

Universidad de Guadalajara



Proyecto integrador de la unidad 3

Muñoz Nuñez Ian Emmanuel

Visión Robótica

"Código"

```
# Se importa opencv
import cv2

# Se declara la funcion 'find_object'
def find_object(imagen, mask, color):
    # Se obtienen los contornos de la mascara que se recibe
    # como parametro en la funcion
    cnts, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Se itera entre todos los contornos obtenidos
    for c in cnts:
        # Se obtiene el area de cada uno de los contornos
        area = cv2.contourArea(c)

        # Si el area del contorno actual es mayor a 3000...
        if area > 3000:
            # ...Se dibujan los contornos mayores a un area de 3000
            cv2.drawContours(imagen, c, -1, color, 3)

            # Se obtienen las posiciones y medidas del cuadro delimitador del objeto
            x, y, w, h = cv2.boundingRect(c)
            # Se dibuja el rectangulo del objeto
            cv2.rectangle(imagen, (x, y), (x+w, y+h), color, 3)

# Se declara la captura de la camara
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 declara el codec del video
fourcc = cv2.VideoWriter_fourcc(*'mp4v')

# Se declara la variable de salida del video
output = cv2.VideoWriter('video.mp4', fourcc, 10, (width, height))

# Se declaran los valores minimos del rango amarillo
yellow_min = (25, 125, 70)
# Se declaran los valores maximos del rango amarillo
yellow_max = (35, 230, 225)

# Se inicializa el iterador
i = 0
# Comienza el ciclo
while i <= 50:
    # Con la variable 'leido' se sabe si hay mas frames o no.
```

```
# En la variable 'video' se guardan los frames que se leen
leido, video = captura.read()

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

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

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

# Se calcula la mascara de la imagen para el color amarillo
yellow_mask = cv2.inRange(video_hsv, yellow_min, yellow_max)

# Se llama a la funcion 'find_object' para dibujar los contornos
# del objeto y su cuadro delimitador
find_object(video, yellow_mask, (0, 255, 255))

# Se escriben las iniciales 'M. N. I. E.' en el video
cv2.putText(video, 'M. N. I. E.', (10, 100), cv2.FONT_HERSHEY_COMPLEX, 3, (255, 255, 0), 5)

# Se muestra el video
cv2.imshow('Video', video)

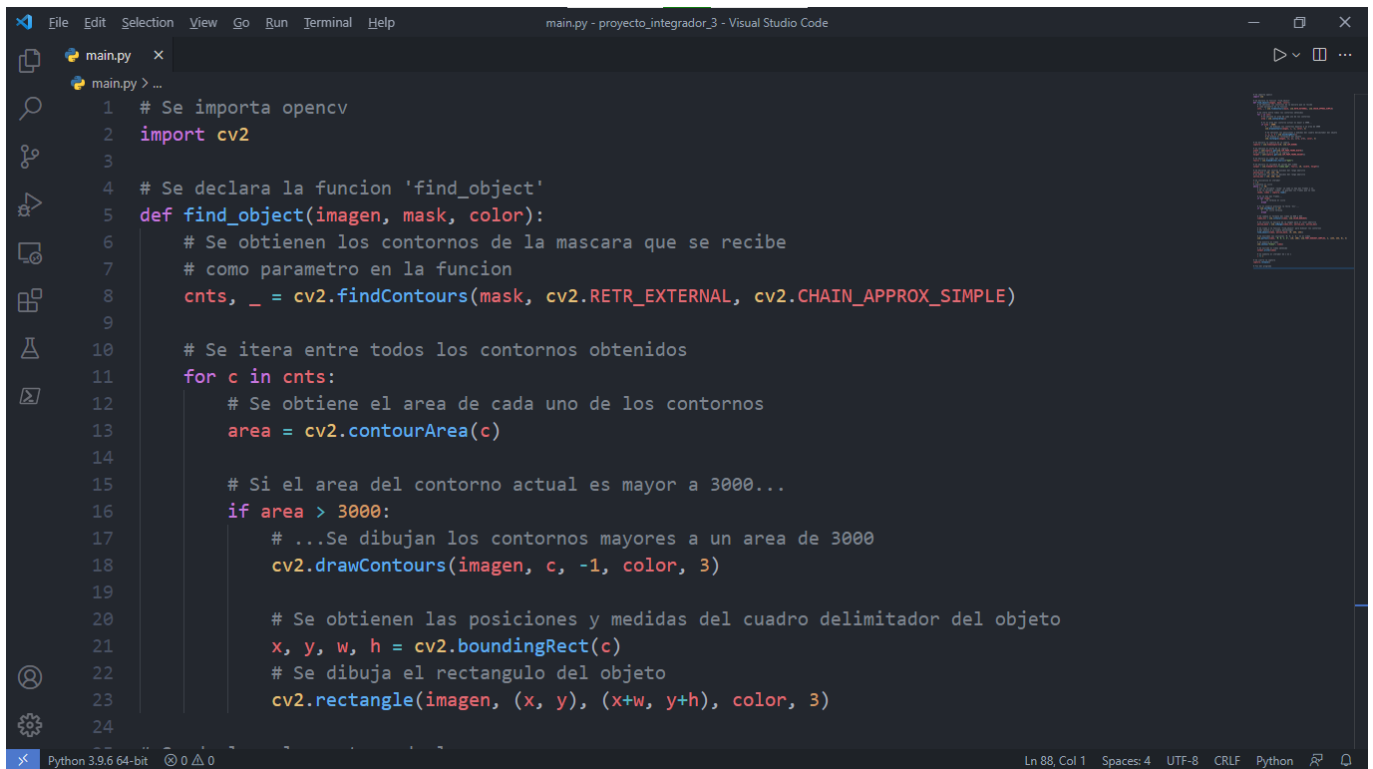
# Se escribe el video obtenido
output.write(video)

# Se aumenta el iterador de 1 en 1
i += 1

# Se cierra la captura
captura.release()

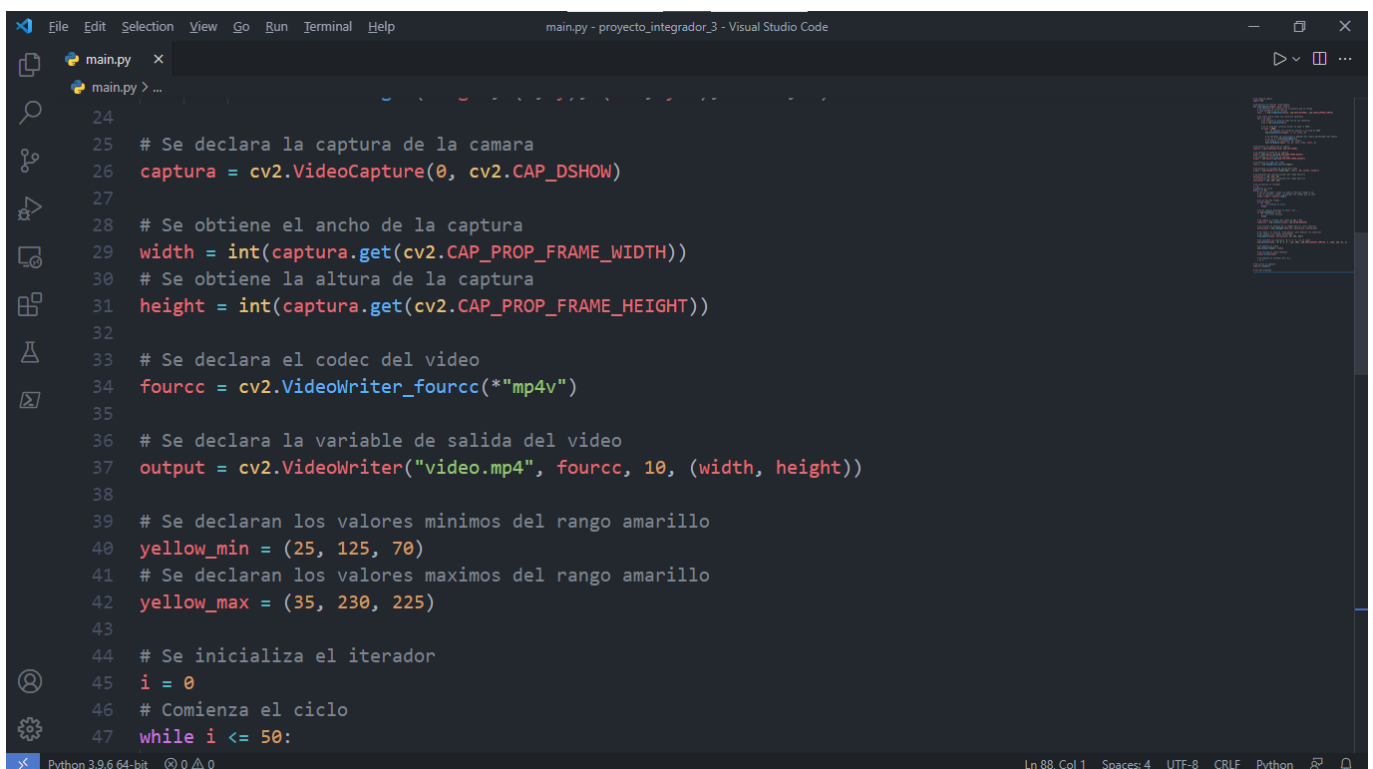
# Fin del programa
```

Para el código del proyecto integrador de la unidad 3, primero se importa *opencv*, luego se define la función *"find_object"*, esta obtiene los contornos de la máscara del objeto y dibuja estos y el cuadro delimitador del objeto. Después se declara la función para capturar el vídeo, también se declaran las variables para el ancho y el alto de la captura, se declara el codec con el que se escribirá el vídeo y por último, se declara la variable para la salida del vídeo. Luego se inicia un ciclo, que termina al hasta completar 50 iteraciones, se obtienen los frames que son capturados por la cámara y se cambia el formato del vídeo de BGR a HSV. Luego se obtiene la máscara para el objeto, se dibujan sus bordes y su cuadro delimitador con la función *"find_object"* y se dibujan las iniciales *"M. N. I. E."*. Por último se muestra y se escribe el vídeo. Al terminar el ciclo se cierra la captura y termina el programa.




```
1 # Se importa opencv
2 import cv2
3
4 # Se declara la funcion 'find_object'
5 def find_object(imagen, mask, color):
6     # Se obtienen los contornos de la mascara que se recibe
7     # como parametro en la funcion
8     cnts, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
9
10    # Se itera entre todos los contornos obtenidos
11    for c in cnts:
12        # Se obtiene el area de cada uno de los contornos
13        area = cv2.contourArea(c)
14
15        # Si el area del contorno actual es mayor a 3000...
16        if area > 3000:
17            # ...Se dibujan los contornos mayores a un area de 3000
18            cv2.drawContours(imagen, c, -1, color, 3)
19
20        # Se obtienen las posiciones y medidas del cuadro delimitador del objeto
21        x, y, w, h = cv2.boundingRect(c)
22        # Se dibuja el rectangulo del objeto
23        cv2.rectangle(imagen, (x, y), (x+w, y+h), color, 3)
24
```

Figura 1: Captura 1 del código del proyecto



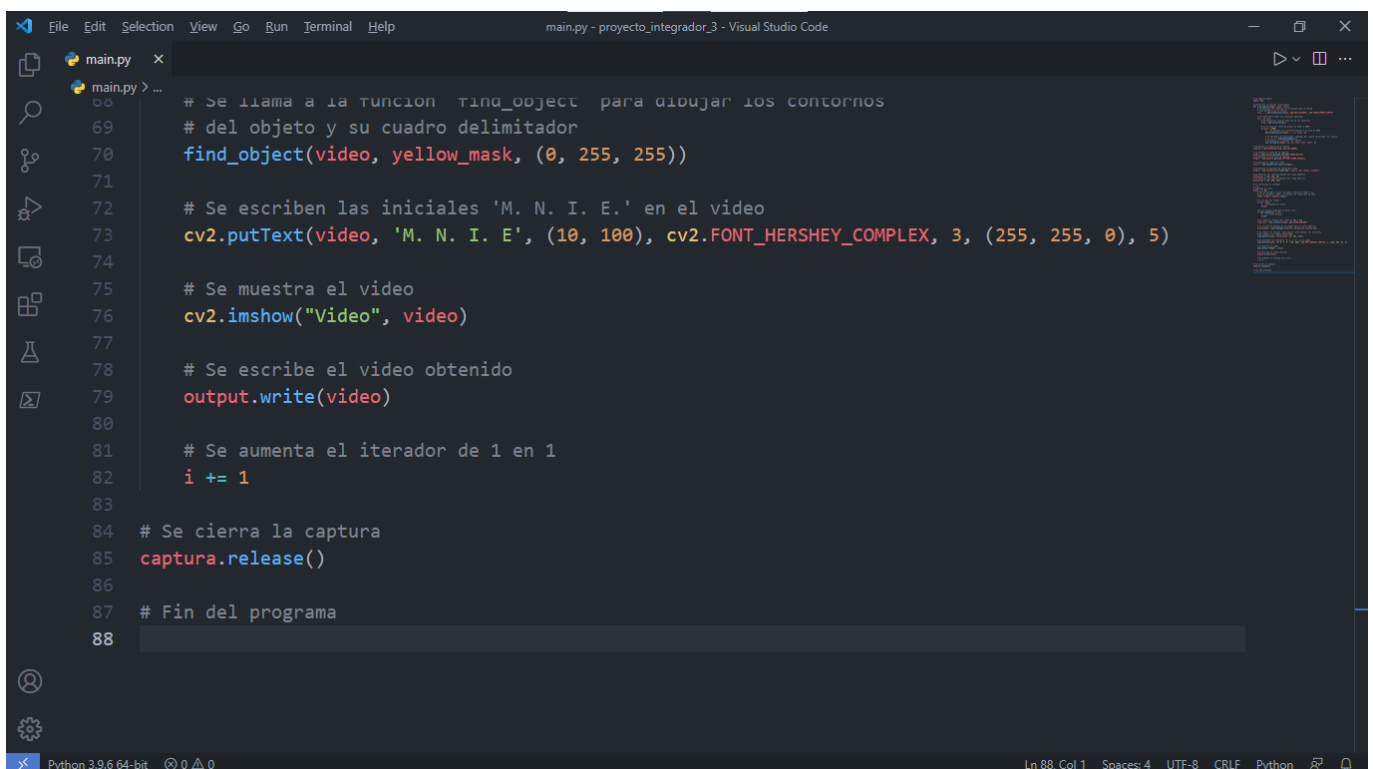
```
24
25 # Se declara la captura de la camara
26 captura = cv2.VideoCapture(0, cv2.CAP_DSHOW)
27
28 # Se obtiene el ancho de la captura
29 width = int(captura.get(cv2.CAP_PROP_FRAME_WIDTH))
30 # Se obtiene la altura de la captura
31 height = int(captura.get(cv2.CAP_PROP_FRAME_HEIGHT))
32
33 # Se declara el codec del video
34 fourcc = cv2.VideoWriter_fourcc(*"mp4v")
35
36 # Se declara la variable de salida del video
37 output = cv2.VideoWriter("video.mp4", fourcc, 10, (width, height))
38
39 # Se declaran los valores minimos del rango amarillo
40 yellow_min = (25, 125, 70)
41 # Se declaran los valores maximos del rango amarillo
42 yellow_max = (35, 230, 225)
43
44 # Se inicializa el iterador
45 i = 0
46 # Comienza el ciclo
47 while i <= 50:
```

Figura 2: Captura 2 del código del proyecto



```
47 while i <= 50:
48     # Con la variable 'leido' se sabe si hay mas frames o no.
49     # En la variable 'video' se guardan los frames que se leen
50     leido, video = captura.read()
51
52     # Si no hay mas frames...
53     if not leido:
54         # ...Se termina el ciclo
55         break
56
57     # Si el usuario presiona la tecla 'esc'...
58     if cv2.waitKey(1) == 27:
59         # ...El ciclo termina
60         break
61
62     # Se cambia el formato del video de BGR a HSV
63     video_hsv = cv2.cvtColor(video, cv2.COLOR_BGR2HSV)
64
65     # Se calcula la mascara de la imagen para el color amarillo
66     yellow_mask = cv2.inRange(video_hsv, yellow_min, yellow_max)
67
68     # Se llama a la funcion 'find_object' para dibujar los contornos
69     # del objeto y su cuadro delimitador
70     find_object(video, yellow_mask, (0, 255, 255))
```

Figura 3: Captura 3 del código del proyecto



```
69     # Se llama a la funcion find_object para dibujar los contornos
70     # del objeto y su cuadro delimitador
71     find_object(video, yellow_mask, (0, 255, 255))
72
73     # Se escriben las iniciales 'M. N. I. E.' en el video
74     cv2.putText(video, 'M. N. I. E.', (10, 100), cv2.FONT_HERSHEY_COMPLEX, 3, (255, 255, 0), 5)
75
76     # Se muestra el video
77     cv2.imshow("Video", video)
78
79     # Se escribe el video obtenido
80     output.write(video)
81
82     # Se aumenta el iterador de 1 en 1
83     i += 1
84
85     # Se cierra la captura
86     captura.release()
87
88     # Fin del programa
```

Figura 4: Captura 4 del código del proyecto

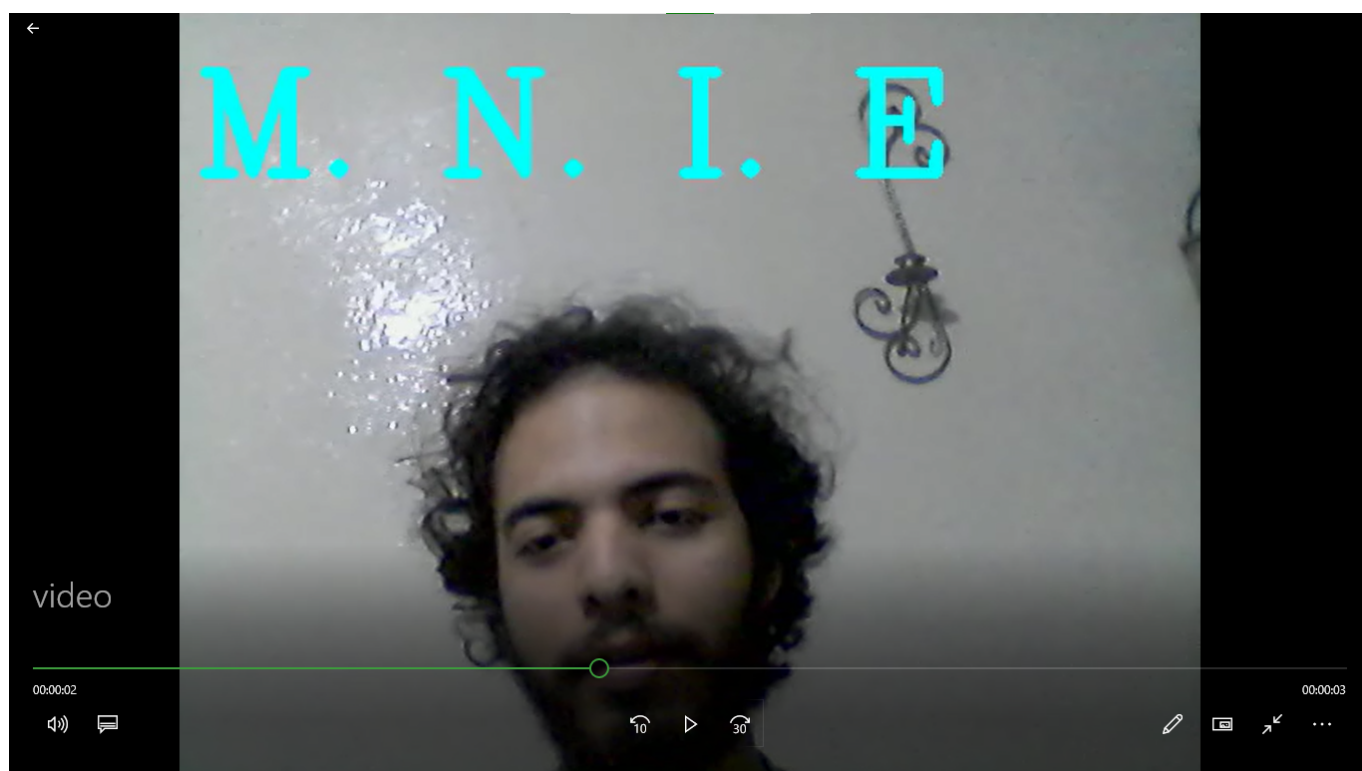


Figura 5: Captura del vídeo obtenido sin el objeto



Figura 6: Captura del vídeo obtenido mostrando el resultado