Polo del Conocimiento

Pol. Con. (Edición núm. 96) Vol. 9, No 8
Agosto 2024, pp. 1057-1065
ISSN: 2550 - 682X
DOI: https://doi.org/10.23857/pc.v9i8.7743

Detección de movimiento corporal para el control de sueño y fatiga en conducción de vehículos aplicando background substraction para el proyecto de conducción segura

Body motion detection for sleep and fatigue monitoring in vehicle driving using background subtraction for the safe driving project

Detecção de movimento corporal para controlo do sono e da fadiga na condução de veículos através da aplicação de subtração de fundo para o projeto de condução segura

Fabián Celso Gunsha-Maji [I]
fabian.gunsha@espoch.edu.ec
https://orcid.org/0000-0001-5637-1052

Ángel José Quevedo-Ríos [II]
angel.quevedo@espoch.edu.ec
https://orcid.org/0000-0003-2304-018X

Juan Carlos Castelo-Valdivieso [III]
j_castelo@espoch.edu.ec
https://orcid.org/0000-0001-9542-8074

Elvis Enrique Argüello [IV]
e_arguello@espoch.edu.ec
https://orcid.org/0000-0001-5083-1011

Correspondencia: fabian.gunsha@espoch.edu.ec

Ciencias Técnicas y Aplicadas
Artículo de Investigación

*** Recibido:** 05 de junio de 2024 ***Aceptado:** 17 de julio de 2024 *** Publicado:** 05 de agosto de 2024

I. Facultad de Mecánica, Escuela Superior Politécnica de Chimborazo (ESPOCH), EC060155, Ecuador.
II. Facultad de Mecánica, Escuela Superior Politécnica de Chimborazo (ESPOCH), EC060155, Ecuador.
III. Facultad de Mecánica, Escuela Superior Politécnica de Chimborazo (ESPOCH), EC060155, Ecuador.
IV. Facultad de Mecánica, Escuela Superior Politécnica de Chimborazo (ESPOCH), EC060155, Ecuador.

Resumen

En la actualidad la visión artificial y el procesamiento de imágenes están marcado un avance significativo debido a la evolución de los procesadores, dentro de este enfoque la detección de movimiento se ha desarrollado algoritmos que permiten ser más eficientes en ambientes donde la detección de movimiento resulta complicada, para los cuales OpenCV (Open Source Computer Vision) establece algunos métodos que son analizados mediante la programación de algoritmos en Python y son mostrados en este artículo que permitirán establecer estrategias de control de sueño y fatiga en la conducción del vehículo.

Palabras clave: Visión Artificial; OpenCV; background substraction.

Abstract

Currently, artificial vision and image processing are experiencing significant progress due to the evolution of processors. Within this approach, motion detection has been developed with algorithms that allow for greater efficiency in environments where motion detection is complicated. For this, OpenCV (Open Source Computer Vision) establishes some methods that are analyzed by programming algorithms in Python and are shown in this article that will allow establishing strategies to control sleep and fatigue while driving a vehicle.

Keywords: Artificial Vision; OpenCV; background subtraction.

Resumo

Atualmente, a visão artificial e o processamento de imagens estão a fazer progressos significativos devido à evolução dos processadores. o OpenCV (Open Source). Visão por Computador) estabelece alguns métodos que são analisados por algoritmos de programação em Python e são mostrados neste artigo que permitirão estabelecer estratégias de controlo do sono e da fadiga na condução do veículo.

Palavras-chave: Visão por Computador; OpenCV; subtração de fundo.

Introducción

En la actualidad la visión artificial y el procesamiento de imágenes están desarrollándose debido a la evolución de los procesadores dotándolos de mayor velocidad y paralelismo, provocando el

desarrollo de algoritmos que permiten ser mas eficientes en ambientes donde la detección de movimiento de un objeto es difícil como el mar debido a las olas o en la naturaleza por el movimiento causado por el viento.

Este documento aborda el uso de los métodos dados por la biblioteca OpenCV (Open Source Computer Vision) a través de un programa desarrollado en Python para la detección de movimiento utilizando BackgroundSubtractor, que permite operar la segmentación en primer plano y en segundo plano, mediante las clases BackgroundSubtractorKNN, BackgroundSubtractorMOG, BackgroundSubtractorMOG2, BackgroundSubtractorGMG, las cuales son evaluadas para verificar su eficiencia. Dentro de este contexto el Artículo

Motion Detection for Video Surveillance [6] muestra un algoritmo para la detección de movimiento (MODE) que es un método no paramétrico basado en pixeles que requiere solo un cuadro para construir el modelo, es independiente de las variaciones de iluminación, dinámicas y problemas de ruido. La detección de primer plano y fondo comienza desde el segundo fotograma en adelante, adicionalmente emplea un método de seguimiento de objetos que detecta y elimina los objetos fantasmas rápidamente mientras evita que los objetos abandonados se descompongan en el fondo.

Este artículo propone como principal característica dos modelos de color para almacenar la luminancia y la cromaticidad por separado. El primer modelo L consta únicamente de información de luminancia que se calcula a partir de RGB como muestra la ecuación 1:

$$L = 0.2116R + 0.7152G + 0.0722B \quad (1)$$

El segundo modelo de color propuesto l-r-g es tridimensional que consta de cromaticidad logarítmica y r-g como muestra la ecuación 2:

$$l = \cos(\theta)\ln\left(\frac{R}{G}\right) + \sin(\theta)\ln\left(\frac{B}{G}\right) \quad (2)$$

donde l es la imagen invariante y θ = 43.58 es la dirección de proyección calculada experimental. La cromaticidad r-g se calcula como muestra las ecuaciones 3 y 4.

$$r = \frac{R}{\sqrt{R^2 + B^2 + G^2}} \quad (3)$$

$$g = \frac{G}{\sqrt{R^2 + B^2 + G^2}} \quad (4)$$

Los autores comentan que su propuesta supera a otras técnicas de detección de movimiento de última generación debido a la propuesta de un nuevo modelo de color para resolver los problemas de sombreado y cambio de iluminación.

Además, se realiza una revisión de artículos en esta temática. El documento está distribuido por Introducción, Algoritmo desarrollado, donde se muestra el programa desarrollado en Python, resultados y finalmente se establecen las Conclusiones.

Algoritmo

Para el desarrollo del algoritmo se consideran tres sustractores de fondo disponibles en OpenCV: K-Nearest

Neighbors (KNN), Mixture of Gaussians (MOG2), and Geometric Multigrid (GMG).

Métodos

K-Nearest Neighbors (KNN). - Este método se desarrolla mediante la segmentación de fondo con el primer plano basada en K vecinos más cercanos, La clase implementa la resta de fondo de los K vecinos más cercanos descrita en [1]. Este método es muy eficiente si el número de pixeles de primer plano es bajo.

Mixture of Gaussians (MOG). - Este método es una mezcla de gaussianos que se propuso inicialmente en [2], basado en [3], esta se basa en una mezcla de k distribuciones gaussianas que modela cada pixel de fondo, con valores para k dentro de 3 y 5, las diferentes distribuciones representan cada una diferentes colores de fondo y de primer plano.

El peso de cada una de las distribuciones utilizadas en el modelo es proporcional a la cantidad de tiempo que cada color permanece en ese pixel. Por lo tanto, cuando el peso de una distribución de pixeles es bajo, ese pixel se clasifica como primer plano. En OpenCV, la implementación de MOG tiene parámetros de entrada que calibran el comportamiento del método como son:

history: Número de fotogramas que utilizara el método para acumular pesos en el modelo, durante todo el periodo de procesamiento. Resultado de valores bajos en mayor sensibilidad a cambios bruscos de luminosidad.

nmixtures: Cuantas distribuciones gaussianas debe realizar durante todo el video de tal manera que los valores más altos aumentan drásticamente el tiempo de procesamiento.

backgroundRatio: Define el peso umbral para la diferencia entre primer plano y el fondo. Los valores más bajos pueden incurrir en objetos falsos.

noiseSigma: Define el nivel de ruido aceptado. Los valores bajos crean objetos falsos.

Mixture of Gaussians (MOG2). - Este método MOG2 se basó en los artículos [4] y [5] con el objetivo de resolver una de las limitaciones que tenía MOG en la cantidad fija de distribuciones utilizadas, usando una variable que determina una cantidad de distribuciones gaussianas, mapeadas pixel a pixel, MOG2 logra una mejor representación de la complejidad de los colores en cada cuadro. Los parámetros de entrada se pueden cambiar por cada video diferente y estos son:

history: Similar al parámetro de MOG que denota el número de fotogramas que se utilizarán para modelar el fondo.

varThreshold: Correlaciona el valor del peso de los pixeles en el fotograma actual con los valores del modelo de tal manera que los valores más bajos de este parámetro tienden a crear objetos falsos.

detectShadows: Habilita o deshabilita la detección de sombras, habilitar este parámetro aumenta los tiempos de procesamiento.

Geometric Multigrid (GMG). - El método GMG, propuesto en [1], modela el fondo con una combinación de Inferencia bayesiana y filtros de Kalman. La primera etapa del método se acumula, para cada pixel, valores ponderados según el tiempo que permanezca un color en esa posición. Para cada cuadro, se agregan nuevas observaciones al modelo, actualizando estos valores. Colores que permanecen estáticos durante un

determinada cantidad de tiempo se consideran antecedentes.

La segunda etapa filtra pixeles en el primer plano para reducir el ruido de la primera etapa. La implementación de Python del método GMG tiene parámetros de entrada que pueden modificarse y estos son:

initializationFrames: Indica cuántos fotogramas tiene el algoritmo que vamos a utilizar para inicializar el método de modelado de fondo. Durante la inicialización, el fotograma resultante siempre es negro. Cuantos más fotogramas se utilicen en esta fase, más estabilizado estará el modelo inicial.

decisionThreshold: Determina el umbral en el que los pixeles se clasifican como fondo o primer plano. En la primera fase, cuando el algoritmo acumula valores basados en el tiempo que un color permanece estático, cada pixel con un valor ponderado más bajo, entonces el umbral se considera parte del fondo. Si la elección de valores altos para este parámetro puede ocasionar la pérdida de detecciones de objetos.

Algoritmo

El algoritmo que se muestra en la figura 1 está desarrollado en Python que permite comparar los métodos. El programa inicia con la lectura vía streaming VideoCapture(0) y la llamada al Algoritmo de segmentación almacenada en la variable **fgbg** según sea el caso a analizar:

- fgbg=cv2.createBackgroundSubtractorKNN()
- fgbg=cv2.createBackgroundSubtractorMOG()
- fgbg=cv2.createBackgroundSubtractorMOG2()
- fgbg=cv2.createBackgroundSubtractorGMG()

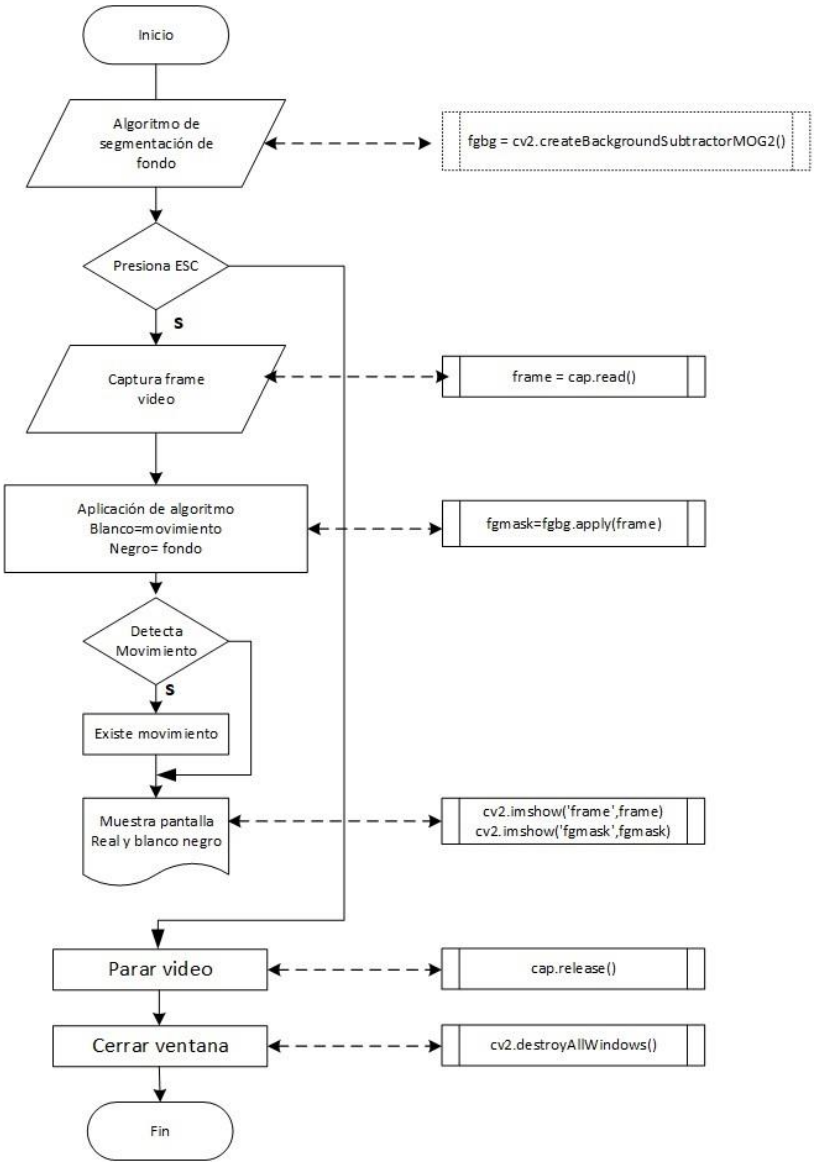

Fig. 1. Algoritmo BackgroundSubtractor

Pol. Con. (Edición núm. 96) Vol. 9, No 8, Agosto 2024, pp. 1057-1065, ISSN: 2550 - 682X

Para posteriormente iniciar un ciclo repetitivo en el que se realiza la captura del frame con frame=cap.read() y la aplicación del algoritmo de Background Subtractor mediante fgmask=fgbg.apply(frame) que permite obtener la imagen en blanco y negro, donde el objeto en movimiento es de color blanco.

Resultados

Los resultados obtenidos en otras investigaciones con BackgroundSubtractor MOG se puede apreciar varias personas en el video, mientras que con el método BackgroundSubtractor MOG2 se observa las sombras en las personas y finalmente con el método BackgroundSubtractor GMG las personas aparecen como una sola y no permite diferenciar , mientras que la figura 2 muestra los resultados experimentales desarrollados con KNN que no están claro el perfil dela persona mientras que MOG2 establece mejor el perfil de movimiento de la persona que permite detectar el movimiento del conductor para establecer alertas.

Fig. 2. Métodos probados experimentalmente

Pol. Con. (Edición núm. 96) Vol. 9, No 8, Agosto 2024, pp. 1057-1065, ISSN: 2550 - 682X

Conclusiones

En los artículos analizados se desataca el método para resolver los problemas de sombreado y cambio de iluminación mediante las ecuaciones que utilizan el color RGB que se muestra en el artículo detección de movimiento para supervigilancia.

El trabajo experimental desarrollado en Python sobre background substration determino que MOG y MOG2 resultaron más estables para detectar el movimiento del conductor, cabe señalar que el número de frames utilizados es 20 que permite una actualización rápida del proceso, pero el color y la iluminación afectan el proceso. Adicionalmente se debe considerar que Python establece 500 frames por defecto si no se declara provoca actualizaciones más lentas del proceso.

Referencias

1. Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," vol. 27, no. 7, 2006, pp. 773–780. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865505003521
2. P. Kaew Trakul Pong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," 2002.
3. C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149), vol. 2, 1999, pp. 246–252 Vol. 2.
4. Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., vol. 2, 2004, pp. 28–31 Vol.2.
5. A. B. Godbehere, A. Matsukawa, and K. Goldberg, "Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation," in 2012 American Control Conference (ACC), 2012, pp. 4305–4312.
6. B. Singh, D. Singh, G. Singh, N. Sharma, and V. Sibbal, "Motion detection for video surveillance," in 2014 International Conference on Signal Propagation and Computer Technology (ICSPCT 2014), 2014, pp. 578–584.

7. Y. Fang, J. Tang, W. Shen, W. Shen, X. Gu, L. Song, and G. Zhai, “Dual attention guided gaze target detection in the wild,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2021, pp. 11 390–11 399.