

Universidad de Guadalajara



Algoritmo 8

Muñoz Nuñez Ian Emmanuel

Visión Robótica

"Cambio de fondo de imagen"

Índice general

1. Ejercicio 1	2
1.1. Código	2
1.2. Capturas del código	3
1.3. Imágenes obtenidas	4
2. Ejercicio 2 (Spider-man)	7
2.1. Código	7
2.2. Capturas del código	8
2.3. Imágenes obtenidas	9
3. Ejercicio 2 (Deadpool)	12
3.1. Código	12
3.2. Capturas del código	13
3.3. Imágenes obtenidas	14
4. Ejercicio 2 (Cebra)	17
4.1. Código	17
4.2. Capturas del código	18
4.3. Imágenes obtenidas	19

Capítulo 1

Ejercicio 1

1.1. Código

```
# Se importa opencv
import cv2

# Se carga la imagen 'sonic.jpg'
imagen = cv2.imread("sonic.jpg")
# Se modifica el tamaño de la imagen
imagen = cv2.resize(imagen, None, fx=0.5, fy=0.5)

# Se carga el fondo 'rotonda.jpg'
fondo = cv2.imread("rotonda.jpg")
# Se modifica el tamaño del fondo para que sea igual al de la imagen
fondo = cv2.resize(fondo, (imagen.shape[1], imagen.shape[0]))

# Se modifica el formato de la imagen de BGR a HSV
imagen_hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
# Se calcula la mascara de la imagen
mask = cv2.inRange(imagen_hsv, (55, 100, 100), (65, 255, 255))
# Se calcula el negativo de la mascara
mask_negativo = cv2.bitwise_not(mask)

# Se aplica la mascara a la imagen
imagen_mask = cv2.bitwise_and(imagen, imagen, mask=mask_negativo)

# Se aplica la mascara al fondo
fondo_mask = cv2.bitwise_and(fondo, fondo, mask=mask)
# Se mezclan el fondo y la imagen
imagen_fondo = cv2.bitwise_or(imagen_mask, fondo_mask)

# Se muestra la imagen con opencv
cv2.imshow("Imagen", imagen)
# Se muestra la mascara con opencv
cv2.imshow("Mascara", mask)
```

```

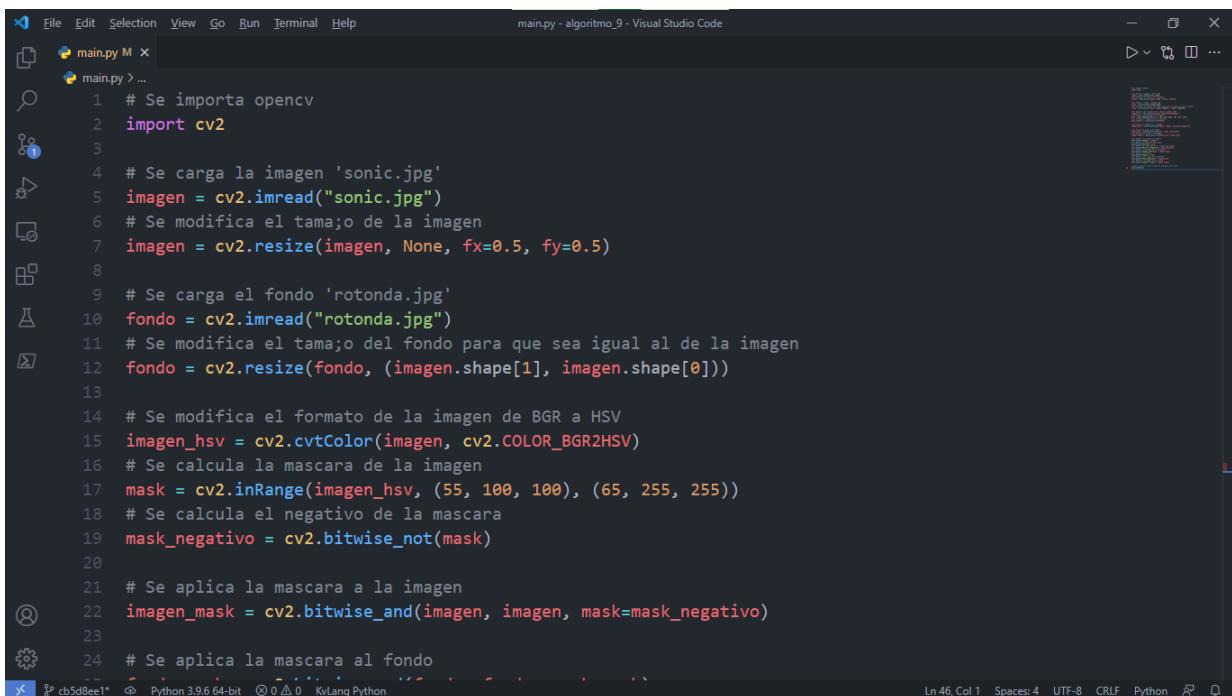
# Se muestra el negativo de la imagen con opencv
cv2.imshow("Mascara (Negativo)", mask_negativo)
# Se muestra la imagen con la mascara
cv2.imshow("Imagen (Mascara)", imagen_mask)
# Se muestra el fondo
cv2.imshow("Fondo", fondo)
# Se muestra el fondo con la mascara
cv2.imshow("Fondo (Mascara)", fondo_mask)
# Se muestra la imagen con el fondo
cv2.imshow("Imagen (Fondo)", imagen_fondo)

# Opencv espera a que el usuario presione una tecla
cv2.waitKey(0)

```

Para el código del ejercicio 1 primero se importa la librería *opencv*, y se cargan las imágenes *"sonic.jpg"* y *"rotonda.jpg"*. Se modifica el tamaño de la imagen *"sonic.jpg"* y se ajusta el tamaño de la imagen *"rotonda.jpg"* para que sean del mismo tamaño. Luego se modifica el formato de la imagen *"sonic.jpg"* de BGR a HSV, se calcula su máscara y la negativa de esta. Después se aplica el negativo de la máscara a la imagen *"sonic.jpg"* y la máscara a la imagen *"rotonda.jpg"*. Por último se mezclan la imagen y el fondo con la función *bitwise_or()* y se muestran todas las imágenes creadas. Al final *opencv* espera a que el usuario presione una tecla para terminar el programa.

1.2. Capturas del código

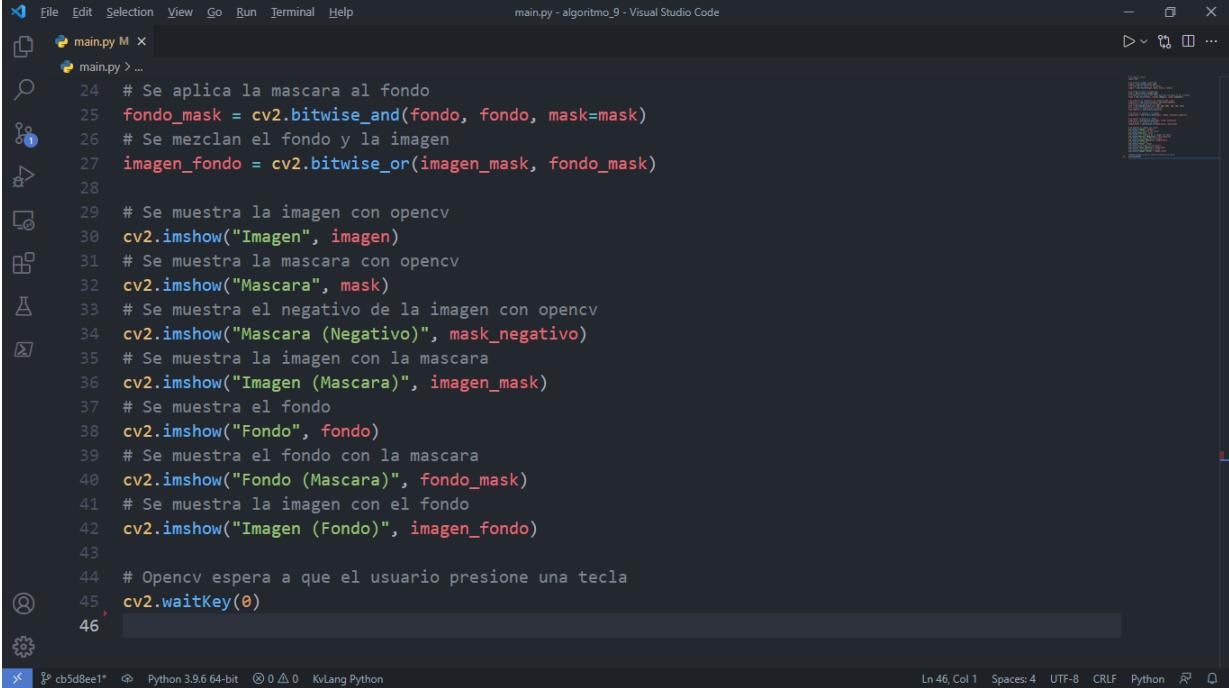


```

File Edit Selection View Go Run Terminal Help main.py - algoritmo_9 - Visual Studio Code
main.py M x
main.py > ...
1 # Se importa opencv
2 import cv2
3
4 # Se carga la imagen 'sonic.jpg'
5 imagen = cv2.imread("sonic.jpg")
6 # Se modifica el tamaño de la imagen
7 imagen = cv2.resize(imagen, None, fx=0.5, fy=0.5)
8
9 # Se carga el fondo 'rotonda.jpg'
10 fondo = cv2.imread("rotonda.jpg")
11 # Se modifica el tamaño del fondo para que sea igual al de la imagen
12 fondo = cv2.resize(fondo, (imagen.shape[1], imagen.shape[0]))
13
14 # Se modifica el formato de la imagen de BGR a HSV
15 imagen_hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
16 # Se calcula la máscara de la imagen
17 mask = cv2.inRange(imagen_hsv, (55, 100, 100), (65, 255, 255))
18 # Se calcula el negativo de la máscara
19 mask_negativo = cv2.bitwise_not(mask)
20
21 # Se aplica la máscara a la imagen
22 imagen_mask = cv2.bitwise_and(imagen, imagen, mask=mask_negativo)
23
24 # Se aplica la máscara al fondo

```

Figura 1.1: Captura 1 del código del ejercicio 1



The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar has a tree view with 'main.py M' and 'main.py > ...'. The main editor area contains the following Python code:

```
24 # Se aplica la mascara al fondo
25 fondo_mask = cv2.bitwise_and(fondo, fondo, mask=mask)
26 # Se mezclan el fondo y la imagen
27 imagen_fondo = cv2.bitwise_or(imagen_mask, fondo_mask)
28
29 # Se muestra la imagen con opencv
30 cv2.imshow("Imagen", imagen)
31 # Se muestra la mascara con opencv
32 cv2.imshow("Mascara", mask)
33 # Se muestra el negativo de la imagen con opencv
34 cv2.imshow("Mascara (Negativo)", mask_negativo)
35 # Se muestra la imagen con la mascara
36 cv2.imshow("Imagen (Mascara)", imagen_mask)
37 # Se muestra el fondo
38 cv2.imshow("Fondo", fondo)
39 # Se muestra el fondo con la mascara
40 cv2.imshow("Fondo (Mascara)", fondo_mask)
41 # Se muestra la imagen con el fondo
42 cv2.imshow("Imagen (Fondo)", imagen_fondo)
43
44 # Opencv espera a que el usuario presione una tecla
45 cv2.waitKey(0)
46
```

The status bar at the bottom indicates: cb5d8ee1*, Python 3.9.6 64-bit, 0 △ 0, KvLang Python, Ln 46, Col 1, Spaces:4, UTF-8, CRLF, Python.

Figura 1.2: Captura 2 del código del ejercicio 1

1.3. Imágenes obtenidas

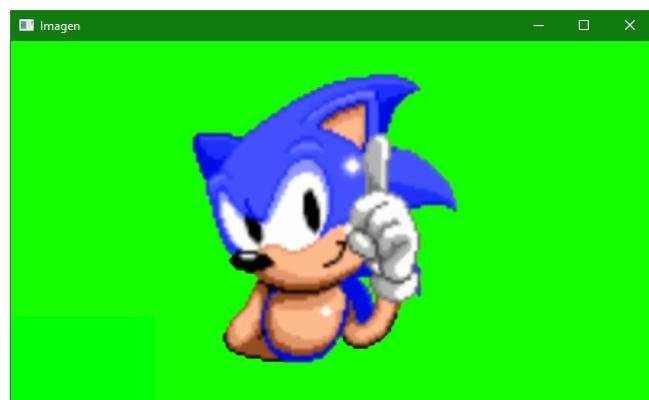


Figura 1.3: Imagen original

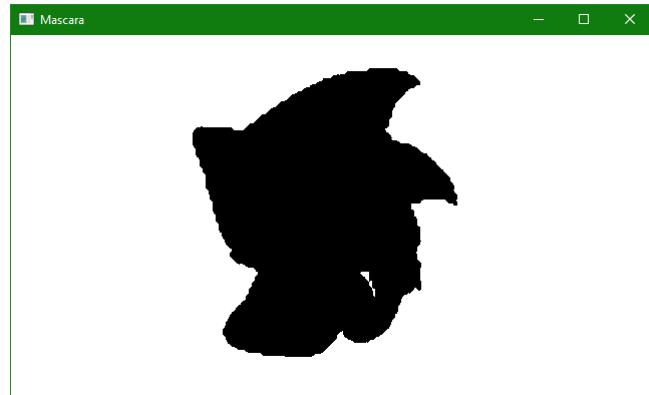


Figura 1.4: Máscara



Figura 1.5: Negativo de la máscara



Figura 1.6: Máscara aplicada a la imagen



Figura 1.7: Fondo

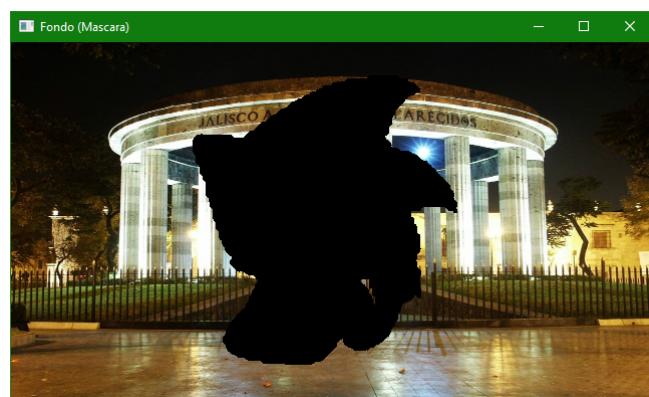


Figura 1.8: Máscara aplicada al fondo



Figura 1.9: Imagen y fondo

Capítulo 2

Ejercicio 2 (Spider-man)

2.1. Código

```
# Se importa opencv
import cv2

# Se carga la imagen 'spiderman.jpg'
imagen = cv2.imread("spiderman.jpg")
# Se modifica el tamaño de la imagen
imagen = cv2.resize(imagen, None, fx=0.5, fy=0.5)

# Se carga el fondo 'matute.jpg'
fondo = cv2.imread("matute.jpg")
# Se modifica el tamaño del fondo para que sea igual al de la imagen
fondo = cv2.resize(fondo, (imagen.shape[1], imagen.shape[0]))

# Se modifica el formato de la imagen de BGR a HSV
imagen_hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
# Se calcula la mascara de la imagen
mask = cv2.inRange(imagen_hsv, (55, 100, 100), (65, 255, 255))
# Se calcula el negativo de la mascara
mask_negativo = cv2.bitwise_not(mask)

# Se aplica la mascara a la imagen
imagen_mask = cv2.bitwise_and(imagen, imagen, mask=mask_negativo)

# Se aplica la mascara al fondo
fondo_mask = cv2.bitwise_and(fondo, fondo, mask=mask)
# Se mezclan el fondo y la imagen
imagen_fondo = cv2.bitwise_or(imagen_mask, fondo_mask)

# Se muestra la imagen con opencv
cv2.imshow("Imagen", imagen)
# Se muestra la mascara con opencv
cv2.imshow("Mascara", mask)
```

```

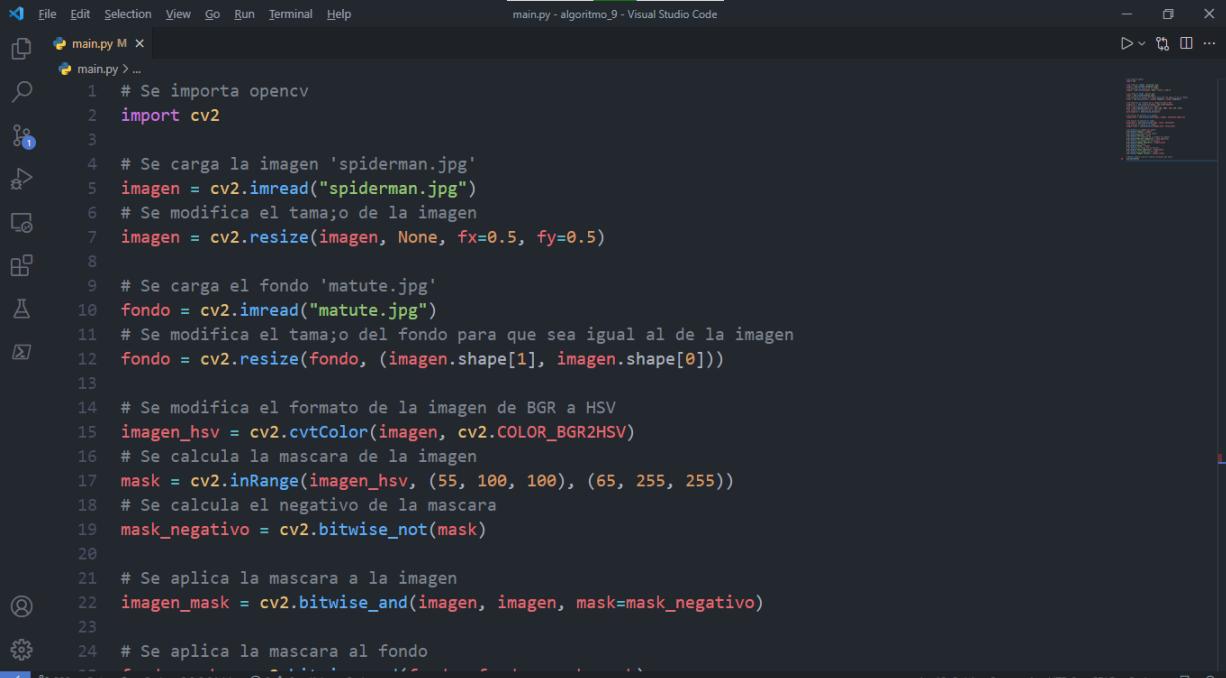
# Se muestra el negativo de la imagen con opencv
cv2.imshow("Mascara (Negativo)", mask_negativo)
# Se muestra la imagen con la mascara
cv2.imshow("Imagen (Mascara)", imagen_mask)
# Se muestra el fondo
cv2.imshow("Fondo", fondo)
# Se muestra el fondo con la mascara
cv2.imshow("Fondo (Mascara)", fondo_mask)
# Se muestra la imagen con el fondo
cv2.imshow("Imagen (Fondo)", imagen_fondo)

# Opencv espera a que el usuario presione una tecla
cv2.waitKey(0)

```

Para el código del ejercicio 2 con la imagen "*Spider-man*" primero se importa la librería *opencv*, y se cargan las imágenes "*spiderman.jpg*" y "*matute.jpg*". Se modifica el tamaño de la imagen "*spiderman.jpg*" y se ajusta el tamaño de la imagen "*matute.jpg*" para que sean del mismo tamaño. Luego se modifica el formato de la imagen "*spiderman.jpg*" de BGR a HSV, se calcula su máscara y la negativa de esta. Despues se aplica el negativo de la máscara a la imagen "*spiderman.jpg*" y la máscara a la imagen "*matute.jpg*". Por ultimo se mezclan la imagen y el fondo con la función *bitwise_or()* y se muestran todas las imágenes creadas. Al final *opencv* espera a que el usuario presione una tecla para terminar el programa.

2.2. Capturas del código



```

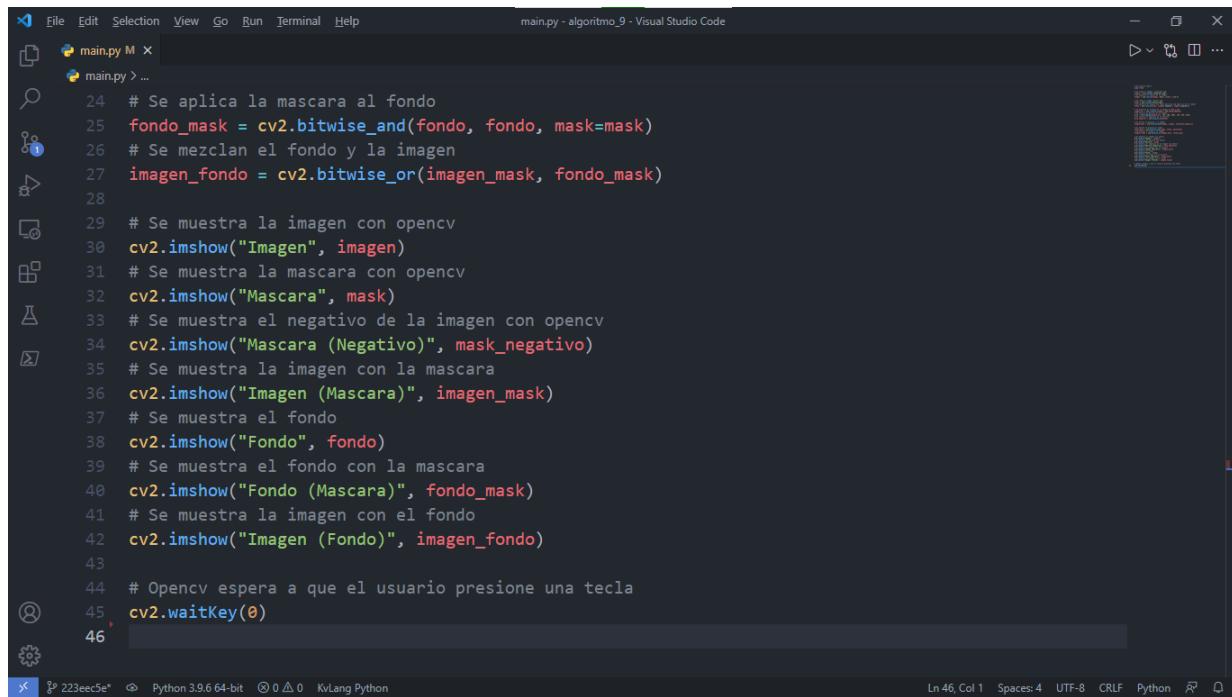
File Edit Selection View Go Run Terminal Help
main.py - algoritmo_9 - Visual Studio Code

main.py M
main.py > ...
1 # Se importa opencv
2 import cv2
3
4 # Se carga la imagen 'spiderman.jpg'
5 imagen = cv2.imread("spiderman.jpg")
6 # Se modifica el tamaño de la imagen
7 imagen = cv2.resize(imagen, None, fx=0.5, fy=0.5)
8
9 # Se carga el fondo 'matute.jpg'
10 fondo = cv2.imread("matute.jpg")
11 # Se modifica el tamaño del fondo para que sea igual al de la imagen
12 fondo = cv2.resize(fondo, (imagen.shape[1], imagen.shape[0]))
13
14 # Se modifica el formato de la imagen de BGR a HSV
15 imagen_hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
16 # Se calcula la mascara de la imagen
17 mask = cv2.inRange(imagen_hsv, (55, 100, 100), (65, 255, 255))
18 # Se calcula el negativo de la mascara
19 mask_negativo = cv2.bitwise_not(mask)
20
21 # Se aplica la mascara a la imagen
22 imagen_mask = cv2.bitwise_and(imagen, imagen, mask=mask_negativo)
23
24 # Se aplica la mascara al fondo

```

Ln 46, Col 1 Spaces: 4 UFT-8 CRLF Python

Figura 2.1: Captura 1 del código del ejercicio 2 con la imagen Spider-man



The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** main.py - algoritmo_9 - Visual Studio Code
- File Explorer:** Shows two files: main.py M and main.py > ...
- Code Editor:** Displays the Python code for changing the background of an image using OpenCV. The code includes comments explaining each step: applying a mask to the background, merging the background and image, displaying the result, and showing the mask and its negative.
- Status Bar:** Shows file information (223eec5e*, Python 3.9.6 64-bit), code statistics (0 △ 0), and encoding (UTF-8 CRLF Python).
- Bottom Status:** Shows Ln 46, Col 1, Spaces:4, and a refresh icon.

Figura 2.2: Captura 2 del código del ejercicio 2 con la imagen Spider-man

2.3. Imágenes obtenidas

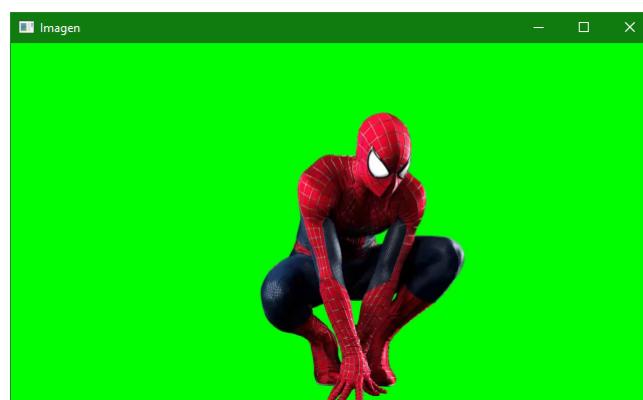


Figura 2.3: Imagen original

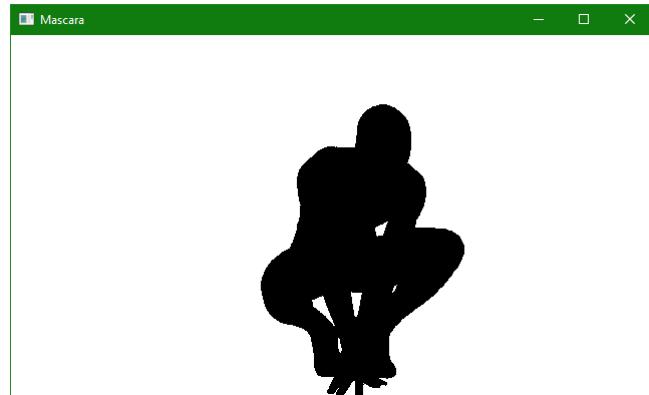


Figura 2.4: Máscara

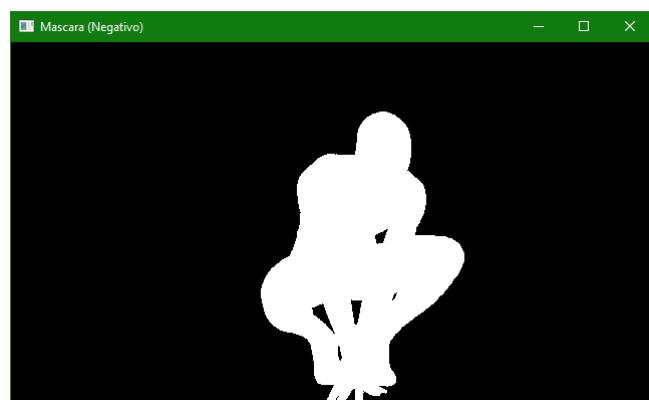


Figura 2.5: Negativo de la máscara



Figura 2.6: Máscara aplicada a la imagen

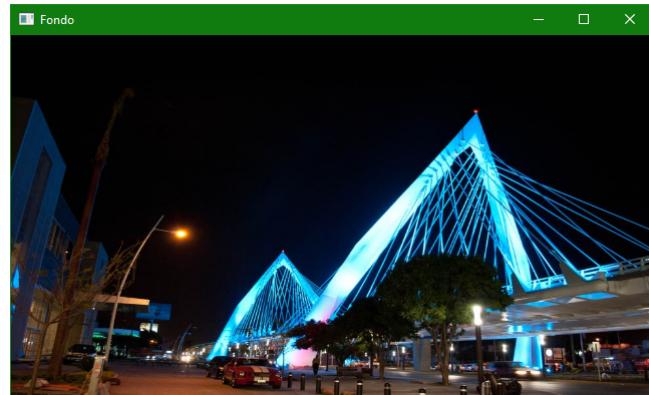


Figura 2.7: Fondo

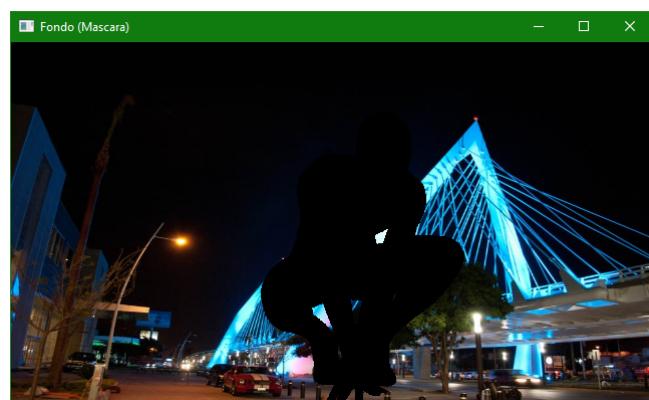


Figura 2.8: Máscara aplicada al fondo

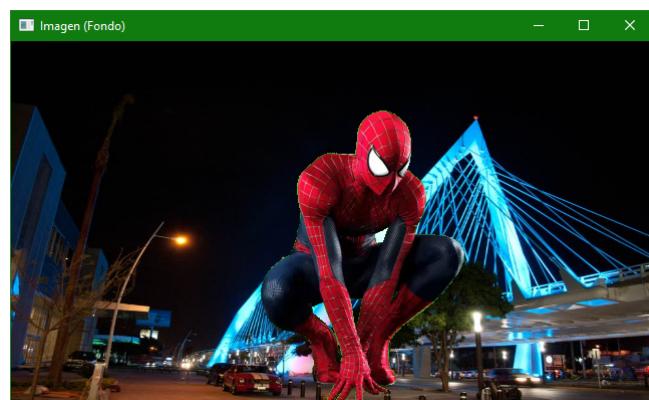


Figura 2.9: Imagen y fondo

Capítulo 3

Ejercicio 2 (Deadpool)

3.1. Código

```
# Se importa opencv
import cv2

# Se carga la imagen 'deadpool.jpg'
imagen = cv2.imread("deadpool.jpg")
# Se modifica el tamaño de la imagen
imagen = cv2.resize(imagen, None, fx=0.5, fy=0.5)

# Se carga el fondo 'degollado.jpg'
fondo = cv2.imread("degollado.jpg")
# Se modifica el tamaño del fondo para que sea igual al de la imagen
fondo = cv2.resize(fondo, (imagen.shape[1], imagen.shape[0]))

# Se modifica el formato de la imagen de BGR a HSV
imagen_hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
# Se calcula la mascara de la imagen
mask = cv2.inRange(imagen_hsv, (60, 50, 50), (70, 255, 255))
# Se calcula el negativo de la mascara
mask_negativo = cv2.bitwise_not(mask)

# Se aplica la mascara a la imagen
imagen_mask = cv2.bitwise_and(imagen, imagen, mask=mask_negativo)

# Se aplica la mascara al fondo
fondo_mask = cv2.bitwise_and(fondo, fondo, mask=mask)
# Se mezclan el fondo y la imagen
imagen_fondo = cv2.bitwise_or(imagen_mask, fondo_mask)

# Se muestra la imagen con opencv
cv2.imshow("Imagen", imagen)
# Se muestra la mascara con opencv
cv2.imshow("Mascara", mask)
```

```

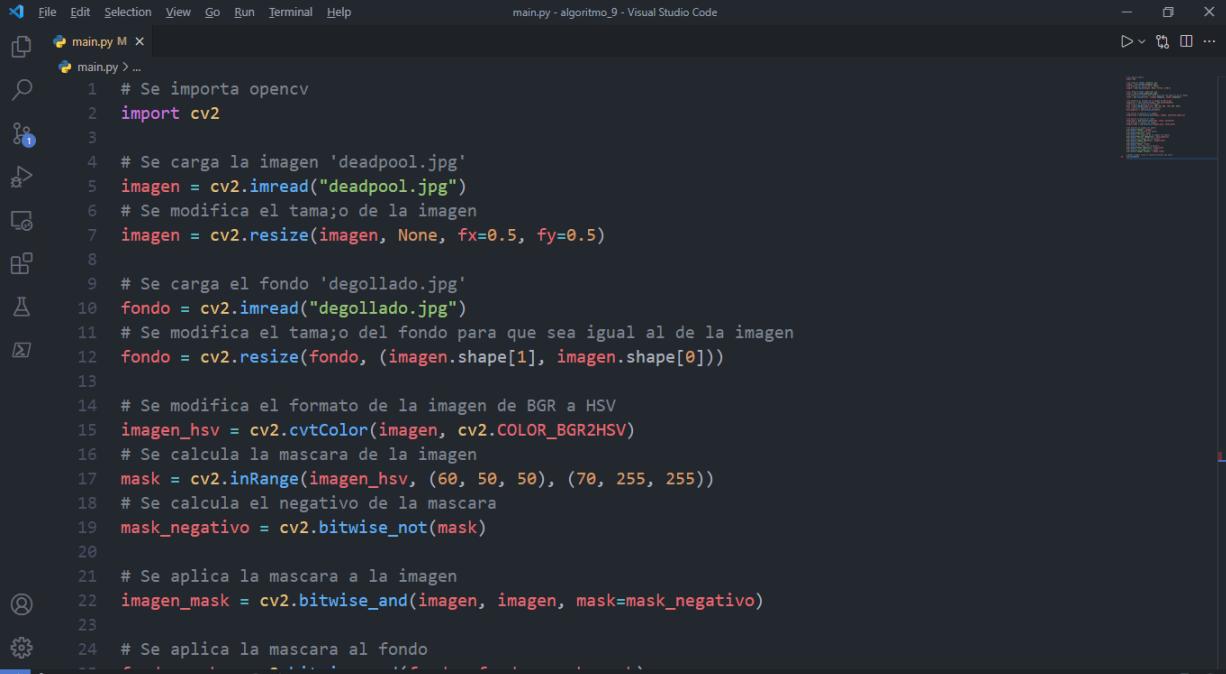
# Se muestra el negativo de la imagen con opencv
cv2.imshow("Mascara (Negativo)", mask_negativo)
# Se muestra la imagen con la mascara
cv2.imshow("Imagen (Mascara)", imagen_mask)
# Se muestra el fondo
cv2.imshow("Fondo", fondo)
# Se muestra el fondo con la mascara
cv2.imshow("Fondo (Mascara)", fondo_mask)
# Se muestra la imagen con el fondo
cv2.imshow("Imagen (Fondo)", imagen_fondo)

# Opencv espera a que el usuario presione una tecla
cv2.waitKey(0)

```

Para el código del ejercicio 2 con la imagen "*Deadpool*" primero se importa la librería *opencv*, y se cargan las imágenes "*deadpool.jpg*" y "*degollado.jpg*". Se modifica el tamaño de la imagen "*deadpool.jpg*" y se ajusta el tamaño de la imagen "*degollado.jpg*" para que sean del mismo tamaño. Luego se modifica el formato de la imagen "*deadpool.jpg*" de BGR a HSV, se calcula su máscara y la negativa de esta. Después se aplica el negativo de la máscara a la imagen "*deadpool.jpg*" y la máscara a la imagen "*degollado.jpg*". Por último se mezclan la imagen y el fondo con la función *bitwise_or()* y se muestran todas las imágenes creadas. Al final *opencv* espera a que el usuario presione una tecla para terminar el programa.

3.2. Capturas del código



```

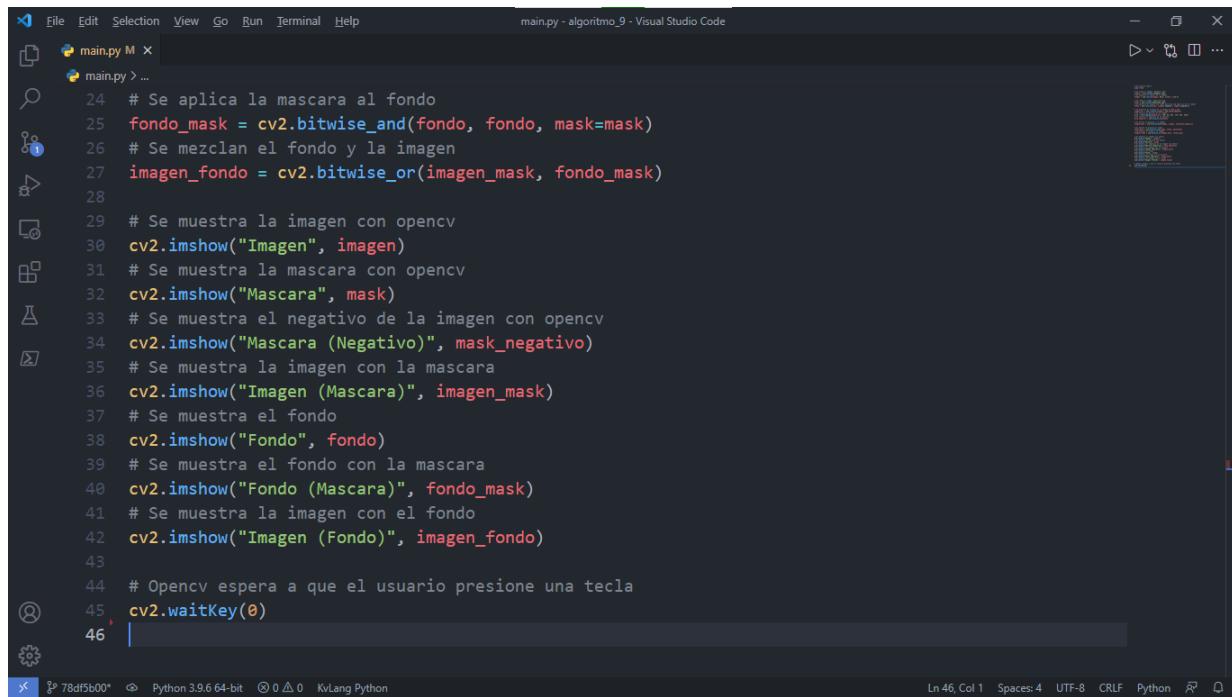
File Edit Selection View Go Run Terminal Help
main.py - algoritmo_9 - Visual Studio Code

main.py M
main.py > ...
1 # Se importa opencv
2 import cv2
3
4 # Se carga la imagen 'deadpool.jpg'
5 imagen = cv2.imread("deadpool.jpg")
6 # Se modifica el tamaño de la imagen
7 imagen = cv2.resize(imagen, None, fx=0.5, fy=0.5)
8
9 # Se carga el fondo 'degollado.jpg'
10 fondo = cv2.imread("degollado.jpg")
11 # Se modifica el tamaño del fondo para que sea igual al de la imagen
12 fondo = cv2.resize(fondo, (imagen.shape[1], imagen.shape[0]))
13
14 # Se modifica el formato de la imagen de BGR a HSV
15 imagen_hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
16 # Se calcula la mascara de la imagen
17 mask = cv2.inRange(imagen_hsv, (60, 50, 50), (70, 255, 255))
18 # Se calcula el negativo de la mascara
19 mask_negativo = cv2.bitwise_not(mask)
20
21 # Se aplica la mascara a la imagen
22 imagen_mask = cv2.bitwise_and(imagen, imagen, mask=mask_negativo)
23
24 # Se aplica la mascara al fondo

```

The screenshot shows the Visual Studio Code interface with the file "main.py" open. The code implements the steps described in the text to create a masked image and blend it with a background image. The code uses OpenCV functions like cv2.imread, cv2.resize, cv2.cvtColor, cv2.inRange, cv2.bitwise_not, and cv2.bitwise_and. The status bar at the bottom indicates the file is 78df5b00*, Python 3.9.6 64-bit, and Kivy Python.

Figura 3.1: Captura 1 del código del ejercicio 2 con la imagen Deadpool



The screenshot shows a Visual Studio Code interface with a dark theme. The main area displays a Python script named 'main.py' with the following code:

```
main.py M | main.py > ...
24 # Se aplica la mascara al fondo
25 fondo_mask = cv2.bitwise_and(fondo, fondo, mask=mask)
26 # Se mezclan el fondo y la imagen
27 imagen_fondo = cv2.bitwise_or(imagen_mask, fondo_mask)
28
29 # Se muestra la imagen con opencv
30 cv2.imshow("Imagen", imagen)
31 # Se muestra la mascara con opencv
32 cv2.imshow("Mascara", mask)
33 # Se muestra el negativo de la imagen con opencv
34 cv2.imshow("Mascara (Negativo)", mask_negativo)
35 # Se muestra la imagen con la mascara
36 cv2.imshow("Imagen (Mascara)", imagen_mask)
37 # Se muestra el fondo
38 cv2.imshow("Fondo", fondo)
39 # Se muestra el fondo con la mascara
40 cv2.imshow("Fondo (Mascara)", fondo_mask)
41 # Se muestra la imagen con el fondo
42 cv2.imshow("Imagen (Fondo)", imagen_fondo)
43
44 # Opencv espera a que el usuario presione una tecla
45 cv2.waitKey(0)
46 |
```

The status bar at the bottom indicates the file is 78df5b00*, using Python 3.9.6 64-bit, and the code is in KvLang Python.

Figura 3.2: Captura 2 del código del ejercicio 2 con la imagen Deadpool

3.3. Imágenes obtenidas

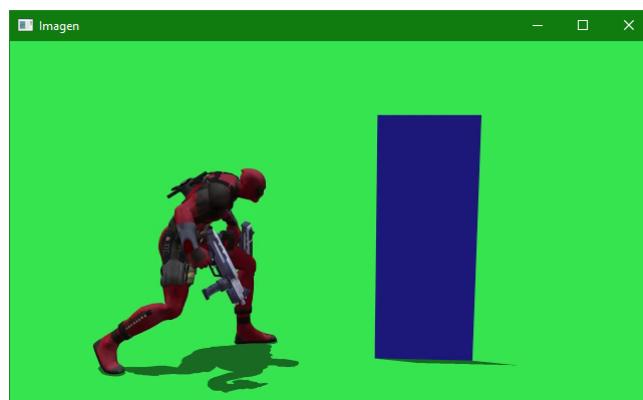


Figura 3.3: Imagen original

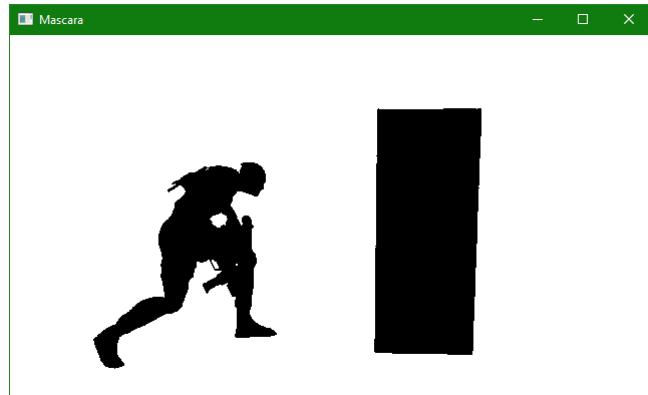


Figura 3.4: Máscara



Figura 3.5: Negativo de la máscara

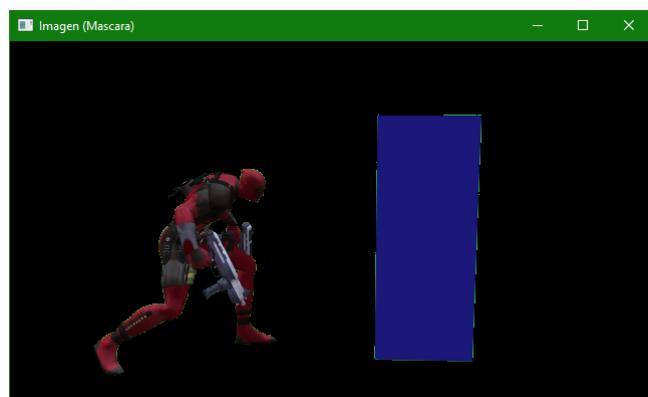


Figura 3.6: Máscara aplicada a la imagen

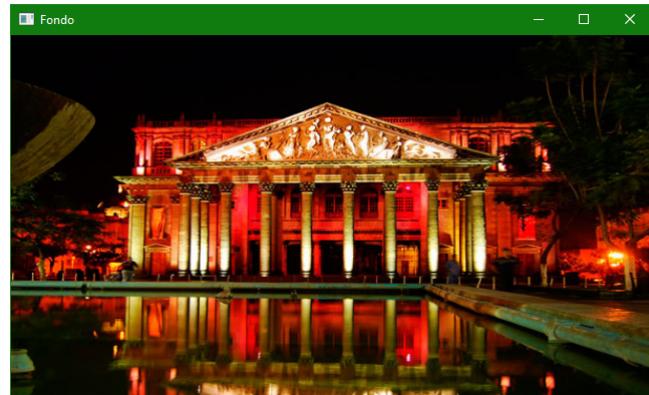


Figura 3.7: Fondo

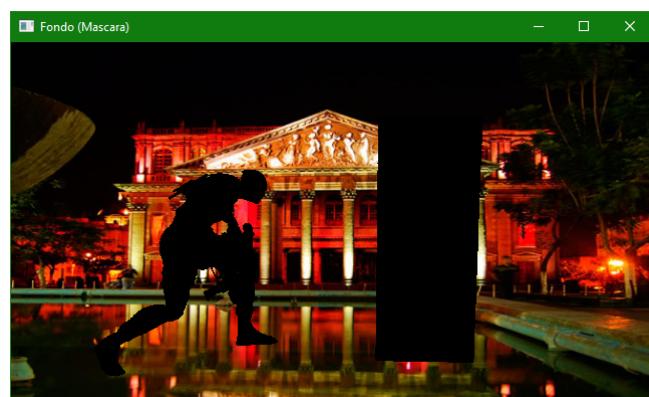


Figura 3.8: Máscara aplicada al fondo



Figura 3.9: Imagen y fondo

Capítulo 4

Ejercicio 2 (Cebra)

4.1. Código

```
# Se importa opencv
import cv2

# Se carga la imagen 'cebra.jpg'
imagen = cv2.imread("cebra.jpg")
# Se modifica el tamaño de la imagen
imagen = cv2.resize(imagen, None, fx=2, fy=2)

# Se carga el fondo 'catedral.jpg'
fondo = cv2.imread("catedral.jpg")
# Se modifica el tamaño del fondo para que sea igual al de la imagen
fondo = cv2.resize(fondo, (imagen.shape[1], imagen.shape[0]))

# Se modifica el formato de la imagen de BGR a HSV
imagen_hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
# Se calcula la mascara de la imagen
mask = cv2.inRange(imagen_hsv, (50, 180, 180), (64, 255, 255))
# Se calcula el negativo de la mascara
mask_negativo = cv2.bitwise_not(mask)

# Se aplica la mascara a la imagen
imagen_mask = cv2.bitwise_and(imagen, imagen, mask=mask_negativo)

# Se aplica la mascara al fondo
fondo_mask = cv2.bitwise_and(fondo, fondo, mask=mask)
# Se mezclan el fondo y la imagen
imagen_fondo = cv2.bitwise_or(imagen_mask, fondo_mask)

# Se muestra la imagen con opencv
cv2.imshow("Imagen", imagen)
# Se muestra la mascara con opencv
cv2.imshow("Mascara", mask)
```

```

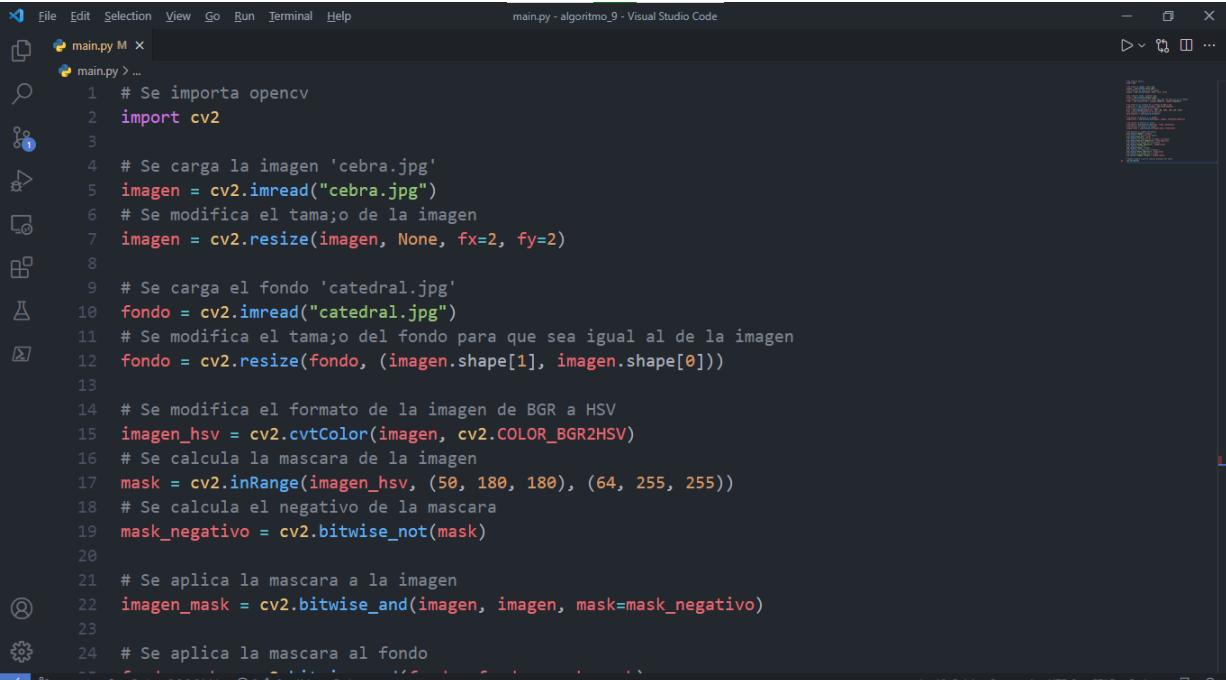
# Se muestra el negativo de la imagen con opencv
cv2.imshow("Mascara (Negativo)", mask_negativo)
# Se muestra la imagen con la mascara
cv2.imshow("Imagen (Mascara)", imagen_mask)
# Se muestra el fondo
cv2.imshow("Fondo", fondo)
# Se muestra el fondo con la mascara
cv2.imshow("Fondo (Mascara)", fondo_mask)
# Se muestra la imagen con el fondo
cv2.imshow("Imagen (Fondo)", imagen_fondo)

# Opencv espera a que el usuario presione una tecla
cv2.waitKey(0)

```

Para el código del ejercicio 2 con la imagen "*Cebra*" primero se importa la librería *opencv*, y se cargan las imágenes "*cebra.jpg*" y "*catedral.jpg*". Se modifica el tamaño de la imagen "*cebra.jpg*" y se ajusta el tamaño de la imagen "*catedral.jpg*" para que sean del mismo tamaño. Luego se modifica el formato de la imagen "*cebra.jpg*" de BGR a HSV, se calcula su máscara y la negativa de esta. Despues se aplica el negativo de la máscara a la imagen "*cebra.jpg*" y la máscara a la imagen "*catedral.jpg*". Por ultimo se mezclan la imagen y el fondo con la función *bitwise_or()* y se muestran todas las imágenes creadas. Al final *opencv* espera a que el usuario presione una tecla para terminar el programa.

4.2. Capturas del código



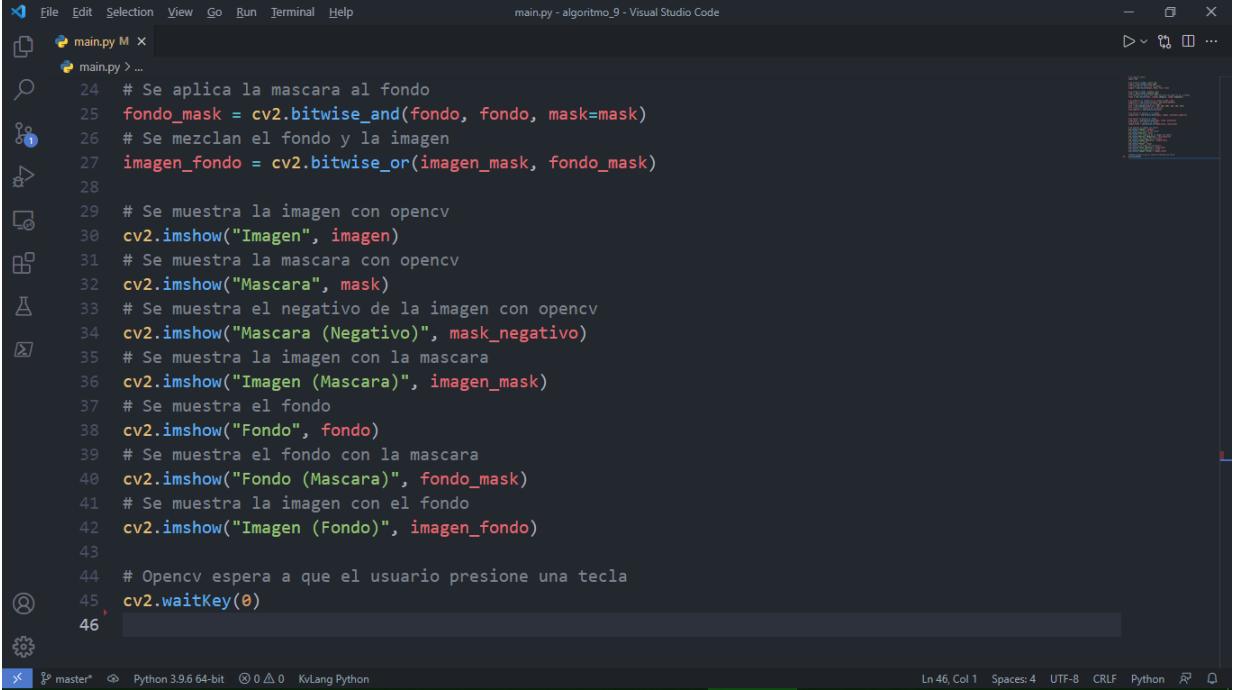
```

File Edit Selection View Go Run Terminal Help
main.py - algoritmo_9 - Visual Studio Code

main.py M
main.py > ...
1 # Se importa opencv
2 import cv2
3
4 # Se carga la imagen 'cebra.jpg'
5 imagen = cv2.imread("cebra.jpg")
6 # Se modifica el tamaño de la imagen
7 imagen = cv2.resize(imagen, None, fx=2, fy=2)
8
9 # Se carga el fondo 'catedral.jpg'
10 fondo = cv2.imread("catedral.jpg")
11 # Se modifica el tamaño del fondo para que sea igual al de la imagen
12 fondo = cv2.resize(fondo, (imagen.shape[1], imagen.shape[0]))
13
14 # Se modifica el formato de la imagen de BGR a HSV
15 imagen_hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
16 # Se calcula la mascara de la imagen
17 mask = cv2.inRange(imagen_hsv, (50, 180, 180), (64, 255, 255))
18 # Se calcula el negativo de la mascara
19 mask_negativo = cv2.bitwise_not(mask)
20
21 # Se aplica la mascara a la imagen
22 imagen_mask = cv2.bitwise_and(imagen, imagen, mask=mask_negativo)
23
24 # Se aplica la mascara al fondo

```

Figura 4.1: Captura 1 del código del ejercicio 2 con la imagen Cebra



The screenshot shows a Visual Studio Code interface with a dark theme. The left sidebar has icons for file operations like Open, Save, and Close. The main editor area contains Python code for image processing using OpenCV. The code performs several operations: applying a mask to the background, merging the background and image, displaying the result, and then displaying the mask, its negative, and the final masked image. It also shows the original background and the result of applying the mask to it. Finally, it displays the original image and waits for user input. The status bar at the bottom indicates the file is in master branch, using Python 3.9.6 64-bit, and shows line 46, column 1.

```
File Edit Selection View Go Run Terminal Help
main.py M ...
main.py > ...
24 # Se aplica la mascara al fondo
25 fondo_mask = cv2.bitwise_and(fondo, fondo, mask=mask)
26 # Se mezclan el fondo y la imagen
27 imagen_fondo = cv2.bitwise_or(imagen_mask, fondo_mask)
28
29 # Se muestra la imagen con opencv
30 cv2.imshow("Imagen", imagen)
31 # Se muestra la mascara con opencv
32 cv2.imshow("Mascara", mask)
33 # Se muestra el negativo de la imagen con opencv
34 cv2.imshow("Mascara (Negativo)", mask_negativo)
35 # Se muestra la imagen con la mascara
36 cv2.imshow("Imagen (Mascara)", imagen_mask)
37 # Se muestra el fondo
38 cv2.imshow("Fondo", fondo)
39 # Se muestra el fondo con la mascara
40 cv2.imshow("Fondo (Mascara)", fondo_mask)
41 # Se muestra la imagen con el fondo
42 cv2.imshow("Imagen (Fondo)", imagen_fondo)
43
44 # Opencv espera a que el usuario presione una tecla
45 cv2.waitKey(0)
46
```

Figura 4.2: Captura 2 del código del ejercicio 2 con la imagen Cebra

4.3. Imágenes obtenidas

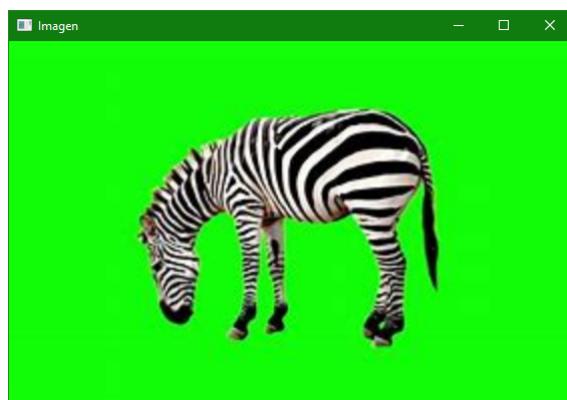


Figura 4.3: Imagen original

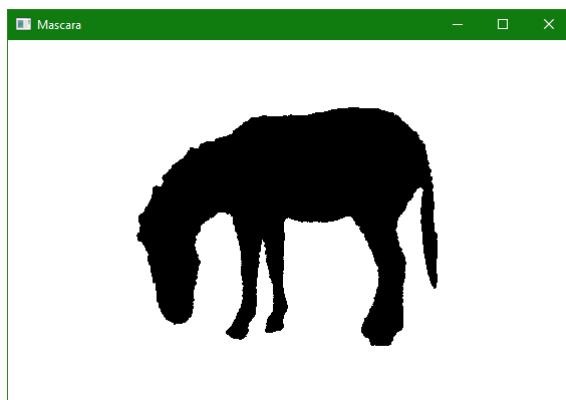


Figura 4.4: Máscara

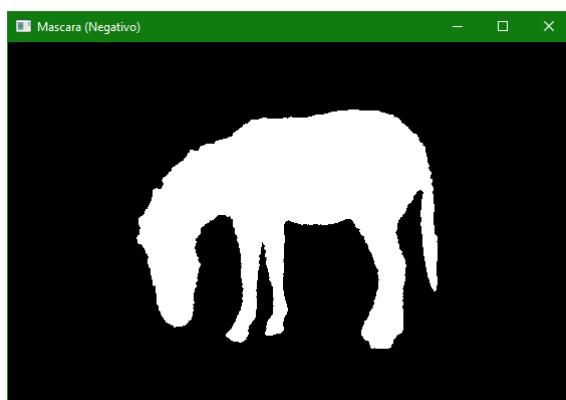


Figura 4.5: Negativo de la máscara



Figura 4.6: Máscara aplicada a la imagen

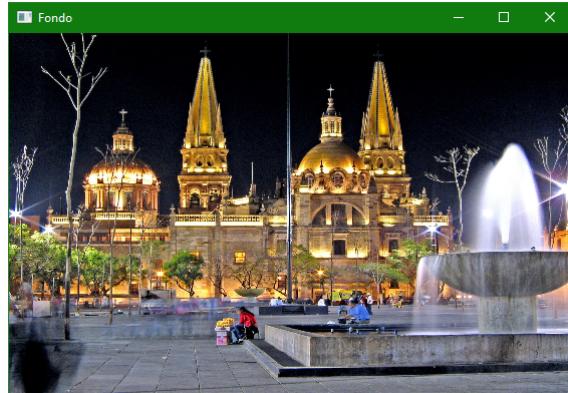


Figura 4.7: Fondo



Figura 4.8: Máscara aplicada al fondo



Figura 4.9: Imagen y fondo