UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Visión Electrónica
Más que un estado sólido
https://revistas.udistrital.edu.co/index.php/visele

VISIÓN ELECTRÓNICA

VISIÓN DE CASO

Sistema embebido de detección de movimiento mediante visión artificial

Embedded System of Motion Detection Through Artificial Vision

Miguel Pérez.[1]***, Gloria Andrea Cavanzo Nisso.***[2]***, Fabian Villavisán Buitrago.***[3]

INFORMACIÓN DEL ARTÍCULO

Historia del artículo:
Enviado: 13/09/2017
Recibido: 22/11/2017
Aceptado: 12/01/2018

Palabras clave:
Raspberry Pi 3
Sistemas embebidos
Substracción de fondo
Tiempo real
Visión artificial.

Keywords:
Raspberry Pi 3
System embedded
Background subtraction
Real time
Artificial vision

RESUMEN

Este artículo presenta el diseño e implementación de un algoritmo de detección de movimiento, basado en la técnica de substracción de fondo, desarrollado en Python e implementado en un sistema Raspberry Pi 3. Se explica detalladamente el algoritmo y sus subsistemas internos, y se hacen pruebas de tiempo de su ejecución montado sobre el sistema embebido con sistema operativo GNU/Linux para determinar si estos algoritmos pueden correr en tiempo real sobre plataformas de bajo costo y tamaño reducido.

ABSTRACT

This paper presents the design and implementation of a Motion detection algorithm with background subtraction technique, developed in Python and implemented in a Raspberry Pi 3 system. The article a detailed explanation of the algorithm and its internal subsystems are found, and further runtime tests of the same mounted on the embedded GNU/Linux operating system to determine if these algorithms can run real-time on platforms of low-cost and small size.

[1]Ingeniero en control e instrumentación electrónica, Universidad Distrital Francisco José de caldas; magíster en Ciencias de la Educación, Universidad San Buenaventura, especialista en Pedagogía y Docencia Universitaria, Universidad de San Buenaventura. Docente de planta de la Universidad Distrital Francisco José de Caldas. Correo electrónico: mrperezp@udistrital.edu.co

[2]Matemática, magíster en Ciencias Matemáticas, Universidad Nacional de Colombia. Docente de la Universidad Distrital Francisco José de Caldas. Correo electrónico: gacavanzon@udistrital.edu.co

[3]Ingeniero en Control e Instrumentación Electrónica, Universidad Distrital Francisco José de Caldas. Correo electrónico: fabian2012@hotmail.es

Citar este artículo como: M. Pérez, G. A. Cavanzo y F. Villavisán. "Sistema embebido de detección de movimiento mediante visión artificial". Visión Electrónica, algo más que un estado sólido, Vol. 12, No. 1, 97-101, enero-junio 2018.
https://doi.org/10.14483/22484728.15087

1. Introducción

En la era moderna, los sistemas basados en visión artificial se han convertido en herramientas indispensables para los sistemas de seguridad privada, ya que son herramientas de bajo costo y de alto margen de precisión. Estos sistemas deben ser implementados en *hardware* cada vez más pequeño, con el fin de lograr una mayor portabilidad, rapidez y bajo costo; es por esto que dichos algoritmos están siendo migrados a plataformas embebidas cada vez más rápidas, lo que ha permitido correr estos sistemas en tiempo real.

Desde esta perspectiva, la Raspberry se considera como un ordenador del tamaño de una tarjeta de crédito. Aunque inicialmente se derivó de las investigaciones desarrolladas en la Universidad de Cambridge, y vio la luz en el 2012 como un ordenador de bajo costo que pudiese llegar al mayor número de usuarios posibles, fue finalmente creada por la Fundación Raspberry PI, sin ánimo de lucro. Este pequeño ordenador puede ser utilizado en proyectos de electrónica o para aquellos sistemas que necesiten portabilidad y economía [1].

En síntesis, la Raspberry Pi "es un ordenador de tamaño de tarjeta de crédito que se conecta a su televisor y a un teclado" [2], lo que significa que es una placa que soporta varios componentes en un ordenador común; así: "Es un pequeño ordenador capaz de ser utilizado por muchas de las cosas que su PC de escritorio hace, como: hojas de cálculo, procesadores de texto y juegos; también reproduce vídeo de alta definición" [2].

La Raspberry Pi 3 (Figura 1) es el tercer desarrollo de esta fundación y promete convertirse en una herramienta fundamental para los sistemas de visión artificial de bajo costo, portátiles, ya que su poder de cálculo y su velocidad se asemeja al de un computador personal.

Figura 1: Raspberry Pi 3 [3].

Desde el punto de vista técnico, posee un nuevo procesador —ARM Cortex A53— de cuatro núcleos a 1.2 GHz de 64 bits y que, según sus creadores, tiene un rendimiento diez veces superior al de la Raspberry Pi original, además de un 50 % más que la Raspberry Pi 2, el modelo anterior. En esta versión se incorporaron tecnologías de puertos de comunicaciones inalámbricas como lo son Bluetooth 4.1 (de bajo consumo) y Wifi 802.11n integrado, también cuenta con cuatro puertos USB, 40 pines GPIO (siglas del inglés *General Purpose Input/Output*), puerto HDMI (siglas del inglés *High-Definition Multimedia Interface*), puerto Ethernet, conector combo compuesto de audio y video de 3,5 mm, interfaz de la cámara (CSI, por sus siglas en inglés), interfaz de pantalla (DSI, por sus siglas en inglés), ranura para tarjetas microSD (ahora *push-pull* en lugar de *push-push*), núcleo de gráficos VideoCore IV 3D, dimensiones de placa de 8.5 por 5.3 cm. La Raspberry Pi 3 tiene un factor de forma idéntica a la anterior Pi 2 (1 y Pi Modelo B +) y tiene una compatibilidad completa con la Raspberry Pi 1 y la Raspberry Pi 2 [4].

En la literatura y en la industria se observan antecedentes de su aplicación en visión artificial. En el caso de Suryatali y Dharmadhikari, se aprovecharon las características de la Raspberry Pi modelo B para implementar un sistema de visión estéreo y algoritmos paralelos de visión artificial [5]; o el sistema de monitoreo facial mediante la incorporación de detección de rostros que implementa [6], el cual, aunque limitado en la detección del rostro de una sola persona, puede fácilmente detectar en qué posición se encuentra su cabeza.

Por otra parte, en el sistema de cobro de peaje mediante visión por computador desarrollado por [7], implementando sobre plataformas Linux embebidas en Raspberry, automáticamente se detecta el tipo de vehículo que ingresa al peaje y posteriormente genera su respectivo cobro, disminuyendo el costo de ejecución del proyecto mediante el uso de sistemas embebidos, si se trata de una gran cantidad de cabinas de cobro de peaje.

Por lo anterior, el presente artículo se centra en un sistema de video vigilancia con detección de movimiento automática, con la idea de facilitar la labor del operador del circuito cerrado de televisión (CCTV) e incrementar su fiabilidad.

El documento se ordena de la siguiente forma: en primer lugar, se indica el modelo matemático sintetizado en un algoritmo de detección de movimiento con substracción de fondo; posteriormente, se indica la metodología de trabajo para los subsistemas fundamentales del algoritmo; después, se establecen las

pruebas y resultados sobre imágenes, finalmente, se establecen las conclusiones.

2. Algoritmo de detección de movimiento

En los sistemas de video vigilancia mordernos la detección de movimiento automática juega un papel fundamental, pues esto facilita la labor del operador del sistema de CCTV e incrementa la fiabilidad del sistema; dichos algoritmos implementan técnicas de sustracción de fondo que permiten aislar el intruso del fondo que lo rodea y así facilitar su detección.

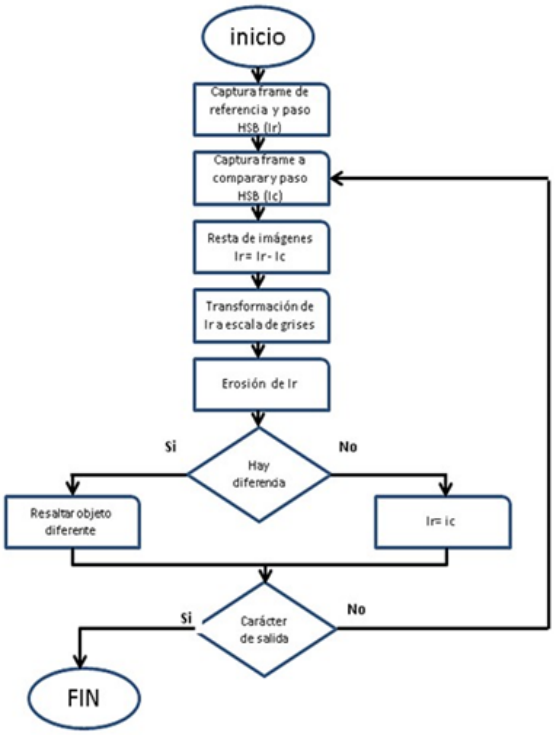
2.1. Substracción de fondo

La sustracción de fondo, también conocida como extracción de primer plano o de objetos en movimiento (*foreground extraction*), consiste en una serie de métodos que permiten distinguir entre zonas de fondo o estáticas (background) y zonas dinámicas que se corresponden con el primer plano (foreground) [8]. En este artículo se implementa el método ViBe [9,10], a nivel de píxel. Para cada píxel se almacenan N muestras de *frames* elegidos anteriormente, y se determina que son fondo aquellos que son matemáticamente invariantes en el tiempo, por medio de una operación de resta pixel a pixel puede guardarse tanto la información de color, por ejemplo en el espacio de colores RGB (siglas del inglés *Red, Green, Blue*) o HSV (siglas del inglés *Hue, Saturation, Value*), como su valor en tono de grises, dependiendo de la imagen de entrada con la que se cuenta.

3. Metodología

El diagrama de flujo de la Figura **2** muestra los subsistemas fundamentales del algoritmo. El primer paso es capturar un *frame* o imagen (imagen referencia ir) proveniente de una cámara convencional, después se realiza un acondicionamiento de dicha imagen, este se hace con el fin de descartar la información que no es relevante para el sistema, por ello, se aplican cambios en la imagen mediante trasformaciones espaciales (RGB a HSV). El cambio de espacio de colores primarios (RGB) a un espacio de progresión de color HSV, es necesario debido a la facilidad en este último para segmentar una imagen por medio de un intervalo de ángulos en coordenadas H (matiz) del espacio HSV (Figura **3**); en este espacio una sombra o un brillo no tiene mucha relevancia en la imagen a rastrear.

Figura 2: Diagrama de flujo algoritmo detección de movimiento.



Fuente: elaboración propia.

Figura 3: Representación de matiz en plano HSV [11].

La transformación

$$H \in [0°, 180°], S, V \in [0, 1] \leftarrow R, G, B \in [0, 255]$$

Con $Min = min(R, G, B)$ y $Max = max(R, G, B)$

Esta dada por (1).

$$H = \begin{cases} \text{indefinido para} & Max = Min \\ 60° \frac{(G-B)}{Max-Min} + 0° & si \;\; Max = R \; y \; G \geq B \\ 60° \frac{(G-B)}{Max-Min} + 360° & si \;\; Max = RyG < B \\ 60° \frac{(B-R)}{Max-Min} + 120° & si \;\; Max = G \\ 60° \frac{(R-G)}{Max-Min} + 240° & si \;\; Max = B \end{cases} \quad (1)$$

$$S = \begin{cases} 0 & si \; Max = 0 \\ 1 - \frac{Min}{Max} & \text{en otro caso} \end{cases} \qquad V = Max \quad (2)$$

Después que la imagen de referencia (Ir) y la imagen a comparar (Ic) son transformadas a formato HSV, se aplica la ecuación (3) que es la que determinará el cambio en la escena a rastrear; lo anterior se hace tomando la imagen de referencia —que es la encargada de decirle al algoritmo los parámetros nominales de la escena— y restarla pixel a pixel con el *frame* capturado en tiempo real Ic. Esta resta es una resta matricial dada por (3).

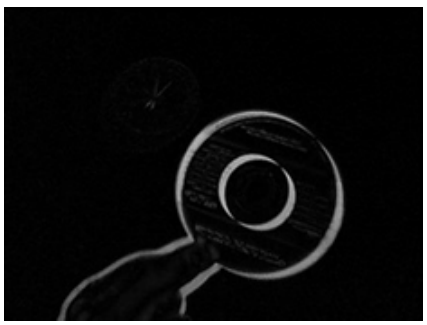
$$Ir_{ij} = Ir_{ij} - Ic_{ij} \quad (3)$$

La imagen Ir_{ij} contiene el nivel de cambio de la escena entre un *frame* y otro, que a la larga es la que determinara el rastreo del movimiento. El siguiente paso es transformar la imagen Ir_{ij} a una imagen a escala de grises aplicando la ecuación (4).

$$Ib(n) = \begin{cases} 1 & si \; Ir_{ij} \geq T \\ 0 & si \; Ir_{ij} < T \end{cases} \quad (4)$$

Donde T es un valor de umbral determinado por el usuario. La figura 4 muestra la imagen binarizada obtenida por el algoritmo

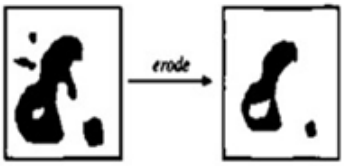
Figura 4: Imagen a escala de binarizada $Ib(n)$.



Fuente: elaboración propia.

En la imagen $Ib(n)$ se observan áreas en blanco que son muy pequeñas para tratarse de un objeto en movimiento, esto se debe a que pudo existir un cambio de brillos entre las imágenes, estos errores deben ser filtrados para que no afecten el resultado final del algoritmo, el filtro de erosión es un filtro basado en la morfología de los elementos que destaca únicamente el esqueleto de la imagen u objeto [12]. Un resultado explícito del resultado de este filtro se puede ver en la Figura 5.

Figura 5: Imagen erosionada [13].



Por último, se implementa un algo que resalte el objeto que se encuentre en la imagen después de la erosión, que es el mismo objeto que se encuentra en movimiento en la escena. La Figura 6, muestra el restado final del algoritmo y se puede observar la detección del movimiento del objeto en tiempo real.

Figura 6: Resultado final del algoritmo.

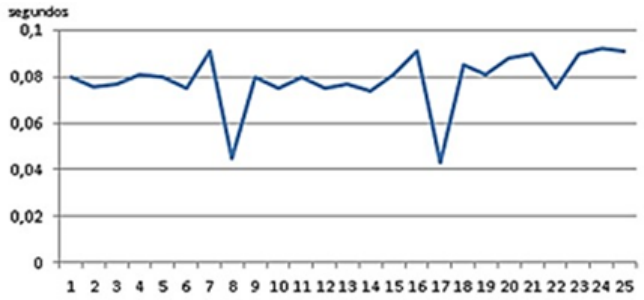
Fuente: elaboración propia.

4. Pruebas y resultados

El algoritmo se implementó en una Raspberry Pi 3 con sistema operativo Debian en su versión para Raspberry. El *software* implementado para el desarrollo de algoritmo es Python y su plataforma de desarrollo Stani's Python Editor (SPE). La librería para el procesamiento de video OpenCv, la librería para operaciones matemáticas Matplotlib y, por último, la librería encargada de medir los tiempos de ejecución de todas y cada una de las instrucciones Time.

Se hicieron veinticinco mediciones de tiempo de ejecución del algoritmo con la librería Time. Los resultados obtenidos se pueden ver en la Figura 7.

Figura 7: Mediciones de tiempo de ejecución, algoritmo de detección de movimiento.



Fuente: elaboración propia.

En la Figura 7 se pueden observar las mediciones de tiempo que se hicieron del algoritmo corriendo en la Raspberry. Se observa caramente que el algoritmo se ejecuta en su totalidad a un promedio de 80 ms, lo cual es un tiempo óptimo para sistemas que se consideren corriendo en tiempo real.

5. Conclusiones

La Raspberry Pi 3 resultó ser un *hardware* estable y compacto para montar sistemas de visión artificial, en especial aquellos que necesiten correr en tiempo real. Por otro lado, el algoritmo de sustracción de fondo ViBe [2,4] presenta un alto desempeño y cómoda implementación para sistemas de detección de movimiento.

Se determinó que el sistema implementado provee resultados a una —o de muestreo— alrededor de 80 ms, la cual da una tasa de 12,5 fotogramas por segundo, colmándose adecuadamente la expectativa frente al tiempo real.

Referencias

[1] Raspberry Pi Blog, "Where can i buy a Raspberry Pi?". [En línea]. Disponible en: https://www.raspberrypi.org/help/faqs/#buyingWhere

[2] Proyecto Byte, "Raspberry Pi 3. Características Técnicas". [En línea]. Disponible en: http://www.proyectobyte.com/android/raspberry-pi-3-caracteristicas-tecnicas

[3] Xataka, "Raspberry Pi 3 añade WiFi y Bluetooth, sigue costando 35 dólares". [En línea]. Disponible en: https://www.xataka.com/makers/raspberry-pi-3-anade-wifi-y-bluetooth-sigue\-costando-35-dolares

[4] M. David y J. Quiroga, "Estudio y Análisis del Algoritmo de Sustracción de Fondo Codebook", XV Simposio de Tratamiento de Señales, Imágenes y Visión Artificial, Bogotá D. C., 2011.

[5] A Suryatali y V. B. Dharmadhikari, "Computer Vision Based Vehicle Detection for Toll Collection System Using Embedded Linux", International Conference on Circuits, Power and Computing Technologies, Nagercoil, 2015. https://doi.org/10.1109/ICCPCT.2015.7159412

[6] A. A. Shah y Z. A. Zaidi, B. S. Chowdhry y J. Daudpoto, "Real time Face Detection/Monitor using Raspberry pi and MATLAB", 10th International Conference on Application of Information and Communication Technologies (AICT), Baku, 2016. https://doi.org/10.1109/ICAICT.2016.7991743

[7] G. Cocorullo, P. Corsonello, F. Frustaci, L. Guachi y S. Perri. "Embedded Surveillance system Background Subtraction and Raspberry Pi", International Annual Conference (AEIT), Naples, 2015. https://doi.org/10.1109/AEIT.2015.7415219

[8] L. Gervasoni1, J. D'amato y R. Barbuzza, "Aplicación de un método de sustracción de fondo a partir de imágenes de vídeo-vigilancia", 15th Argentine Symposium on Technology, Buenos Aires, 2014.

[9] O. Barnich y M. V. Droogenbroeck. "Vibe: a powerful random technique to estimate the background in video sequences", Acoustics, Speech and Signal Processing, Taipei, 2009.

[10] O. Barnich y M. V. Droogenbroeck. "Vibe: A universal background subtraction algorithm for video sequences". *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1709-1724, 2011. https://doi.org/10.1109/TIP.2010.2101613

[11] Wikibooks, "Color Models: RGB, HSV, HSL". [En línea]. Disponible en: https://en.wikibooks.org/wiki/Color_Models:_RGB,_HSV,_HSL

[12] Tanner Helland, "Seven grayscale conversion algorithms (with pseudocode and VB6 source code)". [En línea]. Disponible en: http://www.tannerhelland.com/3643/grayscale%E2%80%93image%E2%80%93algorithm-vb6/

[13] A. C. Bovik, "Handbook of image & video processing". [En línea]. Disponible en: https://www.sciencedirect.com/science/book/9780121197926#ancsection2