ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

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

INFORME DE MATERIA DE GRADUACIÓN

Previa a la obtención del Título de:

INGENIERÍA EN CIENCIAS COMPUTACIONALES

ORIENTACIÓN SISTEMAS MULTIMEDIA

Presentado por:

OBANDO NÚÑEZ ANDREA DEL ROCÍO

ORRALA PARRALES FABRICIO DIÓGENES

GUAYAQUIL – ECUADOR

2013

AGRADECIMIENTO

A Dios, que nos ha conservado con vida y salud.

A nuestras familias, quienes han sido y son el

pilar más fundamental de nuestras vidas.

A nuestros profesores, por todos los conocimientos

y consejos transmitidos.

A KOKOA, Comunidad de Software Libre de ESPOL,

parte importante en nuestro desarrollo académico.

A nuestros amigos dentro y fuera de la universidad,

que con su apoyo nos ayudaron en el día a día.

DEDICATORIA

A Dios

A nuestras familias

A nuestros amigos

A nuestros profesores

TRIBUNAL DE SUSTENTACIÓN

Daniel Ochoa Donoso, PhD.

PROFESOR DE LA MATERIA DE GRADUACIÓN

Gonzalo Luzardo Morocho, MSc.

PROFESOR DELEGADO POR EL DECANO DE LA FACULTAD

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Informe de Graduación, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral"

(Art. 12 Reglamento de Graduación de la ESPOL)

Andrea Obando Núñez.

Fabricio Orrala Parrales.

RESUMEN

Este trabajo tiene como propósito realizar un análisis de las capacidades que tienen los teléfonos móviles que poseen sistema operativo Android, para realizar procesamiento de imágenes en tiempo real. Para esto, se estudiaron las diferentes tecnologías usadas para el diseño e implementación de aplicaciones móviles, así como también las herramientas usadas en visión por computador para procesamiento de imágenes.

De igual manera se detallan los experimentos realizados, la importancia y significado de los datos obtenidos y sobre todo el análisis de los mismos.

El documento está dividido en 4 capítulos que comprenden: la información general, marco teórico, pruebas de adquisición, pruebas de algoritmos de procesamiento de imágenes, análisis de resultados y posteriormente las conclusiones y recomendaciones.

En el capítulo 1 se exponen los objetivos y alcances de las pruebas efectuadas, así como la descripción los tipos de experimentos realizados. Se muestran las características relevantes de software y hardware de los teléfonos usados para la fase experimental.

En el capítulo 2 se presenta el marco teórico de la investigación realizada. Los conceptos y la información acerca de las herramientas y tecnologías usadas en los experimentos. Se pone a conocimiento nuestra investigación sobre el funcionamiento y la arquitectura de la plataforma Android.

En el capítulo 3 se exhiben las pruebas realizadas para medir tiempos de captura, partiendo por definir las condiciones iniciales del experimento hasta el análisis de los datos obtenidos.

En el capítulo 4 se muestran las pruebas con algoritmos básicos de procesamiento digital de imágenes así como el estudio de los resultados obtenidos en estos experimentos. Al igual que en el capítulo 3, se especifica las condiciones en las que se desarrollaron y se hace un análisis de los resultados.

Finalmente se presentan las conclusiones obtenidas y se plantean recomendaciones para futuros trabajos.

Palabras claves: android, ndk, sdk, opencv, real-time, teléfono, procesamiento de imágenes.

ÍNDICE GENERAL

ABREVIATURAS

SDK	Software Development Kit
NDK	Native Development Kit
OpenCV	Open Computer Vision
CPU	Central Processing Unit
GPU	Graphics Processing Unit
ROI	Region of Interest
RAM	Random Access Memory
JVM	Java Virtual Machine
API	Application Programming Interface
DVM	Dalvik Virtual Machine
JIT	Just in Time
GUI	Graphical User Interface
JNI	Java Native Interface
RGB	Red – Green – Blue (Espacio de color)
NS, ns	Nanosegundos

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

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 [13], comunicaciones de Audio en Tiempo Real – Android [2], etc.

Los desarrolladores de programas para teléfonos móviles cuentan con muchos recursos y herramientas para programar aplicaciones. En el caso de 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 es cierto que 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.

CAPÍTULO 1. INFORMACIÓN GENERAL

1.1. Objetivos

1.1.1. 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.

1.1.2. 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 SDK, librerías OpenCV para Java y para C/C++.
- Comparar tiempos de ejecución de diferentes algoritmos de procesamiento de imágenes en programas escritos 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.

1.2. Alcances y limitaciones del proyecto

Para determinar la herramienta que provee los menores tiempos de captura y procesamiento de imágenes en teléfonos que ejecutan sistemas Android, sólo se utilizarán SDK y NDK de Google y las implementaciones de OpenCV para Android SDK y NDK debido a que cuentan con extenso soporte y documentación por parte de sus desarrolladores.

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 [21]. 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

El dispositivo a usar en la fase experimental será un teléfono Samsung Galaxy S III mientas que para pruebas secundarias se utilizará un teléfono Samsung Galaxy S II. Las características de los teléfonos a ser utilizados se detallan en la Tabla 3.

Tabla 3. Características de los teléfonos de prueba.

Características		Galaxy SIII	Galaxy S II
Sistema	**Modelo**	GT I9300	SGH I777
	Fabricante	Samsung	Samsung
	Versión de Android	4.1.2 (Jelly Bean)	4.0.3 (Ice Cream Sandwich)
	Nivel de API	16	15
	Versión de Kernel	3.0.31-306699 se.infra@SEP-94 #1	3.0.15-I777UCLE5-CL652575 se.infra@SEP-74 #3
Procesador	**Procesador**	ARMv7 Processor rev 0 (v71)	ARMv7 Processor rev 1 (v71)
	Núcleos	4	2
	Frecuencia máxima	1400 MHz	1200 MHz
	Set de instrucciones	Armeabi-v7a, armeabi	Armeabi-v7a, armeabi
Memoria	**Memoria RAM**	832 MB	830 MB
	Memoria JVM máxima	64 MB	128 MB
Cámara	**Megapíxeles**	8	8
	Formato de imagen	Jpeg	Jpeg
	Resolución máxima	3264x2448 píxeles	3264x2448 píxeles
	Formato de imagen de vista previa	Yuv420sp	Yuv420sp
	Resolución de vista previa máxima	960x720píxeles	800x480 píxeles
	Fotogramas por segundo	30 fps	30 fps

CAPÍTULO 2. MARCO TEÓRICO

2.1. Android

Es un sistema operativo diseñado para teléfonos y basado en el kernel de Linux. Durante años el proyecto Android siguió su propio camino y su código no fue incluido de nuevo en el árbol principal de Linux [3]. Pero desde la versión 3.3 el kernel de Linux y el de Android se han fusionado [22].

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.

2.1.2. Arquitectura

La arquitectura de Android está comprendida en cinco capas las mismas que son: Kernel de Linux, Bibliotecas, Entorno de Ejecución, Framework de aplicación y Capa de Aplicaciones como se indica en la Figura 2.1.

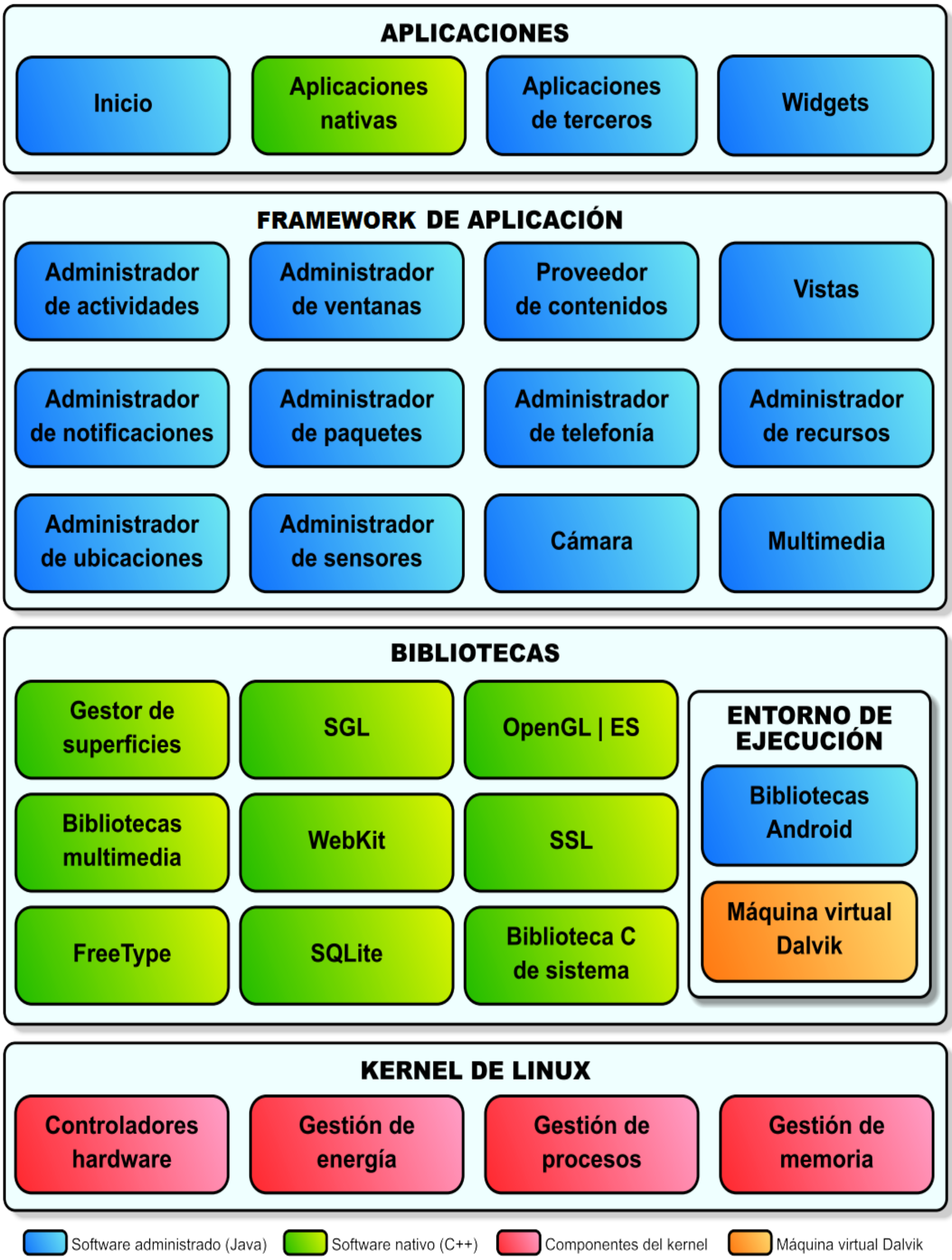

Figura 2.1 Arquitectura de Android [15]

Kernel de Linux: Basado en Linux Kernel 2.6. Android, usa el kernel de Linux como una capa de abstracción de hardware [5]. Y ha sido modificado para las necesidades especiales de administración de energía, la gestión de la memoria y el entorno de ejecución [6].

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 [6].

Framework de Aplicación: Contiene las clases de java para la creación de aplicaciones, que interactúan con el hardware y la interfaz de usuario [4].

Capa de Aplicaciones: Donde se encuentran las aplicaciones preinstaladas y las aplicaciones instaladas por el usuario. Estas aplicaciones hacen uso de todas las capas [5].

2.2. 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. [11]:

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.

2.2.1. 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 2.2.

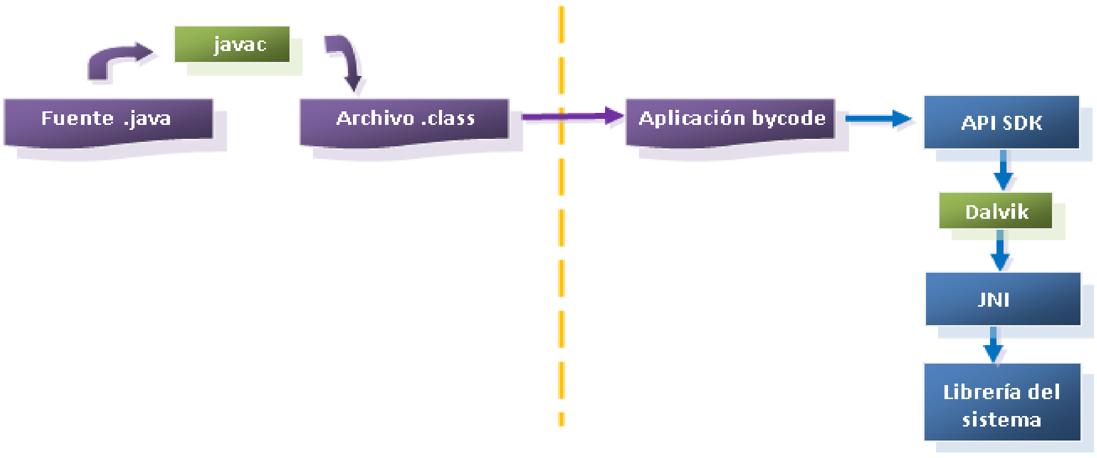

Figura 2.2. Flujo de trabajo de una aplicación usando SDK de Android. [1]

2.2.2.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 GUI 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 de 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.

2.3. 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 ++ [7].

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 [8]

2.3.1.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 2.3.

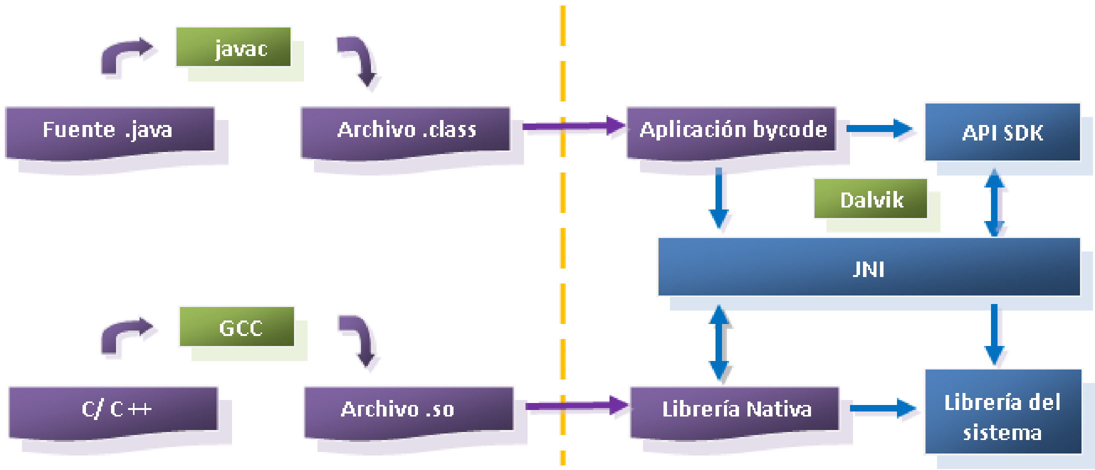

Figura 2.3. Flujo de trabajo de una aplicación usando NDK de Android. [1]

2.3.2. Ventajas de NDK

- Las librerías de aplicaciones escritas en C/C++ están limitadas a la arquitectura del CPU del dispositivo [12]. Pero el código es portable en diferentes plataformas [18].
- Proporciona librerías de sistema para las APIs nativas, garantizando la compatibilidad de todas las versiones de Android [21], 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].

2.4. Programas y Aplicaciones para Android

Las aplicaciones en Android están escritas siguiendo la sintaxis de Java y esquema de programación orientada a objetos que este lenguaje provee, pero tienen su propia manera de ser compiladas y ejecutadas.

Una aplicación en Android está constituida por la combinación de los siguientes componentes [9]:

Activities: Son la representación de la interfaz con la que va a interactuar el usuario, es decir lo que se va a observar. Si una aplicación tiene más de una "pantalla", cada una es un Activity distinto e independiente.

Services: Son componentes que se ejecutan en segundo plano usados especialmente para realizar operaciones de larga duración o también para ejecutar operaciones remotas.

Content Providers: Gestionan y administra el contenido de datos de alguna aplicación ya sea una base de datos o cualquier medio donde se almacenen datos privados que no son accedidos por otros medios.

Broadcast Receivers: Manejan los mensajes y alertas de sistema y la interacción del usuario para con esas alertas. Por ejemplo solicitudes de apagado y su mensaje de aprobación.

Todos estos componentes forman parte de una aplicación para Android, aunque una aplicación no requiere usar todos.

2.5. Captura de imágenes

La captura de imágenes en teléfonos consiste básicamente en el proceso de tomar una fotografía de una escena usando la cámara del dispositivo. Existen muchos factores que influyen en el proceso de captura, como la iluminación del ambiente, enfoque, exposición y demás sensores de cámara. Al final se obtiene una imagen en un espacio de color definido y lista para realizar las operaciones de pre-procesamiento.

2.6. Algoritmos de Procesamiento de imagen

Son operaciones básicas de tratamiento de imágenes que modifican una imagen antes de ser procesada. En general se tratan de algoritmos de reducción de datos para extraer únicamente la información que se necesita para resolver un problema. Esto hace que las posteriores tareas sean más fáciles de implementar El flujo de procesamiento de imágenes está definido de la siguiente manera:

Pre-Procesamiento: Elimina el ruido e información redundante de la imagen de entrada.

Realzado de imágenes: Consiste en un conjunto de técnicas que tratan de conseguir una imagen mejorada y más nítida.

Reducción de datos: Reduce los datos en el dominio espacial o frecuencial.

Análisis de características: Los datos generados desde la reducción de datos son examinados y evaluados para extraer sus características. Estas características son usadas en la solución de la aplicación.

2.6.1 Análisis de imágenes: Operaciones de Pre-Procesamiento

Región de Interés (ROI): Esta operación es útil cuando se quiere realizar un análisis más cercano de un área específica de la imagen. El proceso es bastante simple pues se trata de usar una figura rectangular para definir un área específica dentro de una imagen y descartar información innecesaria. Ver Figura 2.4.

Figura 2.4. Resultado de aplicar operación ROI

Operaciones Geométricas: Estas operaciones son útiles para modificar las coordenadas espaciales de la imagen. Se listan a continuación: crop, alargamiento, encogimiento, traslación, rotación. Ver Figura 2.5.

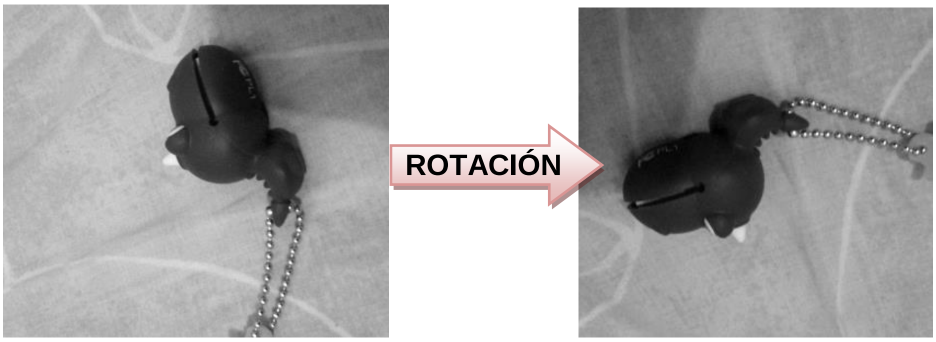

Figura 2.5. Resultado de aplicar operación geométrica de ROTACIÓN (90 grados).

Operaciones Aritméticas: Sirven para agregar, modificar, eliminar información indeseable de una imagen. Son útiles en varios tipos de procesamiento como: modelado de ruido, creación de efectos especiales, detección de movimientos, realzado de imágenes. Son las siguientes: suma, resta, multiplicación y división. La Figura 2.6 muestra el resultado de aplicar la operación RESTA entre dos imágenes.

El proceso consiste en restar de una imagen el valor correspondiente de otra imagen.

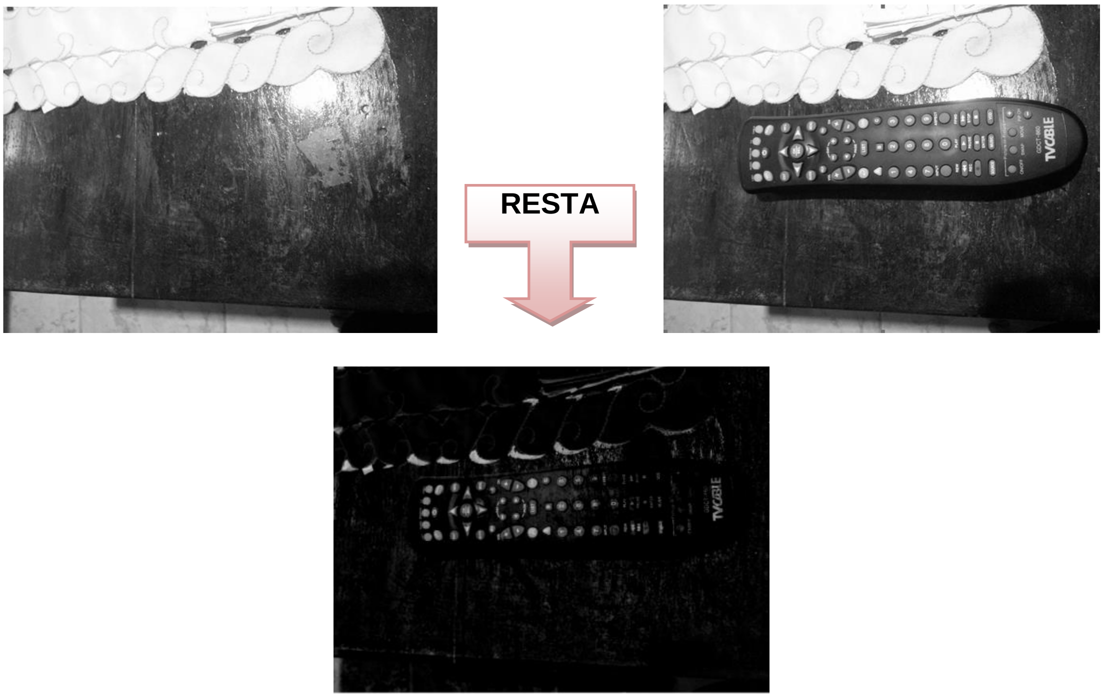

Figura 2.6. Resultado de aplicar operación aritmética RESTA.

Operaciones Lógicas: Son útiles para varios tipos de aplicaciones, tales como: detección de diferencias o similitudes, extracción de áreas de interés, enmascarar regiones, realzado de brillo. Y son las siguientes: and, or, not. La Figura 2.7 muestra el proceso de aplicar la operación AND, para ello recibe dos imágenes de entrada, se las binariza, se efectúa la operación y la imagen resultante es otra imagen binaria.

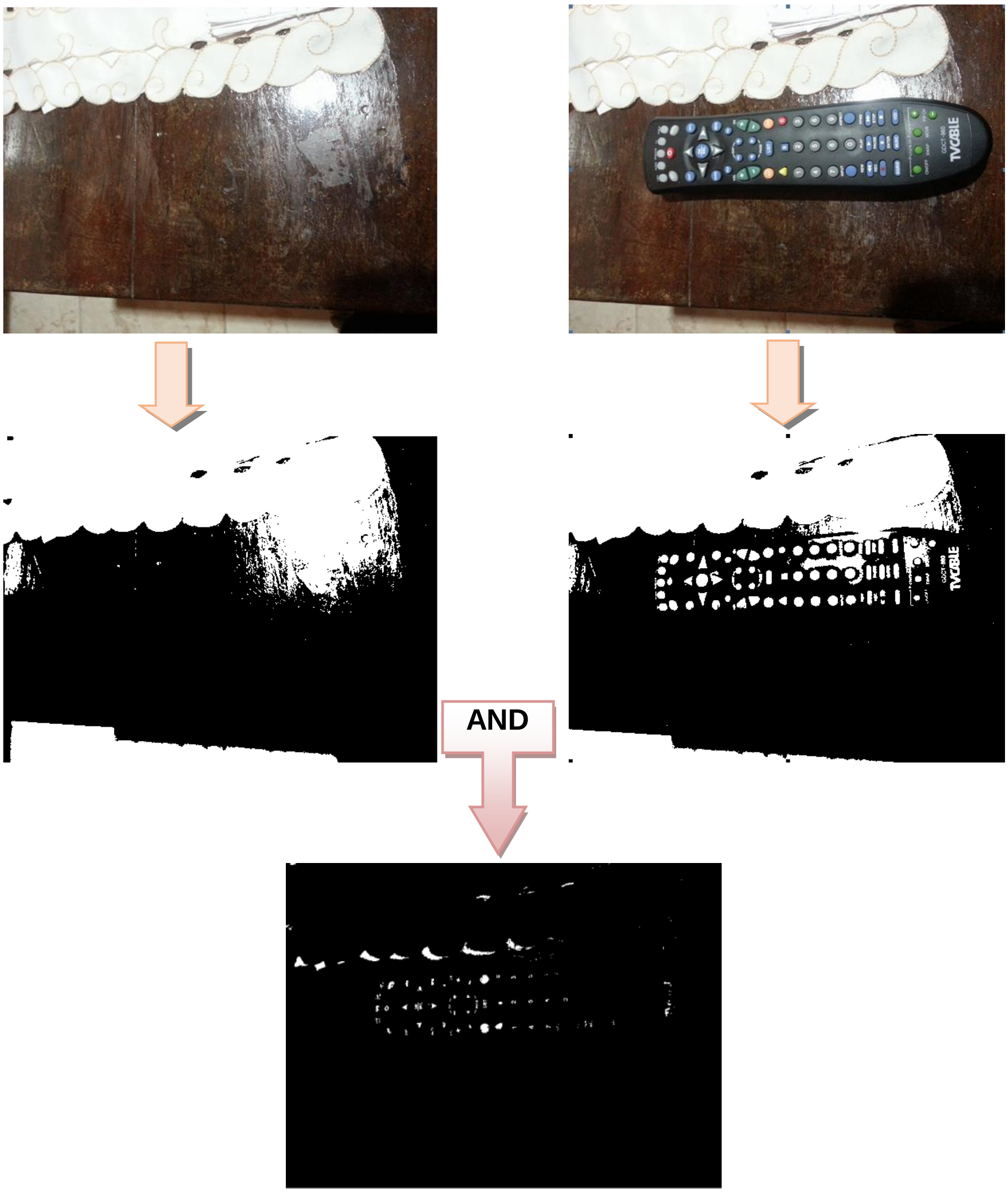

Figura 2.7. Resultado de Aplicar operación aritmética AND.

Operaciones Morfológicas: Los operadores morfológicos básicamente pueden usarse para: suavizar los bordes de una región, separar diferentes regiones de la imagen, unir regiones que hayan sido separadas durante la segmentación, facilitar el cálculo de regiones en una imagen. Estas operaciones son: Dilatación y Erosión.

La figura 2.8 muestra el ejemplo de aplicar el algoritmo de Erosión a una imagen de entrada que antes de procesarla se la binariza. La erosión consiste en examinar cada píxel y cambiarlo de 1 a 0 si alguno de sus pixeles vecinos está en 0. Se consideran pixeles vecinos a los 8 pixeles que rodean al pixel analizado [23].

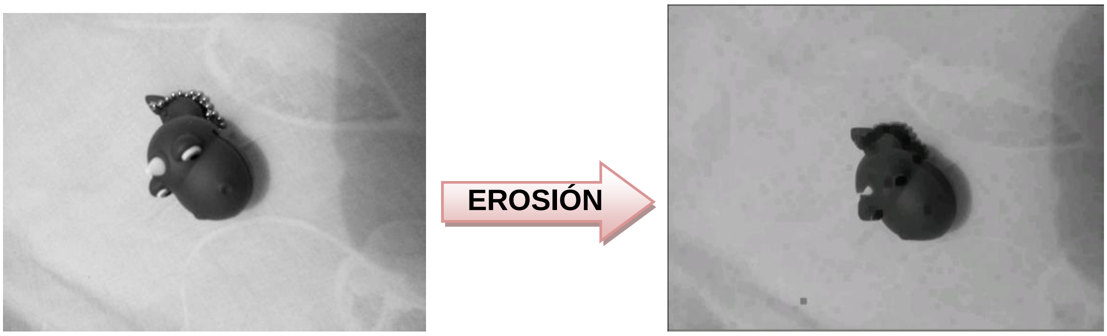

Figura 2.8. Resultado de Aplicar operación morfológica EROSIÓN.

Filtros Espaciales: Son usados típicamente para eliminar ruido, o para ejecutar algún tipo de realzado de imágenes. Los que se usan son: filtro medio, filtro mediano, filtro de realzado. La figura 2.9 muestra el resultado de aplicar el filtro mediano a una imagen. Su principal característica es eliminar el ruido “sal” que se manifiesta en forma de puntos con una intensidad muy distinta de la de la imagen y que aparecen de forma esporádica en distintas partes de la imagen. Cuando el filtro se centra sobre uno de estos valores excesivamente dispares del resto de sus vecinos, su valor será sustituido por la mediana de todos ellos, con lo cual se obtiene un valor más próximo a la intensidad local de la imagen.

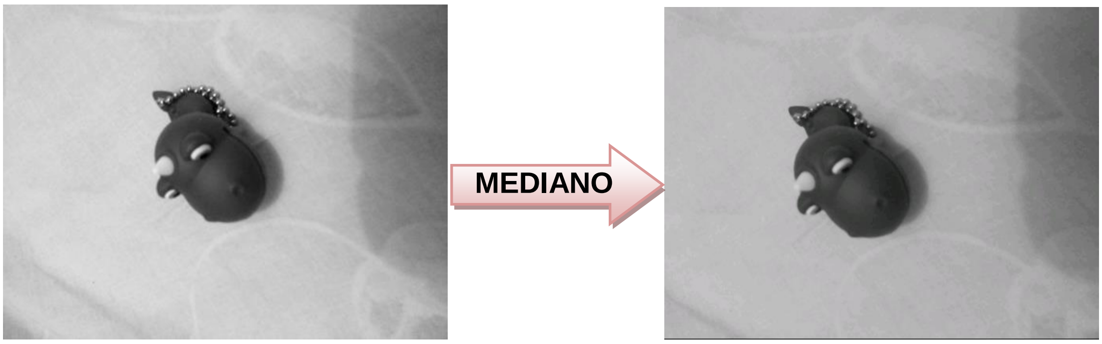

Figura 2.9. Resultado de Aplicar filtro espacial MEDIANO.

Cuantización de imágenes: Su función es reducir los datos de imagen eliminando alguna información de detalle, mapeando grupos de datos a un simple punto. Sus aplicaciones son reducción de niveles de grises, reducción espacial. Para conseguir esta cuantización se pueden seguir 3 vías: promediado, mediana, sub-muestreo (decimation). La Figura 2.10 muestra el resultado de aplicar el filtro promedio a una imagen. Esto consiste en asignar a un pixel específico el resultado de obtener el promedio de píxeles vecinos.

Figura 2.10. Resultado de Aplicar filtro PROMEDIO

2.6.2 Realzado de imágenes

Las técnicas de realzado de imágenes son usadas para enfatizar y resaltar características de imágenes para su posterior análisis y/o visualización. También es útil, donde la visualización humana es requerida antes del futuro procesamiento. Otra aplicación suele ser para mejorar la apariencia de una imagen.

Suavizado de imágenes: Estas operaciones son usadas para dar a una imagen un suavizado o efecto especial y/o para eliminar el ruido. Los algoritmos usados son: Filtros medio y mediano, filtrado paso-bajo. Ver Figura 2.11 y Figura 2.12.

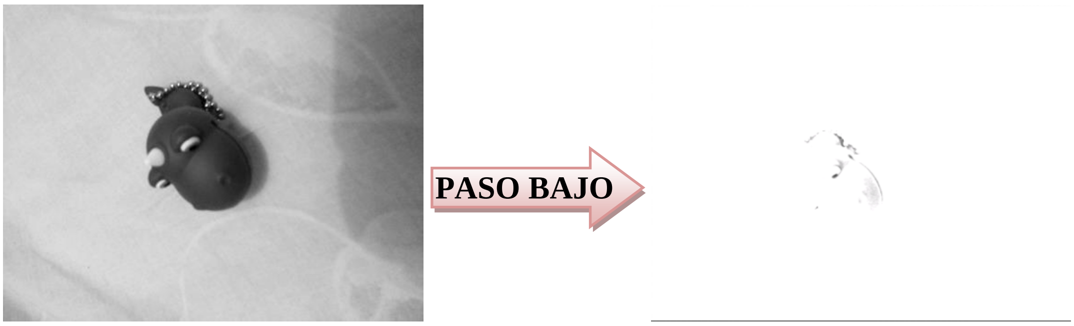

Figura 2.11. Resultado de Aplicar filtro PASO BAJO

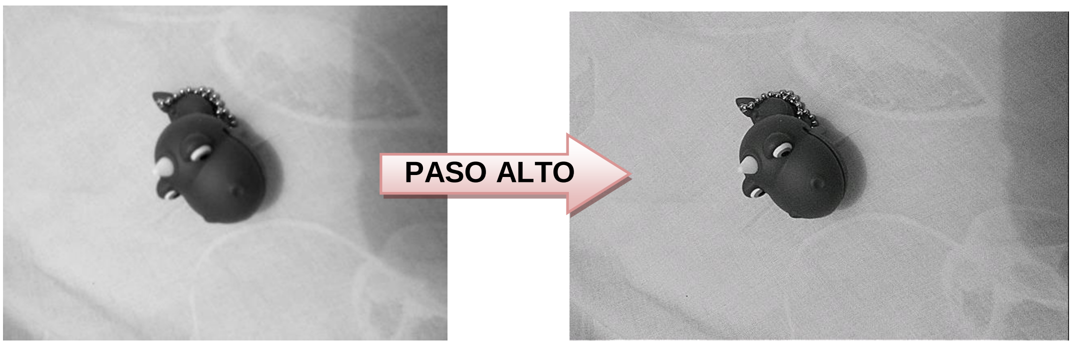

Figura 2.12. Resultado de Aplicar filtro PASO ALTO

2.7. Librería OpenCV

OpenCV es una librería de visión por computador desarrollada por Intel. Al ser de código abierto, se han producido frecuentes mejoras y especializaciones. Está escrita en C y C++, por tanto la portabilidad a varios sistemas operativos y plataformas no es compleja. Dado que dispositivos móviles en la actualidad poseen cámaras cada vez más potentes, y su uso se vuelve común y accesible, la comunidad de OpenCV ha desarrollado una especificación de su librería para plataforma Android.

La diferencia entre la versión para computadores y la versión para Android es que para esta última se ha desarrollado optimizaciones para que puedan ser ejecutadas sin ningún problema en arquitecturas de hardware ARM, etc. [14]. 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. El modelo de ejecución OpenCV es mostrado en la Figura 2.13. 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 [10]. 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.

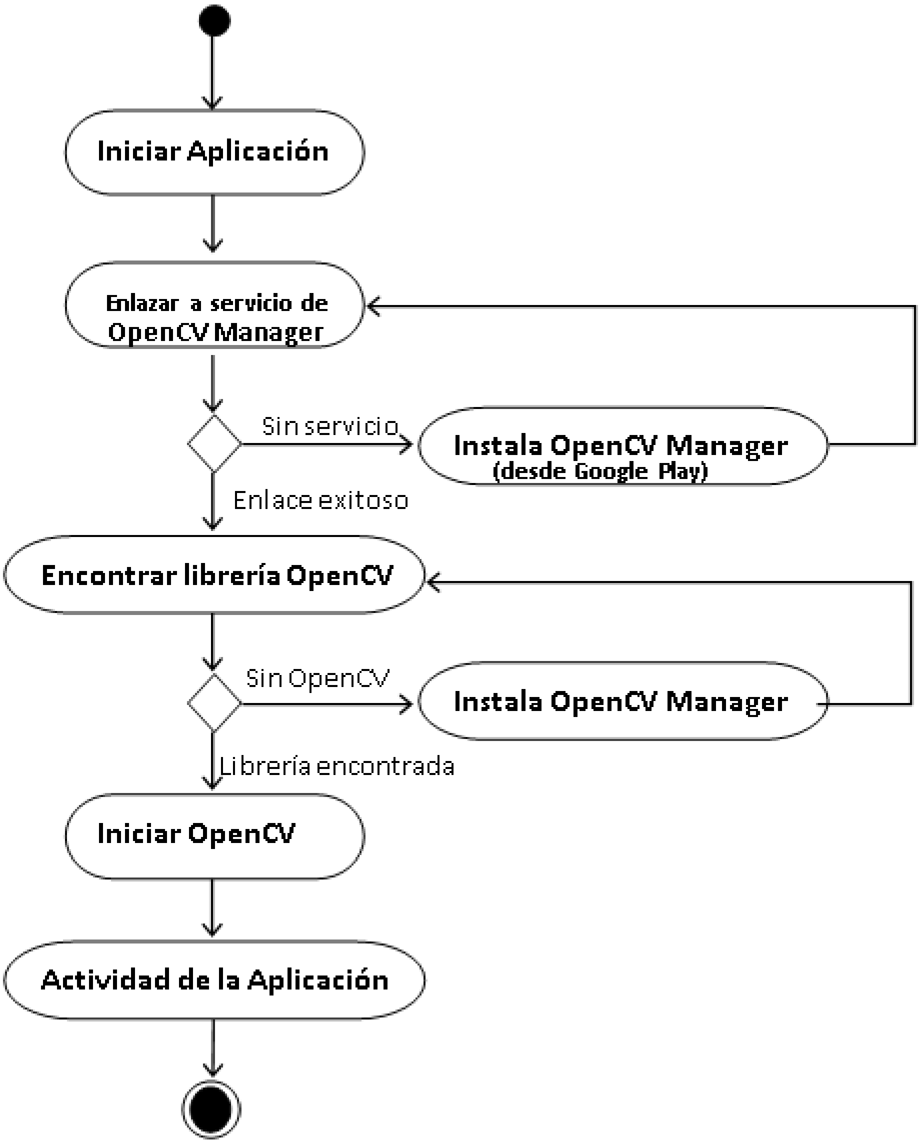

Figura 2.13. Modelo de ejecución de OpenCV en Android

CAPÍTULO 3. ANÁLISIS DE TIEMPOS DE ADQUISICIÓN DE IMÁGENES

El proceso de adquisición de imágenes es complejo en teléfonos móviles. Además de que es lento y depende en gran parte de las características de la cámara, de la calidad de imagen y de las características del teléfono. Debido a esto, se realiza el experimento para analizar el tiempo que le toma al teléfono móvil capturar una imagen de la cámara y convertirla a una imagen en el espacio RGB.

Para el experimento se procede de la siguiente manera:

1. Se define el número de imágenes que se desea capturar.
2. Se obtiene el tiempo inicial.
3. Se realiza la adquisición de la o las imágenes.
4. Se obtiene el tiempo final.
5. El tiempo de duración es la resta entre el tiempo final y el tiempo inicial.

En un inicio se planteó realizar el análisis de la adquisición de datos usando todas las herramientas planteadas. Sin embargo sólo fue posible utilizar las herramientas

OpenCV SDK, OpenCV NDK y Android SDK debido a que estas son las únicas que proveen un soporte claro para el acceso al sensor de la cámara.

La razón de peso para no tener soporte público y acceso nativo para la cámara y sus características en NDK es que el soporte actual resulta ineficiente y termina en procesos críticos del sistema [20].

3.1. Variables analizadas

El dato a analizar en esta sección es el tiempo en nanosegundos que se tarda en adquirir el conjunto de imágenes de la cámara, desde que se inicia la captura hasta que finaliza este proceso. Para esto, el experimento considera que la cámara del teléfono móvil se encuentra inicializada y operativa.

3.2. Datos obtenidos

Los resultados obtenidos del experimento se muestran en la Figura 3.1 y Figura 3.2. Para cada herramienta se realizaron 5, 10, 25, 50 y 100 capturas de imágenes y se muestran los datos de tiempo promedio así como de la desviación estándar que resulta de una misma prueba. Los datos obtenidos sin procesar se encuentran en el Anexo C, en la Tabla 10 para Android SDK y en la Tabla 11 para OpenCV.

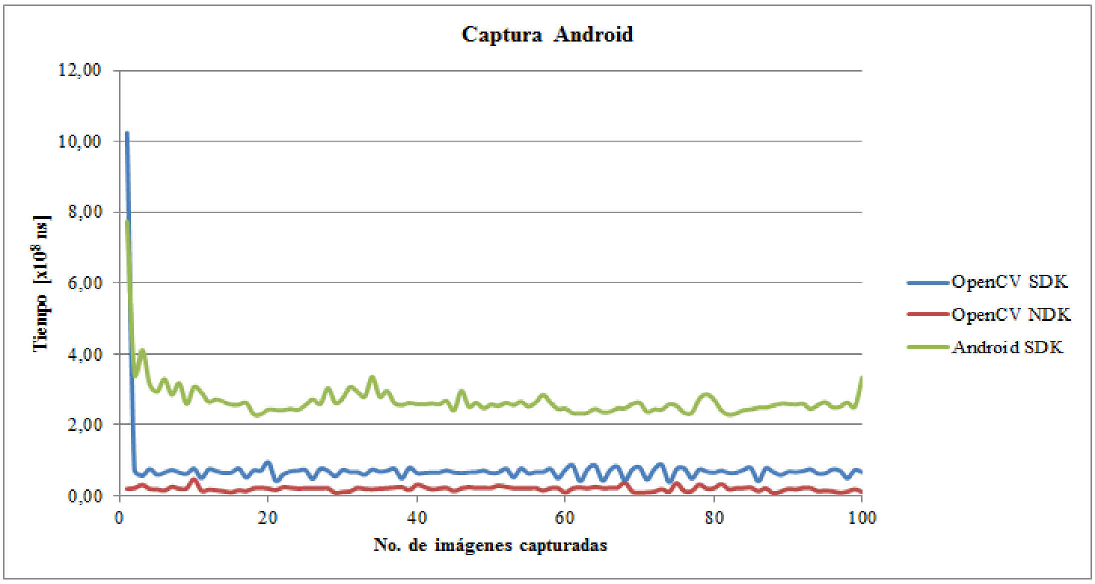

Figura 14. Comparación de promedios de captura de datos de la cámara.

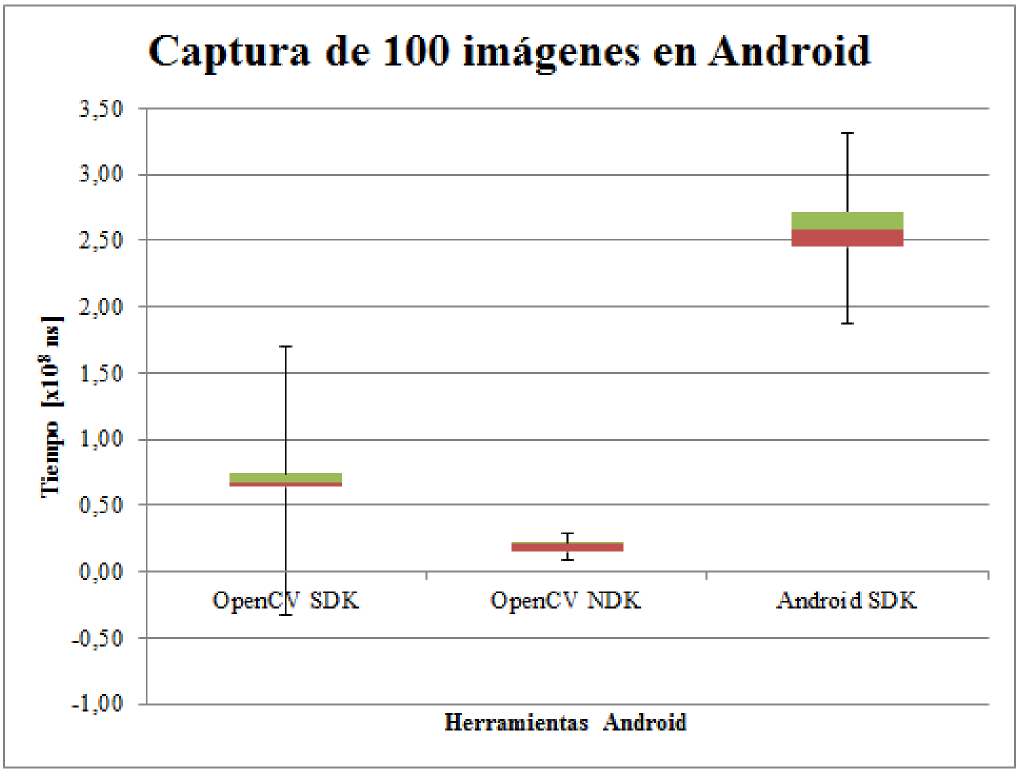

Figura 15. Diagrama de cajas para el tiempo usado en capturar 100 imágenes.

3.3. Análisis de Resultados

Este análisis se realizó en función de la cantidad de pruebas realizadas y el tiempo que tomó en cada una de ellas.

En la Figura 3.1 se muestra que al usar solamente Android SDK los tiempos en pruebas obtenidos son más elevados en comparación con los obtenidos con OpenCV en SDK y NDK. Se observa que, en la captura de imágenes usando los métodos de OpenCV para Android SDK se observa que los tiempos equivalen a la mitad de los tiempos en la implementación para SDK. El comportamiento de la curva es similar pero sus valores están entre 4 y 5 veces mayores que la implementación de OpenCV para SDK y aproximadamente 20 veces menores que la que se usó para NDK con OpenCV.

3.4. Conclusiones del análisis de adquisición

Se observa que el tiempo que toma el teléfono móvil al acceder a la cámara, es mucho menor si se establece un acceso con código nativo. El SDK de Android provee de herramientas para usar la cámara pero no es tan rápido como el acceso que provee la librería OpenCV SDK. Esta última, al ser una librería propiamente de procesamiento de imágenes toma en cuenta optimizaciones para evitar uso innecesario de recursos, [18]. OpenCV para Android cuenta con su propia API de Java, es decir no se basa en el API de Android [19] es por ello que el acceso a este sensor de la cámara es mucho más rápido. De nuestras pruebas se puede observar que el algoritmo para la adquisición usando OpenCV NDK toma menos tiempo.

En la Figura 3.2 se puede ver que el diagrama de cajas para el experimento para medir 100 capturas de imágenes para cada herramienta, los datos recolectados están en la Tabla 8 y los resultados de los cálculos estadísticos están en la Tabla 9 dentro del ANEXO B. De esta manera se verifica que la captura realizada por los algoritmos de OpenCV NDK resulta más rápida en función del tiempo que le toma en obtenerla. De este gráfico se analiza que tanto para OpenCV SDK y Android SDK los datos presentan una dispersión considerable, mientras que para OpenCV NDK se puede ver que están muy cercanos a la media, por tanto se puede observar que las capturas en su mayoría han demorado tiempos similares.

CAPÍTULO 4. ANÁLISIS DE TIEMPOS DE EJECUCIÓN DE ALGORITMOS DE PROCESAMIENTO DE IMÁGENES

En esta sección se analiza el tiempo que tarda un teléfono en ejecutar algoritmos básicos de procesamiento de imágenes a fin de conocer cuáles son las herramientas más apropiadas para el procesamiento de imágenes en tiempo real en teléfonos móviles.

Para el experimento se sigue el mismo esquema de pruebas que en el capítulo anterior.

- Se define el número de imágenes que se desea procesar.
- Se obtiene tiempo inicial
- Se ejecuta el algoritmo de procesamiento
- Se obtiene el tiempo final
- Se resta el tiempo final vs. el tiempo inicial.

4.1. Variables analizadas

Al igual que en la adquisición, la variable a analizar en este experimento es el tiempo en nanosegundos. Se toman los tiempos desde que se inicia el algoritmo hasta que finaliza, teniendo en cuenta que la cámara se encuentra inicializada y operativa. Cabe señalar que el experimento no considera el tiempo de adquisición de la imagen de la cámara.

4.2. Consideraciones

Debido a que existe una gran cantidad de algoritmos que pueden ser usados, se definieron tablas de prueba tal como se muestra en la Tabla 4 y Tabla 5, donde se indica con una “X” para denotar si la herramienta presenta soporte para el algoritmo, y con un “-” cuando no lo hace.

Tabla 4. Algoritmos de pre-procesamiento de imágenes.

PRE-PROCESAMIENTO				
Operación	**Algoritmo de Prueba**	**Android SDK**	**OpenCV SDK**	**OpenCV NDK**
Región de Interés	ROI	X	X	X
Operaciones Geométricas	Rotación	X	X	X
Operaciones Aritméticas	Resta	X	X	X
Operaciones Lógicas	And	X	X	X
Operaciones Morfológicas	Erosión	-	X	X
Filtros espaciales	Mediano	-	X	X
Cuantización de imágenes	Promedio	-	X	X

Tabla 5. Algoritmos de realzado de imágenes.

REALZADO DE IMÁGENES			
Operación	**Algoritmo de Prueba**	**OpenCV SDK**	**OpenCV NDK**
Afinamiento/Sharpening	Filtro Paso-Alto	X	X
Suavizado	Filtro Paso-Bajo	X	X

Es importante considerar que Android no da soporte alguno ni posee métodos para procesamiento de imágenes en su API, es por ello que ni SDK ni NDK implementan operaciones básicas de procesamiento de imágenes. Para poder realizar la comparación se revisaron métodos similares que se encuentran soportados por OpenCV y Android SDK.

OpenCV en su implementación del API para Android en C (NDK) y en Java (SDK) cuenta con soporte extenso para la gran mayoría de algoritmos de procesamiento de imágenes. Dentro del API Android SDK existe un conjunto de herramientas que manejan los distintos sensores del teléfono, de modo que cuando se menciona al Android SDK se hace referencia a la librería Android.Hardware.Camera que es la que maneja aspectos de la cámara del teléfono.

4.3. Datos obtenidos

Los datos obtenidos del experimento fueron organizados de acuerdo a la operación de procesamiento realizada. Se realizaron 5, 10, 25, 50, 100 operaciones de procesamiento de imágenes, de esta manera se tiene una mayor certeza en la estimación del comportamiento en términos de tiempo. Se tomaron mediciones para tiempo promedio y desviación estándar. El Anexo C contiene todos los datos obtenidos en el experimento.

Adicionalmente, en la Tabla 6 se muestran los resultados de 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, con el objetivo de poder analizar el comportamiento. En la Tabla 7 se muestran los resultados de pruebas con algoritmos de realzado, donde se efectúa el mismo análisis realizado con las operaciones de pre-procesamiento. En ambas tablas se muestran los tiempos promedios de cada prueba en nanosegundos y la desviación estándar.

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

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

En la Tabla 6 no se muestran datos del algoritmo de Erosión para Android SDK debido a que no existe una implementación nativa.

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

	REALZADO (x 10^6 ns) 100 IMAGENES			
	OpenCV SDK		OpenCV NDK	
FILTRO	$\overline{X}$	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

Adicionalmente, en el ANEXO B se especifican los resultados de las pruebas que se realizaron a 100 imágenes capturadas. Mientras que los datos de tiempos promedios para las operaciones de procesamiento de imágenes están detallados en ANEXO C.

La operación ROI permite obtener una porción de una imagen de entrada, este proceso tiene extenso soporte en varias herramientas y los resultados de las pruebas se muestran en el Anexo C en la Tabla 12, también podemos observar en la Figura 4.1 los datos obtenidos. Mientras que en la Tabla 13 se muestran las medidas que se tienen al iterar el algoritmo de rotación de imágenes, que al igual que el ROI tiene métodos definidos tanto en OpenCV como en el API de Android.

La figura 4.2 muestra comportamiento del tiempo de ejecución del algoritmo de Rotación, puede notarse similitud con el comportamiento del ROI.

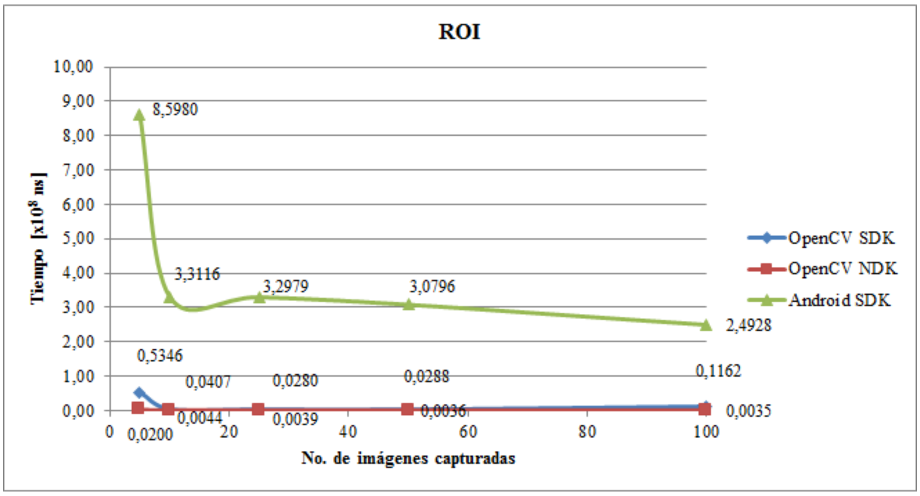

Figura 16. Tiempos de procesamiento de ROI (Región de Interés)

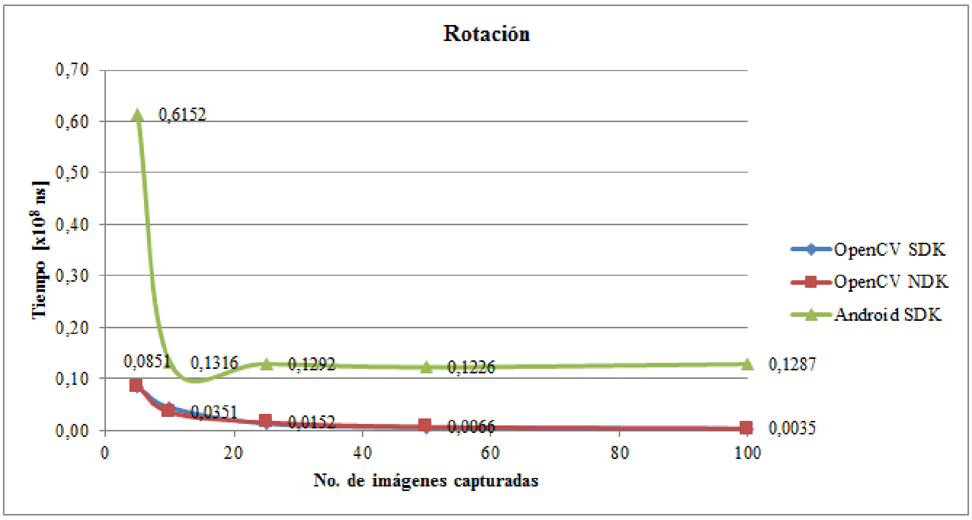

Figura 17. Tiempos de procesamiento de ROTACIÓN.

Las operaciones que se realizan con imágenes binarias como Resta, AND y Erosión requieren de cálculos más complejos, por ende su costo computacional es bastante elevado en un dispositivo limitado como un teléfono móvil. Los datos adquiridos de las mediciones están resumidos en el ANEXO C en Tabla 14 para Resta, Tabla 15 para AND y Tabla 16 para Erosión. Gráficamente podemos ver el comportamiento de estos algoritmos en Figura 4.3, Figura. 4.4 y Figura 4.5 respectivamente.

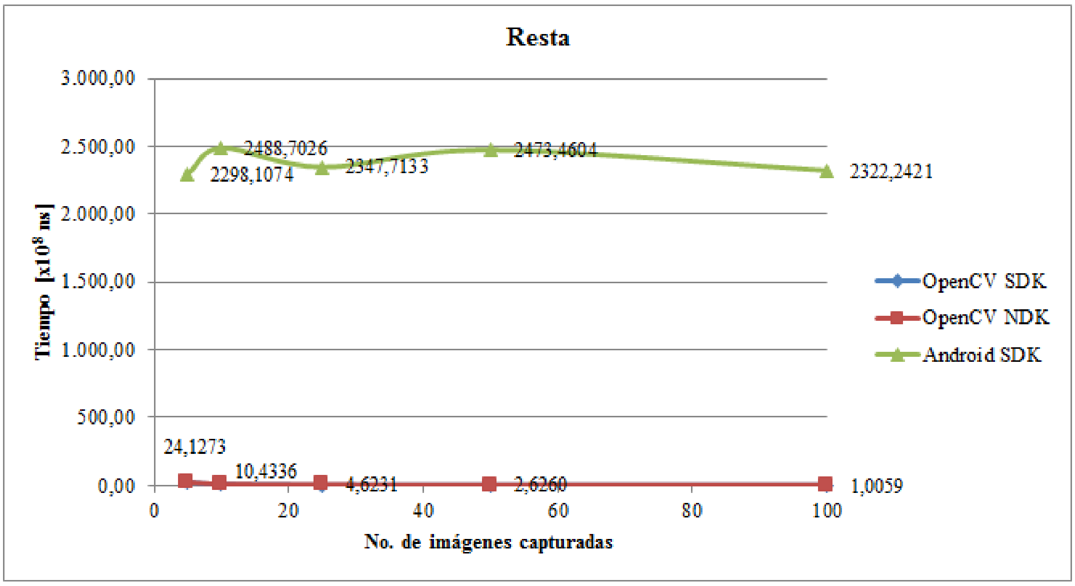

Figura 18. Tiempos de procesamiento de RESTA de Imágenes

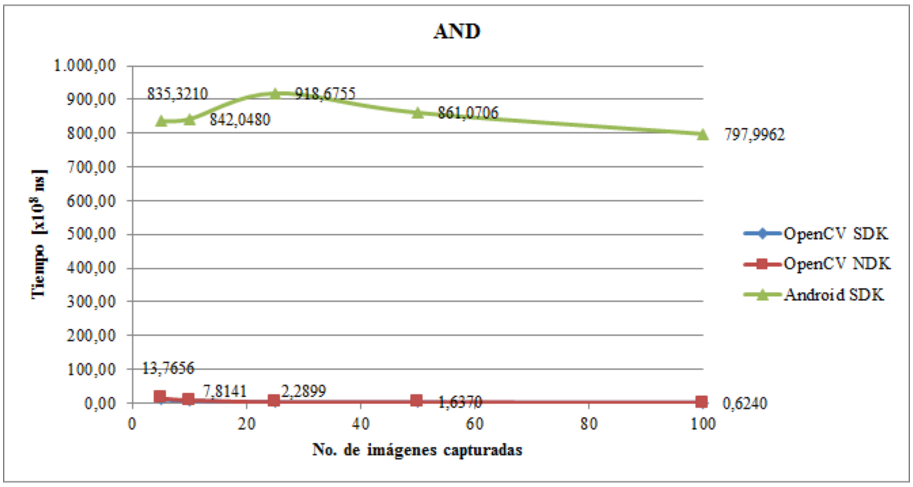

Figura 19. Tiempos de procesamiento de AND de Imágenes

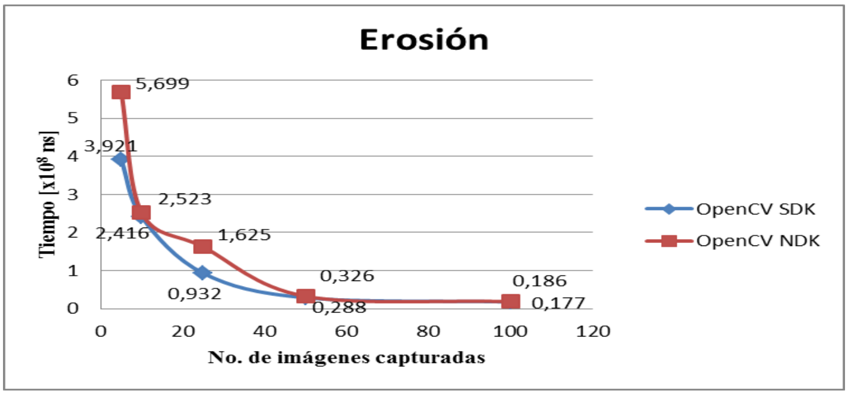

Figura 20. Tiempos de procesamiento de ERODE

La Tabla 17 para Filtro Mediano y Tabla 18 para Filtro Promedio muestran los datos de los promedios de tiempo recolectados al aplicar dichos filtros. La característica de estas operaciones es que recorren pixel por pixel para obtener valores y según esos procesar la imagen de entrada. Por ende se trata de algoritmos un poco complejos pero que son realizables usando métodos y funciones de OpenCV. Los gráficos obtenidos los mostramos en la Figura 4.6 y Figura 4.7.

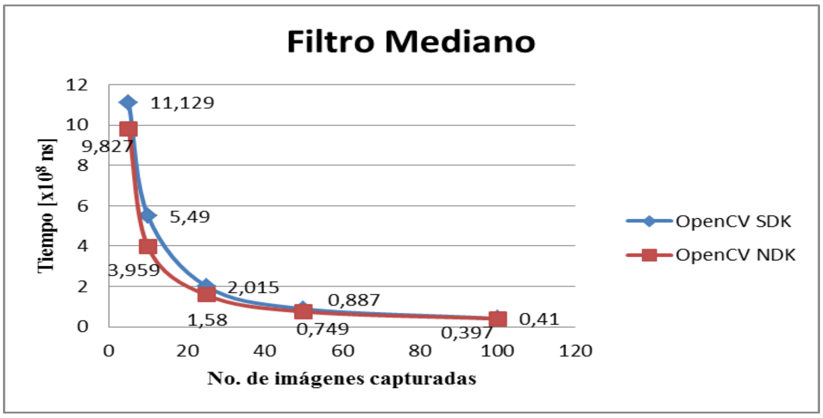

Figura 21. Tiempos de procesamiento de FILTRO MEDIANO

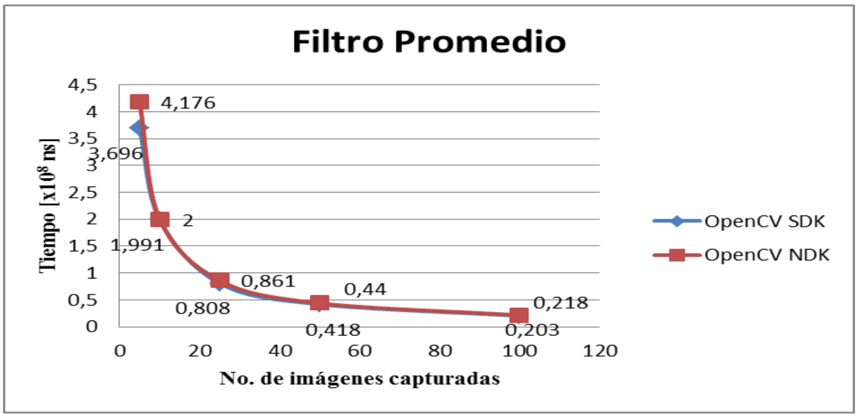

Figura 22 Tiempos de procesamiento de FILTRO PROMEDIO

Para las operaciones de realzado de imágenes se eligió hacer pruebas con filtros Paso-Alto y Paso-Bajo debido a que las cámaras de los teléfonos suelen capturar imágenes con demasiado ruido, ya sea por defectos físicos o ambientales de captura, entonces al ser sometidos a un pre-procesamiento el resultado no es muy óptimo, por ende los realzados con estos filtros son muy necesarios. Estos resultados están tabulados en Tabla 19 para Paso-Alto y Tabla 20 para Paso-Bajo. Sus gráficos corresponden a Figura 4.8. y a la Figura 4.9.

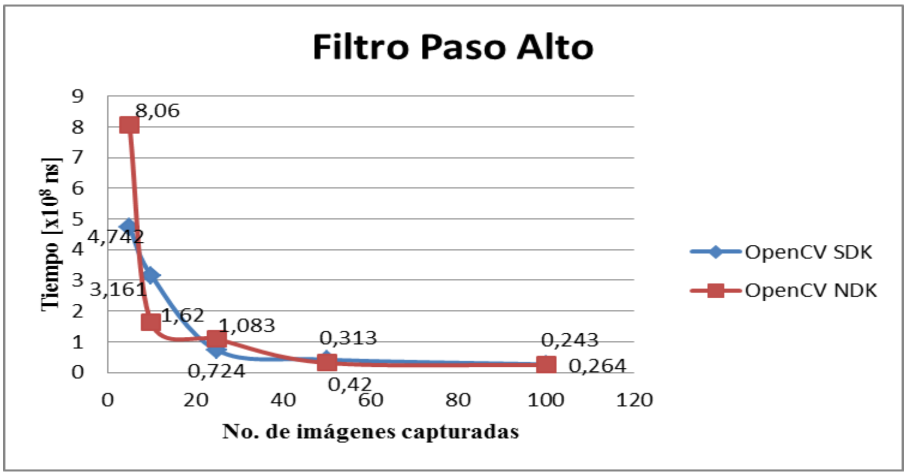

Figura 23. Tiempos de procesamiento de FILTRO PASO-ALTO

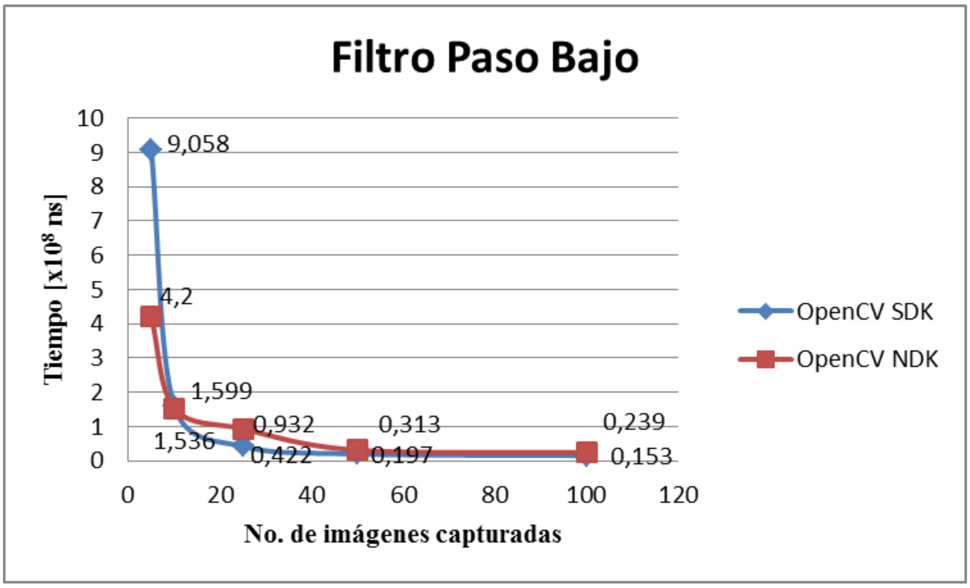

Figura 24. Tiempos de procesamiento de FILTRO PASO-BAJO

4.4. Conclusiones del análisis de procesamiento

De los experimentos realizados se puede afirmar que es factible realizar operaciones de procesamiento de imágenes con un teléfono móvil que ejecuta sistema operativo Android.

Del análisis presentado gráfica a gráfica se puede ver que en algoritmos de procesamiento de imágenes es mucho mejor usar herramientas que ya están optimizadas para ello, como OpenCV para Android. Pues el coste computacional que presenta una re-implementación de algoritmos y estructuras es muy alto.

De la serie de experimentos se concluye que usar la implementación de la librería OpenCV, tanto para SDK como para NDK, resulta más óptimo en el área de visión por computador en términos de tiempo.

Lo que podría ser sujeto de serio análisis en futuras investigaciones, es garantizar que los algoritmos que usen el sensor de la cámara no se ejecuten al iniciar la captura, sino más bien después de un tiempo prudencial para que los retrasos y ralentizaciones iniciales que se evidencian en el análisis de esta sección, se reduzcan en lo posible.

4.5. Análisis de Resultados

Mediante la utilización de gráficos se puede efectuar el análisis en función de la cantidad de pruebas y el tiempo que se tardó en cada una de ellas. La Figura 4.1 muestra que para obtener el ROI de una imagen, en las herramientas analizadas la más lenta es la que usa el SDK de Android, y con un comportamiento muy similar las de OpenCV. En esta gráfica se puede notar que la que usa OpenCV con NDK es más rápida para calcular el ROI que la que usa el OpenCV en SDK. Similar comportamiento presenta la Figura 4.2 en el cálculo de la Rotación.

En visión por computador para hacer alguna modificación a una imagen se la toma como una matriz de pixeles, de ahí que la rotación es un proceso de aplicar una matriz de rotación a la imagen inicial. Se puede observar que la implementación de la librería Android SDK es mucho más lenta que las de OpenCV que son casi iguales. La Figura 4.3 y la Figura. 4.4 muestran comportamiento de algoritmos en las que se hace uso de operaciones mayormente entre imágenes binarias. En el caso particular de la resta se obtiene la diferencia entre una imagen inicial y una imagen final. Este proceso es sencillo y se puede ver por la gráfica que OpenCV en SDK es ligeramente más rápido que la implementación en OpenCV SDK, y mucho más rápido que la de Android SDK.

Para la operación lógica AND, lo que sucede es similar al caso anterior, pues entre la implementación para OpenCV de SDK y NDK se ven tiempos más rápidos que usando la implementación particular de Android SDK.

Para los algoritmos de Erosión y Filtro Mediano mostrados en las Figura 4.5 y Figura 4.6 respectivamente, no se realizaron pruebas usando alguna implementación del API SDK o NDK, debido a que se planteaba analizar los métodos existentes que usaban estas herramientas para ello, pero SDK de Android y el NDK no presentan métodos o funciones que faciliten estas implementaciones. Mientras que la implementación de estos algoritmos usando OpenCV si presentaba soporte, por lo tanto, se analizó OpenCV SDK y OpenCV NDK. Tanto en el algoritmo de Erosión como en el algoritmo de los filtrados, se observa que a menores pruebas, se tiene un mayor tiempo de ejecución. En erosión la implementación de OpenCV para NDK resulta llegar un poco más abajo en la escala del tiempo, es decir es ligeramente más rápida que la que usa OpenCV en SDK. Es decir, cuantas más imágenes se procesan, menos diferencias en tiempo de procesamiento presentan. Casi el mismo comportamiento se replica en el algoritmo de Filtro Mediano como muestra la Figura

4.6. La Figura 4.7 muestra el comportamiento del algoritmo de Filtro Promedio, se observa que es similar al que se obtiene al ejecutar el de erosión y el del Filtro Medio, con la diferencia que en este caso es un poco más rápido la implementación de SDK que la de NDK en OpenCV.

En operaciones de realzado de imágenes es muy común el uso de filtros como el Filtro Paso-Bajo y el Filtro Paso-Alto cuyo comportamiento se muestra en la Fig.4.8 y Figura 4.9. Para Filtro Paso-Alto, se puede observar que a mayor cantidad de pruebas se vuelve un poco más rápido la implementación de Android para OpenCV NDK. Mientras que para el Filtro Paso-Bajo Android OpenCV en SDK es un poco más rápido.

Todos las operaciones de pre-procesamiento y realzado, se ejecutaron en imágenes previamente capturadas, y sin el uso de máscaras. En el ANEXO D se muestran las figuras de los diagramas de cajas correspondientes a los algoritmos analizados.

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.

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.

ANEXOS

ANEXO A

Experimentos Android SDK

Método: calcularTiempos()					
Paquete	edu.realtime .rtcapturasdk	edu.realtime .rtroisdk	edu.realtime .rtrotacionsdk	edu.realtime .rtrestasdk	edu.realtime .rtandsdk
Clase	CapturaActivity.java	RoiActivity.java	RotacionActivity.java	RestaActivity.java	AndActivity.java

```java
//Metodo para calcular tiempos Minimos, Maximos y Promedios
private void calcularTiempos() {
	long tiempoMaximo = 0;
	long tiempoMinimo = 0;
	long tiempoPromedio = 0;

	if (!tiempos.isEmpty()) {
		tiempoMaximo = 0;
		tiempoMinimo = tiempos.get(0);
		tiempoPromedio = 0;

		for (int i = 0; i < tiempos.size(); i++) {
			if (tiempos.get(i) > tiempoMaximo) {
				tiempoMaximo = tiempos.get(i);
			} else if (tiempos.get(i) < tiempoMinimo) {
				tiempoMinimo = tiempos.get(i);
			}
			Log.i(TAG, "******* Arreglo :" + tiempos.get(i));

			tiempoPromedio += tiempos.get(i);
		}
		tiempoPromedio = tiempoPromedio / tiempos.size();
	}
	Log.i(TAG, "******* Minimo :" + tiempoMinimo);
	Log.i(TAG, "******* Maximo :" + tiempoMaximo);
	Log.i(TAG, "******* Promedio :" + tiempoPromedio);

	this.tlblTiempoMin.setText(tiempoMinimo + "");
	this.tlblTiempoMax.setText(tiempoMaximo + "");
	this.tlblTiempoProm.setText(tiempoPromedio + "");
	this.tlblFrames.setText(tiempos.size() + "");
}
```

Método: checkCameraHardware(Context context)					
Paquete	edu.realtime .rtcapturasdk	edu.realtime .rtroisdk	edu.realtime .rtrotacionsdk	edu.realtime .rtrestasdk	edu.realtime .rtandsdk
Clase	CapturaActivity.java	RoiActivity.java	RotacionActivity.java	RestaActivity.java	AndActivity.java

```
/** Funcion para verificar si existe camara */
private boolean checkCameraHardware(Context context) {
      if (context.getPackageManager().hasSystemFeature(
            PackageManager.FEATURE_CAMERA)) {
            return true;
      } else {
            return false;
      }
}
```

Método: onCreate(Bundle savedInstanceState)	
Paquete	edu.realtime.rtcapturasdk
Clase	CapturaActivity.java

```
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      // Inicializar controles de Layout
      this.tlblEstado = (TextView)
super.findViewById(R.id.ltxt_estado);
      this.tlblTiempoMax = (TextView) super
            .findViewById(R.id.ltxt_tiempo_max);
            this.tlblTiempoMin = (TextView) super
                        .findViewById(R.id.ltxt_tiempo_min);
            this.tlblTiempoProm = (TextView) super
                        .findViewById(R.id.ltxt_tiempo_prom);
            this.tlblFrames = (TextView)
super.findViewById(R.id.ltxt_frames);

            // Verificamos ausencia o no de camaras
            if (!checkCameraHardware(this)) {
                  this.tlblEstado.setText("No hay camara");
                  return;
            } else {
                  this.tlblEstado.setText("Camara detectada.");

                  mVideoCaptureView = (SurfaceView)
                  findViewById(R.id.camera_preview);
                  SurfaceHolder videoCaptureViewHolder =
mVideoCaptureView
                              .getHolder();
                  videoCaptureViewHolder

      .setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

            }
      }
```

Método: onCreate(Bundle savedInstanceState)	
Paquete	edu.realtime.rtcapturasdk
Clase	CapturaActivity.java

```
// Inicializar captura de imagenes
private void startVideo() {
	this.tblEstado.setText("Capturando...");

	SurfaceHolder videoCaptureViewHolder = null;
	try {
		mCamera = Camera.open(); // abrir camara
	} catch (RuntimeException e) {
		Log.e("CameraTest", "Camera Open filed");
		return;
	}
	mCamera.setErrorCallback(new ErrorCallback() {
		public void onError(int error, Camera camera) {
		}
	});

	if (null != mVideoCaptureView)
		videoCaptureViewHolder = mVideoCaptureView.getHolder();
	try {
		mCamera.setPreviewDisplay(videoCaptureViewHolder);
	} catch (Throwable t) {
	}

	//Inicializar Buffer de Camara
	Log.v("CameraTest", "Camera PreviewFrameRate = "
				+
mCamera.getParameters().getPreviewFrameRate());
	Size previewSize = mCamera.getParameters().getPreviewSize();
	int dataBufferSize = (int) (previewSize.height *
previewSize.width *
(ImageFormat.getBitsPerPixel(mCamera.getParameters().getPreviewForma
t()) / 8.0));
	mCamera.addCallbackBuffer(new byte[dataBufferSize]);
	mCamera.addCallbackBuffer(new byte[dataBufferSize]);
	mCamera.addCallbackBuffer(new byte[dataBufferSize]);
	mCamera.setPreviewCallbackWithBuffer(new
Camera.PreviewCallback() {
		//Iniciar Captura de Tiempo
		private long timestamp = System.nanoTime();
		private long tiempoDuracion = 0;
	public synchronized void onPreviewFrame(byte[] data, Camera
camera) {

	//Obtenemos y mostramos imagen de entrada parseandola de YUV a
JPEG
		Camera.Parameters parameters = camera.getParameters();
		Size size = parameters.getPreviewSize();
		YuvImage image = new YuvImage(data,
	parameters.getPreviewFormat(), size.width, size.height, null);
```

```
		ByteArrayOutputStream outStream = new
	ByteArrayOutputStream();
		image.compressToJpeg(new Rect(0, 0, size.width,
	size.height), 100, outStream); // make JPG

		imgEntrada =
	BitmapFactory.decodeByteArray(outStream.toByteArray(), 0,
	outStream.size());

		// una vez terminada la compresion se termina de medir
	antes de presentar en pantalla

		tiempoDuracion = (System.nanoTime() - timestamp);
				tiempos.add((tiempoDuracion));

		Log.v("CameraTest", "Time Gap = " + tiempoDuracion);

		// Terminar medida de tiempo
		timestamp = System.nanoTime();
		try {
			camera.addCallbackBuffer(data);
		} catch (Exception e) {
			Log.e("CameraTest", "addCallbackBuffer error");
			return;
		}
		if (tiempos.size() >= noPruebas ) {
			stopVideo();
			return;
		}
		return;
		}
	});
	try {
		mCamera.startPreview();
	} catch (Throwable e) {
		mCamera.release();
		mCamera = null;
		return;
	}
}
```

Método: onCreate(Bundle savedInstanceState)				
Paquete	edu.realtime .rtroisdk	edu.realtime .rtrotacionsdk	edu.realtime .rtrestasdk	edu.realtime .rtandsdk
Clase	RoiActivity.java	RotacionActivity.java	RestaActivity.java	AndActivity.java

```
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		//Inicializacion de controles
		this.tlblEstado = (TextView)
```

```
super.findViewById(R.id.ltxt_estado);
            this.tlblTiempoMax = (TextView) super
                        .findViewById(R.id.ltxt_tiempo_max);
            this.tlblTiempoMin = (TextView) super
                        .findViewById(R.id.ltxt_tiempo_min);
            this.tlblTiempoProm = (TextView) super
                        .findViewById(R.id.ltxt_tiempo_prom);
            this.tlblFrames = (TextView)
super.findViewById(R.id.ltxt_frames);

            this.imgViewEntrada = (ImageView)
findViewById(R.id.imagen_entrada);
            this.imgViewSalida = (ImageView)
findViewById(R.id.imagen_salida);

            // Verificamos ausencia o no de camaras
            if (!checkCameraHardware(this)) {
                  this.tlblEstado.setText("No hay camara");
                  return;
            } else {
                  this.tlblEstado.setText("Camara detectada.");
            }
      }
```

Método: void startVideo()	
Paquete	edu.realtime.rtroisdk
Clase	RoiActivity.java

```
//*** Funcion para Capturar imagenes y hacer procesamiento
private void startVideo() {
      this.tlblEstado.setText("Capturando...");
      try {
            mCamera = getCameraInstance();
      } catch (RuntimeException e) {
            Log.e(TAG, "Camera Open filed");
            return;
      }
      mCamera.setErrorCallback(new ErrorCallback() {
            public void onError(int error, Camera camera) {
            }
      });

      try {
            mCamera.setPreviewCallback(new Camera.PreviewCallback() {
            @Override
            public void onPreviewFrame(byte[] data, Camera camera) {
                  if (camera != null) {
                        try {
                        //Obtenemos y mostramos imagen de entrada
                        parseandola de YUV a JPEG
                        Camera.Parameters parameters =
                        camera.getParameters();
                        Size size = parameters.getPreviewSize();
```

```
                    YuvImage image = new YuvImage(data,
                    parameters.getPreviewFormat(), size.width,
                    size.height, null);
                    ByteArrayOutputStream outStream = new
                    ByteArrayOutputStream();
                    image.compressToJpeg(new Rect(0, 0,
                    size.width, size.height), 100, outStream); //
                    make JPG

                imgEntrada =
                BitmapFactory.decodeByteArray(outStream.toByteArray
                (), 0, outStream.size());

                imgViewEntrada.setImageBitmap(imgEntrada);
                if(medirTiempo == true){
                    Log.d(TAG, "Entra a capturar tiempos");
                    tlblEstado.setText("Capturando tiempos...");
                    //Iniciar Aplicacion de Algoritmo
                    tiempoInicial = System.nanoTime();
                    //  Aplicacion de Roi
                    imgSalida = Bitmap.createBitmap(imgEntrada,
                50, 50, 400, 400);

            imgViewSalida.setImageBitmap(imgSalida);
            imgViewSalida.setScaleType(ScaleType.CENTER);
            tiempoDuracion = (System.nanoTime() - tiempoInicial);
            //Termina Aplicacion de Algorimo
            tiempos.add(tiempoDuracion);
            Log.d(TAG, "Captura tiempo");
                if (tiempos.size() >= noPruebas) {
                    medirTiempo = false;
                    tiempoInicial = 0;
                    tiempoDuracion = 0;
                    tlblEstado.setText("Termino captura
                tiempos");
                    calcularTiempos();
                }
            }
        } catch (Exception e) {
                e.printStackTrace();
                Log.e(TAG, "Error en el callback preview");
            }
        } else { Log.e(TAG, "camara cerrada"); }
      }
});
      mCamera.startPreview();
      e.printStackTrace();
  }
}
      } catch (Throwable e) {}
}
```

Método: void startVideo()	
Paquete	edu.realtime.rtrotacionsdk
Clase	RotacionActivity.java

```
private void startVideo() {
this.tlblEstado.setText("Capturando...");

try {
	mCamera = getCameraInstance();
} catch (RuntimeException e) {
	Log.e(TAG, "Camera Open filed");
	return;
}
mCamera.setErrorCallback(new ErrorCallback() {
	public void onError(int error, Camera camera) {
	}
});

try {
	mCamera.setPreviewCallback(new Camera.PreviewCallback() {
		@Override
		public void onPreviewFrame(byte[] data, Camera camera) {
			if (camera != null) {
				try {
				//Obtenemos y mostramos imagen de entrada
				parseandola de YUV a JPEG
				Camera.Parameters parameters =
				camera.getParameters();
				Size size = parameters.getPreviewSize();

				// Obtener frame de entrada
				YuvImage image = new YuvImage(data,
				parameters.getPreviewFormat(), size.width,
				size.height, null);
				ByteArrayOutputStream outStream = new
				ByteArrayOutputStream();

				image.compressToJpeg(new Rect(0, 0,
				size.width, size.height), 100, outStream); //
				make JPG

				imgEntrada =
				BitmapFactory.decodeByteArray(outStream.toByt
				eArray(), 0, outStream.size());

				imgViewEntrada.setImageBitmap(imgEntrada);

				if(medirTiempo == true){

					Log.d(TAG, "Entra a capturar tiempos");
					tlblEstado.setText("Capturando
				tiempos...");
				// Calculo de tiempo Inicial
				tiempoInicial = System.nanoTime();
```

```
                    int width = imgEntrada.getWidth();
                    int height = imgEntrada.getHeight();
                    int newWidth = 300;
                    int newHeight = 300;

                    // calcular escala
                    float scaleWidth = ((float) newWidth) /
                    width;
                    float scaleHeight = ((float) newHeight)/
                    height;
                    // Crear Rotacion
                    Matrix matrix = new Matrix();
                    matrix.postScale(scaleWidth, scaleHeight);
                    matrix.postRotate(45);
                    //calculo de tiempo final
                    tiempoDuracion = (System.nanoTime() -
                  tiempoInicial);
                    tiempos.add(tiempoDuracion);
                    Log.d(TAG, "Captura tiempo");
                    // Presentar Imagen en layout
                     imgSalida = Bitmap.createBitmap(imgEntrada,
                    0, 0,width, height, matrix, true);

                    BitmapDrawable bmd = new
                    BitmapDrawable(imgSalida);

                    imgViewSalida.setImageDrawable(bmd);
                    imgViewSalida.setScaleType(ScaleType.CENTER);

                    // Calcular tiempos
                    if (tiempos.size() >= (noPruebas + delta)) {
                       medirTiempo = false;
                       tiempoInicial = 0;
                       tiempoDuracion = 0;
                       tlblEstado.setText("Termino captura
                    tiempos");
                       calcularTiempos();
                       stopVideo();
                         return;
                    }
               }
          } catch (Exception e) {
               e.printStackTrace();
               Log.e(TAG, "Error en el callback preview");
          }
     } else { Log.e(TAG, "camara cerrada"); }
     }
});
mCamera.startPreview();
} catch (Throwable e) { e.printStackTrace();
// mCamera.release(); mCamera = null; return;
}
}
```

Método: Bitmap toGrayscale(Bitmap bmpOriginal)		
Paquete	edu.realtime.rtrestasdk	edu.realtime.rtandsdk
Clase	RestaActivity.java	AndActivity.javz

```
      // Conversion de Bitmap a Escala de Grises
public Bitmap toGrayscale(Bitmap bmpOriginal) {
      final int height = bmpOriginal.getHeight();
      final int width = bmpOriginal.getWidth();

      final Bitmap bmpGrayscale = Bitmap.createBitmap(width, height,
Bitmap.Config.ARGB_8888);
      final Canvas c = new Canvas(bmpGrayscale);
      final Paint paint = new Paint();
      final ColorMatrix cm = new ColorMatrix();
      cm.setSaturation(0);
      final ColorMatrixColorFilter f = new
      ColorMatrixColorFilter(cm);
              paint.setColorFilter(f);
              c.drawBitmap(bmpOriginal, 0, 0, paint);
              return bmpGrayscale;
}
```

Método: void startVideo()	
Paquete	edu.realtime.rtrestasdk
Clase	RestaActivity.java

```
// Funcion para capturar imagenes
private void startVideo() {
      this.tlblEstado.setText("Capturando...");

      try {
            mCamera = getCameraInstance();
      } catch (RuntimeException e) {
            Log.e(TAG, "Camera Open filed");
            return;
      }

      try {
            mCamera.setErrorCallback(new ErrorCallback() {
                  public void onError(int error, Camera camera) {
                  }
            });
            mCamera.startPreview();

            mCamera.setPreviewCallback(new Camera.PreviewCallback() {
                  @Override
                  public void onPreviewFrame(byte[] data, Camera
camera) {
                        if (camera != null) {
                              try {
                                    //Obtenemos y mostramos imagen de
                                    entrada parseandola de YUV a JPEG
                                    Camera.Parameters parameters =
```

```
                                    camera.getParameters();
                                    Size size =
parameters.getPreviewSize();
                        YuvImage image = new YuvImage(data,
                        parameters.getPreviewFormat(), size.width,
                        size.height, null);
                        ByteArrayOutputStream outStream = new
                        ByteArrayOutputStream();
                        image.compressToJpeg(new Rect(0, 0,
                        size.width, size.height), 100, outStream); //
                        make JPG

                        if(tCapturarBase > 0){
                              // Captura de Imagen Base:Imagen Fija
                              con la que se restara las demas
                              imagenes capturadas

                              tlblEstado.setText("Capturando
        base...");
                                    imgEntrada =
                                    BitmapFactory.decodeByteArray(out
                                    Stream.toByteArray(), 0,
                                    outStream.size());
                                    imgViewEntrada.setImageBitmap
                                    (toGrayscale(imgEntrada));
                                    imgViewEntrada.setScaleType
                                    (ScaleType.CENTER);
                                    tCapturarBase --;
                                    medirTiempo = true;
                              }else if(tCapturarBase <= 0 &&
                              medirTiempo == true){
                              tlblEstado.setText("Realizando
pruebas...");
                              Bitmap imgS =
                              BitmapFactory.decodeByteArray(outStream
                              .toByteArray(), 0, outStream.size());

                              //Convertir Imagen a Escala de gris
                  Bitmap imgEntradaGris = toGrayscale(imgEntrada);
                  Bitmap imgSalidaGris = toGrayscale(imgS);

                  //////// empezar captura de tiempos
                  tiempoInicial = System.nanoTime();
                  imgSalida = Bitmap.createBitmap(imgS.getWidth(),
              imgS.getHeight(), Bitmap.Config.ARGB_8888);
                        try{
                             //Incicio de Resta de Imagenes
                              for (int j = 0; j <
                        imgEntradaGris.getHeight(); j++){
                                    for (int i = 0; i <
                              imgEntradaGris.getWidth(); i++){
                              imgSalida.setPixel(i, j,
                              (imgEntradaGris.getPixel(i, j) -
                              imgSalidaGris.getPixel(i, j)));
```

```
                                    }
                            }//Fin Resta de Imagenes
                    }catch (Exception ex){
                            //MessageBox.Show(ex.Message);
                    }

                    tiempoDuracion = (System.nanoTime() -
                    tiempoInicial);
                    tiempos.add((tiempoDuracion));
                            //////// finalizar captura de tiempos

                            if(tiempos.size() == noPruebas){
                                    medirTiempo = false;
                                    stopVideo();
                            }
        imgViewSalida.setImageBitmap(imgSalida);
                        imgViewSalida.setScaleType(ScaleType.CENTER);
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                        Log.e(TAG, "Error en el callback preview");
                    }
                } else {
                        Log.e(TAG, "camara cerrada");
                }
        }
});

        } catch (Throwable e) {
                e.printStackTrace();
                // mCamera.release(); mCamera = null; return;
        }
}
```

Método: void startVideo()	
Paquete	edu.realtime.rtandsdk
Clase	AndActivity.java

```
//Funcion para realizar Captura
private void startVideo() {
        this.tlblEstado.setText("Capturando...");

        try {
                mCamera = getCameraInstance();
        } catch (RuntimeException e) {
                Log.e(TAG, "Camera Open filed");
                return;
        }

        try {
                mCamera.setErrorCallback(new ErrorCallback() {
                        public void onError(int error, Camera camera) {
```

```
            }
        });
        mCamera.startPreview();

        mCamera.setPreviewCallback(new Camera.PreviewCallback() {
            @Override
            public synchronized void onPreviewFrame(byte[]
            data, Camera camera) {
                Bitmap imgTmp;
                boolean[][] arrAnd;
                if (camera != null) {
            try {
            // Obtenemos y mostramos imagen de entrada
            // parseandola de YUV a JPEG
            Camera.Parameters parameters =
            camera.getParameters();
            Size size = parameters.getPreviewSize();
            YuvImage image = new YuvImage(data,
            parameters.getPreviewFormat(), size.width,
            size.height, null);
            ByteArrayOutputStream outStream = new
            ByteArrayOutputStream();
            image.compressToJpeg(new Rect(0, 0, size.width,
            size.height), 100, outStream); // make JPG
            imgTmp =
            BitmapFactory.decodeByteArray(outStream.toByteArray
            (), 0, outStream.size());
            if (tCapturarBase > 0) {
                tlblEstado.setText("Capturando base...");
                // 1. Convertimos a binaria
                arrBinEntrada = createBinaryImage(imgTmp);
                imgEntrada =
                obtenerBitmapBinario(arrBinEntrada,imgTmp.get
                Width(), imgTmp.getHeight());
                imgViewEntrada.setImageBitmap(imgEntrada);

imgViewEntrada.setScaleType(ScaleType.CENTER);
                tCapturarBase--;
                medirTiempo = true;
            } else if (tCapturarBase <= 0 && medirTiempo ==
        true) {
                tlblEstado.setText("Realizando pruebas...");
                // 1. Convertimos a binaria la salida
                arrBinSalida = createBinaryImage(imgTmp);
                ////// empezar captura de tiempos
                tiempoInicial = System.nanoTime();
                // 2. realizamos el AND
                arrAnd = andImages(arrBinEntrada,
            arrBinSalida, imgTmp.getWidth(),
            imgTmp.getHeight());
                // ////// finalizar captura de tiempos
                tiempoDuracion = (System.nanoTime() -
            tiempoInicial);
                tiempos.add((tiempoDuracion));
```

```
                    // 3. obtenemos el bitmap de salida para
                mostrar en pantalla
                    imgSalida =
                obtenerBitmapBinario(arrAnd,imgTmp.getWidth(),
                imgTmp.getHeight());
                if (tiempos.size() == noPruebas) {
                    medirTiempo = false;
                    stopVideo();
                }
                imgViewSalida.setImageBitmap(imgSalida);
imgViewSalida.setScaleType(ScaleType.CENTER);
                }
            } catch (Exception e) {
                e.printStackTrace();
                Log.e(TAG, "Error en el callback preview");
            }
                } else {
                        Log.e(TAG, "camara cerrada");
                    }
                }
            });
        } catch (Throwable e) {
            e.printStackTrace();
        }
    }
}
```

Método: Inicializar Opencv		
Paquete	com.rtcaptura.ropencvsdk;	com.example.rtprocesamientosdk
Clase	CapturaSDKActivity.java	MainActivity.java

```
    static{ //Inicializacion Estatica Libreria OpenCV
        if (!OpenCVLoader.initDebug()) {
            Log.e(TAG, "*******  ERROR  ****");
        }
    }
```

Método: OnCreate	
Paquete	com.rtcaptura.ropencvsdk;
Clase	CapturaSDKActivity.java

```
    public void onCreate(Bundle savedInstanceState) {
        Log.i(TAG, "******* onCreate");
        super.onCreate(savedInstanceState);
        mOpenCvCameraView = new CaptureViewBase(this);
        //Inicializar vista de camara
        setContentView(mOpenCvCameraView);
    }
```

Método: cameraOpen()	
Paquete	com.rtcaptura.ropencvsdk;
Clase	CapturaSDKActivity.java

```
public boolean cameraOpen() {
try {
        Log.i(TAG, "******* cameraOpen");
        synchronized (this) {
        cameraRelease();
        mCamera = new VideoCapture(Highgui.CV_CAP_ANDROID);
        if (!mCamera.isOpened()) { // abrir camara
                mCamera.release();
                mCamera = null;
                Log.e("HelloOpenCVView", "Failed to open native camera");
                return false;
        }
        }
        return true;
} catch (Exception ex) {
        Log.i(TAG, "******* cameraOpen" + ex.getMessage());
        return false;
}
}
```

Método: calcularTiempos()			
Paquete	com.rtcaptura.ropencvsdk;	com.example.rtprocesamientosdk	com.rt.filtros
Clase	CaptureViewBase.java	MainActivity.java	FiltrosActivity.java

```
        //Configurar tamano de la camara
private void calcularTiempos() {
        long tiempoMaximo = 0;
        long tiempoMinimo = lstTiempos.get(0);
        long tiempoPromedio = 0;

        for (int i = 0; i < lstTiempos.size(); i++) {
                if (lstTiempos.get(i) > tiempoMaximo) {
                        tiempoMaximo = lstTiempos.get(i);

                } else if (lstTiempos.get(i) < tiempoMinimo) {
                        tiempoMinimo = lstTiempos.get(i);
                }
                Log.i(TAG, "******* Arreglo :" + lstTiempos.get(i));

                tiempoPromedio += lstTiempos.get(i);

        }
        tiempoPromedio = tiempoPromedio / lstTiempos.size();
                // tiempoPromedio = tiempoPromedio / lstTiempos.size();
                Log.i(TAG, "******* Promedio :" + tiempoPromedio);
                Log.i(TAG, "******* Minimo :" + tiempoMaximo);
                Log.i(TAG, "******* Maximo :" + tiempoMinimo);
        }
```

Método: processFrame	
Paquete	com.rtcaptura.ropencvsdk;
Clase	CaptureViewBase.java

```
	//Procesar cada frame
protected Bitmap processFrame(VideoCapture capture) {
	Log.i(TAG, "******* processFrame");
	Mat mRgba = new Mat();
	capture.retrieve(mRgba, Highgui.CV_CAP_ANDROID_COLOR_FRAME_RGBA);
	//Capturar frame
	t2 = System.nanoTime(); //Tomar tiempo al final
	Log.i(TAG, "******* Numero de Pruebas" + lstTiempos.size());

	//Agregar a a lista los tiempos
	if (lstTiempos.size() < PRUEBAS) {
		duracion = t2 - t1;
		lstTiempos.add(duracion);
	} else if (lstTiempos.size() == PRUEBAS) {
		calcularTiempos(); // Calcular tiempos
	}
	// process mRgba
	Bitmap bmp = Bitmap.createBitmap(mRgba.cols(), mRgba.rows(),
	Bitmap.Config.ARGB_8888);
	try {
		// Utils.matToBitmap(imgProcesada, bmp);
		Utils.matToBitmap(mRgba, bmp);
	} catch (Exception e) {
		Log.e("processFrame", "Utils.matToBitmap() throws an exception:
		" + e.getMessage());
		bmp.recycle();
		bmp = null;
	}
	return bmp;
}
```

Método: run()	
Paquete	com.rtcaptura.ropencvsdk;
Clase	CaptureViewBase.java

```
	//Ejecutar hilo
public void run() {
Log.i(TAG, "******* run");
while (true) {
	Bitmap bmp = null;
	synchronized (this) {
		if (mCamera == null)
			break;
		if (!mCamera.grab())
			break;

		t1 = System.nanoTime();//Tomar tiempo de inicio
		bmp = processFrame(mCamera);
	}
	if (bmp != null) {
```

```
            //Presentar imagen capturada en pantalla
            Canvas canvas = getHolder().lockCanvas();
            if (canvas != null) {
                  canvas.drawBitmap(bmp,
                              (canvas.getWidth() - bmp.getWidth()) / 2,
                              (canvas.getHeight() - bmp.getHeight()) / 2,
null);

                  getHolder().unlockCanvasAndPost(canvas);

            }
            bmp.recycle();
      }
   }
   }
```

Método: OnCreate	
Paquete	com.example.rtprocesamientosdk
Clase	MainActivity.java

```
      protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      //Inicializar Image View para visualizar Imagen generada
      imageView = (ImageView) findViewById(R.id.image_view);

      //Ejecutar Algoritmo
      ejecutarOperacionNDK();
      //ejecutarOperacion();

   }
```

Método: restarImgenes	
Paquete	com.example.rtprocesamientosdk
Clase	MainActivity.java

```
      //Operacion Resta de 2 Imagenes
            protected void restarImgenes(Mat mt1, Mat mt2, Mat mtResul) {

                  Mat gray1 = new Mat();
                  Mat gray2 = new Mat();

                  Imgproc.cvtColor(mt1, gray1, Imgproc.COLOR_BGR2GRAY, 1);
                  Imgproc.cvtColor(mt2, gray2, Imgproc.COLOR_BGR2GRAY, 1);

                  t1 = System.nanoTime();
                  Core.subtract(gray2, gray1, mtResul); // Ejecutar Resta
                  t2 = System.nanoTime();

                  duracion = t2 - t1;
                  lstTiempos.add(duracion);

                  gray1.release();
                  gray2.release();
            }
```

Método: AndImagenes	
Paquete	com.example.rtprocesamientosdk
Clase	MainActivity.java

```
//Operacion AND de 2 imagenes
protected void AndImagenes(Mat mt1, Mat mt2, Mat mtResul) {

        Mat gray1 = new Mat();
        Mat gray2 = new Mat();

        Imgproc.cvtColor(mt1, gray1, Imgproc.COLOR_BGR2GRAY, 1);
        Imgproc.cvtColor(mt2, gray2, Imgproc.COLOR_BGR2GRAY, 1);

        t1 = System.nanoTime();
        Core.bitwise_and(gray2, gray1, mtResul); // And de
Imagenes
        t2 = System.nanoTime();

        duracion = t2 - t1;
        lstTiempos.add(duracion);

        gray1.release();
        gray2.release();
}
```

Método: ejecutarOperacion	
Paquete	com.example.rtprocesamientosdk
Clase	MainActivity.java

```
protected void ejecutarOperacion() {
//Cargar Imagenes  del telefono
File imgFile1 = new File(Environment.getExternalStorageDirectory()
            .toString() + "/DCIM/Img1.jpg");
File imgFile2 = new File(Environment.getExternalStorageDirectory()
            .toString() + "/DCIM/Img2.jpg");

Mat mt01 = Highgui.imread(imgFile1.getAbsolutePath());
Mat mt02 = Highgui.imread(imgFile2.getAbsolutePath());

Mat mtResul = new Mat();
Mat mt1 = new Mat();
Mat mt2 = new Mat();

for (int i = 0; i < PRUEBAS; i++) {
      mt01.copyTo(mt1);
      mt02.copyTo(mt2);

      //Realizar Operaciones,restarImgenes(mt1, mt2,mtResul);
      AndImagenes(mt1, mt2, mtResul);
      Bitmap bmp = Bitmap.createBitmap(mtResul.cols(),
mtResul.rows(),Bitmap.Config.ARGB_8888);
```

```
        if (i == PRUEBAS - 1) {
            Utils.matToBitmap(mtResul, bmp);
            imageView.setImageBitmap(bmp);
        }

        mt1.release();
        mt2.release();
        mtResul.release();
        // bmp.recycle();
    }
    //Calcular tiempos
    calcularTiempos();

}
```

Método: RestaImagenes	
Paquete	com.example.rtprocesamientosdk
Clase	nativeOpenCV.java

```
Java_com_example_rtprocesamientosdk_nativeOpenCV_RestaImagenes(JNIEn
v*, jobject, jlong addrImg1, jlong addrImg2, jlong addrResult) {

        LOG("******** RESTA IMAGENES *********************");
        Mat& mImg1  = *(Mat*)addrImg1;
        Mat& mImg2 = *(Mat*)addrImg2;
        Mat& mResult = *(Mat*)addrResult;

        clock_gettime(CLOCK_REALTIME, &t_ini);
        //medianBlur(mGr,mRgb,3);
        subtract(mImg2, mImg1, mResult);
        clock_gettime(CLOCK_REALTIME, &t_fin);

        //Calcular tiempo que dura la operacion y retornar ese valor
        tdif= ((t_fin.tv_sec * 1000000000) + t_fin.tv_nsec) -
        ((t_ini.tv_sec * 1000000000) + t_ini.tv_nsec);
        LOG("******* duracion : %f" , tdif);
        return tdif;
}
```

Método: AndImagenes	
Paquete	com.example.rtprocesamientosdk
Clase	nativeOpenCV.java

```
Java_com_example_rtprocesamientosdk_nativeOpenCV_AndImagenes(JNIEnv*
, jobject, jlong addrImg1, jlong addrImg2, jlong addrResult) {

        LOG("******** AND IMAGENES *********************");

              Mat& mImg1  = *(Mat*)addrImg1;
            Mat& mImg2 = *(Mat*)addrImg2;
            Mat& mResult = *(Mat*)addrResult;

            clock_gettime(CLOCK_REALTIME, &t_ini);
            bitwise_and(mImg2, mImg1, mResult);
```

```
		clock_gettime(CLOCK_REALTIME, &t_fin);

		//Calcular tiempo que dura la operacion y retornar ese
valor
		tdif= ((t_fin.tv_sec * 1000000000) + t_fin.tv_nsec) -
((t_ini.tv_sec * 1000000000) + t_ini.tv_nsec);
		LOG("******* duracion : %f" , tdif);
		return tdif;

}
```

Método para ROI: AndImagenes	
Paquete	com.rtcaptura.ropencvsdk
Clase	CaptureViewBase.java

```
	protected Bitmap processFrame(VideoCapture capture) {
		Log.i(TAG, "******* processFrame");

		Mat mRgba = new Mat();
		capture.retrieve(mRgba,
Highgui.CV_CAP_ANDROID_COLOR_FRAME_RGBA);

			//ROI
			 Mat result = new Mat();
			duracion2 =
			nativeOpenCV.RoiImagenes(mRgba.nativeObj,
			result.nativeObj);

			 if (lstTiempos2.size()<PRUEBAS){
					Log.i(TAG, "******* Count ****" +
					lstTiempos2.size());
					lstTiempos2.add(duracion2);
				}else if (lstTiempos2.size() == PRUEBAS) {
					if (lstTiempos2.size()!=0){
					calcularTiempos2();}
				}

			 Bitmap bmp = Bitmap.createBitmap(result.cols(),
			 result.rows(),
				Bitmap.Config.ARGB_8888);

		try {

			// Utils.matToBitmap(imgProcesada, bmp);
			Utils.matToBitmap(result, bmp);
			result.release();
		} catch (Exception e) {
			Log.e("processFrame", "Utils.matToBitmap() throws
			an exception: " + e.getMessage());
			bmp.recycle();
			bmp = null;
		}
		return bmp;
	}
```

```
//Procesar cada frame OPENCV SDK
      protected Bitmap processFrame(VideoCapture capture) {
            Log.i(TAG, "******* processFrame");

            Mat mRgba = new Mat();
            capture.retrieve(mRgba,
Highgui.CV_CAP_ANDROID_COLOR_FRAME_RGBA);
                  //ROI
            t1 = System.nanoTime();
                  Rect roi = new Rect(0, 0, 400, 400);
                  Mat result = new Mat(mRgba, roi);
            t2 = System.nanoTime();
                  duracion = t2 - t1;
                   if (lstTiempos.size()<PRUEBAS){
                              Log.i(TAG, "******* Count ****" +
                              lstTiempos.size());
                              lstTiempos.add(duracion);
                        }else if (lstTiempos.size() == PRUEBAS) {
                              //if (lstTiempos.size()!=0){
                              calcularTiempos();//}
                        }
                   Bitmap bmp = Bitmap.createBitmap(result.cols(),
                  result.rows(),
                        Bitmap.Config.ARGB_8888);

            try {

                  // Utils.matToBitmap(imgProcesada, bmp);
                  Utils.matToBitmap(result, bmp);
                  result.release();
            } catch (Exception e) {
                  Log.e("processFrame", "Utils.matToBitmap() throws
                  an exception: "
                              + e.getMessage());
                  bmp.recycle();
                  bmp = null;
            }
            return bmp;
      }
```

Método para ROI: RoiImagenes	
Paquete	com.rtcaptura.ropencvsdk
Clase	nativeOpenCv.java

```
Java_com_rtcaptura_ropencvsdk_nativeOpenCV_RoiImagenes(JNIEnv*,
jobject, jlong addrGray, jlong addrRgba) {
      LOG("******** ROI IMAGENES **********************");
            Mat& mGr   = *(Mat*)addrGray;
          Mat& mRgb = *(Mat*)addrRgba;

          clock_gettime(CLOCK_REALTIME, &t_ini);
```

```
          Rect rec(0, 0, 400, 400);
          mRgb = Mat(mGr, rec);
        clock_gettime(CLOCK_REALTIME, &t_fin);

        tdif= ((t_fin.tv_sec * 1000000000) + t_fin.tv_nsec) -
        ((t_ini.tv_sec * 1000000000) + t_ini.tv_nsec);
        LOG("******* duracion : %f" , tdif);
        return tdif;

}
```

Método: MediaImagenes	
Paquete	**com.rt.filtros.nativeOpenCV**
Clase	FiltrosActivity.java

```
    /***** MEDIA IMAGEN ****/
    protected Mat MediaImagenes(Mat frame) {
          Log.i(TAG, "*******  Media de Images ****");
          Log.i(TAG, "******* Count ****" + lstTiempos.size());

          Mat mtResul = new Mat();
          t1 = System.nanoTime();
                Imgproc.medianBlur(frame, mtResul, 3);
          t2=System.nanoTime();

          if (lstTiempos.size()<PRUEBAS){
                duracion = t2 - t1 ;
                lstTiempos.add(duracion);
          }else if (lstTiempos.size() == PRUEBAS) {
                calcularTiempos();
          }
          return mtResul;
    }


JNIEXPORT float JNICALL
Java_com_rt_filtros_nativeOpenCV_MediaImagenes(JNIEnv*, jobject,
jlong addrGray, jlong addrRgba) {

      LOG("******** FILTRO MEDIA *********************");
      Mat& mGr  = *(Mat*)addrGray;
      Mat& mRgb = *(Mat*)addrRgba;
        clock_gettime(CLOCK_REALTIME, &t_ini);
        medianBlur(mGr,mRgb,3);
        clock_gettime(CLOCK_REALTIME, &t_fin);

        tdif= ((t_fin.tv_sec * 1000000000) + t_fin.tv_nsec) -
        ((t_ini.tv_sec * 1000000000) + t_ini.tv_nsec);
        LOG("******* duracion : %f" , tdif);
        return tdif;

}
```

Método: ErodeImagenes	
Paquete	**com.rt.filtros.nativeOpenCV**
Clase	FiltrosActivity.java

```
/***** EROSION IMAGEN ****/
      protected Mat ErodeImagenes(Mat frame) {
            Log.i(TAG, "*******  Erode de Imagenes  ****");
            Log.i(TAG, "******* Count ****" + lstTiempos.size());
            Mat mtResul = new Mat();

            t1 = System.nanoTime();
            Mat mErodeKernel =
Imgproc.getStructuringElement(Imgproc.MORPH_ERODE, new Size(9, 9));
            Imgproc.erode(frame, mtResul, mErodeKernel);

            t2=System.nanoTime();
            if (lstTiempos.size()<PRUEBAS){
                  duracion = t2 - t1 ;
                  lstTiempos.add(duracion);
            }else if (lstTiempos.size() == PRUEBAS) {
                  calcularTiempos();
            }
            return mtResul;
      }

JNIEXPORT float JNICALL
Java_com_rt_filtros_nativeOpenCV_ErodeImagenes(JNIEnv*, jobject,
jlong addrGray, jlong addrRgba) {
      LOG("******** FILTRO ERODE **********************");
      Mat& mGr  = *(Mat*)addrGray;
      Mat& mRgb = *(Mat*)addrRgba;
      clock_gettime(CLOCK_REALTIME, &t_ini);
      Mat mErodeKernel = getStructuringElement(MORPH_ERODE, Size(9,
9));
      erode(mGr, mRgb, mErodeKernel);
      clock_gettime(CLOCK_REALTIME, &t_fin);
      tdif= ((t_fin.tv_sec * 1000000000) + t_fin.tv_nsec) -
((t_ini.tv_sec * 1000000000) + t_ini.tv_nsec);
      LOG("******* duracion : %f" , tdif);
      return tdif;
}
```

Método: PromedioImagenes	
Paquete	**com.rt.filtros.nativeOpenCV**
Clase	FiltrosActivity.java

```
protected Mat PromedioImagenes(Mat frame) {
      Log.i(TAG, "*******  Promedio de Images ****");
      Log.i(TAG, "******* Count ****" + lstTiempos.size());
      Mat mtResul = new Mat();

      t1 = System.nanoTime();
      Size s = new Size(5, 5);
      Imgproc.blur(frame, mtResul, s);
      t2=System.nanoTime() ;
```

```
        if (lstTiempos.size()<PRUEBAS){
              duracion = t2 - t1 ;
              lstTiempos.add(duracion);
        }else if (lstTiempos.size() == PRUEBAS) {
              calcularTiempos();
        }
              return mtResul;
        }

JNIEXPORT float JNICALL
Java_com_rt_filtros_nativeOpenCV_PromedioImagenes(JNIEnv*, jobject,
jlong addrGray, jlong addrRgba) {
      LOG("******** FILTRO PROMEDIO *********************");
      Mat& mGr  = *(Mat*)addrGray;
      Mat& mRgb = *(Mat*)addrRgba;

      clock_gettime(CLOCK_REALTIME, &t_ini);
      blur(mGr, mRgb, Size(5, 5));
      clock_gettime(CLOCK_REALTIME, &t_fin);

 tdif= ((t_fin.tv_sec * 1000000000) + t_fin.tv_nsec) -((t_ini.tv_sec
* 1000000000) + t_ini.tv_nsec);
      LOG("******* duracion : %f" , tdif);
      return tdif;
}
```

Método: RotarImagenes	
Paquete	**com.rt.filtros.nativeOpenCV**
Clase	FiltrosActivity.java

```
protected Mat RotarImagen(Mat mRgba) {
      Mat result = new Mat();
      Log.i(TAG, "*******  Rotar de Imagenes  ****");
      Log.i(TAG, "******* Count ****" + lstTiempos.size());

      t1 = System.nanoTime();
      int degrees = 90;
      Point center = new Point(mRgba.cols() / 2, mRgba.rows() / 2);
      Mat rotImage = Imgproc.getRotationMatrix2D(center, degrees,
1.0);
      Imgproc.warpAffine(mRgba, result, rotImage, mRgba.size());
      t2=System.nanoTime();

      if (lstTiempos.size()<PRUEBAS){
            duracion = t2 - t1 ;
            lstTiempos.add(duracion);
      }else if (lstTiempos.size() == PRUEBAS) {
            calcularTiempos();
      }
      //count= count+1;
      return result;
}
```

```
JNIEXPORT float JNICALL
Java_com_rt_filtros_nativeOpenCV_RotarImagenes(JNIEnv*, jobject,
jlong addrGray, jlong addrRgba) {
	LOG("******** ROTAR PROMEDIO *********************");
	Mat& mGr  = *(Mat*)addrGray;
	Mat& mRgb = *(Mat*)addrRgba;

	clock_gettime(CLOCK_REALTIME, &t_ini);
	int degrees = 90;
	Point2f center(mGr.cols/ 2.0F, mGr.rows/ 2.0F);
	Mat rotImage = getRotationMatrix2D(center, degrees, 1.0);
	warpAffine(mGr,mRgb, rotImage, mRgb.size());
	clock_gettime(CLOCK_REALTIME, &t_fin);

	tdif= ((t_fin.tv_sec * 1000000000) + t_fin.tv_nsec) -
((t_ini.tv_sec * 1000000000) + t_ini.tv_nsec);
	LOG("******* duracion : %f" , tdif);
	 return tdif;
}
```

Método: PasoAltoImagenes	
Paquete	**com.rt.filtros.nativeOpenCV**
Clase	FiltrosActivity.java

```
protected Mat FiltroPasoBajoImgenes(Mat mt1) {
	Log.i(TAG, "*******  Filtro Paso Bajo de Images ****");
	Log.i(TAG, "******* Count ****" + lstTiempos.size());
	Mat mtResul = new Mat();
	t1 = System.nanoTime();
	Mat kernel = new Mat(new Size(3,3), CvType.CV_64F);
	kernel.put(0, 0, 0);kernel.put(1, 0, 1);kernel.put(2, 0, 0);
	kernel.put(0, 1, 1);kernel.put(1, 1, 2);kernel.put(2, 1, 1);
	kernel.put(0, 2, 0);kernel.put(1, 2, 1);kernel.put(2, 2, 0);

	Imgproc.filter2D(mt1, mtResul, mtResul.depth() , kernel);

	t2 = System.nanoTime();
	if (lstTiempos.size()<PRUEBAS){
		duracion = t2 - t1 ;
		lstTiempos.add(duracion);
	}else if (lstTiempos.size() == PRUEBAS) {
		calcularTiempos();
	}
		return mtResul;
	}

JNIEXPORT float JNICALL
Java_com_rt_filtros_nativeOpenCV_PasoAltoImagenes(JNIEnv*, jobject,
jlong addrGray, jlong addrRgba) {
	LOG("******** FILTRO PASO ALTO *********************");
	Mat& mGr  = *(Mat*)addrGray;
```

```
        Mat& mRgb = *(Mat*)addrRgba;

        clock_gettime(CLOCK_REALTIME, &t_ini);
        double m[3][3] = {{0,-1,0},{-1,5,-1},{0,-1,0}};
        Mat kernel = Mat(3,3, CV_64F,m);
        filter2D(mGr,mRgb,mRgb.depth(),kernel);
        clock_gettime(CLOCK_REALTIME, &t_fin);

        tdif= ((t_fin.tv_sec * 1000000000) + t_fin.tv_nsec) -
((t_ini.tv_sec * 1000000000) + t_ini.tv_nsec);
            LOG("******* duracion : %f" , tdif);
            return tdif;
}
```

Método: PasoBajoImagenes	
Paquete	**com.rt.filtros.nativeOpenCV**
Clase	FiltrosActivity.java

```
protected Mat FiltroPasoAltoImgenes(Mat mt1) {
        Log.i(TAG, "*******  Filtro Paso Alto de Images ****");
        Log.i(TAG, "******* Count ****" + lstTiempos.size());

        Mat mtResul = new Mat();
        t1 = System.nanoTime();
        Mat kernel = new Mat(new Size(3,3), CvType.CV_64F);

        kernel.put(0, 0, 0);kernel.put(1, 0, -1);kernel.put(2, 0, 0);
        kernel.put(0, 1, -1);kernel.put(1, 1, 5);kernel.put(2, 1, -1);
        kernel.put(0, 2, 0);kernel.put(1, 2, -1);kernel.put(2, 2, 0);

        Imgproc.filter2D(mt1, mtResul, mtResul.depth() , kernel);

        t2 = System.nanoTime();
        if (lstTiempos.size()<PRUEBAS){
                duracion = t2 - t1 ;
                lstTiempos.add(duracion);
        }else if (lstTiempos.size() == PRUEBAS) {
                calcularTiempos();
        }
        return mtResul;
}


JNIEXPORT float JNICALL
Java_com_rt_filtros_nativeOpenCV_PasoBajoImagenes(JNIEnv*, jobject,
jlong addrGray, jlong addrRgba) {
        LOG("******** FILTRO PASO BAJO **********************");
        Mat& mGr  = *(Mat*)addrGray;
        Mat& mRgb = *(Mat*)addrRgba;
        clock_gettime(CLOCK_REALTIME, &t_ini);

        double m[3][3] = {{0,1,0},{1,2,1},{0,1,0}};
        Mat kernel = Mat(3,3, CV_64F,m);
        filter2D(mGr,mRgb,mRgb.depth(),kernel);
```

```
        clock_gettime(CLOCK_REALTIME, &t_fin);

        tdif= ((t_fin.tv_sec * 1000000000) + t_fin.tv_nsec) -
((t_ini.tv_sec * 1000000000) + t_ini.tv_nsec);
           LOG("******* duracion : %f" , tdif);
        return tdif;
}
```

Método: OnCameraFrame	
Paquete	**com.rt.filtros.nativeOpenCV**
Clase	FiltrosActivity.java

```
//Funcion para procccesar y presentar frames por
CameraBridgeViewBase
public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
      mIntermediateMat = inputFrame.rgba();
      mGray = inputFrame.gray();
      final int viewMode = mFilterMode;
      //Aplicacion de filtros segun opcionescogida por el usuario
      switch (viewMode) {
            case VIEW_Normal:
                  mRgba = mIntermediateMat;
                  break;

            case VIEW_RotarImagen:
                  mRgba = RotarImagen(mRgba);
                  break;

            case VIEW_Erode:
                  mRgba = ErodeImagenes(mGray);
                  break;

            case VIEW_MedianFilter:
                  mRgba = MediaImagenes(mGray);
                  break;

            case VIEW_MediaFilter:
                  mRgba = PromedioImagenes(mGray);

                  break;
            case VIEW_PasoAltoFilter:
                  mRgba = FiltroPasoAltoImgenes(mGray);

                  break;
            case VIEW_PasoBajoFilter:
                  mRgba = FiltroPasoBajoImgenes(mGray);

                  break;

                  //Apliar filtros desde NDK
            case VIEW_NDKMedianFilter:
```

```
            {
            duracion2 =
nativeOpenCV.MediaImagenes(mGray.getNativeObjAddr(),
mRgba.getNativeObjAddr());

      if (lstTiempos2.size()<PRUEBAS){
            Log.i(TAG, "******* Count ****" + lstTiempos2.size());
            lstTiempos2.add(duracion2);
            Log.i(TAG, "add sdk duracion " + duracion2);
      }else if (lstTiempos2.size() == PRUEBAS) {
            if (lstTiempos2.size()!=0){
                  calcularTiempos2();}
            }
            break;
      }

      case VIEW_NDKErodenFilter:
      {
            duracion2 =
nativeOpenCV.ErodeImagenes(mGray.getNativeObjAddr(),
mRgba.getNativeObjAddr());

            if (lstTiempos2.size()<PRUEBAS){
                  Log.i(TAG, "******* Count ****" +
lstTiempos2.size());
                  lstTiempos2.add(duracion2);
                  Log.i(TAG, "add sdk duracion " + duracion2);
            }else if (lstTiempos2.size() == PRUEBAS) {
                  if (lstTiempos2.size()!=0){
                        calcularTiempos2();}
                  }
                  break;
            }

            case VIEW_NDKPromedioFilter:
            {
                  duracion2 =
nativeOpenCV.PromedioImagenes(mGray.getNativeObjAddr(),
mRgba.getNativeObjAddr());

            if (lstTiempos2.size()<PRUEBAS){
                  Log.i(TAG, "******* Count ****" +
lstTiempos2.size());
                  lstTiempos2.add(duracion2);
                  Log.i(TAG, "add sdk duracion " + duracion2);
            }else if (lstTiempos2.size() == PRUEBAS) {
                  if (lstTiempos2.size()!=0){
                        calcularTiempos2();}
                  }
                  break;
            }
            case VIEW_NDKRotarFilter:
            {
                  duracion2 =
```

```
nativeOpenCV.RotarImagenes(mIntermediateMat.getNativeObjAddr(),
mRgba.getNativeObjAddr());

      if (lstTiempos2.size()<PRUEBAS){
            Log.i(TAG, "******* Count ****" + lstTiempos2.size());
            lstTiempos2.add(duracion2);
            Log.i(TAG, "add sdk duracion " + duracion2);
      }else if (lstTiempos2.size() == PRUEBAS) {
            if (lstTiempos2.size()!=0){
                  calcularTiempos2();}
            }
            break;
      }
case VIEW_NDKPasoAltoFilter:
      {
            duracion2 =
nativeOpenCV.PasoAltoImagenes(mGray.getNativeObjAddr(),
mRgba.getNativeObjAddr());

            if (lstTiempos2.size()<PRUEBAS){
                  Log.i(TAG, "******* Count ****" +
lstTiempos2.size());
                  lstTiempos2.add(duracion2);
                  Log.i(TAG, "add sdk duracion " + duracion2);
            }else if (lstTiempos2.size() == PRUEBAS) {
                  if (lstTiempos2.size()!=0){
                  calcularTiempos2();}
            }
            break;
      }

      case VIEW_NDKPasoBajoFilter:
      {
            duracion2 =
nativeOpenCV.PasoBajoImagenes(mGray.getNativeObjAddr(),
mRgba.getNativeObjAddr());

      if (lstTiempos2.size()<PRUEBAS){
            Log.i(TAG, "******* Count ****" + lstTiempos2.size());
                  lstTiempos2.add(duracion2);
                  Log.i(TAG, "add sdk duracion " + duracion2);
            }else if (lstTiempos2.size() == PRUEBAS) {
                  if (lstTiempos2.size()!=0){
                        calcularTiempos2();}
            }
            break;
      }
}
      return mRgba;
}
```

ANEXO B

Tabla 8. Datos de 100 experimento de Captura

DATOS PARA 100 CAPTURAS			
No.	**Android OpenCV SDK**	**Android OpenCV NDK**	**Android SDK**
1	10,2416	0,1966	**7,7450**
2	0,7296	0,2134	3,4430
3	0,5611	0,3014	4,1198
4	0,7533	0,1931	3,1432
5	0,5889	0,1828	2,9300
6	0,6529	0,1487	3,2916
7	0,7236	0,2485	2,8475
8	0,6524	0,1995	3,1792
9	0,6176	0,2077	2,5909
10	0,7616	0,4656	3,0722
11	0,4993	0,1445	2,9106
12	0,7456	0,1711	2,6468
13	0,6919	0,1542	2,7187
14	0,6428	0,1292	2,6507
15	0,6560	0,0987	2,5709
16	0,7679	0,1553	2,5706
17	0,5161	0,1324	2,6312
18	0,7120	0,2096	2,2990
19	0,7068	0,2247	2,2994
20	0,9464	0,2036	2,4307
21	0,4177	0,1615	2,4177
22	0,5984	0,2356	2,4132
23	0,6845	0,2224	2,4505
24	0,7025	0,2043	2,4212
25	0,7272	0,2070	2,5555
26	0,4776	0,2054	2,7205
27	0,7585	0,2085	2,5929
28	0,6998	0,2096	3,0360
29	0,5496	0,0853	2,6187

30	0,7282	0,1092	2,7504
31	0,6669	0,1257	3,0676
32	0,6669	0,2202	2,9310
33	0,5885	0,1934	2,8021
34	0,7400	0,1841	3,3494
35	0,6730	0,2007	2,7945
36	0,7031	0,2105	2,9554
37	0,7620	0,2313	2,6211
38	0,4907	0,2418	2,5566
39	0,7837	0,1641	2,6219
40	0,6376	0,3087	2,5856
41	0,6416	0,2436	2,5785
42	0,6601	0,1792	2,6018
43	0,6564	0,2051	2,5807
44	0,7086	0,2182	2,6753
45	0,6545	0,1339	2,4073
46	0,6389	0,2090	2,9632
47	0,6591	0,2426	2,5121
48	0,6661	0,2157	2,6269
49	0,7056	0,2260	2,4679
50	0,6410	0,2150	2,5718
51	0,6523	0,2841	2,5354
52	0,7622	0,2454	2,6250
53	0,5225	0,2126	2,5585
54	0,7680	0,2113	2,6554
55	0,6269	0,2091	2,5211
56	0,6687	0,2153	2,6342
57	0,6645	0,1481	2,8452
58	0,7562	0,2118	2,6297
59	0,4948	0,2160	2,4518
60	0,7327	0,0922	2,4585
61	0,8533	0,2081	2,3327
62	0,4105	0,2331	2,3161
63	0,7373	0,2070	2,3414
64	0,8484	0,2488	2,4475
65	0,4241	0,2135	2,3543
66	0,7183	0,2211	2,3669

67	0,8170	0,2256	2,4670
68	0,4354	0,3697	2,4688
69	0,7410	0,1235	2,5877
70	0,7977	0,0855	2,6285
71	0,4579	0,0989	2,3619
72	0,7515	0,1181	2,4362
73	0,8707	0,1848	2,4296
74	0,3761	0,1149	2,5804
75	0,7449	0,3539	2,5386
76	0,7710	0,1348	2,3401
77	0,4850	0,1336	2,3386
78	0,7380	0,3143	2,7302
79	0,6784	0,2007	2,8540
80	0,6521	0,2107	2,7169
81	0,7015	0,3276	2,4046
82	0,6428	0,1830	2,2831
83	0,6468	0,2102	2,3233
84	0,7317	0,2169	2,4111
85	0,7837	0,2271	2,4362
86	0,4075	0,1325	2,4857
87	0,7732	0,2097	2,4875
88	0,6697	0,0798	2,5428
89	0,5814	0,1217	2,5976
90	0,6784	0,1968	2,5885
91	0,6643	0,1795	2,5743
92	0,6893	0,2201	2,5918
93	0,7430	0,2148	2,4525
94	0,6253	0,1306	2,5597
95	0,6329	0,1456	2,6442
96	0,7397	0,1278	2,4934
97	0,7038	0,0854	2,5114
98	0,4947	0,1204	2,6349
99	0,7236	0,1815	2,5125
100	0,6623	0,1101	3,3288

Tabla 9. Resultados de cálculos estadísticos.

ESTADISTICOS			
Promedio	0,7611	0,1947	2,6875
Varianza	0,9289	0,0041	0,3430
Desv. Est.	0,9638	0,0637	0,5857
Q1	0,6364	0,1475	2,4523
Mediana	0,6757	0,2062	2,5806
Q3	0,7398	0,2186	2,7173
Min	0,3761	0,0798	2,2831
Max	10,2416	0,4656	7,7450

ANEXO C

Tabla 10. Tiempos de adquisición de imágenes usando Android SDK.

No. Pruebas	**Android SDK x 10^8 (ns)**	
	$\bar{X}$	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

Tabla 11. Tiempos promedios y desviación estándar de adquisición de imágenes con OpenCV.

No. Pruebas	**Android OpenCV x 10^8 (ns)**			
	NDK		**SDK**	
	$\bar{X}$	σ	$\bar{X}$	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

Tabla 12. Tiempos en nanosegundos para experimento de ROI.

No. Pruebas	**ROI $x\ 10^6$ (ns)**		
	Android SDK	**OpenCV SDK**	**OpenCV NDK**
5	8,598	0,535	0,020
10	3,298	0,041	0,004358
25	3,312	0,028	0,003919
50	3,080	0,029	0,003605
100	2,493	0,116	0,003510

Tabla 13. Tiempos en nanosegundos para experimento de ROTACIÓN.

No. Pruebas	**ROTACIÓN $x\ 10^6$ (ns)**		
	Android SDK	**OpenCV SDK**	**OpenCV NDK**
5	11,583	8,428	8,509
10	12,891	4,424	3,508
25	19,808	1,337	1,520
50	19,420	0,590	0,658
100	14,330	0,293	0,351

Tabla 14. Tiempos en nanosegundos para experimento de RESTA.

No. Pruebas	**RESTA $x\ 10^6$ (ns)**		
	Android SDK	**OpenCV SDK**	**OpenCV NDK**
5	2298,107	23,774	24,127
10	2488,703	8,796	10,434
25	2347,713	3,406	4,623
50	2473,460	2,190	2,626
100	2322,242	0,888	1,006

Tabla 15. Tiempos en nanosegundos para experimento de AND.

No. Pruebas	**AND x 10^6 (ns)**		
	Android SDK	**OpenCV SDK**	**OpenCV NDK**
5	835,321	10,012	13,766
10	842,048	6,427	7,814
25	918,675	2,393	2,290
50	861,071	1,484	1,637
100	797,996	0,623	0,624

Tabla 16. Tiempos en nanosegundos para experimento de EROSIÓN.

No. Pruebas	**EROSIÓN x 10^6 (ns)**	
	OpenCV SDK	**OpenCV NDK**
5	3,921	5,699
10	2,416	2,523
25	0,932	1,625
50	0,288	0,326
100	0,177	0,186

Tabla 17. Tiempos en nanosegundos para experimento de FILTRO MEDIANO.

No. Pruebas	**FILTRO MEDIANO x 10^6 (ns)**	
	OpenCV SDK	**OpenCV NDK**
5	11,129	9,827
10	5,490	3,959
25	2,015	1,580
50	0,887	0,749
100	0,410	0,397

Tabla 18. Tiempos en nanosegundos para experimento de FILTRO PROMEDIO.

No. Pruebas	**FILTRO PROMEDIO x 10^6 (ns)**	
	OpenCV SDK	**OpenCV NDK**
5	3,696	4,176
10	1,991	2,0
25	0,808	0,861
50	0,418	0,440
100	0,203	0,218

Tabla 19. Tiempos en nanosegundos para experimento de FILTRO PASO ALTO.

No. Pruebas	**FILTRO PASO ALTO x 10^6 (ns)**	
	OpenCV SDK	**OpenCV NDK**
5	4,742	8,060
10	3,161	1,620
25	0,724	1,083
50	0,420	0,313
100	0,264	0,243

Tabla 20. Tiempos en nanosegundos para experimento de FILTRO PASO BAJO.

No. Pruebas	**FILTRO PASO BAJO x 10^6 (ns)**	
	OpenCV SDK	**OpenCV NDK**
5	9,058	4,200
10	1,599	1,536
25	0,422	0,932
50	0,197	0,313
100	0,153	0,239

ANEXO D

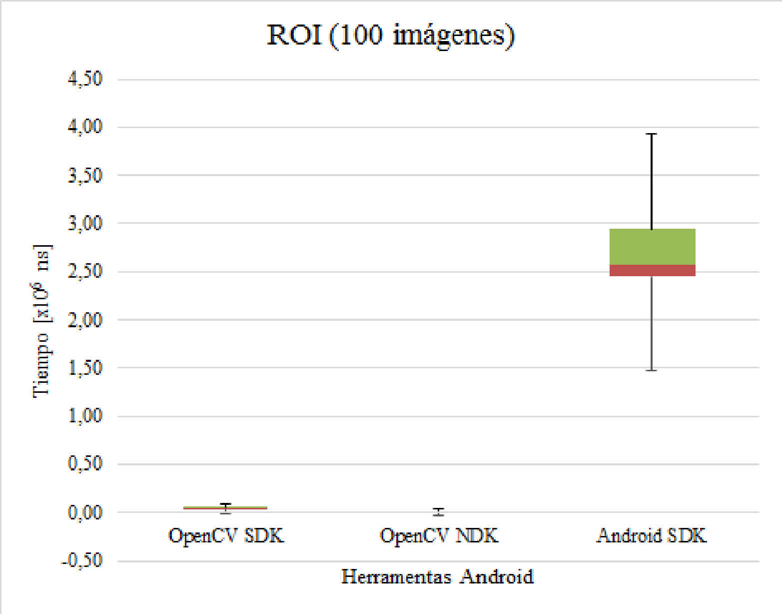

Figura 25. Diagrama de cajas para ROI aplicado a 100 imágenes

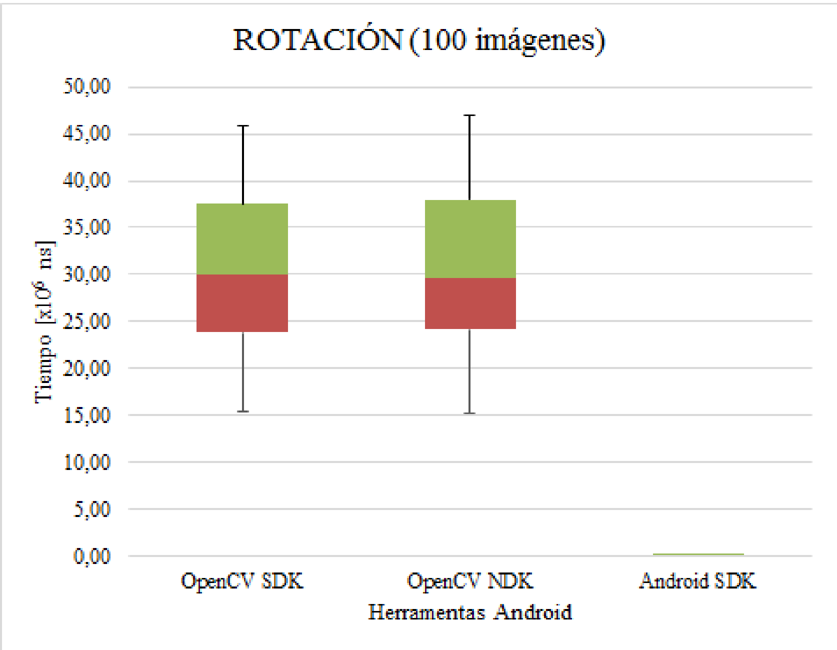

Figura 26. Diagrama de cajas para ROTACIÓN aplicado a 100 imágenes

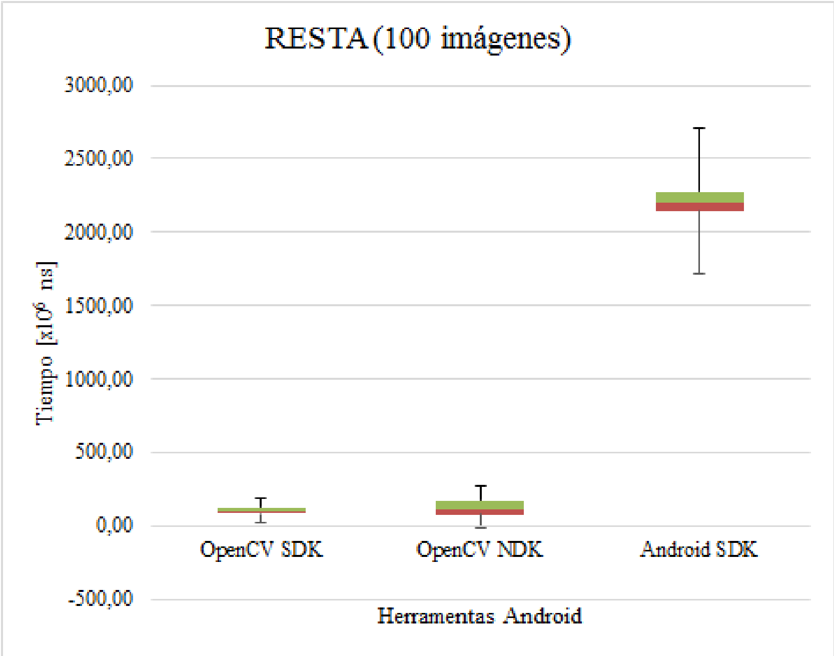

Figura 27. Diagrama de cajas para RESTA aplicada a 100 imágenes

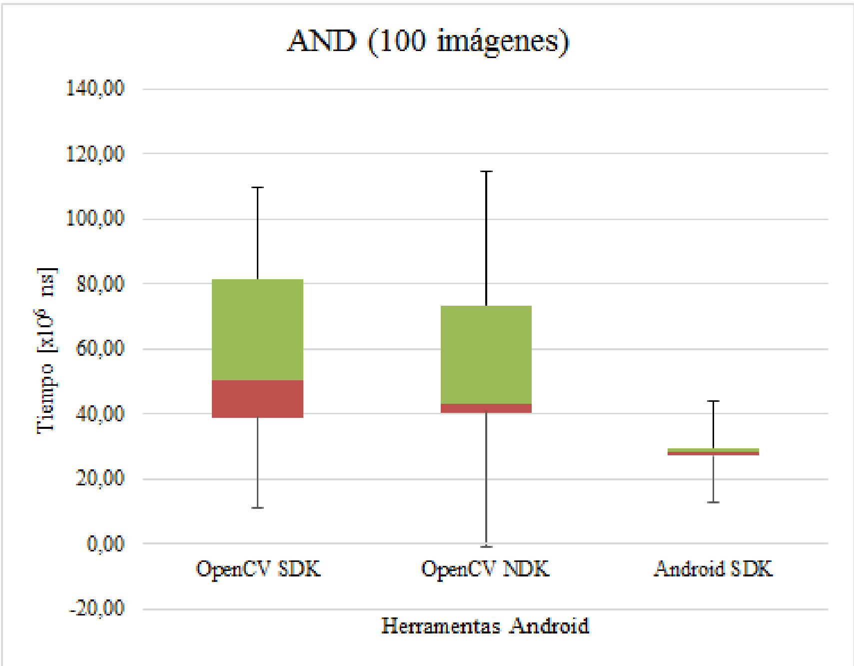

Figura 28. Diagrama de cajas para AND aplicado a 100 imágenes

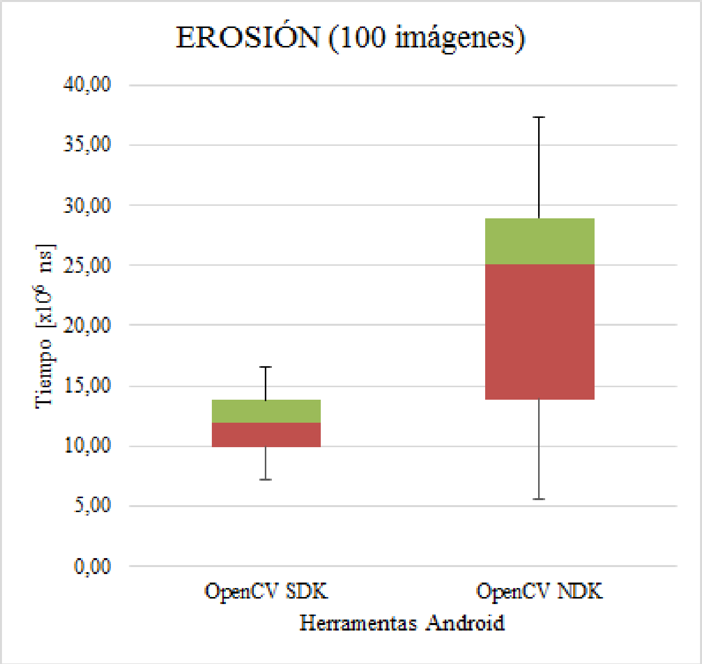

Figura 29. Diagrama de cajas para EROSIÓN aplicado a 100 imágenes

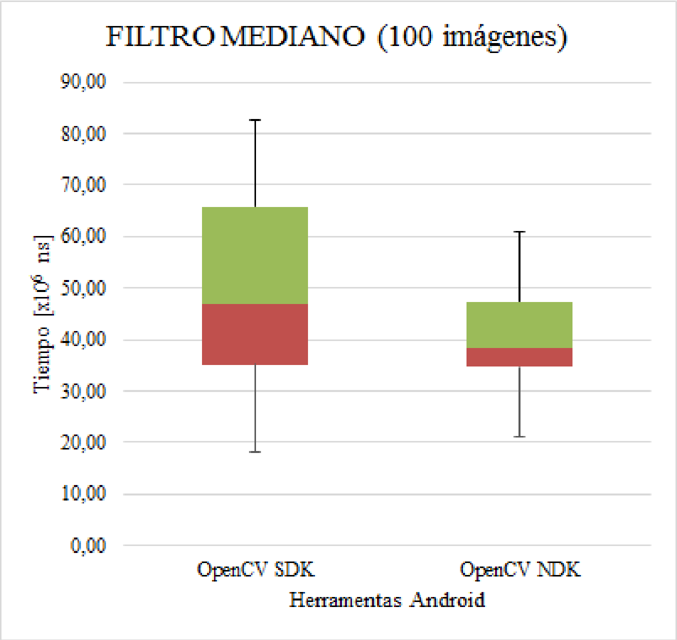

Figura 30. Diagrama de cajas para FILTRO MEDIANO aplicado a 100 imágenes

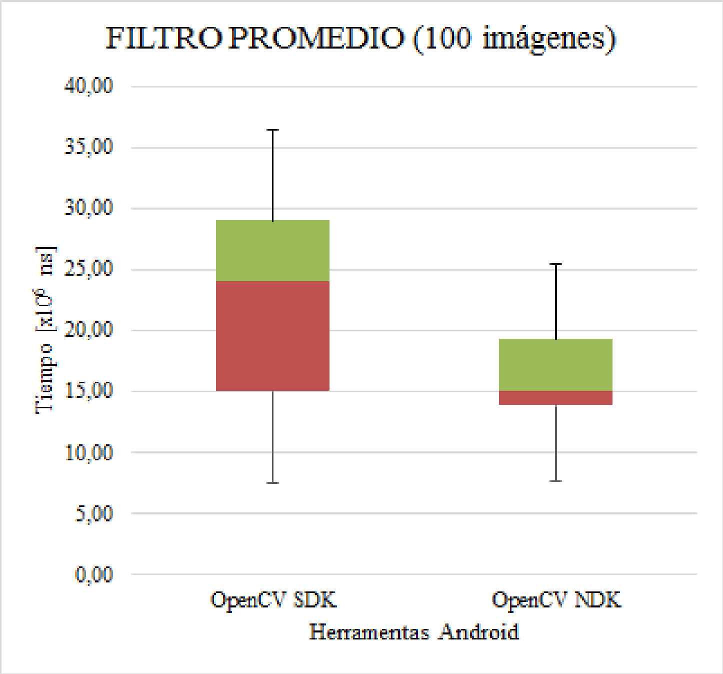

Figura 31. Diagrama de cajas para FILTRO PROMEDIO aplicado a 100 imágenes

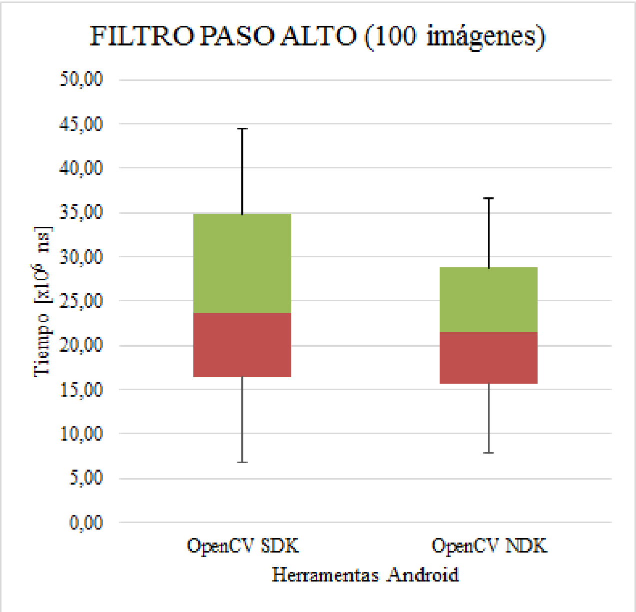

Figura 32. Diagrama de cajas para FILTRO PASO-ALTO aplicado a 100 imágenes

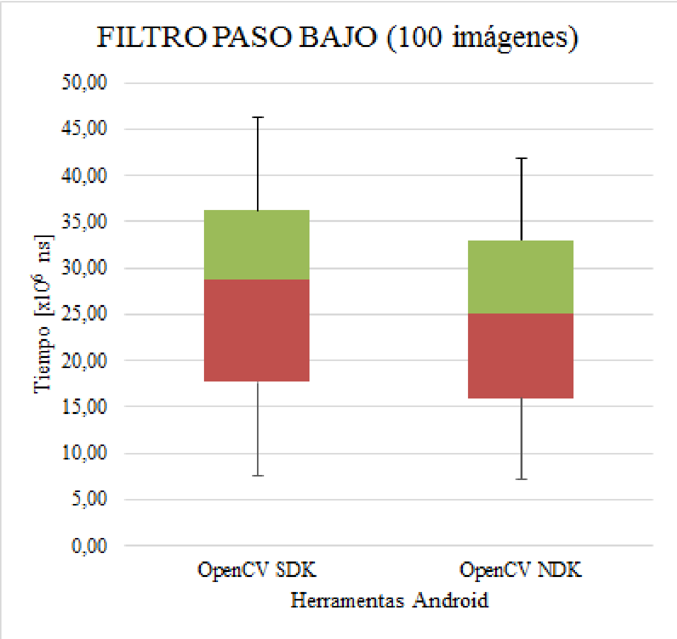

Figura 33. Diagrama de cajas para PASO BAJO aplicado a 100 imágenes

BIBLIOGRAFÍA

[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] Vaughan, S.(March 19,2012). Android and Linux re-merge into one operating system. Recuperado Diciembre 12,2012, de http://www.zdnet.com/blog/open-source/android-and-linux-re-merge-into-one-operating-system/10625

[4] 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

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

[6] 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

[7] 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

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

[9] Google Inc. Application Fundamentals. Recuperado Mayo 2, 2013, de http://developer.android.com/intl/es/guide/components/fundamentals.html

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

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

[12] 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

[13] 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

[14] 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

[15] 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/

[16] Universidad Rey Juan Carlos.Tema 2. Preproceso (realzado y filtrado) de imágenes digitales. Recuperado Junio 1, 2013 de http://www.escet.urjc.es/~visiona/tema2.pdf

[17] Vintimilla B. (2012). Sección 2.5 Operaciones Morfológicas, Procesamiento Digital de Imágenes.

[18] 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

[19] Hellman E. (2013, Febrero 14). Get started with OpenCV on Android™. Recuperad Recuperado Junio 2, 2013 de http://developer.sonymobile.com/knowledge-base/tutorials/android_tutorial/get-started-with-opencv-on-android/

[20] Gibara T.(2009, Julio 11). [Msg. 6] Recuperado Junio 2, 2013 de https://groups.google.com/forum/?fromgroups#!searchin/android-ndk/camera$20/android-ndk/qehnrEEoxa0/FAnRodBr0FYJ

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

[22] Ars Technica. Linux 3.3 released, restores Android components in staging. Recuperado Junio 2, 2013 de http://arstechnica.com/gadgets/2012/03/linux-33-released-restores-android-components-in-staging/

[23] Martínez A. Imágenes Binarias. Recuperado Junio 2, 2012 de http://wellpath.uniovi.es/es/contenidos/seminario/tutorialpdi/html/binaria.htm