simpleza de la interfaz gráfica y la facilidad de acceso a los contenidos por parte del usuario. La pantalla inicial del sistema presenta la aplicación con imágenes y textos referidos al Camino de la Constitución, y permite que el usuario acceda a una pantalla de ayuda acerca de cómo utilizar el sistema o bien comience a utilizarlo. En la Figura 3.19 se muestra la pantalla inicial del sistema.

Es importante recordar que en Android, el diseño y la lógica de una pantalla se programan en archivos separados. Por un lado, en la carpeta **/res/layout/pinicial.xml** se tiene el diseño de la pantalla definido como un archivo XML y por otro lado, en la carpeta **/src/paquete.java/pInicial.java**, se encuentra el código Java [21] que determina la lógica de la pantalla. En el XML se definen los elementos visuales que componen la interfaz de la pantalla principal y se especifican las propiedades o atributos [32].

El XML de la pantalla inicial está compuesto por un **LinearLayout** y dentro de este se incluyeron 3 controles: una etiqueta (**TextView**), un botón (**Button**) y otro tipo de botón (**ImageButton**) (ver Código A.1).

Una vez definido el diseño de la pantalla inicial, se debe programar la lógica mediante código Java. Básicamente lo que se hace es iniciar la nueva actividad mediante un **intent** que responde al evento **onClick** del dispositivo dentro del método **OnCreate**. En el código A.2 se pueden ver los detalles de la implementación de dicha funcionalidad.

Una vez que se ingresa a la aplicación se accede a la cámara, que es inicializada con funciones de OpenCV. El código A.3 permite dicha inicialización.

En la Figura 3.20 se puede ver la pantalla que permite capturar una fotografía del mojón para su análisis y clasificación. Mediante un **canvas** se definió un marco y se presenta una etiqueta que ayuda al usuario a enfocar el mojón para capturar una fotografía. También se pueden ver dos botones en la parte inferior, uno permite "tomar la fotografía y realizar la detección" y el otro "muestra el resultado". Existen dos alternativas para mostrar un menú en una aplicación Android. La primera es mediante la definición del menú en un fichero XML, y la segunda, que es la que se usó en la aplicación, creando el menú directamente mediante código Java, tal como se muestra en el código A.4 del apéndice junto con su explicación.

Una vez realizado el procesamiento de la imagen y el reconocimiento óptico de caracteres (OCR), se puede ver información relacionada al cartel reconocido. Para esta etapa se desarrolló un cartel informativo que indica cual es el mojón

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Figura 3.19: Pantalla inicial.

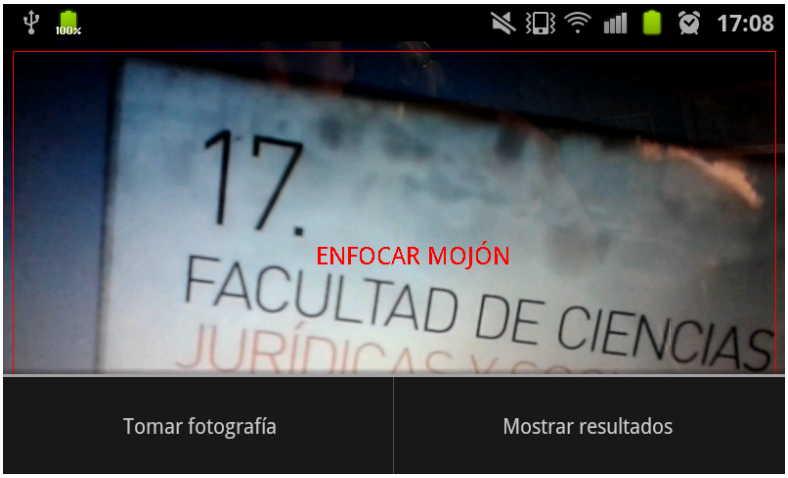

Figura 3.20: Menú de la aplicación.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

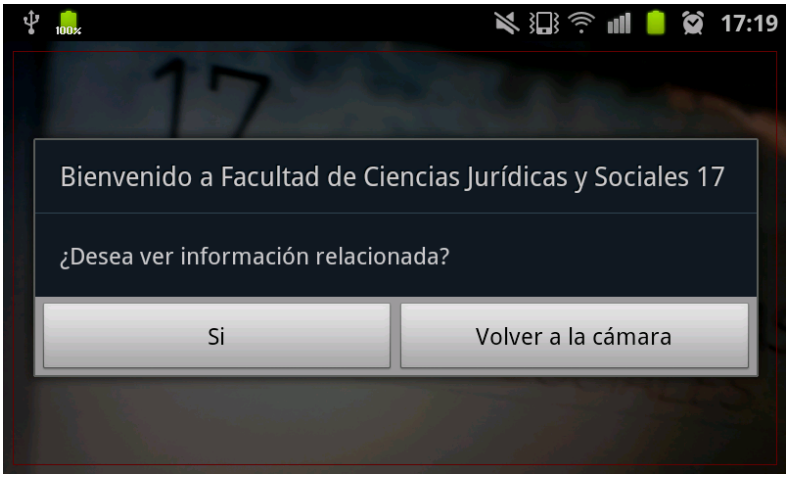

Figura 3.21: Cartel informativo.

identificado. Esta es una forma de chequear si el resultado es correcto, y así el usuario puede decidir si ver la información relacionada o realizar una nueva captura (ver Figura 3.21).

El cartel informativo se generó con la implementación de cuadros de diálogos en Android [33]. La misma se realiza mediante dos métodos de la clase activity, el primero para crear el diálogo por primera vez: **onCreateDialog()**; y el segundo para poder modificarlos de forma dinámica cada vez que se muestran: **onPrepareDialog()** [16]. En esta aplicación sólo se utiliza el primero de éstos y el código completo de la Figura 3.21 se puede ver en A.5.

Para continuar con la implementación se definieron una serie de botones que permiten acceder al contenido o información útil a visualizar. Se decidió visualizar imágenes on-line, links web, documentos en pdf y videos desde YouTube como se muestra en la Figura 3.22. El código que se utilizó para desarrollar los botones es análogo al explicado para la pantalla inicial (ver código A.2).

3.6.3 Desarrollo de métodos para visualizar contenido

El contenido que proporciona la aplicación son sitios web, galería de imágenes en línea, videos de YouTube, y documentos relacionados.

Reproducción de videos

El único recurso que se decidió almacenar de forma local es el video de ayuda al usuario, que puede accederse desde la pantalla inicial. Para reproducir videos

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

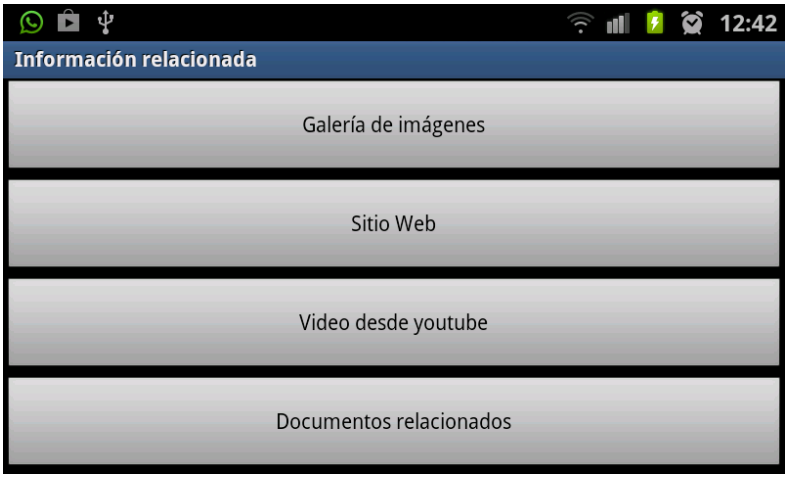

Figura 3.22: Información relacionada.

almacenados como recursos de la aplicación, lo primero que se hizo fue crear un archivo **video.xml**, con un control **VideoView** que ocupe toda la pantalla donde se visualizará dicho video (ver código A.6). Una vez definido el XML, se procedió a generar una actividad en Java a la que se llamó **VideoPlayer.java**. En la misma, se implementó el método **onCreate()**, y a su vez, dentro de este se creo un objeto **VideoView** que se configuró de una forma determinada. Los detalles de implementación para la reproducción de video se muestran en el código A.7.

Sitios Web, Galería en línea, Videos desde YouTube, Documentos relacionados

Se decidió explicar estos cuatro componentes en forma conjunta, dada la similitud para el desarrollo de cada uno de ellos.

Como se puede observar en el código 3.1, la implementación de estos componentes es muy sencilla, en la línea 1 se creó un **intent**, al que en la línea 2, se le asignó el path del link al que se desea acceder. Por último, en la línea 3 se inicia la actividad. El código XML se encuentra en el Apéndice A (código A.8). Se definió la reproducción de video desde YouTube de manera análoga, dado que Android reconoce el link y proporciona la opción para iniciar la reproducción de un video de YouTube. Para la galería en línea, se creó una cuenta en flickr [3] y se subieron imágenes para su reproducción desde internet, por lo tanto el código Android que se utilizó para desarrollar dicha funcionalidad también es similar al explicado anteriormente.

En el caso de "Documentos relacionados", se decidió que el usuario pueda rea-

[3] http://www.flickr.com/

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

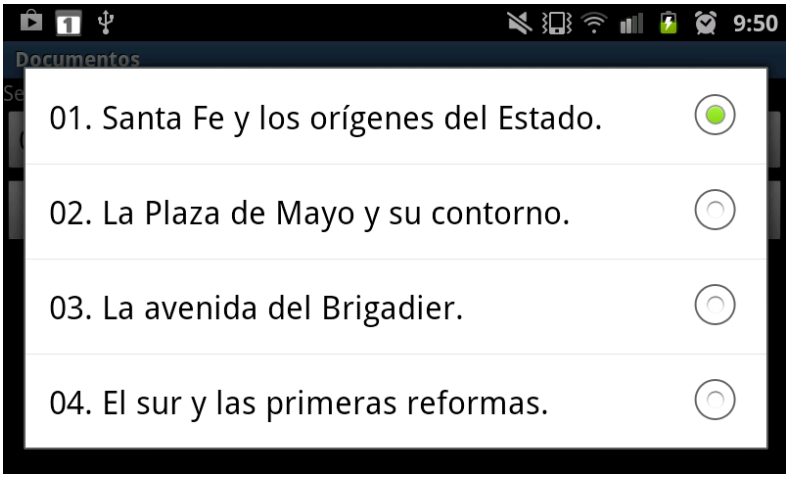

Figura 3.23: Opciones de documentos relacionados.

lizar la descarga de documentos en formato pdf. Para que puedan seleccionar estos documentos se definió un control de selección llamado **Spinner** [34] (listas desplegables) que permite al usuario seleccionar el documento que desea visualizar, tal como se muestra en la Figura 3.23. En el apéndice A se pueden ver los códigos A.9 y A.10 que permiten la implementación de esta funcionalidad junto con una explicación detallada.

Cod. 3.1: Código Java de acceso a link web.

```
1  Intent intent = new Intent(Intent.ACTION_VIEW);
2  intent.setData(Uri.parse("http://www.santafeciudad.gov.ar/
3     bicentenario/ver/multimedia/camino_constitucion.php"));
4  startActivity(intent);
5
```

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

CAPÍTULO 4

Experimentos y resultados

En este capítulo, se presentan las pruebas y se discuten los resultados de cada etapa del método propuesto. Primeramente, se detallan y discuten los experimentos realizados con respecto a la etapa de extracción del identificador único, utilizando diferentes dispositivos y resoluciones. Luego, se presentan pruebas discutiendo la efectividad y velocidad de cada método de clasificación. También se evalúa el desempeño del sistema en condiciones adversas de iluminación, como lo es un escenario nocturno sin iluminación dedicada. Finalmente, se discuten los resultados obtenidos en cada unos de los experimentos.

4.1 Experimentos

En esta sección se presentan tres experimentos. El primero, permite evaluar el comportamiento de la aplicación con imágenes de múltiples resoluciones con diferentes dispositivos móviles en la etapa de extracción del identificador único. El segundo, evalúa el porcentaje de efectividad y la velocidad de cada método de clasificación. El tercero, permite comprobar el funcionamiento del sistema en condiciones de iluminación adversas utilizando imágenes capturadas con diferentes resoluciones.

Tabla 4.1: Resultados de clasificación con luz natural.

Dispositivo	Resolución	Cantidad de Imágenes	Aciertos
Samsung Galaxy i5550.	1.3 Mpx.	81	75,3 %
Tablet Asus	1.3 Mpx.	45	68,8 %
Tablet Asus	3.0 Mpx.	45	66,6 %
Tablet Asus	5.0 Mpx.	45	71,1 %

Experimento 1

El experimento consistió en realizar pruebas de la aplicación en los diferentes lugares donde se encuentran instalados los mojones con condiciones de iluminación natural. Con este experimento se busca evaluar detalladamente el procesamiento y el recorte de los dígitos utilizados como identificador único de la aplicación. Además, se pretende determinar la robustez del sistema frente a imágenes capturadas con distintas resoluciones (1.3 Mpx, 3.0 Mpx y 5 Mpx) y dispositivos diferentes.

En la Tabla 4.1 se muestran los dos dispositivos utilizados:

- Smartphone Samsung Galaxy i5550, sistema operativo Android 2.3.
- Tablet ASUS Transformer TF101.CPU NVIDIA Tegra 2 1.0GHz. Android 3.2 Honeycomb O.S.

Además, se pueden observar las resoluciones utilizadas, la cantidad de imágenes involucradas en el experimento y por último el porcentaje de aciertos para cada caso.

Se observa un rendimiento promedio del 70 % en la etapa de recorte de los dígitos, habiendo obtenido el dispositivo Samsung Galaxy i5550 resultados ligeramente mejores que la Tablet Asus por su mejor calidad de captura, en cuanto a posicionamiento de la cámara en el momento de realizar las capturas durante el trabajo de campo.

Experimento 2

Utilizando las imágenes que presentaron casos de éxito en el experimento anterior, se decidió realizar nuevas pruebas evaluando el tiempo total de los tres métodos de clasificación, ya que el tiempo de procesamiento (hasta el recorte de los dígitos inclusive) es el mismo para todos los casos. Para darle uniformidad al experimento, se utilizaron los casos de éxito correspondientes a las capturas realizadas con el dispositivo Samsung Galaxy i5550 con una resolución de 1.3 Mpx. También se registró la tasa de aciertos en el reconocimiento de los diferentes mojones.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Tabla 4.2: Resultados de clasificación.

Método de clasificación	Aciertos	Tiempo de ejecución
Vector de distancias a píxeles no nulos	100 %	5475 milisegundos
Momentos invariantes de Hu	66.1 %	1329 milisegundos
Tesseract OCR	100 %	298 milisegundos

La Tabla 4.2 presenta los métodos de clasificación, la tasa de aciertos y el tiempo de ejecución para cada uno de ellos junto con los resultados preliminares. Estos resultados son promedios de evaluar los 61 casos en que la segmentación fue exitosa.

Luego de realizar los experimentos 1 y 2, se pudo apreciar que los pasos más críticos en el procesamiento de la imagen resultaron ser aquellos correspondientes al recorte de los dígitos, más específicamente, la detección de las líneas vertical y horizontal de los marcos de los mojones. Este hecho se debió a que los umbrales para la detección de líneas varían dependiendo de la distancia del usuario al mojón y de la perspectiva con que se realice la captura.

En la etapa de extracción de características, los momentos invariantes de Hu mostraron cierta inestabilidad cuando las imágenes contenían los mismos dígitos pero con presencia de ruido y los resultados obtenidos son inferiores a los esperados.

El método basado en las distancias demostró ser muy bueno a pesar de su sencillez y gracias a que con el preprocesamiento se logra un dígito bien rotado y escalado. A pesar de ésto, se requiere mucho tiempo de cómputo debido a que se trabaja con un vector de 250 elementos para cada patrón.

Finalmente, el OCR demostró ser el que obtuvo más aciertos y respecto de los tiempos de ejecución, también Tesseract OCR fue el de mejor rendimiento. Evidentemente, la opción de clasificación elegida para la aplicación final es Tesseract por su desempeño y velocidad de cómputo.

Experimento 3

Este experimento consistió en realizar pruebas de la aplicación en condiciones nocturnas sin iluminación dedicada y utilizando Tesseract OCR, en vista de los buenos resultados obtenidos en el experimento 2, tanto en porcentaje de acierto como en la velocidad de ejecución. El objetivo de este experimento fue evaluar la robustez de la aplicación en condiciones nocturnas y para esto se utilizaron 76 imágenes capturadas con la Tablet ASUS Transformer TF101, Android 3.2 y diferentes resoluciones. La Tabla 4.3 muestra el dispositivo utilizado, la resolución empleada, la cantidad de imágenes y el porcentaje de acierto obtenido en dichas condiciones.

Se puede observar que los porcentajes de acierto en condiciones nocturnas son

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Tabla 4.3: Resultados de clasificación en condiciones nocturnas con Tablet Asus y diferentes resoluciones.

Dispositivo	Resolución	Cantidad de Imágenes	Aciertos
Tablet Asus	1.3 Mpx.	27	44,4 %
Tablet Asus	3.0 Mpx.	24	42,6 %
Tablet Asus	5.0 Mpx.	25	48,0 %

similares para las diferentes resoluciones, logrando un promedio de 45 %, inferior a las condiciones de iluminación diurna. Esto indica que otro de los aspectos importantes de la aplicación se debe a las capturas en estas condiciones, donde los resultados quedaron sujetos a la iluminación que se encuentra en el área del mojón, siendo muy buenos en lugares como "Teatro Municipal" (mojón 10), y no tanto en "Museo y Convento de San Francisco" (mojón 05), como se muestra en la Figura 4.1.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Figura 4.1: Experimento con capturas nocturnas realizadas en el mismo horario. (a)Mojón 10. (b)Mojón 5.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

CAPÍTULO 5

Conclusiones y desarrollos futuros

En este capítulo se presentan las conclusiones finales y los aspectos que dan lugar a trabajos futuros a partir de este proyecto.

5.1 Conclusiones finales

En este Proyecto final se presenta el desarrollo de un sistema que utiliza procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo, y con el que se han alcanzado todos los objetivos propuestos. La aplicación reconoce automáticamente las imágenes de los mojones, considerando el circuito turístico *Camino de la Constitución* de la ciudad de Santa Fe y devuelve información multimedial relacionada. Se ha realizado una base de datos de imágenes considerando diferentes condiciones de iluminación, resoluciones de la cámara y dispositivos de captura.

Se han evaluado diferentes alternativas de procesamiento de imágenes para lograr un adecuado procesamiento de las fotografías obtenidas por el usuario, logrando la normalización de las mismas y la identificación de zonas relevantes para la identificación de los mojones.

Se evaluaron diferentes técnicas de extracción de características y clasificadores, con el fin de intentar proporcionar al usuario resultados correctos y con bajo costo computacional.

Se diseñó una interfaz de usuario, utilizando componentes sencillos de Android,

para lograr una aplicación simple e intuitiva. Finalmente, se ensambló el sistema considerando los mejores resultados obtenidos en la etapa de desarrollo-evaluación para lograr una aplicación potente y veloz.

5.2 Desarrollos futuros

Con el fin de mejorar el desempeño general de la aplicación, se proponen los siguientes trabajos futuros:

- Mejorar el método de procesamiento para lograr una mayor tasa de segmentaciones correctas, y considerando imágenes con perspectiva.
- Hacer una evaluación más profunda del comportamiento del sistema con imágenes nocturnas, inclusive considerando iluminación dedicada en las capturas.
- Involucrar a la Sub-secretaría de Turismo de la Ciudad de Santa Fe en el desarrollo de una versión mejorada de la aplicación.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Bibliografía

[1] Z. Yovcheva, D. Buhalis, and C. Gatzidis. Overview of smartphone augmented reality applications for tourism. *e-Review of Tourism Research*, 10(2):63–66, 2012.

[2] Ismail N. Yewguan S. Muzzammil K., Anuar A. Real-time video processing using native programming on android platform. *IEEE 8th International Colloquium on Signal Processing and its Applications*, 1(1):276–281, 2012.

[3] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 125–134, Washington, DC, USA, 2008.

[4] Nisarg Gandhewar and Rahila Sheikh. Google Android: An emerging software platform for mobile devices. *International Journal on Computer Science and Engineering*, 1(1):12–17, 2010.

[5] George Shih, Paras Lakhani, and Paul Nagy. Is android or iPhone the platform for innovation in imaging informatics. *Journal of Digital Imaging*, 23(1):2–7, February 2010.

[6] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, 2008.

[7] Daniel Wagner, Thomas Pintaric, Florian Ledermann, and Dieter Schmalstieg. Towards massively multi-user augmented reality on handheld devices. In Hans-W. Gellersen, Roy Want, and Albrecht Schmidt, editors, *Pervasive Computing*, number 3468 in Lecture Notes in Computer Science, pages 208–219. Springer Berlin Heidelberg, January 2005.

[8] M. Butler. Android: Changing the mobile landscape. *IEEE Pervasive Computing*, 10(1):4–7, 2011.

[9] Ben Close, John Donoghue, John Squires, Phillip De Bondi, Michael Morris, Wayne Piekarski, Bruce Thomas, Bruce Thomas, and Unisa Edu Au. Arquake: An outdoor/indoor augmented reality first person application. pages 139–146, 2000.

[10] A.M. Bernardos and J.R. Casar. Analyzing business models for mobile augmented reality. In *Intelligence in Next Generation Networks (ICIN), 2011 15th International Conference on*, pages 97–102, 2011.

[11] Lester Madden. *Professional Augmented Reality Browsers for Smartphones: Programming for junaio, Layar and Wikitude*. John Wiley & Sons, May 2011.

[12] S.J. Vaughan-Nichols. Augmented reality: No longer a novelty? *Computer*, 42(12):19–22, 2009.

[13] Hossein R. Babaei, Pagiel L. Mohurutshe, and Arash Habibi Lashkari. Image-processing with augmented reality (AR). volume 8768, pages 87684G–87684G–5, 2013.

[14] Jeff McAffer, Jean-Michel Lemieux, and Chris Aniszczyk. *Eclipse Rich Client Platform*. Addison-Wesley Professional, 2nd edition, 2010.

[15] Tim Boudreau, Jesse Glick, Simeon Greene, Vaughn Spurlin, and Jack J. Woehr. *NetBeans: The Definitive Guide*. O'Reilly Media, Inc., March 2012.

[16] J. F. DiMarzio. *ANDROID A PROGRAMMERS GUIDE*. McGraw Hill Professional, July 2008.

[17] Marko Gargenta. *Learning Android: [building applications for the market]*. O'Reilly, Beijing [u.a., 2011.

[18] Rick Rogers, John Lombardo, Zigurd Mednieks, and Blake Meike. *Android Application Development: Programming with the Google SDK*. O'Reilly Media, Inc., 1st edition, 2009.

[19] Chris Haseman and Safari Tech Books Online. *Android essentials*. Berkeley, CA; New York, NY, 2008.

[20] Sayed Hashimi, Satya Komatineni, and Dave MacLean. *Pro Android 2*. March 2010.

[21] J.G. de Jalón de la Fuente. *Aprenda Java como si estuviera en primero*. Aprenda ..., como si estuviera en primero. Universidad de Navarra. Escuela Superior de Ingenieros Industriales, 1999.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

[22] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (2nd Edition)*. Prentice Hall, January 2002.

[23] R.M. Haralick, Stanley R. Sternberg, and Xinhua Zhuang. Image analysis using mathematical morphology. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-9(4):532–550, 1987.

[24] S. Chen and R.M. Haralick. Recursive erosion, dilation, opening, and closing transforms. *Image Processing, IEEE Transactions on*, 4(3):335–345, 1995.

[25] Zolqernine Othman, Habibollah Haron, Mohammed Rafiq Abdul Kadir, and Mohammed Rafiq. Comparison of canny and sobel edge detection in mri images. 2009.

[26] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, 1986.

[27] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.

[28] Costantino Grana, Daniele Borghesani, and Rita Cucchiara. Connected component labeling techniques on modern architectures. In Pasquale Foggia, Carlo Sansone, and Mario Vento, editors, *Image Analysis and Processing ICIAP 2009*, number 5716 in Lecture Notes in Computer Science, pages 816–824. January 2009.

[29] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

[30] Hao Zhang, A.C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE.*, volume 2, pages 2126–2136, 2006.

[31] Ray Smith. An overview of the Tesseract OCR engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. 9th International Conference on*, volume 2, pages 629–633. IEEE, 2007.

[32] Mark L. Murphy. *Android Programming Tutorials*. CommonsWare, LLC, 3rd edition, 2010.

[33] Paul J. Deitel, Harvey M. Deitel, Abbey Deitel, and Michael Morgano. *Android for Programmers: An App-Driven Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 1st edition, 2011.

[34] Corinne Hoisington. *Android Boot Camp for Developers using Java, Comprehensive: A Beginners Guide to Creating Your First Android Apps, 1st ed.* Cengage Learning.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

APÉNDICE A

Código fuente de la interfaz de usuario

En este capítulo se presenta detalladamente el código fuente desarrollado para la interfaz de usuario de la aplicación.

A.1 Código XML pantalla inicial

En la Figura 3.19 se muestra la pantalla inicial del sistema y en el código A.1 se puede ver claramente cada componente.

El primer elemento que se puede ver es un LinearLayout. Los layout son elementos no visibles que determinan cómo se distribuyen en el espacio los controles que incluímos en su interior, más específicamente, un LinearLayout distribuye los controles simplemente uno debajo de otro siguiendo la orientación que se indique en **android:orientation**, que como se puede observar, en este caso es vertical. En el layout se incluyeron 3 controles: una etiqueta (TextView), un botón (Button) y otro tipo de botón (ImageButton). En todos ellos hemos establecido las siguientes propiedades: **android:id**. El ID del elemento, se utiliza para identificarlo en el código Java. Se puede ver que el identificador va precedido de **@+id/**. Esto hace que cuando se compila el proyecto se genere automáticamente una nueva constante en la clase **R** para dicho elemento. De esta forma, por ejemplo, como el botón tiene asignado el ID BtnIngresar, se puede acceder al él desde el código Java haciendo referencia a la constante **R.id.BtnIngresar**. **android:layout_height** y **android:layout_width**: son las dimensiones del elemento con respecto al la-

yout que lo contiene. Se le puede asignar la propiedad **wrap_content** para indicar que las dimensiones del control se deben ajustar al contenido del mismo, o bien **match_parent** para indicar que el ancho o el alto del control se debe ajustar al ancho o alto del layout contenedor respectivamente. Por último, en los elementos se definió la propiedad **android:text**, que indica el texto que aparece en el elemento. Hay dos alternativas para hacer ésto. La primera, es especificar el valor de la propiedad **android:text**. En la segunda, se debe utilizar alguna de las cadenas de texto (Strings) definidas como recurso del proyecto. En el fichero **strings.xml**, se encuentran todos los Strings definidos para la aplicación, entonces se debe indicar como valor de la propiedad **android:text** el identificador del String precedido del prefijo **@string/**. En otras palabras, podemos definir el texto directamente sobre la propiedad o hacer referencia a un String almacenado como recurso de la aplicación [19]. La segunda opción permite tener localizadas todas las cadenas de texto que se utilizan en el proyecto, lo que podría facilitar por ejemplo la traducción de la aplicación a otro idioma.

Cod. A.1: Código XML pantalla inicial.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android=
3        "http://schemas.android.com/apk/res/android"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:background="drawable/fondo"
7      android:orientation="vertical">
8
9      <TextView
10         android:id="+id/bienvenido"
11         android:layout_width="fill_parent"
12         android:layout_height="wrap_content"
13         android:text="string/bienvenido" />
14         <Button
15             android:id="+id/BtnIngresar"
16             android:layout_width="match_parent"
17             android:layout_height="wrap_content"
18             android:text="string/ingresar" />
19         <ImageButton
20             android:id="+id/BtnAyuda"
21             android:layout_width="match_parent"
22             android:layout_height="wrap_content"
23             android:contentDescription="icono_ayuda"
24             android:background="android:color/transparent"
25             android:src="drawable/ayuda"/>
26 </LinearLayout>
```

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

A.2 Código Java pantalla inicial

En el código A.2, se presenta el código Java de la pantalla inicial que permite tener acceso a la cámara y a la ayuda de la aplicación.

En la línea 6, se le indica a Android que el contenido de este archivo se tiene que volcar en el layout pinicial.xml. Recordemos que el fichero **R** que genera automáticamente la aplicación Android, permite relacionar el código Java con los XML del proyecto, es decir, relaciona el diseño con la lógica. En la línea 8 se obtiene una referencia al botón ingresar definido en el XML y se lo asigna a la variable btnIngresar, a la que luego se le implementará el evento **onClick** del botón, en las líneas 12 a 20. El contenido importante de este evento se encuentra en las líneas 15 y 18, donde se crea y se inicia un Intent, que es el que permite la comunicación entre los distintos componentes del sistema, en este caso, iniciar la actividad, **MainActivity.class**, que es donde se inicializa la cámara. De manera análoga, se implementó el evento onClick para el botón de ayuda.

Cod. A.2: Código Java pantalla inicial de la interfaz de usuario.

```
1   public class pInicial extends Activity {
2
3   Override
4   public void onCreate(Bundle savedInstanceState) {
5   super.onCreate(savedInstanceState);
6         setContentView(R.layout.pinicial);
7   //Obtenemos una referencia a los controles de la interfaz
8   Button btnIngresar = (Button)findViewById(R.id.BtnIngresar);
9   ImageButton btnAyuda=
10                (ImageButton)findViewById(R.id.BtnAyuda);
11  //Implementamos el evento click del boton
12  btnIngresar.setOnClickListener(new OnClickListener() {
13  public void onClick(View v) {
14  //Creamos el Intent
15  Intent intent =new Intent(pInicial.this,MainActivity.class);
16
17  //Iniciamos la nueva actividad
18  startActivity(intent);
19    }
20    });
21  }
22  }
```

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

A.3 Código de inicialización de la cámara con OpenCV

Una vez que se ingresa a la aplicación se accede a la cámara, que es inicializada con funciones de OpenCV para poder realizar el procesamiento correspondiente. El código A.3 permite dicha inicialización.

Cod. A.3: Inicia la cámara utilizando OpenCV.

```
1  private BaseLoaderCallback  mOpenCVCallBack =
2    new BaseLoaderCallback(this) {
3    Override
4    public void onManagerConnected(int status) {
5    switch (status) {
6    case LoaderCallbackInterface.SUCCESS:
7        {
8        Log.i(TAG, "OpenCV loaded successfully");
9        // Create and set View
10       mView = new ImageManipulationsView(mAppContext);
11       setContentView(mView);
12
13       // Check native OpenCV camera
14       if( !mView.openCamera() ) {
15        AlertDialog ad =
16           new AlertDialog.Builder(mAppContext).create();
17        ad.setCancelable(false);
18        ad.setMessage("Fatal error: can't open camera!");
19        ad.setButton("OK",new DialogInterface.OnClickListener()
20        {
21         public void onClick(DialogInterface dialog,int which){
22         dialog.dismiss();
23         finish();
24       }
25       });
26       ad.show();
27
28       } break;
29       default:
30       {
31       super.onManagerConnected(status);
32       } break;
33     }
34   }
35   };
```

Como se puede ver, se crea una instancia **mView** de la clase **ImageManipulationView.java** (clase que contiene los métodos de procesamiento de imágenes), y luego asigna este contenido a la pantalla en la línea 11. Esta clase, a su vez, hereda todos los métodos de **GestionaCamara.java**, que es donde están definidos los métodos para poder gestionar la cámara del dispositivo utilizando la OpenCV, entre ellos **openCamera()**, que es llamado desde el objeto **mView**. Luego, en la línea 14 se verifica si la cámara se ha abierto correctamente y en caso negativo se lanza un mensaje de alerta utilizando el método **AlertDialog**. En la Figura 3.20 se puede ver la pantalla que permite capturar una fotografía del mojón para su análisis y clasificación. Mediante un Canvas se definió un marco y se presenta una etiqueta que ayudan al usuario a enfocar el mojón para tomar una fotografía.

A.4 Código del menú para realizar captura y ver resultado

En la Figura 3.20 se pueden ver dos botones en la parte inferior, uno permite "tomar la fotografía y realizar la detección" y el otro "muestra el resultado".

El código Java que permite la creación del menú se muestra en el código A.4. En el método **onCreateOptionMenu()**, se añaden las opciones que se desean visualizar utilizando el método **add()** sobre el objeto Menú que llega como parámetro de la función. El mismo recibe 4 parámetros, entre ellos el texto de la opción, que es el único que se utilizó en el menú de la aplicación. La implementación de cada una de las opciones del menú se incluye en la función **onOptionsItemSelected()** de la actividad que mostrará el menú. Esta función recibe como parámetro el item de menú que presionó el usuario.

Cod. A.4: Código Java del menú.

```
1     public boolean onCreateOptionsMenu(Menu menu) {
2         Log.i(TAG, "onCreateOptionsMenu");
3         mItemModeResultado  = menu.add("Mostrar resultado");
4         mItemModeProcess  = menu.add("Capturar frame");
5         return true;
6     }
7
8     public boolean onOptionsItemSelected(MenuItem item) {
9         Log.i(TAG, "Menu Item selected " + item);
10
11        if (item == mItemModeResultado){
12         synchronized (mView) {
13        if(mView.getFinalizado()){
```

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

```
14              num= mView.getNumero();
15              onCreateDialog(num);
16              mView.setFinalizado();
17            }
18           }
19          }
20          else if (item == mItemModeProcess){
21            viewMode = MODE_PROCESS;
22            mView.setFinalizado();
23            clickProgressCirculo();
24            viewMode= VIEW_MODE_RGBA;
25          }
26          return true;
27      }
```

A.5 Código del cuadro de diálogo para informar el resultado.

A continuación se puede ver el código Java del cartel informativo que indica cual es el mojón identificado. El resultado se puede ver en la Figura 3.21.

El cartel informativo se generó con el código A.5.

Cod. A.5: Código Java cartel informativo.

```
1  private Dialog onCreateDialog(Integer n)
2    {
3    AlertDialog.Builder builder = new AlertDialog.Builder(this);
4
5  switch (n) {
6    case 1:
7      builder.setTitle("Bienvenido a Plaza 25 de Mayo " +
8         n.toString() );
9      break;
10   case 2:
11     builder.setTitle("Bienvenido a Casa del Brigadier
12     Estanislao L\'opez "+ n.toString() );
13     break;
14     .
15     .
16     .
17   case 17:
18       builder.setTitle("Bienvenido a Facultad de Ciencias
```

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

```
19        Jurídicas y Sociales" + n.toString() );
20    }
21
22    builder.setMessage("¿Desea ver información relacionada?");
23    builder.setPositiveButton(
24      "Si",  new DialogInterface.OnClickListener() {
25
26    public void onClick(DialogInterface dialog, int which) {
27    Log.i("Dialogos", "Confirmacion Aceptada.");
28
29    //Creamos el Intent
30    Intent intent =new Intent(MainActivity.this,listaInfo.class);
31    intent.putExtra("numero", num);
32    //Iniciamos la nueva actividad
33    startActivity(intent);
34    dialog.cancel();
35    }
36      });
37
38    builder.setNegativeButton(
39     "Volver a la cámara",new DialogInterface.OnClickListener(){
40     public void onClick(DialogInterface dialog, int which) {
41    Log.i("Dialogos", "Confirmacion Cancelada.");
42
43    viewMode = VIEW_MODE_RGBA;
44    dialog.cancel();
45    }
46    });
47    builder.show();
48    return builder.create();
49  }
50
```

El método **onCreateDialog()** recibe como parámetro el ID del diálogo que se quiere crear. Se puede asignar cualquier ID al cuadro de diálogo, pero deben identificar de forma única a cada diálogo de la aplicación como muestra el código A.5. En este caso se tomó como ID para cada cuadro de diálogo el identificador único de cada cartel, de manera de mostrar un texto de que imagen se ha identificado luego de la clasificación. Existen diferentes tipos de cuadro de diálogo, en este caso se eligió el "Diálogo de confirmación" que permite mostrar un mensaje sencillo y solicitar al usuario la confirmación de una acción o no. En este caso, el usuario puede ver información relacionada al mojón reconocido o volver a la cámara para realizar otra captura. Este tipo de diálogos se construyen mediante la **claseAlertDialog**, más específicamente con la subclase **AlertDialog.Builder**. Para la construcción se creó un objeto de tipo **AlertDialog.Builder** y se establecieron las propiedades del diálogo mediante sus métodos correspondientes: título [**setTitle()**], mensaje [**setMessage()**] y comportamiento del botón [**setPositi-**

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

veButton()], como se muestra en las líneas 7 a 24 del código A.5. Dentro de las función **setPositiveButton()** y **setNegativeButton()**, se implementa el evento **onClick()**, que es donde se define que acción se realiza en cada uno de los casos.

A.6 Código reproducción de videos

Para reproducir videos almacenados como recursos de la aplicación, se creo un archivo **video.xml**, con un control **VideoView**. Este control ocupa todo el ancho de la pantalla y allí se visualizará el video (ver código A.6).

Cod. A.6: XML para reproducción de video.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/
3     res/android"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:background="#FFFFFFFF" >
7      <VideoView
8          android:id="+id/videoView_video"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:layout_alignParentBottom="true"
12         android:layout_alignParentTop="true"
13         android:layout_centerInParent="true" />
14 </RelativeLayout>
```

Luego de realizar la codificación del XML se procedió al desarrollo de la parte en Java, el código que permite la reproducción del video se muestra en A.7.

Cod. A.7: Código Java para reproducción de video.

```
1  public class VideoPlayer extends Activity {
2
3   Override
4   public void onCreate(Bundle savedInstanceState) {
5     super.onCreate(savedInstanceState);
6     setContentView(R.layout.video);
7     VideoView videoView =
8        (VideoView) findViewById(R.id.videoView_video);
9     Uri path = Uri.parse(
10       "android.resource://com.example.captureframe/" +
11       R.raw.cc);
12    MediaController mc = new MediaController(this);
```

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

```
13      videoView.setMediaController(mc);
14      videoView.setVideoURI(path);
15      videoView.start();
16    }
17  }
```

En la línea 6 se indica a Android que el contenido de esta actividad se va a asignar al layout **video.xml**. En la línea 7 se obtiene una referencia al control definido en el XML, y se guarda en el objeto **videoView**. Para poder reproducir un video de forma local, primero se debe almacenar el video como recurso de la aplicación. Todos los recursos de la misma se encuentran en la carpeta res del proyecto, luego se debe crear una carpeta llamada **raw/** dentro de **res/** y guardar el video a reproducir en **res/raw**. Una vez realizado este paso, se guardó el path en una variable con este mismo nombre. En la línea 12, se le asigna al objeto **videoView** un **mediaController**, que es un componente que proporciona Android para que el usuario pueda pausar, adelantar y retroceder la reproducción, es decir, un control de reproducción. Por último, en la línea 14 y 15 se asigna el path y se inicia la reproducción. De esta forma se consigue que al iniciar la actividad el video empiece a reproducirse.

A.7 Código XML de acceso a sitios web, galería y videos en línea

A continuación se muestra el código A.8 que permite la reproducción de dicho link:

Cod. A.8: Código XML para reproducción de link web.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/
3     res/android"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:orientation="vertical" >
7      <WebView
8          android:id="+id/webView1"
9          android:layout_width="match_parent"
10         android:layout_height="match_parent" />
11 </LinearLayout>
```

Como se puede ver, tiene definido el componente **WebView** que provee Android para la visualización de este tipo de componentes. Dentro del mismo, se definieron

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

las propiedades **layout_width** y **layout_height** con el valor **"match_parent"** para ajustar el ancho y el alto a los del layout que contiene el componente.

A.8 Código Java para el desarrollo de un spinner

Para poder comprender como funciona este control de selección, primero tenemos que explicar que hace y como funciona un Adaptador (**Adapter**). Un adaptador es como una interfaz o punto de acceso a los datos que existen atrás del control de selección, en el caso de la presente aplicación, un **Spinner**. Dicho de otro modo, los controles de selección pueden acceder a los datos que contienen a través de un adaptador. Además de proveer los datos a los controles visuales, el adaptador también es responsable de generar a partir de estos datos las vistas específicas que se mostrarán dentro del control de selección. Por ejemplo, si cada elemento de una lista está formado a su vez por una imagen y varias etiquetas, el responsable de generar y establecer el contenido de todos estos "sub-elementos" a partir de los datos será el propio adaptador. Android tiene varios tipos de adaptadores, sin embargo, aquí se explica solo el que se utilizó que es **ArrayAdapter**. Es el más sencillo de todos los adaptadores, y provee de datos a un control de selección a partir de un array de objetos de cualquier tipo.

La forma en que se crea un adaptador se muestra en el código A.9. En la línea 1 definimos un arreglo en Java con los datos a mostrar en el control de selección. En la siguiente línea creamos el adaptador, al que le pasamos 3 parámetros:

1. El contexto. Normalmente es una referencia a la actividad donde se crea el adaptador.

2. El ID del layout sobre el que se muestran los datos del control. En este caso se le pasó el ID de un layout predefinido de Android, **(android.R.layout.simple_spinner_item)**, formado únicamente por un control **TextView**, pero se puede pasar el ID de cualquier layout del proyecto con cualquier estructura y conjunto de controles al que se quiera aplicar.

3. El array que contiene los datos a mostrar.

Cod. A.9: Adaptador de Android.

```
1 final String[] datos =
2  new String[]{"01. Santa Fe y los orígenes del Estado.",
3  "02. La Plaza de Mayo   y su contorno."};
4
```

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

```
5   ArrayAdapter<String> adaptador =
6   new ArrayAdapter<String>(this,
7   android.R.layout.simple_spinner_item, datos);
```

Con ésto se creó el adaptador para los datos a mostrar y sólo queda asignar este adaptador a nuestro control de selección para que el mismo muestre los datos en la aplicación. Para unir los datos del adaptador al **Spinner**, se utilizó el código A.10.

En la línea 1 se obtiene una referencia del control a través de su ID. En la línea 2 se asignó el ID del layout. Para el caso del control **Spinner**, este layout solo se aplica al elemento seleccionado de la lista, en este caso se utilizó uno predefinido en Android para las listas desplegables **(android.R.layout. simple_spinner_dropdown_item)**, formado por una etiqueta de texto con la descripción de la opción y un marcador circular a la derecha que indica si la opción está o no seleccionada. Por último, en la línea 4 se asigna el adaptador al control mediante el método **setAdapter()**. El resultado se puede observar en la Figura 3.23

Cod. A.10: Código Java de unión entre un adaptador y el Spinner.

```
1     cmbOpciones2 = (Spinner)findViewById(R.id.CmbOpciones3);
2     adaptador.setDropDownViewResource(
3     android.R.layout.simple_spinner_dropdown_item);
4     cmbOpciones2.setAdapter(adaptador);
```

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

APÉNDICE B

Instalación y configuración del entorno de desarrollo

En este capítulo se pretende explicar la configuración del entorno de desarrollo para poder programar aplicaciones Android que incluyan procesamiento de imágenes con la biblioteca OpenCV. Primero, se describe el proceso de instalación de las herramientas de desarrollo. Luego, la integración de la biblioteca OpenCV con Android y por último la forma de ejecutar una aplicación Android que contiene funciones de OpenCV.

B.1 Instalación de herramientas de desarrollo

Lo primero que hay que instalar es el Sun Java Development Kit (JDK) 7 que se puede descargar desde el sitio oficial de Oracle[1]. Luego se debe descargar el SDK de Android del sitio oficial de Android[2]. Los componentes que se precisan para que la biblioteca OpenCV funcione correctamente son (Figura B.1):

- Android SDK Tools, versión 20 o superior.
- Plataforma del SDK de Android 2.2 (API 8) o superior.

Si bien la plataforma mínima del SDK de Android para que la OpenCV Java API funcione correctamente es la 2.2, para que dicha biblioteca funcione en óptimas

[1]http://www.oracle.com/technetwork/java/javase/downloads/index.html
[2]http://developer.android.com/sdk/index.html

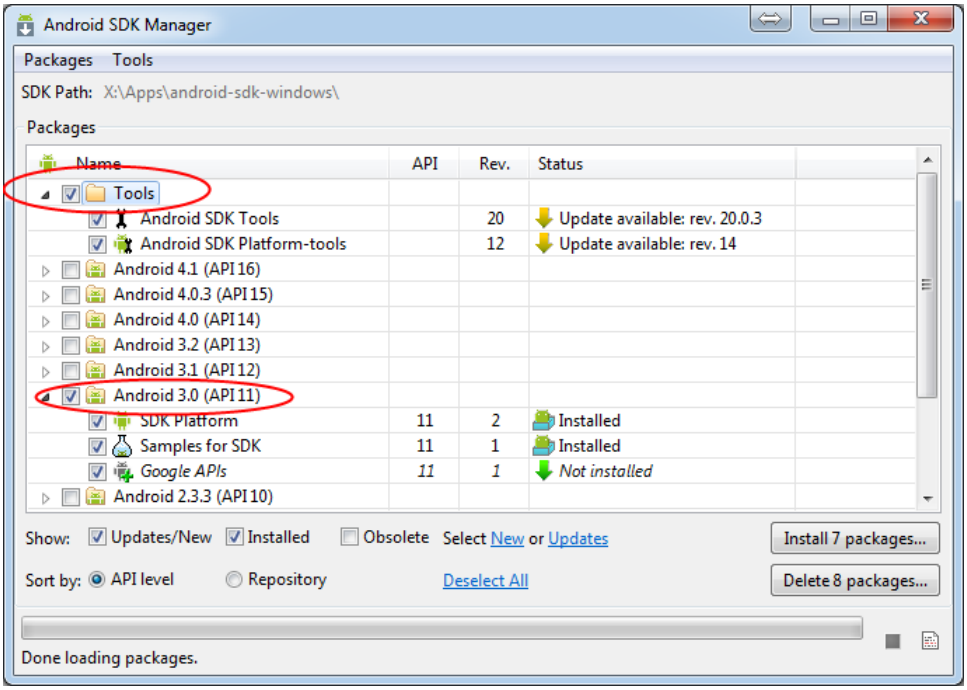

Figura B.1: Componentes necesarios del SDK de Android.

condiciones se requiere la plataforma 3.0 (API 11) o superior. La configuración del dispositivo objetivo se realiza en el archivo **AndroidManifest.xml** del proyecto Android.

Se decidió instalar la versión Juno de Eclipse dado que es la mas nueva y estable que posee dicho IDE, la misma se puede descargar de forma gratuita del sitio oficial de Eclipse[3]. Para poder programar en Android sobre Eclipse se necesita instalar el Plugin ADT.

ADT es un plugin para Eclipse que proporciona un conjunto de herramientas que se integran con el IDE de Eclipse. Ofrece acceso a muchas características que ayudan a desarrollar aplicaciones Android rápidamente. Contiene herramientas de diseño de interfaz de usuario para creación rápida de prototipos. Debido a que ADT es un plugin para Eclipse, se obtiene la funcionalidad de un IDE robusto. A continuación se describen las características importantes de ADT en Eclipse:

- Permite la comprobación de sintaxis en tiempo de ejecución, y la documentación integrada para la API de Android.
- Proporciona un editor XML personalizado que permite trabajar sobre archivos XML en una interfaz de usuario basada en formularios de la aplicación

[3]http://www.eclipse.org/

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

Android. Un editor de diseño gráfico permite diseñar interfaces de usuario con solo arrastrar y soltar.

Para instalar el plugin en Eclipse se deben seguir los siguientes pasos:

- Iniciar **Eclipse**, ir a la solapa **Ayuda** y luego a **Install new software**. Luego, aparecerá una ventana como la de la FiguraB.2.
- Hacer click en **Add** (Agregar), esquina superior derecha.
- En la ventana emergente **Add Repository** hay que ingresar **ADT Plugin** para el nombre y la siguiente URL en location **https://dl-ssl.google.com/android/eclipse/**. Luego, clickear **OK**.
- Una vez que aparecen los complementos, seleccionar el checkbox que dice **Developer tools** (herramientas de desarrollo) y hacer click en **siguiente**.
- Una vez que finaliza la descargar e instalación del plugin, reiniciar **Eclipse**.

B.2 Integración de biblioteca OpenCV

Primero, se debe descargar la biblioteca OpenCV [4] y colocarla en algún directorio, en este caso se creó el siguiente: **C:/Work/ OpenCV4Android/**. La versión que se utilizó en este proyecto es la 4.2.4 y la misma se debe descomprimir en el directorio creado anteriormente.

El próximo paso consiste en importar la biblioteca en Eclipse. Para esto se debe iniciar el IDE, y seleccionar como espacio de trabajo el directorio creado con anterioridad.

Hacer click derecho en la ventana del **package explorer** (explorador de proyectos en el espacio de trabajo) y seleccionar **import**.

En el panel principal seleccionar **General**, **Existing Android Code into workspace** (Proyectos existentes en el espacio de trabajo) y presionar el botón **siguiente**.

En **Root directory** (Seleccionar ruta del directorio), ingresar la ruta del path donde se encuentra la biblioteca descomprimida. Eclipse cargará automáticamente la biblioteca y los proyectos que se encuentran como ejemplos en la misma carpeta. Hacer click en **finalizar** para completar el proceso de importación.

[4] http://sourceforge.net/projects/opencvlibrary/files/opencv-android/

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

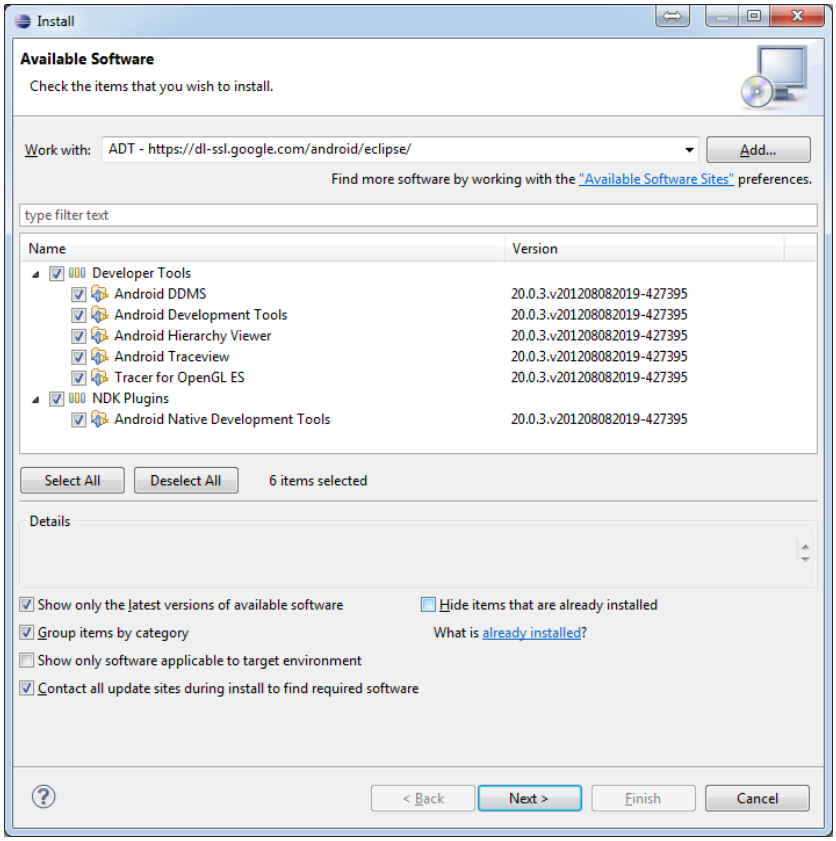

Figura B.2: Instalación de plugin ADT.

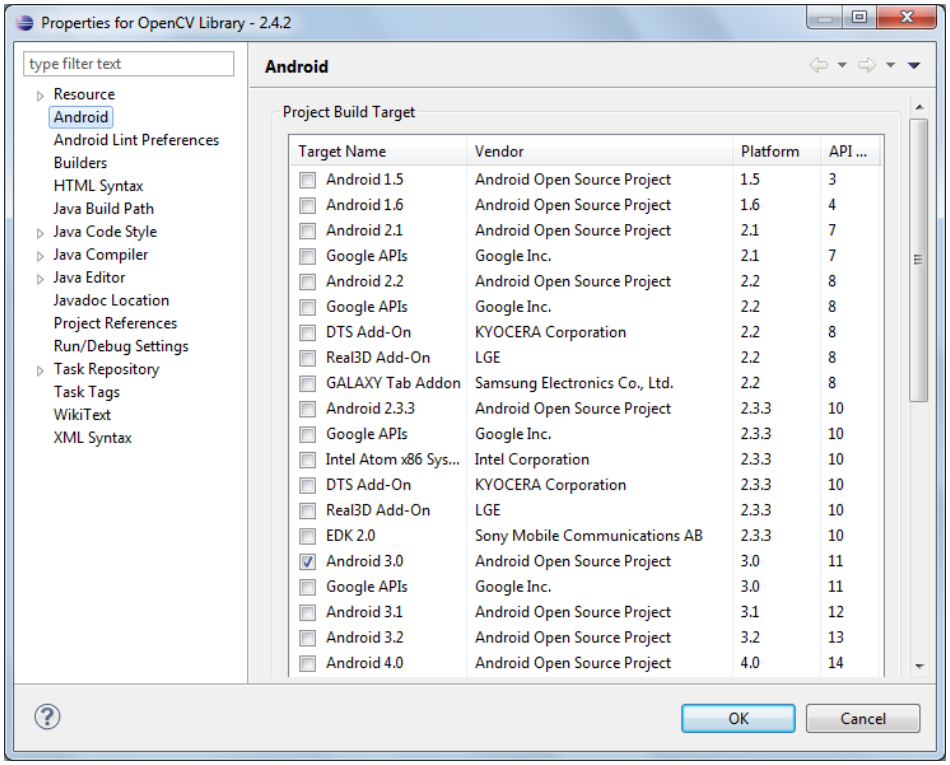

Figura B.3: Corrección de error en biblioteca OpenCV.

Puede que una vez importados la biblioteca y los ejemplos en el espacio de trabajo, estos marquen errores. Para solucionar este inconveniente, se debe hacer click derecho en la biblioteca OpenCV y luego en **Propiedades**. Aparecerá una ventana como la de la Figura B.3. Hacer click en la solapa **Android** del lado derecho de la misma, luego en el checkbox **Android 3.0 (API 11)** y por último en **Ok**. Hacer lo mismo para los proyectos que se deseen ejecutar.

En caso de que sigan apareciendo errores hacer click derecho en el **proyecto**, **herramientas de Android** (Android tools) y seleccionar **corregir propiedades del proyecto** (Fix projects properties).

Ahora todos los proyectos Android que utilicen la biblioteca OpenCV deben aparecer sin errores, excepto aquellos que contengan código nativo en C++. Estos últimos se deben compilar de una manera especial que no se explica en el presente apéndice porque este tipo de compilación no se utilizó en la aplicación.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

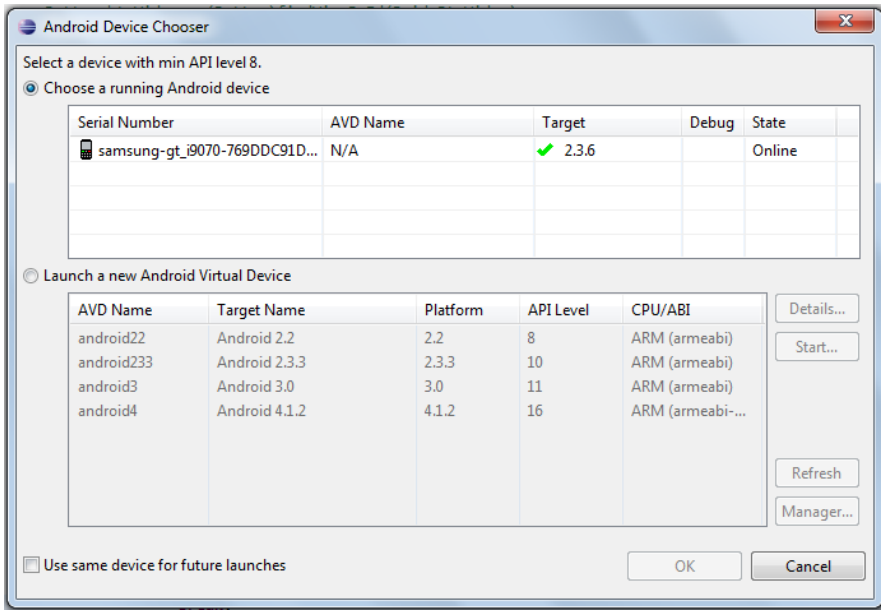

Figura B.4: Ejecutar aplicación Android.

B.3 Ejecución del proyecto Android

En Eclipse, un proyecto se puede ejecutar sobre un emulador o sobre un dispositivo físico. Trabajar directamente sobre el dispositivo móvil es mejor, dado que las aplicaciones se compilan más rápidamente. Para ejecutar un proyecto Android se debe ir al botón **ejecutar**, **Run As**(correr como) y seleccionar **Android Application**. Si se ha conectado un dispositivo, se va a visualizar una pantalla como la de la Figura B.4.

En esta pantalla se puede seleccionar un emulador creado previamente o el dispositivo móvil conectado a la computadora.

La primera vez que se ejecuta la aplicación Android con funciones de OpenCV, aparecerá un cuadro de diálogo que dice que no está instalado el paquete de OpenCV Manager[5] en el dispositivo. Si se tiene acceso a Google Play desde el dispositivo se debe seleccionar **Si** y se descargará el paquete desde el Market.

En caso de no trabajar con el dispositivo móvil, desde el emulador no se tiene acceso al Market de Android, y por lo tanto, se debe instalar el paquete desde el directorio donde se encuentra instalada la biblioteca OpenCV. Para esto, se debe

[5]OpenCV Manager se encarga de seleccionar la mejor versión de OpenCV para el hardware del dispositivo.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.

ejecutar la siguiente línea de comandos en la consola de windows o linux:

```
1 <Android SDK path>/platform-tools/adb install <OpenCV4Android
2 SDK path>/apk/OpenCV_2.4.6_Manager_2.8_armv7a-neon.apk
```

Con esto se concluye la instalación y configuración del entorno de desarrollo para aplicaciones Android con OpenCV.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
P. Sosa, E. M. Albornoz & C. E. Martínez; "Desarrollo de un software de procesamiento digital de imágenes para dispositivos móviles con aplicaciones en turismo (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2013.