

Universidad de Guadalajara



Algoritmo 18

Muñoz Nuñez Ian Emmanuel

Visión Robótica

"Operaciones Morfológicas"

Índice general

1. Recursos de imagen	3
2. Ejercicio 1	7
3. Ejercicio 2	9
4. Ejercicio 3	11
5. Ejercicio 4	13
6. Ejercicio 5	15
7. Ejercicio 6	17
8. Ejercicio 7	19
9. Ejercicio 8	21

Índice de figuras

1.1. Imagen 1	3
1.2. Imagen 2	4
1.3. Imagen 3	4
1.4. Imagen 4	5
1.5. Imagen 5	5
1.6. Imagen 6	6
2.1. Captura del código del ejercicio 1	8
2.2. Imagen obtenida en el ejercicio 1	8
3.1. Captura del código del ejercicio 2	10
3.2. Imagen obtenida en el ejercicio 2	10
4.1. Captura del código del ejercicio 3	12
4.2. Imagen obtenida en el ejercicio 3	12
5.1. Captura del código del ejercicio 4	14
5.2. Imagen obtenida en el ejercicio 4	14
6.1. Captura del código del ejercicio 5	16
6.2. Imagen obtenida en el ejercicio 5	16
7.1. Captura del código del ejercicio 6	18
7.2. Imagen obtenida en el ejercicio 6	18
8.1. Captura del código del ejercicio 7	20
8.2. Imagen obtenida en el ejercicio 7	20
9.1. Captura del código del ejercicio 8	22
9.2. Imagen obtenida en el ejercicio 8	22

Capítulo 1

Recursos de imagen

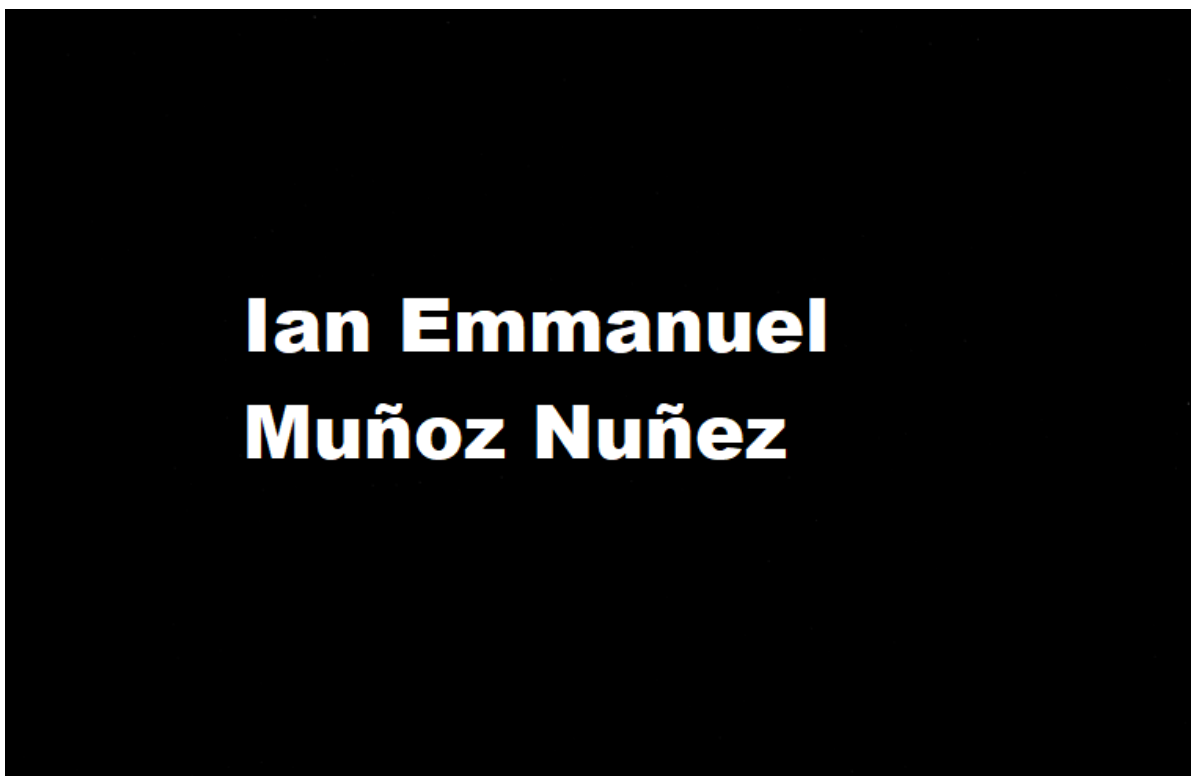


Figura 1.1: Imagen 1

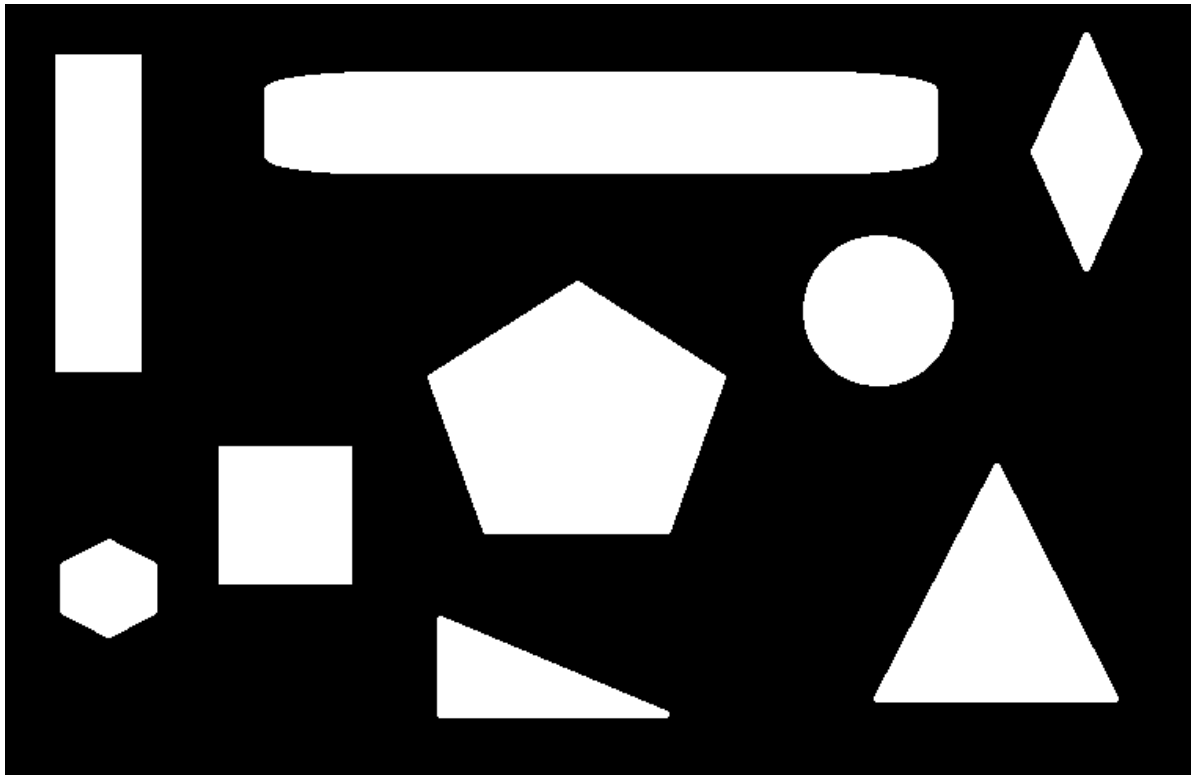


Figura 1.2: Imagen 2



Figura 1.3: Imagen 3

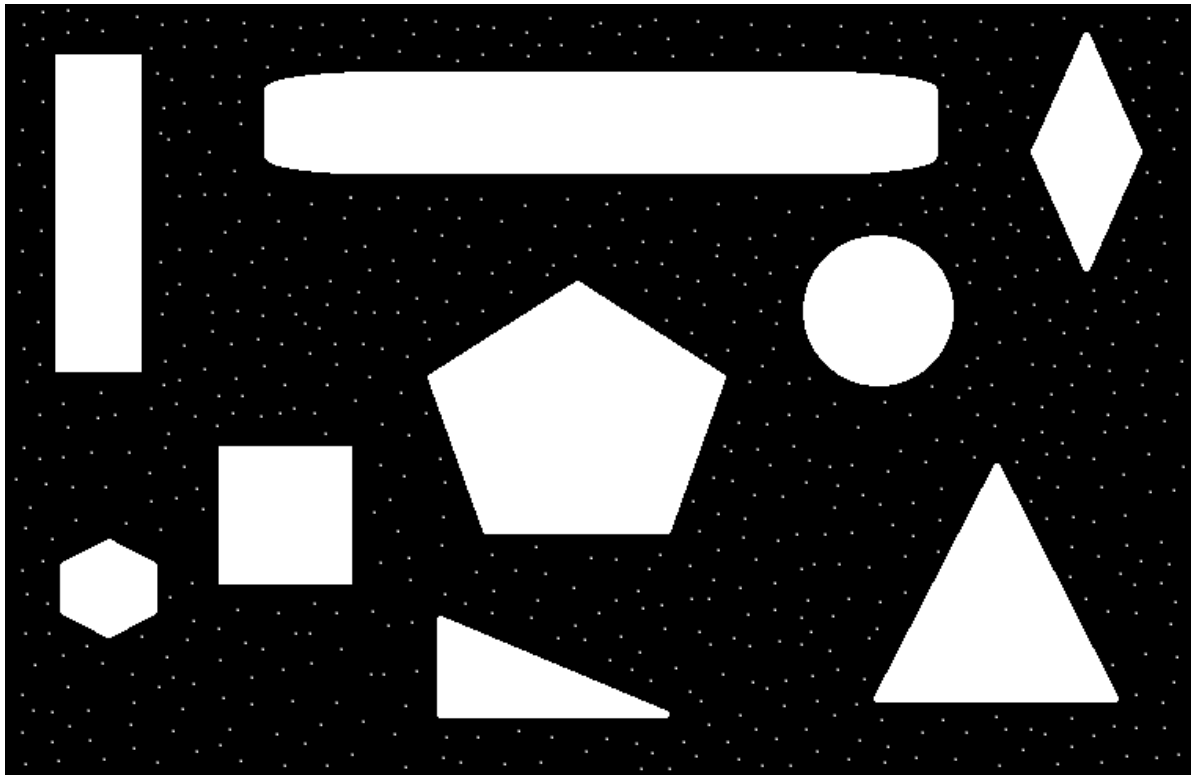


Figura 1.4: Imagen 4

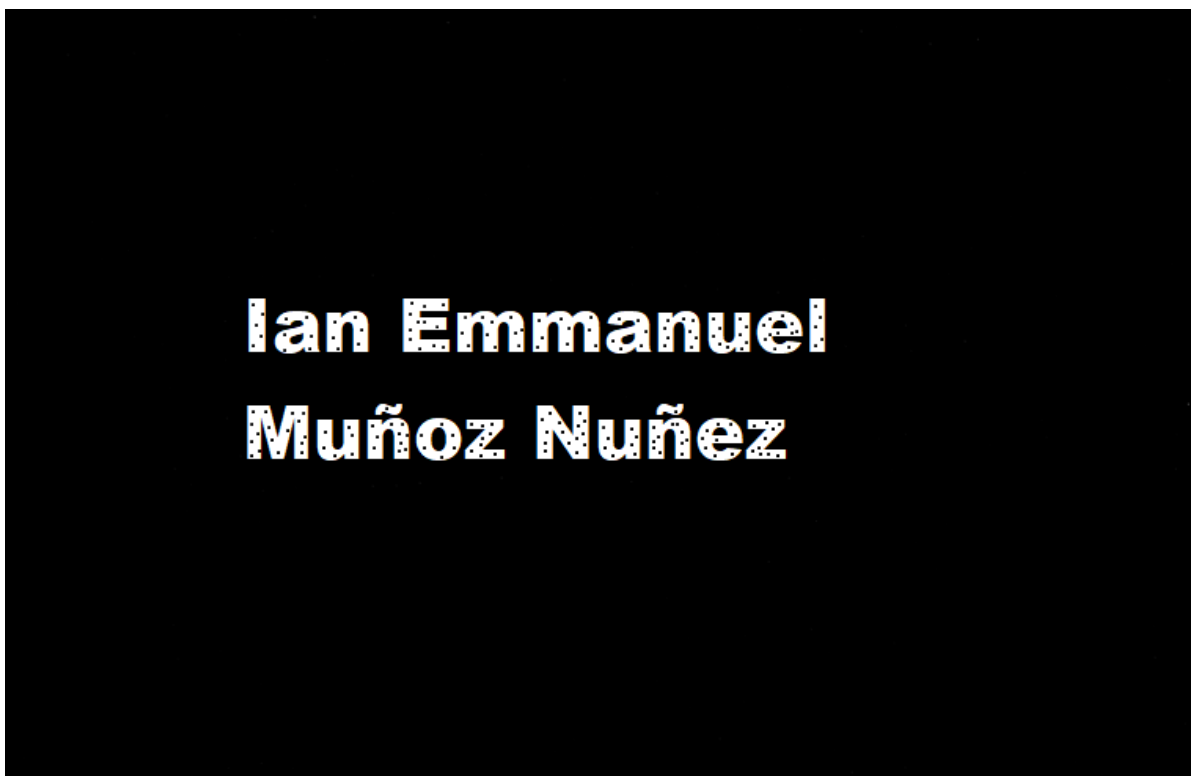


Figura 1.5: Imagen 5

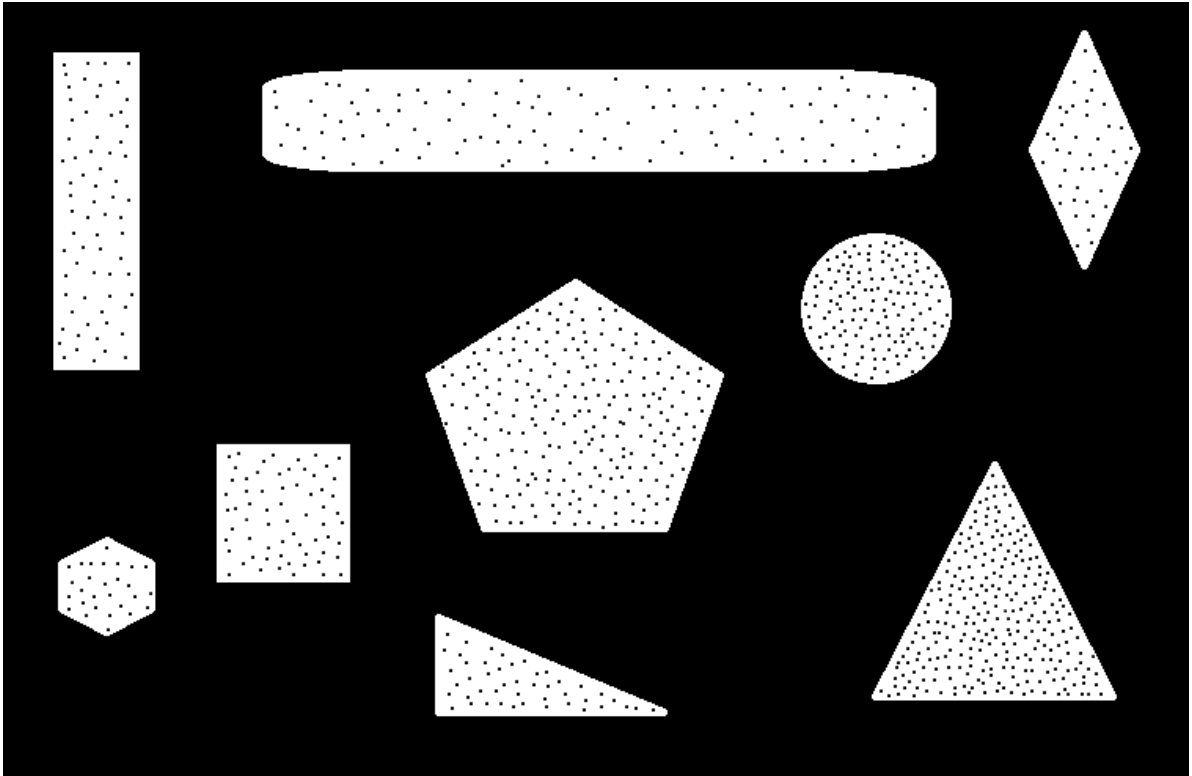


Figura 1.6: Imagen 6

Capítulo 2

Ejercicio 1

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

# Se carga la imagen 'imagen_1.png'
imagen = cv2.imread('imagen_1.png')

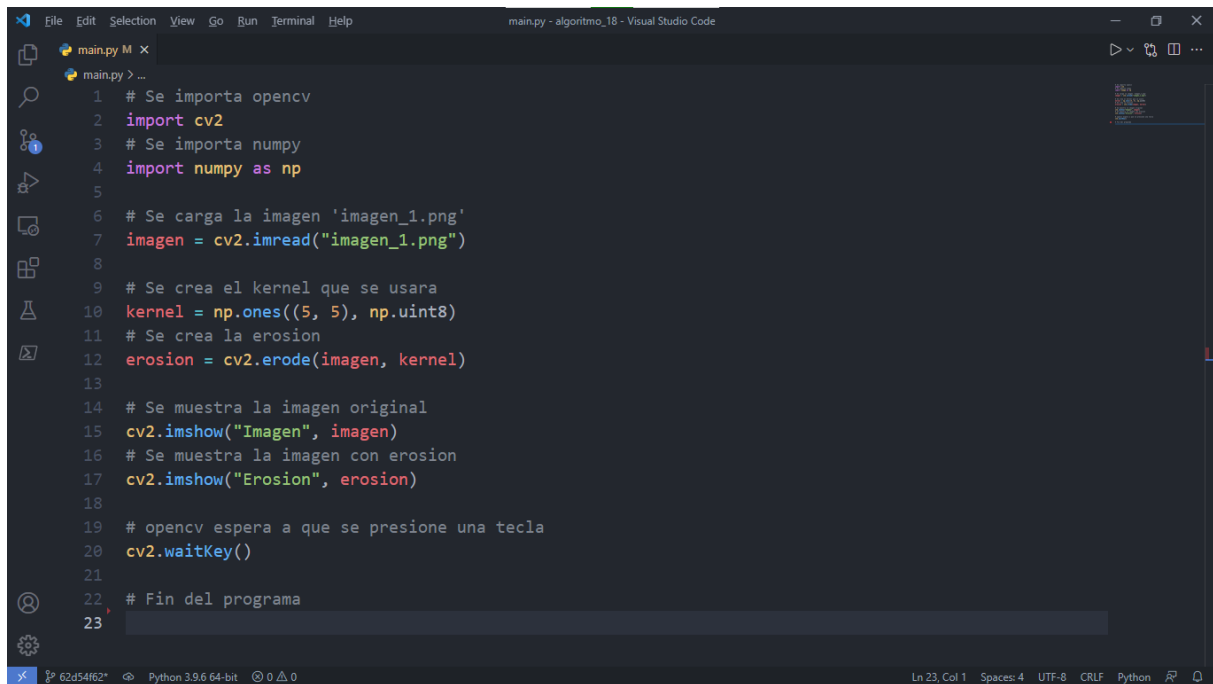
# Se crea el kernel que se usara
kernel = np.ones((5, 5), np.uint8)
# Se crea la erosion
erosion = cv2.erode(imagen, kernel)

# Se muestra la imagen original
cv2.imshow('Imagen', imagen)
# Se muestra la imagen con erosion
cv2.imshow('Erosion', erosion)

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

# Fin del programa
```

Para el código del ejercicio 1 primero se importan las librerías necesarias (**opencv**, **numpy**), luego se lee la imagen "**imagen_1.png**" previamente creada, se genera el **kernel** con una matriz de unos de 5 por 5 y se genera la erosión de la imagen. Por último se muestra la imagen original y la imagen obtenida. 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_1.png'
7 imagen = cv2.imread("imagen_1.png")
8
9 # Se crea el kernel que se usara
10 kernel = np.ones((5, 5), np.uint8)
11 # Se crea la erosion
12 erosion = cv2.erode(imagen, kernel)
13
14 # Se muestra la imagen original
15 cv2.imshow("Imagen", imagen)
16 # Se muestra la imagen con erosion
17 cv2.imshow("Erosion", erosion)
18
19 # opencv espera a que se presione una tecla
20 cv2.waitKey()
21
22 # Fin del programa
23
```

Figura 2.1: Captura del código del ejercicio 1

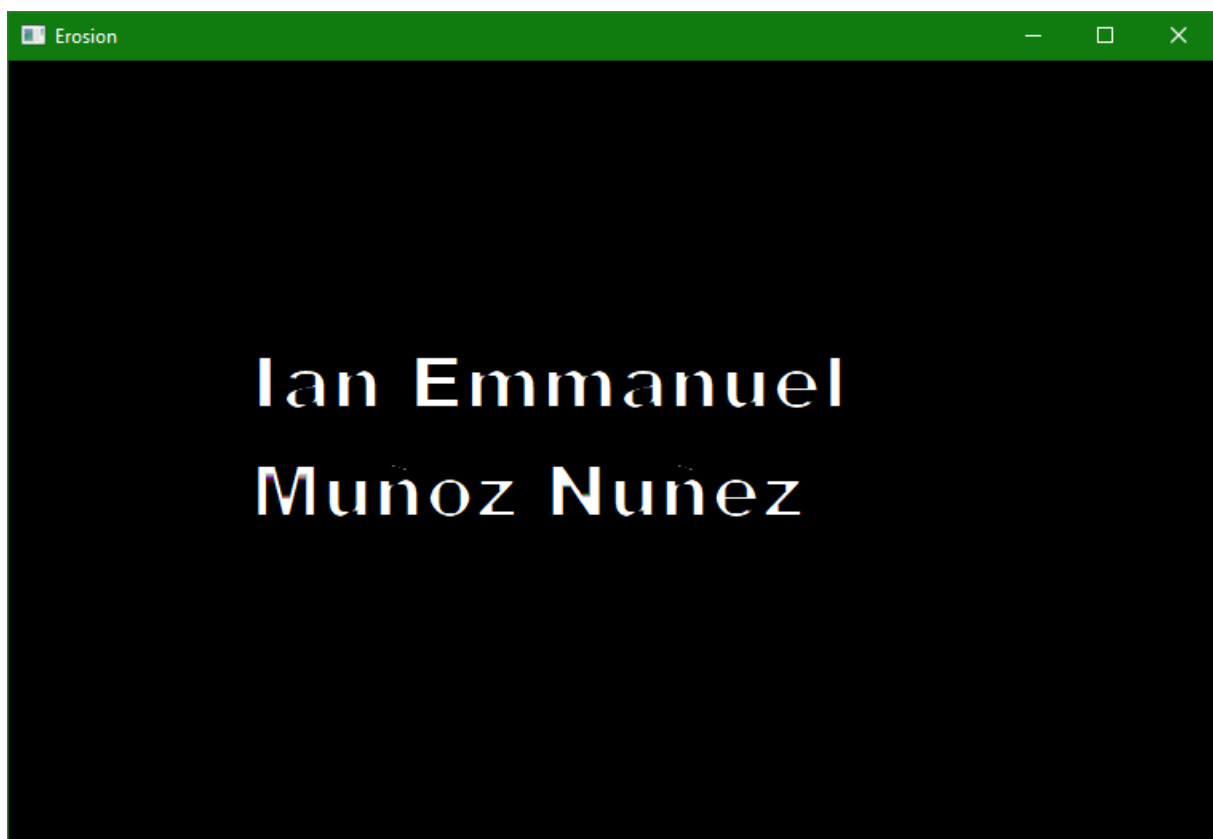


Figura 2.2: Imagen obtenida en el ejercicio 1

Capítulo 3

Ejercicio 2

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

# Se carga la imagen 'imagen_2.png'
imagen = cv2.imread('imagen_2.png')

# Se crea el kernel que se usara
kernel = np.ones((5, 5), np.uint8)
# Se crea la erosion
erosion = cv2.erode(imagen, kernel)

# Se muestra la imagen original
cv2.imshow('Imagen', imagen)
# Se muestra la imagen con erosion
cv2.imshow('Erosion', erosion)

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

# Fin del programa
```

Para el código del ejercicio 1 primero se importan las librerías necesarias (**opencv**, **numpy**), luego se lee la imagen "**imagen_2.png**" previamente creada, se genera el **kernel** con una matriz de unos de 5 por 5 y se genera la erosión de la imagen. Por último se muestra la imagen original y la imagen obtenida. Al final **opencv** espera a que el usuario presione una tecla para terminar el programa.

```
File Edit Selection View Go Run Terminal Help
main.py - algoritmo_18 - Visual Studio Code
main.py M x
main.py > ...
1 # Se importa opencv
2 import cv2
3 # Se importa numpy
4 import numpy as np
5
6 # Se carga la imagen 'imagen_2.png'
7 imagen = cv2.imread("imagen_2.png")
8
9 # Se crea el kernel que se usara
10 kernel = np.ones((5, 5), np.uint8)
11 # Se crea la erosion
12 erosion = cv2.erode(imagen, kernel)
13
14 # Se muestra la imagen original
15 cv2.imshow("Imagen", imagen)
16 # Se muestra la imagen con erosion
17 cv2.imshow("Erosion", erosion)
18
19 # opencv espera a que se presione una tecla
20 cv2.waitKey()
21
22 # Fin del programa
23
```

Figura 3.1: Captura del código del ejercicio 2

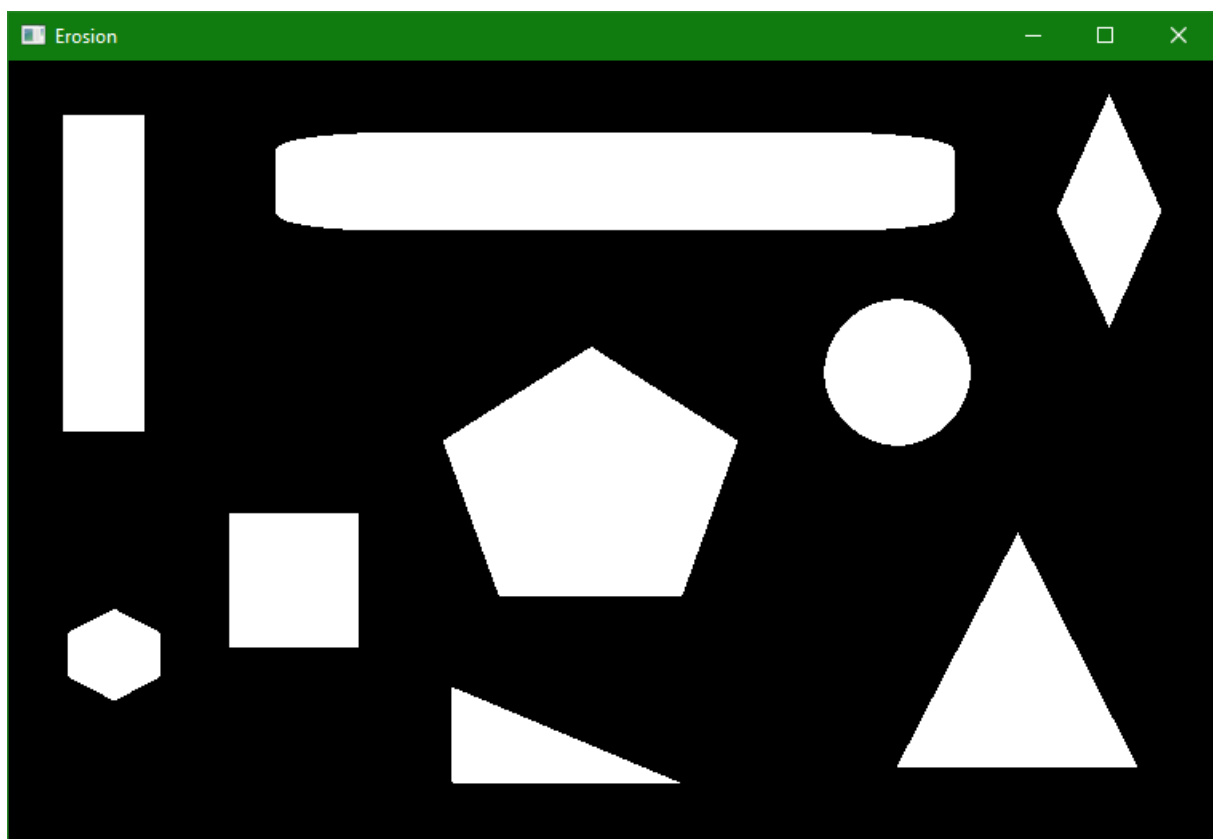


Figura 3.2: Imagen obtenida en el ejercicio 2

Capítulo 4

Ejercicio 3

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

# Se carga la imagen 'imagen_1.png'
imagen = cv2.imread('imagen_1.png')

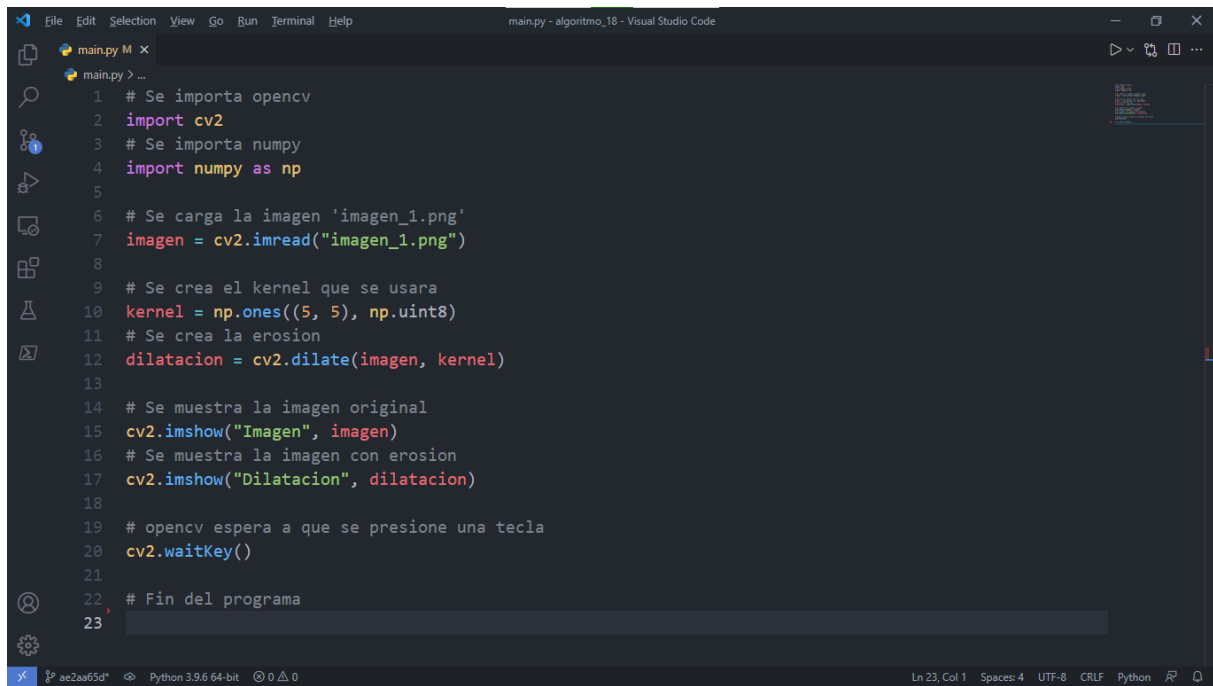
# Se crea el kernel que se usara
kernel = np.ones((5, 5), np.uint8)
# Se crea la erosion
dilatacion = cv2.dilate(imagen, kernel)

# Se muestra la imagen original
cv2.imshow('Imagen', imagen)
# Se muestra la imagen con erosion
cv2.imshow('Dilatacion', dilatacion)

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

# Fin del programa
```

Para el código del ejercicio 1 primero se importan las librerías necesarias (**opencv**, **numpy**), luego se lee la imagen "**imagen_1.png**" previamente creada, se genera el **kernel** con una matriz de unos de 5 por 5 y se genera la dilatación de la imagen. Por último se muestra la imagen original y la imagen obtenida. 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_1.png'
7 imagen = cv2.imread("imagen_1.png")
8
9 # Se crea el kernel que se usara
10 kernel = np.ones((5, 5), np.uint8)
11 # Se crea la erosion
12 dilatacion = cv2.dilate(imagen, kernel)
13
14 # Se muestra la imagen original
15 cv2.imshow("Imagen", imagen)
16 # Se muestra la imagen con erosion
17 cv2.imshow("Dilatacion", dilatacion)
18
19 # opencv espera a que se presione una tecla
20 cv2.waitKey()
21
22 # Fin del programa
23
```

Figura 4.1: Captura del código del ejercicio 3

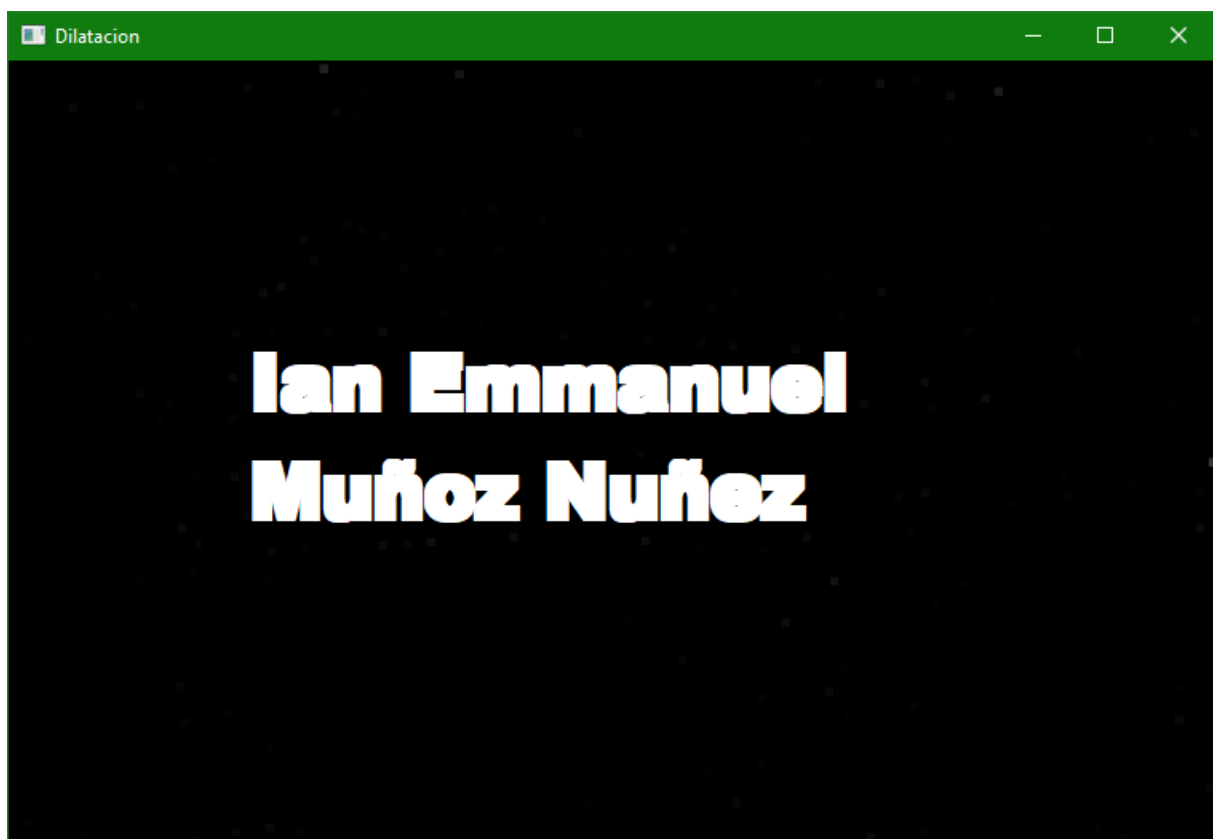


Figura 4.2: Imagen obtenida en el ejercicio 3

Capítulo 5

Ejercicio 4

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

# Se carga la imagen 'imagen_2.png'
imagen = cv2.imread('imagen_2.png')

# Se crea el kernel que se usara
kernel = np.ones((5, 5), np.uint8)
# Se crea la erosion
dilatacion = cv2.dilate(imagen, kernel)

# Se muestra la imagen original
cv2.imshow('Imagen', imagen)
# Se muestra la imagen con erosion
cv2.imshow('Dilatacion', dilatacion)

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

# Fin del programa
```

Para el código del ejercicio 1 primero se importan las librerías necesarias (**opencv**, **numpy**), luego se lee la imagen "**imagen_2.png**" previamente creada, se genera el **kernel** con una matriz de unos de 5 por 5 y se genera la dilatación de la imagen. Por último se muestra la imagen original y la imagen obtenida. Al final **opencv** espera a que el usuario presione una tecla para terminar el programa.

```
File Edit Selection View Go Run Terminal Help
main.py - algoritmo_18 - Visual Studio Code
main.py M x
main.py > ...
1 # Se importa opencv
2 import cv2
3 # Se importa numpy
4 import numpy as np
5
6 # Se carga la imagen 'imagen_2.png'
7 imagen = cv2.imread("imagen_2.png")
8
9 # Se crea el kernel que se usara
10 kernel = np.ones((5, 5), np.uint8)
11 # Se crea la erosion
12 dilatacion = cv2.dilate(imagen, kernel)
13
14 # Se muestra la imagen original
15 cv2.imshow("Imagen", imagen)
16 # Se muestra la imagen con erosion
17 cv2.imshow("Dilatacion", dilatacion)
18
19 # opencv espera a que se presione una tecla
20 cv2.waitKey()
21
22 # Fin del programa
23
```

Figura 5.1: Captura del código del ejercicio 4

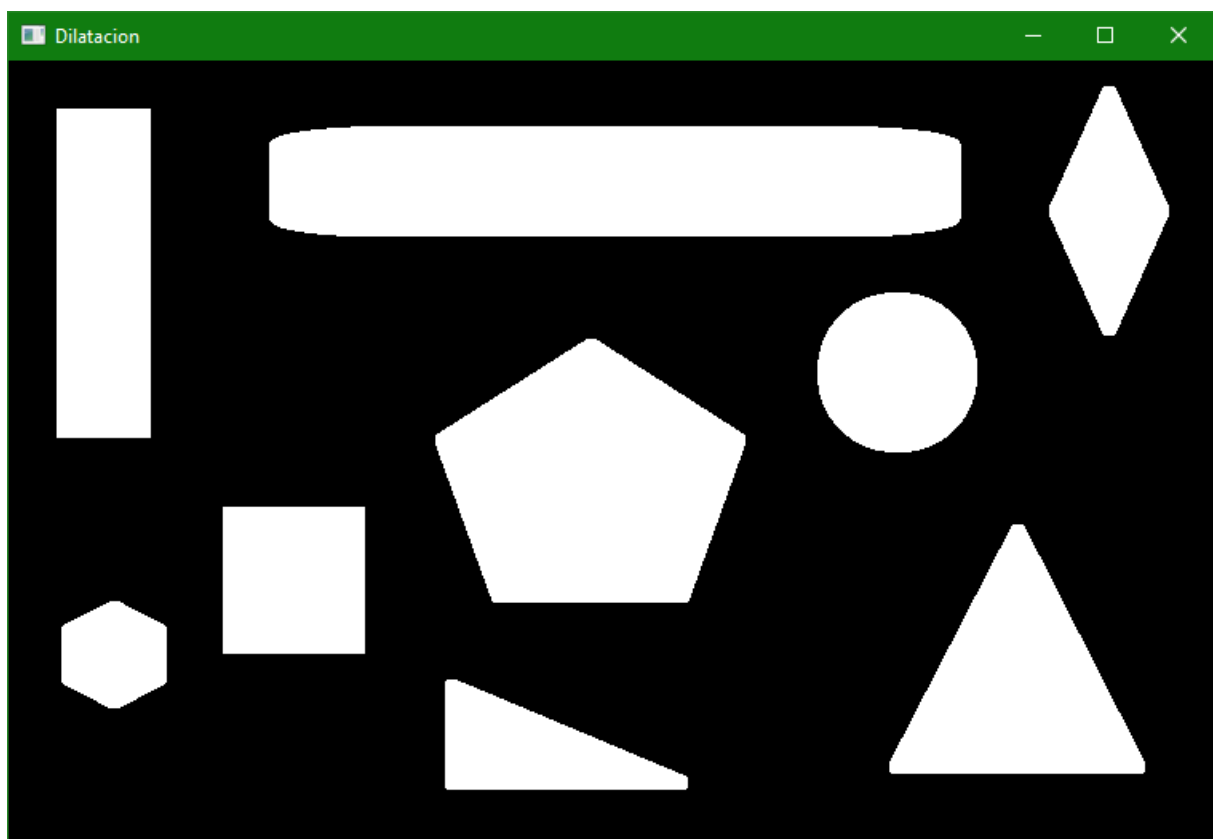


Figura 5.2: Imagen obtenida en el ejercicio 4

Capítulo 6

Ejercicio 5

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

# Se carga la imagen 'imagen_3.png'
imagen = cv2.imread('imagen_3.png')

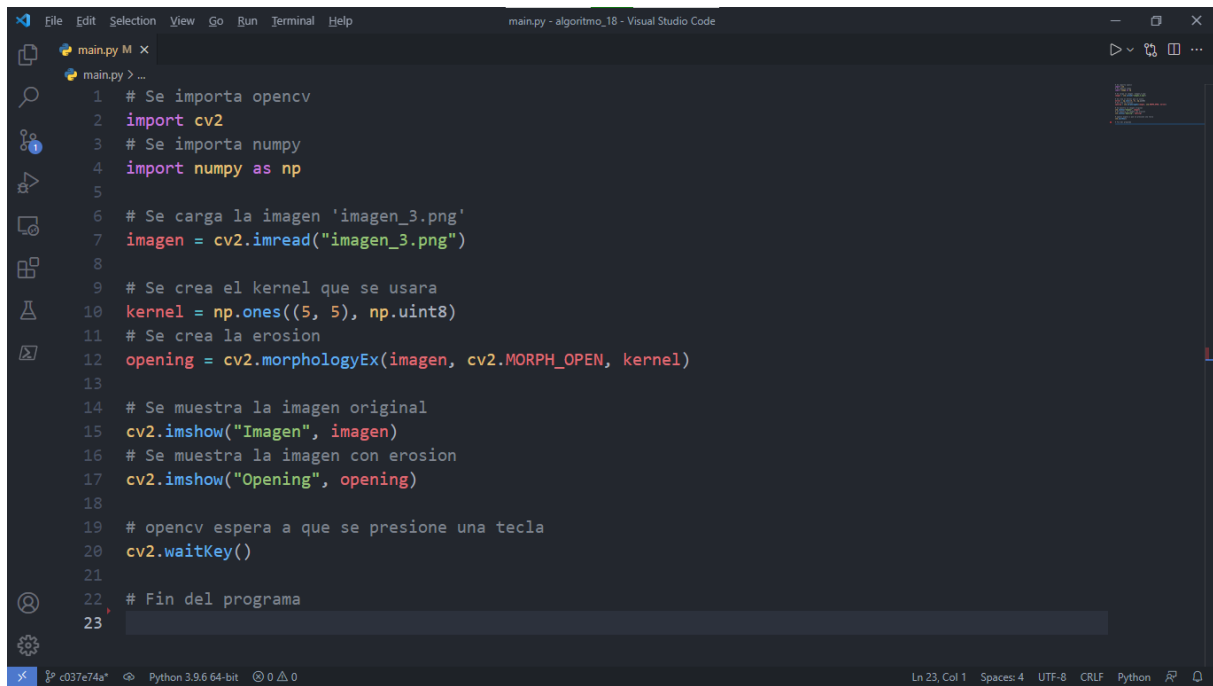
# Se crea el kernel que se usara
kernel = np.ones((5, 5), np.uint8)
# Se crea la erosion
opening = cv2.morphologyEx(imagen, cv2.MORPH_OPEN, kernel)

# Se muestra la imagen original
cv2.imshow('Imagen', imagen)
# Se muestra la imagen con erosion
cv2.imshow('Opening', opening)

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

# Fin del programa
```

Para el código del ejercicio 1 primero se importan las librerías necesarias (**opencv**, **numpy**), luego se lee la imagen "**imagen_3.png**" previamente creada, se genera el **kernel** con una matriz de unos de 5 por 5 y se genera el opening de la imagen. Por último se muestra la imagen original y la imagen obtenida. 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_3.png'
7 imagen = cv2.imread("imagen_3.png")
8
9 # Se crea el kernel que se usara
10 kernel = np.ones((5, 5), np.uint8)
11 # Se crea la erosion
12 opening = cv2.morphologyEx(imagen, cv2.MORPH_OPEN, kernel)
13
14 # Se muestra la imagen original
15 cv2.imshow("Imagen", imagen)
16 # Se muestra la imagen con erosion
17 cv2.imshow("Opening", opening)
18
19 # opencv espera a que se presione una tecla
20 cv2.waitKey()
21
22 # Fin del programa
23
```

Figura 6.1: Captura del código del ejercicio 5

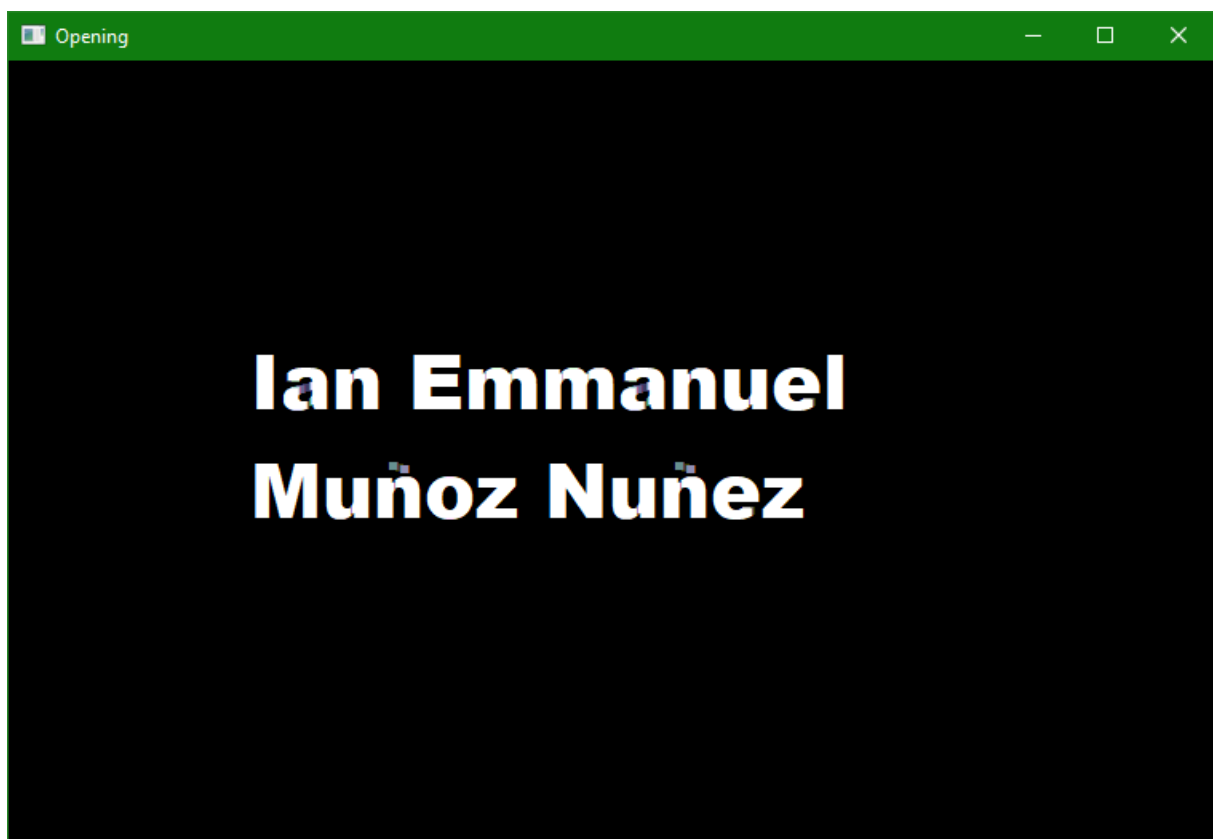


Figura 6.2: Imagen obtenida en el ejercicio 5

Capítulo 7

Ejercicio 6

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

# Se carga la imagen 'imagen_4.png'
imagen = cv2.imread('imagen_4.png')

# Se crea el kernel que se usara
kernel = np.ones((5, 5), np.uint8)
# Se crea la erosion
opening = cv2.morphologyEx(imagen, cv2.MORPH_OPEN, kernel)

# Se muestra la imagen original
cv2.imshow('Imagen', imagen)
# Se muestra la imagen con erosion
cv2.imshow('Opening', opening)

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

# Fin del programa
```

Para el código del ejercicio 1 primero se importan las librerías necesarias (**opencv**, **numpy**), luego se lee la imagen "**imagen_4.png**" previamente creada, se genera el **kernel** con una matriz de unos de 5 por 5 y se genera el opening de la imagen. Por último se muestra la imagen original y la imagen obtenida. Al final **opencv** espera a que el usuario presione una tecla para terminar el programa.

```
File Edit Selection View Go Run Terminal Help
main.py - algoritmo_18 - Visual Studio Code
main.py M x
main.py > ...
1 # Se importa opencv
2 import cv2
3 # Se importa numpy
4 import numpy as np
5
6 # Se carga la imagen 'imagen_4.png'
7 imagen = cv2.imread("imagen_4.png")
8
9 # Se crea el kernel que se usara
10 kernel = np.ones((5, 5), np.uint8)
11 # Se crea la erosion
12 opening = cv2.morphologyEx(imagen, cv2.MORPH_OPEN, kernel)
13
14 # Se muestra la imagen original
15 cv2.imshow("Imagen", imagen)
16 # Se muestra la imagen con erosion
17 cv2.imshow("Opening", opening)
18
19 # opencv espera a que se presione una tecla
20 cv2.waitKey()
21
22 # Fin del programa
23
```

Figura 7.1: Captura del código del ejercicio 6

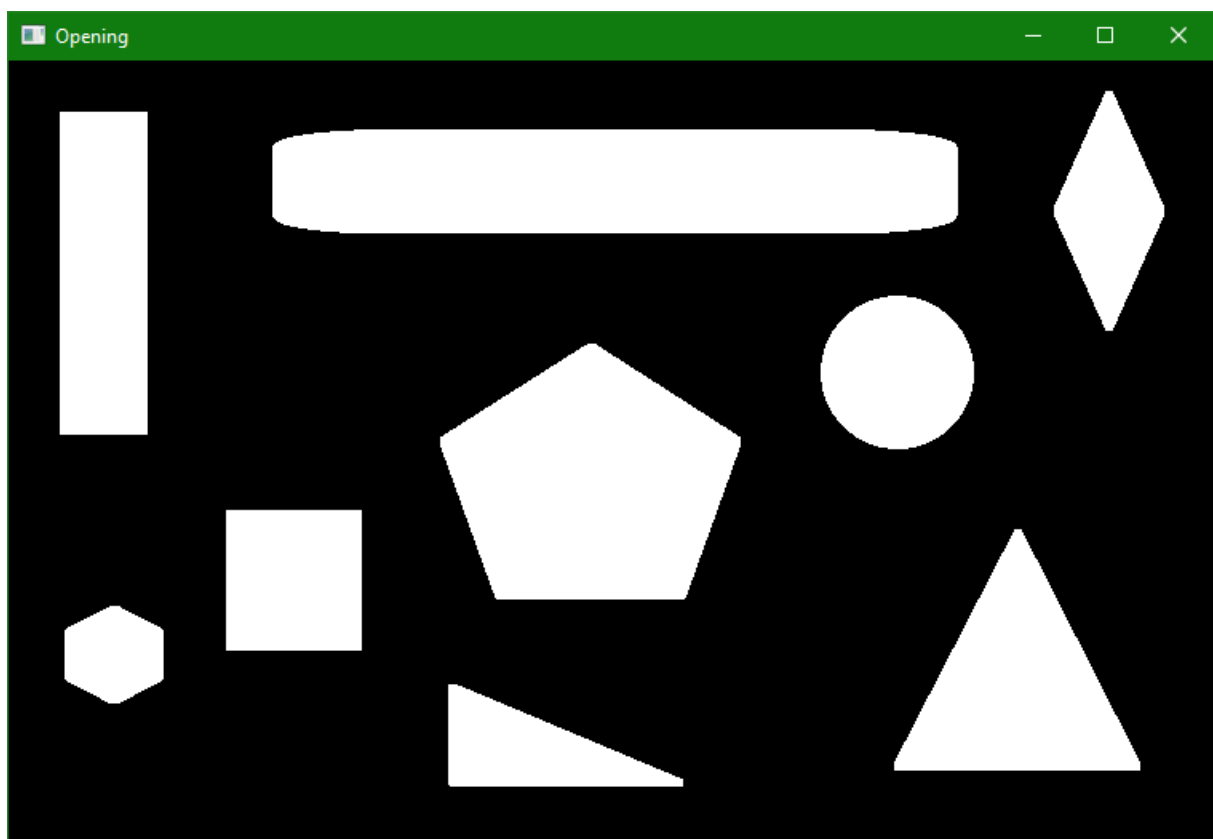


Figura 7.2: Imagen obtenida en el ejercicio 6

Capítulo 8

Ejercicio 7

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

# Se carga la imagen 'imagen_5.png'
imagen = cv2.imread('imagen_5.png')

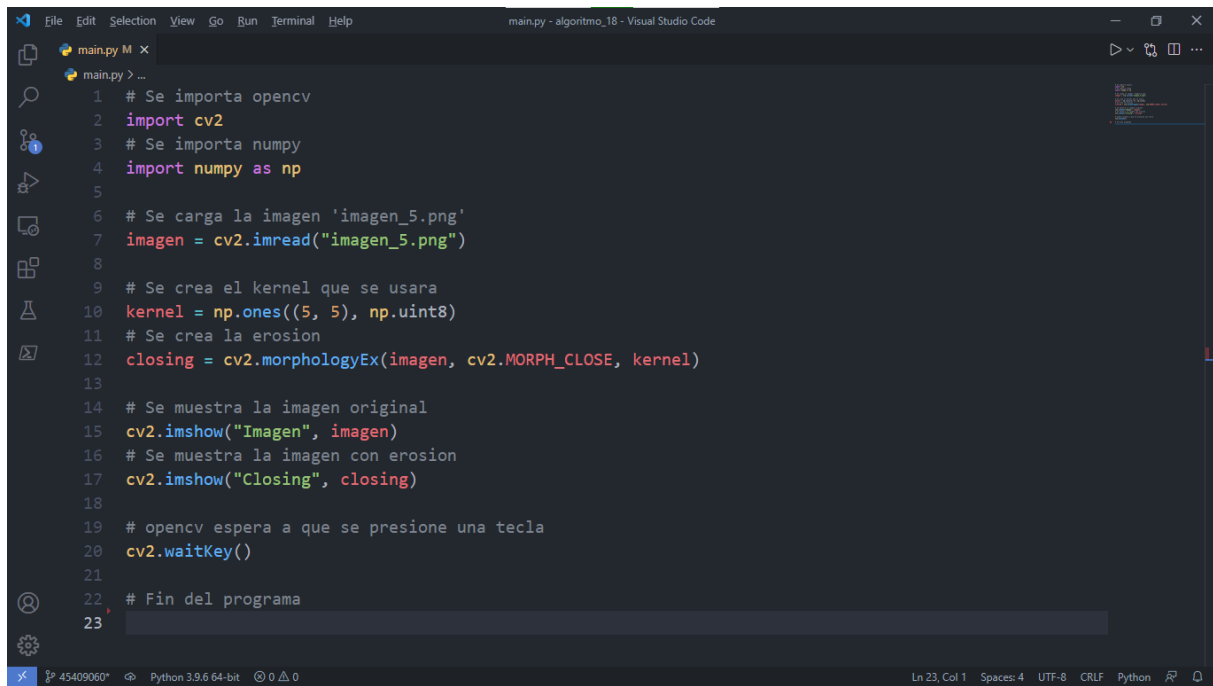
# Se crea el kernel que se usara
kernel = np.ones((5, 5), np.uint8)
# Se crea la erosion
closing = cv2.morphologyEx(imagen, cv2.MORPH_CLOSE, kernel)

# Se muestra la imagen original
cv2.imshow('Imagen', imagen)
# Se muestra la imagen con erosion
cv2.imshow('Closing', closing)

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

# Fin del programa
```

Para el código del ejercicio 1 primero se importan las librerías necesarias (**opencv**, **numpy**), luego se lee la imagen "**imagen_5.png**" previamente creada, se genera el **kernel** con una matriz de unos de 5 por 5 y se genera el closing de la imagen. Por último se muestra la imagen original y la imagen obtenida. 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_5.png'
7 imagen = cv2.imread("imagen_5.png")
8
9 # Se crea el kernel que se usara
10 kernel = np.ones((5, 5), np.uint8)
11 # Se crea la erosion
12 closing = cv2.morphologyEx(imagen, cv2.MORPH_CLOSE, kernel)
13
14 # Se muestra la imagen original
15 cv2.imshow("Imagen", imagen)
16 # Se muestra la imagen con erosion
17 cv2.imshow("Closing", closing)
18
19 # opencv espera a que se presione una tecla
20 cv2.waitKey()
21
22 # Fin del programa
23
```

Figura 8.1: Captura del código del ejercicio 7

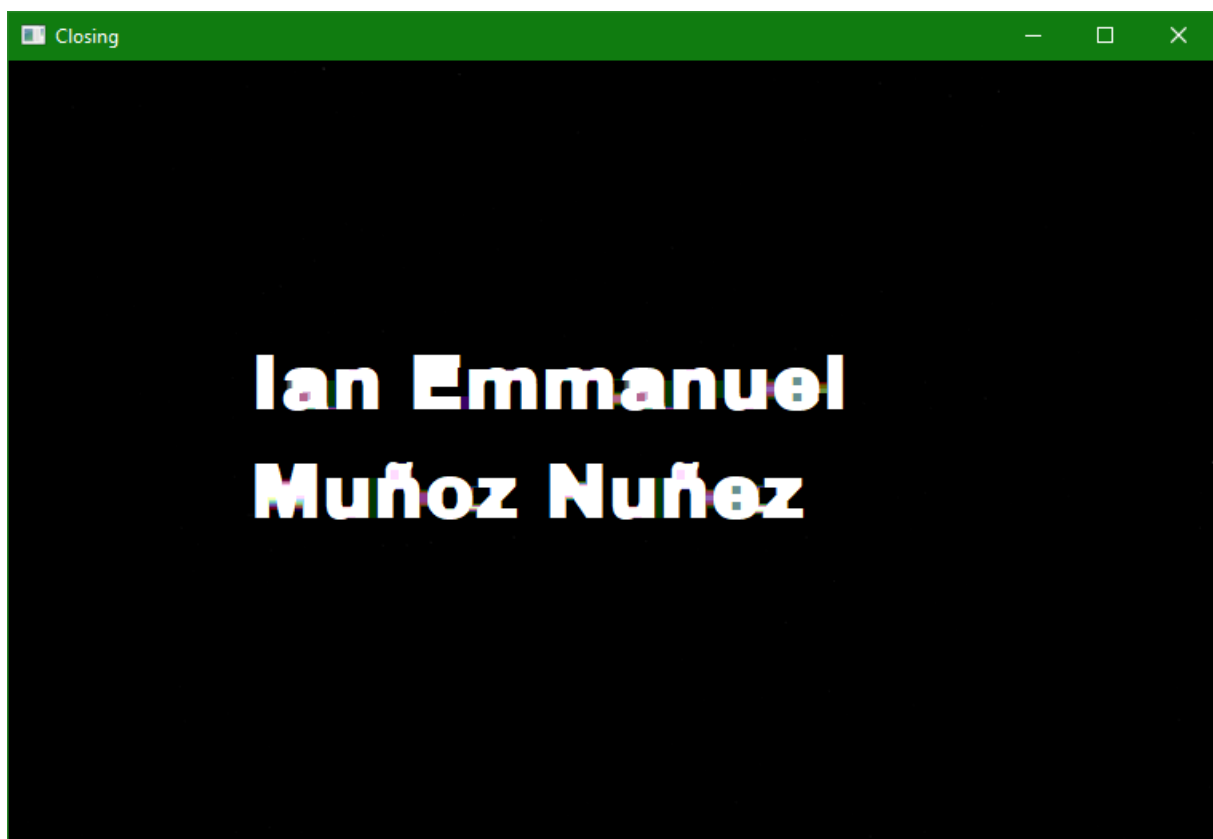


Figura 8.2: Imagen obtenida en el ejercicio 7

Capítulo 9

Ejercicio 8

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

# Se carga la imagen 'imagen_6.png'
imagen = cv2.imread('imagen_6.png')

# Se crea el kernel que se usara
kernel = np.ones((5, 5), np.uint8)
# Se crea la erosion
closing = cv2.morphologyEx(imagen, cv2.MORPH_CLOSE, kernel)

# Se muestra la imagen original
cv2.imshow('Imagen', imagen)
# Se muestra la imagen con erosion
cv2.imshow('Closing', closing)

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

# Fin del programa
```

Para el código del ejercicio 1 primero se importan las librerías necesarias (**opencv**, **numpy**), luego se lee la imagen "**imagen_6.png**" previamente creada, se genera el **kernel** con una matriz de unos de 5 por 5 y se genera el closing de la imagen. Por último se muestra la imagen original y la imagen obtenida. Al final **opencv** espera a que el usuario presione una tecla para terminar el programa.

```
File Edit Selection View Go Run Terminal Help
main.py - algoritmo_18 - Visual Studio Code
main.py M x
main.py > ...
1 # Se importa opencv
2 import cv2
3 # Se importa numpy
4 import numpy as np
5
6 # Se carga la imagen 'imagen_6.png'
7 imagen = cv2.imread("imagen_6.png")
8
9 # Se crea el kernel que se usara
10 kernel = np.ones((5, 5), np.uint8)
11 # Se crea la erosion
12 closing = cv2.morphologyEx(imagen, cv2.MORPH_CLOSE, kernel)
13
14 # Se muestra la imagen original
15 cv2.imshow("Imagen", imagen)
16 # Se muestra la imagen con erosion
17 cv2.imshow("Closing", closing)
18
19 # opencv espera a que se presione una tecla
20 cv2.waitKey()
21
22 # Fin del programa
23
```

Figura 9.1: Captura del código del ejercicio 8

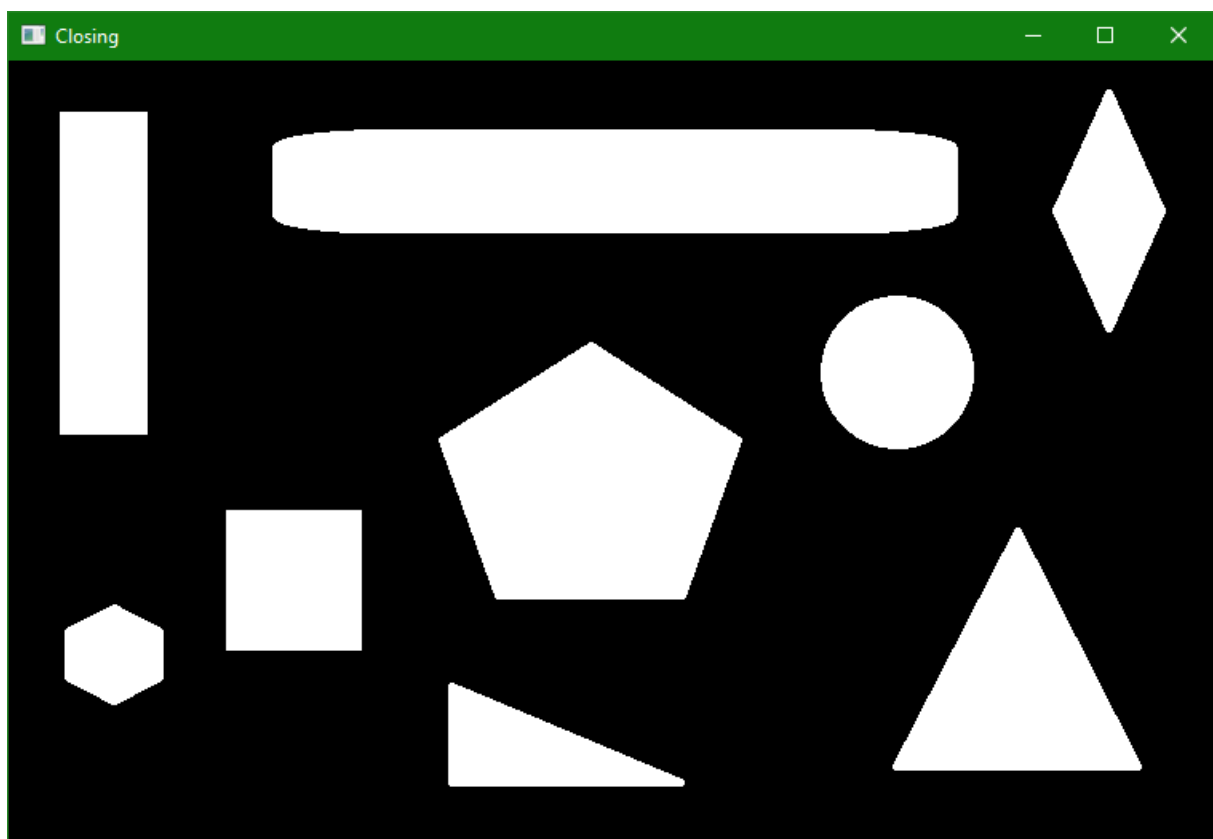


Figura 9.2: Imagen obtenida en el ejercicio 8