Detección de movimiento con OpenCV y Python

Comentarios(10)

Luis del Valle Hernández

En este artículo voy a explicar cómo podemos hacer la **detección de movimiento con OpenCV y Python**. Antes de comenzar debes tener claro que ventajas nos aporta la **visión artificial** y cómo podemos **empezar a programar y desarrollar nuestras propias aplicaciones de visión artificial**. El primer paso es preparar el sistema. Yo utilizo una plataforma donde tienes todo disponible en el mismo instalador, **Anaconda**. También hay que instalar la librería OpenCV para Python. Entiendo que a partir de ahora ya tienes todo dispuesto para empezar a programar así que comencemos.

Indice de contenidos

Detección de movimiento con OpenCV y Python

En muchas aplicaciones basadas en la visión artificial, se utiliza la detección de movimiento. Por ejemplo, cuando queremos contar las personas que pasan por un determinado lugar o cuántos coches han pasado por un peaje. En todos estos casos, lo primero que tenemos que hacer es extraer las personas o vehículos que hay en la escena.

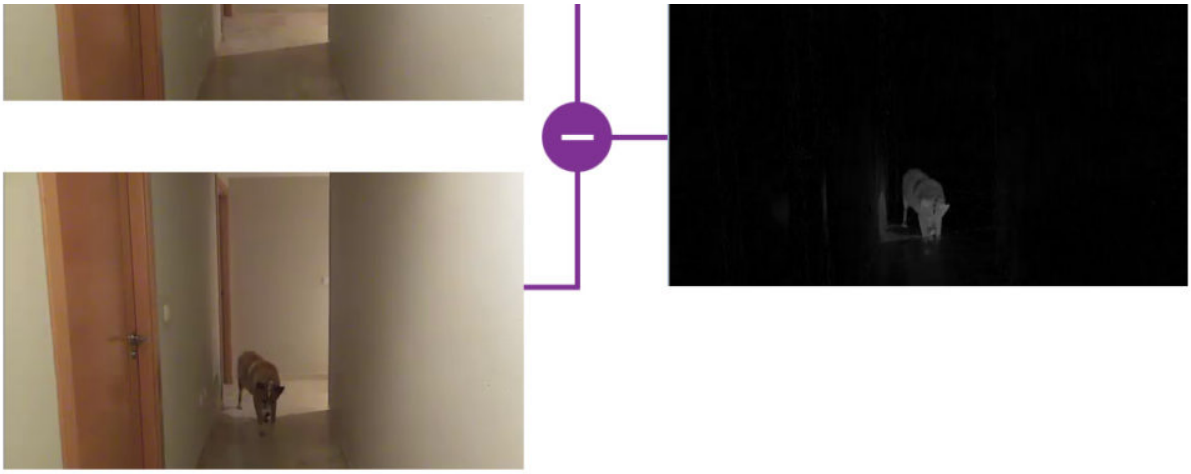
Existen diferentes técnicas, métodos o algoritmos que posibilitan la detección de movimiento. Al igual que en otras materias, en la visión artificial no hay casos genéricos. Dependerá de cada situación usar uno u otro. En este artículo te voy a mostrar 4 métodos para hacer la detección de movimiento con OpenCV y Python.

Los métodos que te voy a explicar a continuación están basados en la **substracción del fondo**, una técnica muy común en la detección de movimiento.

El objeto de este artículo no es entrar en detalle en la base matemática y científica que hay detrás de esta técnica. En el mundo en el que vivimos, el tiempo es un bien preciado. Voy a intentar ser práctico y te voy a dar los fundamentos de la sustracción de fondo y como utilizar esta técnica para la detección de movimiento con OpenCV y Python.

Substracción de fondo

La substracción de fondo consiste en tomar una imagen de la escena sin movimiento y restar los sucesivos fotogramas que vamos obteniendo de un vídeo. A la imagen sin movimiento se le llama **fondo o segundo plano** (background en inglés). El fotograma que vamos a analizar sería el **primer plano** (foreground en inglés). Por lo tanto, tenemos un fondo al que vamos restando los diferentes fotogramas.



El resultado, como ves en la imagen, es una escena con fondo negro. Donde se detecta movimiento, el color es diferente. Es una técnica muy sencilla. No requiere que el sujeto u objeto que se está intentando detectar deba tener algo que lo identifique como un sensor, baliza o traje especial. Por el contrario, la **substracción de fondo** es muy **sensible** a los **cambios de iluminación** como las sombras o los cambios producidos por la luz natural. Otra desventaja es que si el sujeto u objeto tiene un color parecido al del fondo, o **no se detecta el movimiento** o se detecta mal.

Dentro de la técnica de la substracción de fondo, existen dos modalidades. Dependerá de cómo se obtiene el fondo o segundo plano, con imagen de referencia o con fotogramas anteriores.

Substracción con imagen de referencia

Esta modalidad consiste en tener una imagen de referencia donde no haya ningún objeto en movimiento. A partir de esta imagen se obtienen los elementos en movimiento restando cada fotograma con la imagen de referencia. Normalmente se coge el primer fotograma de una secuencia de vídeo.

Es muy sensible a los cambios de luz. Imagínate que tomas la imagen de referencia en una habitación con luz natural. A las 10:00 de la mañana habrá unas condiciones de luz. Pero a las 18:00 de la tarde habrá otras. También es muy sensible a los movimientos de la cámara. Un movimiento muy pequeño puede hacer que se detecten falsos positivos en la escena. Por el contrario, este método funciona muy bien en entornos con iluminación controlada y detecta perfectamente la silueta de los objetos en movimiento.

dependerá de factores como la velocidad de los objetos.

Una de las mayores desventajas es que si el objeto o la persona en movimiento se quedan quietos, no se detecta. No es capaz de detectar siluetas. Sin embargo, es un método bastante robusto a los cambios de iluminación y a los movimientos de cámara. Consigue estabilizarse pasado un tiempo.

Fases del proceso de detección de movimiento

Ahora veremos cuales son las fases que debemos seguir para crear un algoritmo que nos permita la detección de movimiento con OpenCV. El proceso realizará varias tareas.

- Conversión a escala de grises y eliminación de ruido.
- Operación de substracción entre el segundo plano y el primer plano.
- Aplicar un umbral a la imagen resultado de la resta.
- Detección de contornos o blobs.

Algo muy común en las ciencias de la computación, en particular en la visión artificial, son los parámetros. Cada parámetro puede tener un rango de valores. El valor correcto dependerá de muchos factores. Está en nuestras manos adaptar cada valor a una situación concreta.

Existen diferentes técnicas que nos permiten estimar cual es el conjunto de valores que nos da mejor resultados. Una de ellas es **Simulated Annealing**. No te voy a hablar de esta técnica ya que es muy compleja y no es el objeto de este artículo, pero debes tenerlo en cuenta cuando tu objetivo es hacer una aplicación profesional. Es la única manera de ajustar los parámetros para obtener unos resultados aceptables.

Conversión a escala de grises y eliminación de ruido

Antes de realizar ninguna operación con las imágenes, es conveniente convertir a escala de grises. Resulta menos complejo y más óptimo trabajar con este tipo de imágenes. Por otro lado hay que minimizar el ruido provocado por la propia cámara y por la iluminación. Esto se hace a través de promediar cada píxel con sus vecinos. Se conoce comúnmente como suavizado.

```
# Suavizado de la imagen
gris = cv2.GaussianBlur(gris, (21, 21), 0)
```

Substracción entre el segundo plano y el primer plano

A la hora de restar una imagen con otra, debemos tener en cuenta que podemos obtener valores negativos. Para evitar esto lo que haremos será restar y obtener los valores absolutos de cada píxel. OpenCV nos proporciona un método para hacer esto.

```
# Resta absoluta
resta = cv2.absdiff(fondo, gris)
```

Aplicación de umbral

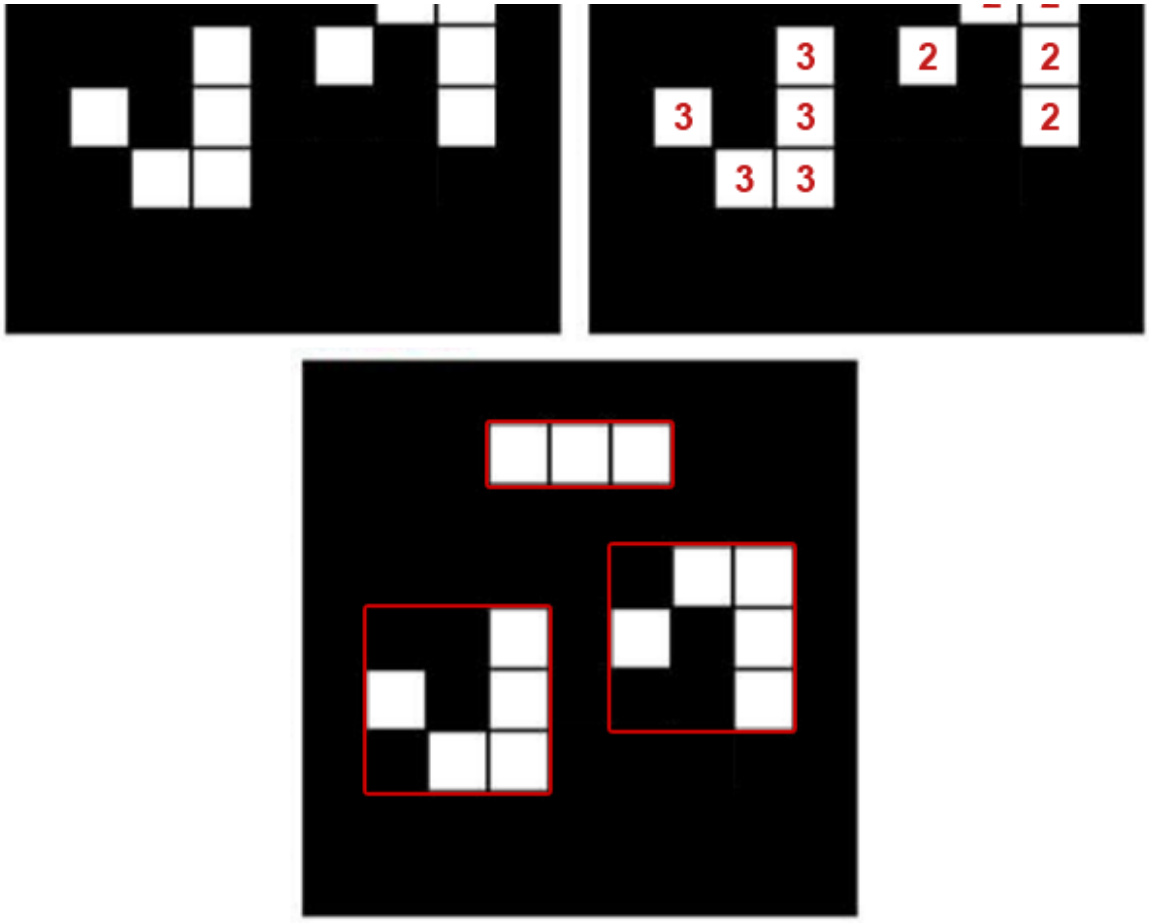
En esta parte del proceso lo que hacemos es quedarnos con aquellos píxeles que superen un umbral. El objetivo es binarizar la imagen es decir, tener dos posibles valores. Todos aquellos que superen el umbral serán píxeles blancos y los que no lo superen serán píxeles negros. Esto nos servirá para seleccionar el objeto en movimiento.

En OpenCV existe un método para aplicar un umbral.

```
# Aplicamos el umbral a la imagen
umbral = cv2.threshold(resta, 25, 255, cv2.THRESH_BINARY)
```

Detección de contornos o blobs

Una vez que tenemos la imagen con píxeles blancos o negros, tenemos que detectar los contornos o blobs. Un blob es un conjunto de píxeles que están conectados entre si es decir, tiene vecinos con el mismo valor. Cuando hablamos de vecinos es que estén al lado. En la siguiente imagen te muestro un ejemplo.



Se han detectado 3 blobs o contornos. Todos ellos tienen píxeles que están conectados entre si de alguna manera.

En OpenCV hay un método que busca los contornos de una imagen.

```
1 # Buscamos contorno en la imagen
2 im, contornos, hierarchy = cv2.findContours(umbral,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE
```

Por último y dentro de esta fase, tenemos que descartar los contornos más pequeños. Son aquellos que han pasado del suavizado y de la aplicación del umbral. Eso lo hacemos determinando un área mínima por el cual decidimos que son susceptibles de ser analizados.

Código

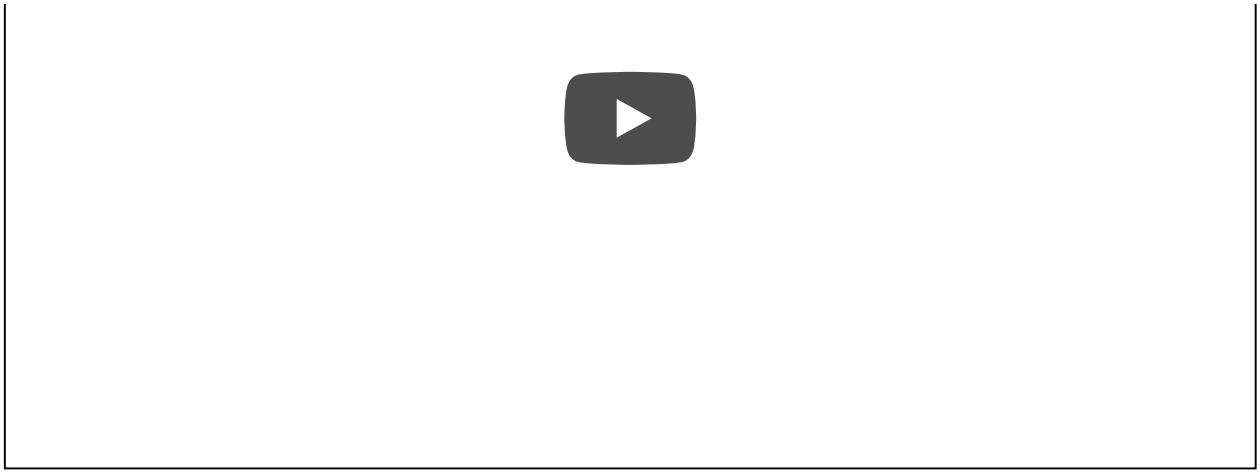
Ahora toca programar. Con todo lo que hemos visto podemos llegar a crear una aplicación para la detección de movimiento con OpenCV y Python. En este caso vamos a empezar con uno muy ligero, básico. Esto nos permitirá utilizarlo en aplicaciones donde los

OpenCV y Python

A continuación te dejo el código de este algoritmo basado en todo lo anteriormente explicado. Nos servirá para la detección de movimiento con OpenCV y Python básico.

```
    # Buscamos contorno en la imagen
    im, contornos, hierarchy = cv2.findContours(contornosimg,cv2.RETR_TREE,cv2.CHAIN_AP

    # Recorremos todos los contornos encontrados
    for c in contornos:
        # Eliminamos los contornos más pequeños
        if cv2.contourArea(c) < 500:
            continue


        # Obtenemos el bounds del contorno, el rectángulo mayor que engloba al contorno
        (x, y, w, h) = cv2.boundingRect(c)
        # Dibujamos el rectángulo del bounds
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Mostramos las imágenes de la cámara, el umbral y la resta
    cv2.imshow("Camara", frame)
    cv2.imshow("Umbral", umbral)
    cv2.imshow("Resta", resta)
    cv2.imshow("Contorno", contornosimg)

    # Capturamos una tecla para salir
    key = cv2.waitKey(1) & 0xFF

    # Tiempo de espera para que se vea bien
    time.sleep(0.015)

    # Si ha pulsado la letra s, salimos
    if key == ord("s"):
        break

# Liberamos la cámara y cerramos todas las ventanas
camara.release()
```

Puedes ver el resultado a continuación.



Métodos para la detección de movimiento con OpenCV y Python

Y como todo en la vida, existe un camino corto y en principio, «más sencillo». Este camino consiste en utilizar los métodos que nos proporciona OpenCV para la detección de movimiento. Están basados en trabajos realizados por investigadores y que se han incorporado al core de OpenCV. Dentro de la detección de movimiento con OpenCV hay varios algoritmos. Los tres que veremos aquí son:

- BackgroundSubtractorMOG.
- BackgroundSubtractorMOG2.
- BackgroundSubtractorKNN

BackgroundSubtractorMOG

Detectar la detección de movimiento con OpenCV y Python utilizando este método, puede ser sencillo con respecto a la programación. Lo realmente complejo es dar con los parámetros idóneos para tu proyecto. Aquí no queda otra que leer el **paper o documento científico** en el que se basó.

```
1 cv2.bgsegm.createBackgroundSubtractorMOG(history=200, nmixtures=5, backgroundRatio=0.7,
```

Donde:

- *history*: tamaño del histórico.
- *nmistures*: número de mezclas Gaussianas.
- *backgroundRatio*: relación de fondo.

```
# Importación de librerías
import numpy as np
import cv2

# Capturamos el vídeo
cap = cv2.VideoCapture('detector-movimiento-opencv.mp4')

# Llamada al método
fgbg = cv2.bgsegm.createBackgroundSubtractorMOG(history=200, nmixtures=5, backgroundRat

# Deshabilitamos OpenCL, si no hacemos esto no funciona
cv2.ocl.setUseOpenCL(False)

while(1):
    # Leemos el siguiente frame
    ret, frame = cap.read()

    # Si hemos llegado al final del vídeo salimos
    if not ret:
        break

    # Aplicamos el algoritmo
    fgmask = fgbg.apply(frame)

    # Copiamos el umbral para detectar los contornos
    contornosimg = fgmask.copy()

    # Buscamos contorno en la imagen
    im, contornos, hierarchy = cv2.findContours(contornosimg,cv2.RETR_TREE,cv2.CHAIN_AP

    # Recorremos todos los contornos encontrados
    for c in contornos:
        # Eliminamos los contornos más pequeños
```

Puedes ver el resultado a continuación.

cómo funciona este método, puedes leer la documentación científica en la que se basó. **Documento 1** y **documento 2**.

```
cv2.createBackgroundSubtractorMOG2(history=500, varThreshold=50, detectShadows=False)
```

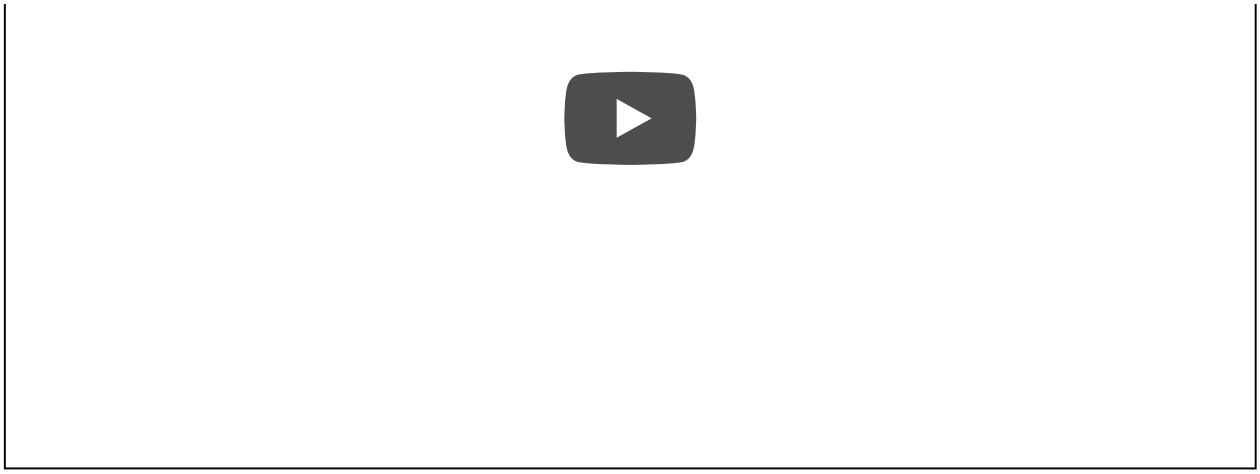
Donde:

- *history*: tamaño del histórico.
- *varThreshold*: umbral de la distancia de Mahalanobis al cuadrado entre el píxel y el modelo para decidir si un píxel está bien descrito por el fondo.
- *detectShadows*: con un valor verdadero (True) detecta las sombras. Esto reduce la velocidad así que si no se utiliza ponerlo a falso.

A continuación te dejo el código de ejemplo.

```
# Importación de librerías
import numpy as np
import cv2

# Capturamos el vídeo
cap = cv2.VideoCapture('detector-movimiento-opencv.mp4')

# Llamada al método
fgbg = cv2.createBackgroundSubtractorMOG2(history=500, varThreshold=50, detectShadows=F

# Deshabilitamos OpenCL, si no hacemos esto no funciona
cv2.ocl.setUseOpenCL(False)

while(1):
    # Leemos el siguiente frame
    ret, frame = cap.read()

    # Si hemos llegado al final del vídeo salimos
    if not ret:
        break

    # Aplicamos el algoritmo
    fgmask = fgbg.apply(frame)

    # Copiamos el umbral para detectar los contornos
    contornosimg = fgmask.copy()

    # Buscamos contorno en la imagen
    im, contornos, hierarchy = cv2.findContours(contornosimg,cv2.RETR_TREE,cv2.CHAIN_AP

    # Recorremos todos los contornos encontrados
    for c in contornos:
        # Eliminamos los contornos más pequeños
```

Puedes ver el resultado a continuación.



BackgroundSubtractorKNN

Este método se basa en el algoritmo **K-nearest neigbours**. Igual que los otros dos, tenemos que ser capaces de buscar los parámetros más adecuados para nuestro proyecto. Esto nos permitirá la detección de movimiento con OpenCV de una manera óptima.

```
1 cv2.createBackgroundSubtractorKNN(history=500, dist2Threshold=400, detectShadows=False)
```

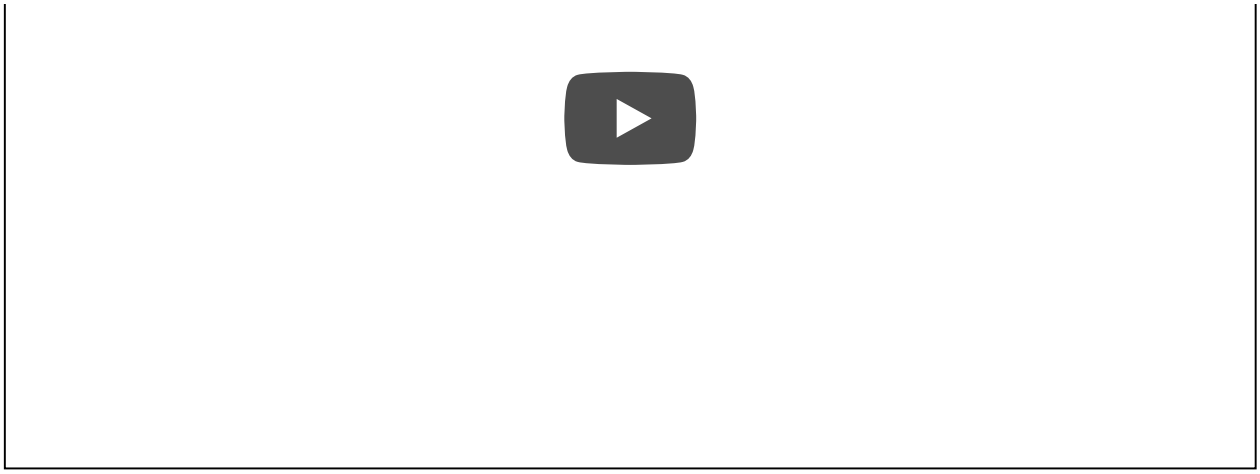
Donde:

- *history*: tamaño del histórico.
- *dist2Threshold*: umbral de la distancia al cuadrado entre el píxel y la muestra para decidir si un píxel está cerca de esa muestra.
- *detectShadows*: con un valor verdadero (True) detecta las sombras. Esto reduce la velocidad así que si no se utiliza ponerlo a falso.

A continuación te dejo el código de ejemplo.

```
# Importación de librerías
import numpy as np
import cv2

# Capturamos el vídeo
cap = cv2.VideoCapture('detector-movimiento-opencv.mp4')

# Llamada al método
fgbg = cv2.createBackgroundSubtractorKNN(history=500, dist2Threshold=400, detectShadows

# Deshabilitamos OpenCL, si no hacemos esto no funciona
cv2.ocl.setUseOpenCL(False)

while(1):
    # Leemos el siguiente frame
    ret, frame = cap.read()

    # Si hemos llegado al final del vídeo salimos
    if not ret:
        break

    # Aplicamos el algoritmo
    fgmask = fgbg.apply(frame)

    # Copiamos el umbral para detectar los contornos
    contornosimg = fgmask.copy()

    # Buscamos contorno en la imagen
    im, contornos, hierarchy = cv2.findContours(contornosimg,cv2.RETR_TREE,cv2.CHAIN_AP

    # Recorremos todos los contornos encontrados
    for c in contornos:
        # Eliminamos los contornos más pequeños
```

Puedes ver el resultado a continuación.



Conclusión

La detección de movimiento con OpenCV es una tarea que se puede complicar todo lo que queramos. Dependerá del objetivo de nuestro proyecto. Mi consejo es que si estás interesado te suscribas al curso de **Introducción a la Visión Artificial con OpenCV y Python**. **La única manera de hacer bien las cosas en entenderlas**.

Hola Luis, muy buen ejemplo. Quize ejecutar el codigo para una camara IP y me salio el siguiente error:

```
Traceback (most recent call last):
File "detect_mov.py", line 46, in <module>
im, contornos, hierarchy = cv2.findContours(contornosimg,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
ValueError: not enough values to unpack (expected 3, got 2)
```

Responder • Compartir ›

Luis del Valle Moderador ➚ Christian Cajusol Mio • hace 9 meses

Hola Chistian, gracias por los comentarios. Efectivamente, la versión 4.0 de OpenCV para el método findContours sólo devuelve dos parámetros. Quita im es decir, te quedaría así

contornos, hierarchy = cv2.findContours(contornosimg,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

Esto ya no tendría que dar error.

Responder • Compartir ›

Christian Cajusol Mio ➚ Luis del Valle • hace 9 meses

Genial Luis, funciono!, muchas gracias

Responder • Compartir ›

Luis del Valle Moderador ➚ Christian Cajusol Mio • hace 9 meses

De nada Christian ;)

Responder • Compartir ›

Paul • hace 10 meses

Hola Luis, muy buen contenido. ¿Como podría adaptarlo a una raspberry con el módulo picamera?

Responder • Compartir ›

Luis del Valle Moderador ➚ Paul • hace 10 meses

Hola Paul, en principio con este mismo código debería de funcionar con una Raspberry Pi si tienes montado OpenCV con Python.

Responder • Compartir ›

Alex M. • hace un año

Hola, muchas gracias por el artículo, he seguido todo al pie de la letra pero al ejecutarlo me salen los siguientes errores, alguna idea?

Responder • Compartir ›

Luis del Valle Moderador ➚ Alex M. • hace un año

Alex M. ➚ Luis del Valle • hace 10 meses

Muchas gracias ya me va perfecto, me gustaría a parte a partir de una región determinada del vídeo que haga conteo de todos los objetos que pasan, podría ser interesante, no se si tendrás alguna idea de como poder hacerlo... Un saludo

^ | v • Responder • Compartir ›

Lewis • hace un año

Hola Luis muy buen articulo!, para hacerlo con una cámara en movimiento que recomiendas. Y si quisiera agregarle un contador a esto como lo harias. Gracias un saludo!

^ | v • Responder • Compartir ›

Luis del Valle Moderador ➚ Lewis • hace un año

Hola Lewis, muchas gracias por tus comentarios. Con una cámara en movimiento es difícil pero debes actualizar el fondo cada poco tiempo dependiendo de cuánto se mueva.

1 ^ | v • Responder • Compartir ›

Lewis ➚ Luis del Valle • hace un año

Muchas gracias por la respuesta he decidido hacerlo estático por el momento. Pero el problema es que obtengo una ralentización extremadamente grande cuando ejecuto al frame. Cual podría ser una solución a esta lentitud y que el vídeo se reprodujese correctamente a la velocidad normal mientras se detectan los vehículos.

^ | v • Responder • Compartir ›

Luis del Valle Moderador ➚ Lewis • hace un año

Hola Lewis, la única solución es que bajes la resolución de las imágenes es decir, que sean imágenes más pequeñas.

1 ^ | v • Responder • Compartir ›

Milton Sanchez • hace un año

Hola, gracias por compartir la informacion. En el algoritmo que desarrollas se analiza un video previamente grabado; cómo cambiaría el codigo para analizar video en tiempo real? Gracias!!!

^ | v • Responder • Compartir ›

Luis del Valle Moderador ➚ Milton Sanchez • hace un año

El algoritmo sería exactamente igual lo único que tienes que cambiar la fuente para que adquiera el vídeo desde una webcam.

^ | v • Responder • Compartir ›

Pep q • hace 2 años

Hola, gracias por el articulo me ha sido de mucha utilidad, me podrías comentar que modalidad sería la más correcta para detectar un objeto que se acerca a la cámara estando esta en movimiento en la misma dirección que el objeto que se aproxima pero a menor velocidad. Gracias de nuevo

^ | v • Responder • Compartir ›

Gracias Luis

Responder • Compartir ›

Luis del Valle Moderador ➚ Pep q • hace 2 años

Gracias a ti Pep :)

Responder • Compartir ›

Jhonny Bautista • hace 2 años

Buenas, disculpa, en si cual es la linea que detecta el movimiento? Quiero hacer que la cámara comience a grabar vídeo, pero solo cuando detecte que hay movimiento. Espero me puedas ayudar. Gracias de antemano.

Responder • Compartir ›

Luis del Valle Moderador ➚ Jhonny Bautista • hace 2 años

Si entra dentro del for (for c in contornos:) es que ha detectado algo con el detector de movimiento con OpenCV.

Responder • Compartir ›

Juancarlos Orozco Navarro • hace 2 años

hola, me sale este error en la linea fgbg = cv2.createBackgroundSubtractorMOG2(history=500, varThreshold=50, detectShadows=False)
Error:

'module' object has no attribute 'createBackgroundSubtractorMOG2'

Responder • Compartir ›

Luis del Valle Moderador ➚ Juancarlos Orozco Navarro • hace 2 años

Eso tiene pinta de que la versión que has instalado no es la correcta. Comprueba la versión de OpenCV que tienes instalada.

Responder • Compartir ›

Juancarlos Orozco Navarro ➚ Luis del Valle • hace 2 años

Estoy utilizando la versión 2.4.9.1

Responder • Compartir ›

Luis del Valle Moderador ➚ Juancarlos Orozco Navarro • hace 2 años

Deberías actualizarte a la 3 para poder utilizar estas funciones

Responder • Compartir ›

Juancarlos Orozco Navarro ➚ Luis del Valle • hace 2 años

Ok, Muchas Gracias

Responder • Compartir ›

Luis del Valle Moderador ➚ Juancarlos Orozco Navarro • hace 2 años

Gracias a ti ;)

Luis del Valle Moderador ➚ JOAN SEBATIAN SIABATO FONSECA • hace 2 años

Hola Joan, no lo tengo, yo trabajo con Python. Pero seguro que si haces una búsqueda en Google lo encuentras enseguida.

▲ | ▼ • Responder • Compartir ›

TAMBIÉN EN **PROGRAMAR FÁCIL CON ARDUINO**

Dudas sobre Arduino y ESP8266

2 comentarios • hace 5 meses

Luis del Valle — Muchas gracias Kike

Introducción a Node-RED y Raspberry Pi con un sistema de alarma con Arduino

63 comentarios • hace un año

Luis del Valle — Hola Genaro, por supuesto que se puede hacer eso. No hay problema de conectar un equipo con Raspberry Pi y Node-RED con un

¿Cómo puedes hacer tus propios proyectos con Arduino?

2 comentarios • hace 6 meses

Luis del Valle — ¿Por qué?

Amplificador HX711 con Arduino para crear una báscula digital

6 comentarios • hace 5 meses

Luis del Valle — Hola Sebastián, tienes que calcular el valor absoluto de esa medida obtenida con la báscula digital con Arduino y HX711 es

Tutoriales de Arduino

Sensor temperatura Arduino
Tutorial pulsadores
Tutorial RTC reloj Arduino
Tutorial potenciómetro Arduino
Tutorial display 7 Arduino
Tutorial motor paso a paso Arduino
Tutorial librerías Arduino
Tutorial ultrasonidos Arduino
Tutorial I2C Arduino
Tutorial consumo Arduino
Tutorial weareable Arduino
Tutorial interrupciones Arduino
Tutorial ADS1115 ADC Arduino
Tutorial pilas Arduino
Tutorial riego Arduino
Arduino Shield EchidnaShield
Tutorial if else Arduino

Tutoriales ESP8266

Introducción ESP8266
Tutorial NodeMCU
IDE Arduino NodeMCU
Tutorial ESP-01
Relé WiFi Sonoff

Tutoriales IoT

Proyectos IoT con Arduino
MQTT ESP8266 Raspberry Pi
Redes LPWAN Arduino
IFTTT Arduino
Geolocalización Arduino
Arduino MKRFOX1200 Sigfox

Instalación OpenCV y Python
Introducción Vision Artificial
Detección movimiento OpenCV
Introducción OpenCV
Qué es Machine Learning
Sensor Kinect
Realidad Aumentada
Detector Canny OpenCV
Big Data Visión Artificial
Kinect y Processing
Deep Learning

Tutoriales Raspberry Pi

Tutorial Raspberry Pi
Proyectos Raspberry Pi
Vídeo Raspberry Pi
Servidor web Raspberry Pi
Servidor Raspberry Pi
Flask Raspberry Pi
Monitor bebé Raspberry Pi

Podcast

Entrevista Obijuan
Entrevista BricoGeek
Entrevista David Cuartielles
Entrevista Rinconingenieril
Entrevista Staticboards
Entrevista La Hora Maker
Entrevista OSHWDem
Entrevista Colepower
Entrevista Aprendiendoarduino
Entrevista BitFab

Circuit Playground Express y el robot de cartón

Hace tiempo que no grabo podcast. El último fue allá por el mes de agosto y se publicó en septiembre. Desde entonces no he parado. Aunque tu no me …

[+ info...]

Presentación de EnigmaIOT

El pasado 14 de septiembre tuve el placer de participar junto a otros miembros de The Things Network Madrid en el evento Tech.Party 2019, organizado …

[+ info...]

Cuánto cuesta crear un proyecto Maker profesional

¿Cuánto cuesta hacer un proyecto Maker? A veces creemos que el mero hecho de que la tecnología que se utiliza para hacer un proyecto sea abierta, …

[+ info...]