La Mecatrónica en México, Vol. 4, No. 3, páginas 104 – 111, Septiembre 2015
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2015 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

Sistema de señalización de movimiento basado en dispositivos Android y Flujo Óptico

Blanco-Silva Gonzalo-Elías, Jesús Carlos Pedraza Ortega, Efrén Gorrostieta Hurtado, Juan Manuel Ramos Arreguín

Universidad Autónoma de Querétaro, Facultad de Informática
Av. de las Ciencias S/N Campus Juriquilla, Juriquilla, Querétaro, Qro.
jramos@mecamex.net

Resumen

Este trabajo presenta un sistema de señalización de movimiento para plataformas móviles, el cual emplea como núcleo de funcionamiento el algoritmo de Horn & Schunck, el cual es utilizado para la determinación de la dirección del movimiento y la velocidad de movimiento de puntos en una imagen. El sistema es comparado frente a una Computadora personal empleando el software Matlab con la finalidad de analizar los errores de medición entre ambos dispositivos y su respectivo tiempo de cómputo.

Palabras clave: Android, Flujo Óptico, Android, Dispositivo Móvil.

1. Introducción

La estimación del movimiento es el proceso mediante el cual se estiman las velocidades de objetos desplazándose en un espacio 3D y sus respectivas proyecciones en un plano 2D. Este tópico se encuentra sujeto a múltiples investigaciones y cuenta con una amplia variedad de algoritmos los cuales pueden ser clasificados de múltiples maneras. Una tentativa es la propuesta por Tabatabai [1], donde una primera clasificación es la existente entre técnicas de detección del movimiento 3D y técnicas de detección del movimiento 2D. Dentro de cada técnica existen distintos métodos que a su vez, algunos de ellos, pueden sub-clasificarse, y al mismo tiempo complementase con la clasificación propuesta por Barrón [2] tal como se muestra en la Figura 1. Dentro de esta diversidad de técnicas surgen dos enfoques el primero denominado campo de movimiento, el cual es una representación bidimensional de un movimiento tridimensional, y el segundo un enfoque que analiza el movimiento a partir de la obtención del flujo óptico, el cual implica la determinación la dirección del movimiento y la velocidad del movimiento en todos los puntos de la imagen [3].

Siendo este último la mejor medida del movimiento en el espacio bidimensional, y que aún hoy en día continúa siendo campo de investigación en busca de nuevas estrategias [4], métodos de evaluación [5] y técnicas para su cálculo [6, 7]. La razón es que los sistemas de visión artificial todavía no han logrado calcularlo de forma óptima (precisión elevada en poco tiempo).

Sin embargo, el cómputo del flujo óptico así como de algunos otros algoritmos, sufre de alta complejidad computacional. Es decir contrario a tareas donde no existe un requerimiento temporal, aplicaciones robóticas y espaciales, generalmente requieren el cómputo del flujo óptico en tiempo real. Así mismo, dada la aplicación el flujo óptico puede ser solo parte de un pre-procesamiento, lo que implica que debe contar con cierto requerimiento espacial que permita compartir recursos con otros procesos. Aunado a esto y según el dominio que se desee abordar, en algunos casos es necesario contar con cierta autonomía energética, y tener en consideración ciertas limitantes tanto de peso como de espacio [3].

La Mecatrónica en México, Vol. 4, No. 3, páginas 104 – 111, Septiembre 2015
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2015 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

Por lo cual resulta explorar los las estrategias dominantes en cuanto a la implementación de algoritmos y aplicaciones, si bien tradicionalmente los dispositivos y arquitecturas que toman forma de este tipo de procesos y aplicaciones son PCs, Procesadores Digitales de Señales (DSP por sus siglas en inglés), Microcontroladores y más recientemente Arreglos de Compuertas de Campo Programables (FPGA por sus siglas en inglés) [3], no se cuenta con la exploración de aplicaciones en cuanto al panorama informativo dominante actual, el cual tiende al uso de dispositivos móviles para esta finalidad [8].

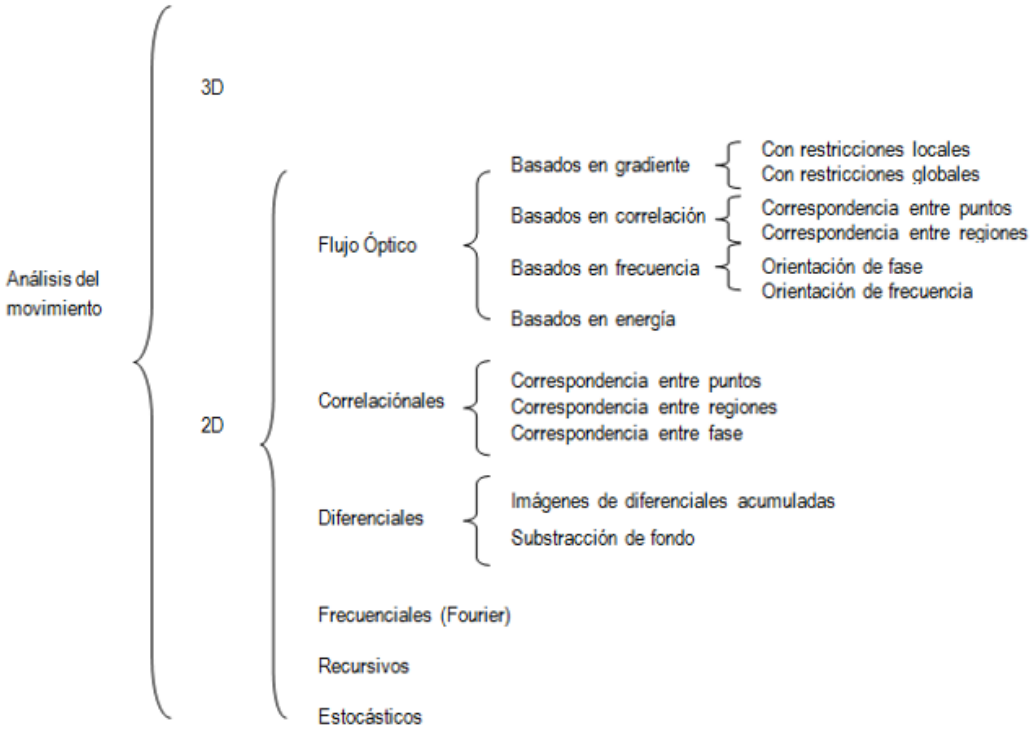


Figura 1.- Clasificación de las distintas técnicas para el análisis del movimiento, propuesta por Tabatabai (1998), con su respectiva complementación por Barrón (1994).

Si bien este panorama cuenta con limitaciones del mundo real y el entorno operativo de dispositivos móviles es diferente al tradicional, por lo cual es importante explorar las capacidades de los mismos.

Considerando este el presente artículo plantea el desarrollo de un sistema de señalización de movimiento empleando algoritmos de Flujo óptico, sobre una plataforma móvil con sistema operativo Android, el cual pueda ser adaptado a cualquier dispositivo móvil, así como explorar las limitantes, ventajas y desventajas, que surgen al emplear este tipo de plataformas dentro del marco de algoritmos de Flujo Óptico. En particular ha sido seleccionado el algoritmo de Horn & Schunck [3], debido a su simplicidad y desempeño en comparación con algoritmos similares [2, 4].

2. Algoritmo de Horn & Schunck

En todas las estrategias de estimación de flujo óptico se parte de la hipótesis de que los niveles de gris permanecen constantes ante movimientos espaciales en un tiempo dado. Dicha hipótesis da lugar a la ecuación general de flujo óptico (1).

$$Im(x,y,t) = Im(x+dx, y+dy, t+dt) \quad (1)$$

Donde $I(x,y,t): R^3 \rightarrow R$ corresponde a la representación escalar que representa la intensidad de cada pixel (x,y)en niveles de la escala de grises en un tiempo o dimensión temporal t. El interés radicara en

La Mecatrónica en México, Vol. 4, No. 3, páginas 104 – 111, Septiembre 2015
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2015 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

obtener el campo vectorial de condensación local $(dx, dy): R^2 \rightarrow R^2$ que representa el desplazamiento del patrón de iluminación, tal como lo presenta la Figura 2.

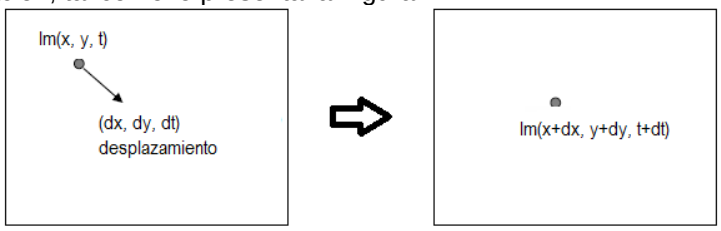


Figura 2.- Desplazamiento de patrones de iluminación en una imagen, representando la constancia de intensidad de cada pixel.

Así bien considerando que la intensidad luminosa varía de forma suave respecto de x, y, t y desarrollando por series de Taylor se tiene:

$$I(x, y, t) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \epsilon \quad (2)$$

Donde ϵ es el conjunto de términos de segundo orden y superiores. Así, despreciando ϵ, cancelando $I(x, y, t)$ de ambos lados de la igualdad, dividiendo por dt y considerando que este tiende a cero, se obtiene la ecuación:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \quad (3)$$

Que puede ser representada, de forma más compacta, como:

$$I_x u + I_y v + I_t = 0 \quad (4)$$

Donde I_x, I_y, I_t son las derivadas parciales de la imagen respecto a x, y, t respectivamente, y $v = (u, v)$ representa el vector de flujo óptico en cada punto a determinar. Esta expresión es conocida como la "Ecuación de restricción de flujo Óptico" (ERFO) [6].

Sin embargo (4) cuenta con dos vectores desconocidos u y v en una sola ecuación lineal. Por lo que es necesario introducir ciertas restricciones adicionales a fin de poder calcular estos vectores. Horn & Schunck (1981) introduce una ecuación adicional de restricción de suavidad. Esta segunda restricción es desarrollada en base a la observación a menudo en secuencias de video, los pixeles vecinos a un pixel de la imagen están sujetos a un movimiento similar al del pixel mismo. Este comportamiento resulta en un vector de campo de suave variación y una ecuación de restricción de la suavidad. El término de restricción es obtenido definiendo una función de costo adecuado. El primer componente de este costo es para representar el monto total de cambio en la imagen como una función de la posición de la imagen (x, y). Calculado como el cuadrado de las magnitudes de los gradientes de los vectores de flujo óptico y dado por:

$$\varepsilon_s^2 = u_x^2 + u_y^2 + v_x^2 + v_y^2 \quad (5)$$

Otro componente de la función de costo es acerca de la constancia de iluminación sobre una misma secuencia de imágenes:

$$\varepsilon_b = I_x u + I_y v + I_t \quad (6)$$

Dado (10) y (11), el problema puede ser reformulado como una minimización combinada de la función de costo calculado sobre la imagen completa:

$$\varepsilon^2 = \iint (\varepsilon_b^2 + \alpha^2 \varepsilon_s^2) dx dy \quad (7)$$

Aquí, α es un término de peso usado para ajustar la contribución relativa de los dos términos en la función de costo. Es posible ajustar la suavidad del campo de flujo óptico mediante el ajuste de α. Incrementando el valor de este parámetro se incrementara el peso del termino de suavidad en la función de costo y por lo tanto, el algoritmo dará lugar a un campo de flujo óptico más suave. Este parámetro juega un papel importante solo para áreas donde el gradiente de iluminación es pequeño, previniendo ajustes aleatorios para la estimación de la velocidad de flujo ocasionados por ruido en la estimación de las derivadas. Esta formulación acepta que durante la minimización, la iluminación se

La Mecatrónica en México, Vol. 4, No. 3, páginas 104 – 111, Septiembre 2015
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2015 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

verá minimizada pero no será exactamente cero. Este no es un problema debido a que en la práctica los valores de iluminación cambian muy levemente para ruido o cuantización de errores.

Los valores del par (u, v) que minimizan la función de costo presentada en (12) son el vector de campo del flujo óptico. Esto puede ser reformulado en la Ecuación (13):

$$\begin{aligned} I_x^2 u + I_x I_y v &= \alpha^2 \nabla^2 u - I_x I_t \\ I_x^2 I_y u + I_y v &= \alpha^2 \nabla^2 v - I_y I_t \end{aligned} \quad (8)$$

La Laplaciana de u y v puede ser aproximada usando la expresión mostrada en la Ecuación 14 donde $\bar{u}$ y $\bar{v}$ corresponden al promedio local de u y v respectivamente:

$$\begin{aligned} \nabla^2 u &= (\bar{u}_{i,j,k} - u_{i,j,k}) \\ \nabla^2 v &= (\bar{v}_{i,j,k} - v_{i,j,k}) \end{aligned} \quad (9)$$

Reemplazando las Laplacianas de u y v con su aproximación dada en (14), la Ecuación 13 puede ser reescrita como en (15).

$$\begin{aligned} (\alpha^2 + I_x^2 + I_y^2) u &= (\alpha^2 + I_y^2)\bar{u} - I_x I_y \bar{v} - I_x I_t \\ (\alpha^2 + I_x^2 + I_y^2) u &= -I_x I_y \bar{u} + (\alpha^2 + I_x^2)\bar{v} - I_y I_t \end{aligned} \quad (10)$$

Resolviendo u y v de la Ecuación (15), los vectores de flujo óptico son:

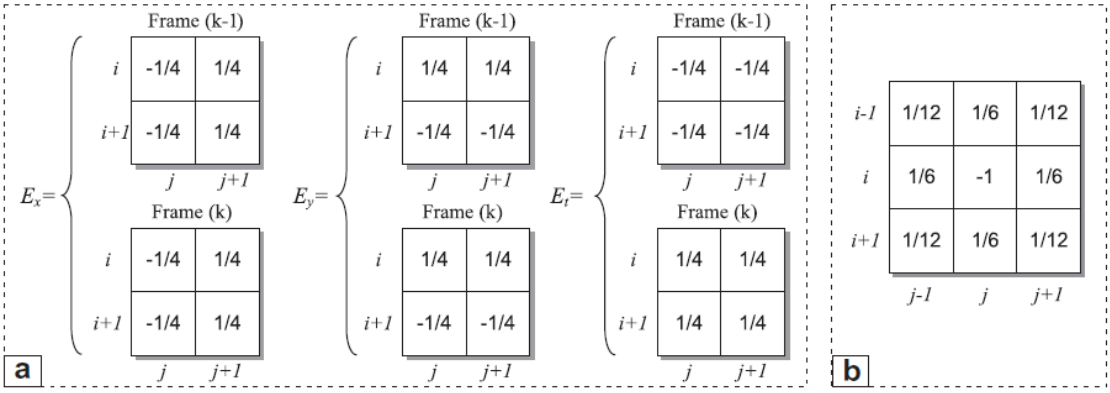
$$\begin{aligned} u &= \frac{(\alpha^2 + I_y^2)\bar{u} - I_x I_y \bar{v} - I_x I_t}{(\alpha^2 + I_x^2 + I_y^2)} \\ v &= \frac{-I_x I_y \bar{u} + (\alpha^2 + I_x^2)\bar{v} - I_y I_t}{(\alpha^2 + I_x^2 + I_y^2)} \end{aligned} \quad (11)$$

En la práctica, la solución es obtenida por el método iterativo de Gauss-Seidel:

$$\begin{aligned} u^{n+1} &= \bar{u}^n - I_x \frac{(I_x \bar{u}^n + I_y \bar{v}^n + I_t)}{(\alpha^2 + I_x^2 + I_y^2)} \\ v^{n+1} &= \bar{v}^n - I_y \frac{(I_x \bar{u}^n + I_y \bar{v}^n + I_t)}{(\alpha^2 + I_x^2 + I_y^2)} \end{aligned} \quad (12)$$

Donde n representa el número de iteración. Para el desarrollo computacional de la Ecuación (17), los gradientes y Laplacianos pueden ser estimados numéricamente. Esto es posible mediante diversas formas, un acercamiento al uso de este tipo de filtros espaciales es presentado en [4] y mostrado en la Figura 3. Para el cálculo del gradiente, Horn & Schunck (1981) utiliza una diferenciación de primer orden en un arreglo de ocho pixeles mostrado en la Figura 3.a. El Laplaciano es estimado mediante la substracción del valor en un punto promedio, ponderando los valores de los pixeles vecinos. Ellos aproximan la Laplaciana de los vectores de campo mediante el uso de una máscara de 3x3 mostrada en la Figura 3.b, para el cálculo de los promedios locales de vectores de flujo óptico, donde los subíndices (i, j, k) representan los números de fila, columna e imagen respectivamente. Haciendo uso de la máscara presentada en la Figura 3.a, el cálculo de I_x puede ser obtenido como en la Ecuación (18), de igual forma para los gradientes I_y y I_t.

$$I_x = \frac{1}{4}(I_{i,j+1,k-1} - I_{i,j,k-1} + I_{i+1,j+1,k-1} - I_{i+1,j,k-1} + I_{i,j+1,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i+1,j,k}) \quad (13)$$



La Mecatrónica en México, Vol. 4, No. 3, páginas 104 – 111, Septiembre 2015
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2015 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

Figura 3.- (a) Calculo número de (I_x, I_y, I_t) usando diferenciación de primer orden de ocho pixeles. (i, j, k) Indican fila, columna y número de imagen respectivamente. (b) Matriz de peso de 3x3 para la estimación de promedios locales $\bar{u}$ y $\bar{v}$ de los vectores de flujo óptico.

3. Desarrollo

Una vez considerado el algoritmo y su naturaleza se prosigue a su implementación en el entorno seleccionado, para lo cual se plantea el diagrama a bloques presentado en la Figura 4.

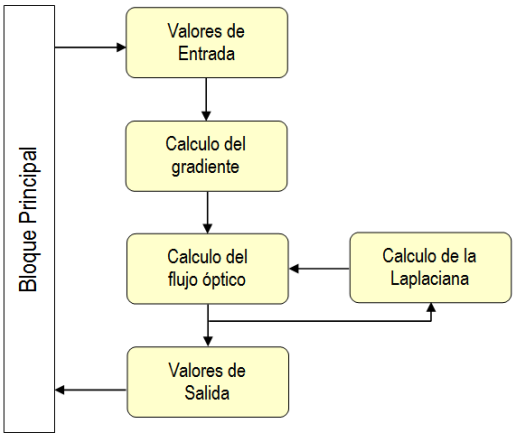


Figura 4.- Diagrama a bloques del algoritmo seleccionado.

Dentro del cual el bloque principal representa una arquitectura general de procesamiento de imágenes [9]; es decir mediante una etapa de adquisición, procesado y etapa final, la cual represente una interpretación del movimiento en la escena, tal como se presenta en la Figura 5.

Figura 5.- Etapas que constituyen un sistema para el procesamiento de imágenes.

Donde la etapa de procesamiento se encontrara aunada a un sistema de interpretación de valores de salida del algoritmo, a fin de identificar movimiento en la escena.

3.1 Desarrollo del sistema de monitoreo en entorno Android.

Una vez definidas las etapas y jerarquía de las mismas, es necesario elaborarlas en un entorno de programación que permita la generación de paquetes para el sistema operativo Android; es decir la generación de un archivo apk. El cual es un formato empleado para distribuir e instalar componentes empaquetados para la plataforma Android y contiene un conjunto la información necesaria para el funcionamiento del sistema.
Este tipo de archivos pueden ser desarrollados por programas tales como Eclipse IDE o Android Studio, las cuales son plataformas empleadas para el desarrollo de entornos de desarrollo integrados (por sus siglas en inglés IDE) [8].

Así pues se prosigue al desarrollo de un sistema de procesamiento de imágenes que emplee la cámara de un dispositivo Android como dispositivo de adquisición, la pantalla del mismo como dispositivo de salida e internamente se desarrolle la etapa de procesamiento encargada del sistema

La Mecatrónica en México, Vol. 4, No. 3, páginas 104 – 111, Septiembre 2015
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2015 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

de vigilancia y monitoreo. Para ello se ha escogido el entorno Eclipse IDE con los complementos necesarios. Teniendo como resultado un sistema que efectúa una comparativa entre pixeles con desplazamiento en la imagen mediante un umbral porcentual de los mismos si este umbral es superado se emplea una discriminación de colores para el reconocimiento de movimiento en la escena, así mismo la imagen puede ser segmentada a una zona de reconocimiento específica y el tamaño de la imagen puede ser parcializado dentro de la aplicación. Para la prueba de funcionamiento se ha seleccionado un dispositivo Inco Flex con Sistema Operativo Android versión 4.2 Jelly Bean, Pantalla 4.5", Camara 5 Mp y Procesador Cortex A9 Dual-Core. Teniendo como resultado el sistema presentado en la Figura 6.

Figura 6.- Sistema de monitoreo y vigilancia, implementando el algoritmo de Horn & Schunk.

4. Resultados

Una vez concebido el sistema es necesario analizar el desempeño del algoritmo propuesto, para lo cual se ha dispuesto de un sistema complementario al propuesto que sea capaz de cargar imágenes desde la memoria interna del dispositivo seleccionado, realice el procesamiento y cálculo del flujo óptico y al mismo tiempo exporte los vectores de desplazamiento obtenidos por el algoritmo en un formato legible por otra plataforma tal como es Matlab en la cual se ha desarrollado el mismo algoritmo bajo las mismas características a fin de realizar un procesamiento lo más parecido en ambos dispositivos, para esto se han desarrollado las etapas de procesamiento mediante el uso de bucles y no mediante el uso de funciones especiales. Así mismo, ha sido seleccionado como dispositivo comparativo una computadora portátil con procesador Intel Core i5 a 2.67 GHz, 8 Gb RAM, Sistema Operativo Windows 7 64 Bits.

4.1 *Banco de pruebas*

Como banco de imágenes para prueba del desempeño se han escogido las secuencias de imágenes Venus, Urban2, Urban3, Rubber Whale, Hydrangea, Grove2, Grove3 y Dimetrodon, mostradas en la Figura 7 y contenidas en Middlebury database [4].

Figura 9.- Secuencias de imágenes contenidas en Middlebury database, cada una comprende 8 cuadros y son clasificados de acuerdo a su naturaleza: Textura Oculta (RubberWhale, Hydrangea, Dimetrodon), Sintética (Grove, Urban) o Stereo (Venus).

4.2 *Banco de pruebas*

La Mecatrónica en México, Vol. 4, No. 3, páginas 104 – 111, Septiembre 2015
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2015 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

La medida más comúnmente usada para medir el desempeño del flujo óptico es el error promedio (AE por sus siglas en inglés). El AE es calculado entre un vector de flujo estimado (u, v) y un vector de referencia (u_r, v_r). Este puede ser calculado mediante el producto punto de los vectores, dividiendo por el producto de sus distancias, y entonces calculado la inversa del coseno.

$$AE = \cos^{-1}\frac{1 + u * u_r + v * v_r}{\sqrt{1 + u^2 + v^2}\sqrt{1 + u_r{}^2 + v_r{}^2}} \quad (19)$$

Esta medición fue introducida por [2]. Donde el objetivo del AE es aportar una medición relativa de desempeño que evite la división por cero. Errores de flujo amplio son penalizados menos que errores con flujo pequeño, teniendo como base una constante escalar arbitraria (1.0) para convertir las unidades de pixeles a grados.
Una aproximación más acertada en la obtención de AE es el uso del vector normalizado, mediante la obtención del producto punto y a continuación obtener el coseno inverso de su propio producto escalar.

Error de punto final EE

De igual forma es posible medir el error absoluto en un punto final de flujo (EE por siglas en inglés), usado por Otte & Nagel (1994) [10] definido como:

$$EE = \sqrt{(u - u_r)^2 + (v - v_r)^2} \quad (20)$$

Es posible obtener variantes de las mediciones anteriores, por ejemplo promediando los errores angulares en la imagen completa, se obtiene una medida global llamada error medio angular (AAE) para la imagen. Con el fin de evaluar el rendimiento de toda la secuencia, también es posible definir AAE como medición de una secuencia (SAAE) así como su respectiva desviación estándar (SD), mediante estas métricas es posible obtener histogramas completos no solo de esta medida sino para cada una de las medidas de desempeño de acuerdo al porcentaje de pixeles que tienen un error de medida alrededor de x [5].

4.3 *Resultados experimentales*

A continuación se presenta el Cuadro 1 el cual incluye una comparativa entre los valores obtenidos mediante la plataforma Android y el software Matlab, la comparación se realiza entre AAE y AEE, para las secuencias de video seleccionadas.

Cuadro 1. Evaluación de procesamiento Android vs PC del algoritmo propuesto

Secuencia	Tiempo Android	Tiempo PC	AAE	AEE
Venus	29.480	28.971	0.159	0.276
Urban 3	57.324	57.647	0.186	0.384
Urban 2	56.703	56.098	0.177	0.382
Rubber Whale	39.932	39.227	0.053	0.071
Hydrangea	39.265	39.671	0.176	0.338
Grove 3	50.497	50.919	0.248	0.539
Grove 2	50.800	49.847	0.212	0.374
Dimetrodon	34.088	34.628	0.071	0.098

El cuadro incluye el contraste en cuanto a tiempos sobre plataformas y los errores seleccionados entre ellos para un total de 50 iteraciones, y es posible concluir, que en cuanto a tiempos ambas plataformas presentan un valor muy aproximado, y en cuanto a los errores analizados aun cuando se

trata de 50 iteraciones y la propagación el error tiende a ser muy evidente por el error decimal se encuentra en términos de un valor permisible por la aplicación.

5. Conclusiones

El presente artículo describe el desarrollo de un sistema de señalización de movimiento mediante un discriminante de color que identifica zonas específicas de movimiento en secuencias de imágenes, tanto umbrales como valores de segmentación y escalamiento de imágenes pueden ser establecidas dentro de la aplicación, la cual implementa la detección de desplazamientos de pixeles mediante el algoritmo de Horn & Schunk previamente descrito sobre un paquete de apk, exportable a cualquier dispositivo Android, la idea de emplear este empaquetamiento es exportarlo a diferentes dispositivos y como trabajo a futuro implantarlo en cámaras, con la finalidad de tener sistemas de video inteligentes que sean autónomos en cuanto a la detección de movimiento. Así mismo se presentan pruebas del algoritmo en cuanto a desempeño y su respectiva comparativa entre dispositivos móviles y PCs, las cuales satisfactoriamente entregan resultados no muy diferentes entre dispositivos.

Referencias

[1] Tabatabai A., Jasindchi R. y Naveen T. “Motion estimation methods for video compression – A review”. Elsevier Science Ltd, J. Franklin Inst., 335B(8):1411–1441. (1998)
[2] Barron J. y Fleet D. “Performance of Optical-Flow Techniques”. International Journal of Computer Vision 12(1): 43-77. (1994).
[3] Horn B. y Schunck B. “Determining optical flow”. AI 17: 185–203. (1981)
[4] Koray G. y Saranli A. “An FPGA based high performance optical flow hardware design for computer vision applications”. Journal of Microprocessors and Mycrosystems, 37: 270-286. (2013)
[5] Baker S., Scharstein D., Lewis J., Roth S., Black M., y Richard S. “A database and evaluation methodology for optical flow”. International Journal of Computer Vision 92, 1-31. (2011)
[6] Sun D., Roth S., Lewis J. P. y Black M. “Learning Optical Flow”. ECCV 2008, Part III: 83-97. (2008)
[7] Sun D., Roth S. y Black M. “Secres of Optical Flow Estimation and Their Principles”. Computer Vision and Pattern Recognition CVPR : 2432-2439. (2010)
[8] Andrus J. y Nieh J. “Teaching Operating Systems Using Android”. SIGCSE’12 : 613-618. (2012)
[9] Pajares, G. y Cruz J. “Visión por Computador: Imágenes Digitales y Aplicaciones”. Editorial Rama. (2008)
[10] Otte M. y Nagel H. “Optical Flow estimation: Advances and compaisions” Lecture Notes in Computer Vision Science Volume 800: 49-60. (1994)