

Universidad de Guadalajara



Algoritmo 19

Muñoz Nuñez Ian Emmanuel

Visión Robótica

*Transformada Hough*

### *Ejercicio 1*

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

# Se carga la imagen 'imagen.png'
imagen = cv2.imread('imagen.png')

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

# Se obtienen los bordes de la imagen
bordes = cv2.Canny(grises, 50, 100)
# Se obtienen las lineas de la imagen
lines = cv2.HoughLinesP(bordes, 1, np.pi/180, 100, minLineLength=10,
maxLineGap=250)

# Se un ciclo para dibujar en la imagen cada una de las lineas
for line in lines:
    # Se obtienen las coordenadas de las lineas
    x1, y1, x2, y2 = line[0]

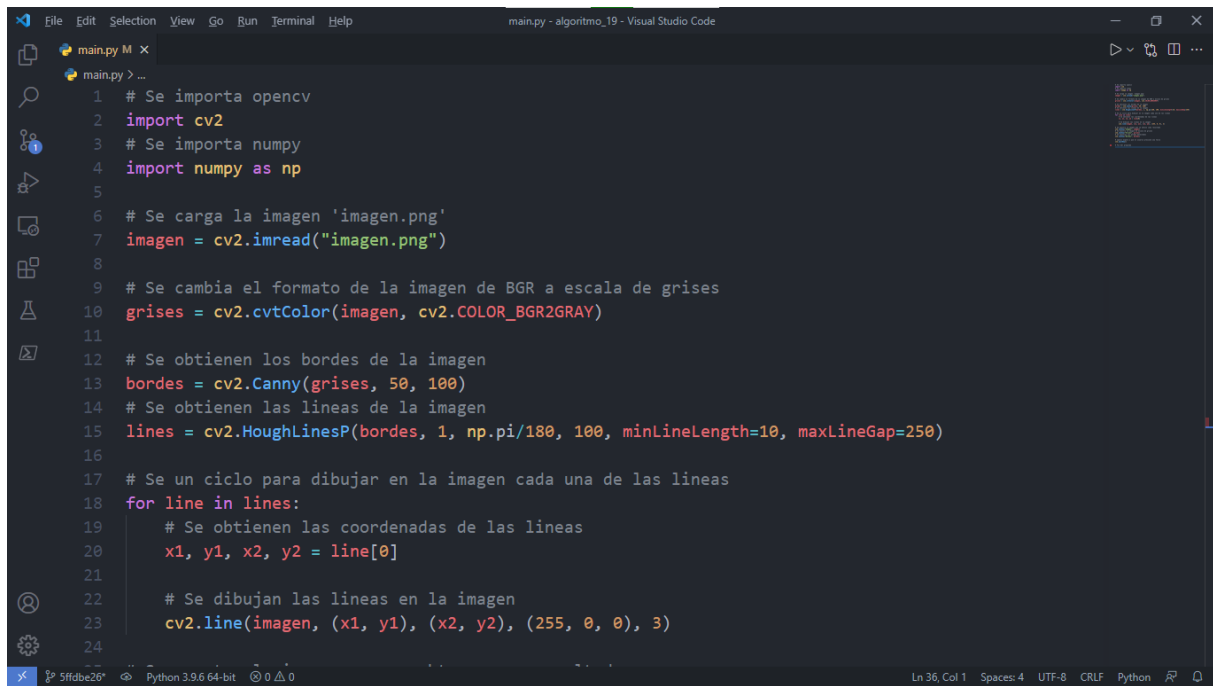
    # Se dibujan las lineas en la imagen
    cv2.line(imagen, (x1, y1), (x2, y2), (255, 0, 0), 3)

# Se muestra la imagen que se obtuvo como resultado
cv2.imshow('Imagen', imagen)
# Se muestra la imagen en escala de grises
cv2.imshow('Grises', grises)
# Se muestran los bordes detectados
cv2.imshow('Bordes', bordes)

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

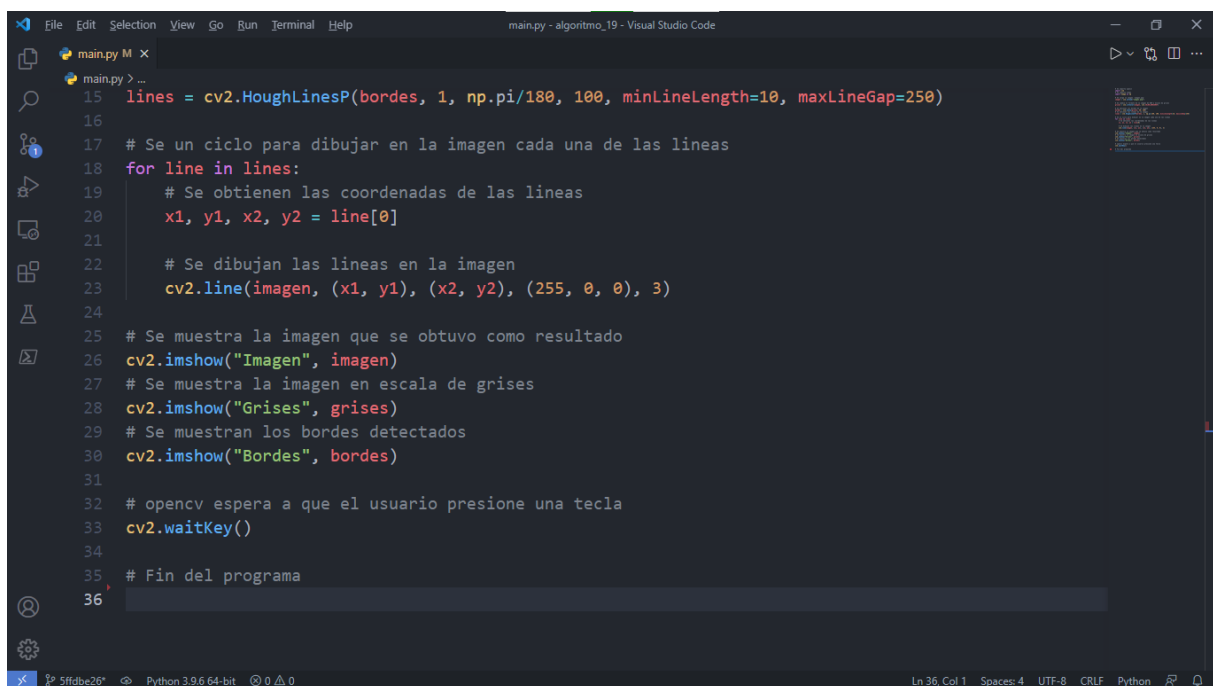
# Fin del programa
```

Para el código del ejercicio 1 primero se importan las librerías básicas (***opencv*** y ***numpy***), luego se carga la imagen con la que se trabajará, se cambia el formato de esta de BGR a escala de grises. Después se obtienen los bordes de la imagen y las líneas de esta con la transformada Hough. Luego se dibujan las líneas con un ciclo y por último se muestran la imagen obtenida, la imagen en escala de grises y los bordes obtenidos. Al final ***opencv*** espera a que el usuario presione una tecla para terminar el programa.



```
1 # Se importa opencv
2 import cv2
3 # Se importa numpy
4 import numpy as np
5
6 # Se carga la imagen 'imagen.png'
7 imagen = cv2.imread("imagen.png")
8
9 # Se cambia el formato de la imagen de BGR a escala de grises
10 grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
11
12 # Se obtienen los bordes de la imagen
13 bordes = cv2.Canny(grises, 50, 100)
14 # Se obtienen las líneas de la imagen
15 lines = cv2.HoughLinesP(bordes, 1, np.pi/180, 100, minLineLength=10, maxLineGap=250)
16
17 # Se un ciclo para dibujar en la imagen cada una de las líneas
18 for line in lines:
19     # Se obtienen las coordenadas de las líneas
20     x1, y1, x2, y2 = line[0]
21
22     # Se dibujan las líneas en la imagen
23     cv2.line(imagen, (x1, y1), (x2, y2), (255, 0, 0), 3)
24
```

Figura 1: Captura 1 del código del ejercicio 1



```
15 lines = cv2.HoughLinesP(bordes, 1, np.pi/180, 100, minLineLength=10, maxLineGap=250)
16
17 # Se un ciclo para dibujar en la imagen cada una de las líneas
18 for line in lines:
19     # Se obtienen las coordenadas de las líneas
20     x1, y1, x2, y2 = line[0]
21
22     # Se dibujan las líneas en la imagen
23     cv2.line(imagen, (x1, y1), (x2, y2), (255, 0, 0), 3)
24
25 # Se muestra la imagen que se obtuvo como resultado
26 cv2.imshow("Imagen", imagen)
27 # Se muestra la imagen en escala de grises
28 cv2.imshow("Grises", grises)
29 # Se muestran los bordes detectados
30 cv2.imshow("Bordes", bordes)
31
32 # opencv espera a que el usuario presione una tecla
33 cv2.waitKey()
34
35 # Fin del programa
36
```

Figura 2: Captura 2 del código del ejercicio 2

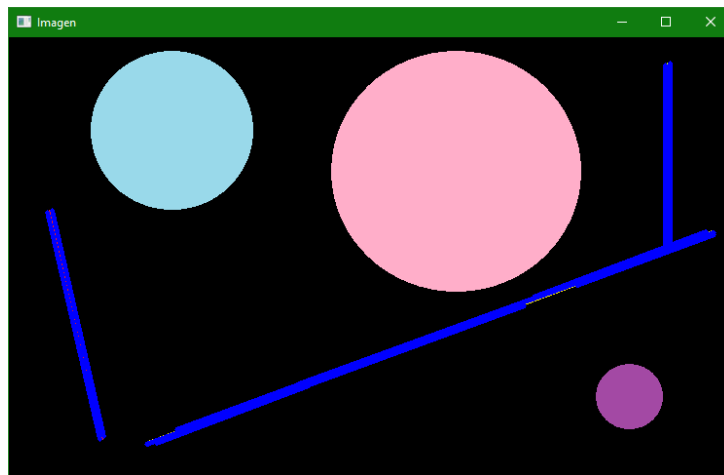


Figura 3: Imagen obtenida con el algoritmo

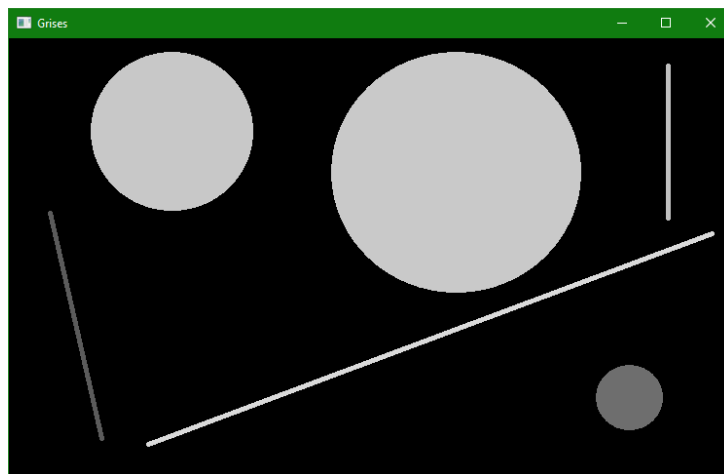


Figura 4: Imagen original en escala de grises

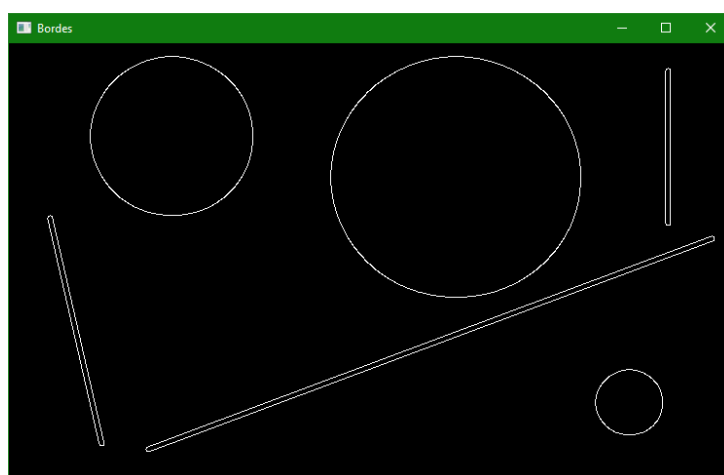


Figura 5: Bordes de la imagen

## Ejercicio 2

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

# Se carga la imagen 'imagen.png'
imagen = cv2.imread('imagen.png')

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

# Se obtienen los bordes de la imagen
_, binary = cv2.threshold(grises, 90, 255, cv2.THRESH_BINARY)
# Se obtienen los círculos de la imagen
circles = cv2.HoughCircles(binary, cv2.HOUGH_GRADIENT, 1, 20, param1=10,
param2=16, minRadius=1, maxRadius=130)

# Si se encontrarn círculos...
if circles is not None:
    # ...Se cambia el tipo de dato a uint16
    circles = np.uint16(np.around(circles))
    # Se inicia un ciclo para dibujar cada uno de los círculos
    for i in circles[0, :]:
        # Se dibuja un círculo que abarque todo el círculo
        cv2.circle(imagen, (i[0], i[1]), i[2], (0, 255, 255), 2)
        # Se dibuja un círculo pequeño en el centro del círculo
        cv2.circle(imagen, (i[0], i[1]), 2, (0, 0, 255), 3)

# Se muestra la imagen que se obtuvo como resultado
cv2.imshow('Imagen', imagen)
# Se muestra la imagen en escala de grises
cv2.imshow('Grises', grises)
# Se muestra la imagen binaria
cv2.imshow('Binaria', binary)

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

# Fin del programa
```

Para el código del ejercicio 2 primero se importan las librerías básicas (*opencv* y *numpy*), luego se carga la imagen con la que se trabajará, se cambia el formato de esta de BGR a escala de grises. Después se obtiene una imagen binaria y los círculos de la imagen con la transformada Hough. Luego se dibujan los círculos con un ciclo y por último se muestran la imagen obtenida, la imagen en escala de grises y la imagen binaria. Al final *opencv* espera a que el usuario presione una tecla para terminar el programa.

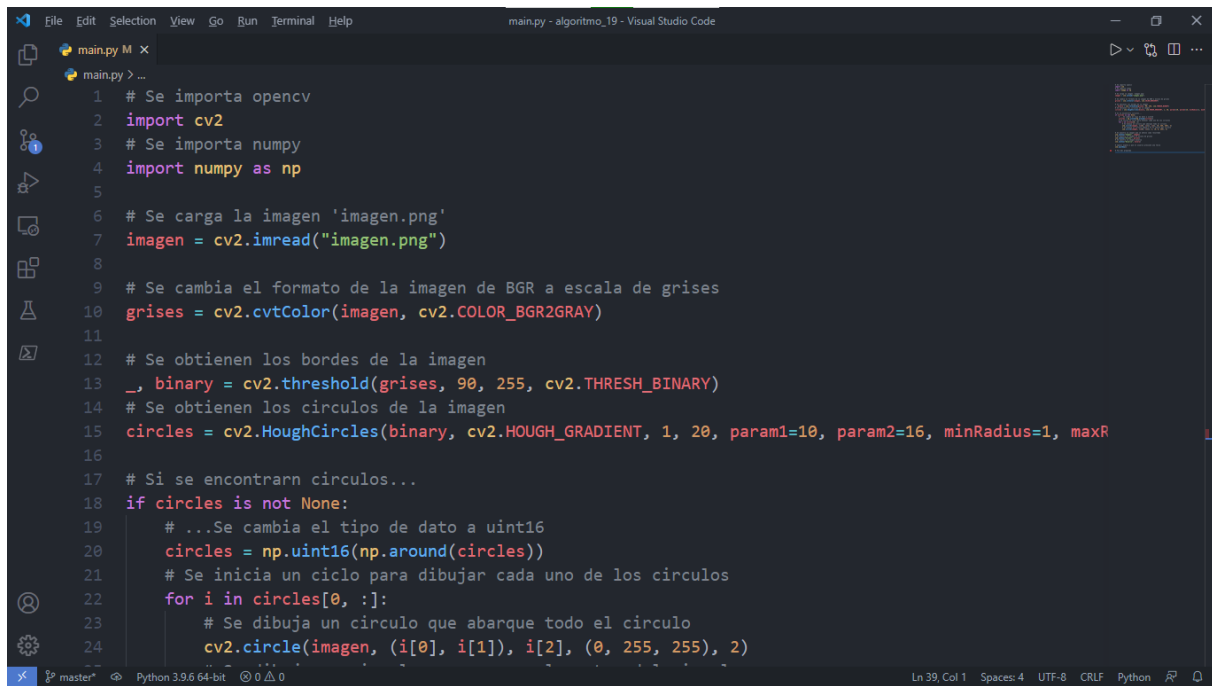


Figura 6: Captura 1 del código del ejercicio 2

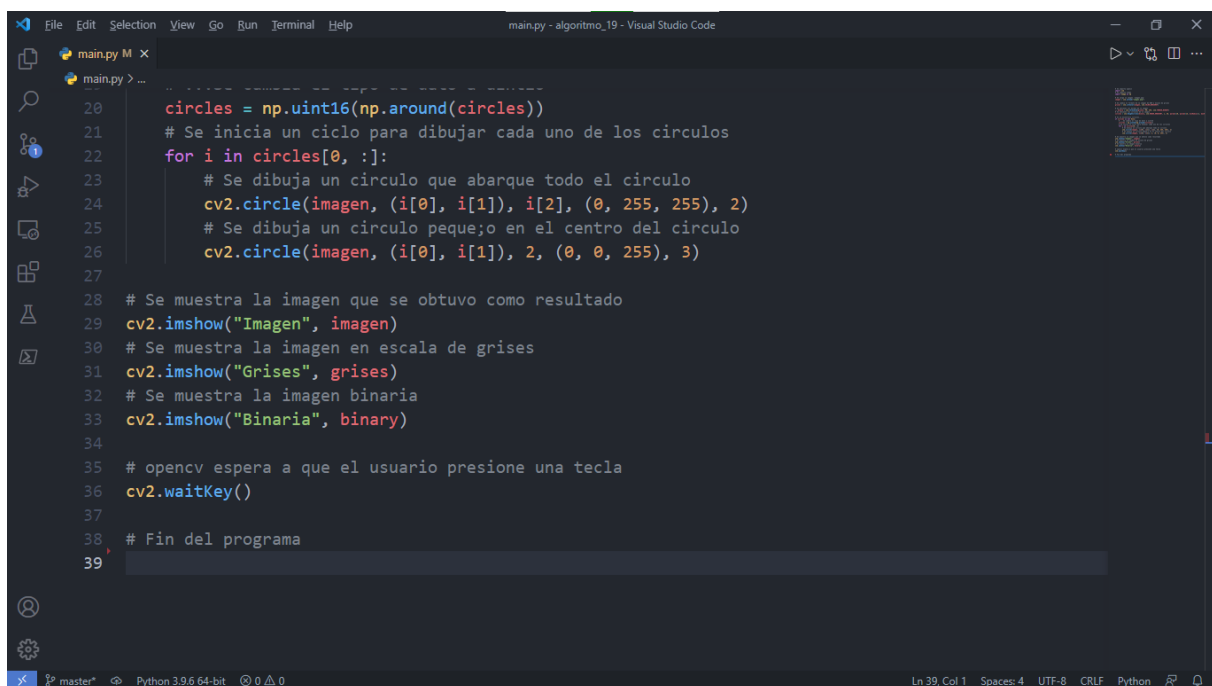


Figura 7: Captura 2 del código del ejercicio 2

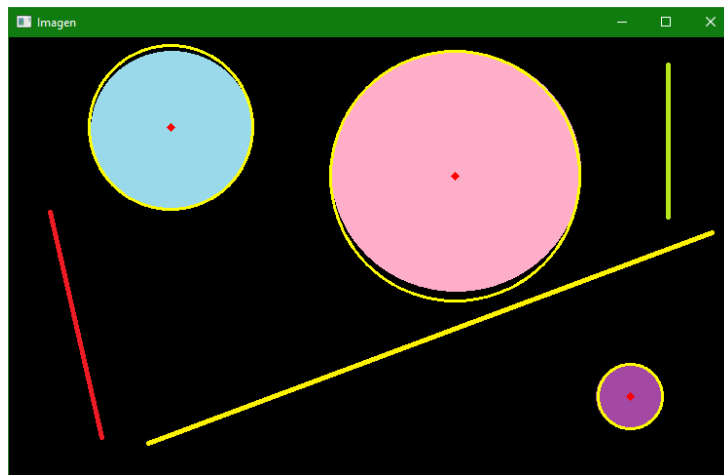


Figura 8: Imagen obtenida

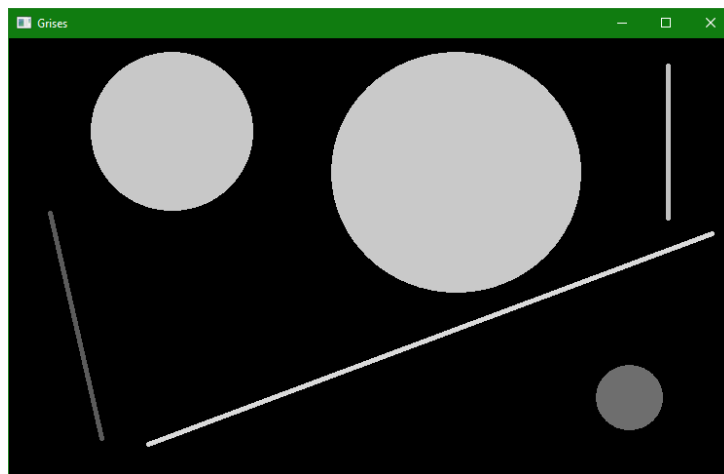


Figura 9: Imagen original en escala de grises

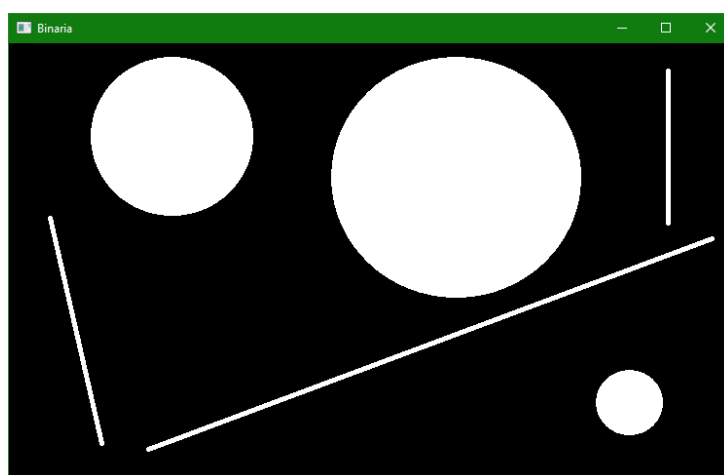


Figura 10: Imagen binaria