JÓVENES EN LA CIENCIA
REVISTA DE DIVULGACIÓN CIENTÍFICA

PLATAFORMA DE DESARROLLO DE PROCESAMIENTO DE IMÁGENES EN DISPOSITIVOS MÓVILES

Rosales Mora, Alexis Jared (1), García Capulín, Carlos Hugo (2)

1 [Licenciatura en Ingeniería en Sistemas Computacionales, Universidad de Guanajuato] | [aj.rosalesmora@ugto.mx]

2 [Departamento de Ingeniería Electrónica, División de Ingenierías, Campus Irapuato-Salamanca, Universidad de Guanajuato] | [carlosg@ugto.mx]

Resumen

En la actualidad existen aplicaciones móviles cada vez más complejas, pero limitadas al uso de los recursos en los dispositivos móviles. Por lo tanto, el eficiente manejo de hardware se convierte en un elemento clave en el desempeño de una aplicación, especialmente si se trata de una aplicación que requiere un alto cálculo computacional como es el caso del procesamiento de imágenes. En vista de esta problemática, en el desarrollo de esta investigación se implementa el uso del lenguaje nativo del dispositivo, correspondiendo a el uso del lenguaje C y la plataforma de desarrollo NDK, obteniendo como consecuencia un mejor desempeño en los algoritmos implementados.

Abstract

Nowadays, mobile applications are increasingly complex, but limited in terms of the use of resources on mobile devices. Therefore, the efficient handling of hardware becomes a key element in the performance of an application, especially if it is an application that requires a high computational calculation such as the case of image processing. In view of this problem, in this research the use of the native language is implemented, corresponding to the use of the language C and the NDK development platform, obtaining as a consequence a better performance in the algorithms implemented.

Palabras Clave Cómputo móvil; Android; Lenguaje nativo; NDK;

INTRODUCCIÓN

Android es una de las plataformas más populares utilizadas en sistemas embebidos, televisores, tabletas electronicas, y especialmente en teléfonos inteligentes. Generalmente las aplicaciones de android están desarrolladas en lenguaje java, sin embargo es muy lento en caso de que se requiera operaciones de cálculo tales como procesamiento de imágenes [1]. Android NDK es un conjunto de herramientas que permiten implementar lenguajes de código nativo como C y C++, esto brinda un mejor desempeño en aplicaciones que necesitan mayor velocidad y capacidad de cálculo [2][3].

El procesamiento digital de imágenes abarca procesos cuyas entradas y salidas son imágenes, los cuales son procesos que extraen atributos de imágenes, hasta e incluyendo el reconocimiento de objetos individuales dentro de una imagen [4].

En esta investigación se plantea el desarrollo de una aplicación para el procesamiento de imágenes en dispositivos móviles. Los algoritmos que se implementaran son el negativo, binarización, suavizado y extracción de bordes. Dado que los algoritmos mencionados requieren el uso de gran cantidad de memoria así como gran capacidad de cálculo, se implementara el uso del NDK y lenguaje C.

MATERIALES Y MÉTODOS

Se hizo uso del lenguaje nativo del dispositivo móvil para mejorar el desempeño de la aplicación, debido a que para el cálculo necesario del procesamiento de imágenes, es más eficiente usar un lenguaje como C, en comparación con JAVA. Sin embargo, android no permite realizar algunas acciones mediante el lenguaje nativo, como es el caso de abrir o guardar archivos en el dispositivo móvil. Debido a lo antes mencionado se codificó en JAVA los metódos necesarios para abrir una imagen desde la galería del dispositivo, capturar y guardar una fotografía mediante la camara del dispositivo. Como se sabe, el lenguaje C no cuenta con funciones directas que permitan trabajar imágenes, por lo tanto se hizo uso de una biblioteca nativa de android llamada jnigraphics, la que permite trabajar con archivos de imagen tipo Bitmaps.

Se creó una clase particular dónde se desarrollaran en lenguaje C todos los algoritmos de procesamiento de imágenes que se tienen como objetivo realizar. La clase tiene un metódo principal el cuál recibe como parametros, la imagen de entrada, una imagen dónde se guardaran los valores de los píxeles procesados, y un identificador que determina el procesamiento que se desea aplicar. Éste metódo hace el llamado del algoritmo de procesamiento requerido y regresa el tiempo de ejecución que demoró en procesar la imagen de entrada.

Algoritmos de procesamiento de imágenes implementados

Negativo. El algoritmo procesa los píxeles de la imagen original de manera que sus valores quedan invertidos, es decir, los píxeles con mayor intensidad ahora serán los de menor intensidad y viceversa.

Binarización. El algoritmo se implementó de manera en que la binarización sea multinivel, es decir, el usuario indica la cantidad de umbrales que desea aplicar, y el algoritmo genera los umbrales e intensidades de color correspondientes, haciendo que estos valores sean equidistantes.

Suavizado. El algoritmo se implementó para que el usuario determine las dimensiones de la ventana a considerar en el procesamiento de la imagen original. Se desarrollaron dos variaciones del algoritmo, una dónde el suavizado se aplica usando el promedio de la ventana y la segunda ultilizando la mediana de la ventana en turno.

Extracción de bordes. El algoritmo implementado para la extracción de bordes, hace uso del operador de convolución. El usuario determina que máscara usar para la convolución, las máscaras disponibles para aplicar son: Operador Laplaciano, Operador de Prewitt PNW y Operador de Sobel SNW.

RESULTADOS Y DISCUSIÓN

Negativo de una imagen

Para el algoritmo del negativo de una imagen, se realizo la prueba con una imagen de entrada en escala de grises.

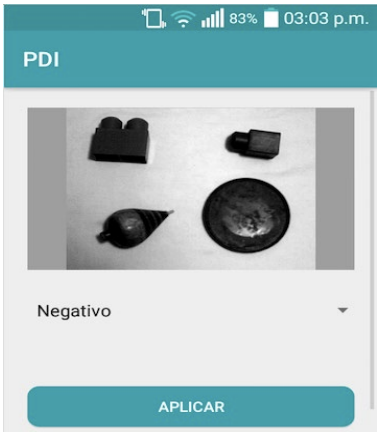

IMAGEN 1: Captura de pantalla de la aplicación, dónde se aprecia la imagen de entrada.

IMAGEN 2: Captura de pantalla de la aplicación, dónde se aprecia la imagen resultante una vez aplicado el procesamiento.

En la imagen 1, se muestra la imagen de entrada y así mismo, el algoritmo que se le aplicara. Como se puede apreciar en la imagen 2, la imagen resultante de la implementación del algoritmo, da como resultado las intensidades de color invertidas, es decir, el negativo de la imagen original. También se muestra el tiempo de ejecución del algoritmo, el cual fue de poco más de 1 décima de segundo.

Binarización de una imagen

Para la prueba del algoritmo de la binarización, la imagen de entrada esta en escala de grises, debido a que favorece el procesamiento de la imagen. En caso de que la imagen de entrada este a color, la aplicación la convierte a escala de grises para posteriormente binarizarla.

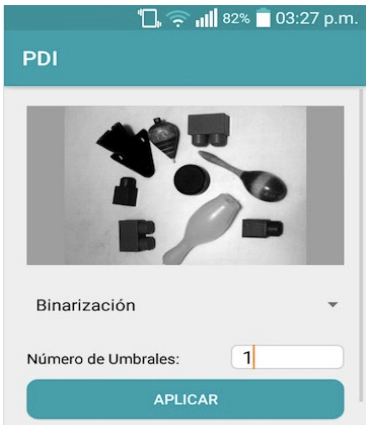

IMAGEN 3: Captura de pantalla de la aplicación, dónde se aprecia la imagen de entrada.

IMAGEN 4: Imagen resultante cuando el número de umbrales es 1.

En la

IMAGEN 5: Imagen resultante cuando el número de umbrales es 2.

imagen 3, se ilustra la imagen de entrada y se indica que el número de umbrales es uno. Como resultado de la binarización con un umbral, se obtiene la la imagen 4, en la cual se puede percibir que solo tenemos dos intensidades de color: el blanco y negro, por lo tanto el funcionamiento es el esperado. En la imagen 5 se aprecia el resultado de una binarización con dos umbrales, de este modo, la imagen resultante tiene tres intensidades de color: blanco, negro y gris. En las imágenes 4 y 5, se muestra el tiempo de ejecución del algoritmo, el cual fue de 8 centésimas de segundo y 14 centésimas de segundo respectivamente.

Suavizado de una imagen

Para el algoritmo de suavizado, se usaron dos vertientes: suavizado por promedio y suavizado por mediana. Para ambos tipos de suavizado se uso la misma imagen de entrada, la cual tiene ruido en forma de píxeles blancos y negros, más conocido como sal y pimienta.

IMAGEN 6: Captura de pantalla, dónde se muestra la imagen de entrada para el suavizado.

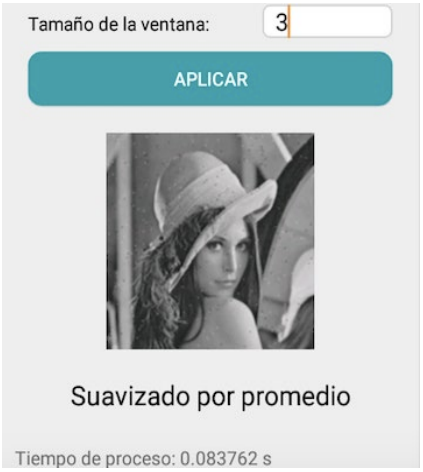

IMAGEN 7: Imagen resultante para el suavizado por promedio con una ventana de dimensiones 3x3.

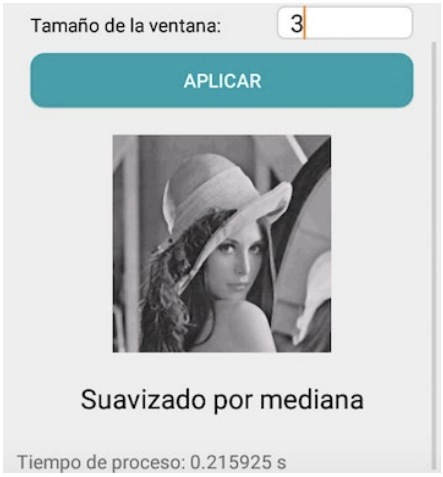

IMAGEN 8: Imagen resultante para el suavizado por mediana con una ventana de dimensiones 3x3.

En la imagen 6, se muestra la imagen de entrada para el suavizado. El resultado de aplicar el suavizado por promedio a la imagen original, con una ventana de 3x3, se obtiene el resultado que se muestra en la imagen 7, se puede percibir que el ruido sal y pimienta no fue completamente eliminado, solamente su intensidad bajo. Para el caso del suavizado por mediana con una ventana de 3x3, el ruido fue completamente eliminado como se puede apreciar en la imagen 8, sin embargo le toma un poco más de una décima de segundo ejecutarse.

Extracción de bordes

El algoritmo de extracción de bordes fue probado con la misma imagen de entrada, sin embargo usando dos máscaras diferentes para la convolución: operador laplaciano y operador prewit PNW.

En la imagen 9, se muestra la imagen original que sera procesada, en las imágenes 10 y 11, podemos ver el resultado de la extracción de bordes usando la máscara del operador laplaciano y operador prewit PNW respectivamente. El tiempo de ejecución de ambos casos es similar, sin embargo se percibe que los bordes son más marcados con el operador prewit, en comparación con el operador laplaciano.

IMAGEN 9: Captura de pantalla, dónde se muestra la imagen de entrada para la extracción de

IMAGEN 10: Imagen resultante de la extracción de bordes usando el operador laplaciano.

IMAGEN 11: Imagen resultante de la extracción de bordes usando el operador prewit.

CONCLUSIONES

Teniendo en cuenta los inconvenientes que genera la limitada cantidad de recursos que puede tener un teléfono móvil, en esta investigación se busco optimizar el desarrollo de una aplicación que procese imágenes. Concluyendo que con el uso de la plataforma NDK que nos brinda Android y nos permite hacer uso del lenguaje C, el funcionamiento de la aplicación se beneficia en cuanto al desempeño de los algoritmos de procesamiento de imágenes aplicados. Al final de la investigación se obtuvieron resultados que cumplen con las expectativas, teniendo en cuenta los pocos recursos del dispositivo en que se realizaron las pruebas.

AGRADECIMIENTOS

Los autores agradecen el apoyo de la Universidad de Guanajuato a través de la Dirección de Apoyo a la Investigación y Posgrado (DAIP), a los organizadores del Verano de Investigación 2018, quienes a traves de este tipo de programas fomentan el gusto por la investigación cientifica.

REFERENCIAS

[1]. Son, K. C., & Lee, L. Y. (2011). The method of Android application speed up by using NDK. In Awareness Science and Technology (iCAST), 2011 3rd International Conference on (pp. 382-385).
[2]. Lee, J. K., & Lee, J. Y. (2011). Android programming techniques for improving performance. In Awareness Science and Technology (iCAST), 2011 3rd International Conference on (pp. 386-389).
[3]. Lee, S., & Jeon, J. W. (2010). Evaluating performance of Android platform using native C for embedded systems. In Control Automation and systems (iccas), 2010 international conference on (pp. 1160-1163).
[4]. Gonzalez, R. C., & Woods, R. E. (2002). Digital image processing second edition. Beijing: Publishing House of Electronics Industry, 455.