Creación de una imagen con una secuencia de movimientos a partir de un vídeo para tablets Android.

Cristian Rodríguez

Resum— El propósito de este articulo, es demostrar como se puede plasmar una secuencia combinada de movimientos en una única imagen en Android. Se muestra tanto la metodología utilizada como las ideas principales iniciales y finales que han surgido a lo largo del proyecto. Se detalla la información más relevante del proyecto, que se centra en la idea principal, la metodología, el desarrollo y los resultado obtenidos. Principalmente se quiere obtener como resultado una imagen donde se pueda apreciar el movimiento del objeto o persona en cuestión en una imagen. También se muestran los problemas encontrados en el proceso de alineación de las imágenes para que corresponda el marco visual.

Paraules clau—Android, Imagenes, JNI, matcher, NDK, OpenCv, Sequencia.

Abstract— The purpose of this article is to demonstrate how to display a sequence of movements in a single image in android. This works shows both methodology used and the main initial and final ideas emerged douring the project which led the results shown in this document. It also details the most relevant project information, where the main idea, the methodology, the development and the results obtained are explained. The main objective, as a resault, is to get a single image where the user can see the movement of an object in a series of shots. It's also explained the aligning problems found when more tan two pictures must be overlayed to fit the framework of the image.

Index Terms— Android, images, JNI, matcher, NDK, OpenCv, sequence.

1 Introducción

El objetivo principal de este proyecto es conseguir plasmar el movimiento de una persona o un objeto en una imagen a partir de un video o una secuencia de imágenes. El motivo de realizar este trabajo, es el de aplicar los conocimientos adquiridos en la carrera sobre el tratamiento de imágenes y la visión artificial.
El problema radica en crear un algoritmo que permita la superposición de imágenes sin solaparse donde se pueda observar el movimiento de la persona u/o objeto.
Los objetivos principales de este proyecto es conseguir hacer una aplicación para una Tablet Android que permita seleccionar fotos de la galería, grabar o seleccionar un video de la galería y procesar las imágenes correctamente. Todas las fotos o videos tomados relacionados con el objetivo anterior se tiene que hacer con trípode para que las imágenes encajen a la perfección. En este documento también se explicara el inconveniente que supone hacer las fotografía sin trípode y los resultados diferentes que podemos obtener dependiendo del modo que realicemos la captura de imágenes.

- *E-mail de contacte: pitus660@gmail.com*
- *Menció realitzada: Computació*
- *Treball tutoritzat per: Francisco Javier Sánchez Pujadas (CVC)*
- *Curs 2013/14*

El documento contiene los siguientes apartados:
Estado del Arte, metodología, planificación, superposición de las imágenes, implementación en Android, alineación de imágenes, resultados, utilización de la aplicación, trabajos futuros y conclusiones. También se incluye un apartado de agradecimientos y la bibliografía que se ha utilizado para hacer el proyecto.

2 Estado del arte

En la actualidad existen en el mercado tres aplicaciones que realizan lo mismo. Hablamos de empresas grandes como es Sony, HTC y Samsung, cada una lo hace de una forma distintas pero las tres obtienen resultados parecidos. En el caso de Sony la aplicación en si se llama "Captura de animación"[12] y solo esta disponible para los modelos Sony xperia Z1, Zf y Z ultra. Esta aplicación te permite hacer un video de como máximo 8 segundos y hace un proceso automático de selección de fotogramas que es trasparente al usuario. En el resultado se puede observar el objeto en movimiento enfocado y el fondo desenfocado.
A lo que respecta a la aplicación de HTC, se trata de una aplicación que viene integrada en todos los HTC actuales y te permite realizar este efecto capturando una secuencia de fotos consecutivas.
Por ultimo la aplicación de Samsung exclusiva para el Samsgung Galaxy S4 y Note 3 llamada "Drama shot" [15] el cual permite capturar un video y procesarlo para poder

obtener este efecto. Por ultimo permite editar con el dedo la mascara del objeto creado, seleccionando la área que queremos mostrar.

También existe la posibilidad de realizar esto con la ayuda del programa de edición de fotos: Photoshop, el cual cargando unas imágenes podemos alinearlas de forma automática y hacer la superposición de estas. Para ello ,crea para cada imagen una mascara en la cual solo debería mostrarse la persona/objeto que esta en movimiento. El problema es que tienes que modificar a mano las mascaras para obtener el resultado esperado. Lo mencionado anteriormente implica mucho más tiempo a la hora de generar una secuencia de movimiento, debido a que parte de las modificaciones se han de hacer a mano y no todas las personas tienen dominio/conocimiento de Photoshop. Para realizar las pruebas me he basado en el siguiente tutorial[14].

2 METODOLOGÍA

Para poder llevar a cabo el proyecto he utilizado una metodología ágil como es la metodología en espiral[6].
He elegido esta metodología porque es la que más se adapta al trabajo individual que tengo que realizar, para llevar a cabo el proyecto de final de Grado y cumplir todos los objetivos previsto en el desarrollo de esta aplicación.

A continuación se muestra un resumen de los ciclos que se han realizado en este proyecto:

- **Primer ciclo:**
  Investigar la forma de poder obtener los fotogramas de un video en Android e implementar dicha funcionalidad.

- **Segundo ciclo:**
  Crear una aplicación en el lenguaje mas cómodo y rápido para el procesamiento de imágenes que haga la superposición entre imágenes.

- **Tercer ciclo:**
  Capturar imágenes para poder realizar pruebas de manera correcta y optimizar el código del ciclo anterior para poder obtener mejores resultados

- **Cuarto ciclo:**
  Creación de las mascaras, superposición de todas las imágenes y necesidad de tener el fondo de la escena.

- **Quinto ciclo:**
  Aplicar una función logística a la mascara para obtener mejores resultados.

- **Sexto ciclo:**
  Elección del pixel que tenga la diferencia mas grande respecto al fondo.

- **Séptimo ciclo:**
  Implementación en Android del ciclo anterior.

- **Octavo ciclo:**
  Instalación e implantación de la librería de OpenCV en el proyecto.

- **Noveno ciclo:**
  Preparación de las clases para la utilización de OpenCV

- **Decimo ciclo:**
  Adaptación de OpenCV

- **Undécimo ciclo:**
  Obtención y aplicación de la matriz de transformación

- **Duodécimo:**
  Generación de la matriz de transformación por mínimos cuadrados.

4 PLANIFICACIÓN

En esta sección se muestra la planificación utilizada para poder realizar el desarrollo del proyecto en el tiempo estimado. En la siguiente tabla se puede observar la evolución de la planificación.

Evolución	Inicial	Semana 10	Final
Interfaz	60h	60h	60h
Algoritmo	140h	120h	120h
Unificación	50h	40h	40h
Alineación	-	67h	70h
Documentación	50h	13h	18h
Total	**300h**	**300h**	**308h**

Tabla1. Planificación de las tareas.

Podemos observar como la planificación inicial fue muy ajustada, pero con los distintos problemas surgidos que se mencionan en este documento ha ido adaptándose a las necesidades. En el primer caso no se considero el concepto de alinear las imágenes, seguidamente en la semana 10 al ver que todo iba sobre ruedas se decidió empezar a implementar, por ultimo, en la fase final se pudo implementar todo correctamente a pesar de sobrepasar las horas totales de la planificación inicial.

3 SUPERPOSICIÓN DE LAS IMÁGENES

A continuación se detallan los distintos algoritmos creados para poder llevar a cabo la superposición de imágenes con objetos o personas en movimiento. Para ello se ha utilizado Matlab[4] debido a que es un lenguaje que tiene como punto fuerte el tratamiento de matrices de forma optima. Cabe desatacar que el hecho de realizarlo con

matlab implica poder ver los resultados al instante sin tener que instalar ninguna aplicación y eso es un punto fuerte para poder realizar las pruebas necesarias para encontrar el mejor algoritmo. Por ultimo mencionar la diferencia de tiempos con respecto al algoritmo final desarrollado en C para android, en matlab daba unos tiempos muy elevados para el problema propuesto.

3.1 3 imágenes sin fondo y comparando

La primera idea que se implemento fue la de escoger 3 imágenes y hacer comparaciones entre ellas para decidir que pixel pertenece a la imagen donde esta el movimiento de la persona/objeto , siguiendo la siguiente hipótesis:

> "Si el valor del pixel de la primera imagen es diferente al valor del pixel de la segunda imagen y el valor pixel de la segunda imagen es igual al valor del pixel de la tercera imagen, entonces el pixel bueno es el de la primera imagen porque es donde se encuentra la persona/objeto que esta en movimiento. Así haciendo todas las combinaciones posibles con las tres imágenes para cada valor de pixel se decide que valor de pixel es el bueno y se pone en la misma posición en la imagen final.".

Esta fue la primera solución implementada, pero el resultado no era el esperado, debido a que es muy difícil que el valor de los pixeles sea el mismo. Para solucionar lo anterior se consideraba que dos pixeles eran diferentes si sus diferencias superaban un umbral. Esta idea daba mejor resultado, pero el hecho de definir un umbral a mano, hacia que no siempre se obtuviera buenos resultados. Por este ultimo motivo esta solución fue descartada.

3.2 Creación de una mascara

La siguiente solución implementada fue la creación de una mascara para cada imagen. Para poder crear la mascarara se necesita el fondo común que tienes todas las imágenes, la mascara es la diferencia de los pixeles en valor absoluto de la imagen y el fondo. Esto permite tener todas las mascaras de todas las imágenes y así después añadir a la imagen de fondo todas las imágenes aplicando su correspondiente mascara. Esto supuso una mejorar importante en el momento de seleccionar el objeto que se movía, pero implica tener una imagen de fondo. El mayor problema que este método daba era el hecho de que las imagen resultante no quedaba uniformes, sino que se podía observar distintos parches de las distintas imágenes. También hay que destacar que para seleccionar la mascara se aplicaba una función logística a la substracción de la imagen con el fondo para solo dejar pasar unos ciertos valores y así también definir que cantidad de fondo o imagen queremos mostrar. Esto provoca que las sombras no queden del todo definidas y que en algunas ocasiones se vea como un efecto de cortar y pegar. Para poder realizar la función logística implicaba también definir a mano un umbral para que dado un valor de pixel concreto considerarlo persona/objeto o fondo. Tenemos el mismo problema que en el apartado anterior, definir un umbral a mano. Este es el mismo motivo por el cual la solución fue descartada.

3.1 Selección el pixel que tiene la mayor diferencia respecto al fondo

Por ultimo en este apartado se explica la solución utilizada para este proyecto y con la cual se han obtenido los mejores resultados.
La idea es comparar cada pixel de cada imagen con la imagen de fondo y para ese pixel guardar el valor del pixel el cual tiene la máxima distancia euclidiana con respecto al pixel de la imagen de fondo. Son operaciones mas simples que las soluciones anteriores y siempre dan buenos resultados.

4 IMPLEMENTACIÓN EN ANDROID

Para hacer una aplicación en android se utiliza Java, debido a que Android es un sistema operativo para smartphone el cual pose un maquina virtual que ejecuta aplicaciones hechas en Java. A su vez también permite ejecutar aplicaciones hechas con JNI (Java native interface) que se puede programar en C y en C++ el cual en algunos casos es más eficaz que Java.

4.1 Comparativa C vs Java

En este proyecto he utilizado C únicamente en la unión de las imágenes, el cual es un proceso bastante costoso. Realice distintas pruebas con Java y C recorriendo todos los pixeles de una imagen. En la siguiente figura se puede observar la diferencia de tiempos en hacer distintas operaciones con una imagen de 5MP (2592x1944) en C y en Java.

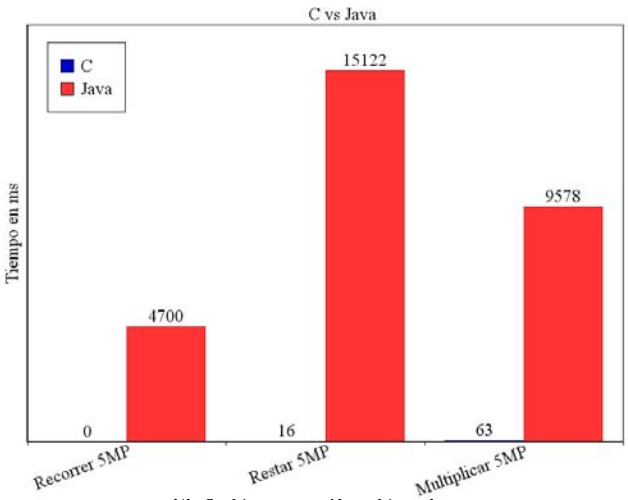


Fig1. Comparativa C vs Java

En la figura anterior se puede observar como el hecho de recorrer las imágenes con punteros desde C es mucho más efectivo que hacerlo desde Java, accediendo a la posición de cada pixel por índices.

4.2 Problemas encontrados

El principal problema encontrado era el hecho de poder ejecutar código nativo desde java. Como poder compilar un archivo c con el NDK[1] de Android de forma sencilla. Con la ayuda de dos libros[2][7] en los cuales se explica de forma eficiente como poder compilar y llamar a una función hecha en C o C++ desde java solucione mi problema.

5 ALINEACIÓN DE IMÁGENES CON OPENCV

5.1 ¿Qué es OpenCV?

OpenCV[5] es una librería libre de visión artificial que esta disponible desde 1999. Desde entonces ha sido utilizada en infinidad de aplicaciones, sistema de seguridad, reconocimiento de objetos y tratamiento de imágenes. Esto es gracias a que ha sido programada en C y C++, se puede utilizar en distintos sistemas operativos tanto de ordenador como para Smartphone. En su pagina web esta disponible la versión más reciente de su SDK, el cual se puede descargar para Windows, Linux/Mac, Android i IOS.

5.1 Instalación de Opencv

OpenCv para android se puede instalar de dos maneras: utilizando el OpenCV Manager o no.
La diferencia básica, es que utilizando el OpenCV manager (es una aplicación para Android que la tienes que tener instalada en el dispositivo) siempre tienes el ultimo SDK de openCV .De la otra manera, tienes que descargarte el SDK y añadirlo a tu proyecto.

Yo he utilizado la segunda opción ,debido a que no veía muy lógico tener que instalar una aplicación de terceros para poder utilizar la mía.
Este proyecto utiliza la versión 2.4.7 (11-11-2013) del SDK de OpenCV. Para incluirlo en mi proyecto me base en el tutorial oficial de OpenCV[13].

5.1 Explicación del algoritmo

Para poder realizar la alineación de las imágenes, debemos seguir una serie de pasos que se muestran a continuación.
Primero utilizaremos la librería OpenCv, explicada en el apartado anterior, para poder alinear las imágenes de forma correcta.
Cada imagen será comparada con el fondo, se extraerán las características de la imagen y se buscaran las coincidencias con la imagen de fondo para así poder calcular una transformación que se adecue a cada caso.
Para ello tenemos que aplicar un detector, un extractor y un 'matcher', para encontrar la relación de los pixeles de la imagen de fondo con la imagen donde se encuentra la persona/objeto en movimiento.

- Detector
  Un detector sirve para extraer propiedades de las imágenes como son las esquinas que están presentes en las imágenes. A continuación se muestra una grafica comparativa de los distintos Detectores:

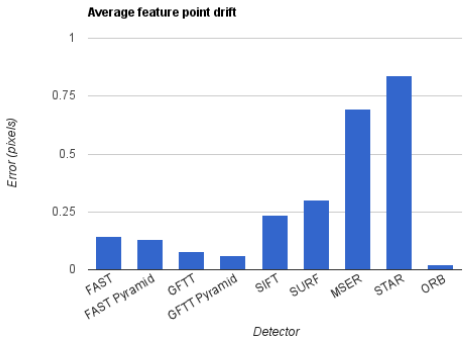


Fig1. Comparación de los errores de detección utilizando distintos algoritmos de detección[11].

En este caso he utilizado un detector del tipo ORB, es el que da mejores resultados y es muy parecido al SURF (el más utilizado), pero debido a que SURF no esta liberado y viendo los buenos resultados que se obtienen utilizando ORB, decidí hacerlo con este.

- Extractor
  Una vez tenemos ya todos los puntos detectados, el extractor sirve para encontrar los vectores que describen estos puntos y que en el siguiente paso se utilizaran para encontrar las coincidencias. En este caso también he utilizado el ORB.

- Matcher
  Una vez tenemos todos las características de las dos imágenes, pasamos ha realizar el 'match', para ello he utilizado el algoritmo BRUTEFORCEHAMMING, el cual , como su nombre indica, hace todas las combinaciones posibles de parejas de puntos y escoge el que tiene una distancia Haming con el error más pequeño.

5.3 Selección de los mejores puntos

Una vez llegados a este punto, tenemos todas las relaciones posibles entre las dos imágenes. Estas relaciones no significan que sean las correctas, debido a que el matcher puede haber fallado a la hora de decir que un pixel corresponde a otro pixel totalmente distinto.
A continuación se muestra una figura donde se puede apreciar lo mencionado anteriormente:

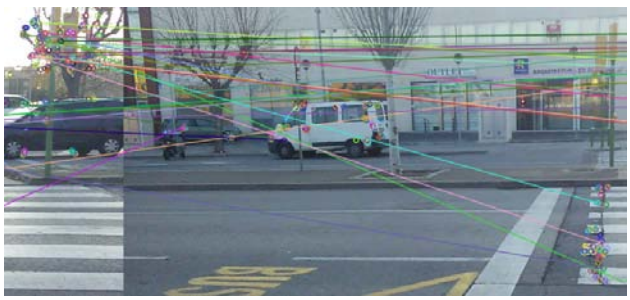


Fig2: Errores provocados por el match.

En la figura 2 podemos observar como relaciona los puntos del semáforo (imagen de la izquierda) con el paso de peatones(imagen de la derecha). Es por esto que se debe hacer una selección de los mejores puntos.

Para contrastar con lo mencionado anteriormente a continuación se muestra el resultado de un buen match.

Fig3: Puntos seleccionados correctamente.

En la figura 3 podemos observar como el match selecciona bien los puntos y están todos relacionado correctamente. Esta figura sirve para contrastar con la figura 2 mencionada anteriormente y observa la diferencia de tener un buen match y no.

En mi caso el ORB devuelve como máximo 500 puntos para cada imagen, más de la mitad de puntos son erróneos. Para ello decidí solo coger los 50 primeros puntos que tienen una distancia mínima entre ellos (Entre el punto de la imagen de fondo y el punto de la imagen donde se encuentra la persona/objeto en movimiento). Para coger estos puntos utilicé el algoritmo de ordenación Selection Sort, el cual ordena la lista de forma creciente según la distancia de los puntos, para después seleccionar los primeros 50 puntos con menor distancia.

Para solucionar este problema se aplicaron dos técnicas:

1. Selección aleatoria de 3 puntos
   Una transformación se puede realizar con 3 puntos, esto te permite hacer una rotación, escalado y traslación. Utilizando la función *getAffineTransform* que esta incluida en el SDK de OpenCV, se puede obtener la matriz de transformación que mas tarde será aplicada a la imagen que se ha de modificar.
   Se hacen n iteraciones escogiendo tres puntos de forma aleatoria, diferentes entre ellos y que tengan una distancia mínima en la imagen. Para cada iteración se calculan, con estos tres puntos , la matriz de transformación. Una vez obtenida la matriz de transformación se aplica esta a todos los puntos de la imagen con la persona/objeto en movimiento y se compara el resultado con la imagen de fondo. Una vez acabado todo este proceso se seleccionan los tres puntos que más aciertos han tenido.

2. Resolución por mínimos cuadrados
   Se calcula la matriz de transformación utilizando un algoritmo de resoluciones para un sistema de ecuaciones sobre determinado (mínimos Cuadrados) el cual encuentra la solución para ese sistema donde el error global es el mínimo para todos los puntos.
   Para ello se recorre el array de matches y se va aplicando este algoritmo de manera reiterativa disminuyendo en cada paso la longitud del array hasta conseguir que el array tenga una longitud de 4. En cada paso del bucle se calcula la matriz de transformación para esa longitud y se aplica esa transformación a todos los puntos para poder obtener donde se encuentran de la imagen de fondo. Por ultimo se calcula la distancia euclidiana del punto obtenido con el punto original. Una vez aplicada las transformaciones a todos los puntos se calcula la media de las distancia euclidianas, y es este el valor que luego se comparara para saber si tenemos una buena transformación.

   Una vez finalizado el proceso tendremos la matriz de transformación que mas se ajusta a la imagen y que posiblemente de mejores resultados.

   Este método es el utilizado en la solución final debido alto rango de aciertos con respecto al anterior.

6 RESULTADOS

Los resultados obtenidos han sido positivos, en el caso de las fotos realizadas con trípode. Por el contrario, los resultados obtenidos con fotos o videos sin trípode no son del todo correctos, depende mucho de la escena. Es por eso que en algunos casos obtenemos resultados mas o menos buenos y en otros no queda una imagen nítida.

A continuación si muestran dos ejemplos de cada tipo de imagen, con trípode y sin el:

Fig4. Resultado con trípode

En el apéndice A1 se muestran más fotos realizadas con trípode utilizando esta aplicación.

Fig5. Resultado sin trípode

En el apéndice A2 se muestran más fotos realizadas sin trípode utilizando esta aplicación.

Es muy complicado que aunque tengas unas buena matriz de transformación los pixeles coincidan exactamente, al hacer la selección de pixeles uno a uno, si las imágenes transformadas no coinciden con el fondo podemos obtener unos resultados erróneos.

7 UTILIZACIÓN DE LA APLICACIÓN

En esta sección se explica el funcionamiento y las características concretas que tiene la aplicación.

La aplicación permite al usuario cargar imágenes o videos que tenga alojadas en el dispositivo o inclusive grabar el video en directo y procesarlo a continuación. También permite visualizar todas las imágenes realizadas desde la aplicación. En el Apéndice A3 se encuentran las capturas de pantalla de la interfaz.

En el caso de las imágenes, el usuario puede elegir las fotos que quiera siempre y cuando elija primero el fondo que quiere que tenga su imagen final. Una vez seleccionado todas las imágenes, empezara el proceso de unión que puede tardar varios segundos, dependiendo de la calidad de cada imagen y el numero de imágenes seleccionadas. Una vez completado el proceso la imagen se guarda automáticamente y el usuario vera el resultado final en la galería de fotos de su samartphone, desde el cual la puede compartirla con sus amigos.

Por otro lado el usuario puede cargar o grabar un video, el cual es procesado y sus fotogramas se muestran en la parte inferior de la aplicación. El usuario puede elegir los fotogramas que quiere utilizar para obtener su imagen final, el cual cada vez que selecciona un fotograma puede ver el resultado que se obtendrá. Una vez seleccionados todos los fotogramas, el usuario tiene la posibilidad de guardar la imagen resultante.

La aplicación también dispone de una parte de opciones donde el usuario puede configurar los parámetros que se muestran a continuación:

1. El intervalo de tiempo de captura de un fotograma. Es decir cada cuantos segundos se guardara un fotograma: 0.2, 0.5, 0.8, 1.0, 1.5 segundos.

2. Seleccionar el tiempo máximo de grabación del video. Es la duración máximo que podrá tener el video: 2, 5, 8, 10, 15, 20, 25, 30 segundos.

3. Selección de la extensión que tendrá la imagen final: JPEG,PNG o WEBP.

4. Seleccionar la calidad de la imagen resultante: Valor entre 0 y 100.

8 TRABAJOS FUTUROS

La optimización de parte de la aplicación para que el proceso sea más rápido y el usuario no tenga que esperar tanto cada vez que selecciona un fotograma. Mejorar la selección de características y modificar el código para que las fotos sin trípode obtuvieran buenos resultados.
Investigar la manera de conseguir mejores resultados implementando distintos algoritmos o incluso modificando la unión de las imágenes para corregir el error.
También, se plantea la posibilidad de mejorar la extracción de los fotogramas del video de forma nativa utilizando ff-mpeg, debido a que ahora, por falta de tiempo, la extracción se hace mediante Java.
A su vez se podría implementar la captura tanto de video como de imágenes sin tener que salir de la aplicación y así poder realizar secuencias de imágenes para procesarlas.

Por ultimo mejora la interfaz grafica de la aplicación, haciendo que se adapte a todo tipo de dispositivos Android , para así poder ampliar el mercado.

9 CONCLUSIONES

Para concluir, la creación de una secuencia de imágenes de forma automática es un tarea complicada debido a que depende mucho de la persona que realiza las fotografías y la escena que se quiera plasmar. Todo lo relacionado con la luz afecta mucho a la hora de comparar imágenes y es por eso que se pude convertir en un problema difícil de resolver.
También mencionar la manera de alinear las imágenes para que correspondan es una tarea difícil debido a que importa mucho la selección de puntos y el propio match. Un ejemplo claro donde no funcionaria correctamente la aplicación es en una escena donde existen arboles y que los mejores puntos sean las hojas de los arboles, debido a que en cada imagen a causa del viento las hojas se moverían y no correspondería con el fondo.
Quiero destacar que este proyecto me ha servido para aplicar los conocimientos de visión artificial y técnicas grafiques que he aprendido a lo largo de la carrera en una aplicación real que solo unos pocos han conseguido hacer de manera eficiente. Esta aplicación me ha servido para

afirmar el gran potencial que tiene el sistema operativo Android y la cantidad de herramientas que ofrece su SDK. Por ultimo, mencionar la importancia de la librería OpenCV para el tratamiento de imágenes y la utilidad que tiene esta aplicada a proyectos de visión Artificial.

AGRADECIMIENTOS

Quiero dar las gracias a mi tutor del TFG: Francisco Javier Sánchez Pujadas por ayudarme a resolver los problemas planteados y ayudar en la toma de decisiones. Por otra parte, en especial a Jairo Vadillo Martin, que me ha ayudado ha realizar los experimentos y a Daniel Jiménez Mendoza, el cual me ha asesorado en el diseño de la interfaz.
Para acabar a todas las persones que han hecho de modelo en la toma de fotografías o vídeos expuestos en este articulo:
Marc Serra Roig, Bianca Molina Tortosa, Javier Molina Sánchez y Manuel Peña Avellanada.

BIBLIOGRAFÍA

[1] Android NDK | Android Developers [Fecha de consulta: 30 de octubre de 2013] Disponible en: http://developer.android.com/tools/sdk/ndk

[2]F. Liu. Marzo 2013. Android Native Development Kit Cookbook. ISBN: 978-1-84969-150-5

[3]. Android SDK | Android Developers [Fecha de consulta : 30 de octubre de 2013]
Disponible en : http://developer.android.com/sdk/

[4]. MATLAB Documentation - MathWorks España [Fecha de consulta: 30 de octubre de
2013] Disponible en: http://www.mathworks.es/es/help/matlab/

[5]OpenCV [Fecha de consulta: 20 de octubre de 2013] Disponible en: http://opencv.org/

[6]B Boehm. 1986. A spiral model of software development and enhancement. SIGSOFT Softw. Eng. Notes 11, 4 (August 1986), 14-24. http://doi.acm.org/10.1145/12944.12948[Fecha de consulta: 10 de octubre de 2013

[7] S Ratabouil. January 2012. Android NDK Beginner's Guide ISBN 978-1-84969-152-9

[8]D Lelis. Noviembre 2012. Mastering OpenCV with Practical Computer Vision Projects. ISBN 978-1-84951-782-9

[9]. Welcome to opencv documentation! [Fecha de consulta: 11 de enero de 2014] Disponible en: http://docs.opencv.org/

[10]. R. Laganière. Mayo 2011. OpenCV 2 Computer Vision Application Programming Cookbook ISBN 978-1-849513-24-1

[11] Comparison of the OpenCV's feature detection algorithms – II |Autor: IevgenKhvedchenia | 13 de julio de 2011| ! [Fecha de consulta: 15 de enero de 2014] Disponible en: http://computer-vision-talks.com/2011/07/comparison-of-the-opencvs-feature-detection-algorithms-ii/

[12] Motion Shot | Support | Sony |Autor : Sony Corporation | 24 de diciembre de 2013 | Disponible en: http://x-application.sony.net/motionshot/en/

[13] Application Development with Static Initialization | Autor: OpenCV [Fecha de consulta: 10 de diciembre de 2013] Disponible en: http://docs.opencv.org/trunk/doc/tutorials/introduction/android_binary_package/dev_with_OCV_on_Android.html

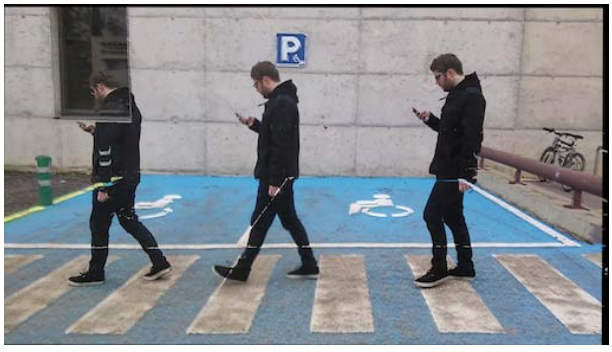
[14] Image Sequences in Photoshop | Autor: Sean Duggan | 15 de julio de 2010 | [Fecha de consulta: 5 de enero de 2014] Disponible en: http://layersmagazine.com/image-sequences-in-photoshop.html

[15] Mastering Drama Shot on the GALAXY S4 and Note 3 | Autor: Your Mobile Life | 21 de octubre 2013 |
[Fecha de consulta: 10 de enero de 2014] Disponible en: http://www.techradar.com/news/phone-and-communications/mobile-phones/mastering-drama-shot-on-the-galaxy-s4-and-note-3-1191465

Apéndice

A1. Fotos realizadas con trípode

A2. Fotos realizadas sin trípode



A3. CAPTURAS DE PANTALLA