EVALUACIÓN DE POSIBILIDADES DE PROCESAMIENTO DE IMÁGENES EN REAL-TIME PARA SISTEMAS MÓVILES

Obando Núñez Andrea Del Rocío [(1)], Orrala Parrales Fabricio Diógenes [(2)]
Facultad de Ingeniería en Electricidad y Computación (FIEC)
Escuela Superior Politécnica del Litoral (ESPOL)
Campus Gustavo Galindo, Km 30.5 Vía Perimetral
Apartado 09-01-5863 Guayaquil, Ecuador
aobando@espol.edu.ec [(1)], fadiorra@espol.edu.ec [(2)]

RESUMEN

Este trabajo tiene como propósito realizar un análisis del rendimiento que tienen los teléfonos móviles con sistema operativo Android, con el fin de comparar las capacidades para realizar procesamiento de imágenes en tiempo real. Para esto, se estudiaron las tecnologías de desarrollo NDK y SDK proporcionadas por Google para el diseño e implementación de aplicaciones móviles. También se estudiaron las herramientas usadas en visión por computador para procesamiento de imágenes.

Para analizar estas capacidades se realizaron dos tipos de experimentos claves. El primero que mide los tiempos promedio de captura de imágenes y el segundo que mide los tiempos promedios de ejecución de algoritmos de procesamiento de imágenes. En ambos experimentos se utilizó NDK y SDK con los métodos propios de las tecnologías provistas por Google y luego combinándolas con la librería OpenCV.

***PALABRAS CLAVE**: android, ndk, sdk, opencv, tiempo real, teléfono inteligente, procesamiento de imágenes.*

ABSTRACT

This paper aims to conduct a performance analysis that has mobile phones with Android operating system, to compare the abilities for image processing in real time. This research studied NDK and SDK as development tools provided by Google to design and implement mobile applications. The tools used in computer vision for image processing is also studied.

To analyze these capacities it has made two key experiments. The first, that measures the image capture average time and the second that measures run time of the image processing algorithm. Both experiments used NDK and SDK with the methods and tools provided by Google and then both combined with OpenCV.

***KEYWORDS**: android, ndk, sdk, opencv, real-time, smarphone, image processing.*

1. INTRODUCCIÓN

La utilidad de los dispositivos móviles es diversa. Desde la recopilación de datos como imágenes y documentos de texto, hasta la transferencia de información de manera eficiente y en cualquier lugar, hacen a estos dispositivos una herramienta potencialmente útil para diferentes propósitos. Por ejemplo el uso en Automatización Industrial [11], comunicaciones de Audio en Tiempo Real – Android [2], etc.

Los teléfonos que tienen sistema operativo Android, se encuentran las provistas por Google como el Kit de Desarrollo de Software (SDK por sus siglas en inglés) y el Kit de Desarrollo Nativo (NDK por sus siglas en inglés). Cabe señalar, que a pesar que se puede utilizar diferentes lenguajes de programación, los desarrolladores optan por hacer uso del SDK, debido a la familiaridad del lenguaje [1] el cual usa la sintaxis y la semántica de Java.

Por otro lado, en las aplicaciones donde se procesan datos en tiempo real, los tiempos de ejecución son críticos y es necesario diseñar programas que optimicen los recursos con el objetivo

de minimizar los tiempos de respuesta de las aplicaciones. En un teléfono inteligente, este escenario se complica debido a las limitaciones propias del hardware. Si bien la capacidad de procesamiento de estos dispositivos aún es reducida en relación a las computadoras personales; a medida que avanza la tecnología la brecha se acorta, hoy en día existen dispositivos de hasta 8 núcleos con altas capacidades de procesamiento.

El presente trabajo se enfoca en analizar las capacidades de captura y de procesamiento de imágenes en un tipo de teléfono móvil que ejecuta el Sistema Operativo Android. También se evalúa las librerías en términos de tiempos de captura de imágenes y tiempos de ejecución de algoritmos básicos de procesamiento de imágenes.

2. INFORMACIÓN GENERAL

Se dividen en objetivos, alcances y limitaciones las cuales se muestran de manera detallada a continuación.

2.1.1 OBJETIVO GENERAL

Evaluar cuantitativamente el rendimiento de librerías de procesamiento de imágenes para teléfonos que poseen sistema operativo Android, utilizando los API NDK y SDK de Android para tener un punto de referencia al momento de elegir las herramientas más adecuadas para desarrollar programas móviles que hagan procesamiento de imágenes.

2.1.2 OBJETIVOS ESPECÍFICOS

- Investigar acerca de la arquitectura del Sistema Operativo Android.
- Reconocer librerías que permitan desarrollar aplicaciones de visión por computador en teléfonos móviles con sistema operativo Android.
- Comparar tiempos de captura de imágenes en programas escritos usando SD, librerías OpenCV para Java y para C/C++.
- Comparar tiempos de ejecución de diferentes algoritmos de procesamiento de imágenes usando el soporte SDK, NDK, librerías OpenCV en lenguajes Java y C/C++.
- Seleccionar la herramienta de programación que tenga el menor tiempo de ejecución para capturar y procesar imágenes en tiempo real.

2.2 ALCANCES Y LIMITACIONES DEL PROYECTO

Para realizar la comparación con NDK se utilizará la versión de Android 4.0.3. Desde la versión 1.5 de Android se da soporte a código nativo en NDK [15]. Se centrará estrictamente en el proceso de adquisición y procesamiento de imágenes más no en la forma y métodos para pintar píxeles en la pantalla del teléfono. Como dato de entrada se trabajará con imágenes a color captadas por la cámara del teléfono.

En este trabajo, únicamente se evaluarán las operaciones de adquisición y procesamiento de imágenes señaladas en la Tabla 1.

Tabla 1. Operaciones de captura y procesamiento de imágenes.

FASE DE PROCESAMIENTO	OPERACIÓN
Adquisición de Imágenes	Captura y Adquisición
Operaciones de Pre – Procesamiento	Región de Interés (ROI) Operaciones Geométricas Operaciones Aritméticas Operaciones Lógicas Operaciones Morfológicas Filtros especiales Cuantización de imágenes
Realzado de imágenes	Afinamiento de imágenes Suavizado de imágenes

Se usará Java para compilar el código de los experimentos y las especificaciones de la versión de Java que se utilizarán están descritas en la Tabla 2.

Tabla 2. Especificaciones de Java a utilizar.

ESPECIFICACIÓN DE JAVA	VERSIÓN
Java	1.7.0_13
Java ™ SE Runtime Environment	1.7.0_13-b20
Java Hotspot ™ Client VM	23.7-b01

3. MARCO TEÓRICO

3.1 ANDROID

Es un sistema operativo diseñado para teléfonos y basado en el kernel de Linux. Para desarrollar programas en dispositivos Android, Google proporciona tres herramientas de desarrollo: SDK, NDK y la más reciente RenderScript [1].

Este conjunto de herramientas permiten comunicar aplicaciones con los componentes del teléfono donde se ejecuten

3.1.1 ARQUITECTURA

La arquitectura de Android está comprendida en cinco capas las mismas que son mostradas en la Figura 1.

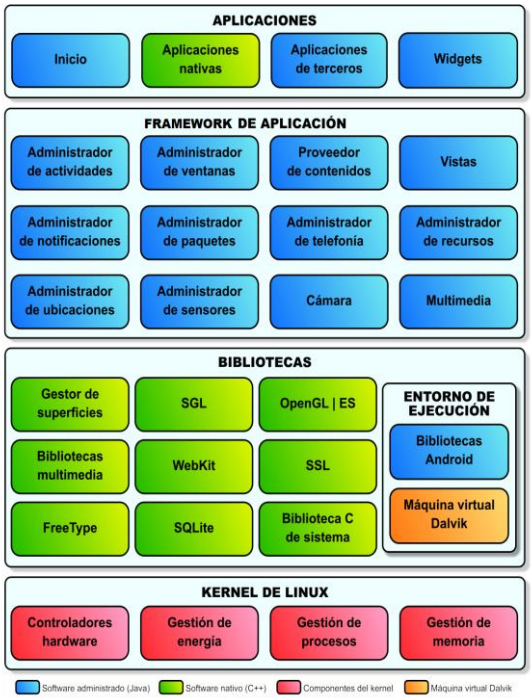

Figura 1. Arquitectura de Android [13]

Kernel de Linux: Basado en Linux Kernel 2.6. Android, usa el kernel de Linux como una capa de abstracción de hardware [4]. Y ha sido modificado para las necesidades especiales de administración de energía, la gestión de la memoria y el entorno de ejecución [5].

Bibliotecas o Librerías: Contiene los códigos que provee las principales características de Android, además que son las que ayudan a comunicar de mejor manera los componentes de hardware con el software.

Entorno de ejecución: Consiste en la máquina Virtual Dalvik y el núcleo de las librerías de Java. La máquina virtual de Dalvik (DVM) es un intérprete de códigos que han sido transformados de bytecode de Java a bytecode de Dalvik. Propiamente Dalvik esta compilado a código nativo, mientras que las librerías que interpreta están escritas en Java [5].

Framework de Aplicación: Contiene las clases de java para la creación de aplicaciones, que interactúan con el hardware y la interfaz de usuario [3].

Capa de Aplicaciones: Donde se encuentran las aplicaciones preinstaladas y las aplicaciones instaladas por el usuario. Estas aplicaciones hacen uso de todas las capas [4].

3.2 SDK

Software Development Kit (SDK) es un API de librerías y paquetes necesarios para el desarrollo y pruebas de aplicaciones Android. SDK posee herramientas separadas en dos grupos: SDK y plataforma. [9]:

SDK Tools (Herramientas de SDK): Importante para el desarrollo, en estas herramientas se encuentra el emulador, Android SDK Manager.

Platform Tools (Herramientas de Plataforma): Posee herramientas dependientes de la plataforma de Android.

3.2.1 FLUJO DE TRABAJO DE SDK

El código en SDK se compila a bytecode y se empaqueta en una aplicación. Cuando se inicia la aplicación el motor de ejecución de Android (DalvikVM) interpreta el bytecode o usa JIT para compilarlo y convertirlo a instrucciones de máquina para luego ser ejecutados [1]. El proceso de flujo de trabajo se muestra en la Figura 2.

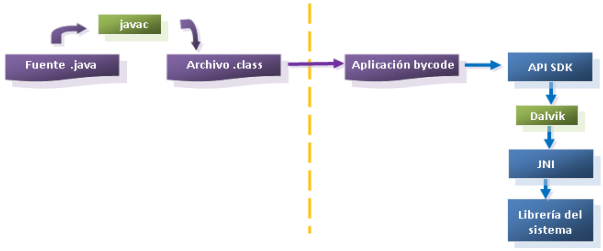

Figura 2. Flujo de trabajo de una aplicación usando SDK de Android. [1]

3.2.2 VENTAJAS DE SDK

- Es de código abierto, lo que permite a los desarrolladores de Android compartir proyectos y solucionar problemas comunes en cualquier lugar del mundo ya que son basados en un mismo código fuente.
- Tiene una herramienta llamada GU Toolkit. Que es una interfaz que usa el método “arrastrar y

soltar" para facilitar el diseño de las aplicaciones.

- Al usar la sintaxis de Java, se aporta una gran cantidad de soporte a desarrollo, pues la comunidad de desarrolladores para Java es numerosa.
- Usa la máquina virtual de Dalvik (DVM) por lo que tiene los beneficios del Garbage Collector de Java, como el hecho de que en determinado momento se liberarán recursos innecesarios.

3.3 NDK

Native Development Kit (NDK) es un conjunto de herramientas que permite incorporar los componentes que hacen uso de código nativo en las aplicaciones de Android. Permite implementar parte de las aplicaciones y las librerías de código nativo para Android en lenguajes como C y C ++ [6].

Para comunicar una aplicación nativa con código en C y C++ se usa Java Native Interface (JNI). Esto permite que código escrito en Java pueda ser ejecutado dentro de la Máquina Virtual de Java (JVM) para interoperar con aplicaciones y librerías escritas en otros lenguajes de programación como C, C++ y ensamblador [7].

3.3.1 FLUJO DE TRABAJO DE NDK

Android no provee de un API completo y específico para NDK, por tanto existe diferentes formas de usar NDK para crear librerías en lenguaje C/C++ que posteriormente pueden ser usadas por las aplicaciones. Cuando la aplicación se ejecuta en los dispositivos móviles, el código nativo se carga y se ejecuta a través de JNI [1] como un proceso de sistema operativo. El flujo de trabajo es como muestra la Figura 3.

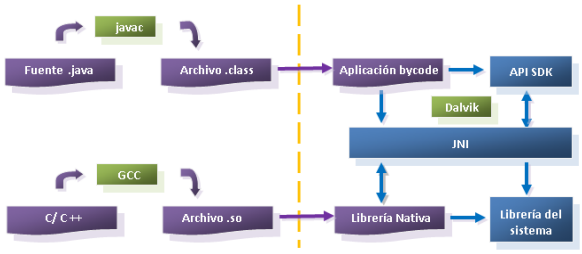

Figura 3. Flujo de trabajo de una aplicación usando NDK de Android. [1]

3.3.2 VENTAJAS DE NDK

- Las librerías de aplicaciones escritas en C/C++ están limitadas a la arquitectura del CPU del dispositivo [10]. Pero el código es portable en diferentes plataformas [14].
- Proporciona librerías de sistema para las APIs nativas, garantizando la compatibilidad de todas las versiones de Android [15], como por ejemplo:
  - libc (Librería de C).
  - libm (Librería de matemáticas).
  - JNI (Java native interface).
  - libz (compresión Zlib).
  - liblog (logging de Android).
  - OpenGL ES 1.1 y OpenGL ES 2.0 (Gráficas 3D).
  - libjnigraphics (Acceso al buffer de pixeles).
  - OpenSL ES (Librería para audio).
- Proporciona un mejor soporte que SDK para OpenGL, que es crítico para el procesamiento gráfico [1].

3.4 OPENCV

OpenCV es una librería de visión por computador desarrollada por Intel. La versión para Android ha desarrollado optimizaciones para que puedan ser ejecutadas sin ningún problema en arquitecturas de hardware ARM, etc. [12]. Para fines de desarrollo se ha implementado interfaces de comunicación en móviles con JNI de tal modo que se tienen métodos en Java que invocan funciones en C/C++. Con ello se facilita el desarrollo de aplicaciones de visión por computador en teléfonos.

En este trabajo se utilizará la implementación de OpenCV para código nativo de Android denominada OpenCV NDK y OpenCV para Android SDK llamada OpenCV SDK

La optimización de OpenCV en fase de pruebas para Android requiere de una aplicación mediadora llamada OpenCV-Maganer [8] cuyo modelo de ejecución se muestra en la Figura 4. Básicamente cuando el OpenCV Manager está ejecutándose, éste sugiere la instalación de la librería OpenCV acorde a la arquitectura del teléfono, de ser necesario. A partir de ahí es que se realiza la ejecución de la aplicación que necesita tal librería.

Al ser una librería que escrita en C/C++ y modificada para Android, se tiene la capacidad de usar código nativo para Android tanto en lenguaje C/C++

como en Java a fin de obtener mejoras en rendimiento y escalabilidad.

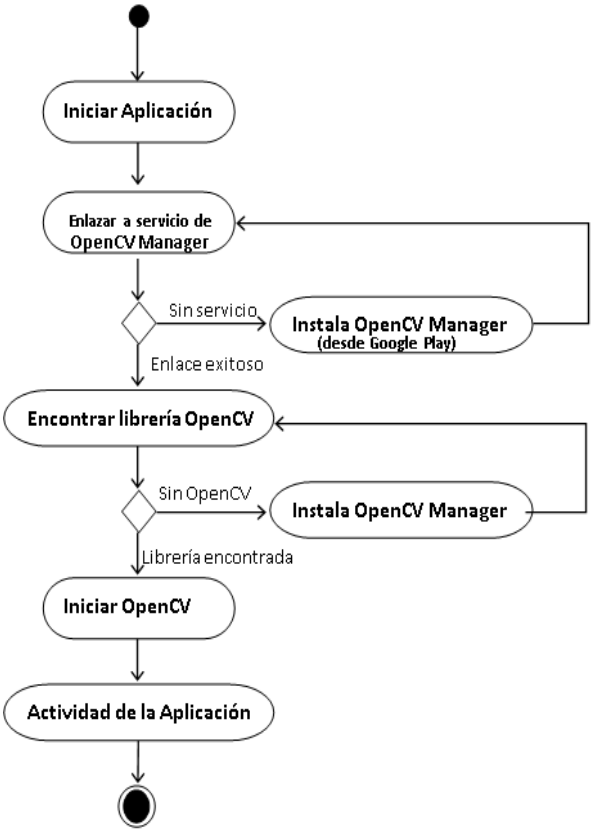

Figura 4. Modelo de ejecución de OpenCV para Android

4. METODOLOGÍA

Esta investigación consta de dos tipos de experimento.

1. Análisis de tiempo de adquisición de imágenes.
2. Análisis de tiempo de ejecución de algoritmos de procesamiento de imágenes.

Para cada uno de los experimentos se procedió de la siguiente forma:

1. Obtener tiempo inicial en nano segundos.
2. Realizar experimento (captura/ ejecución de algoritmo).
3. Obtenemos tiempo final en nano segundos.
4. El tiempo de duración es la resta entre el tiempo final y el tiempo inicial.

Para el primer experimento para Android SDK y para OpenCV. Para cada herramienta se realizó 5, 10, 25, 50 y 100 capturas de imágenes por experimento y se calculó el promedio así como de la desviación estándar que resulta de una misma prueba.

Para el segundo experimento se toman los tiempos desde que se inicia el algoritmo hasta que finaliza, por cada frame procesado, teniendo en cuenta que la cámara se encuentra inicializada y operativa. Cabe señalar que en estos experimentos no se considera tiempo de adquisición. Se hicieron pruebas para 5, 10, 25, 50 y 100, frames por experimento y se calculó el promedio así como de la desviación estándar que resulta de una misma prueba.

5. RESULTADOS

Para el primer experimento, en Android SDK los tiempos en pruebas obtenidos son más elevados que si se usaran OpenCV en SDK y NDK. Los datos obtenidos se muestran en la Tabla 3.

Tabla 3. Tiempos de adquisición de imágenes usando Android SDK.

No. Pruebas	**Android SDK x 10^8 (ns)**	
	$\overline{X}$	σ
5	4,4744	2,7645
10	3,3839	1,8063
25	3,2146	0,8900
50	2,7366	0,6610
100	2,6875	0,5857

Los tiempos en OpenCV en SDK equivalen al doble de los tiempos en la implementación para OpenCV NDK. La captura en Android SDK y aproximadamente es 20 veces mayor que las pruebas realizadas con OpenCV NDK. Los datos de esta prueba se muestran en la Tabla 5.

Tabla 5. Tiempos promedios y desviación estándar de adquisición de imágenes con OpenCV

No. Pruebas	**Android OpenCV x 10^8 (ns)**			
	NDK		**SDK**	
	$\overline{X}$	σ	$\overline{X}$	σ
5	0,1797	0,1218	2,2290	3,7071
10	0,0999	0,0204	1,6324	3,0533
25	0,6559	0,0550	0,9965	1,7264
50	0,1546	0,2080	0,8389	1,2993
100	0,8644	0,0637	0,7611	0,9638

En la Tabla 6 se muestran los resultados del segundo experimento, pruebas con algoritmos de pre-procesamiento, presentando los tiempos promedios obtenidos al momento de aplicar cada operación a 100 imágenes capturadas en las herramientas analizadas. De esta forma podemos analizar el comportamiento.

Tabla 6. Tiempos promedios para el experimento de algoritmos de Pre-procesamiento

PROCESAMIENTO (x 10^6 ns) 100 IMÁGENES		
ANDROID SDK	$\overline{X}$	σ
Roi	2,8890	0,9629
Rotación	0,0319	0,0004
Resta	2302,2449	185858,1973
And	32,2810	207,6880
OpenCV SDK	$\overline{X}$	σ
Roi	0,0503	0,0019
Rotación	31,4975	69,9415
Resta	117,6567	3612,6673
And	60,7952	777,2553
Erosión	11,8305	7,6840
OpenCV NDK	$\overline{X}$	σ
Roi	0,0060	0,0007
Rotación	32,1166	78,7626
Resta	143,3369	9296,1274
And	64,3704	1730,4599
Erosión	22,2938	70,1568

Podemos observar que en la implementación del algoritmo ROI, los tiempos en SDK Android son aproximadamente 50 veces más que los que toman SDK OpenCV. Por otra parte OpenCV NDK es 8 veces mejor en tiempo de ejecución que OpenCV SDK.

Para el algoritmo de ROTACION, el comportamiento cambia, la ventaja de SDK Android aparece, más de 500 veces mayor a los tiempos de ejecución de OpenCV SDK y OpenCV NDK.

El algoritmo RESTA, AND y EROSION presenta ventaja en OpenCV SDK con una mínima diferencia con OpenCV NDK. En Android SDK los tiempos de ejecución son elevados, aproximadamente 20 mayor que OpenCv SDK, para los algoritmos de Resta y AND. El algoritmo de Erosión no fue comparado en Andorid SDK debido que su implementación en esta librería no estaba definida.

En la Tabla 7 se muestran los resultados de pruebas con algoritmos de realzado, se efectúa un análisis igual al de las operaciones de pre-procesamiento. En ambas secciones se muestran los tiempos promedios de cada prueba en nanosegundos y la desviación estándar.

Tabla 7. Tiempos en nanosegundos para el experimento de algoritmos de Realzado de imágenes

Filtro	**REALZADO (x 10^6 ns) 100 IMÁGENES**			
	OpenCV SDK		**OpenCV NDK**	
	$\overline{X}$	σ	$\overline{X}$	σ
Mediano	51,5500	283,5162	43,9612	184,5674
Promedio	22,6654	56,0156	17,3290	37,8137
Paso Alto	26,2092	92,7242	23,1639	60,3026
Paso Bajo	27,7116	100,9910	25,1301	76,6836

Los filtros Mediano, Promedio, Paso Alto, Paso Bajo, presentan mejores tiempos de ejecución en OpenCV NDK. Pero los tiempos de ejecución no pasan de ser 1.5 veces mejores de OpenCV SDK.

6. CONCLUSIONES

Con las primeras pruebas de captura de imágenes, se evaluaron los tiempos de ejecución para Android SDK, y se pudo observar que tiempos eran entre 4 y 5 veces más que los obtenidos en las capturas con OpenCV para SDK y aproximadamente 20 veces más lento en las capturas de imágenes en NDK con OpenCV, tomando en cuenta la captura de 5, 10, 25, 50 y 100 imágenes. Pudiendo concluir en este experimento que las capturas realizadas en OpenCV NDK son más óptimas pues son las que tienen menores tiempos de ejecución.

La segunda parte de esta investigación implicó la evaluación de varios algoritmos de procesamiento de imágenes.

Para algoritmos de pre-procesamiento: ROI, Rotación, Resta y AND, se pudo observar que los tiempos de ejecución en Android SDK eran 20, 41, 2615 veces mayores a los de SDK y NDK OpenCV, para la captura de 100 imágenes. Notablemente, el uso

de estos algoritmos en Android SDK, no es aconsejable si se piensa en realizar programas en tiempo real.

Al comparar OpenCV SDK y OpenCV NDK se notó que OpenCV NDK para ROI es aproximadamente 33 veces más óptimo en tiempos de ejecución que OpenCV SDK. Por otra parte los algoritmos de Rotación, AND y Erosión en OpenCV NDK son mejores en tiempo de ejecución, aunque con muy poca diferencia con los tiempos de OpenCV SDK. En la Resta OpenCV SDK presentó mejor tiempo de ejecución que OpenCV NDK, pero así mismo esta diferencia no es grande. Sin embargo la resta en Android SDK es un proceso lento.

Los algoritmos de Realzado Mediano y Paso alto en tiempo de ejecución para 100 captura de imágenes fueron más óptimos en OpenCV NDK, la diferencia con OpenCV SDK tampoco fue mucha. Por otro lado el Filtro fue más óptimo en tiempos de ejecución en OpenCV SDK. El Filtro Paso Bajo en OpenCV SDK fue aproximadamente 2 veces más óptimo en tiempos de ejecución que en OpenCV NDK.

Con los experimentos realizados se puede concluir que OpenCV en su implementación nativa para NDK brinda una ventaja en tiempos de captura de imágenes, y para procesamiento de imágenes existe similitud entre OpenCV SDK y OpenCV NDK

7. RECOMENDACIONES

Para futuros experimentos recomendamos:

Para desarrollar aplicaciones donde el tiempo de captura de imágenes es crítico, se recomienda utilizar los algoritmos de procesamiento en OpenCV NDK.

Para desarrollar aplicaciones de procesamiento de imágenes, se recomienda utilizar los algoritmos de procesamiento en OpenCV SDK o en OpenCV NDK.

Realizar los experimentos con más de 100 imágenes capturadas, ya que los tiempos de ejecución se reducen a medida que van incrementando las capturas de imágenes.

En futuros experimentos similares es recomendable que se identifiquen desde donde empieza cada procesamiento, y separar la medición de tiempos de otros procesos como lo son la captura y el pintado de pixeles de la pantalla.

8. REFERENCIAS

[1] Quian, X., Zhu, G. & Li, X.(2012). Comparison and Analysis of the Three Programming Models in Google Android. Recuperado Octubre 10,2012, de http://people.apache.org/~xli/papers/applc2012-android-programming-models.pdf

[2] Universitat Politècnica de València (2012). Android real-time audio communications over local wireless. Waves, 4, 35 - 42. Recuperado Mayo 5,2013 de http://www.iteam.upv.es/revista/2012/4_ITEAM_2012.pdf

[3] Al, A. (Agosto 10, 2012).Android Architecture For System Application Software Stack.Recuperado Diciembre 20, 2012, de http://android-app-tutorial.blogspot.com/2012/08/architecture-system-application-stack.html#.UPyp7q4kRvA

[4] Kumar, S. (Mayo 10, 2012). Architecture of Android. Recuperado Diciembre 21, 2012, de http://www.androidaspect.com/2012/10/architecture-of-android.html

[5] Brahler, S. (2010).Analysis of the Android Architecture.Universidad del Estado de Baden-Württemberg.p.46. Recuperado Enero 10, 2013, de http://os.ibds.kit.edu/downloads/sa_2010_braehler-stefan_android-architecture.pdf

[6] Alvares, J. (Mayo 2, 2012). Conectar programas C/C++ con aplicaciones Android. Recuperado Febrero 10, 2012, de http://universo.emergya.es/espacios/jialvarez/conectar-programas-cc-con-aplicaciones-android

[7] Oracle. (2011). Java SE Documentation. Recuperado Mayo 7, 2013, de http://docs.oracle.com/javase/6/docs/technotes/guides/jni/spec/intro.html

[8] OpenCV.(2013). The OpenCV Manager Manual Release 2.4.5.0. Recuperado Mayo 7, 2013, de http://docs.opencv.org/trunk/opencv2manager.pdf

[9] Google Inc. Tools Help. Recuperado Mayo 2, 2013, de http://developer.android.com/tools/help/index.html

[10] Etheridge, Darren.(Marzo, 2012). Developing Android applications for ARM® Cortex™-A8 cores.Texas Instruments: Autor. Recuperado Mayo 3, 2013, de http://www.ti.com/lit/wp/spry193/spry193.pdf

[11] Di Cerbo M. & Rudolf A. (Enero 29,2012). Using Android in Industrial Automation. Universidad de Ciencias Aplicadas de Northwestern Switzerland.p.93. Recuperado Mayo 3, 2013, de http://android.serverbox.ch/wp-content/uploads/2010/01/android_industrial_automation.pdf

[12] Shore C. (2010).Developing Power-Efficient Software Systems on ARM Platforms. Volumen 8 (4),48-53. Recuperado Mayo 30, 2013, de http://www.iqmagazineonline.com/current/pdf/Pg48-53.pdf

[13] Vico A. (2011, Febrero 17). Arquitectura de Android. La columna 80. Recuperado Mayo 31, 2013 de http://columna80.wordpress.com/2011/02/17/arquitectura-de-android/

[14] Muzzammil K., Anuar A., Atiqah N. & Soo Y. (2012). Real-Time Video Processing Using Native Programming on Android Platform. IEEE - International Colloquium on Signal Processing and its Applications, 8, 277-281. Recuperado Junio 2, 2013 de http://eprints2.utem.edu.my/4099/1/cspa_Real-Time_Video_Processing_Using_Native.pdf

[15] Google Inc. Android NDK. Recuperado Junio 2, 2013 de http://developer.android.com/tools/sdk/ndk/index.html