

Universidad de Guadalajara



Algoritmo 15

Muñoz Nuñez Ian Emmanuel

Visión Robótica

*"Filtrado de imágenes"*

# Índice general

<b>1. Ejercicio a)</b>	<b>3</b>
1.1. Código . . . . .	3
1.2. Capturas del código del ejercicio . . . . .	4
1.3. Imágenes obtenidas . . . . .	6
<b>2. Ejercicio b)</b>	<b>7</b>
2.1. Código . . . . .	7
2.2. Capturas del código del ejercicio . . . . .	8
2.3. Imágenes obtenidas . . . . .	9
<b>3. Ejercicio c)</b>	<b>10</b>
3.1. Código . . . . .	10
3.2. Capturas del código del ejercicio . . . . .	11
3.3. Imágenes obtenidas . . . . .	12
<b>4. Ejercicio d)</b>	<b>13</b>
4.1. Código . . . . .	13
4.2. Capturas del código del ejercicio . . . . .	14
4.3. Imágenes obtenidas . . . . .	16
<b>5. Ejercicio e)</b>	<b>17</b>
5.1. Código . . . . .	17
5.2. Capturas del código del ejercicio . . . . .	18
5.3. Imágenes obtenidas . . . . .	19

# Índice de figuras

1.1. Captura 1 del código del ejercicio . . . . .	4
1.2. Captura 2 del código del ejercicio . . . . .	5
1.3. Imagen original . . . . .	6
1.4. Imagen obtenida en el ejercicio . . . . .	6
2.1. Captura 1 del código del ejercicio . . . . .	8
2.2. Captura 2 del código del ejercicio . . . . .	8
2.3. Imagen original . . . . .	9
2.4. Imagen obtenida en el ejercicio . . . . .	9
3.1. Captura 1 del código del ejercicio . . . . .	11
3.2. Captura 2 del código del ejercicio . . . . .	11
3.3. Imagen original . . . . .	12
3.4. Imagen obtenida en el ejercicio . . . . .	12
4.1. Captura 1 del código del ejercicio . . . . .	14
4.2. Captura 2 del código del ejercicio . . . . .	15
4.3. Imagen original . . . . .	16
4.4. Imagen obtenida en el ejercicio . . . . .	16
5.1. Captura 1 del código del ejercicio . . . . .	18
5.2. Captura 2 del código del ejercicio . . . . .	19
5.3. Imagen original . . . . .	19
5.4. Imagen en escala de grises . . . . .	20
5.5. Imagen obtenida en el ejercicio . . . . .	20

# Capítulo 1

## Ejercicio a)

### 1.1. Código

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

# Se lee la imagen 'rubik.jpg'
imagen = cv2.imread('rubik.jpg')
# Se modifica el tamaño de la imagen
imagen = cv2.resize(imagen, None, fx=0.3, fy=0.3)

# Se crea el kernel o filtro de la imagen
kernel = np.array([
    [1/9, 1/9, 1/9],
    [1/9, 1/9, 1/9],
    [1/9, 1/9, 1/9]
], dtype=np.float32)

# Se aplica el filtro a la imagen
filtro = cv2.filter2D(imagen, -1, kernel)

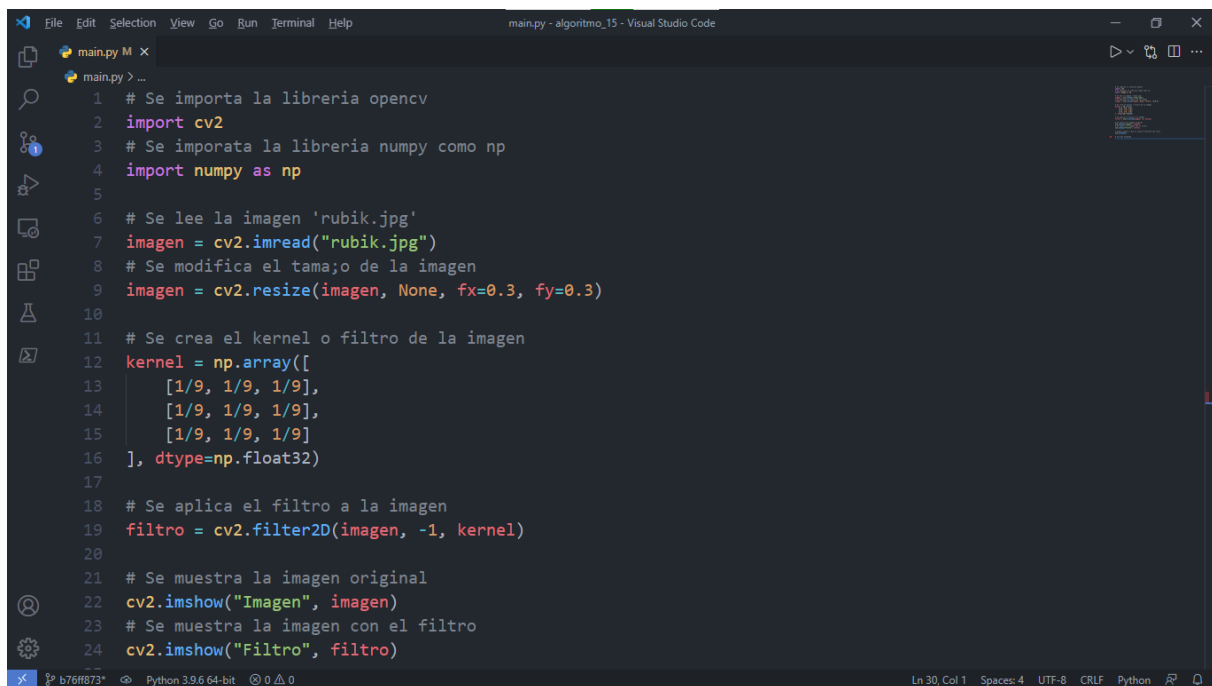
# Se muestra la imagen original
cv2.imshow('Imagen', imagen)
# Se muestra la imagen con el filtro
cv2.imshow('Filtro', filtro)

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

# Fin del programa
```

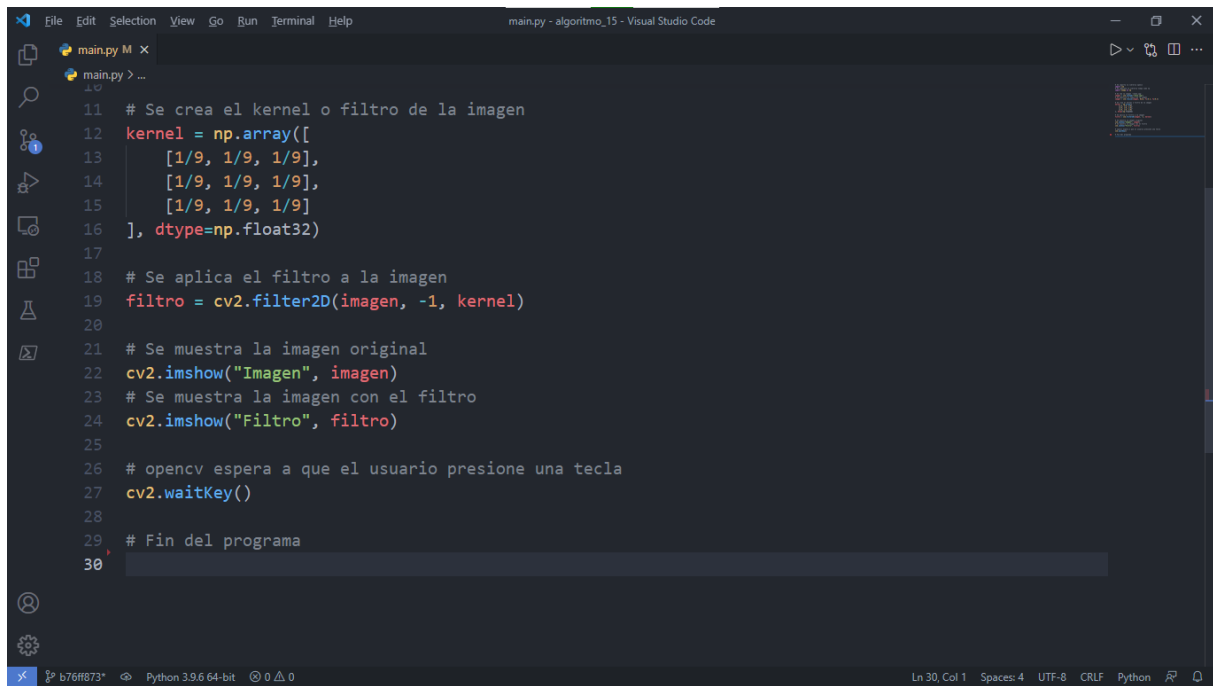
Para el código del ejercicio a) primero se importan las librerías necesarias ("**opencv**" y "**numpy**"). Después se carga la imagen '**rubik.jpg**' y se modifica su tamaño. Luego se genera el kernel y se aplica el filtro a la imagen. Por último se muestran la imagen original y la imagen con el filtro. Al final, "**opencv**" espera a que el usuario presione una tecla para terminar el programa.

## 1.2. Capturas del código del ejercicio



```
1 # Se importa la libreria opencv
2 import cv2
3 # Se importa la libreria numpy como np
4 import numpy as np
5
6 # Se lee la imagen 'rubik.jpg'
7 imagen = cv2.imread("rubik.jpg")
8 # Se modifica el tamaño de la imagen
9 imagen = cv2.resize(imagen, None, fx=0.3, fy=0.3)
10
11 # Se crea el kernel o filtro de la imagen
12 kernel = np.array([
13     [1/9, 1/9, 1/9],
14     [1/9, 1/9, 1/9],
15     [1/9, 1/9, 1/9]
16 ], dtype=np.float32)
17
18 # Se aplica el filtro a la imagen
19 filtro = cv2.filter2D(imagen, -1, kernel)
20
21 # Se muestra la imagen original
22 cv2.imshow("Imagen", imagen)
23 # Se muestra la imagen con el filtro
24 cv2.imshow("Filtro", filtro)
```

Figura 1.1: Captura 1 del código del ejercicio



```
11 # Se crea el kernel o filtro de la imagen
12 kernel = np.array([
13     [1/9, 1/9, 1/9],
14     [1/9, 1/9, 1/9],
15     [1/9, 1/9, 1/9]
16 ], dtype=np.float32)
17
18 # Se aplica el filtro a la imagen
19 filtro = cv2.filter2D(imagen, -1, kernel)
20
21 # Se muestra la imagen original
22 cv2.imshow("Imagen", imagen)
23 # Se muestra la imagen con el filtro
24 cv2.imshow("Filtro", filtro)
25
26 # opencv espera a que el usuario presione una tecla
27 cv2.waitKey()
28
29 # Fin del programa
30
```

Figura 1.2: Captura 2 del código del ejercicio

### 1.3. Imágenes obtenidas



Figura 1.3: Imagen original



Figura 1.4: Imagen obtenida en el ejercicio

# Capítulo 2

## Ejercicio b)

### 2.1. Código

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

# Se lee la imagen 'rubik.jpg'
imagen = cv2.imread('rubik.jpg')
# Se modifica el tamaño de la imagen
imagen = cv2.resize(imagen, None, fx=0.3, fy=0.3)

# Se crea el kernel o filtro de la imagen
kernel = np.ones((5, 5), dtype=np.float32)*1/25

# Se aplica el filtro a la imagen
filtro = cv2.filter2D(imagen, -1, kernel)

# Se muestra la imagen original
cv2.imshow('Imagen', imagen)
# Se muestra la imagen con el filtro
cv2.imshow('Filtro', filtro)

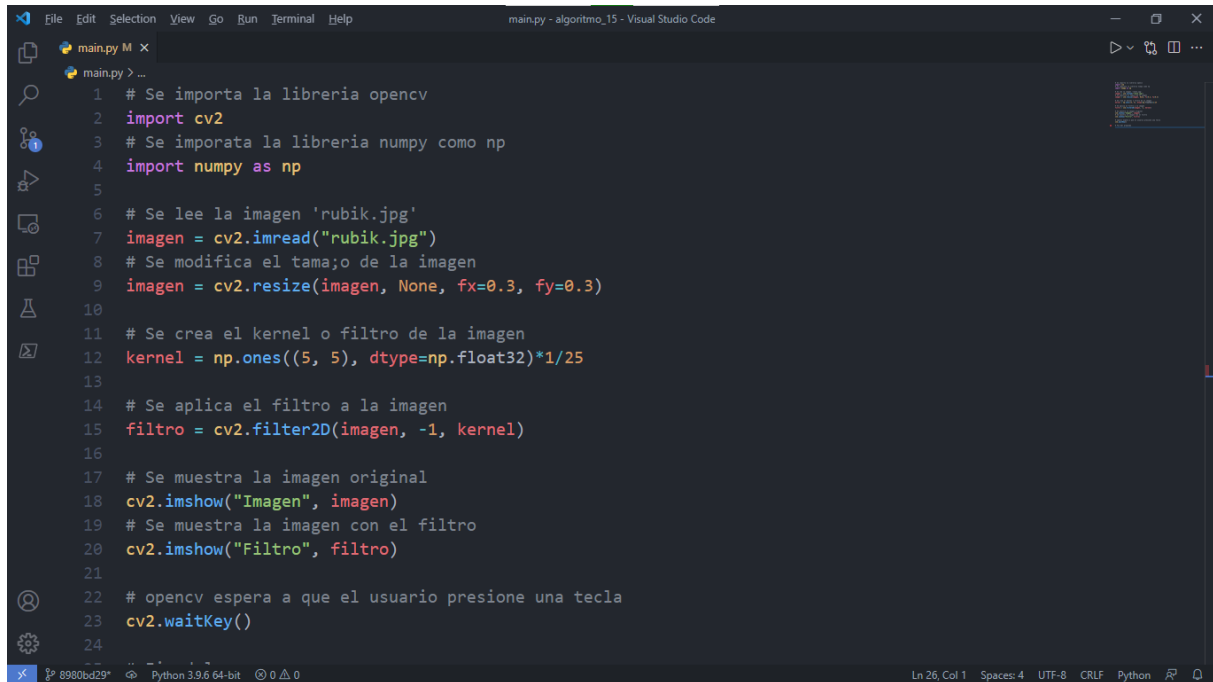
# opencv espera a que el usuario presione una tecla
cv2.waitKey()

# Fin del programa
```

Para el código del ejercicio b) primero se importan las librerías necesarias ("*opencv*" y "*numpy*"). Después se carga la imagen '*rubik.jpg*' y se modifica su tamaño. Luego se genera el kernel y se aplica el filtro a la imagen. Por último se muestran la imagen original y la imagen con el filtro. Al final, "*opencv*" espera a que el usuario presione una tecla para terminar el programa.

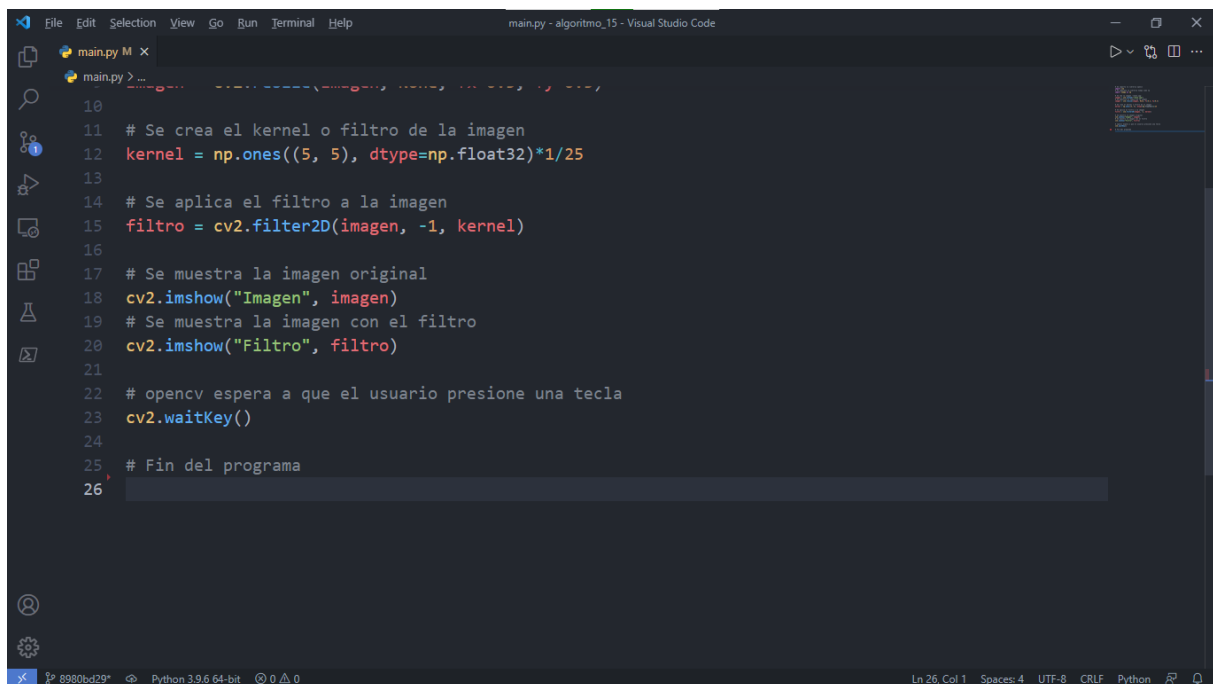


## 2.2. Capturas del código del ejercicio



```
1 # Se importa la libreria opencv
2 import cv2
3 # Se importa la libreria numpy como np
4 import numpy as np
5
6 # Se lee la imagen 'rubik.jpg'
7 imagen = cv2.imread("rubik.jpg")
8 # Se modifica el tamaño de la imagen
9 imagen = cv2.resize(imagen, None, fx=0.3, fy=0.3)
10
11 # Se crea el kernel o filtro de la imagen
12 kernel = np.ones((5, 5), dtype=np.float32)*1/25
13
14 # Se aplica el filtro a la imagen
15 filtro = cv2.filter2D(imagen, -1, kernel)
16
17 # Se muestra la imagen original
18 cv2.imshow("Imagen", imagen)
19 # Se muestra la imagen con el filtro
20 cv2.imshow("Filtro", filtro)
21
22 # opencv espera a que el usuario presione una tecla
23 cv2.waitKey()
24
```

Figura 2.1: Captura 1 del código del ejercicio



```
10
11 # Se crea el kernel o filtro de la imagen
12 kernel = np.ones((5, 5), dtype=np.float32)*1/25
13
14 # Se aplica el filtro a la imagen
15 filtro = cv2.filter2D(imagen, -1, kernel)
16
17 # Se muestra la imagen original
18 cv2.imshow("Imagen", imagen)
19 # Se muestra la imagen con el filtro
20 cv2.imshow("Filtro", filtro)
21
22 # opencv espera a que el usuario presione una tecla
23 cv2.waitKey()
24
25 # Fin del programa
26
```

Figura 2.2: Captura 2 del código del ejercicio

## 2.3. Imágenes obtenidas



Figura 2.3: Imagen original



Figura 2.4: Imagen obtenida en el ejercicio

# Capítulo 3

## Ejercicio c)

### 3.1. Código

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

# Se lee la imagen 'rubik.jpg'
imagen = cv2.imread('rubik.jpg')
# Se modifica el tamaño de la imagen
imagen = cv2.resize(imagen, None, fx=0.3, fy=0.3)

# Se crea el kernel o filtro de la imagen
kernel = np.ones((7, 7), dtype=np.float32)*1/49

# Se aplica el filtro a la imagen
filtro = cv2.filter2D(imagen, -1, kernel)

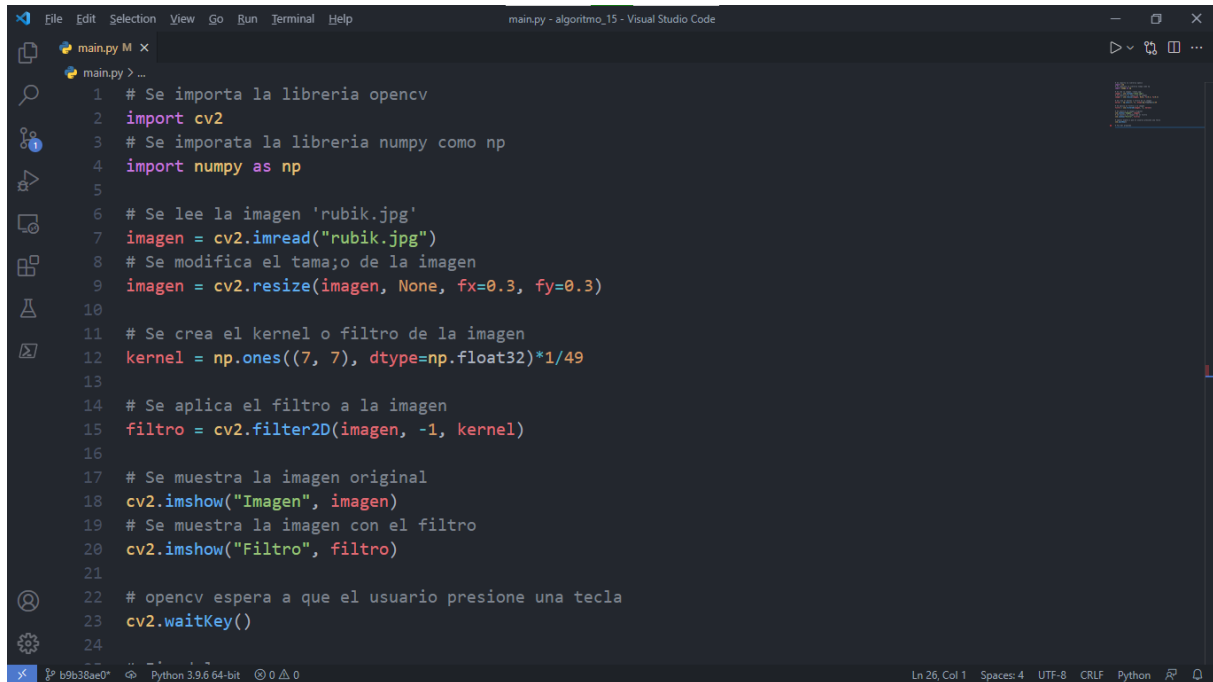
# Se muestra la imagen original
cv2.imshow('Imagen', imagen)
# Se muestra la imagen con el filtro
cv2.imshow('Filtro', filtro)

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

# Fin del programa
```

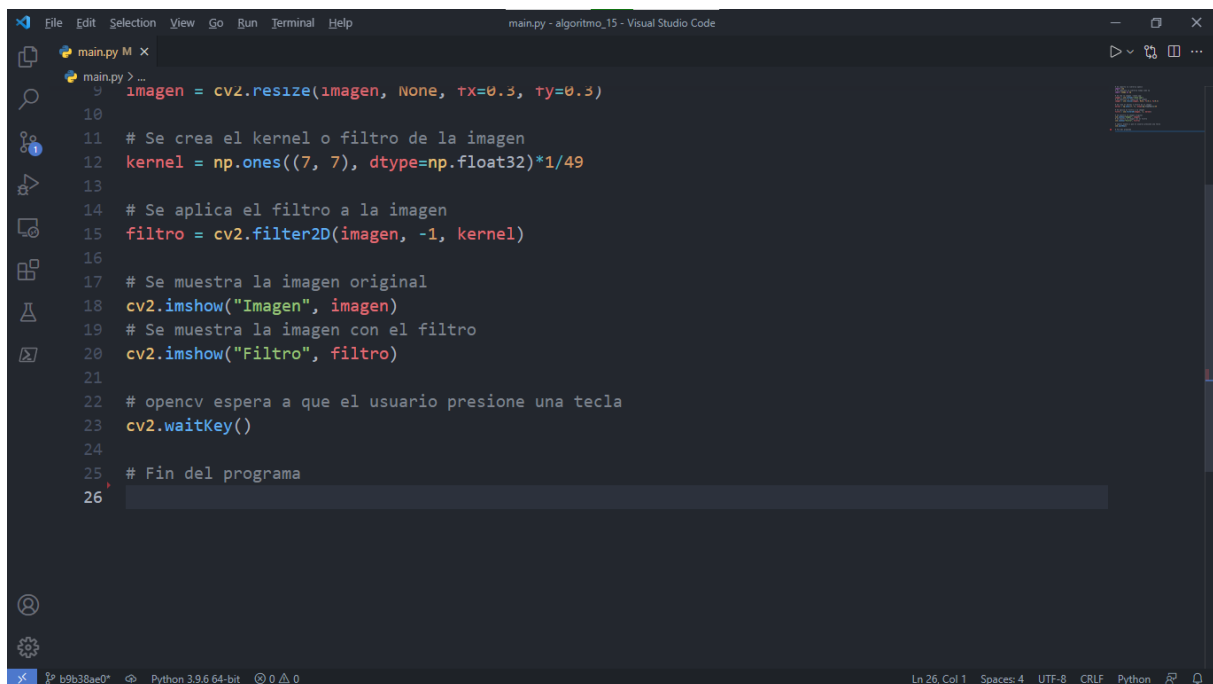
Para el código del ejercicio c) primero se importan las librerías necesarias ("*opencv*" y "*numpy*"). Después se carga la imagen '*rubik.jpg*' y se modifica su tamaño. Luego se genera el kernel y se aplica el filtro a la imagen. Por último se muestran la imagen original y la imagen con el filtro. Al final, "*opencv*" espera a que el usuario presione una tecla para terminar el programa.

## 3.2. Capturas del código del ejercicio



```
1 # Se importa la libreria opencv
2 import cv2
3 # Se importa la libreria numpy como np
4 import numpy as np
5
6 # Se lee la imagen 'rubik.jpg'
7 imagen = cv2.imread("rubik.jpg")
8 # Se modifica el tamaño de la imagen
9 imagen = cv2.resize(imagen, None, fx=0.3, fy=0.3)
10
11 # Se crea el kernel o filtro de la imagen
12 kernel = np.ones((7, 7), dtype=np.float32)*1/49
13
14 # Se aplica el filtro a la imagen
15 filtro = cv2.filter2D(imagen, -1, kernel)
16
17 # Se muestra la imagen original
18 cv2.imshow("Imagen", imagen)
19 # Se muestra la imagen con el filtro
20 cv2.imshow("Filtro", filtro)
21
22 # opencv espera a que el usuario presione una tecla
23 cv2.waitKey()
24
```

Figura 3.1: Captura 1 del código del ejercicio



```
9 imagen = cv2.resize(imagen, None, fx=0.3, fy=0.3)
10
11 # Se crea el kernel o filtro de la imagen
12 kernel = np.ones((7, 7), dtype=np.float32)*1/49
13
14 # Se aplica el filtro a la imagen
15 filtro = cv2.filter2D(imagen, -1, kernel)
16
17 # Se muestra la imagen original
18 cv2.imshow("Imagen", imagen)
19 # Se muestra la imagen con el filtro
20 cv2.imshow("Filtro", filtro)
21
22 # opencv espera a que el usuario presione una tecla
23 cv2.waitKey()
24
25 # Fin del programa
26
```

Figura 3.2: Captura 2 del código del ejercicio

### 3.3. Imágenes obtenidas



Figura 3.3: Imagen original



Figura 3.4: Imagen obtenida en el ejercicio

# Capítulo 4

## Ejercicio d)

### 4.1. Código

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

# Se lee la imagen 'rubik.jpg'
imagen = cv2.imread('rubik.jpg')
# Se modifica el tamaño de la imagen
imagen = cv2.resize(imagen, None, fx=0.3, fy=0.3)

# Se crea el kernel o filtro de la imagen
kernel = np.array([
    [0, -1, 0],
    [-1, 4, -1],
    [0, -1, 0]
], dtype=np.float32)

# Se aplica el filtro a la imagen
filtro = cv2.filter2D(imagen, -1, kernel)

# Se muestra la imagen original
cv2.imshow('Imagen', imagen)
# Se muestra la imagen con el filtro
cv2.imshow('Filtro', filtro)

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

# Fin del programa
```

Para el código del ejercicio d) primero se importan las librerías necesarias ("**opencv**" y "**numpy**"). Después se carga la imagen '**rubik.jpg**' y se modifica su tamaño. Luego se genera el kernel y se aplica el filtro a la imagen. Por último se muestran la imagen original y la imagen con el filtro. Al final, "**opencv**" espera a que el usuario presione una tecla para terminar el programa.

## 4.2. Capturas del código del ejercicio

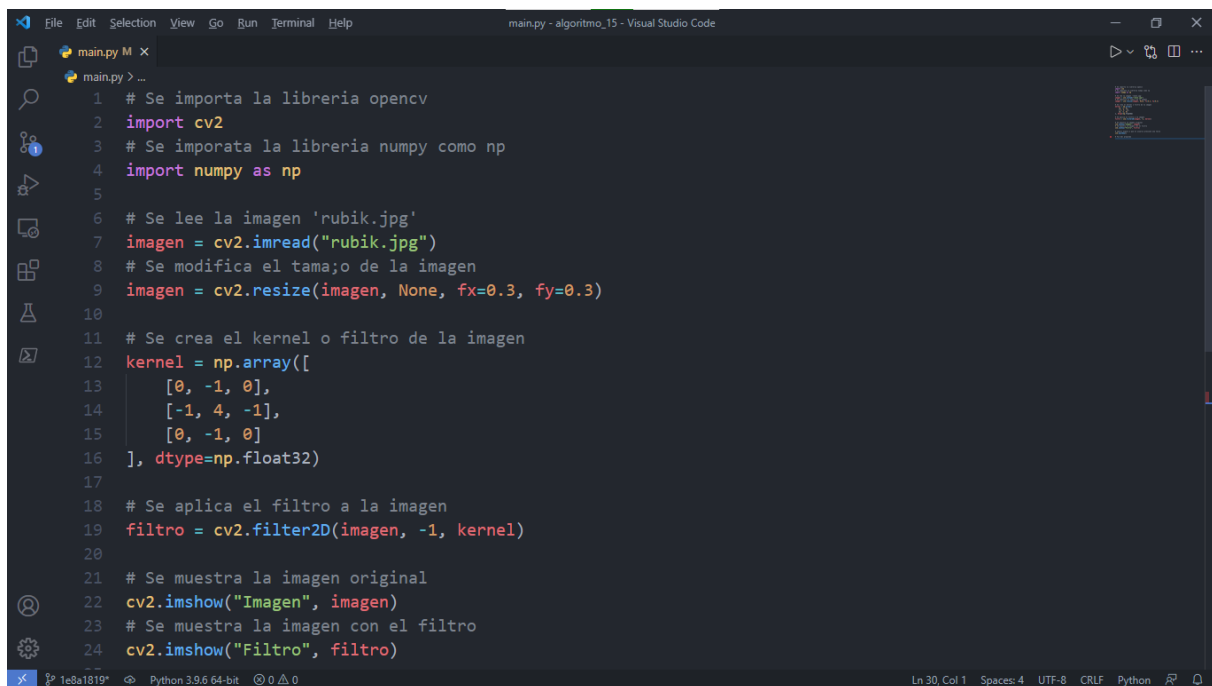
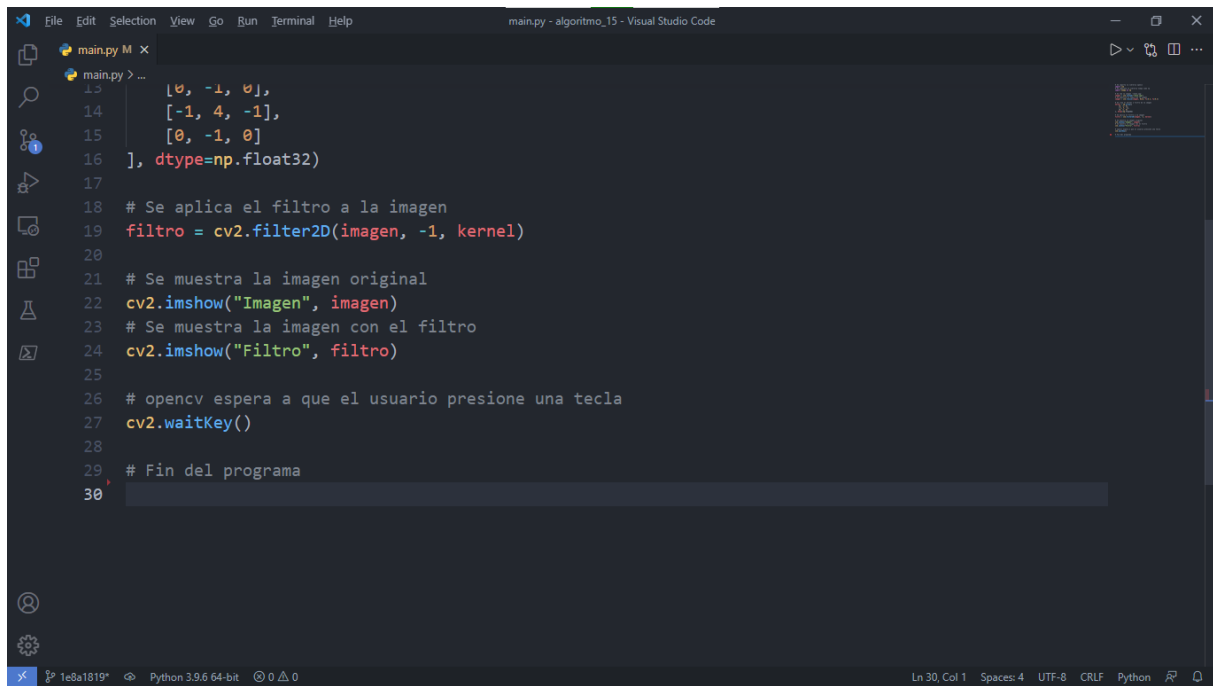
A screenshot of a Visual Studio Code editor window titled 'main.py - algoritmo\_15 - Visual Studio Code'. The editor shows a Python script in a file named 'main.py'. The code imports 'cv2' and 'numpy' as 'np'. It reads an image 'rubik.jpg', resizes it to 30% of its original dimensions, and creates a 3x3 kernel with values [[0, -1, 0], [-1, 4, -1], [0, -1, 0]]. The kernel is applied to the image using 'cv2.filter2D'. Finally, the original image and the filtered image are displayed using 'cv2.imshow'. The status bar at the bottom indicates 'Python 3.9.6 64-bit' and 'Ln 30, Col 1'.

Figura 4.1: Captura 1 del código del ejercicio



```
File Edit Selection View Go Run Terminal Help
main.py - algoritmo_15 - Visual Studio Code

main.py M x
13
14     [0, -1, 0],
15     [-1, 4, -1],
16     [0, -1, 0]
17 ], dtype=np.float32)
18
19 # Se aplica el filtro a la imagen
20 filtro = cv2.filter2D(imagen, -1, kernel)
21
22 # Se muestra la imagen original
23 cv2.imshow("Imagen", imagen)
24 # Se muestra la imagen con el filtro
25 cv2.imshow("Filtro", filtro)
26
27 # opencv espera a que el usuario presione una tecla
28 cv2.waitKey()
29
30 # Fin del programa
31
```

Figura 4.2: Captura 2 del código del ejercicio



### 4.3. Imágenes obtenidas



Figura 4.3: Imagen original

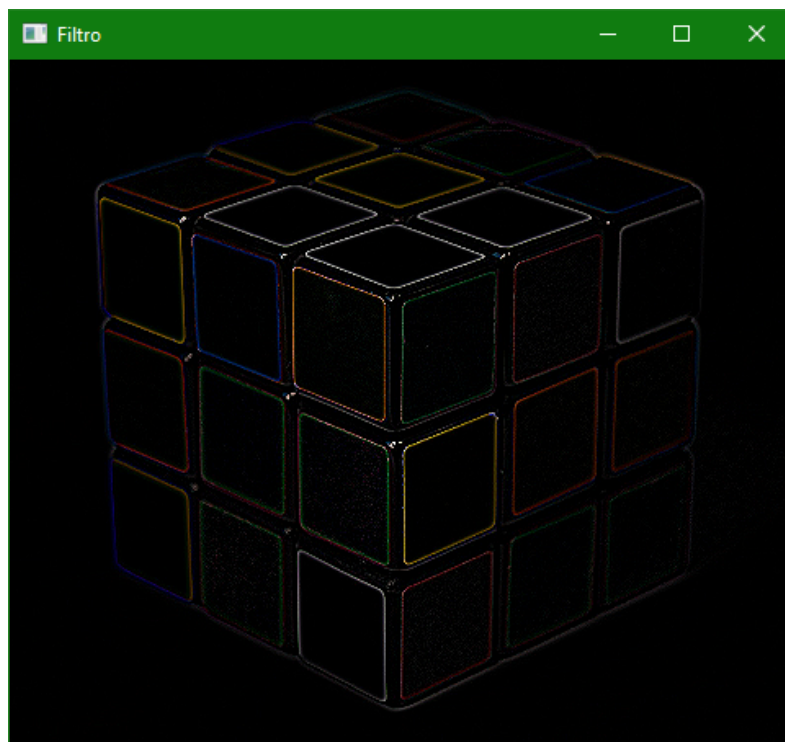


Figura 4.4: Imagen obtenida en el ejercicio

# Capítulo 5

## Ejercicio e)

### 5.1. Código

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

# Se lee la imagen 'rubik.jpg'
imagen = cv2.imread('rubik.jpg')
# Se modifica el tamaño de la imagen
imagen = cv2.resize(imagen, None, fx=0.3, fy=0.3)

grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)

# Se crea el kernel o filtro de la imagen
kernel = np.array([
    [0, -1, 0],
    [-1, 4, -1],
    [0, -1, 0]
], dtype=np.float32)
# Se aplica el filtro a la imagen
filtro = cv2.filter2D(grises, -1, kernel)

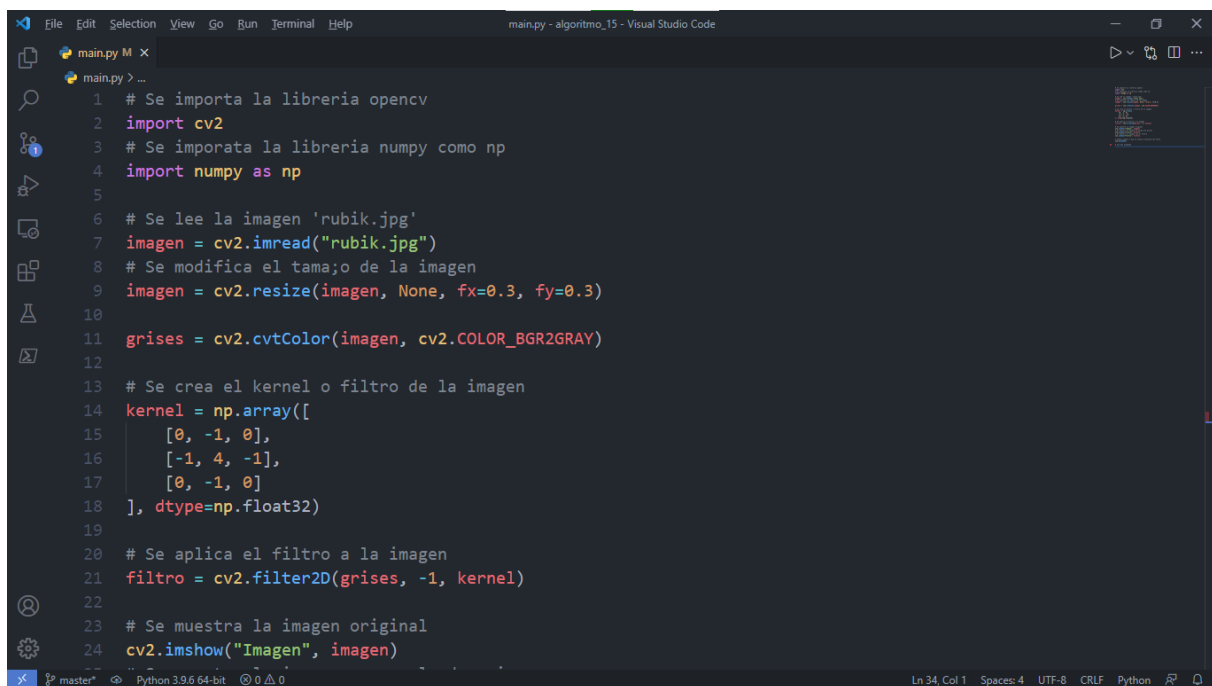
# Se muestra la imagen original
cv2.imshow('Imagen', imagen)
# Se muestra la imagen en escala de grises
cv2.imshow('Grises', grises)
# Se muestra la imagen con el filtro
cv2.imshow('Filtro', filtro)

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

# Fin del programa
```

Para el código del ejercicio e) primero se importan las librerías necesarias ("**opencv**" y "**numpy**"). Después se carga la imagen '**rubik.jpg**', se modifica su tamaño y se modifica su formato a escala de grises. Luego se genera el kernel y se aplica el filtro a la imagen en escala de grises. Por último se muestran la imagen original, la imagen en escala de grises y la imagen con el filtro. Al final, "**opencv**" espera a que el usuario presione una tecla para terminar el programa.

## 5.2. Capturas del código del ejercicio

A screenshot of a Visual Studio Code editor window. The title bar reads "main.py - algoritmo\_15 - Visual Studio Code". The editor contains a Python script with the following code:

```
1 # Se importa la libreria opencv
2 import cv2
3 # Se importa la libreria numpy como np
4 import numpy as np
5
6 # Se lee la imagen 'rubik.jpg'
7 imagen = cv2.imread("rubik.jpg")
8 # Se modifica el tamaño de la imagen
9 imagen = cv2.resize(imagen, None, fx=0.3, fy=0.3)
10
11 grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
12
13 # Se crea el kernel o filtro de la imagen
14 kernel = np.array([
15     [0, -1, 0],
16     [-1, 4, -1],
17     [0, -1, 0]
18 ], dtype=np.float32)
19
20 # Se aplica el filtro a la imagen
21 filtro = cv2.filter2D(grises, -1, kernel)
22
23 # Se muestra la imagen original
24 cv2.imshow("Imagen", imagen)
```

The status bar at the bottom indicates "Ln 34, Col 1", "Spaces: 4", "UTF-8", "CRLF", "Python", and a Python icon.

Figura 5.1: Captura 1 del código del ejercicio

```
File Edit Selection View Go Run Terminal Help
main.py - algoritmo_15 - Visual Studio Code
main.py M x
main.py > ...
16     [-1, 4, -1],
17     [0, -1, 0]
18 ], dtype=np.float32)
19
20 # Se aplica el filtro a la imagen
21 filtro = cv2.filter2D(grises, -1, kernel)
22
23 # Se muestra la imagen original
24 cv2.imshow("Imagen", imagen)
25 # Se muestra la imagen en escala de grises
26 cv2.imshow("Grises", grises)
27 # Se muestra la imagen con el filtro
28 cv2.imshow("Filtro", filtro)
29
30 # opencv espera a que el usuario presione una tecla
31 cv2.waitKey()
32
33 # Fin del programa
34
```

Figura 5.2: Captura 2 del código del ejercicio

### 5.3. Imágenes obtenidas



Figura 5.3: Imagen original

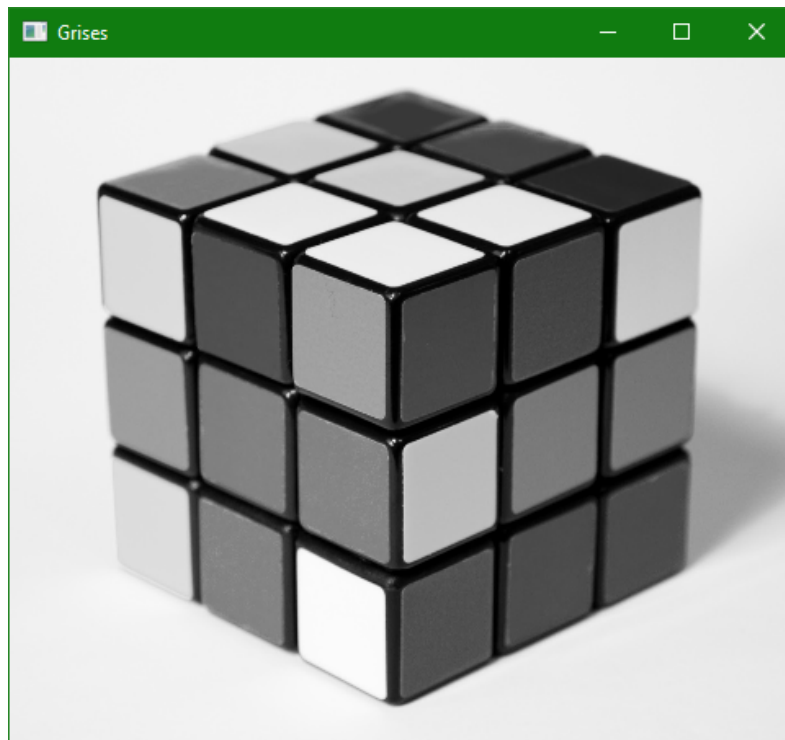


Figura 5.4: Imagen en escala de grises

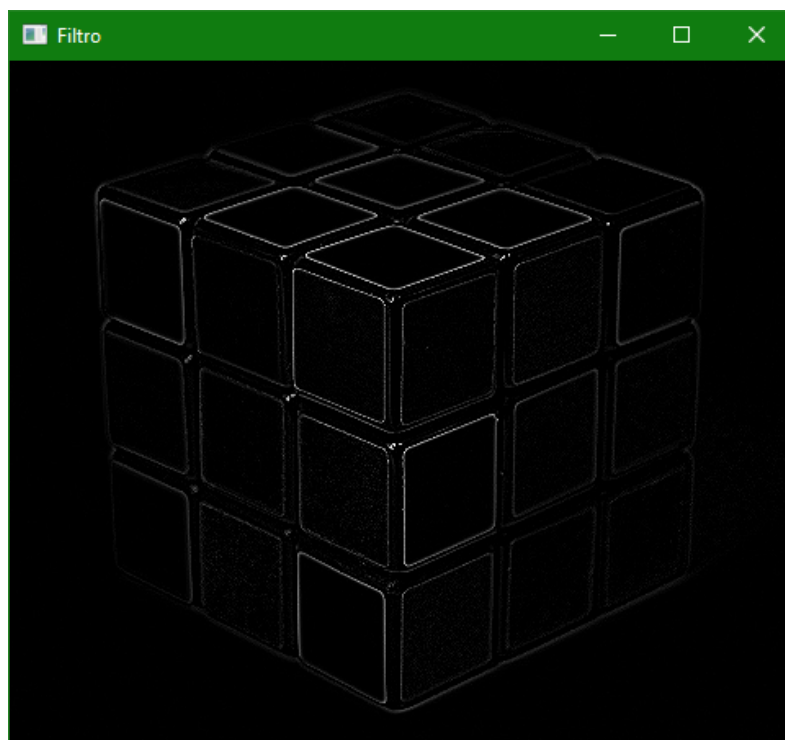


Figura 5.5: Imagen obtenida en el ejercicio