

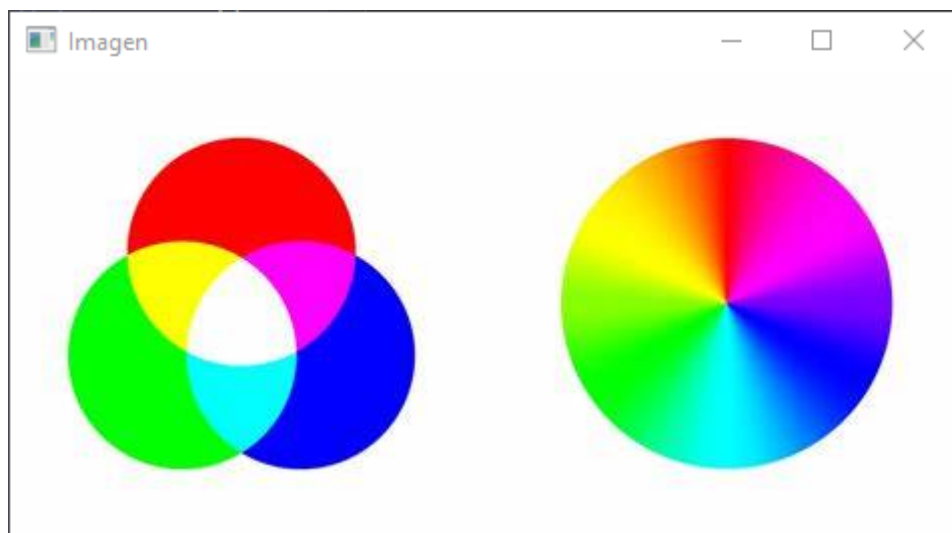
Algoritmo 2

Muñoz Nuñez Ian Emmanuel

Conversión a escala de grises

Visión Robótica

La imagen principal que se usara para el algoritmo es la siguiente.



Al final se usaran las últimas dos imagenes creadas en el algoritmo 1.

Filtros RGB

Filtro R

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np

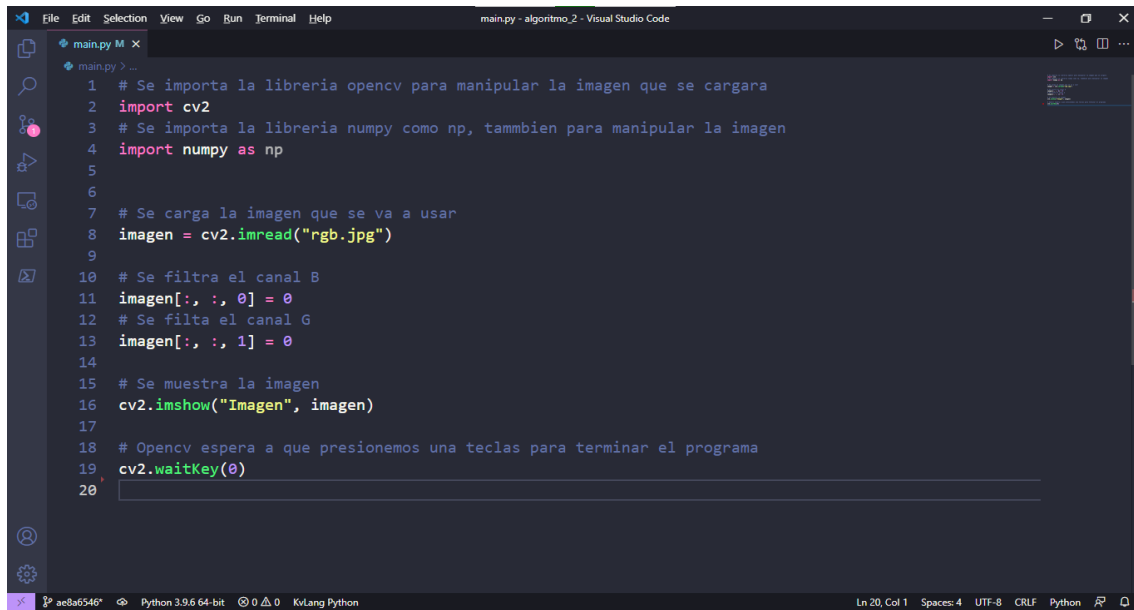
# Se carga la imagen que se va a usar
imagen = cv2.imread("rgb.jpg")

# Se filtra el canal B
imagen[:, :, 0] = 0
# Se filtra el canal G
imagen[:, :, 1] = 0

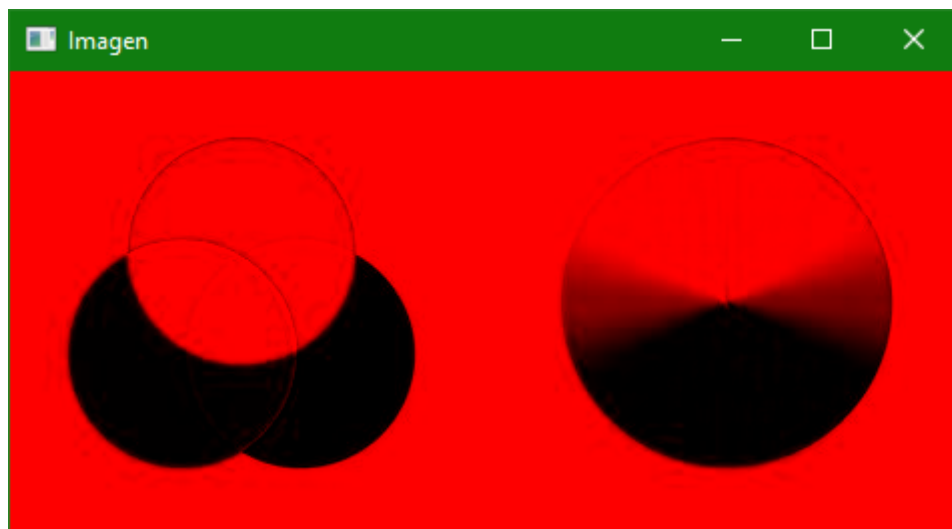
# Se muestra la imagen
cv2.imshow("Imagen", imagen)

# OpenCV espera a que presionemos una tecla para terminar el programa
cv2.waitKey(0)
```

En el código anterior primero se importan las librerías necesarias para realizarlo (opencv, numpy). Con ***opencv*** se lee una imagen que se encuentra en el mismo directorio del programa, se deja el valor de los canales B y G con 0, y así filtrar el canal R, de esta forma la sola tendrá el canal R con el valor de 255. Después se muestra la imagen, al final ***opencv*** espera a que el usuario presione una tecla para terminar el programa.

Se muestra una imagen del código

```
1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5
6
7 # Se carga la imagen que se va a usar
8 imagen = cv2.imread("rgb.jpg")
9
10 # Se filtra el canal B
11 imagen[:, :, 0] = 0
12 # Se filtra el canal G
13 imagen[:, :, 1] = 0
14
15 # Se muestra la imagen
16 cv2.imshow("Imagen", imagen)
17
18 # Opencv espera a que presionemos una teclas para terminar el programa
19 cv2.waitKey(0)
20
```

Se muestra una imagen de lo que realiza el código

Filtro G

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np

# Se carga la imagen que se va a usar
imagen = cv2.imread("rgb.jpg")

# Se filtra el canal B
imagen[:, :, 0] = 0
# Se filtra el canal R
imagen[:, :, 2] = 0

# Se muestra la imagen
cv2.imshow("Imagen", imagen)

# Opcv espera a que presionemos una tecla para terminar el programa
cv2.waitKey(0)
```

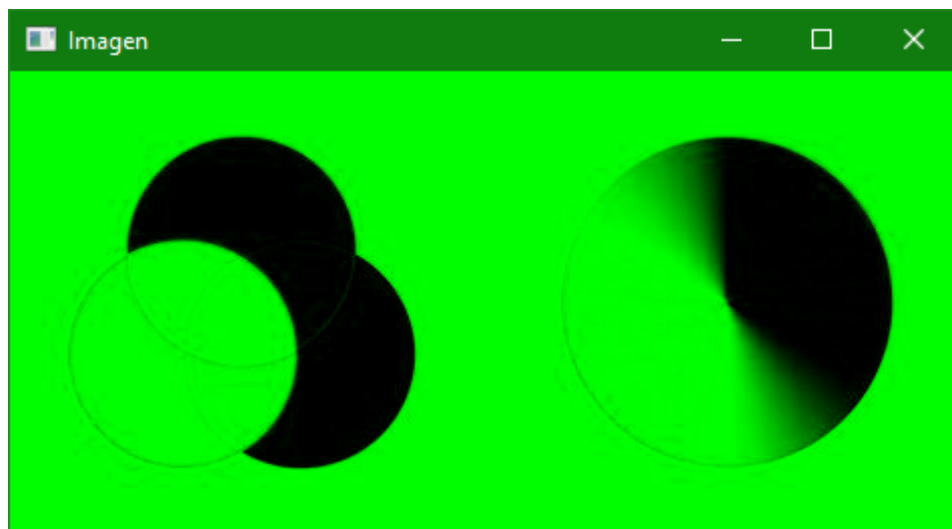
En el código anterior primero se importan las librerías necesarias para realizarlo (opencv, numpy). Con **opencv** se lee la imagen que se encuentra en el mismo directorio del programa, se deja el valor de los canales B y R en 0, y así filtrar el canal G, de esta forma la imagen solo tendrá el canal G con un valor de 255. Después se muestra la imagen con **opencv**, al final **opencv** espera a que el usuario presione alguna tecla para terminar el programa.

Se muestra una imagen del código

```
File Edit Selection View Go Run Terminal Help
main.py - algoritmo_2 - Visual Studio Code

main.py M x
main.py > ...
1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5
6
7 # Se carga la imagen que se va a usar
8 imagen = cv2.imread("rgb.jpg")
9
10 # Se filtra el canal B
11 imagen[:, :, 0] = 0
12 # Se filtra el canal R
13 imagen[:, :, 2] = 0
14
15 # Se muestra la imagen
16 cv2.imshow("Imagen", imagen)
17
18 # Opencv espera a que presionemos una teclas para terminar el programa
19 cv2.waitKey(0)
20
```

Se muestra una imagen de lo que el código realiza



Filtro B

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np

# Se carga la imagen que se va a usar
imagen = cv2.imread("rgb.jpg")

# Se filtra el canal G
imagen[:, :, 1] = 0
# Se filtra el canal R
imagen[:, :, 2] = 0

# Se muestra la imagen
cv2.imshow("Imagen", imagen)

# Opcv espera a que presionemos una tecla para terminar el programa
cv2.waitKey(0)
```

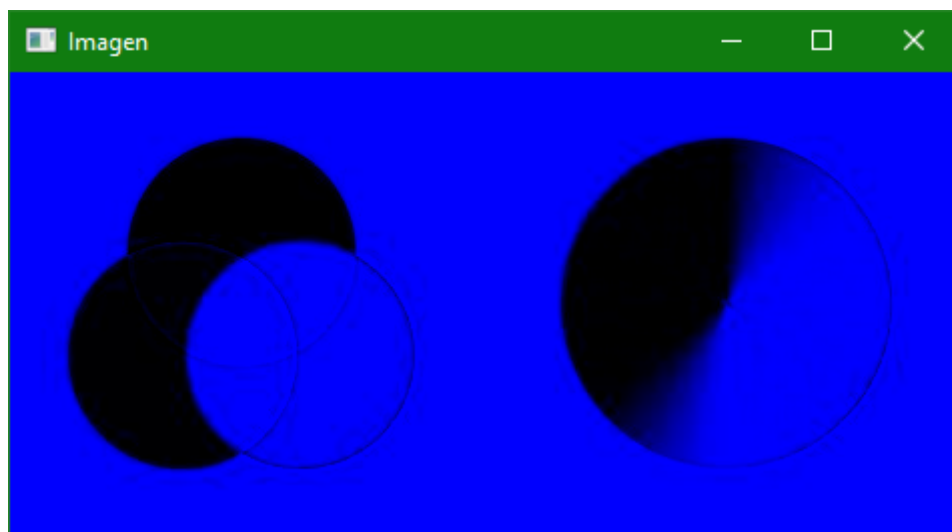
Para el código anterior primero se importan las librerías necesarias para realizarlo (opencv, numpy). Con **opencv** se lee la imagen que se encuentra en el mismo directorio del programa, se deja el valor de los canales G y R en 0, y así filtrar en canal B, de esta forma la imagen solo tendrá el canal B con un valor de 255. Después se muestra la imagen con **opencv**, al final **opencv** espera a que el usuario presione una tecla para terminar el programa.

Se muestra una imagen del código

```
File Edit Selection View Go Run Terminal Help
main.py - algoritmo_2 - Visual Studio Code

main.py M x
main.py > ...
1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5
6
7 # Se carga la imagen que se va a usar
8 imagen = cv2.imread("rgb.jpg")
9
10 # Se filtra el canal G
11 imagen[:, :, 1] = 0
12 # Se filtra el canal R
13 imagen[:, :, 2] = 0
14
15 # Se muestra la imagen
16 cv2.imshow("Imagen", imagen)
17
18 # Opencv espera a que presionemos una teclas para terminar el programa
19 cv2.waitKey(0)
20
```

Se muestra una imagen de lo que realiza el código



Método RGB

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np

# Se carga la imagen que se va a usar
imagen = cv2.imread("rgb.jpg")

# Se genera una matriz para crear una escala de grises desde el canal R
R = imagen[:, :, 2]
# Se genera una matriz para crear una escala de grises desde el canal G
G = imagen[:, :, 1]
# Se genera una matriz para crear una escala de grises desde el canal B
B = imagen[:, :, 0]

# Se concatenan las matrices de manera vertical para poder compararlas
RGB = np.vstack((R, G, B))

# Se muestra la imagen
cv2.imshow("Imagen", imagen)
# Se muestran las matrices concatenadas
cv2.imshow("RGB", RGB)

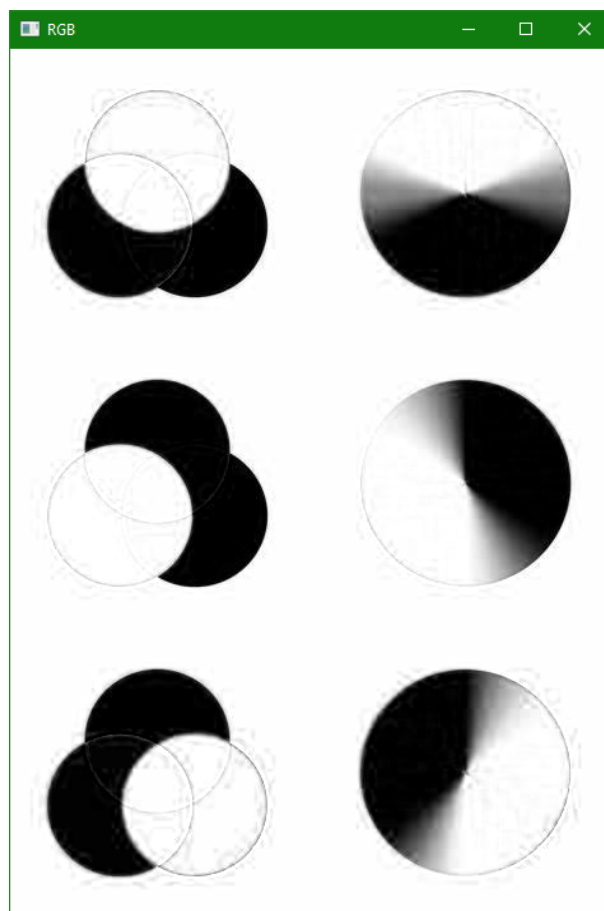
# Opencv espera a que presionemos una tecla para terminar el programa
cv2.waitKey(0)
```

Para el método RGB, primero se importan las librerías necesarias para realizarlo (opencv, numpy). Con ***opencv*** se lee la imagen que se encuentra en el mismo directorio del programa, luego se declaran las variables R, G y B, con las que se creara la escala de grises, al tenerlas se usa ***numpy*** para concatenarlas de manera vertical y así poder compararlas de una manera más sencilla. Después se muestra la imagen original junto con la imagen en sus distintas escalas de gris, al final ***opencv*** espera a que el usuario oprima una tecla para terminar el programa.

Se muestra una imagen del código

```
File Edit Selection View Go Run Terminal Help
main.py - algoritmo.2 - Visual Studio Code
main.py X
1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5
6
7 # Se carga la imagen que se va a usar
8 imagen = cv2.imread("rgb.jpg")
9
10 # Se genera una matriz para crear una escala de grises desde el canal R
11 R = imagen[:, :, 2]
12 # Se genera una matriz para crear una escala de grises desde el canal G
13 G = imagen[:, :, 1]
14 # Se genera una matriz para crear una escala de grises desde el canal B
15 B = imagen[:, :, 0]
16
17 # Se concatenan las matrices de manera vertical para poder compararlas
18 RGB = np.vstack((R, G, B))
19
20 # Se muestra la imagen
21 cv2.imshow("Imagen", imagen)
22 # Se muestran las matrices concatenadas
23 cv2.imshow("RGB", RGB)
24
25 # Opencv espera a que presionemos una teclas para terminar el programa
26 cv2.waitKey(0)
27
```

Se muestra una imagen de lo que realiza el código



Métodos promedio, BT.601 Y BT.709

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tammbien para manipular la imagen
import numpy as np
```

```
# Se carga la imagen que se va a usar
imagen = cv2.imread("rgb.jpg")
```

```
# Se genera una matriz para crear una escala de grises desde el canal R
R = imagen[:, :, 2]
# Se genera una matriz para crear una escala de grises desde el canal G
G = imagen[:, :, 1]
# Se genera una matriz para crear una escala de grises desde el canal B
B = imagen[:, :, 0]
```

```
# Se crea una escala de grises con el promedio de los canales
promedio = R*0.33 + G*0.33 + B*0.33
# Se convierte el promedio a un tipo de dato numpy.uint8
promedio = promedio.astype(np.uint8)
# Se crea una escala de grises con la ponderacion de los canales BT.601
bt_601 = R*0.299 + G*0.587 + B*0.114
# Se convierte la ponderacion a un tipo de dato numpy.uint8
bt_601 = bt_601.astype(np.uint8)
# Se crea una escala de grises con la ponderacion de los canales BT.709
bt_709 = R*0.2126 + G*0.7152 + B*0.0722
# Se convierte la ponderacion a un tipo de dato numpy.uint8
bt_709 = bt_709.astype(np.uint8)
```

```
# Se concatenan las matrices de manera vertical para poder compararlas
grises = np.vstack((promedio, bt_601, bt_709))
```

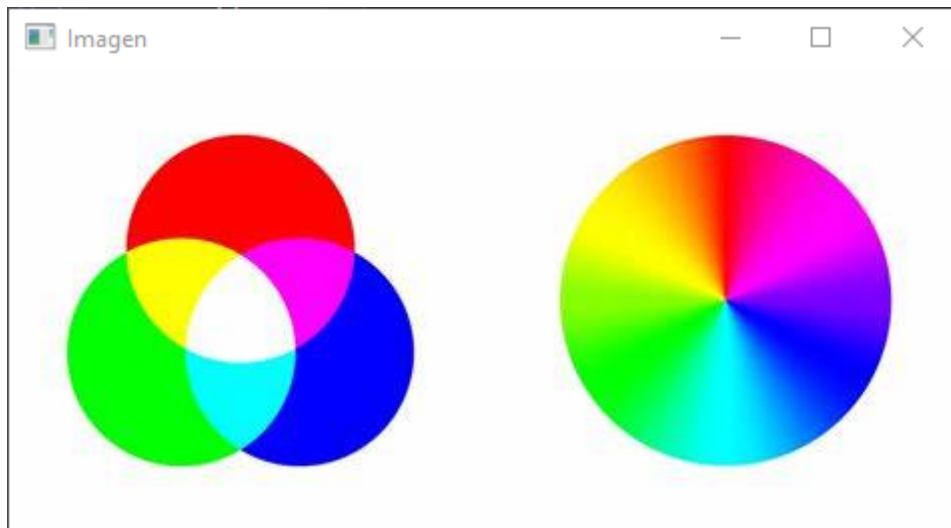
```
# Se muestra la imagen
cv2.imshow("Imagen", imagen)
# Se muestra la imagen con las matrices
```

```
cv2.imshow("Gris", grises)
```

```
# Opencv espera a que presionemos una teclas para terminar el programa  
cv2.waitKey(0)
```

Para el código anterior primero se importan las librerías necesarias para realizarlo (opencv, numpy). Con **opencv** se lee la imagen que se encuentra en el mismo directorio del programa, luego se declaran las variables R, G y B, con las que se crearan las escalas de gris, al tenerlas, estas variables se usan para declarar el método "promedio", el método "BT.601" y el método "BT.709", al haber declarado estos métodos se usa **numpy** para concatenarlos y así compararlos más fácil. Después se usa **opencv** para mostrar la imagen original y sus distintas escalas de gris, hasta arriba está el promedio de los canales, le sigue la ponderación BT.601 y hasta abajo está la ponderación BT.709, al final **opencv** espera a que el usuario presione una tecla para terminar el programa.

Se muestra la imagen original



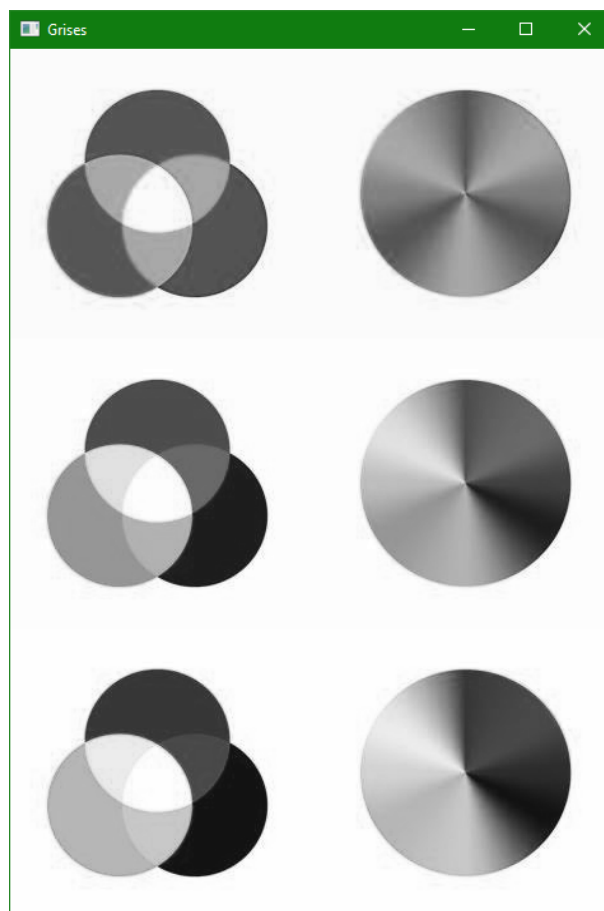
Se muestra el código del programa

```

1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5
6
7 # Se carga la imagen que se va a usar
8 imagen = cv2.imread("rgb.jpg")
9
10 # Se genera una matriz para crear una escala de grises desde el canal R
11 R = imagen[:, :, 2]
12 # Se genera una matriz para crear una escala de grises desde el canal G
13 G = imagen[:, :, 1]
14 # Se genera una matriz para crear una escala de grises desde el canal B
15 B = imagen[:, :, 0]
16
17 # Se crea una escala de grises con el promedio de los canales
18 promedio = R*0.33 + G*0.33 + B*0.33
19 # Se convierte el promedio a un tipo de dato numpy.uint8
20 promedio = promedio.astype(np.uint8)
21 # Se crea una escala de grises con la ponderacion de los canales BT.601
22 bt_601 = R*0.299 + G*0.587 + B*0.114
23 # Se convierte la ponderacion a un tipo de dato numpy.uint8
24 bt_601 = bt_601.astype(np.uint8)
25 # Se crea una escala de grises con la ponderacion de los canales BT.709
26 bt_709 = R*0.2126 + G*0.7152 + B*0.0722
27 # Se convierte la ponderacion a un tipo de dato numpy.uint8
28 bt_709 = bt_709.astype(np.uint8)
29
30 # Se concatenan las matrices de manera vertical para poder compararlas
31 grises = np.vstack((promedio, bt_601, bt_709))
32
33 # Se muestra la imagen
34 cv2.imshow("Imagen", imagen)
35 # Se muestra la imagen con las matrices
36 cv2.imshow("Grises", grises)
37
38 # OpenCV espera a que presionemos una tecla para terminar el programa
39 cv2.waitKey(0)
40

```

Se muestra una imagen de lo que el código realiza



Punto C (Algoritmo 1)

Método RGB (R)

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np
# Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
import matplotlib.pyplot as plt

# Se crea una imagen RGB de tamaño 9x8
imagen = np.zeros((9, 8, 3), dtype= np.uint8)

# Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=1 y 'x'=1
imagen[1, 1, 0] = 255
# Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=1 y 'x'=1
imagen[1, 1, 1] = 255

# Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=3 y 'x'=6
imagen[3, 6, 0] = 255
# Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=3 y 'x'=6
imagen[3, 6, 2] = 255

# Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=7 y 'x'=3
imagen[7, 3, 1] = 255
# Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=7 y 'x'=3
imagen[7, 3, 2] = 255

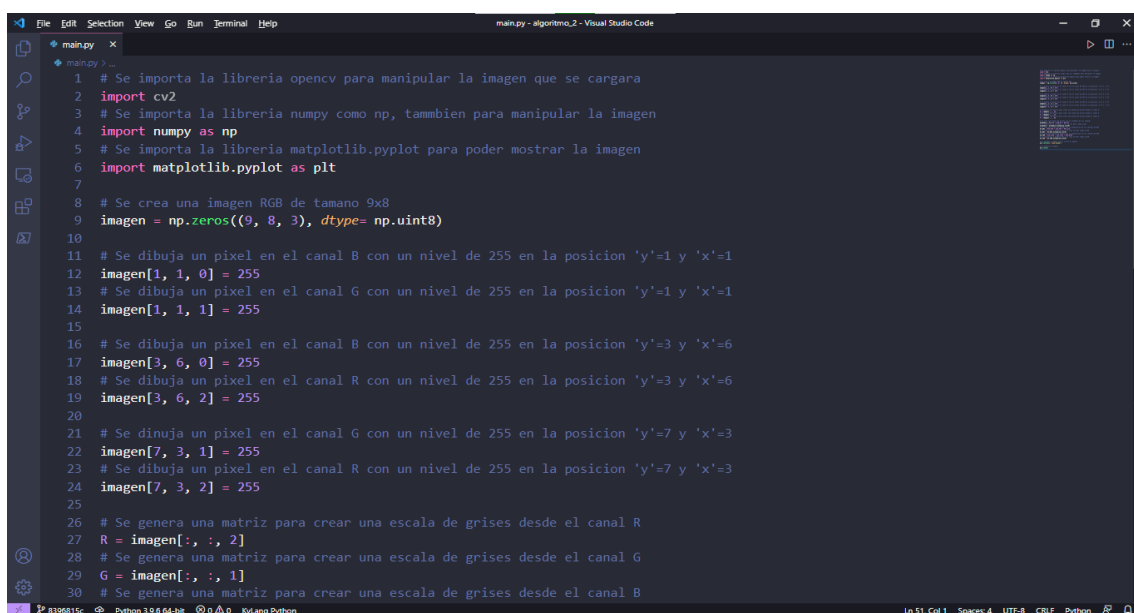
# Se genera una matriz para crear una escala de grises desde el canal R
R = imagen[:, :, 2]

# Se agrega la imagen con el filtro al pyplot
plt.imshow(R, cmap="gray")
```

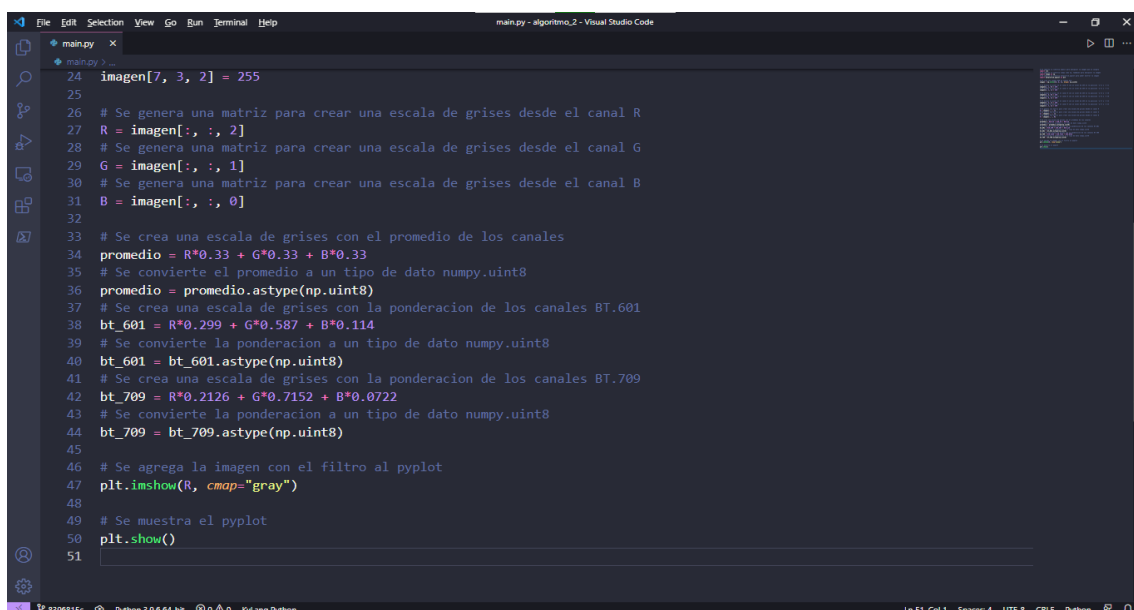
```
# Se muestra el pyplot
plt.show()
```

Para el código anterior, primero se importan las librerías necesarias para realizarlo (opencv, numpy, matplotlib). Después, con **numpy** se comienza a crear la imagen hecha en el algoritmo 1, se agregan los pixeles y se declara la variable R, después se agrega la matriz R al **pyplot**, al final se muestra la imagen con el **pyplot**.

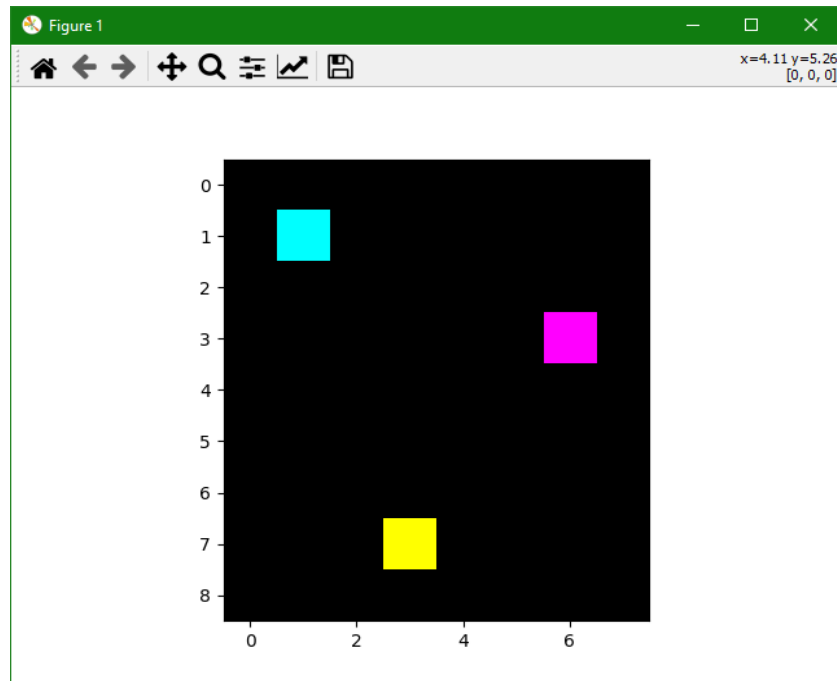
Se muestran imágenes del código



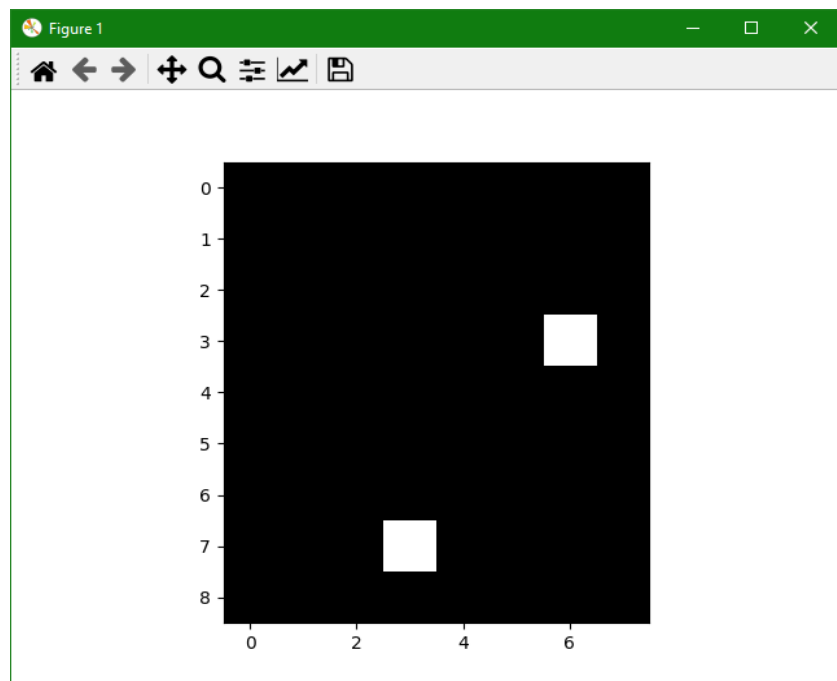
```
1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5 # Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
6 import matplotlib.pyplot as plt
7
8 # Se crea una imagen RGB de tamaño 9x8
9 imagen = np.zeros((9, 8, 3), dtype= np.uint8)
10
11 # Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=1 y 'x'=1
12 imagen[1, 1, 0] = 255
13 # Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=1 y 'x'=1
14 imagen[1, 1, 1] = 255
15
16 # Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=3 y 'x'=6
17 imagen[3, 6, 0] = 255
18 # Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=3 y 'x'=6
19 imagen[3, 6, 2] = 255
20
21 # Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=7 y 'x'=3
22 imagen[7, 3, 1] = 255
23 # Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=7 y 'x'=3
24 imagen[7, 3, 2] = 255
25
26 # Se genera una matriz para crear una escala de grises desde el canal R
27 R = imagen[:, :, 2]
28 # Se genera una matriz para crear una escala de grises desde el canal G
29 G = imagen[:, :, 1]
30 # Se genera una matriz para crear una escala de grises desde el canal B
```



```
24 imagen[7, 3, 2] = 255
25
26 # Se genera una matriz para crear una escala de grises desde el canal R
27 R = imagen[:, :, 2]
28 # Se genera una matriz para crear una escala de grises desde el canal G
29 G = imagen[:, :, 1]
30 # Se genera una matriz para crear una escala de grises desde el canal B
31 B = imagen[:, :, 0]
32
33 # Se crea una escala de grises con el promedio de los canales
34 promedio = R*0.33 + G*0.33 + B*0.33
35 # Se convierte el promedio a un tipo de dato numpy.uint8
36 promedio = promedio.astype(np.uint8)
37 # Se crea una escala de grises con la ponderación de los canales BT.601
38 bt_601 = R*0.299 + G*0.587 + B*0.114
39 # Se convierte la ponderación a un tipo de dato numpy.uint8
40 bt_601 = bt_601.astype(np.uint8)
41 # Se crea una escala de grises con la ponderación de los canales BT.709
42 bt_709 = R*0.2126 + G*0.7152 + B*0.0722
43 # Se convierte la ponderación a un tipo de dato numpy.uint8
44 bt_709 = bt_709.astype(np.uint8)
45
46 # Se agrega la imagen con el filtro al pyplot
47 plt.imshow(R, cmap="gray")
48
49 # Se muestra el pyplot
50 plt.show()
51
```

Imagen original

Se muestra una imagen de lo que realiza el código



Método RGB (G)

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np
# Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
import matplotlib.pyplot as plt

# Se crea una imagen RGB de tamaño 9x8
imagen = np.zeros((9, 8, 3), dtype= np.uint8)

# Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=1 y 'x'=1
imagen[1, 1, 0] = 255
# Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=1 y 'x'=1
imagen[1, 1, 1] = 255

# Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=3 y 'x'=6
imagen[3, 6, 0] = 255
# Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=3 y 'x'=6
imagen[3, 6, 2] = 255

# Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=7 y 'x'=3
imagen[7, 3, 1] = 255
# Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=7 y 'x'=3
imagen[7, 3, 2] = 255

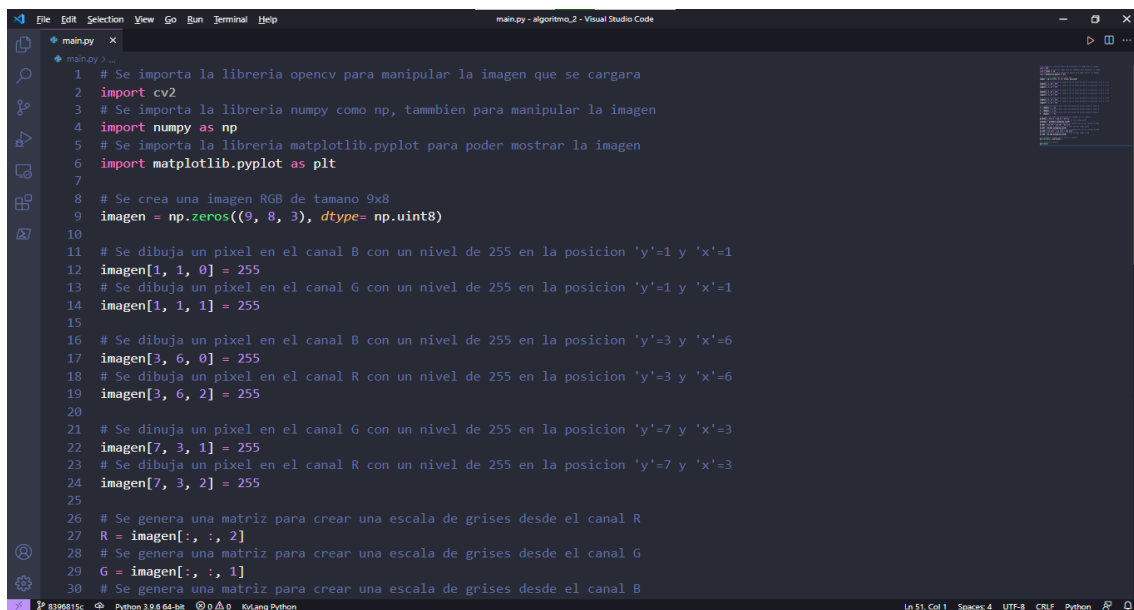
# Se genera una matriz para crear una escala de grises desde el canal G
G = imagen[:, :, 1]

# Se agrega la imagen con el filtro al pyplot
plt.imshow(G, cmap="gray")

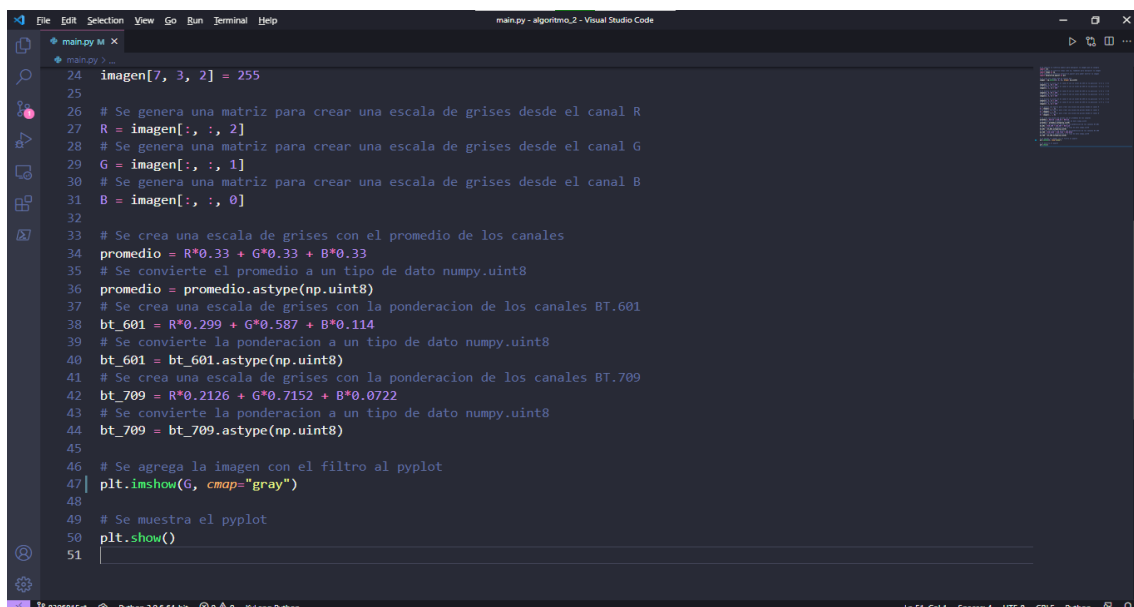
# Se muestra el pyplot
plt.show()
```


Para el código anterior, primero se importan las librerías necesarias para realizarlo (opencv, numpy, matplotlib). Después, con *numpy* se genera la base de la imagen que se hizo en el algoritmo 1 y se agregan los pixeles. Luego se declara la variable G y se agrega al *pyplot*, al final se muestra la imagen con el *pyplot*.

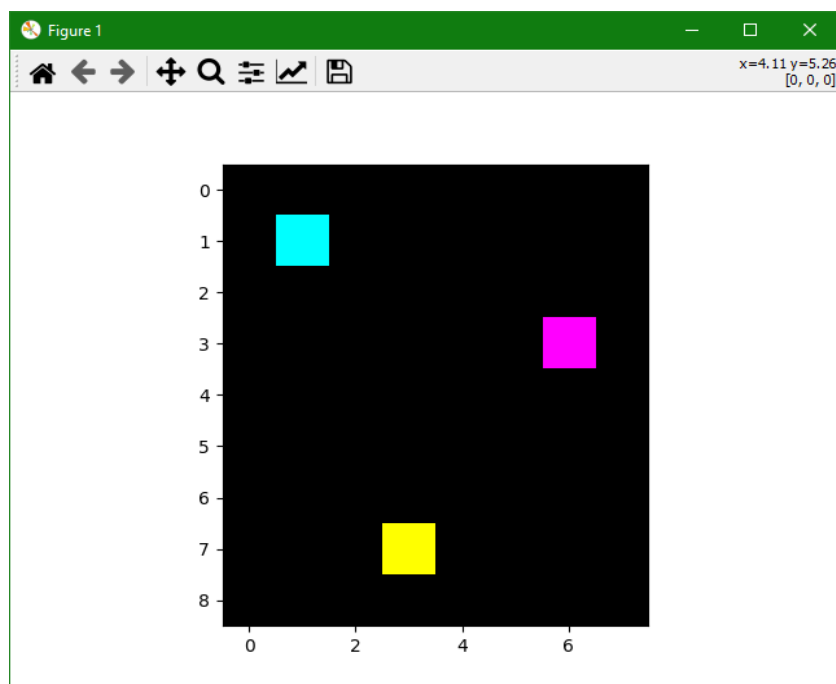
Se muestran imágenes del código



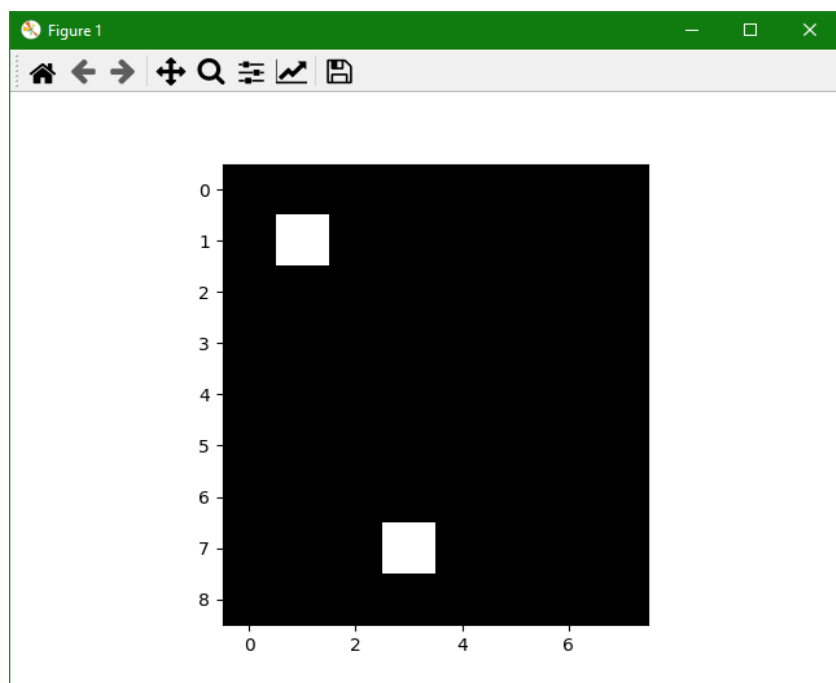
```
1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5 # Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
6 import matplotlib.pyplot as plt
7
8 # Se crea una imagen RGB de tamaño 9x8
9 imagen = np.zeros((9, 8, 3), dtype= np.uint8)
10
11 # Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=1 y 'x'=1
12 imagen[1, 1, 0] = 255
13 # Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=1 y 'x'=1
14 imagen[1, 1, 1] = 255
15
16 # Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=3 y 'x'=6
17 imagen[3, 6, 0] = 255
18 # Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=3 y 'x'=6
19 imagen[3, 6, 2] = 255
20
21 # Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=7 y 'x'=3
22 imagen[7, 3, 1] = 255
23 # Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=7 y 'x'=3
24 imagen[7, 3, 2] = 255
25
26 # Se genera una matriz para crear una escala de grises desde el canal R
27 R = imagen[:, :, 2]
28 # Se genera una matriz para crear una escala de grises desde el canal G
29 G = imagen[:, :, 1]
30 # Se genera una matriz para crear una escala de grises desde el canal B
```



```
24 imagen[7, 3, 2] = 255
25
26 # Se genera una matriz para crear una escala de grises desde el canal R
27 R = imagen[:, :, 2]
28 # Se genera una matriz para crear una escala de grises desde el canal G
29 G = imagen[:, :, 1]
30 # Se genera una matriz para crear una escala de grises desde el canal B
31 B = imagen[:, :, 0]
32
33 # Se crea una escala de grises con el promedio de los canales
34 promedio = R*0.33 + G*0.33 + B*0.33
35 # Se convierte el promedio a un tipo de dato numpy.uint8
36 promedio = promedio.astype(np.uint8)
37 # Se crea una escala de grises con la ponderación de los canales BT.601
38 bt_601 = R*0.299 + G*0.587 + B*0.114
39 # Se convierte la ponderación a un tipo de dato numpy.uint8
40 bt_601 = bt_601.astype(np.uint8)
41 # Se crea una escala de grises con la ponderación de los canales BT.709
42 bt_709 = R*0.2126 + G*0.7152 + B*0.0722
43 # Se convierte la ponderación a un tipo de dato numpy.uint8
44 bt_709 = bt_709.astype(np.uint8)
45
46 # Se agrega la imagen con el filtro al pyplot
47 plt.imshow(G, cmap="gray")
48
49 # Se muestra el pyplot
50 plt.show()
51
```

Imagen original

Se muestra una imagen de lo que el código realiza



Método RGB (B)

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np
# Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
import matplotlib.pyplot as plt

# Se crea una imagen RGB de tamaño 9x8
imagen = np.zeros((9, 8, 3), dtype= np.uint8)

# Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=1 y 'x'=1
imagen[1, 1, 0] = 255
# Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=1 y 'x'=1
imagen[1, 1, 1] = 255

# Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=3 y 'x'=6
imagen[3, 6, 0] = 255
# Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=3 y 'x'=6
imagen[3, 6, 2] = 255

# Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=7 y 'x'=3
imagen[7, 3, 1] = 255
# Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=7 y 'x'=3
imagen[7, 3, 2] = 255

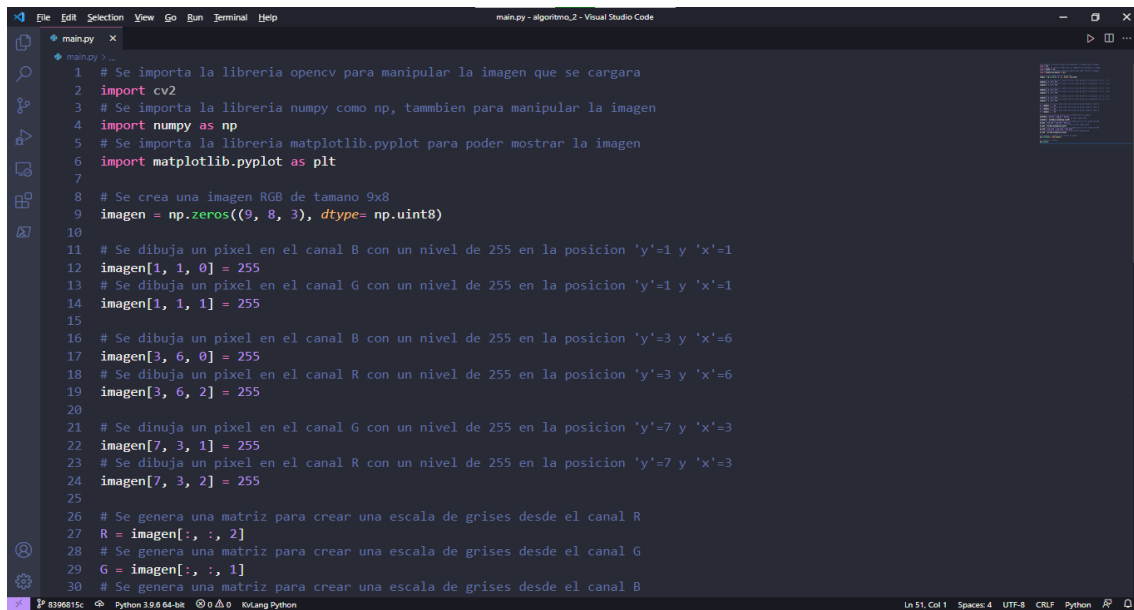
# Se genera una matriz para crear una escala de grises desde el canal B
B = imagen[:, :, 0]

# Se agrega la imagen con el filtro al pyplot
plt.imshow(B, cmap="gray")

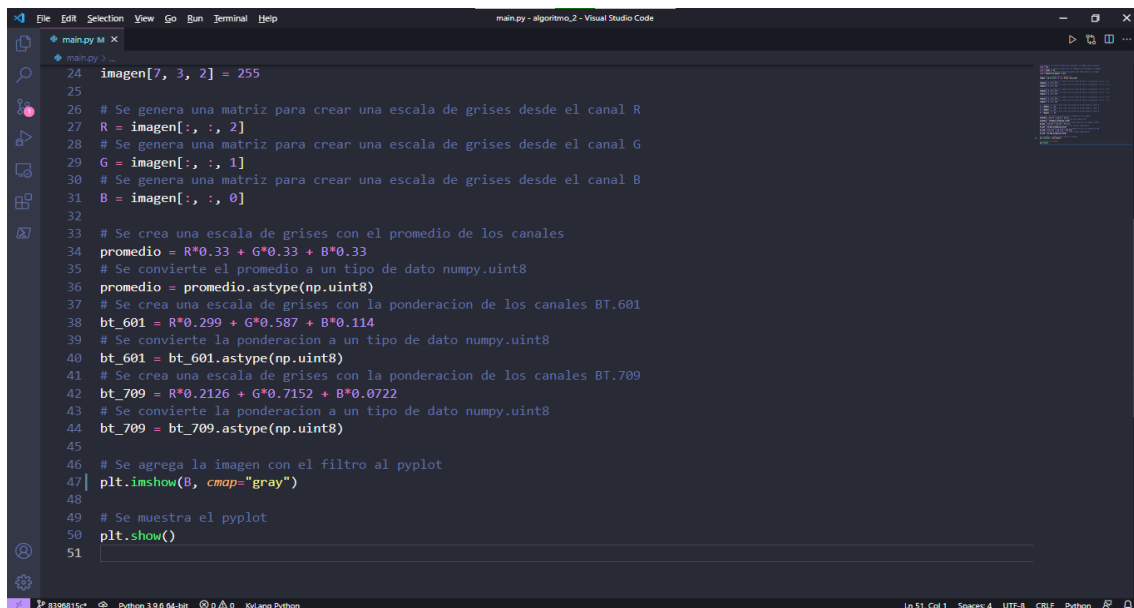
# Se muestra el pyplot
plt.show()
```

Para el código anterior, primero se importan las librerías necesarias para realizarlo (opencv, numpy, matplotlib). Después, con *numpy* se genera la base de la imagen que se hizo en el algoritmo 1 y se agregan los pixeles. Luego se declara la variable B y se agrega al *pyplot*, al final se muestra la imagen con el *pyplot*.

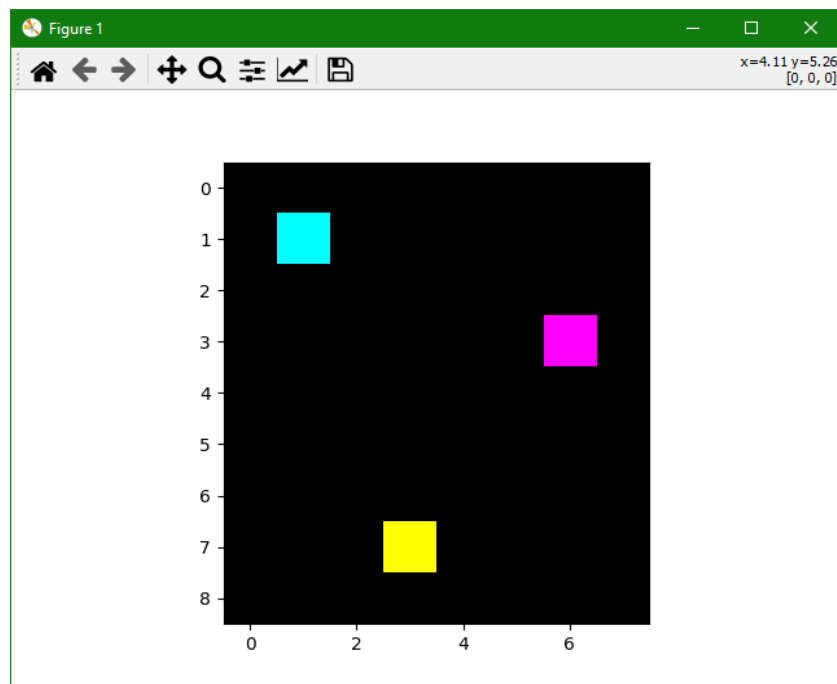
Se muestran imágenes del código



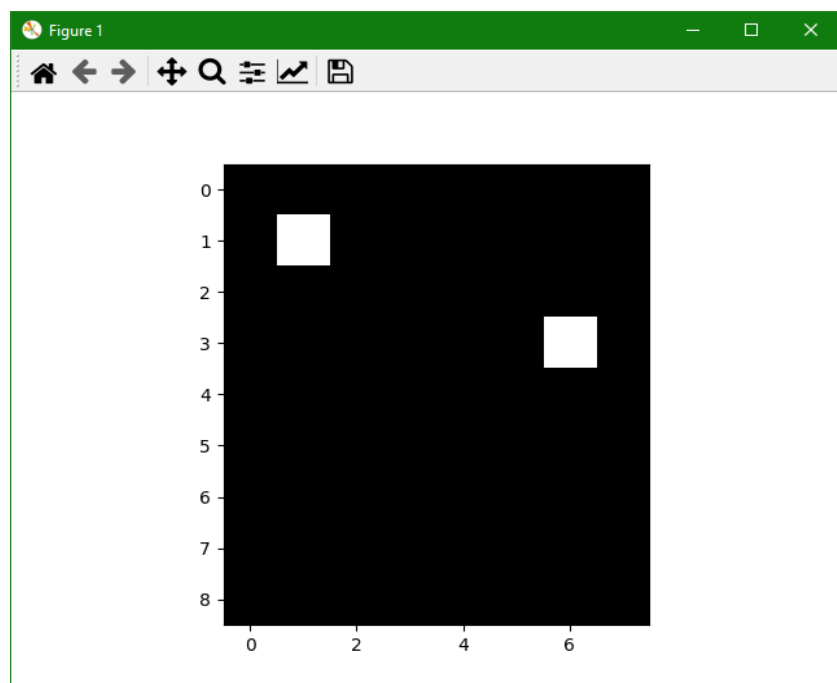
```
1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5 # Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
6 import matplotlib.pyplot as plt
7
8 # Se crea una imagen RGB de tamaño 9x8
9 imagen = np.zeros((9, 8, 3), dtype= np.uint8)
10
11 # Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=1 y 'x'=1
12 imagen[1, 1, 0] = 255
13 # Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=1 y 'x'=1
14 imagen[1, 1, 1] = 255
15
16 # Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=3 y 'x'=6
17 imagen[3, 6, 0] = 255
18 # Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=3 y 'x'=6
19 imagen[3, 6, 2] = 255
20
21 # Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=7 y 'x'=3
22 imagen[7, 3, 1] = 255
23 # Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=7 y 'x'=3
24 imagen[7, 3, 2] = 255
25
26 # Se genera una matriz para crear una escala de grises desde el canal R
27 R = imagen[:, :, 2]
28 # Se genera una matriz para crear una escala de grises desde el canal G
29 G = imagen[:, :, 1]
30 # Se genera una matriz para crear una escala de grises desde el canal B
```



```
24 imagen[7, 3, 2] = 255
25
26 # Se genera una matriz para crear una escala de grises desde el canal R
27 R = imagen[:, :, 2]
28 # Se genera una matriz para crear una escala de grises desde el canal G
29 G = imagen[:, :, 1]
30 # Se genera una matriz para crear una escala de grises desde el canal B
31 B = imagen[:, :, 0]
32
33 # Se crea una escala de grises con el promedio de los canales
34 promedio = R*0.33 + G*0.33 + B*0.33
35 # Se convierte el promedio a un tipo de dato numpy.uint8
36 promedio = promedio.astype(np.uint8)
37 # Se crea una escala de grises con la ponderación de los canales BT.601
38 bt_601 = R*0.299 + G*0.587 + B*0.114
39 # Se convierte la ponderación a un tipo de dato numpy.uint8
40 bt_601 = bt_601.astype(np.uint8)
41 # Se crea una escala de grises con la ponderación de los canales BT.709
42 bt_709 = R*0.2126 + G*0.7152 + B*0.0722
43 # Se convierte la ponderación a un tipo de dato numpy.uint8
44 bt_709 = bt_709.astype(np.uint8)
45
46 # Se agrega la imagen con el filtro al pyplot
47 plt.imshow(B, cmap="gray")
48
49 # Se muestra el pyplot
50 plt.show()
51
```

Imagen original

Se muestra una imagen de lo que realiza el código



Método Promedio

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np
# Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
import matplotlib.pyplot as plt

# Se crea una imagen RGB de tamaño 9x8
imagen = np.zeros((9, 8, 3), dtype= np.uint8)

# Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=1 y 'x'=1
imagen[1, 1, 0] = 255
# Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=1 y 'x'=1
imagen[1, 1, 1] = 255

# Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=3 y 'x'=6
imagen[3, 6, 0] = 255
# Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=3 y 'x'=6
imagen[3, 6, 2] = 255

# Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=7 y 'x'=3
imagen[7, 3, 1] = 255
# Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=7 y 'x'=3
imagen[7, 3, 2] = 255

# Se genera una matriz para crear una escala de grises desde el canal R
R = imagen[:, :, 2]
# Se genera una matriz para crear una escala de grises desde el canal G
G = imagen[:, :, 1]
# Se genera una matriz para crear una escala de grises desde el canal B
B = imagen[:, :, 0]

# Se crea una escala de grises con el promedio de los canales
```

```

promedio = R*0.33 + G*0.33 + B*0.33
# Se convierte el promedio a un tipo de dato numpy.uint8
promedio = promedio.astype(np.uint8)

```

```

# Se agrega la imagen con el filtro al pyplot
plt.imshow(promedio, cmap="gray")

```

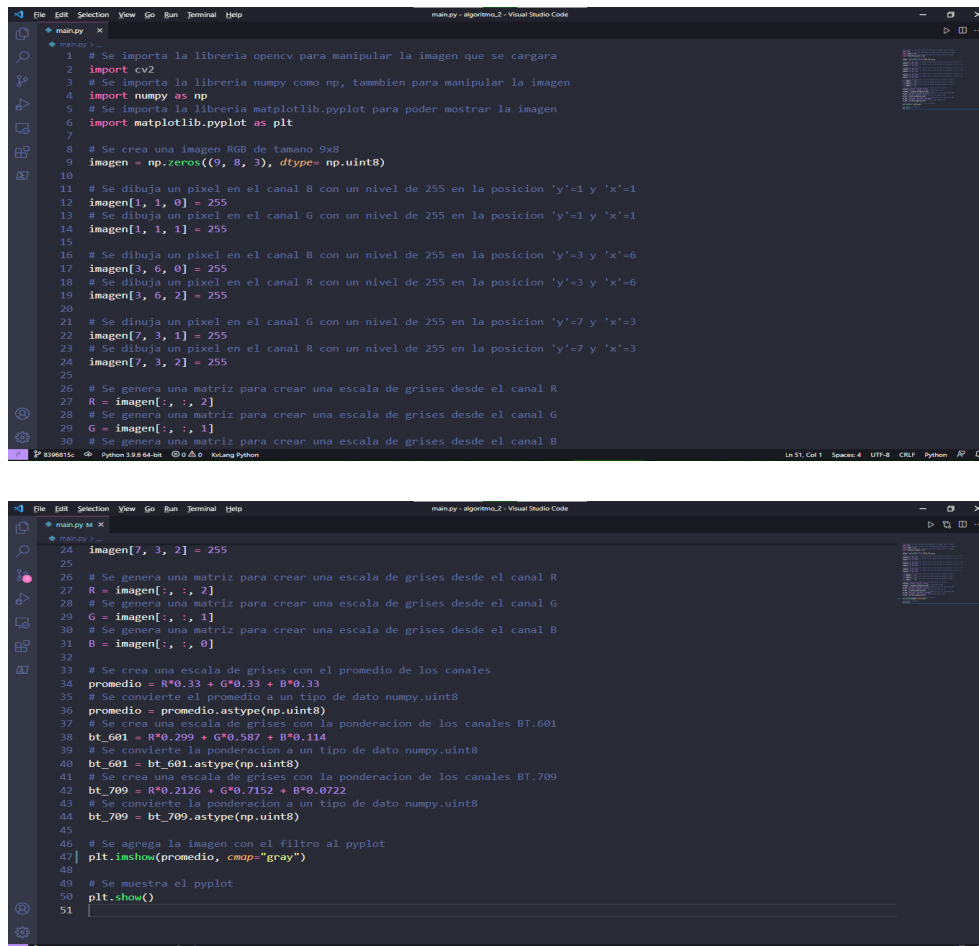
```

# Se muestra el pyplot
plt.show()

```

Para el código anterior, primero se importan las librerías necesarias para realizarlo (opencv, numpy, matplotlib). Después, con **numpy** se genera la base de la imagen que se hizo en el algoritmo 1 y se agregan los pixeles. Luego se declaran las variables R, G y B, con estas se genera el promedio de RGB y se agrega al **pyplot**. Al final se muestra la imagen con el **pyplot**.

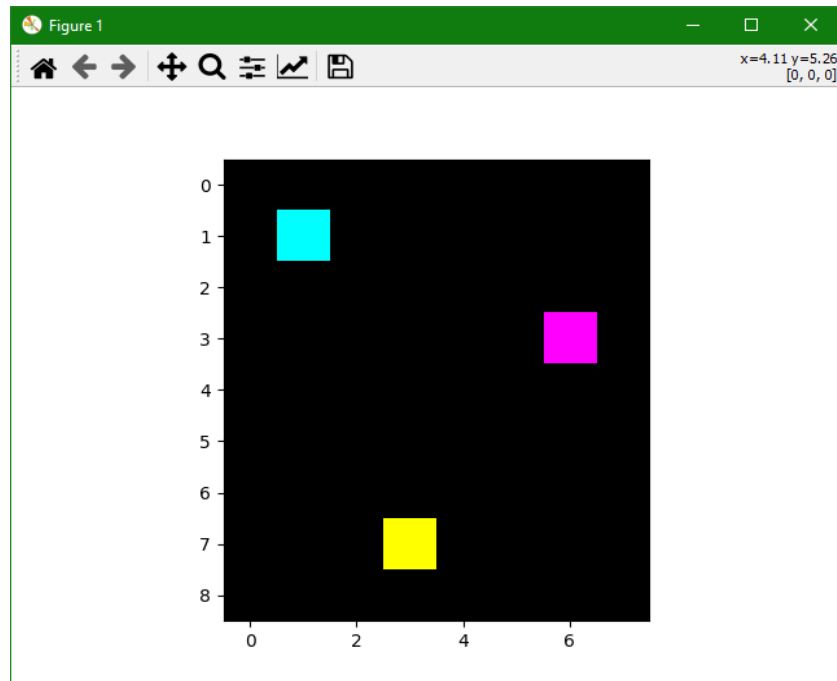
Se muestran imágenes del código



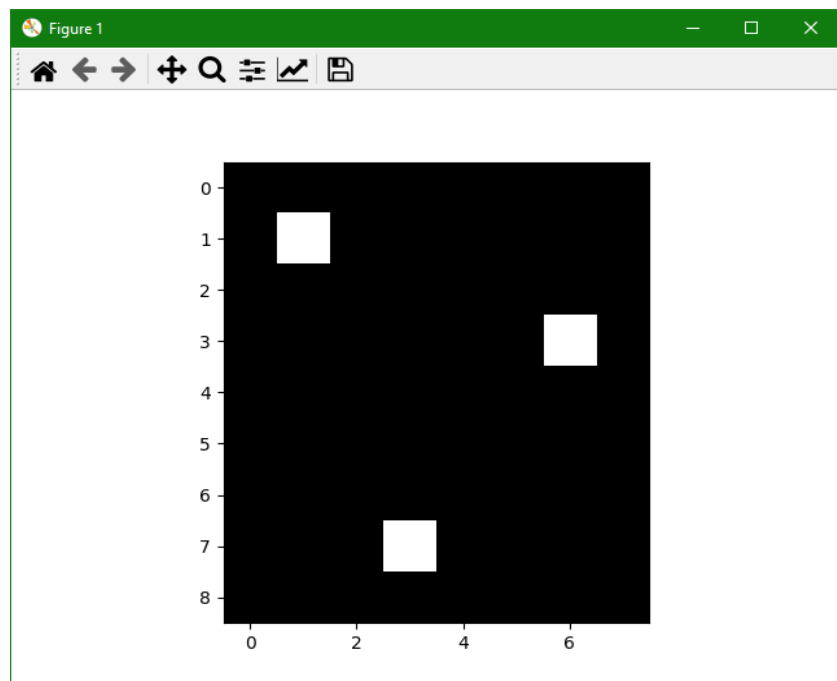
```

1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5 # Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
6 import matplotlib.pyplot as plt
7
8 # Se crea una imagen RGB de tamaño 9x8
9 imagen = np.zeros((9, 8, 3), dtype= np.uint8)
10
11 # Se dibuja un pixel en el canal B con un nivel de 255 en la posicion 'y'=1 y 'x'=1
12 imagen[1, 1, 0] = 255
13 # Se dibuja un pixel en el canal G con un nivel de 255 en la posicion 'y'=1 y 'x'=1
14 imagen[1, 1, 1] = 255
15
16 # Se dibuja un pixel en el canal B con un nivel de 255 en la posicion 'y'=3 y 'x'=6
17 imagen[3, 6, 0] = 255
18 # Se dibuja un pixel en el canal R con un nivel de 255 en la posicion 'y'=3 y 'x'=6
19 imagen[3, 6, 2] = 255
20
21 # Se dibuja un pixel en el canal G con un nivel de 255 en la posicion 'y'=7 y 'x'=3
22 imagen[7, 3, 1] = 255
23 # Se dibuja un pixel en el canal R con un nivel de 255 en la posicion 'y'=7 y 'x'=3
24 imagen[7, 3, 2] = 255
25
26 # Se genera una matriz para crear una escala de grises desde el canal R
27 R = imagen[:, :, 2]
28 # Se genera una matriz para crear una escala de grises desde el canal G
29 G = imagen[:, :, 1]
30 # Se genera una matriz para crear una escala de grises desde el canal B
31 B = imagen[:, :, 0]
32
33 # Se crea una escala de grises con el promedio de los canales
34 promedio = R*0.33 + G*0.33 + B*0.33
35 # Se convierte el promedio a un tipo de dato numpy.uint8
36 promedio = promedio.astype(np.uint8)
37 # Se crea una escala de grises con la ponderacion de los canales BT.601
38 bt_601 = R*0.299 + G*0.587 + B*0.114
39 # Se convierte la ponderacion a un tipo de dato numpy.uint8
40 bt_601 = bt_601.astype(np.uint8)
41 # Se crea una escala de grises con la ponderacion de los canales BT.709
42 bt_709 = R*0.2126 + G*0.7152 + B*0.0722
43 # Se convierte la ponderacion a un tipo de dato numpy.uint8
44 bt_709 = bt_709.astype(np.uint8)
45
46 # Se agrega la imagen con el filtro al pyplot
47 plt.imshow(promedio, cmap="gray")
48
49 # Se muestra el pyplot
50 plt.show()
51

```

Imagen original

Se muestra una imagen de lo que el código realiza



Método BT.601

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np
# Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
import matplotlib.pyplot as plt

# Se crea una imagen RGB de tamaño 9x8
imagen = np.zeros((9, 8, 3), dtype= np.uint8)

# Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=1 y 'x'=1
imagen[1, 1, 0] = 255
# Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=1 y 'x'=1
imagen[1, 1, 1] = 255

# Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=3 y 'x'=6
imagen[3, 6, 0] = 255
# Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=3 y 'x'=6
imagen[3, 6, 2] = 255

# Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=7 y 'x'=3
imagen[7, 3, 1] = 255
# Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=7 y 'x'=3
imagen[7, 3, 2] = 255

# Se genera una matriz para crear una escala de grises desde el canal R
R = imagen[:, :, 2]
# Se genera una matriz para crear una escala de grises desde el canal G
G = imagen[:, :, 1]
# Se genera una matriz para crear una escala de grises desde el canal B
B = imagen[:, :, 0]

# Se crea una escala de grises con la ponderación de los canales BT.601
```

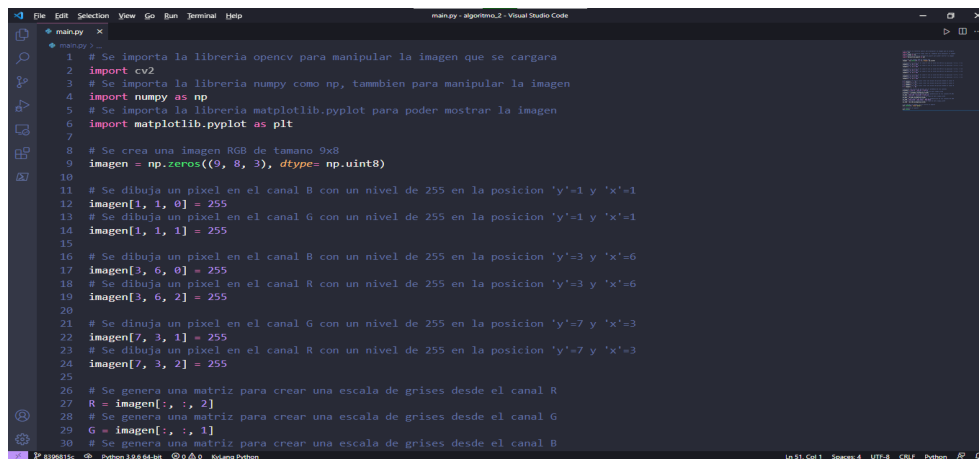
```
bt_601 = R*0.299 + G*0.587 + B*0.114
# Se convierte la ponderacion a un tipo de dato numpy.uint8
bt_601 = bt_601.astype(np.uint8)
```

```
# Se agrega la imagen con el filtro al pyplot
plt.imshow(bt_601, cmap="gray")
```

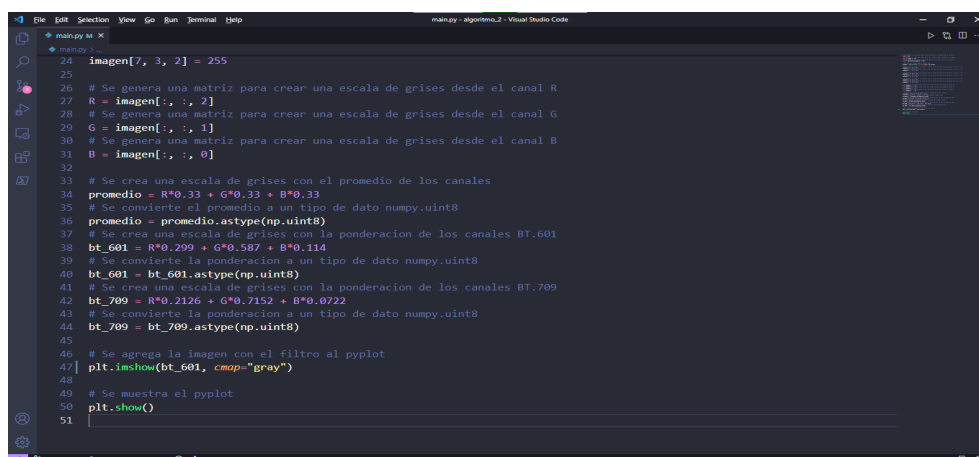
```
# Se muestra el pyplot
plt.show()
```

Para el código anterior, primero se importan las librerías necesarias para realizarlo (opencv, numpy, matplotlib). Después, con **numpy** se genera la base de la imagen que se hizo en el algoritmo 1 y se agregan los pixeles. Luego se declaran las variables R, G y B, con estas se genera la ponderación BT.601 y se agrega al **pyplot**. Al final se muestra la imagen con el **pyplot**.

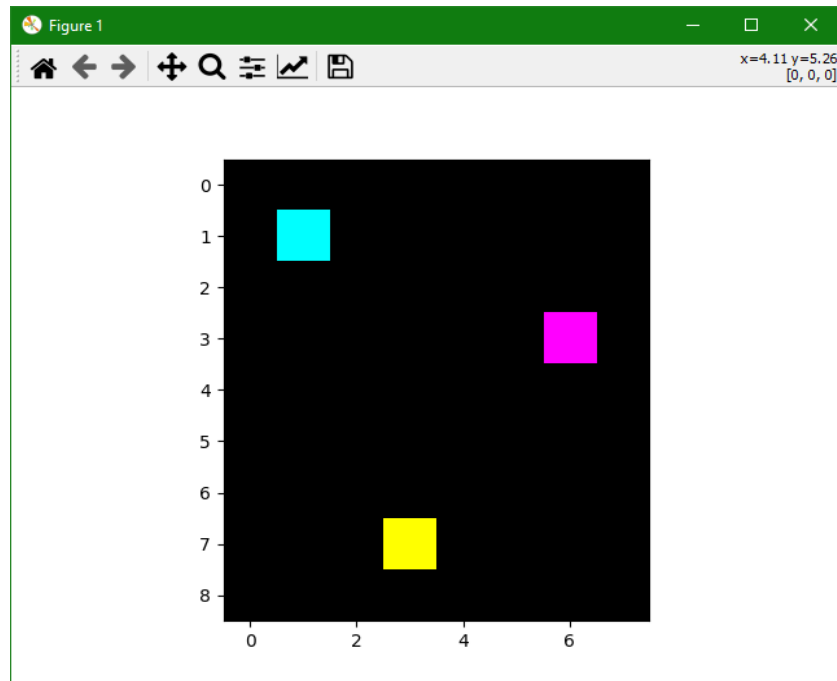
Se muestran imágenes del código



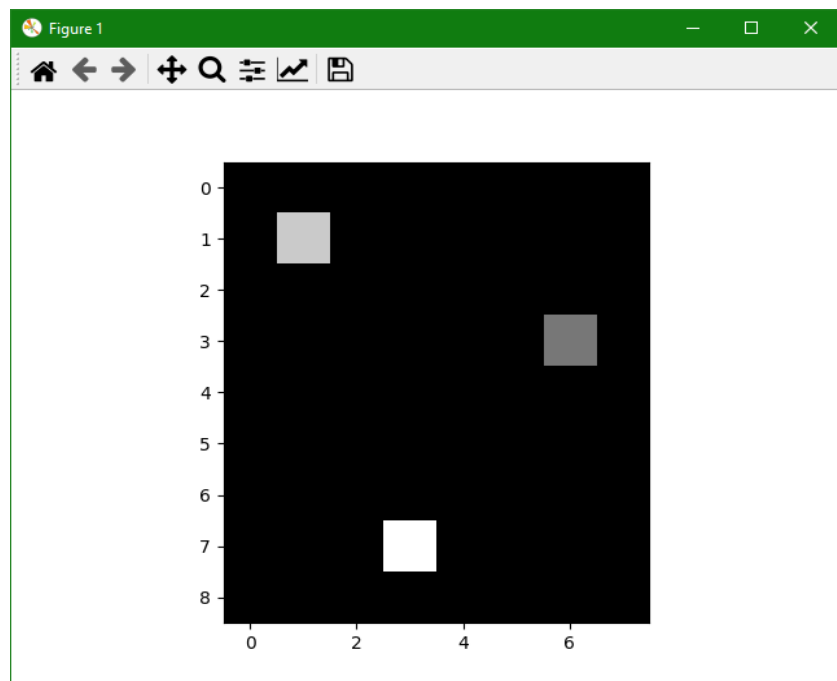
```
1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5 # Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
6 import matplotlib.pyplot as plt
7
8 # Se crea una imagen RGB de tamaño 9x3
9 imagen = np.zeros((9, 3), dtype= np.uint8)
10
11 # Se dibuja un pixel en el canal B con un nivel de 255 en la posicion 'y'=1 y 'x'=1
12 imagen[1, 1, 0] = 255
13 # Se dibuja un pixel en el canal G con un nivel de 255 en la posicion 'y'=1 y 'x'=1
14 imagen[1, 1, 1] = 255
15
16 # Se dibuja un pixel en el canal B con un nivel de 255 en la posicion 'y'=3 y 'x'=6
17 imagen[3, 6, 0] = 255
18 # Se dibuja un pixel en el canal R con un nivel de 255 en la posicion 'y'=3 y 'x'=6
19 imagen[3, 6, 2] = 255
20
21 # Se dibuja un pixel en el canal G con un nivel de 255 en la posicion 'y'=7 y 'x'=3
22 imagen[7, 3, 1] = 255
23 # Se dibuja un pixel en el canal R con un nivel de 255 en la posicion 'y'=7 y 'x'=3
24 imagen[7, 3, 2] = 255
25
26 # Se genera una matriz para crear una escala de grises desde el canal R
27 R = imagen[:, :, 2]
28 # Se genera una matriz para crear una escala de grises desde el canal G
29 G = imagen[:, :, 1]
30 # Se genera una matriz para crear una escala de grises desde el canal B
```



```
24 imagen[7, 3, 2] = 255
25
26 # Se genera una matriz para crear una escala de grises desde el canal R
27 R = imagen[:, :, 2]
28 # Se genera una matriz para crear una escala de grises desde el canal G
29 G = imagen[:, :, 1]
30 # Se genera una matriz para crear una escala de grises desde el canal B
31 B = imagen[:, :, 0]
32
33 # Se crea una escala de grises con el promedio de los canales
34 promedio = R*0.33 + G*0.33 + B*0.33
35 # Se convierte el promedio a un tipo de dato numpy.uint8
36 promedio = promedio.astype(np.uint8)
37 # Se crea una escala de grises con la ponderacion de los canales BT.601
38 bt_601 = R*0.299 + G*0.587 + B*0.114
39 # Se convierte la ponderacion a un tipo de dato numpy.uint8
40 bt_601 = bt_601.astype(np.uint8)
41 # Se crea una escala de grises con la ponderacion de los canales BT.709
42 bt_709 = R*0.2126 + G*0.7152 + B*0.0722
43 # Se convierte la ponderacion a un tipo de dato numpy.uint8
44 bt_709 = bt_709.astype(np.uint8)
45
46 # Se agrega la imagen con el filtro al pyplot
47 plt.imshow(bt_601, cmap="gray")
48
49 # Se muestra el pyplot
50 plt.show()
51
```

Imagen original

Se muestra una imagen de lo que realiza el código



Método BT.709

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np
# Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
import matplotlib.pyplot as plt

# Se crea una imagen RGB de tamaño 9x8
imagen = np.zeros((9, 8, 3), dtype= np.uint8)

# Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=1 y 'x'=1
imagen[1, 1, 0] = 255
# Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=1 y 'x'=1
imagen[1, 1, 1] = 255

# Se dibuja un pixel en el canal B con un nivel de 255 en la posición 'y'=3 y 'x'=6
imagen[3, 6, 0] = 255
# Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=3 y 'x'=6
imagen[3, 6, 2] = 255

# Se dibuja un pixel en el canal G con un nivel de 255 en la posición 'y'=7 y 'x'=3
imagen[7, 3, 1] = 255
# Se dibuja un pixel en el canal R con un nivel de 255 en la posición 'y'=7 y 'x'=3
imagen[7, 3, 2] = 255

# Se genera una matriz para crear una escala de grises desde el canal R
R = imagen[:, :, 2]
# Se genera una matriz para crear una escala de grises desde el canal G
G = imagen[:, :, 1]
# Se genera una matriz para crear una escala de grises desde el canal B
B = imagen[:, :, 0]

# Se crea una escala de grises con la ponderación de los canales BT.709
```

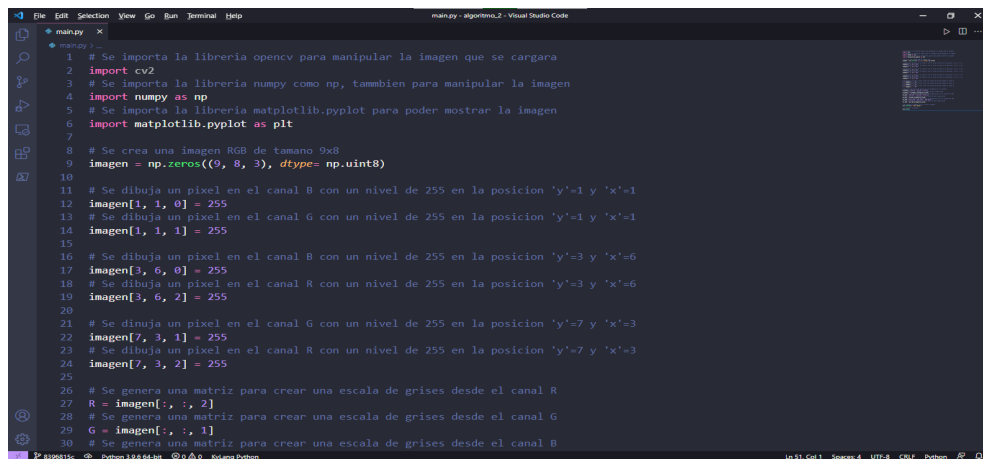
```
bt_709 = R*0.2126 + G*0.7152 + B*0.0722
# Se convierte la ponderacion a un tipo de dato numpy.uint8
bt_709 = bt_709.astype(np.uint8)
```

```
# Se agrega la imagen con el filtro al pyplot
plt.imshow(bt_709, cmap="gray")
```

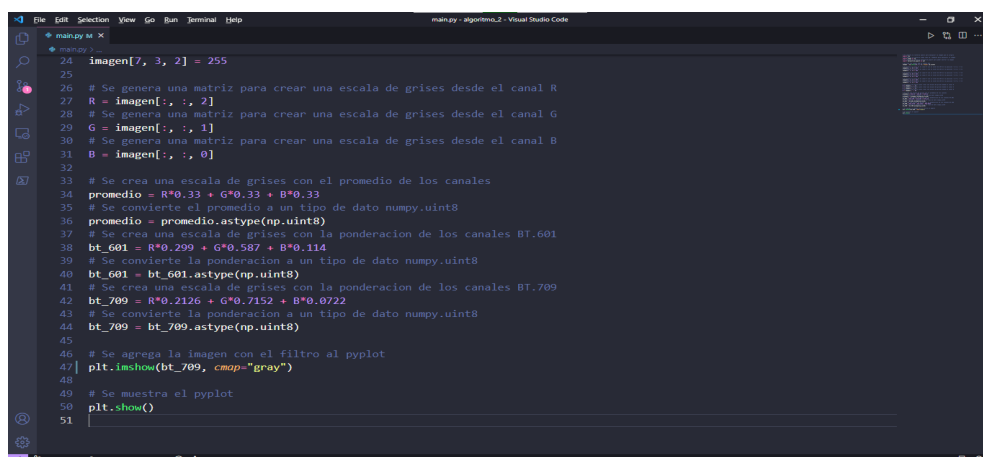
```
# Se muestra el pyplot
plt.show()
```

Para el código anterior, primero se importan las librerías necesarias para relizarlo (opencv, numpy, matplotlib). Después, con ***numpy*** se genera la base de la imagen que se hizo en el algoritmo 1 y se agregan los pixeles. Luego se declaran las variables R, G y B, con estas se genera la ponderación BT.709 y se agrega al ***pyplot***. Al final se muestra la imagen con el ***pyplot***.

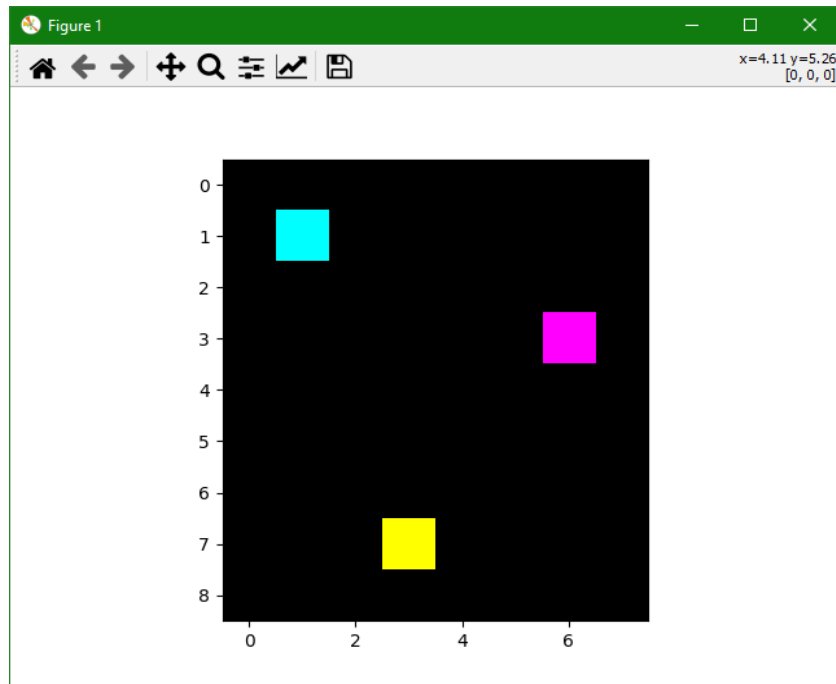
Se muestran imágenes del código



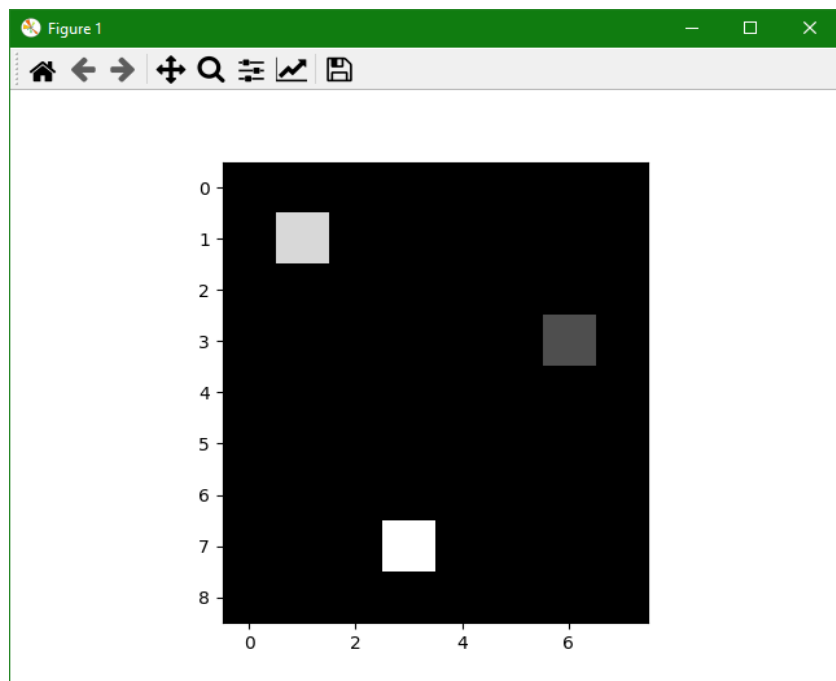
```
1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5 # Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
6 import matplotlib.pyplot as plt
7
8 # Se crea una imagen RGB de tamaño 9x3
9 imagen = np.zeros((9, 3), dtype= np.uint8)
10
11 # Se dibuja un pixel en el canal B con un nivel de 255 en la posicion 'y'=1 y 'x'=1
12 imagen[1, 1, 0] = 255
13 # Se dibuja un pixel en el canal G con un nivel de 255 en la posicion 'y'=1 y 'x'=1
14 imagen[1, 1, 1] = 255
15
16 # Se dibuja un pixel en el canal B con un nivel de 255 en la posicion 'y'=3 y 'x'=6
17 imagen[3, 6, 0] = 255
18 # Se dibuja un pixel en el canal R con un nivel de 255 en la posicion 'y'=3 y 'x'=6
19 imagen[3, 6, 2] = 255
20
21 # Se dibuja un pixel en el canal G con un nivel de 255 en la posicion 'y'=7 y 'x'=3
22 imagen[7, 3, 1] = 255
23 # Se dibuja un pixel en el canal R con un nivel de 255 en la posicion 'y'=7 y 'x'=3
24 imagen[7, 3, 2] = 255
25
26 # Se genera una matriz para crear una escala de grises desde el canal R
27 R = imagen[:, :, 2]
28 # Se genera una matriz para crear una escala de grises desde el canal G
29 G = imagen[:, :, 1]
30 # Se genera una matriz para crear una escala de grises desde el canal B
```



```
24 imagen[7, 3, 2] = 255
25
26 # Se genera una matriz para crear una escala de grises desde el canal R
27 R = imagen[:, :, 2]
28 # Se genera una matriz para crear una escala de grises desde el canal G
29 G = imagen[:, :, 1]
30 # Se genera una matriz para crear una escala de grises desde el canal B
31 B = imagen[:, :, 0]
32
33 # Se crea una escala de grises con el promedio de los canales
34 promedio = R*0.33 + G*0.33 + B*0.33
35 # Se convierte el promedio a un tipo de dato numpy.uint8
36 promedio = promedio.astype(np.uint8)
37 # Se crea una escala de grises con la ponderacion de los canales BT.601
38 bt_601 = R*0.299 + G*0.587 + B*0.114
39 # Se convierte la ponderacion a un tipo de dato numpy.uint8
40 bt_601 = bt_601.astype(np.uint8)
41 # Se crea una escala de grises con la ponderacion de los canales BT.709
42 bt_709 = R*0.2126 + G*0.7152 + B*0.0722
43 # Se convierte la ponderacion a un tipo de dato numpy.uint8
44 bt_709 = bt_709.astype(np.uint8)
45
46 # Se agrega la imagen con el filtro al pyplot
47 plt.imshow(bt_709, cmap="gray")
48
49 # Se muestra el pyplot
50 plt.show()
51
```

Imagen original

Se muestra una imagen de lo que el código realiza



*Punto D (Algoritmo 1)**Método RGB (R)*

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np
# Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
import matplotlib.pyplot as plt

# Se crea una imagen RGB de tamaño 16x8
imagen = np.zeros((16, 8, 3), dtype= np.uint8)

# Se dibuja un rectangulo en el canal B con un nivel de 125 desde la fila 1 hasta la 2
# desde la columna 1 hasta la 6
imagen[1:3, 1:7, 0] = 125
# Se dibuja un rectangulo en el canal G con un nivel de 225 desde la fila 1 hasta la 2 y
# desde la columna 1 hasta la 6
imagen[1:3, 1:7, 1] = 225

# Se dibuja un rectangulo en el canal B con un nivel de 100 desde la fila 4 hasta la 10 y
# desde la columna 2 hasta la 5
imagen[4:11, 2:6, 0] = 100
# Se dibuja un rectangulo en el canal R con un nivel de 250 desde la fila 4 hasta la 10 y
# desde la columna 2 hasta la 5
imagen[4:11, 2:6, 2] = 250

# Se dibuja un rectangulo en el canal G con un nivel de 200 desde la fila 12 hasta la 14
y
# desde la columna 1 hasta la 6
imagen[12:15, 1:7, 1] = 200
# Se dibuja un rectangulo en el canal R con un nivel de 190 desde la fila 12 hasta la 14 y
# desde la columna 1 hasta la 6
imagen[12:15, 1:7, 2] = 190
```

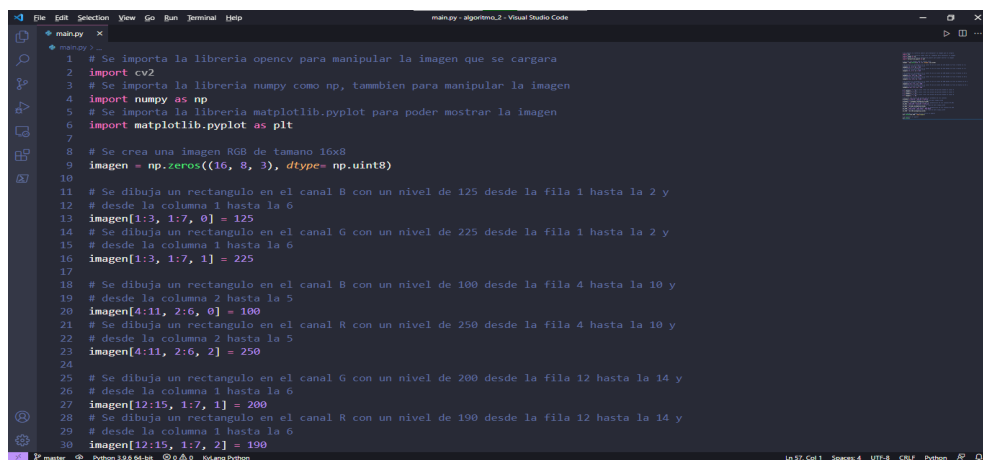
```
# Se genera una matriz para crear una escala de grises desde el canal R
R = imagen[:, :, 2]
```

```
# Se agrega la imagen con el filtro al pyplot
plt.imshow(R, cmap="gray")
```

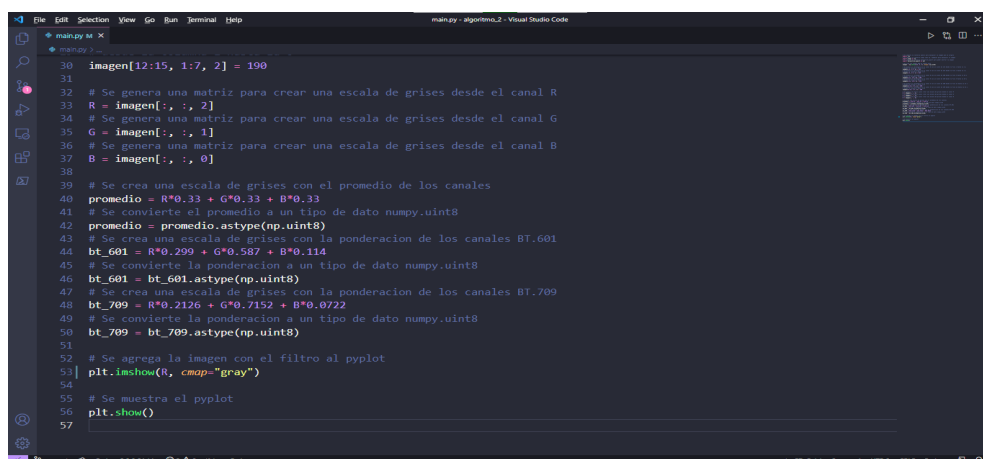
```
# Se muestra el pyplot
plt.show()
```

Para el código anterior, primer se importan las librerías necesarias para realizarlo (opencv, numpy, matplotlib). Después, con *numpy* se genera la base de la imagen que se hizo en el algoritmo 1, se agregan los rectangulos y se declara la variable R, luego esta se agrega al *pyplot* y al final se muestra la imagen con el *pyplot*.

Se muestran imágenes del código

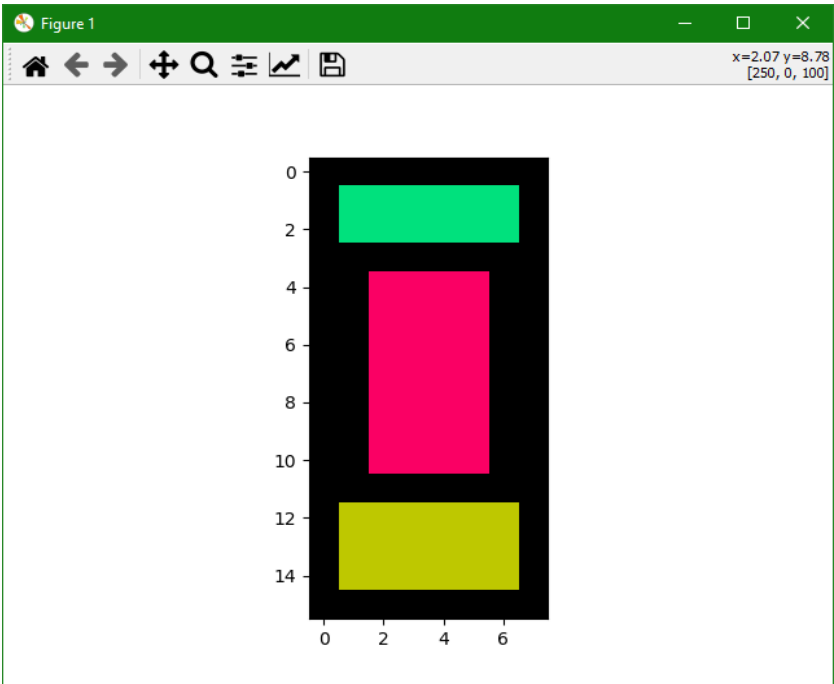


```
1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5 # Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
6 import matplotlib.pyplot as plt
7
8 # Se crea una imagen RGB de tamaño 16x8
9 imagen = np.zeros((16, 8, 3), dtype= np.uint8)
10
11 # Se dibuja un rectangulo en el canal B con un nivel de 125 desde la fila 1 hasta la 2 y
12 # desde la columna 1 hasta la 6
13 imagen[1:3, 1:7, 0] = 125
14 # Se dibuja un rectangulo en el canal G con un nivel de 225 desde la fila 1 hasta la 2 y
15 # desde la columna 1 hasta la 6
16 imagen[1:3, 1:7, 1] = 200
17
18 # Se dibuja un rectangulo en el canal B con un nivel de 100 desde la fila 4 hasta la 10 y
19 # desde la columna 2 hasta la 5
20 imagen[4:11, 2:6, 0] = 100
21 # Se dibuja un rectangulo en el canal R con un nivel de 250 desde la fila 4 hasta la 10 y
22 # desde la columna 2 hasta la 5
23 imagen[4:11, 2:6, 2] = 250
24
25 # Se dibuja un rectangulo en el canal G con un nivel de 200 desde la fila 12 hasta la 14 y
26 # desde la columna 1 hasta la 6
27 imagen[12:15, 1:7, 1] = 200
28 # Se dibuja un rectangulo en el canal R con un nivel de 190 desde la fila 12 hasta la 14 y
29 # desde la columna 1 hasta la 6
30 imagen[12:15, 1:7, 2] = 190
```

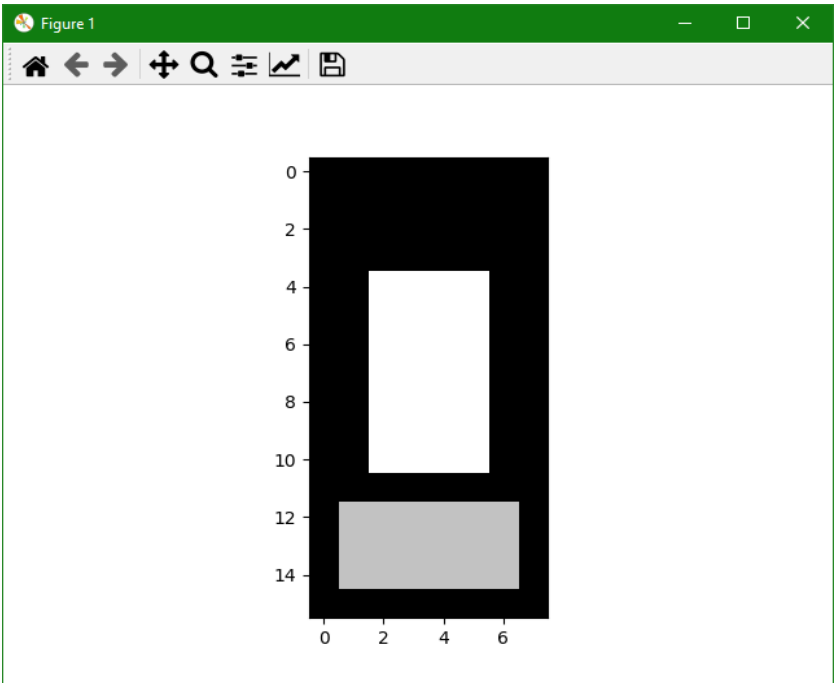


```
31
32 # Se genera una matriz para crear una escala de grises desde el canal R
33 R = imagen[:, :, 2]
34 # Se genera una matriz para crear una escala de grises desde el canal G
35 G = imagen[:, :, 1]
36 # Se genera una matriz para crear una escala de grises desde el canal B
37 B = imagen[:, :, 0]
38
39 # Se crea una escala de grises con el promedio de los canales
40 promedio = R*0.33 + G*0.33 + B*0.33
41 # Se convierte el promedio a un tipo de dato numpy.uint8
42 promedio = promedio.astype(np.uint8)
43 # Se crea una escala de grises con la ponderacion de los canales BT.601
44 bt_601 = R*0.299 + G*0.587 + B*0.114
45 # Se convierte la ponderacion a un tipo de dato numpy.uint8
46 bt_601 = bt_601.astype(np.uint8)
47 # Se crea una escala de grises con la ponderacion de los canales BT.709
48 bt_709 = R*0.2126 + G*0.7152 + B*0.0722
49 # Se convierte la ponderacion a un tipo de dato numpy.uint8
50 bt_709 = bt_709.astype(np.uint8)
51
52 # Se agrega la imagen con el filtro al pyplot
53 plt.imshow(R, cmap="gray")
54
55 # Se muestra el pyplot
56 plt.show()
57
```


Imagen original



Se muestra una imagen de lo que realiza el código



Método RGB (G)

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np
# Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
import matplotlib.pyplot as plt

# Se crea una imagen RGB de tamaño 16x8
imagen = np.zeros((16, 8, 3), dtype= np.uint8)

# Se dibuja un rectangulo en el canal B con un nivel de 125 desde la fila 1 hasta la 2
# desde la columna 1 hasta la 6
imagen[1:3, 1:7, 0] = 125
# Se dibuja un rectangulo en el canal G con un nivel de 225 desde la fila 1 hasta la 2 y
# desde la columna 1 hasta la 6
imagen[1:3, 1:7, 1] = 225

# Se dibuja un rectangulo en el canal B con un nivel de 100 desde la fila 4 hasta la 10 y
# desde la columna 2 hasta la 5
imagen[4:11, 2:6, 0] = 100
# Se dibuja un rectangulo en el canal R con un nivel de 250 desde la fila 4 hasta la 10 y
# desde la columna 2 hasta la 5
imagen[4:11, 2:6, 2] = 250

# Se dibuja un rectangulo en el canal G con un nivel de 200 desde la fila 12 hasta la 14
y
# desde la columna 1 hasta la 6
imagen[12:15, 1:7, 1] = 200
# Se dibuja un rectangulo en el canal R con un nivel de 190 desde la fila 12 hasta la 14 y
# desde la columna 1 hasta la 6
imagen[12:15, 1:7, 2] = 190

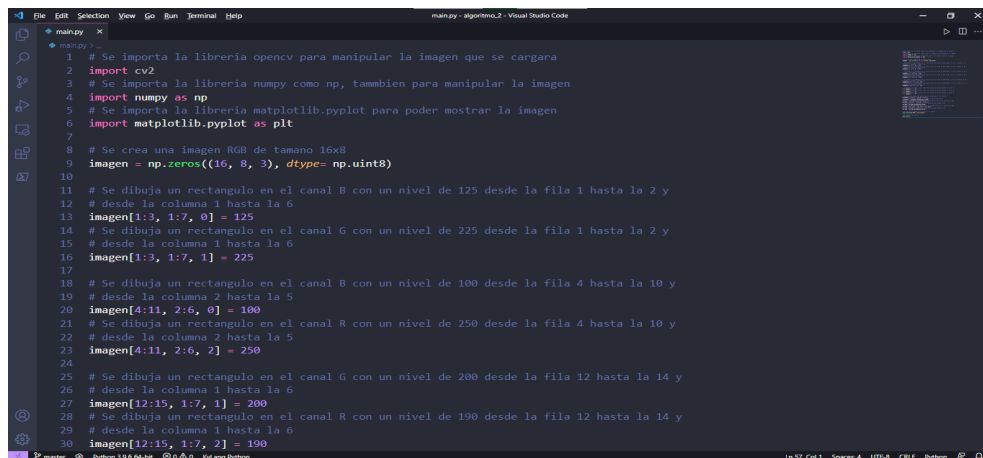
# Se genera una matriz para crear una escala de grises desde el canal G
G = imagen[:, :, 1]
```

```
# Se agrega la imagen con el filtro al pyplot
plt.imshow(G, cmap="gray")
```

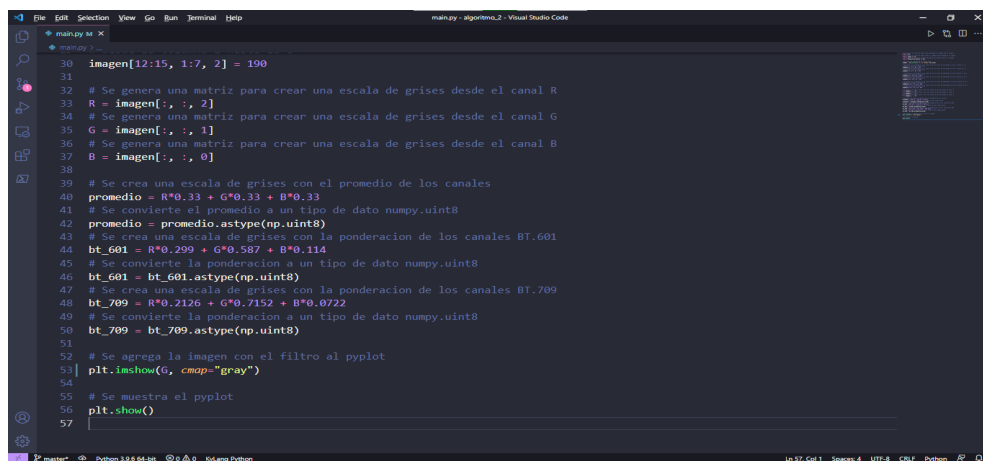
```
# Se muestra el pyplot
plt.show()
```

Para el código anterior, primero se importan las librerías necesarias para realizarlo (opencv, numpy, matplotlib). Después, con *numpy* se genera la base de la imagen que se hizo en el algoritmo 1, se agregan los rectángulos y se declara la variable G, luego esta se agrega al *pyplot* y al final se muestra la imagen con el *pyplot*.

Se muestran imágenes del código

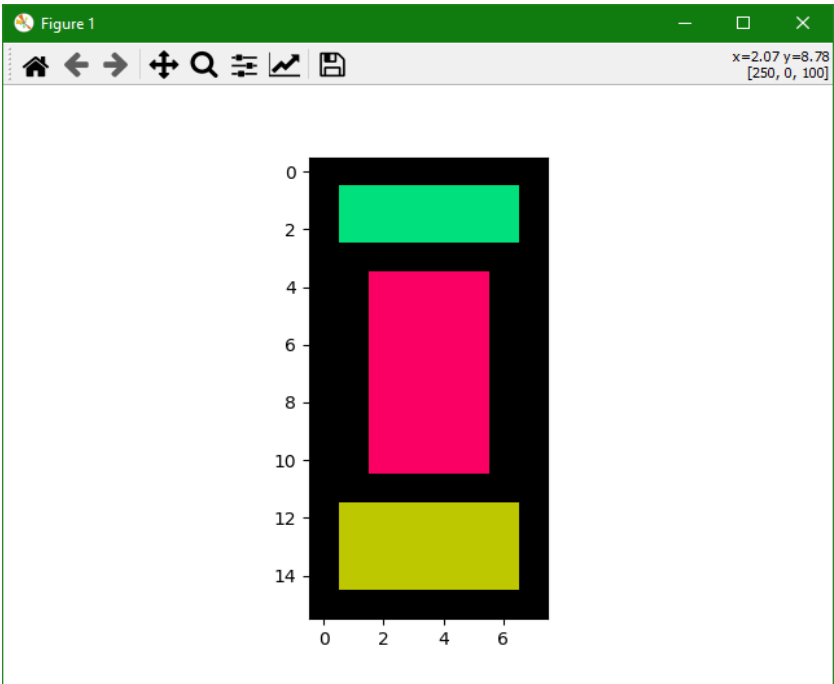


```
1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5 # Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
6 import matplotlib.pyplot as plt
7
8 # Se crea una imagen RGB de tamaño 16x8
9 imagen = np.zeros((16, 8, 3), dtype= np.uint8)
10
11 # Se dibuja un rectangulo en el canal B con un nivel de 125 desde la fila 1 hasta la 2 y
12 # desde la columna 1 hasta la 6
13 imagen[1:3, 1:7, 0] = 125
14 # Se dibuja un rectangulo en el canal G con un nivel de 225 desde la fila 1 hasta la 2 y
15 # desde la columna 1 hasta la 6
16 imagen[1:3, 1:7, 1] = 225
17
18 # Se dibuja un rectangulo en el canal B con un nivel de 100 desde la fila 4 hasta la 10 y
19 # desde la columna 2 hasta la 5
20 imagen[4:11, 2:6, 0] = 100
21 # Se dibuja un rectangulo en el canal R con un nivel de 250 desde la fila 4 hasta la 10 y
22 # desde la columna 2 hasta la 5
23 imagen[4:11, 2:6, 2] = 250
24
25 # Se dibuja un rectangulo en el canal G con un nivel de 200 desde la fila 12 hasta la 14 y
26 # desde la columna 1 hasta la 6
27 imagen[12:15, 1:7, 1] = 200
28 # Se dibuja un rectangulo en el canal R con un nivel de 190 desde la fila 12 hasta la 14 y
29 # desde la columna 1 hasta la 6
30 imagen[12:15, 1:7, 2] = 190
```

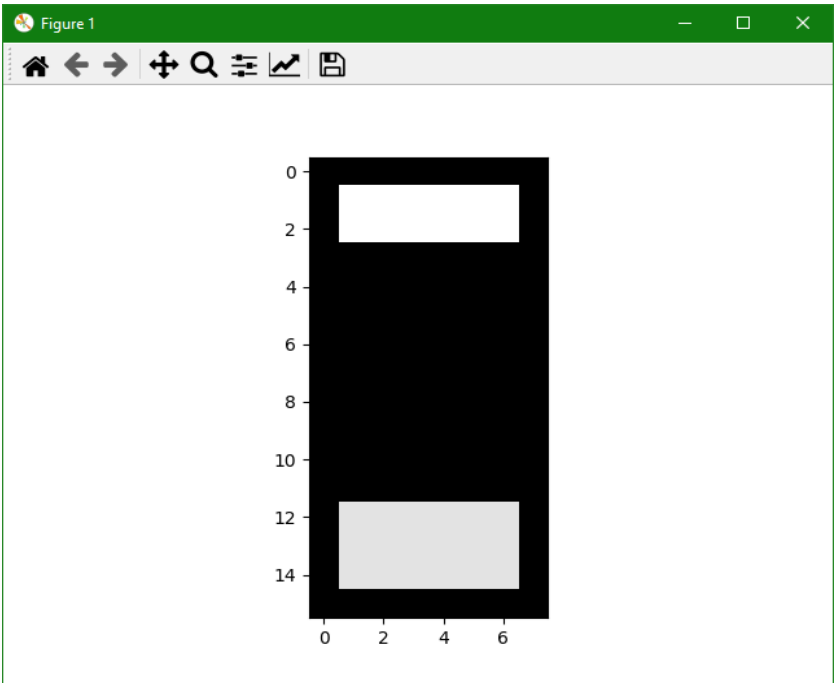


```
31
32 # Se genera una matriz para crear una escala de grises desde el canal R
33 R = imagen[:, :, 2]
34 # Se genera una matriz para crear una escala de grises desde el canal G
35 G = imagen[:, :, 1]
36 # Se genera una matriz para crear una escala de grises desde el canal B
37 B = imagen[:, :, 0]
38
39 # Se crea una escala de grises con el promedio de los canales
40 promedio = R*0.33 + G*0.33 + B*0.33
41 # Se convierte el promedio a un tipo de dato numpy.uint8
42 promedio = promedio.astype(np.uint8)
43 # Se crea una escala de grises con la ponderacion de los canales BT.601
44 bt_601 = R*0.299 + G*0.587 + B*0.114
45 # Se convierte la ponderacion a un tipo de dato numpy.uint8
46 bt_601 = bt_601.astype(np.uint8)
47 # Se crea una escala de grises con la ponderacion de los canales BT.709
48 bt_709 = R*0.2126 + G*0.7152 + B*0.0722
49 # Se convierte la ponderacion a un tipo de dato numpy.uint8
50 bt_709 = bt_709.astype(np.uint8)
51
52 # Se agrega la imagen con el filtro al pyplot
53 plt.imshow(G, cmap="gray")
54
55 # Se muestra el pyplot
56 plt.show()
57
```

Imagen original



Se muestra una imagen de lo que el código realiza



Método RGB (B)

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np
# Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
import matplotlib.pyplot as plt

# Se crea una imagen RGB de tamaño 16x8
imagen = np.zeros((16, 8, 3), dtype= np.uint8)

# Se dibuja un rectangulo en el canal B con un nivel de 125 desde la fila 1 hasta la 2
# desde la columna 1 hasta la 6
imagen[1:3, 1:7, 0] = 125
# Se dibuja un rectangulo en el canal G con un nivel de 225 desde la fila 1 hasta la 2 y
# desde la columna 1 hasta la 6
imagen[1:3, 1:7, 1] = 225

# Se dibuja un rectangulo en el canal B con un nivel de 100 desde la fila 4 hasta la 10 y
# desde la columna 2 hasta la 5
imagen[4:11, 2:6, 0] = 100
# Se dibuja un rectangulo en el canal R con un nivel de 250 desde la fila 4 hasta la 10 y
# desde la columna 2 hasta la 5
imagen[4:11, 2:6, 2] = 250

# Se dibuja un rectangulo en el canal G con un nivel de 200 desde la fila 12 hasta la 14
y
# desde la columna 1 hasta la 6
imagen[12:15, 1:7, 1] = 200
# Se dibuja un rectangulo en el canal R con un nivel de 190 desde la fila 12 hasta la 14 y
# desde la columna 1 hasta la 6
imagen[12:15, 1:7, 2] = 190

# Se genera una matriz para crear una escala de grises desde el canal B
B = imagen[:, :, 0]
```

```
# Se agrega la imagen con el filtro al pyplot
plt.imshow(B, cmap="gray")
```

```
# Se muestra el pyplot
plt.show()
```

Para el código anterior, primero se importan las librerías necesarias para realizarlo (opencv, numpy, matplotlib). Después, con *numpy* se genera la base de la imagen que se hizo en el algoritmo 1, se agregan los rectángulos y se declara la variable B, luego esta se agrega al *pyplot* y al final se muestra la imagen con el *pyplot*.

Se muestran imágenes del código

```

1 # Se importa la libreria opencv para manipular la imagen que se cargara
2 import cv2
3 # Se importa la libreria numpy como np, tambien para manipular la imagen
4 import numpy as np
5 # Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
6 import matplotlib.pyplot as plt
7
8 # Se crea una imagen RGB de tamaño 16x8
9 imagen = np.zeros((16, 8, 3), dtype= np.uint8)
10
11 # Se dibuja un rectangulo en el canal B con un nivel de 125 desde la fila 1 hasta la 2 y
12 # desde la columna 1 hasta la 6
13 imagen[1:3, 1:7, 0] = 125
14 # Se dibuja un rectangulo en el canal G con un nivel de 225 desde la fila 1 hasta la 2 y
15 # desde la columna 1 hasta la 6
16 imagen[1:3, 1:7, 1] = 225
17
18 # Se dibuja un rectangulo en el canal B con un nivel de 100 desde la fila 4 hasta la 10 y
19 # desde la columna 2 hasta la 5
20 imagen[4:11, 2:6, 0] = 100
21 # Se dibuja un rectangulo en el canal R con un nivel de 250 desde la fila 4 hasta la 10 y
22 # desde la columna 2 hasta la 5
23 imagen[4:11, 2:6, 2] = 250
24
25 # Se dibuja un rectangulo en el canal G con un nivel de 200 desde la fila 12 hasta la 14 y
26 # desde la columna 1 hasta la 6
27 imagen[12:15, 1:7, 1] = 200
28 # Se dibuja un rectangulo en el canal R con un nivel de 190 desde la fila 12 hasta la 14 y
29 # desde la columna 1 hasta la 6
30 imagen[12:15, 1:7, 2] = 190

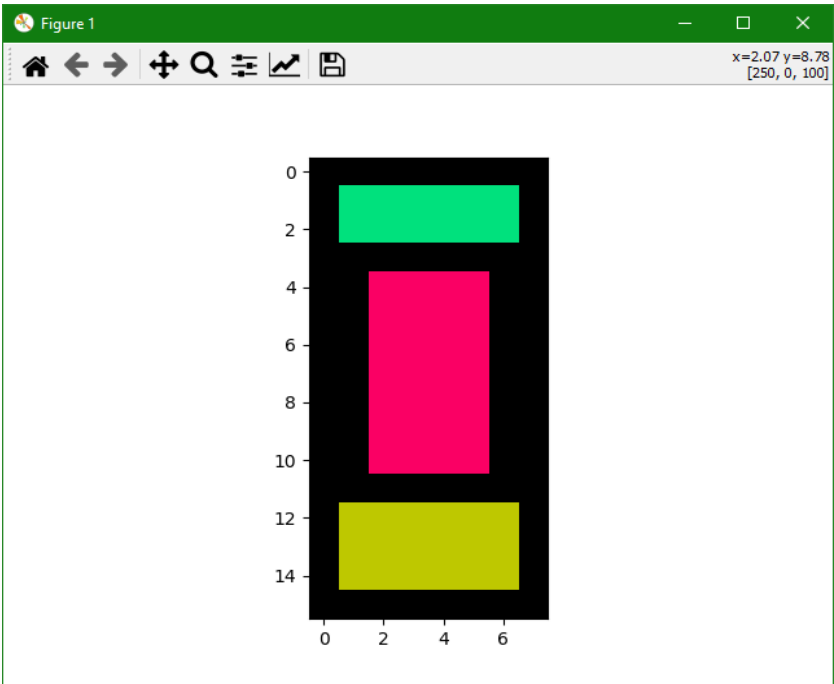
```

```

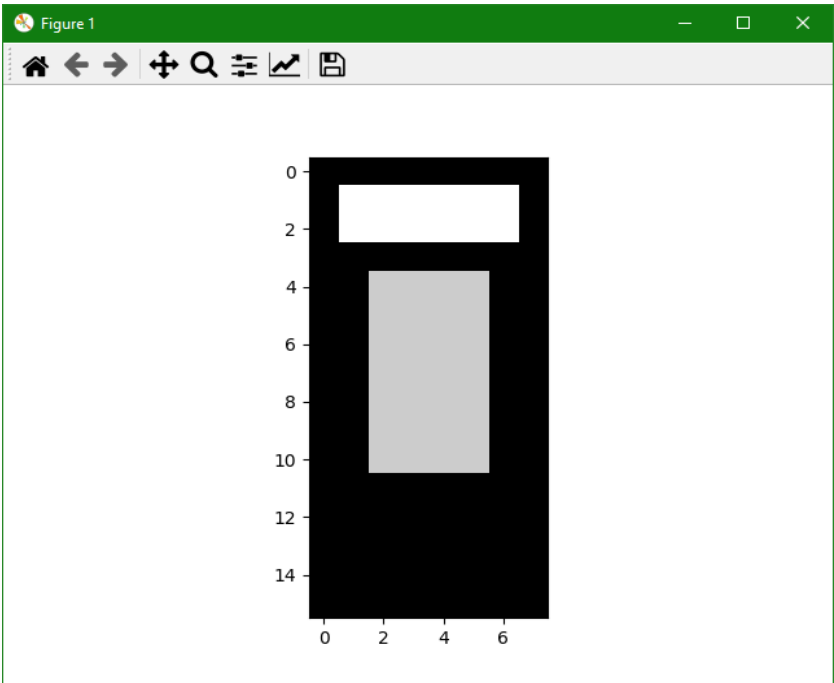
31
32 # Se genera una matriz para crear una escala de grises desde el canal R
33 R = imagen[:, :, 2]
34 # Se genera una matriz para crear una escala de grises desde el canal G
35 G = imagen[:, :, 1]
36 # Se genera una matriz para crear una escala de grises desde el canal B
37 B = imagen[:, :, 0]
38
39 # Se crea una escala de grises con el promedio de los canales
40 promedio = R*0.33 + G*0.33 + B*0.33
41 # Se convierte el promedio a un tipo de dato numpy.uint8
42 promedio = promedio.astype(np.uint8)
43 # Se crea una escala de grises con la ponderacion de los canales BT.601
44 bt_601 = R*0.299 + G*0.587 + B*0.114
45 # Se convierte la ponderacion a un tipo de dato numpy.uint8
46 bt_601 = bt_601.astype(np.uint8)
47 # Se crea una escala de grises con la ponderacion de los canales BT.709
48 bt_709 = R*0.2126 + G*0.7152 + B*0.0722
49 # Se convierte la ponderacion a un tipo de dato numpy.uint8
50 bt_709 = bt_709.astype(np.uint8)
51
52 # Se agrega la imagen con el filtro al pyplot
53 plt.imshow(B, cmap="gray")
54
55 # Se muestra el pyplot
56 plt.show()
57

```

Imagen original



Se muestra una imagen de lo que realiza el código



Método promedio

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np
# Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
import matplotlib.pyplot as plt

# Se crea una imagen RGB de tamaño 16x8
imagen = np.zeros((16, 8, 3), dtype= np.uint8)

# Se dibuja un rectangulo en el canal B con un nivel de 125 desde la fila 1 hasta la 2
# desde la columna 1 hasta la 6
imagen[1:3, 1:7, 0] = 125
# Se dibuja un rectangulo en el canal G con un nivel de 225 desde la fila 1 hasta la 2 y
# desde la columna 1 hasta la 6
imagen[1:3, 1:7, 1] = 225

# Se dibuja un rectangulo en el canal B con un nivel de 100 desde la fila 4 hasta la 10 y
# desde la columna 2 hasta la 5
imagen[4:11, 2:6, 0] = 100
# Se dibuja un rectangulo en el canal R con un nivel de 250 desde la fila 4 hasta la 10 y
# desde la columna 2 hasta la 5
imagen[4:11, 2:6, 2] = 250

# Se dibuja un rectangulo en el canal G con un nivel de 200 desde la fila 12 hasta la 14
y
# desde la columna 1 hasta la 6
imagen[12:15, 1:7, 1] = 200
# Se dibuja un rectangulo en el canal R con un nivel de 190 desde la fila 12 hasta la 14 y
# desde la columna 1 hasta la 6
imagen[12:15, 1:7, 2] = 190

# Se genera una matriz para crear una escala de grises desde el canal R
R = imagen[:, :, 2]
```



```
# Se genera una matriz para crear una escala de grises desde el canal G
G = imagen[:, :, 1]
# Se genera una matriz para crear una escala de grises desde el canal B
B = imagen[:, :, 0]
```

```
# Se crea una escala de grises con el promedio de los canales
promedio = R*0.33 + G*0.33 + B*0.33
# Se convierte el promedio a un tipo de dato numpy.uint8
promedio = promedio.astype(np.uint8)
```

```
# Se agrega la imagen con el filtro al pyplot
plt.imshow(promedio, cmap="gray")
```

```
# Se muestra el pyplot
plt.show()
```

Para el código anterior, primero se importan las librerías necesarias para relizarlo (opencv, numpy, matplotlib). Después, con **numpy** se genera la base de la imagen que se hizo en el algoritmo 1, se agregan los rectangulos y se declaran las variables R, G y B, con estas se genera el promedio RGB y se agrega al **pyplot**. Al final se muestra la imagen en escala de grises con el **pyplot**.

Imágenes del código

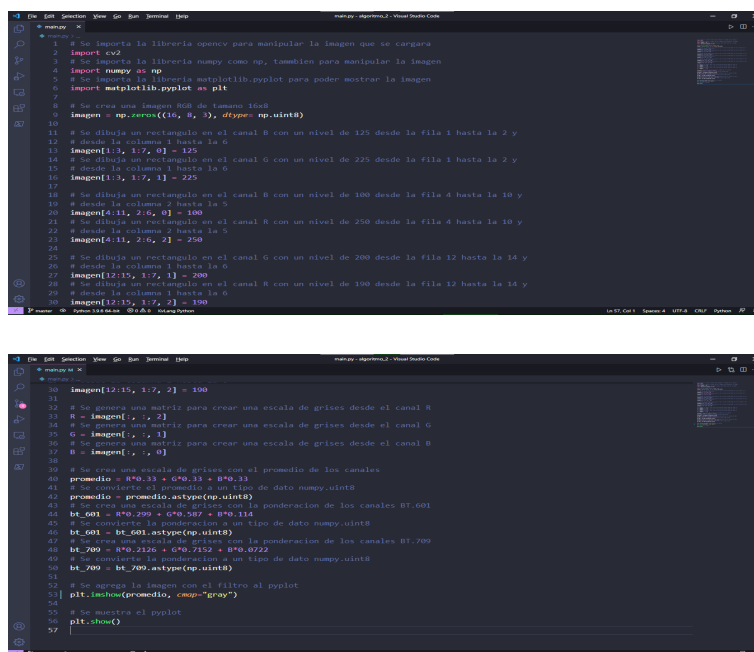
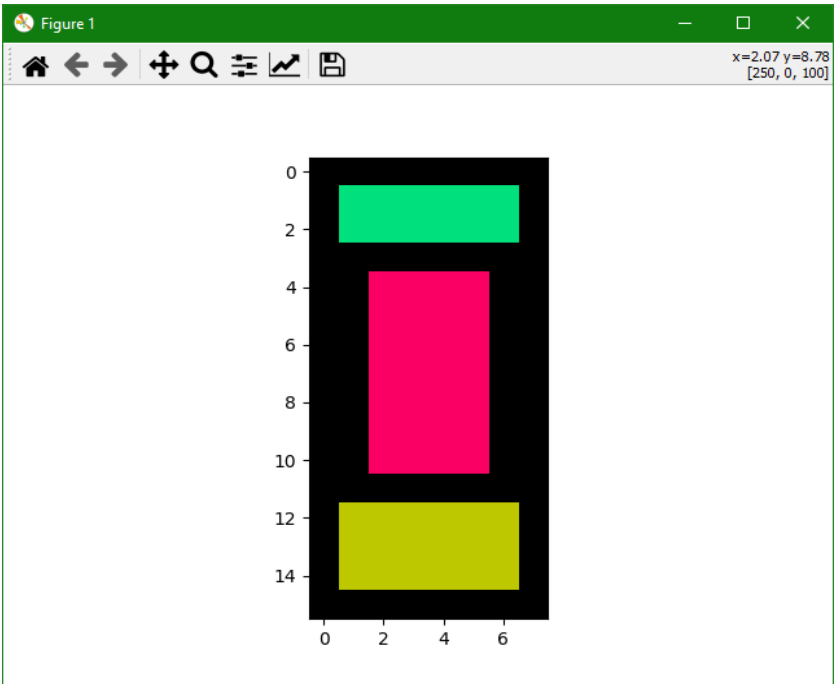
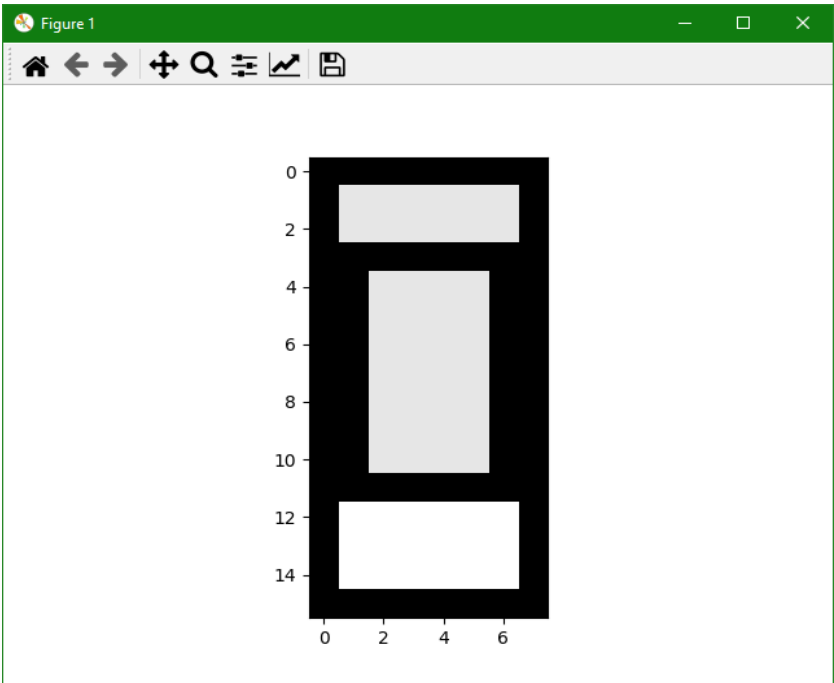


Imagen original



Se muestra una imagen de lo que el código realiza



Método BT.601

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np
# Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
import matplotlib.pyplot as plt

# Se crea una imagen RGB de tamaño 16x8
imagen = np.zeros((16, 8, 3), dtype= np.uint8)

# Se dibuja un rectangulo en el canal B con un nivel de 125 desde la fila 1 hasta la 2
# desde la columna 1 hasta la 6
imagen[1:3, 1:7, 0] = 125
# Se dibuja un rectangulo en el canal G con un nivel de 225 desde la fila 1 hasta la 2 y
# desde la columna 1 hasta la 6
imagen[1:3, 1:7, 1] = 225

# Se dibuja un rectangulo en el canal B con un nivel de 100 desde la fila 4 hasta la 10 y
# desde la columna 2 hasta la 5
imagen[4:11, 2:6, 0] = 100
# Se dibuja un rectangulo en el canal R con un nivel de 250 desde la fila 4 hasta la 10 y
# desde la columna 2 hasta la 5
imagen[4:11, 2:6, 2] = 250

# Se dibuja un rectangulo en el canal G con un nivel de 200 desde la fila 12 hasta la 14
y
# desde la columna 1 hasta la 6
imagen[12:15, 1:7, 1] = 200
# Se dibuja un rectangulo en el canal R con un nivel de 190 desde la fila 12 hasta la 14 y
# desde la columna 1 hasta la 6
imagen[12:15, 1:7, 2] = 190

# Se genera una matriz para crear una escala de grises desde el canal R
R = imagen[:, :, 2]
```

```
# Se genera una matriz para crear una escala de grises desde el canal G
G = imagen[:, :, 1]
# Se genera una matriz para crear una escala de grises desde el canal B
B = imagen[:, :, 0]
```

```
# Se crea una escala de grises con la ponderacion de los canales BT.601
bt_601 = R*0.299 + G*0.587 + B*0.114
# Se convierte la ponderacion a un tipo de dato numpy.uint8
bt_601 = bt_601.astype(np.uint8)
```

```
# Se agrega la imagen con el filtro al pyplot
plt.imshow(bt_601, cmap="gray")
```

```
# Se muestra el pyplot
plt.show()
```

Para el código anterior, primero se importan las librerías necesarias para realizarlo (opencv, numpy, matplotlib). Después, con **numpy** se genera la base de la imagen que se hizo en el algoritmo 1, se agregan los rectángulos y se declaran las variables R, G y B, con estas se genera la ponderación BT.601 y se agrega al **pyplot**. Al final se muestra la imagen en escala de grises en el **pyplot**.

Imágenes del código

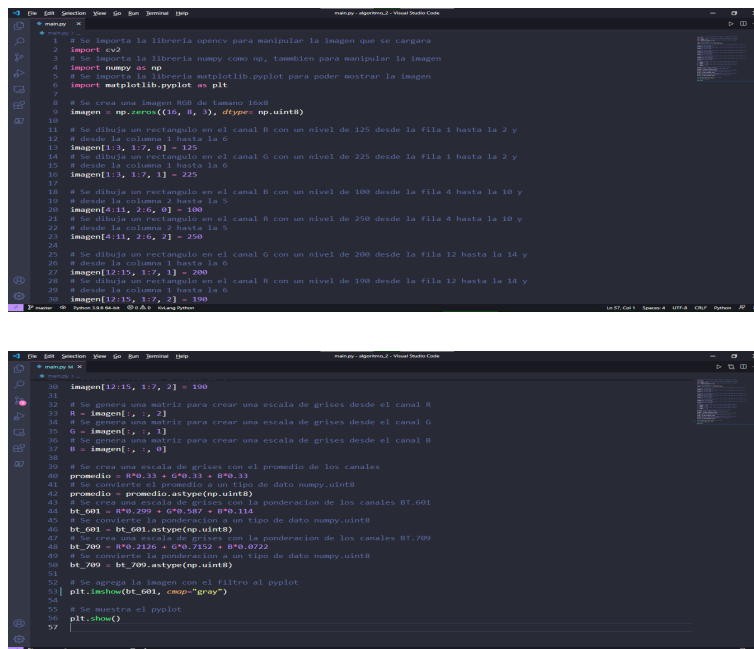
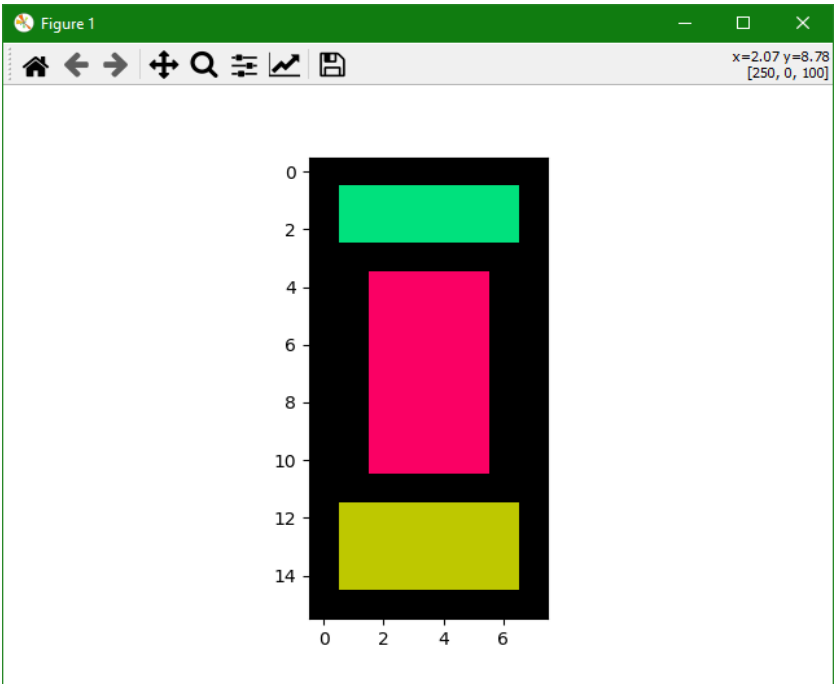
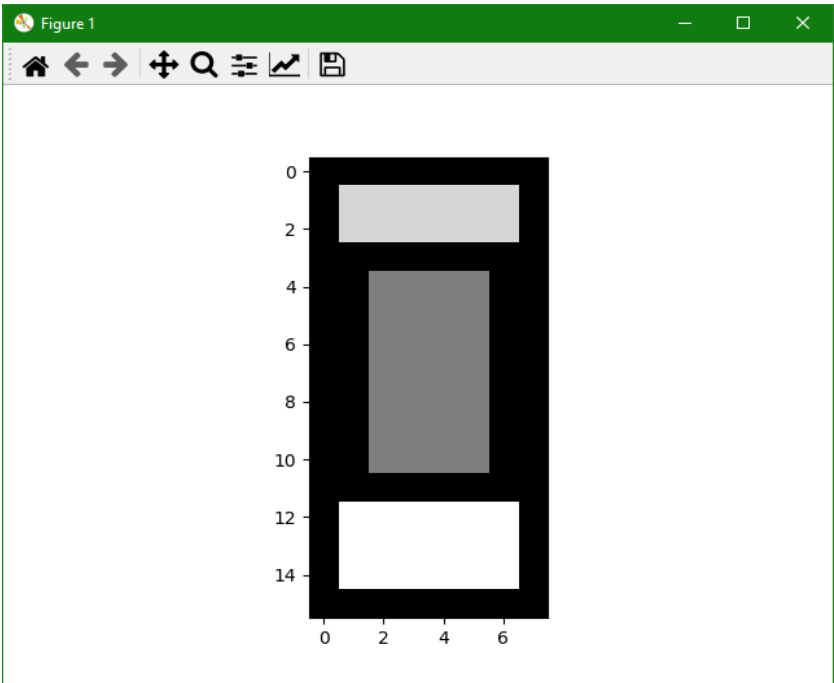


Imagen original



Se muestra una imagen de lo que realiza el código



Método BT.709

```
# Se importa la libreria opencv para manipular la imagen que se cargara
import cv2
# Se importa la libreria numpy como np, tambien para manipular la imagen
import numpy as np
# Se importa la libreria matplotlib.pyplot para poder mostrar la imagen
import matplotlib.pyplot as plt

# Se crea una imagen RGB de tamaño 16x8
imagen = np.zeros((16, 8, 3), dtype= np.uint8)

# Se dibuja un rectangulo en el canal B con un nivel de 125 desde la fila 1 hasta la 2
# desde la columna 1 hasta la 6
imagen[1:3, 1:7, 0] = 125
# Se dibuja un rectangulo en el canal G con un nivel de 225 desde la fila 1 hasta la 2 y
# desde la columna 1 hasta la 6
imagen[1:3, 1:7, 1] = 225

# Se dibuja un rectangulo en el canal B con un nivel de 100 desde la fila 4 hasta la 10 y
# desde la columna 2 hasta la 5
imagen[4:11, 2:6, 0] = 100
# Se dibuja un rectangulo en el canal R con un nivel de 250 desde la fila 4 hasta la 10 y
# desde la columna 2 hasta la 5
imagen[4:11, 2:6, 2] = 250

# Se dibuja un rectangulo en el canal G con un nivel de 200 desde la fila 12 hasta la 14
y
# desde la columna 1 hasta la 6
imagen[12:15, 1:7, 1] = 200
# Se dibuja un rectangulo en el canal R con un nivel de 190 desde la fila 12 hasta la 14 y
# desde la columna 1 hasta la 6
imagen[12:15, 1:7, 2] = 190

# Se genera una matriz para crear una escala de grises desde el canal R
R = imagen[:, :, 2]
```

```
# Se genera una matriz para crear una escala de grises desde el canal G
G = imagen[:, :, 1]
# Se genera una matriz para crear una escala de grises desde el canal B
B = imagen[:, :, 0]
```

```
# Se crea una escala de grises con la ponderacion de los canales BT.709
bt_709 = R*0.2126 + G*0.7152 + B*0.0722
# Se convierte la ponderacion a un tipo de dato numpy.uint8
bt_709 = bt_709.astype(np.uint8)
```

```
# Se agrega la imagen con el filtro al pyplot
plt.imshow(bt_709, cmap="gray")
```

```
# Se muestra el pyplot
plt.show()
```

Para el código anterior, primero se importan las librerías necesarias para realizarlo (opencv, numpy, matplotlib). Después, con **numpy** se genera la base de la imagen que se hizo en el algoritmo 1, se agregan los rectángulos y se declaran las variables R, G y B, con estas se genera la ponderación BT.709 y se agrega al **pyplot**. Al final se muestra la imagen en escala de grises con el **pyplot**.

Imágenes del código

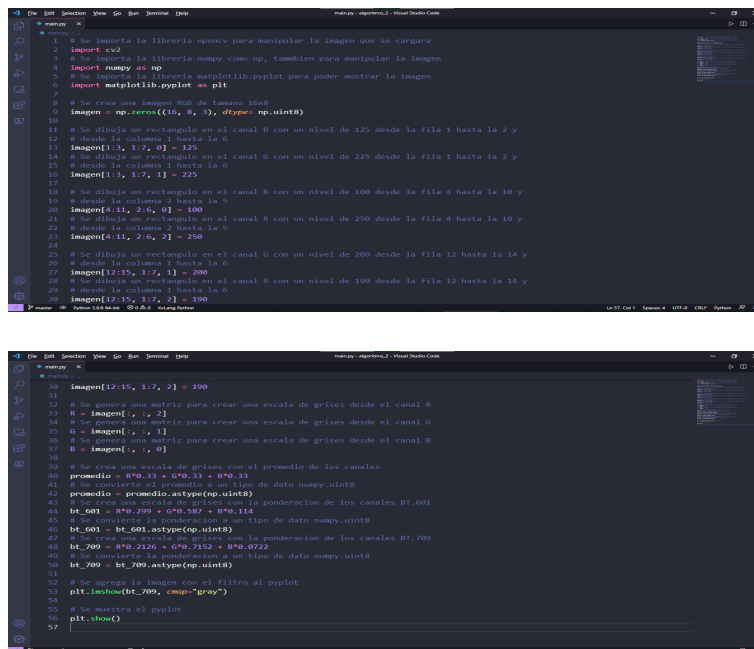
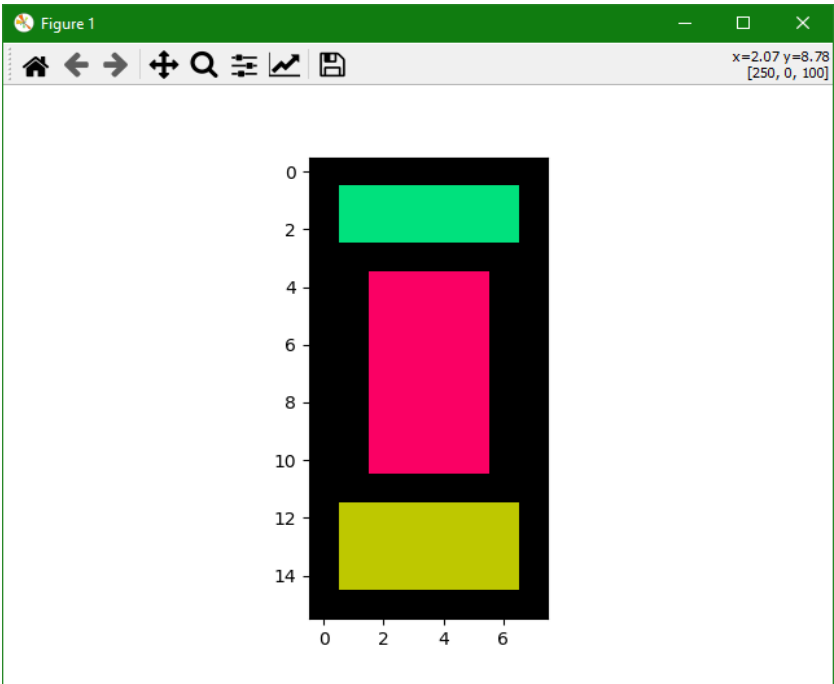


Imagen original



Se muestra una imagen de lo que el código realiza

