

Universidad de Guadalajara



Algoritmo 14

Muñoz Nuñez Ian Emmanuel

Visión Robótica

"Detección de contornos"

”Ejercicio 1”

```
# Se importa la libreria cv2
import cv2
# Se importa la libreria numpy como np
import numpy as np

# Se crea una imagen de 600x800 de 3 canales de tipo numpy.uint8
imagen = np.zeros((600, 800, 3), dtype=np.uint8)

# Se declara el color blanco
blanco = (255, 255, 255)
# Se declara el color morado
morado = (255, 0, 255)

# Se dibujan las figuras en la imagen original
cv2.rectangle(imagen, (50, 50), (100, 100), blanco, -1)
cv2.circle(imagen, (650, 450), 90, blanco, -1)
cv2.circle(imagen, (160, 220), 50, blanco, -1)
cv2.rectangle(imagen, (300, 40), (750, 250), blanco, -1)
cv2.circle(imagen, (250, 430), 150, blanco, -1)
cv2.rectangle(imagen, (440, 300), (540, 400), blanco, -1)

# Se cambia el formato de la imagen de BGR a escala de grises
grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)

# Se calcula el contorno de las imagenes con la imagen en escala de grises
cnts, _ = cv2.findContours(grises, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

# Se dibujan los contornos en la imagen original
cv2.drawContours(imagen, cnts, -1, morado, 2)

# Se muestra la imagen con los contornos de las figuras
cv2.imshow("Imagen", imagen)

# cv2 espera a que el usuario presione una tecla
cv2.waitKey()
```

Para el código del ejercicio 1 primero se importan las librerías necesarias (***opencv*** y ***numpy***). Luego se crea la imagen con 600 píxeles de alto por 800 de ancho y con tres canales. Después se generan las figuras y se convierte la imagen a escala de grises, se obtienen los contornos y estos se dibujan en la imagen original, por último se muestra la imagen con los contornos, al final ***opencv*** espera a que el usuario presione una tecla para terminar el programa.

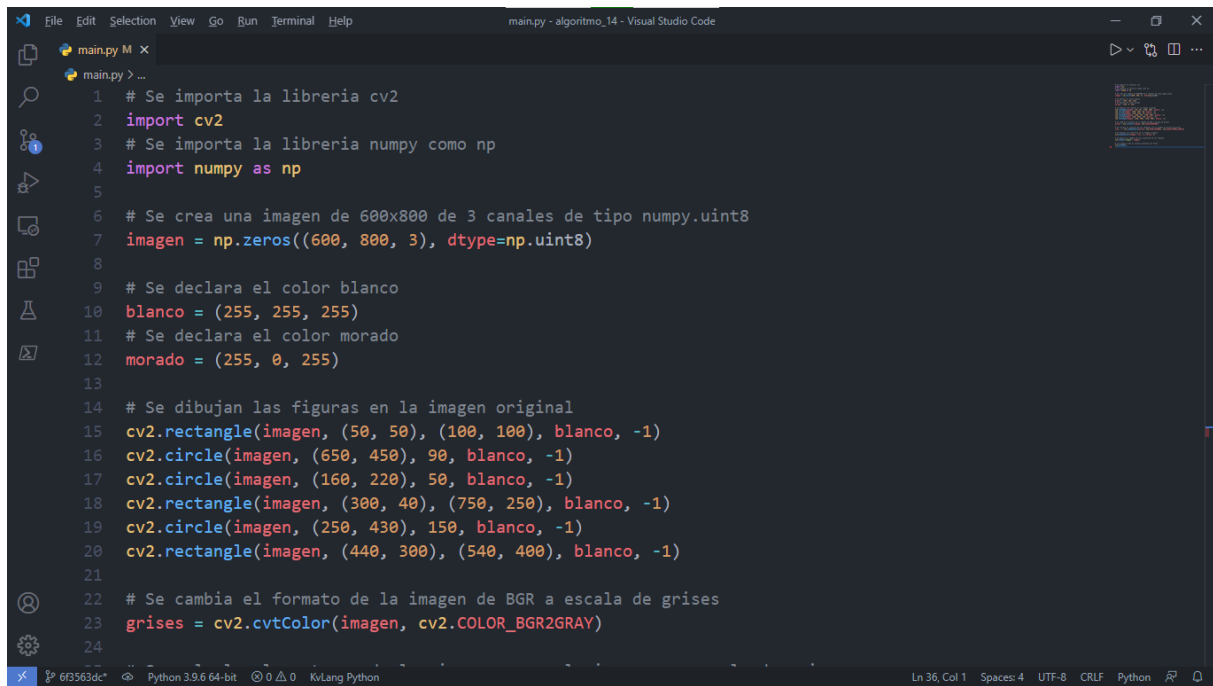


Figura 1: Captura 1 del código del ejercicio 1

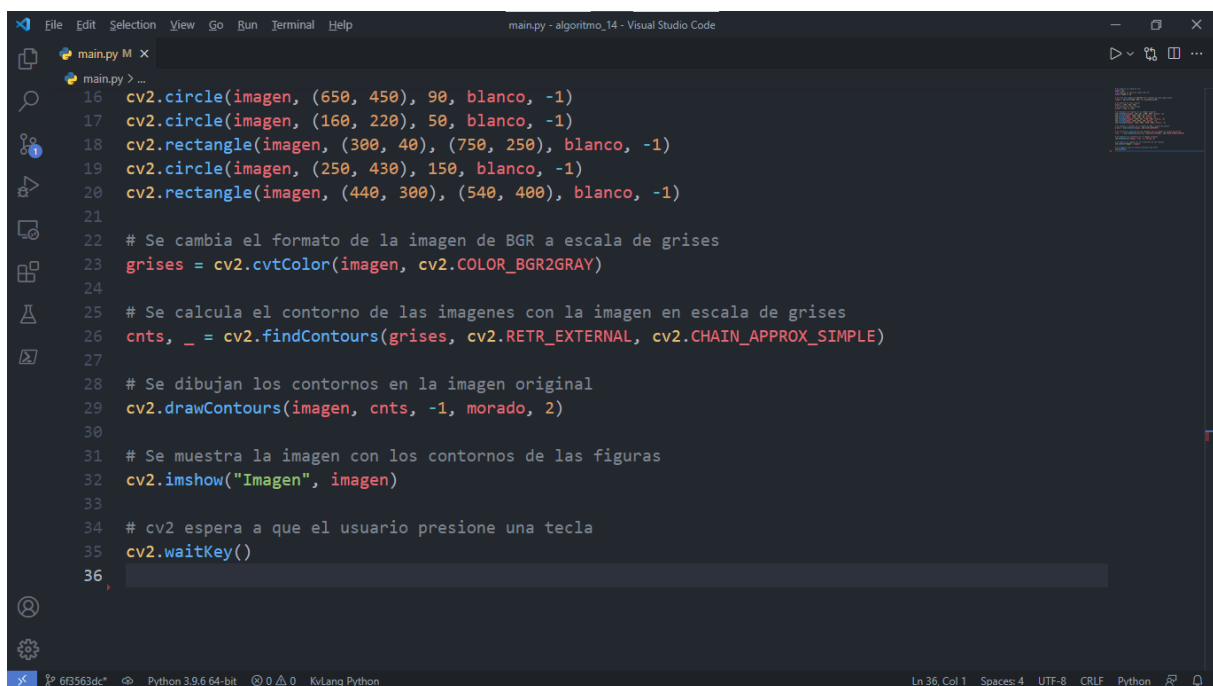


Figura 2: Captura 2 del código del ejercicio 2

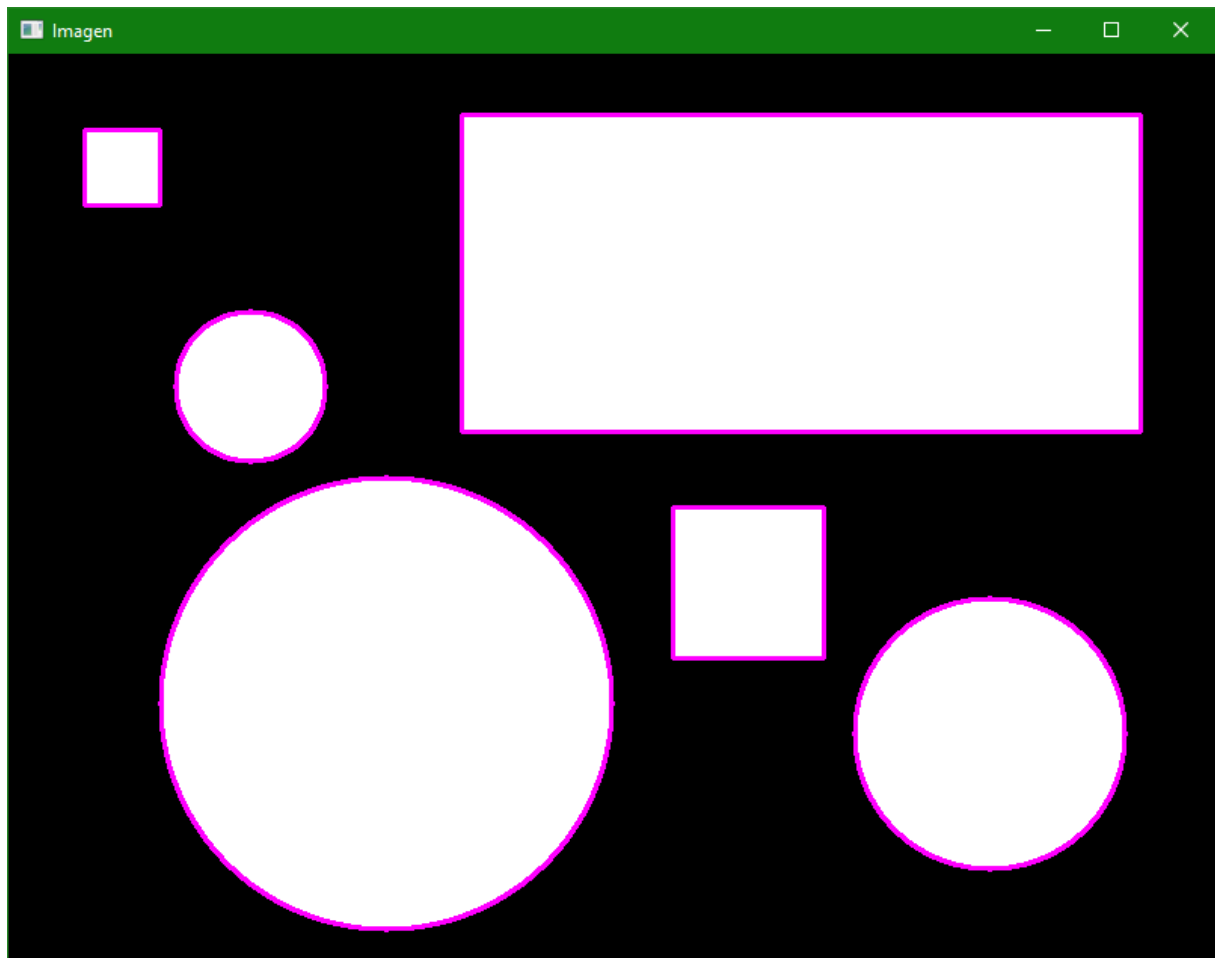


Figura 3: Imagen obtenida con el código anterior

"Ejercicio 2"

```
# Se importa la libreria cv2
import cv2
# Se importa la libreria numpy como np
import numpy as np

# Se crea una imagen de 600x800 de 3 canales de tipo numpy.uint8
imagen = np.zeros((600, 800, 3), dtype=np.uint8)

# Se declara el color blanco
blanco = (255, 255, 255)
# Se declara el color morado
morado = (255, 0, 255)

# Se dibujan las figuras en la imagen original
cv2.circle(imagen, (650, 450), 90, blanco, -1)
cv2.circle(imagen, (160, 150), 50, blanco, -1)
cv2.circle(imagen, (200, 430), 150, blanco, -1)
cv2.circle(imagen, (400, 100), 40, blanco, -1)
cv2.circle(imagen, (700, 200), 60, blanco, -1)
cv2.circle(imagen, (580, 80), 20, blanco, -1)
cv2.circle(imagen, (470, 250), 80, blanco, -1)

# Se cambia el formato de la imagen de BGR a escala de grises
grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)

# Se calcula el contorno de las imagenes con la imagen en escala de grises
cnts, _ = cv2.findContours(grises, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

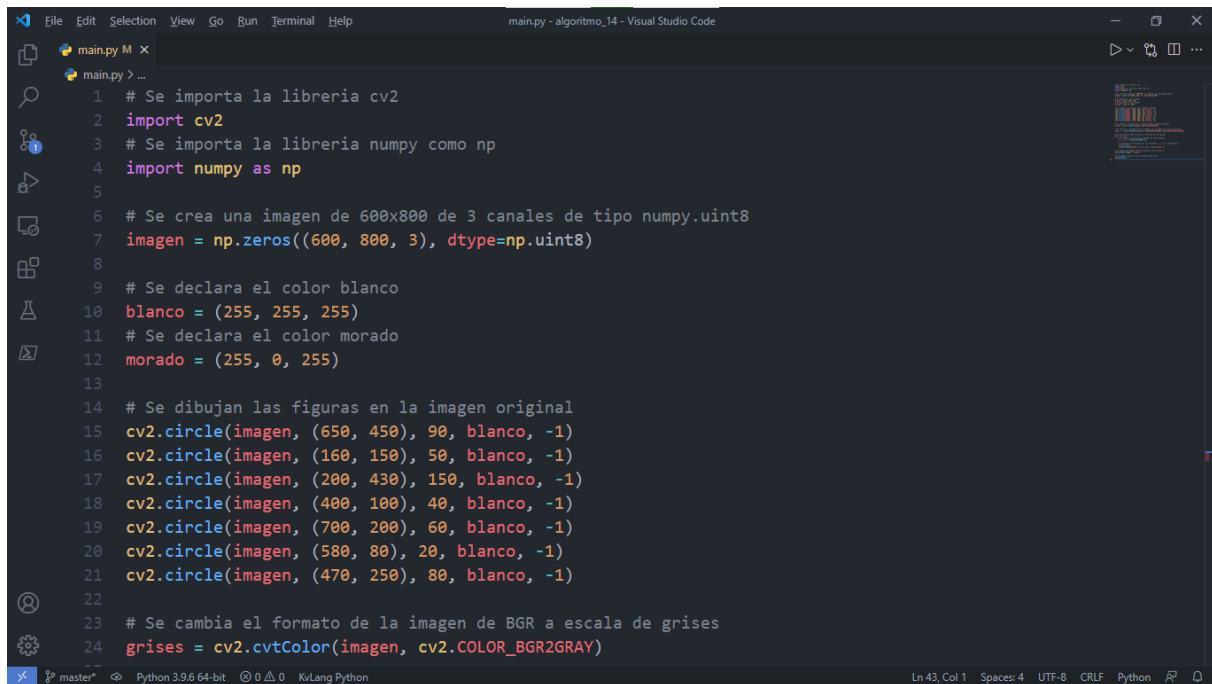
# Se itera entra cada uno de los contornos de las figuras
for c in cnts:
    # Se toman las posiciones y medidas de cada contorno
    x, y, w, h = cv2.boundingRect(c)

    # Se dibuja un rectangulo con las variables x, y, w y h que encierra
    # cada figura
    cv2.rectangle(imagen, (x, y), (x+w, y+h), morado, 2)

# Se muestra la imagen con los contornos de las figuras
cv2.imshow("Imagen", imagen)

# cv2 espera a que el usuario presione una tecla
cv2.waitKey()
```

Para el código del ejercicio 2 primero se importan las librerías necesarias (*opencv* y *numpy*). Luego se crea una imagen con 600 píxeles de alto por 800 de ancho y con tres canales. Después se generan los círculos y se convierte la imagen a escala de grises, se obtienen los contornos, se obtienen los cuadros delimitadores y estos se dibujan en la imagen original, por último se muestra la imagen con los delimitadores, al final *opencv* espera a que el usuario presione una tecla para terminar el programa.



```
1 # Se importa la libreria cv2
2 import cv2
3 # Se importa la libreria numpy como np
4 import numpy as np
5
6 # Se crea una imagen de 600x800 de 3 canales de tipo numpy.uint8
7 imagen = np.zeros((600, 800, 3), dtype=np.uint8)
8
9 # Se declara el color blanco
10 blanco = (255, 255, 255)
11 # Se declara el color morado
12 morado = (255, 0, 255)
13
14 # Se dibujan las figuras en la imagen original
15 cv2.circle(imagen, (650, 450), 90, blanco, -1)
16 cv2.circle(imagen, (160, 150), 50, blanco, -1)
17 cv2.circle(imagen, (200, 430), 150, blanco, -1)
18 cv2.circle(imagen, (400, 100), 40, blanco, -1)
19 cv2.circle(imagen, (700, 200), 60, blanco, -1)
20 cv2.circle(imagen, (580, 80), 20, blanco, -1)
21 cv2.circle(imagen, (470, 250), 80, blanco, -1)
22
23 # Se cambia el formato de la imagen de BGR a escala de grises
24 grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
```

Figura 4: Captura 1 del código del ejercicio 2

```
File Edit Selection View Go Run Terminal Help
main.py - algoritmo_14 - Visual Studio Code

main.py M x
main.py > ...
22 cv2.circle(imagen, (470, 230), 80, blanco, -1)
23
24 # Se cambia el formato de la imagen de BGR a escala de grises
25 grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
26
27 # Se calcula el contorno de las imagenes con la imagen en escala de grises
28 cnts, _ = cv2.findContours(grises, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
29
30 # Se itera entra cada uno de los contornos de las figuras
31 for c in cnts:
32     # Se toman las posiciones y medidas de cada contorno
33     x, y, w, h = cv2.boundingRect(c)
34
35     # Se dibuja un rectangulo con las variables x, y, w y h que encierra
36     # cada figura
37     cv2.rectangle(imagen, (x, y), (x+w, y+h), morado, 2)
38
39 # Se muestra la imagen con los contornos de las figuras
40 cv2.imshow("Imagen", imagen)
41
42 # cv2 espera a que el usuario presione una tecla
43 cv2.waitKey()
```

Figura 5: Captura 2 del código del ejercicio 2

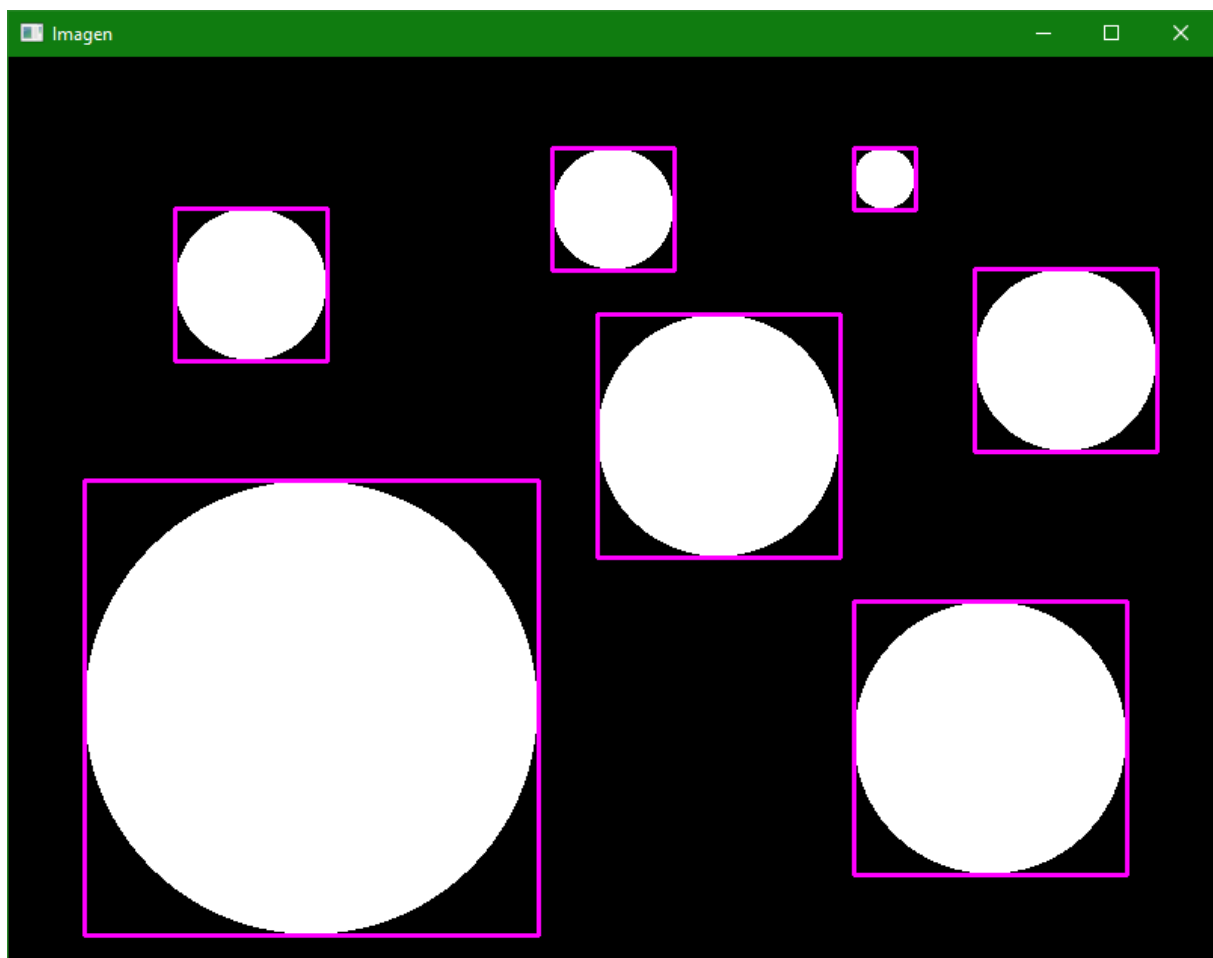


Figura 6: Imagen obtenida con el código anterior