

Universidad de Guadalajara



Algoritmo 11

Muñoz Nuñez Ian Emmanuel

Visión Robótica

"Cambio de fondo de videos"

```
# Se importa opencv
import cv2
# Se importa sleep de time
from time import sleep

# Se carga el video 'leon.mp4'
captura = cv2.VideoCapture("leon.mp4")
# Se carga el fondo 'rotonda.jpg'
fondo = cv2.imread("rotonda.jpg")

# Se obtiene el ancho de la captura
width = int(captura.get(cv2.CAP_PROP_FRAME_WIDTH))
# Se obtiene el alto de la captura
height = int(captura.get(cv2.CAP_PROP_FRAME_HEIGHT))

# Se declara el codec que se usara para grabar los videos
fourcc = cv2.VideoWriter_fourcc(*"mp4v")

# Se define un archivo de salida con el nombre 'original.mp4' con el codec
# previamente declarado (fourcc), con 30 frames por segundo y con el ancho
# y alto de la captura
output_original = cv2.VideoWriter("original.mp4", fourcc, 30, (width, height))
# Se define un archivo de salida con el nombre 'fondo_video.mp4' con el codec
# previamente declarado (fourcc), con 30 frames por segundo y con el ancho
# y alto de la captura
output_fondo = cv2.VideoWriter("fondo_video.mp4", fourcc, 30, (width, height))

# Se inicializa el iterador en 0
i = 0
# Comienza el ciclo
while i < 150:
    # Con la variable 'leido' se sabe si hay mas frames en el video o no.
    # Se asignan los frames del video a la variable 'video'
    leido, video = captura.read()

    # Si no hay mas frames en el video...
    if leido == False:
        # ...El ciclo termina
        break

    # Se modifica el tamaño del fondo para que tenga las mismas
    # dimensiones que el video.
    fondo = cv2.resize(fondo, (video.shape[1], video.shape[0]))

    # Se modifica el formato del video de BGR a HSV
    video_hsv = cv2.cvtColor(video, cv2.COLOR_BGR2HSV)
```

```
# Se calcula la mascara del video
mask = cv2.inRange(video_hsv, (45, 150, 150), (55, 255, 255))
# Se calcula el negativo de la mascara
mask_negativo = cv2.bitwise_not(mask)
# Se aplica la mascara al video
video_mask = cv2.bitwise_and(video, video, mask=mask_negativo)

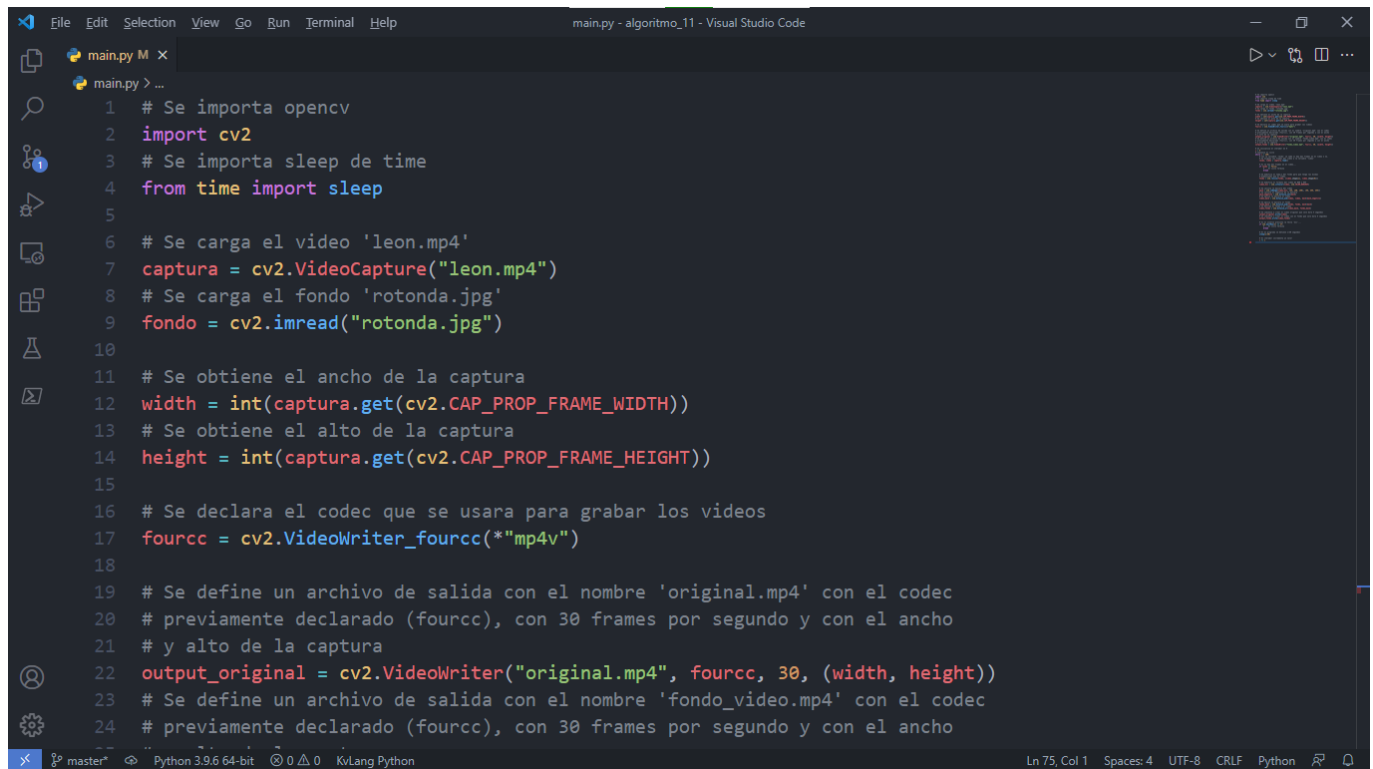
# Se aplica la mascara al fondo
fondo_mask = cv2.bitwise_and(fondo, fondo, mask=mask)
# Se mezclan el fondo y el video
video_fondo = cv2.bitwise_or(video_mask, fondo_mask)

# Se comienza a crear el video original que solo dura 5 segundos
output_original.write(video)
# Se comienza a crear el video con el fondo que solo dura 5 segundos
output_fondo.write(video_fondo)
# Si el usuario presiona la tecla 'esc'...
if cv2.waitKey(1) == 27:
    # ...El ciclo termina
    break

# El el programa se detiene 1/30 segundos
sleep(1/30)

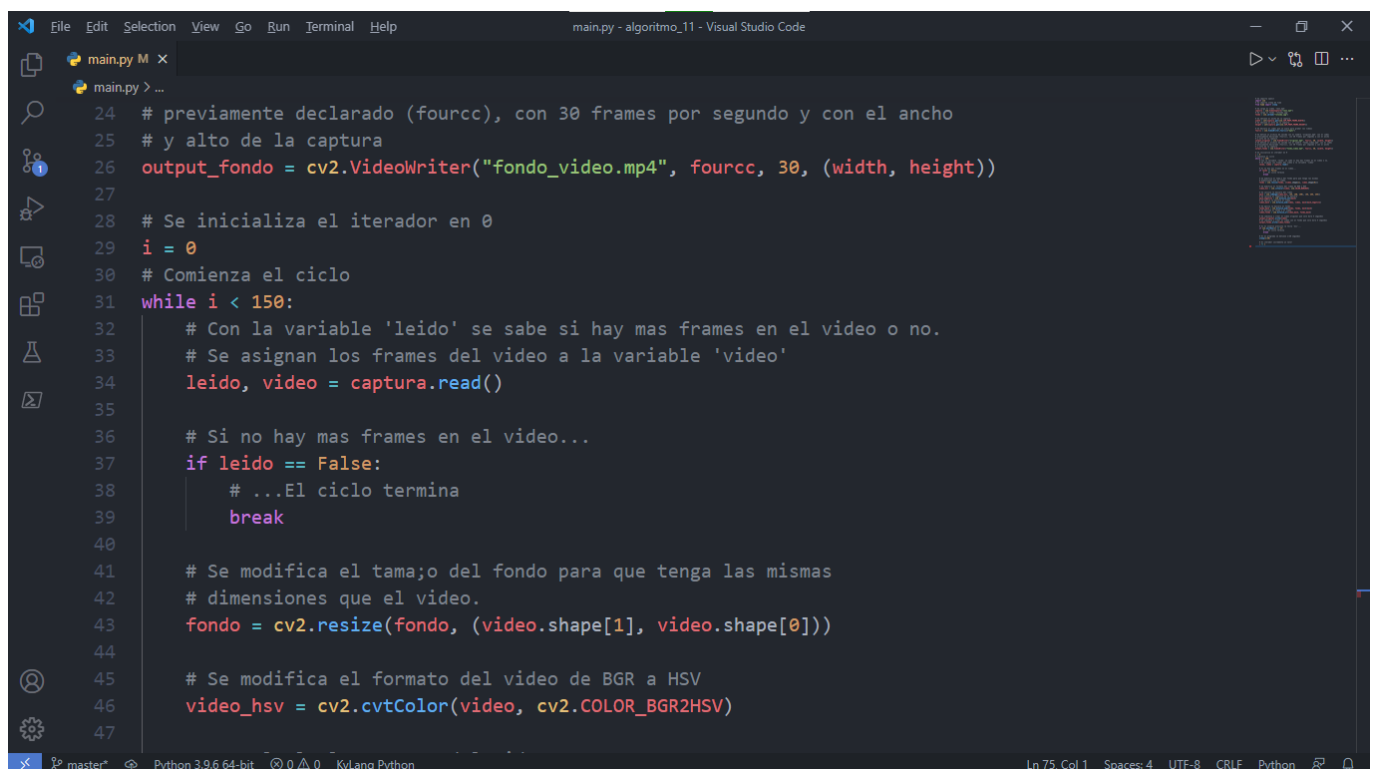
# El iterador incrementa un valor
i += 1
```

Para el código del algoritmo 11 primero se importan las librerías necesarias (**"opencv"** y **"time"**), se carga el vídeo original y la imagen de fondo que se usara para el algoritmo. Luego se obtiene el ancho y alto del vídeo, se declara el codec de los vídeos y se definen los archivos de salida. Después se comienza el ciclo para crear los archivos, se lee la captura, se cambia el tamaño del fondo para que sea de la misma magnitud del vídeo, se cambia el formato del vídeo de BGR a HSV, también se calcula la máscara del video, su negativo y se aplica la máscara al video y al fondo, al final se mezclan el fondo y el video, por último se escribe el vídeo original y con el cambio de fondo con un frame cada vez que se corre el ciclo. Si se acaban los frames del vídeo o el usuario presiona la tecla **"esc"** el ciclo termina. Por último, en cada ciclo el programa espera $\frac{1}{30}$ segundos y el iterador aumenta un valor.



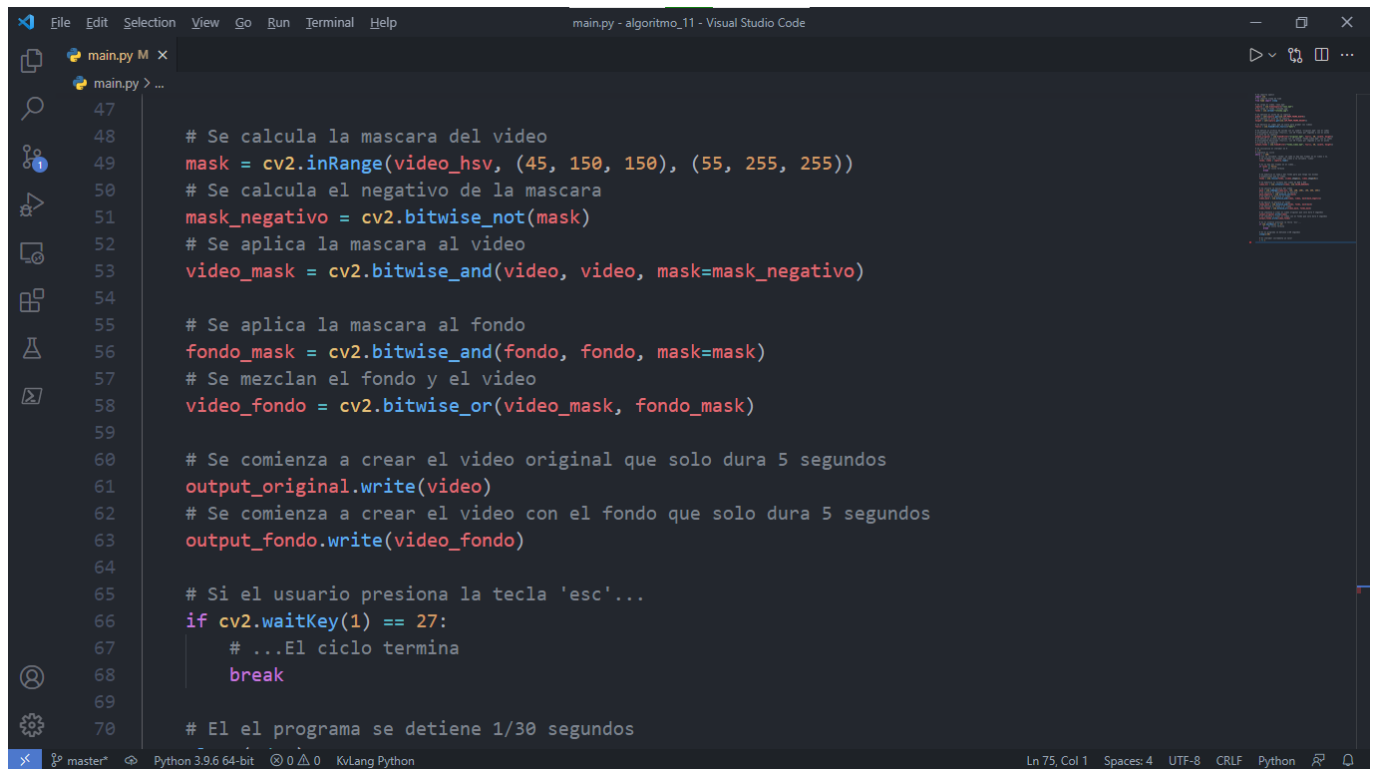
```
1 # Se importa opencv
2 import cv2
3 # Se importa sleep de time
4 from time import sleep
5
6 # Se carga el video 'leon.mp4'
7 captura = cv2.VideoCapture("leon.mp4")
8 # Se carga el fondo 'rotonda.jpg'
9 fondo = cv2.imread("rotonda.jpg")
10
11 # Se obtiene el ancho de la captura
12 width = int(captura.get(cv2.CAP_PROP_FRAME_WIDTH))
13 # Se obtiene el alto de la captura
14 height = int(captura.get(cv2.CAP_PROP_FRAME_HEIGHT))
15
16 # Se declara el codec que se usara para grabar los videos
17 fourcc = cv2.VideoWriter_fourcc(*"mp4v")
18
19 # Se define un archivo de salida con el nombre 'original.mp4' con el codec
20 # previamente declarado (fourcc), con 30 frames por segundo y con el ancho
21 # y alto de la captura
22 output_original = cv2.VideoWriter("original.mp4", fourcc, 30, (width, height))
23 # Se define un archivo de salida con el nombre 'fondo_video.mp4' con el codec
24 # previamente declarado (fourcc), con 30 frames por segundo y con el ancho
```

Figura 1: Captura 1 del código del algoritmo 11



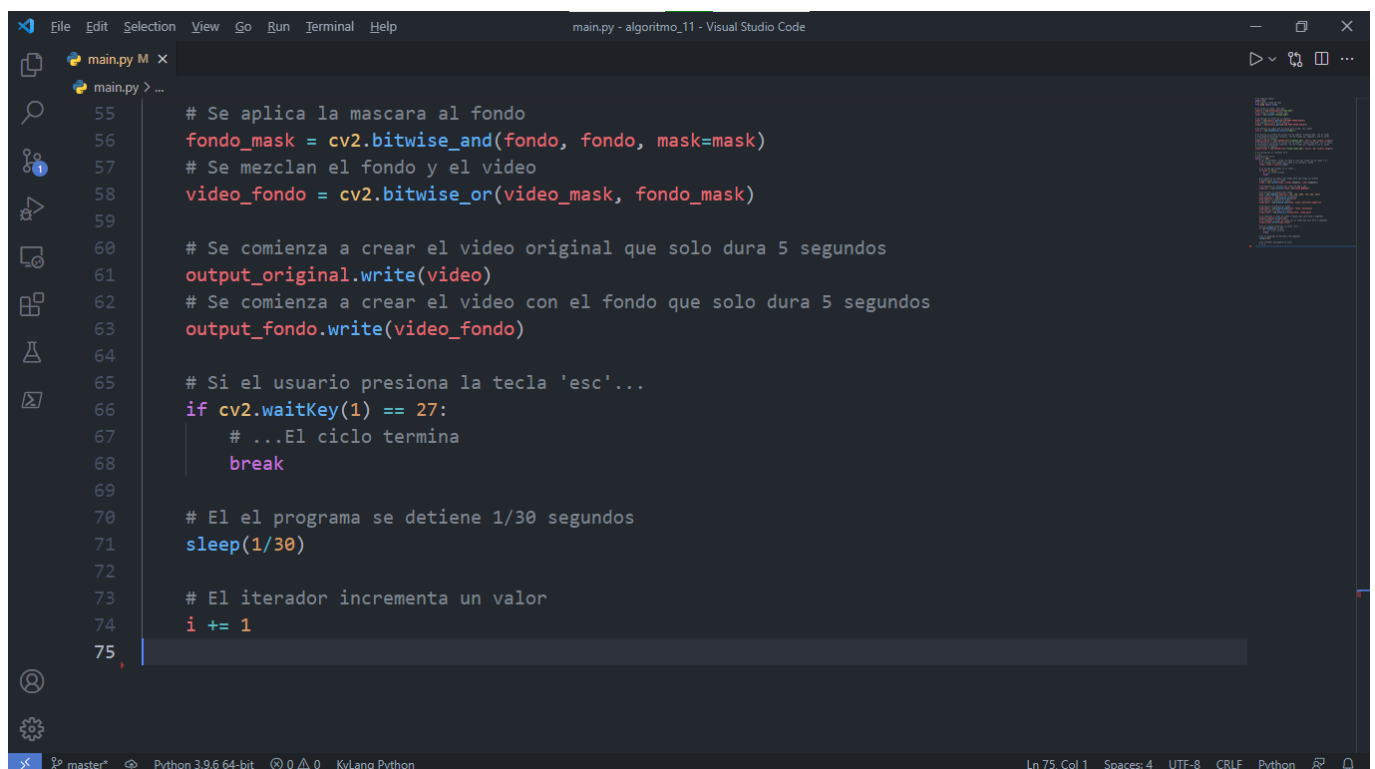
```
24 # previamente declarado (fourcc), con 30 frames por segundo y con el ancho
25 # y alto de la captura
26 output_fondo = cv2.VideoWriter("fondo_video.mp4", fourcc, 30, (width, height))
27
28 # Se inicializa el iterador en 0
29 i = 0
30 # Comienza el ciclo
31 while i < 150:
32     # Con la variable 'leido' se sabe si hay mas frames en el video o no.
33     # Se asignan los frames del video a la variable 'video'
34     leido, video = captura.read()
35
36     # Si no hay mas frames en el video...
37     if leido == False:
38         # ...El ciclo termina
39         break
40
41     # Se modifica el tamaño del fondo para que tenga las mismas
42     # dimensiones que el video.
43     fondo = cv2.resize(fondo, (video.shape[1], video.shape[0]))
44
45     # Se modifica el formato del video de BGR a HSV
46     video_hsv = cv2.cvtColor(video, cv2.COLOR_BGR2HSV)
47
```

Figura 2: Captura 2 del código del algoritmo 11



```
47
48 # Se calcula la mascara del video
49 mask = cv2.inRange(video_hsv, (45, 150, 150), (55, 255, 255))
50 # Se calcula el negativo de la mascara
51 mask_negativo = cv2.bitwise_not(mask)
52 # Se aplica la mascara al video
53 video_mask = cv2.bitwise_and(video, video, mask=mask_negativo)
54
55 # Se aplica la mascara al fondo
56 fondo_mask = cv2.bitwise_and(fondo, fondo, mask=mask)
57 # Se mezclan el fondo y el video
58 video_fondo = cv2.bitwise_or(video_mask, fondo_mask)
59
60 # Se comienza a crear el video original que solo dura 5 segundos
61 output_original.write(video)
62 # Se comienza a crear el video con el fondo que solo dura 5 segundos
63 output_fondo.write(video_fondo)
64
65 # Si el usuario presiona la tecla 'esc'...
66 if cv2.waitKey(1) == 27:
67     # ...El ciclo termina
68     break
69
70 # El el programa se detiene 1/30 segundos
```

Figura 3: Captura 3 del código del algoritmo 11



```
55 # Se aplica la mascara al fondo
56 fondo_mask = cv2.bitwise_and(fondo, fondo, mask=mask)
57 # Se mezclan el fondo y el video
58 video_fondo = cv2.bitwise_or(video_mask, fondo_mask)
59
60 # Se comienza a crear el video original que solo dura 5 segundos
61 output_original.write(video)
62 # Se comienza a crear el video con el fondo que solo dura 5 segundos
63 output_fondo.write(video_fondo)
64
65 # Si el usuario presiona la tecla 'esc'...
66 if cv2.waitKey(1) == 27:
67     # ...El ciclo termina
68     break
69
70 # El el programa se detiene 1/30 segundos
71 sleep(1/30)
72
73 # El iterador incrementa un valor
74 i += 1
75
```

Figura 4: Captura 4 del código del algoritmo 11