

Universidad de Guadalajara



Algoritmo 16

Muñoz Nuñez Ian Emmanuel

Visión Robótica

"Algoritmo Harris"

```
# Se importa opencv
import cv2
# Se importa numpy
import numpy as np

# Se declara la variable con la que se captura el video
captura = cv2.VideoCapture(0, cv2.CAP_DSHOW)

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

# Se genera el codec que se usara para escribir el video
fourcc = cv2.VideoWriter_fourcc(*'mp4v')

# Se define el archivo de salida del video original
output_original = cv2.VideoWriter('original.mp4', fourcc, 10, (width, height))
# Se define el archivo de salida del video
output = cv2.VideoWriter('resultado.mp4', fourcc, 10, (width, height))

# Se inicializa un iterador
i = 0
# Comienza el ciclo y termina hasta que haya completado 50 iteraciones
while i <= 50:
    # Con la variable 'leido' se sabe si hay mas frames o no. Se asignan los
    # frames capturados a la variable video
    leido, video = captura.read()

    # Si ya no hay mas frames en el video...
    if not leido:
        # ...Se termina el ciclo
        break

    # Si se presiona la tecla 'esc'...
    if cv2.waitKey(1) == 27:
        # ...El ciclo termina
        break

    # Se muestra el video original
    cv2.imshow('Original', video)
    # Se escribe el video original
    output_original.write(video)

    # Se cambia el formato del video a escala de grises
    grises = cv2.cvtColor(video, cv2.COLOR_BGR2GRAY)
    # Se cambia el tipo de dato de 'uint8' a 'float32'
    grises = np.float32(grises)
```

```
# Se aplica el algoritmo Harris al video en escala de grises
dst = cv2.cornerHarris(grises, 2, 3, 0.04)
dst = cv2.dilate(dst, None)
video[dst > 0.01 * dst.max()] = [0, 0, 255]

# Se muestra el video resultante
cv2.imshow('Resultado', video)
# Se escribe el video resultante
output.write(video)

# El iterador aumenta 1 unidad
i += 1

# Se cierra la captura del video
captura.release()

# Fin del programa
```

Para el código del algoritmo primero se importan las librerías *opencv* y *numpy*, luego se declara la captura del vídeo y se obtienen las dimensiones y el codec, después se declaran las variables con las que se escribirán los vídeos. Se comienza el ciclo para tomar cada uno de los frames del vídeo, este ciclo termina hasta completar 50 iteraciones, se muestra el video original, y después se cambia el formato de cada frame de BGR a escala de grises y se cambia el tipo de dato de 'uint8' a 'float32'. Luego se aplica el algoritmo Harris y se muestra el vídeo que se obtuvo como resultado. Por último se cierra la captura del vídeo.

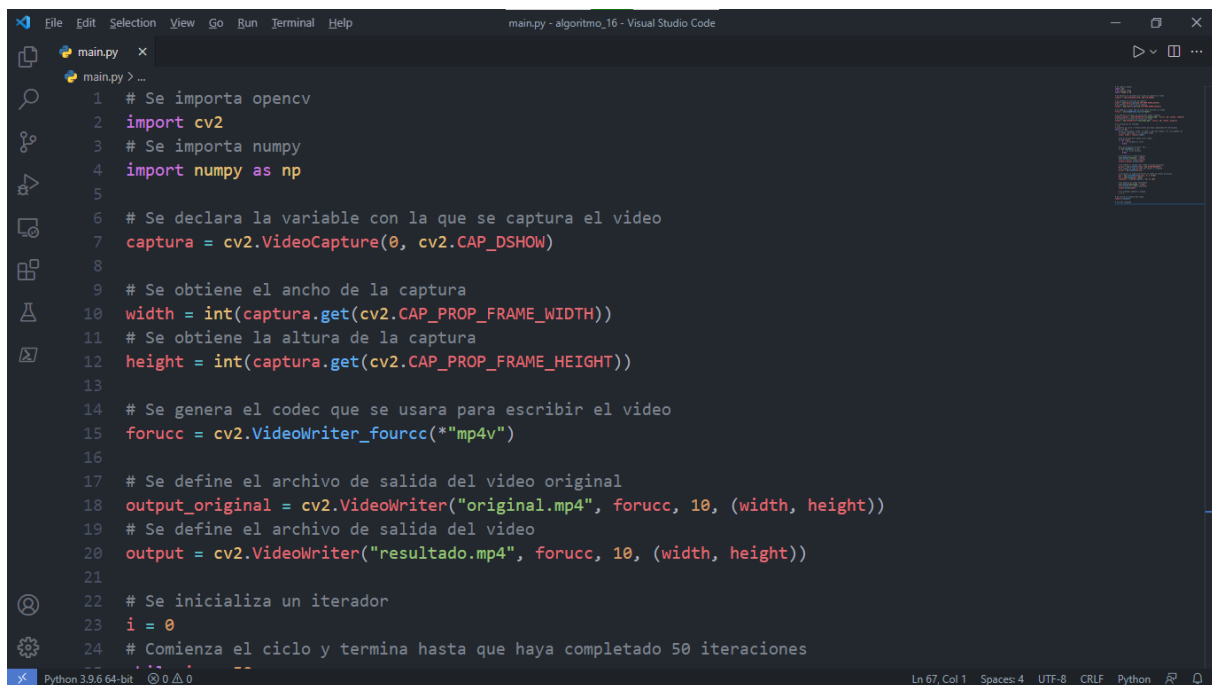
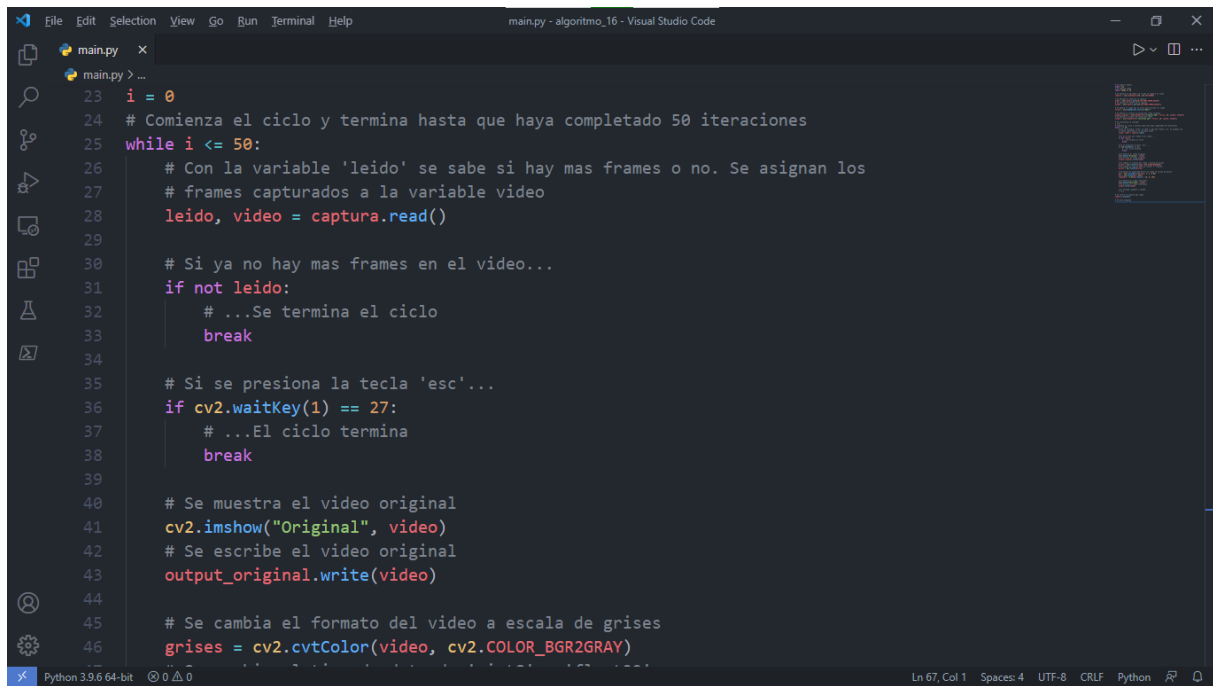
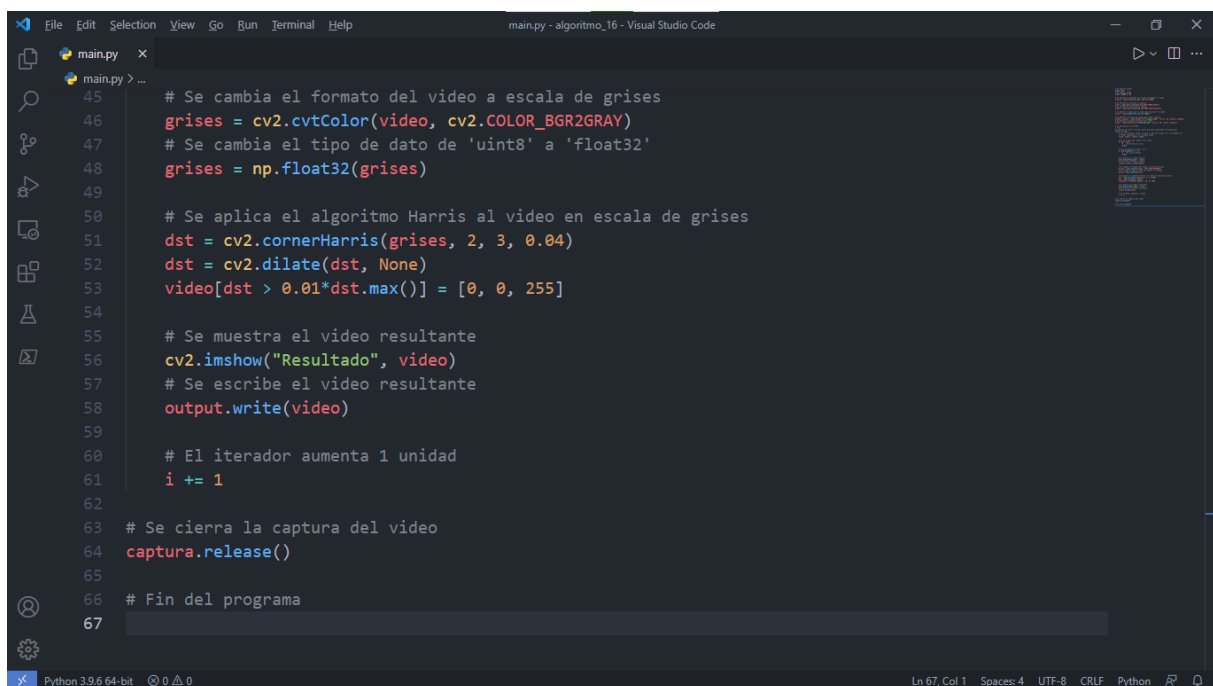


Figura 1: Captura 1 del código



```
23 i = 0
24 # Comienza el ciclo y termina hasta que haya completado 50 iteraciones
25 while i <= 50:
26     # Con la variable 'leido' se sabe si hay mas frames o no. Se asignan los
27     # frames capturados a la variable video
28     leido, video = captura.read()
29
30     # Si ya no hay mas frames en el video...
31     if not leido:
32         # ...Se termina el ciclo
33         break
34
35     # Si se presiona la tecla 'esc'...
36     if cv2.waitKey(1) == 27:
37         # ...El ciclo termina
38         break
39
40     # Se muestra el video original
41     cv2.imshow("Original", video)
42     # Se escribe el video original
43     output_original.write(video)
44
45     # Se cambia el formato del video a escala de grises
46     grises = cv2.cvtColor(video, cv2.COLOR_BGR2GRAY)
```

Figura 2: Captura 2 del código



```
45 # Se cambia el formato del video a escala de grises
46 grises = cv2.cvtColor(video, cv2.COLOR_BGR2GRAY)
47 # Se cambia el tipo de dato de 'uint8' a 'float32'
48 grises = np.float32(grises)
49
50 # Se aplica el algoritmo Harris al video en escala de grises
51 dst = cv2.cornerHarris(grises, 2, 3, 0.04)
52 dst = cv2.dilate(dst, None)
53 video[dst > 0.01*dst.max()] = [0, 0, 255]
54
55 # Se muestra el video resultante
56 cv2.imshow("Resultado", video)
57 # Se escribe el video resultante
58 output.write(video)
59
60 # El iterador aumenta 1 unidad
61 i += 1
62
63 # Se cierra la captura del video
64 captura.release()
65
66 # Fin del programa
67
```

Figura 3: Captura 3 del código

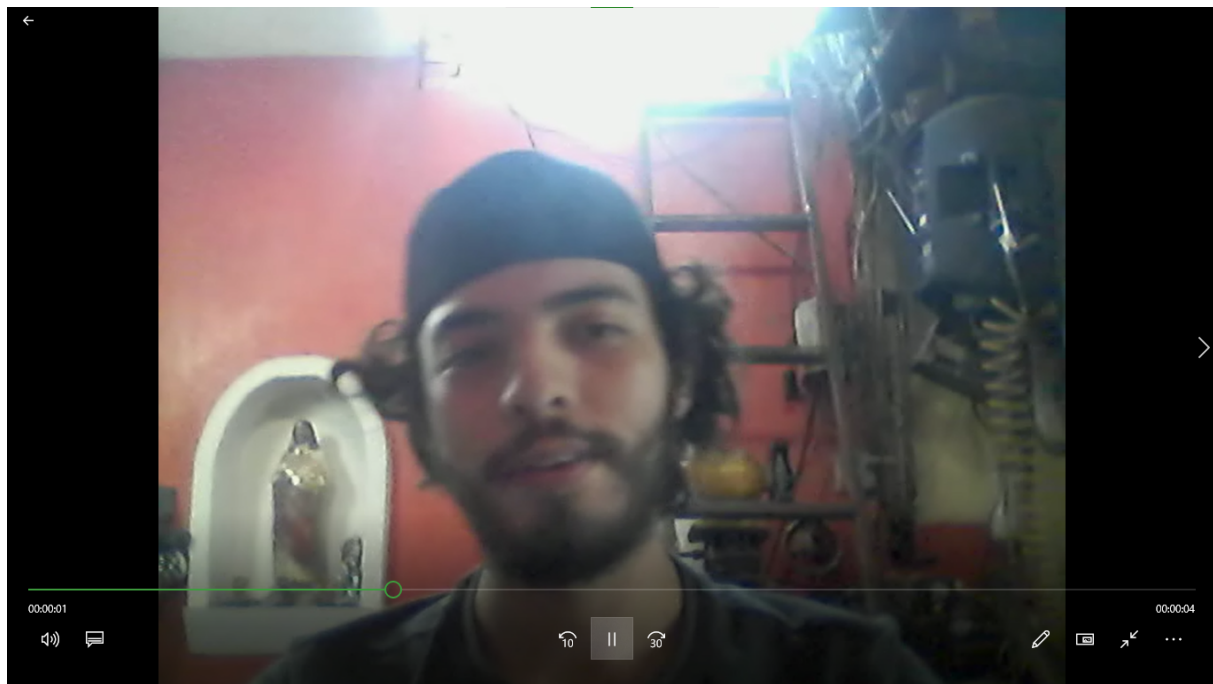


Figura 4: Captura del vídeo original

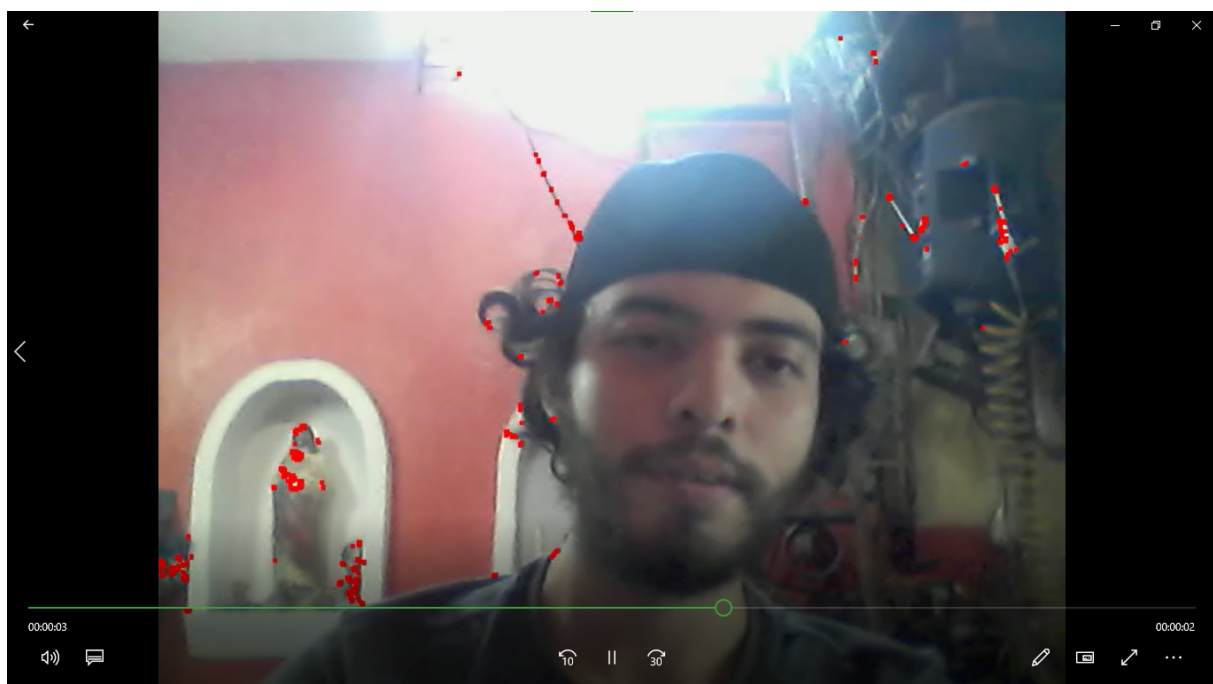


Figura 5: Captura del vídeo obtenido