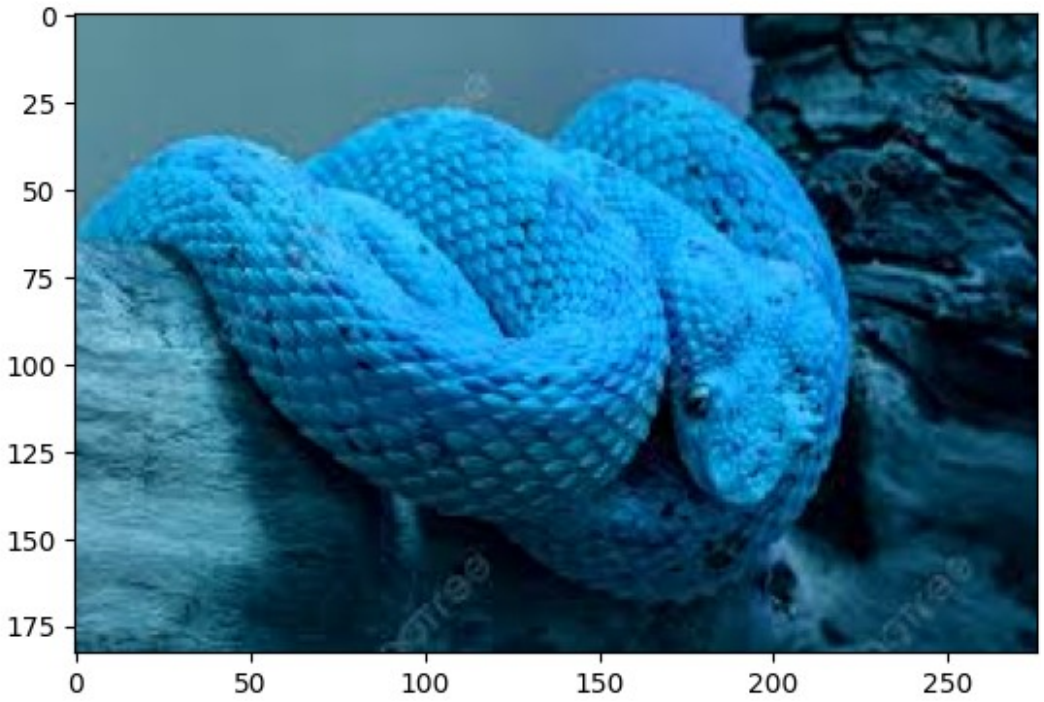
Name:=Jainish Barbhaya

Roll No:=4 EnrollNo:=230823007

Reading Image

```
import cv2
import matplotlib.pyplot as plt
img = cv2.imread('snacke.jpeg')
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x17b1616b9b0>
```



Removing Axis

```
import cv2
import matplotlib.pyplot as plt
img = cv2.imread('snacke.jpeg')
plt.imshow(img)
```

```
#to remove axis...
plt.imshow(img)
#plt.xticks([])
#plt.yticks([])
plt.axis('off')
```

```
(np.float64(-0.5), np.float64(275.5), np.float64(182.5), np.float64(-0.5))
```

Shape Of Image

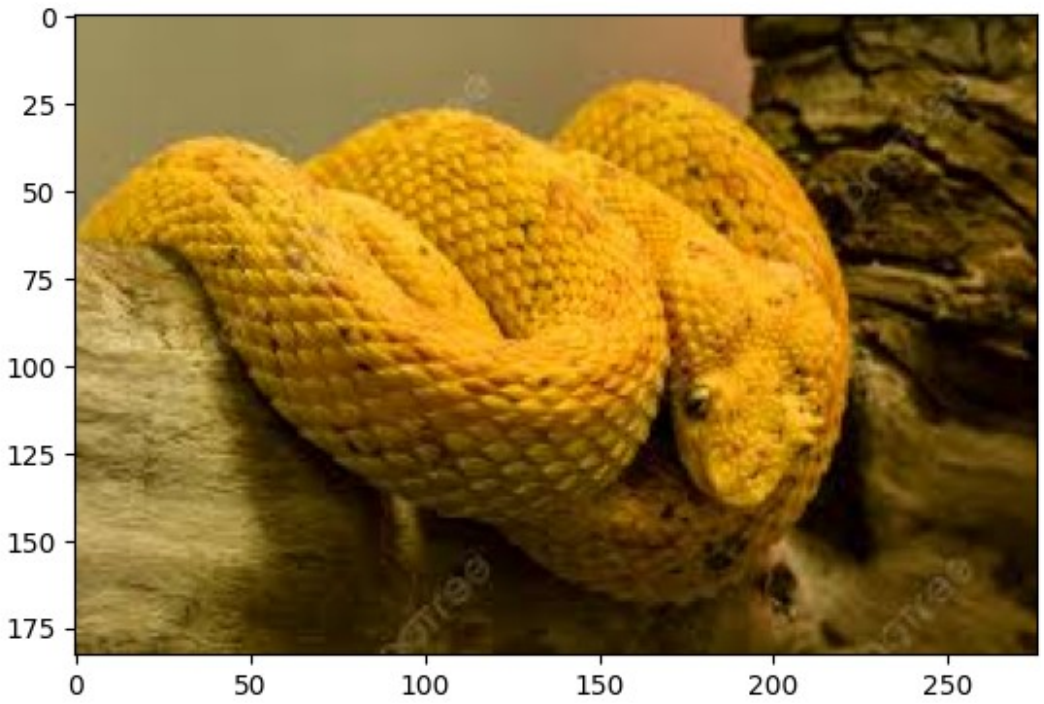
```
import cv2
import matplotlib.pyplot as plt
img = cv2.imread('snacke.jpeg')
plt.imshow(img)

#to remove axis...
plt.imshow(img)
#plt.xticks([])
#plt.yticks([])
plt.axis('off')

#to know the shape(resolution) of image
r,c=img.shape[:2]
print(r)
print(c)
```

```
183
276
```

BGR2RGB Conversion

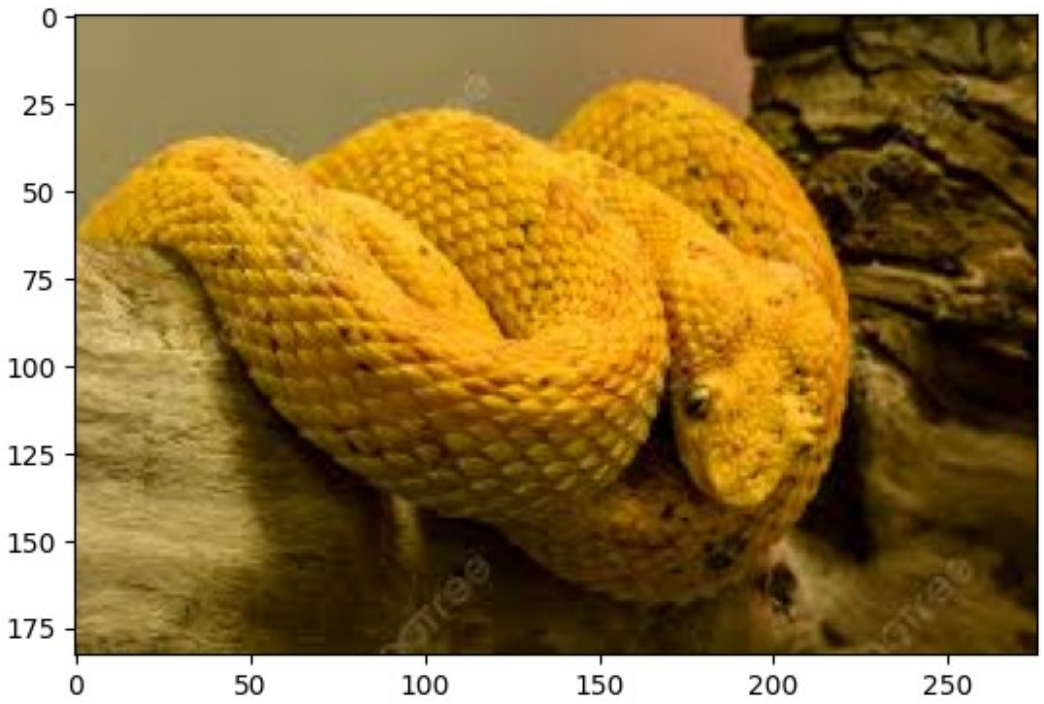
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread('snacke.jpeg')
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x1f562bca0f0>
```




```
import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread('snacke.jpeg')
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.imshow(img)
k1 = np.array([[0,0,0],[0,1,0],[0,0,0]])
k1output = cv2.filter2D(img,-1,k1)
plt.imshow(k1output)
```

```
<matplotlib.image.AxesImage at 0x1f564dd7770>
```

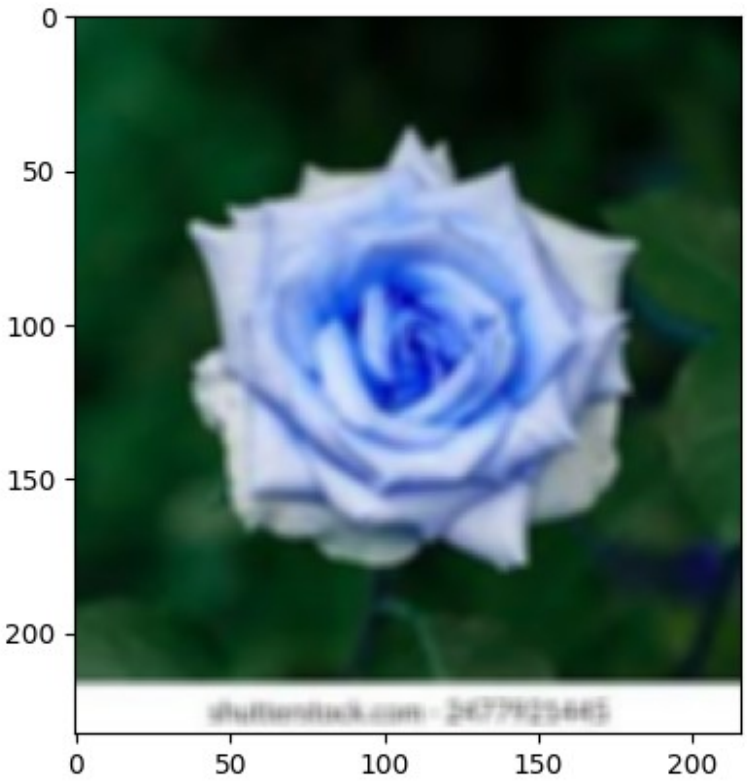



```
import cv2
import matplotlib.pyplot as plt
img=cv2.imread('flower.jpeg')
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x1f564dbfda0>
```

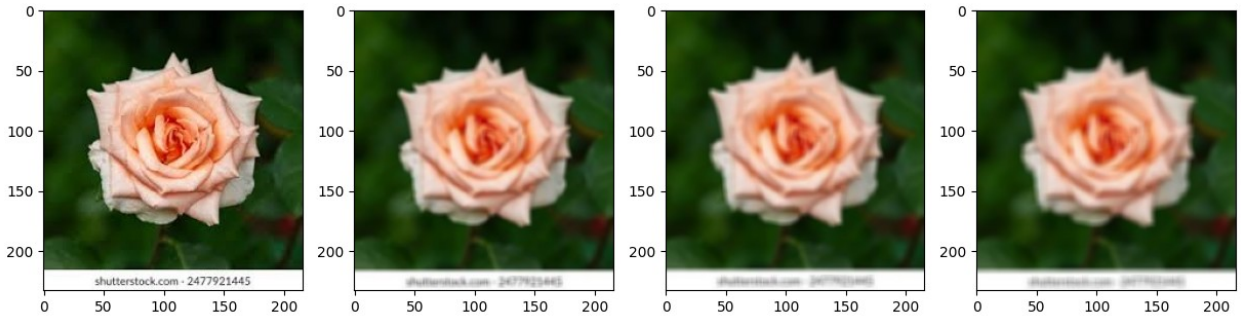

```
#Bluring
import numpy as np
k1=np.ones((4,4),np.float32)/16
img=cv2.filter2D(img,-1,k1)
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x1f562c26960>
```



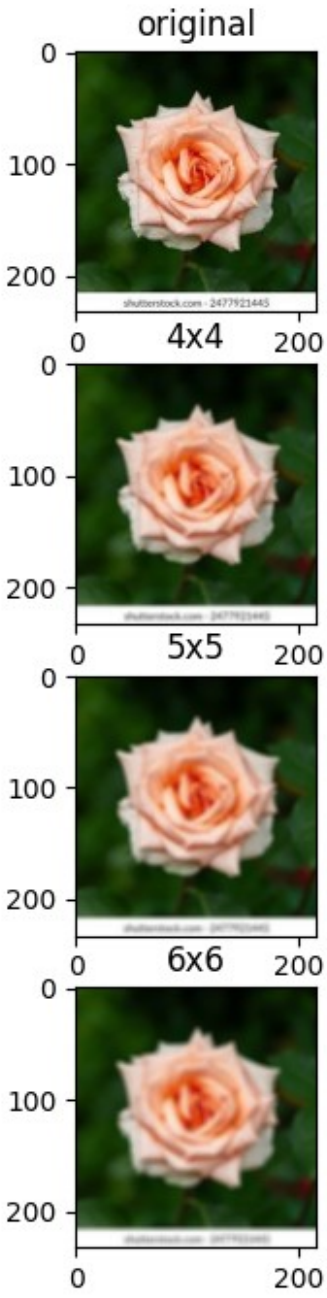
Bluring

```
from matplotlib import rcParams
img=cv2.imread('flower.jpeg')
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
k1=np.ones((4,4),np.float32)/16
k2=np.ones((5,5),np.float32)/25
k3=np.ones((6,6),np.float32)/36

img1=cv2.filter2D(img,-1,k1)
img2=cv2.filter2D(img,-1,k2)
img3=cv2.filter2D(img,-1,k3)
rcParams['figure.figsize'] = 15,8
fig,ax = plt.subplots(1,4)
ax[0].imshow(img)
ax[1].imshow(img1)
ax[2].imshow(img2)
ax[3].imshow(img3)

<matplotlib.image.AxesImage at 0x1f562c87bf0>
```



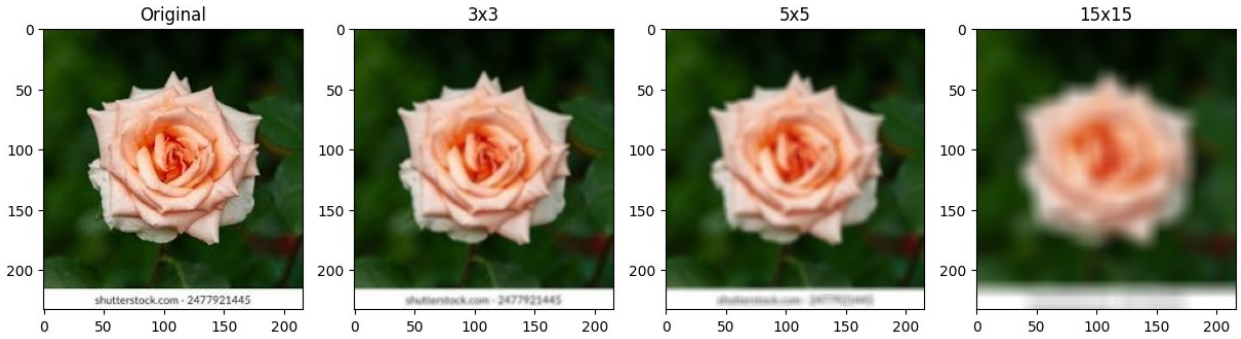

```
#from matplotlib import rcParams
img=cv2.imread('flower.jpeg')
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
k1=np.ones((4,4),np.float32)/16
k2=np.ones((5,5),np.float32)/25
k3=np.ones((6,6),np.float32)/36
img1=cv2.filter2D(img,-1,k1)
img2=cv2.filter2D(img,-1,k2)
img3=cv2.filter2D(img,-1,k3)

#rcParams['figure.figsize'] = 15,8
plt.subplot(4,1,1),plt.imshow(img),plt.title('original')
plt.subplot(4,1,2),plt.imshow(img1),plt.title('4x4')
plt.subplot(4,1,3),plt.imshow(img2),plt.title('5x5')
plt.subplot(4,1,4),plt.imshow(img3),plt.title('6x6')
```

```
(<Axes: title={'center': '6x6'}>,
 <matplotlib.image.AxesImage at 0x1f564daa210>,
 Text(0.5, 1.0, '6x6'))
```



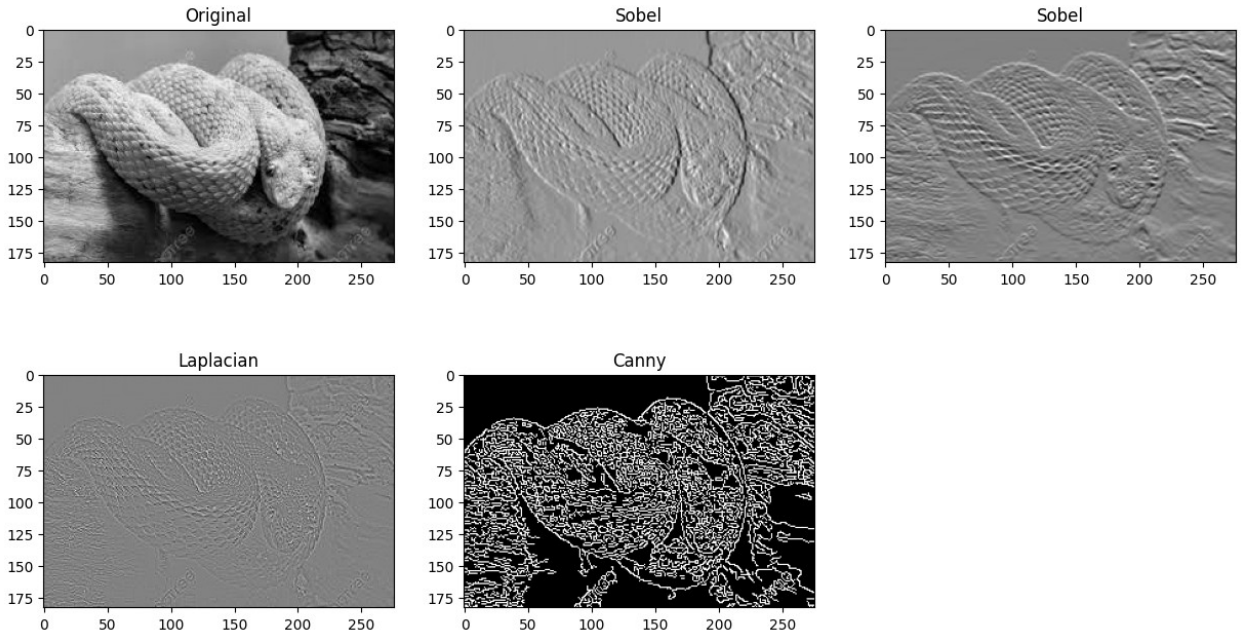

```
img=cv2.imread('flower.jpeg')
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
k3 = cv2.blur(img,(3,3))
k5 = cv2.blur(img,(5,5))
k9 = cv2.blur(img,(15,15))
plt.subplot(1,4,1),plt.imshow(img),plt.title('Original')
plt.subplot(1,4,2),plt.imshow(k3),plt.title('3x3')
plt.subplot(1,4,3),plt.imshow(k5),plt.title('5x5')
plt.subplot(1,4,4),plt.imshow(k9),plt.title('15x15')
```

```
(<Axes: title={'center': '15x15'}>,
 <matplotlib.image.AxesImage at 0x1f566283d70>,
 Text(0.5, 1.0, '15x15'))
```



Edge Detaction

```
img = cv2.imread('snacke.jpeg',0)
img1 = cv2.Sobel(img,cv2.CV_64F,1,0,5)
img2 = cv2.Sobel(img,cv2.CV_64F,0,1,5)
img3 = cv2.Laplacian(img,cv2.CV_64F)
img4 = cv2.Canny(img,50,150)
plt.subplot(2,3,1),plt.imshow(img,cmap = 'gray'),plt.title('Original')
plt.subplot(2,3,2),plt.imshow(img1,cmap = 'gray'),plt.title('Sobel')
plt.subplot(2,3,3),plt.imshow(img2,cmap = 'gray'),plt.title('Sobel')
plt.subplot(2,3,4),plt.imshow(img3,cmap =
'gray'),plt.title('Laplacian')
plt.subplot(2,3,5),plt.imshow(img4,cmap = 'gray'),plt.title('Canny')

(<Axes: title={'center': 'Canny'}>,
 <matplotlib.image.AxesImage at 0x1f562c87f80>,
 Text(0.5, 1.0, 'Canny'))
```




```
# plt.imshow(img,cmap="gray")
```

Gaussian Blur

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('flower2.jpeg')
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)

# Displaying the original image
plt.imshow(img)
# Averaging the image
img = cv2.GaussianBlur(img, (5,5), cv2.BORDER_DEFAULT)
# Displaying the blurred image
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x1f566cfcdd0>
```

Median Blur

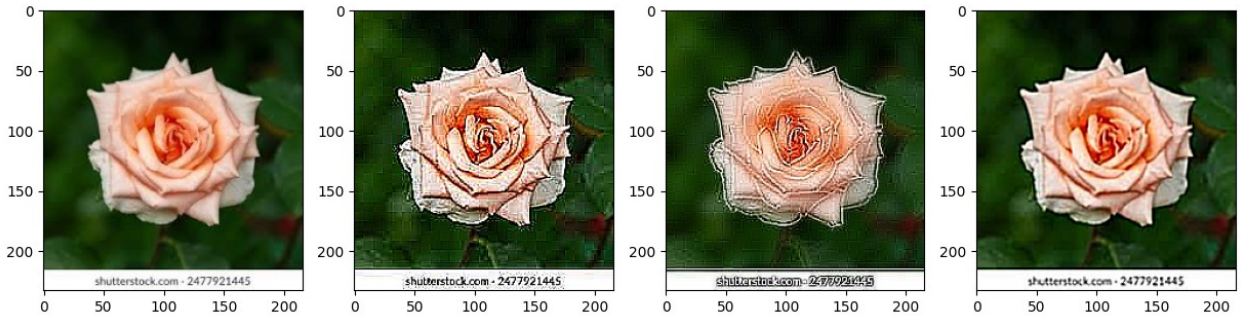
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('flower2.jpeg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.imshow(img)
img = cv2.medianBlur(img, 5)

plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x1f566cf09e0>
```

Sharpning

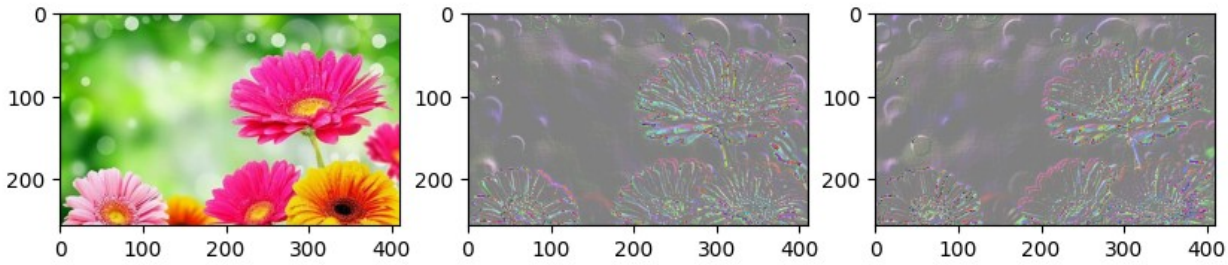
```
import cv2
import matplotlib.pyplot as plt
import numpy as np

img = cv2.cvtColor(cv2.imread('flower.jpeg'),cv2.COLOR_BGR2RGB)
k1 = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
k2 = np.array([[1,1,1],[1,-7,1],[1,1,1]])
k3 = np.array([[-1,-1,-1,-1,-1],[-1,2,2,2,-1],[-1,2,8,2,-1],[-
1,2,2,2,-1],[-1,-1,-1,-1,-1]])/8.0

img1 = cv2.filter2D(img,-1,k1)
img2 = cv2.filter2D(img,-1,k2)
img3 = cv2.filter2D(img,-1,k3)

plt.subplot(1,4,1),plt.imshow(img)
plt.subplot(1,4,2),plt.imshow(img1)
plt.subplot(1,4,3),plt.imshow(img2)
plt.subplot(1,4,4),plt.imshow(img3)
```

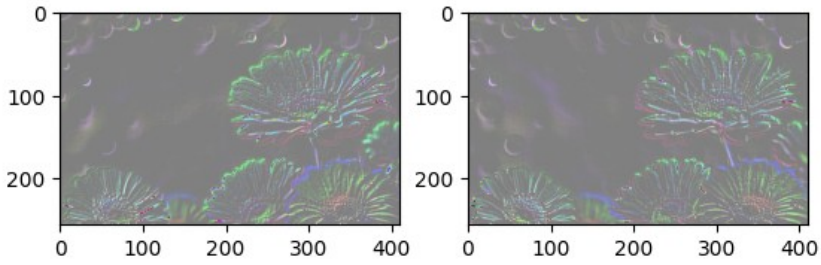
```
(<Axes: >, <matplotlib.image.AxesImage at 0x1f56a857a10>)
```



Embossing

```
import cv2
import matplotlib.pyplot as plt
import numpy as np

from matplotlib import rcParams
#img = cv2.imread('img1.jpeg',0)
img = cv2.cvtColor(cv2.imread('flower2.jpeg'),cv2.COLOR_BGR2RGB)
rcParams['figure.figsize']=10,10

k1 = np.array([[0,-1,-1],[1,0,-1],[1,1,0]])
k2 = np.array([[-1,-1,0],[-1,0,1],[0,1,1]])
k3 = np.array([[1,0,0],[0,0,0],[0,0,-1]])
k4 = np.array([[0,0,1],[0,0,0],[-1,0,0]])

img1 = cv2.filter2D(img,-1,k1) + 128
img2 = cv2.filter2D(img,-1,k2) + 128
img3 = cv2.filter2D(img,-1,k3) + 128
img4 = cv2.filter2D(img,-1,k4) + 128

plt.subplot(2,3,1),plt.imshow(img)
plt.subplot(2,3,2),plt.imshow(img1)
plt.subplot(2,3,3),plt.imshow(img2)
plt.subplot(2,3,4),plt.imshow(img3)
plt.subplot(2,3,5),plt.imshow(img4)
#Saving image
cv2.imwrite('emoboss3.jpg',img3)

True
```

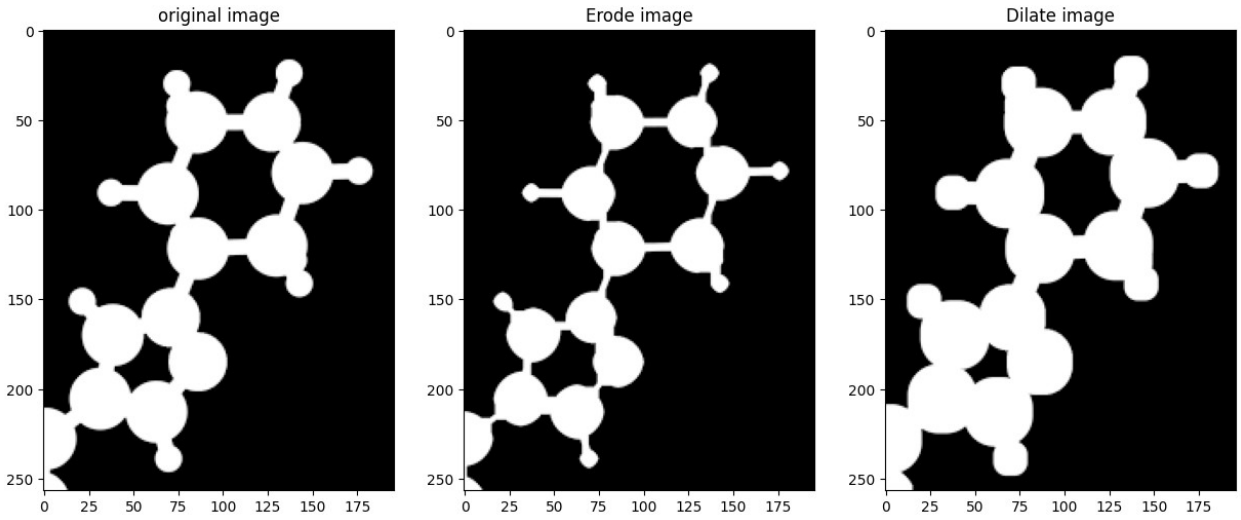




Erode & Dilate

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
img=cv2.imread("images.png")
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
k1=np.ones((5,5))
#erode (image, kernal, iterations-Layers)
img_erode=cv2.erode(img,k1, iterations=1)
img_dilate=cv2.dilate(img,k1, iterations=1)
plt.figure(figsize=(15,6))
plt.subplot(1,3,1), plt.imshow(img), plt.title("original image")
plt.subplot(1,3,2), plt.imshow(img_erode), plt.title("Erode image")
plt.subplot(1,3,3), plt.imshow(img_dilate), plt.title("Dilate image")
```

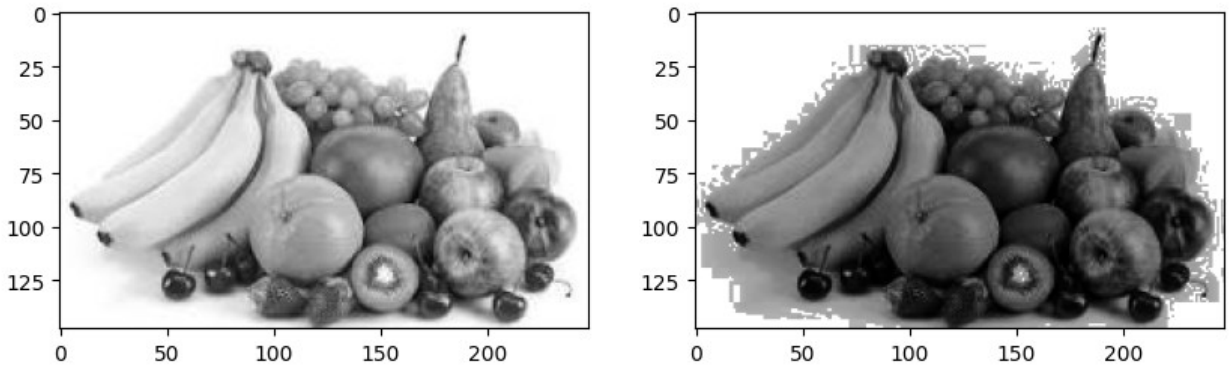
```
(<Axes: title={'center': 'Dilate image'}>,
 <matplotlib.image.AxesImage at 0x1f56a52fef0>,
 Text(0.5, 1.0, 'Dilate image'))
```



Contrast

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rcParams
img = cv2.imread('fruits.jpeg',0)
img1 = cv2.equalizeHist(img)
plt.subplot(1,2,1),plt.imshow(img,cmap='gray')
plt.subplot(1,2,2),plt.imshow(img1,cmap='gray')
```

```
(<Axes: >, <matplotlib.image.AxesImage at 0x1f56b885730>)
```

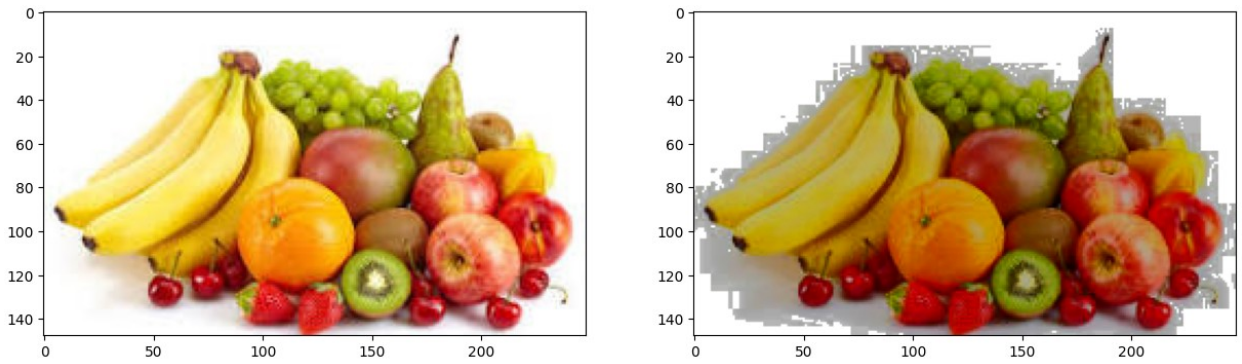


Contras Effect On Color Image

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
```

```
from matplotlib import rcParams
img=cv2.cvtColor(cv2.imread("fruits.jpeg"),cv2.COLOR_BGR2RGB)
img_YUV=cv2.cvtColor(img,cv2.COLOR_BGR2YUV)
img_YUV[:,:,0]=cv2.equalizeHist(img_YUV[:,:,0])
img1=cv2.cvtColor(img_YUV,cv2.COLOR_YUV2BGR)

rcParams['figure.figsize']=15,15
plt.subplot(1,2,1),plt.imshow(img)
plt.subplot(1,2,2),plt.imshow(img1)
```

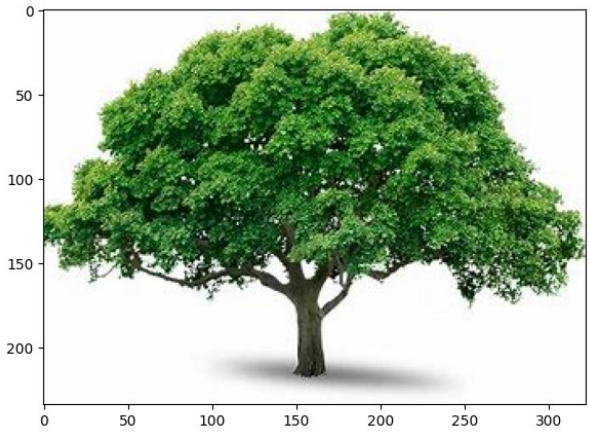
```
(<Axes: >, <matplotlib.image.AxesImage at 0x1f56b9169f0>)
```

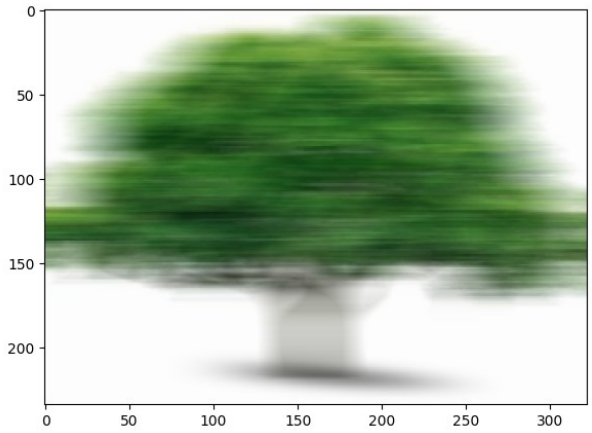


Motion Blur(Horizontal)

```
img = cv2.cvtColor(cv2.imread('trees.jpg'),cv2.COLOR_BGR2RGB)
size=50
kernel_motion_blur = np.zeros((size, size))
kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
kernel_motion_blur = kernel_motion_blur / size
img1 = cv2.filter2D(img,-1,kernel_motion_blur)
plt.subplot(1,2,1),plt.imshow(img)
plt.subplot(1,2,2),plt.imshow(img1)
```

```
(<Axes: >, <matplotlib.image.AxesImage at 0x1f56b8b5b80>)
```

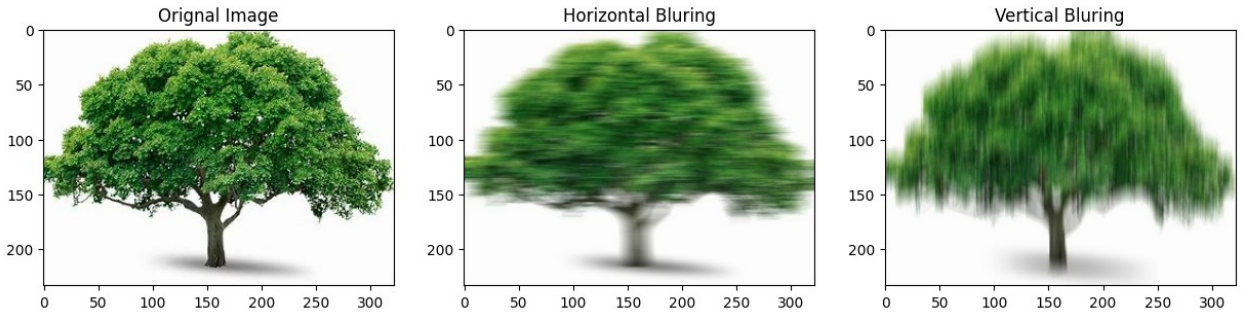




Motion Blur

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rcParams
img=cv2.cvtColor(cv2.imread('trees.jpg'),cv2.COLOR_BGR2RGB)
size=20

#Horizontal
k1=np.zeros((size,size))
k1[int(size/2),:]=np.ones(size)
k1=k1/size
img1=cv2.filter2D(img,-1,k1)

#vertical
k2=np.zeros((size,size))
k2[:,int(size/2)]=np.ones(size)
k2=k2/size
img2=cv2.filter2D(img,-1,k2)

#Increase Image size
rcParams['figure.figsize'] = 15,15
plt.subplot(1,3,1),plt.imshow(img),plt.title("Orignal Image")
plt.subplot(1,3,2),plt.imshow(img1),plt.title("Horizontal Bluring")
plt.subplot(1,3,3),plt.imshow(img2),plt.title("Vertical Bluring")
```

```
(<Axes: title={'center': 'Vertical Bluring'}>,
 <matplotlib.image.AxesImage at 0x1f56bc9ff80>,
 Text(0.5, 1.0, 'Vertical Bluring'))
```



MenuDriven Program

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

print("Image Processing Menu")
print("1. Blur")
print("2. Sharpening")
print("3. Embossing")
print("4. Edge Detection")
print("5. Erode")
print("6. Dilate")
print("7. Contrast on Color")
print("8. Contrast on Grayscale")
print("9. Motion Blur")
print("0. Exit")

choice = int(input("Enter your choice: "))

if choice == 0:
    print("Exiting the program.")
else:
    image_path = input("Enter the path to the image file: ")
    image = cv2.imread(image_path)

    if image is None:
        print("Error: Unable to open image file.")
    else:
        if choice == 1:
            result = cv2.GaussianBlur(image, (15, 15), 0)
        elif choice == 2:
            kernel = np.array([[0, -1, 0], [-1, 5,-1], [0, -1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 3:
            kernel = np.array([[0, -1, -1], [1, 0, -1], [1, 1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 4:
```

```
            result = cv2.Canny(image, 100, 200)
        elif choice == 5:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.erode(image, kernel, iterations=1)
        elif choice == 6:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.dilate(image, kernel, iterations=1)
        elif choice == 7:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            result = cv2.convertScaleAbs(image, alpha=alpha, beta=0)
        elif choice == 8:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            result = cv2.convertScaleAbs(gray, alpha=alpha, beta=0)
        elif choice == 9:
            size = 15
            kernel_motion_blur = np.zeros((size, size))
            kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
            kernel_motion_blur = kernel_motion_blur / size
            result = cv2.filter2D(image, -1, kernel_motion_blur)
        else:
            print("Invalid choice.")
            result = None

        if result is not None:
            # Convert BGR image to RGB for displaying with matplotlib
            if len(result.shape) == 3:  # Color image
                result_rgb = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
            else:  # Grayscale image
                result_rgb = result

            plt.imshow(result_rgb, cmap='gray')
            plt.title('Processed Image')
            plt.axis('off')
            plt.show()

Image Processing Menu
1. Blur
2. Sharpening
3. Embossing
4. Edge Detection
5. Erode
6. Dilate
7. Contrast on Color
8. Contrast on Grayscale
9. Motion Blur
0. Exit
```

```
Enter your choice:  1
Enter the path to the image file:  fruits.jpeg
```


```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

print("Image Processing Menu")
print("1. Blur")
print("2. Sharpening")
print("3. Embossing")
print("4. Edge Detection")
print("5. Erode")
print("6. Dilate")
print("7. Contrast on Color")
print("8. Contrast on Grayscale")
print("9. Motion Blur")
print("0. Exit")

choice = int(input("Enter your choice: "))

if choice == 0:
    print("Exiting the program.")
else:
    image_path = input("Enter the path to the image file: ")
    image = cv2.imread(image_path)

    if image is None:
        print("Error: Unable to open image file.")
```

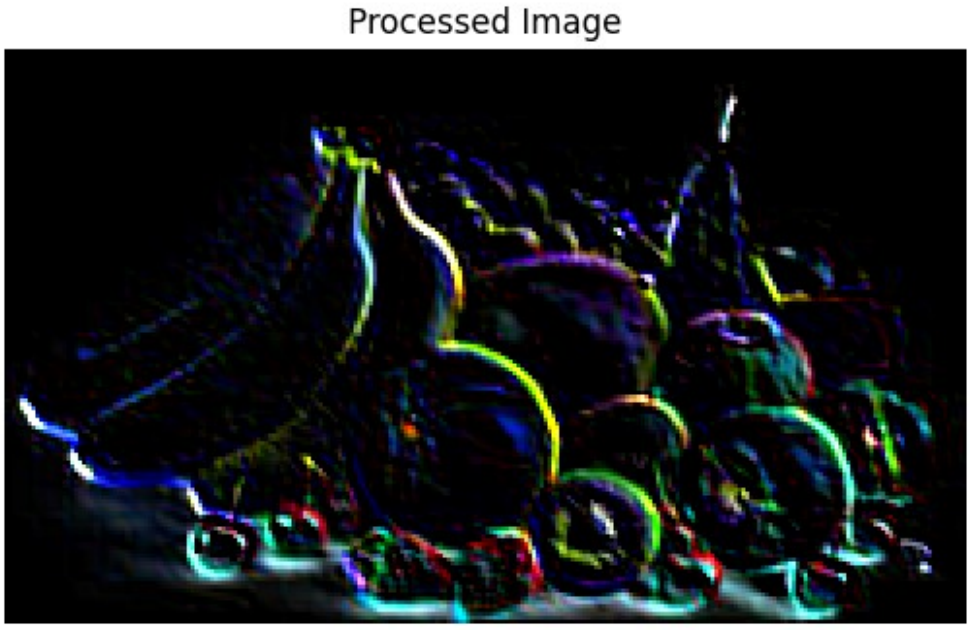
```
    else:
        if choice == 1:
            result = cv2.GaussianBlur(image, (15, 15), 0)
        elif choice == 2:
            kernel = np.array([[0, -1, 0], [-1, 5,-1], [0, -1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 3:
            kernel = np.array([[0, -1, -1], [1, 0, -1], [1, 1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 4:
            result = cv2.Canny(image, 100, 200)
        elif choice == 5:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.erode(image, kernel, iterations=1)
        elif choice == 6:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.dilate(image, kernel, iterations=1)
        elif choice == 7:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            result = cv2.convertScaleAbs(image, alpha=alpha, beta=0)
        elif choice == 8:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            result = cv2.convertScaleAbs(gray, alpha=alpha, beta=0)
        elif choice == 9:
            size = 15
            kernel_motion_blur = np.zeros((size, size))
            kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
            kernel_motion_blur = kernel_motion_blur / size
            result = cv2.filter2D(image, -1, kernel_motion_blur)
        else:
            print("Invalid choice.")
            result = None

        if result is not None:
            # Convert BGR image to RGB for displaying with matplotlib
            if len(result.shape) == 3:  # Color image
                result_rgb = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
            else:  # Grayscale image
                result_rgb = result

            plt.imshow(result_rgb, cmap='gray')
            plt.title('Processed Image')
            plt.axis('off')
            plt.show()

Image Processing Menu
1. Blur
```

```
2. Sharpening
3. Embossing
4. Edge Detection
5. Erode
6. Dilate
7. Contrast on Color
8. Contrast on Grayscale
9. Motion Blur
0. Exit

Enter your choice:  2
Enter the path to the image file:  fruits.jpeg
```


```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

print("Image Processing Menu")
print("1. Blur")
print("2. Sharpening")
print("3. Embossing")
print("4. Edge Detection")
print("5. Erode")
print("6. Dilate")
print("7. Contrast on Color")
print("8. Contrast on Grayscale")
print("9. Motion Blur")
print("0. Exit")
```

```
choice = int(input("Enter your choice: "))

if choice == 0:
    print("Exiting the program.")
else:
    image_path = input("Enter the path to the image file: ")
    image = cv2.imread(image_path)

    if image is None:
        print("Error: Unable to open image file.")
    else:
        if choice == 1:
            result = cv2.GaussianBlur(image, (15, 15), 0)
        elif choice == 2:
            kernel = np.array([[0, -1, 0], [-1, 5,-1], [0, -1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 3:
            kernel = np.array([[0, -1, -1], [1, 0, -1], [1, 1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 4:
            result = cv2.Canny(image, 100, 200)
        elif choice == 5:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.erode(image, kernel, iterations=1)
        elif choice == 6:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.dilate(image, kernel, iterations=1)
        elif choice == 7:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            result = cv2.convertScaleAbs(image, alpha=alpha, beta=0)
        elif choice == 8:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            result = cv2.convertScaleAbs(gray, alpha=alpha, beta=0)
        elif choice == 9:
            size = 15
            kernel_motion_blur = np.zeros((size, size))
            kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
            kernel_motion_blur = kernel_motion_blur / size
            result = cv2.filter2D(image, -1, kernel_motion_blur)
        else:
            print("Invalid choice.")
            result = None

        if result is not None:
            # Convert BGR image to RGB for displaying with matplotlib
            if len(result.shape) == 3:  # Color image
                result_rgb = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
```

```
            else:  # Grayscale image
                result_rgb = result

            plt.imshow(result_rgb, cmap='gray')
            plt.title('Processed Image')
            plt.axis('off')
            plt.show()

Image Processing Menu
1. Blur
2. Sharpening
3. Embossing
4. Edge Detection
5. Erode
6. Dilate
7. Contrast on Color
8. Contrast on Grayscale
9. Motion Blur
0. Exit

Enter your choice:  3
Enter the path to the image file:  fruits.jpeg
```

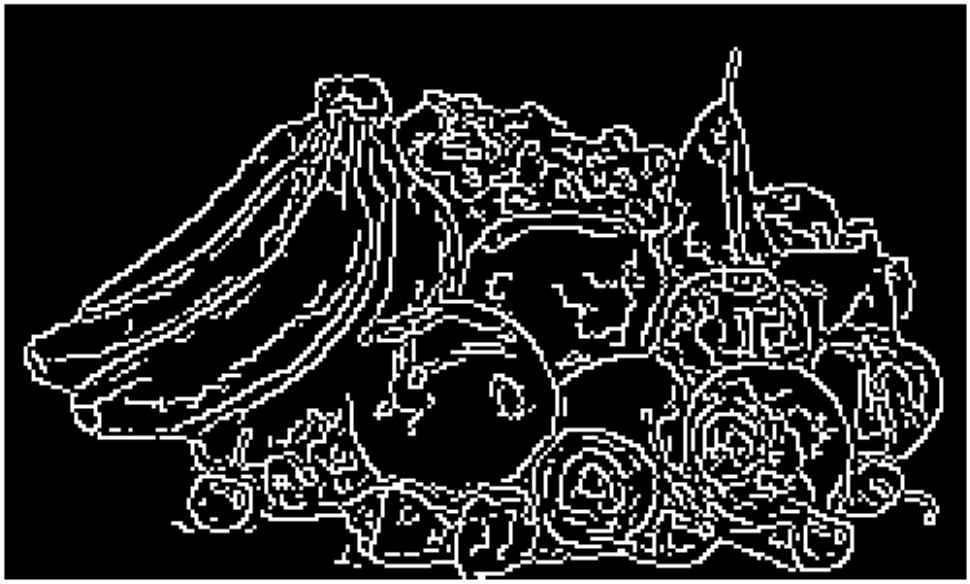



```
import cv2
import numpy as np
import matplotlib.pyplot as plt

print("Image Processing Menu")
print("1. Blur")
```

```
print("2. Sharpening")
print("3. Embossing")
print("4. Edge Detection")
print("5. Erode")
print("6. Dilate")
print("7. Contrast on Color")
print("8. Contrast on Grayscale")
print("9. Motion Blur")
print("0. Exit")

choice = int(input("Enter your choice: "))

if choice == 0:
    print("Exiting the program.")
else:
    image_path = input("Enter the path to the image file: ")
    image = cv2.imread(image_path)

    if image is None:
        print("Error: Unable to open image file.")
    else:
        if choice == 1:
            result = cv2.GaussianBlur(image, (15, 15), 0)
        elif choice == 2:
            kernel = np.array([[0, -1, 0], [-1, 5,-1], [0, -1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 3:
            kernel = np.array([[0, -1, -1], [1, 0, -1], [1, 1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 4:
            result = cv2.Canny(image, 100, 200)
        elif choice == 5:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.erode(image, kernel, iterations=1)
        elif choice == 6:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.dilate(image, kernel, iterations=1)
        elif choice == 7:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            result = cv2.convertScaleAbs(image, alpha=alpha, beta=0)
        elif choice == 8:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            result = cv2.convertScaleAbs(gray, alpha=alpha, beta=0)
        elif choice == 9:
            size = 15
            kernel_motion_blur = np.zeros((size, size))
            kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
```

```
            kernel_motion_blur = kernel_motion_blur / size
            result = cv2.filter2D(image, -1, kernel_motion_blur)
        else:
            print("Invalid choice.")
            result = None

        if result is not None:
            # Convert BGR image to RGB for displaying with matplotlib
            if len(result.shape) == 3:  # Color image
                result_rgb = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
            else:  # Grayscale image
                result_rgb = result

            plt.imshow(result_rgb, cmap='gray')
            plt.title('Processed Image')
            plt.axis('off')
            plt.show()


Image Processing Menu
1. Blur
2. Sharpening
3. Embossing
4. Edge Detection
5. Erode
6. Dilate
7. Contrast on Color
8. Contrast on Grayscale
9. Motion Blur
0. Exit

Enter your choice:  4
Enter the path to the image file:  fruits.jpeg
```

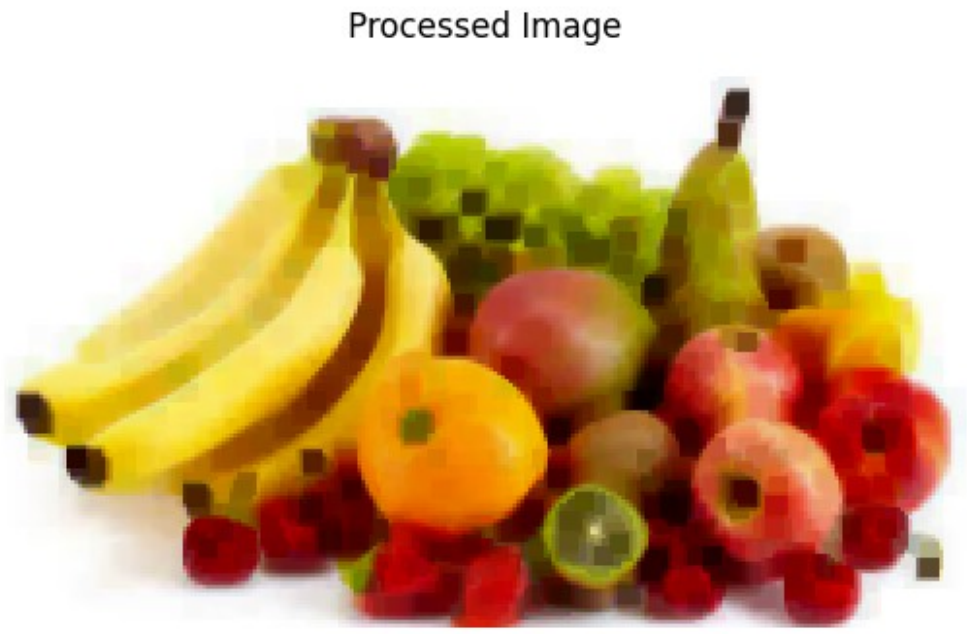
Processed Image



```
import cv2
import numpy as np
import matplotlib.pyplot as plt

print("Image Processing Menu")
print("1. Blur")
print("2. Sharpening")
print("3. Embossing")
print("4. Edge Detection")
print("5. Erode")
print("6. Dilate")
print("7. Contrast on Color")
print("8. Contrast on Grayscale")
print("9. Motion Blur")
print("0. Exit")

choice = int(input("Enter your choice: "))

if choice == 0:
    print("Exiting the program.")
else:
    image_path = input("Enter the path to the image file: ")
    image = cv2.imread(image_path)

    if image is None:
        print("Error: Unable to open image file.")
    else:
        if choice == 1:
            result = cv2.GaussianBlur(image, (15, 15), 0)
        elif choice == 2:
```

```
            kernel = np.array([[0, -1, 0], [-1, 5,-1], [0, -1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 3:
            kernel = np.array([[0, -1, -1], [1, 0, -1], [1, 1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 4:
            result = cv2.Canny(image, 100, 200)
        elif choice == 5:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.erode(image, kernel, iterations=1)
        elif choice == 6:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.dilate(image, kernel, iterations=1)
        elif choice == 7:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            result = cv2.convertScaleAbs(image, alpha=alpha, beta=0)
        elif choice == 8:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            result = cv2.convertScaleAbs(gray, alpha=alpha, beta=0)
        elif choice == 9:
            size = 15
            kernel_motion_blur = np.zeros((size, size))
            kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
            kernel_motion_blur = kernel_motion_blur / size
            result = cv2.filter2D(image, -1, kernel_motion_blur)
        else:
            print("Invalid choice.")
            result = None

        if result is not None:
            # Convert BGR image to RGB for displaying with matplotlib
            if len(result.shape) == 3:  # Color image
                result_rgb = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
            else:  # Grayscale image
                result_rgb = result

            plt.imshow(result_rgb, cmap='gray')
            plt.title('Processed Image')
            plt.axis('off')
            plt.show()

Image Processing Menu
1. Blur
2. Sharpening
3. Embossing
4. Edge Detection
5. Erode
```

```
6. Dilate
7. Contrast on Color
8. Contrast on Grayscale
9. Motion Blur
0. Exit

Enter your choice:  5
Enter the path to the image file:  fruits.jpeg
```




```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

print("Image Processing Menu")
print("1. Blur")
print("2. Sharpening")
print("3. Embossing")
print("4. Edge Detection")
print("5. Erode")
print("6. Dilate")
print("7. Contrast on Color")
print("8. Contrast on Grayscale")
print("9. Motion Blur")
print("0. Exit")

choice = int(input("Enter your choice: "))

if choice == 0:
    print("Exiting the program.")
```

```
else:
    image_path = input("Enter the path to the image file: ")
    image = cv2.imread(image_path)

    if image is None:
        print("Error: Unable to open image file.")
    else:
        if choice == 1:
            result = cv2.GaussianBlur(image, (15, 15), 0)
        elif choice == 2:
            kernel = np.array([[0, -1, 0], [-1, 5,-1], [0, -1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 3:
            kernel = np.array([[0, -1, -1], [1, 0, -1], [1, 1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 4:
            result = cv2.Canny(image, 100, 200)
        elif choice == 5:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.erode(image, kernel, iterations=1)
        elif choice == 6:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.dilate(image, kernel, iterations=1)
        elif choice == 7:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            result = cv2.convertScaleAbs(image, alpha=alpha, beta=0)
        elif choice == 8:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            result = cv2.convertScaleAbs(gray, alpha=alpha, beta=0)
        elif choice == 9:
            size = 15
            kernel_motion_blur = np.zeros((size, size))
            kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
            kernel_motion_blur = kernel_motion_blur / size
            result = cv2.filter2D(image, -1, kernel_motion_blur)
        else:
            print("Invalid choice.")
            result = None

        if result is not None:
            # Convert BGR image to RGB for displaying with matplotlib
            if len(result.shape) == 3:  # Color image
                result_rgb = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
            else:  # Grayscale image
                result_rgb = result

            plt.imshow(result_rgb, cmap='gray')
```

```
            plt.title('Processed Image')
            plt.axis('off')
            plt.show()

Image Processing Menu
1. Blur
2. Sharpening
3. Embossing
4. Edge Detection
5. Erode
6. Dilate
7. Contrast on Color
8. Contrast on Grayscale
9. Motion Blur
0. Exit

Enter your choice:  6
Enter the path to the image file:  fruits.jpeg
```

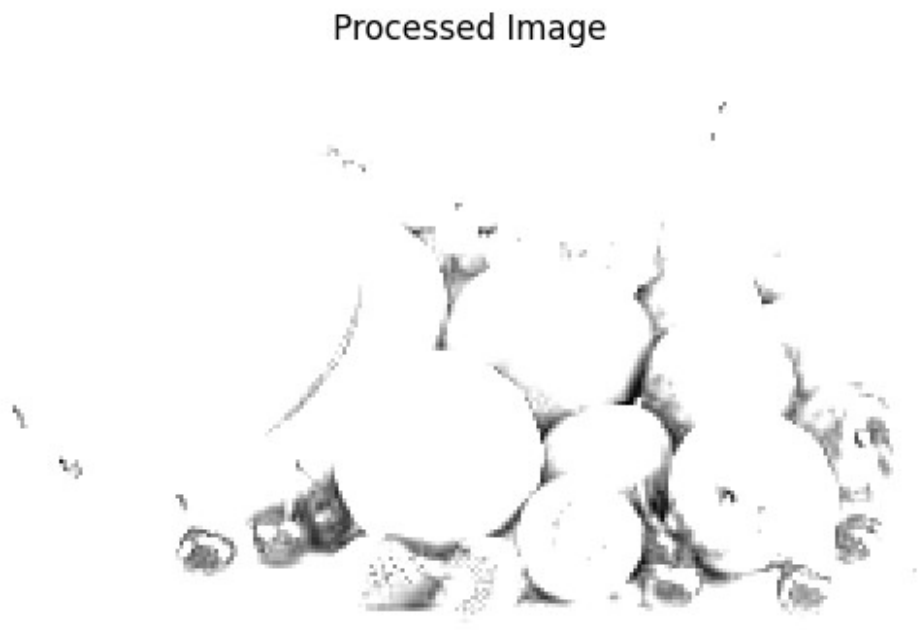

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

print("Image Processing Menu")
print("1. Blur")
print("2. Sharpening")
print("3. Embossing")
print("4. Edge Detection")
print("5. Erode")
```

```
print("6. Dilate")
print("7. Contrast on Color")
print("8. Contrast on Grayscale")
print("9. Motion Blur")
print("0. Exit")

choice = int(input("Enter your choice: "))

if choice == 0:
    print("Exiting the program.")
else:
    image_path = input("Enter the path to the image file: ")
    image = cv2.imread(image_path)

    if image is None:
        print("Error: Unable to open image file.")
    else:
        if choice == 1:
            result = cv2.GaussianBlur(image, (15, 15), 0)
        elif choice == 2:
            kernel = np.array([[0, -1, 0], [-1, 5,-1], [0, -1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 3:
            kernel = np.array([[0, -1, -1], [1, 0, -1], [1, 1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 4:
            result = cv2.Canny(image, 100, 200)
        elif choice == 5:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.erode(image, kernel, iterations=1)
        elif choice == 6:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.dilate(image, kernel, iterations=1)
        elif choice == 7:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            result = cv2.convertScaleAbs(image, alpha=alpha, beta=0)
        elif choice == 8:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            result = cv2.convertScaleAbs(gray, alpha=alpha, beta=0)
        elif choice == 9:
            size = 15
            kernel_motion_blur = np.zeros((size, size))
            kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
            kernel_motion_blur = kernel_motion_blur / size
            result = cv2.filter2D(image, -1, kernel_motion_blur)
        else:
            print("Invalid choice.")
```

```
            result = None

        if result is not None:
            # Convert BGR image to RGB for displaying with matplotlib
            if len(result.shape) == 3:  # Color image
                result_rgb = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
            else:  # Grayscale image
                result_rgb = result

            plt.imshow(result_rgb, cmap='gray')
            plt.title('Processed Image')
            plt.axis('off')
            plt.show()

Image Processing Menu
1. Blur
2. Sharpening
3. Embossing
4. Edge Detection
5. Erode
6. Dilate
7. Contrast on Color
8. Contrast on Grayscale
9. Motion Blur
0. Exit

Enter your choice:  7
Enter the path to the image file:  fruits.jpeg
Enter the alpha value for contrast adjustment (1.0-3.0):  1.5
```

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

print("Image Processing Menu")
print("1. Blur")
print("2. Sharpening")
print("3. Embossing")
print("4. Edge Detection")
print("5. Erode")
print("6. Dilate")
print("7. Contrast on Color")
print("8. Contrast on Grayscale")
print("9. Motion Blur")
print("0. Exit")

choice = int(input("Enter your choice: "))

if choice == 0:
    print("Exiting the program.")
else:
    image_path = input("Enter the path to the image file: ")
    image = cv2.imread(image_path)

    if image is None:
        print("Error: Unable to open image file.")
    else:
        if choice == 1:
            result = cv2.GaussianBlur(image, (15, 15), 0)
        elif choice == 2:
            kernel = np.array([[0, -1, 0], [-1, 5,-1], [0, -1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 3:
            kernel = np.array([[0, -1, -1], [1, 0, -1], [1, 1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 4:
            result = cv2.Canny(image, 100, 200)
        elif choice == 5:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.erode(image, kernel, iterations=1)
        elif choice == 6:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.dilate(image, kernel, iterations=1)
        elif choice == 7:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            result = cv2.convertScaleAbs(image, alpha=alpha, beta=0)
        elif choice == 8:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
```

```
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            result = cv2.convertScaleAbs(gray, alpha=alpha, beta=0)
        elif choice == 9:
            size = 15
            kernel_motion_blur = np.zeros((size, size))
            kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
            kernel_motion_blur = kernel_motion_blur / size
            result = cv2.filter2D(image, -1, kernel_motion_blur)
        else:
            print("Invalid choice.")
            result = None

        if result is not None:
            # Convert BGR image to RGB for displaying with matplotlib
            if len(result.shape) == 3:  # Color image
                result_rgb = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
            else:  # Grayscale image
                result_rgb = result

            plt.imshow(result_rgb, cmap='gray')
            plt.title('Processed Image')
            plt.axis('off')
            plt.show()

Image Processing Menu
1. Blur
2. Sharpening
3. Embossing
4. Edge Detection
5. Erode
6. Dilate
7. Contrast on Color
8. Contrast on Grayscale
9. Motion Blur
0. Exit

Enter your choice:  8
Enter the path to the image file:  fruits.jpeg
Enter the alpha value for contrast adjustment (1.0-3.0):  2.9
```




```
import cv2
import numpy as np
import matplotlib.pyplot as plt

print("Image Processing Menu")
print("1. Blur")
print("2. Sharpening")
print("3. Embossing")
print("4. Edge Detection")
print("5. Erode")
print("6. Dilate")
print("7. Contrast on Color")
print("8. Contrast on Grayscale")
print("9. Motion Blur")
print("0. Exit")

choice = int(input("Enter your choice: "))

if choice == 0:
    print("Exiting the program.")
else:
    image_path = input("Enter the path to the image file: ")
    image = cv2.imread(image_path)

    if image is None:
        print("Error: Unable to open image file.")
    else:
        if choice == 1:
            result = cv2.GaussianBlur(image, (15, 15), 0)
        elif choice == 2:
```

```
            kernel = np.array([[0, -1, 0], [-1, 5,-1], [0, -1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 3:
            kernel = np.array([[0, -1, -1], [1, 0, -1], [1, 1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 4:
            result = cv2.Canny(image, 100, 200)
        elif choice == 5:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.erode(image, kernel, iterations=1)
        elif choice == 6:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.dilate(image, kernel, iterations=1)
        elif choice == 7:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            result = cv2.convertScaleAbs(image, alpha=alpha, beta=0)
        elif choice == 8:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            result = cv2.convertScaleAbs(gray, alpha=alpha, beta=0)
        elif choice == 9:
            size = 15
            kernel_motion_blur = np.zeros((size, size))
            kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
            kernel_motion_blur = kernel_motion_blur / size
            result = cv2.filter2D(image, -1, kernel_motion_blur)
        else:
            print("Invalid choice.")
            result = None

        if result is not None:
            # Convert BGR image to RGB for displaying with matplotlib
            if len(result.shape) == 3:  # Color image
                result_rgb = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
            else:  # Grayscale image
                result_rgb = result

            plt.imshow(result_rgb, cmap='gray')
            plt.title('Processed Image')
            plt.axis('off')
            plt.show()

Image Processing Menu
1. Blur
2. Sharpening
3. Embossing
4. Edge Detection
5. Erode
```

```
6. Dilate
7. Contrast on Color
8. Contrast on Grayscale
9. Motion Blur
0. Exit

Enter your choice:  9
Enter the path to the image file:  fruits.jpeg
```

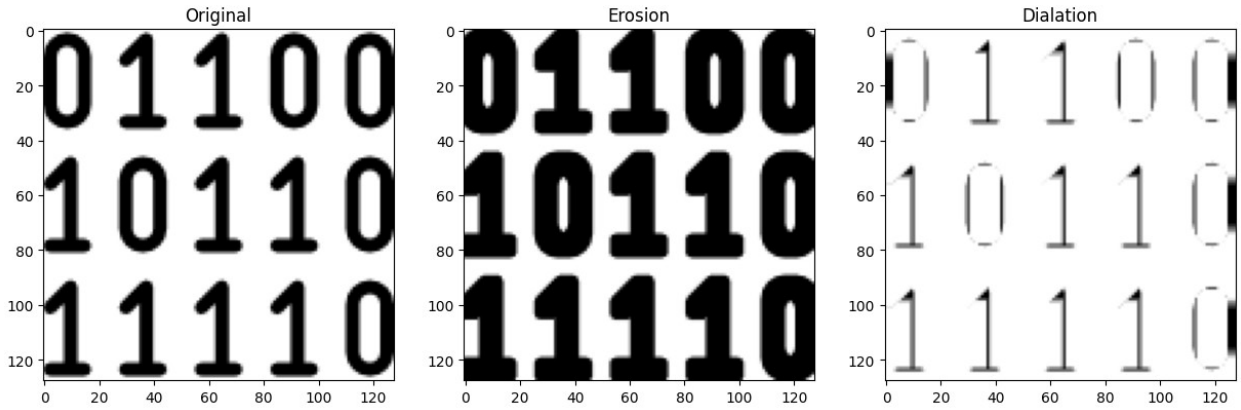

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

print("Image Processing Menu")
print("1. Blur")
print("2. Sharpening")
print("3. Embossing")
print("4. Edge Detection")
print("5. Erode")
print("6. Dilate")
print("7. Contrast on Color")
print("8. Contrast on Grayscale")
print("9. Motion Blur")
print("0. Exit")

choice = int(input("Enter your choice: "))

if choice == 0:
    print("Exiting the program.")
```

```
else:
    image_path = input("Enter the path to the image file: ")
    image = cv2.imread(image_path)

    if image is None:
        print("Error: Unable to open image file.")
    else:
        if choice == 1:
            result = cv2.GaussianBlur(image, (15, 15), 0)
        elif choice == 2:
            kernel = np.array([[0, -1, 0], [-1, 5,-1], [0, -1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 3:
            kernel = np.array([[0, -1, -1], [1, 0, -1], [1, 1, 0]])
            result = cv2.filter2D(image, -1, kernel)
        elif choice == 4:
            result = cv2.Canny(image, 100, 200)
        elif choice == 5:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.erode(image, kernel, iterations=1)
        elif choice == 6:
            kernel = np.ones((5, 5), np.uint8)
            result = cv2.dilate(image, kernel, iterations=1)
        elif choice == 7:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            result = cv2.convertScaleAbs(image, alpha=alpha, beta=0)
        elif choice == 8:
            alpha = float(input("Enter the alpha value for contrast
adjustment (1.0-3.0): "))
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            result = cv2.convertScaleAbs(gray, alpha=alpha, beta=0)
        elif choice == 9:
            size = 15
            kernel_motion_blur = np.zeros((size, size))
            kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
            kernel_motion_blur = kernel_motion_blur / size
            result = cv2.filter2D(image, -1, kernel_motion_blur)
        else:
            print("Invalid choice.")
            result = None

        if result is not None:
            # Convert BGR image to RGB for displaying with matplotlib
            if len(result.shape) == 3:  # Color image
                result_rgb = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
            else:  # Grayscale image
                result_rgb = result

            plt.imshow(result_rgb, cmap='gray')
```

```
            plt.title('Processed Image')
            plt.axis('off')
            plt.show()

Image Processing Menu
1. Blur
2. Sharpening
3. Embossing
4. Edge Detection
5. Erode
6. Dilate
7. Contrast on Color
8. Contrast on Grayscale
9. Motion Blur
0. Exit

Enter your choice:  0

Exiting the program.
```

Morphological Image Processing Operations

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rcParams
img = cv2.imread('binaryimages.png')
#img = cv2.imread('sample.jpg',0)
k1 = np.ones((5,5),np.uint8)
img1 = cv2.erode(img,k1,iterations=1)
img2 = cv2.dilate(img,k1,iterations=1)
rcParams['figure.figsize']=15,15
plt.subplot(1,3,1),plt.imshow(img),plt.title("Original")
plt.subplot(1,3,2),plt.imshow(img1),plt.title("Erosion")
plt.subplot(1,3,3),plt.imshow(img2),plt.title("Dialation")

(<Axes: title={'center': 'Dialation'}>,
 <matplotlib.image.AxesImage at 0x237dd38b3b0>,
 Text(0.5, 1.0, 'Dialation'))
```



vignette Filter

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
img=cv2.cvtColor(cv2.imread("flower3.jpeg"),cv2.COLOR_BGR2RGB)
rows,cols=img.shape[:2]
kx=cv2.getGaussianKernel(cols,200)
ky=cv2.getGaussianKernel(rows,200)
k=ky*kx.T
mask=255*k/np.linalg.norm(k)
img1=np.copy(img)
for i in range(3):
    img1[:,:,i]=img1[:,:,i]*mask
plt.subplot(1,2,1),plt.imshow(img)
plt.subplot(1,2,2),plt.imshow(img1)
```

```
(<Axes: >, <matplotlib.image.AxesImage at 0x237dd4d1b80>)
```