Digital Image Processing

Title: Edge Detection

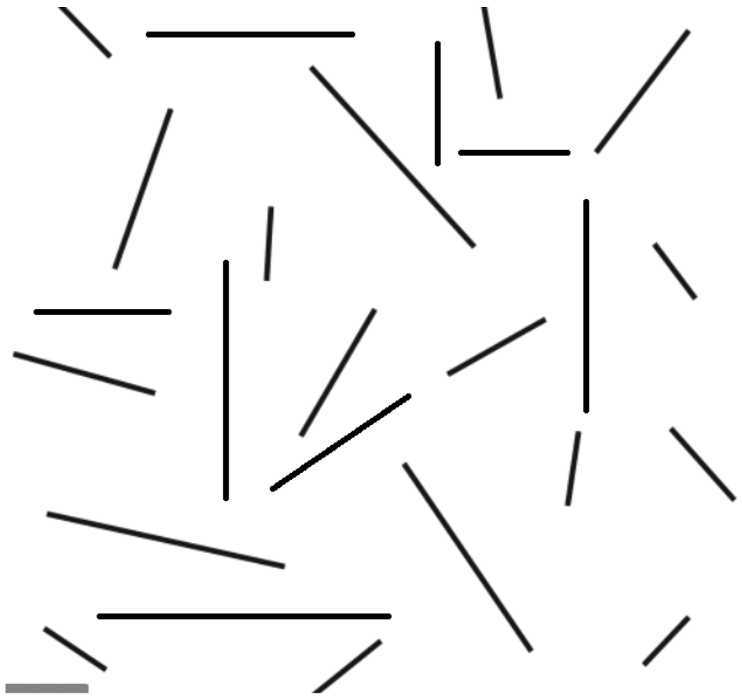
Tools Used: Python

Procedure: Open IDLE and perform the following tasks

Task 1

Read the image below, apply the masks to detect horizontal, vertical and diagonal lines and compare the results of different masks. Below are four different line detection filters.

$$\begin{array}{cccc}
\begin{matrix}-1 & -1 & -1\\ 2 & 2 & 2\\ -1 & -1 & -1\end{matrix} &
\begin{matrix}2 & -1 & -1\\ -1 & 2 & -1\\ -1 & -1 & 2\end{matrix} &
\begin{matrix}-1 & 2 & -1\\ -1 & 2 & -1\\ -1 & 2 & -1\end{matrix} &
\begin{matrix}-1 & -1 & 2\\ -1 & 2 & -1\\ 2 & -1 & -1\end{matrix} \\
\text{Horizontal} & \text{-45°} & \text{Vertical} & \text{+ 45°}
\end{array}$$

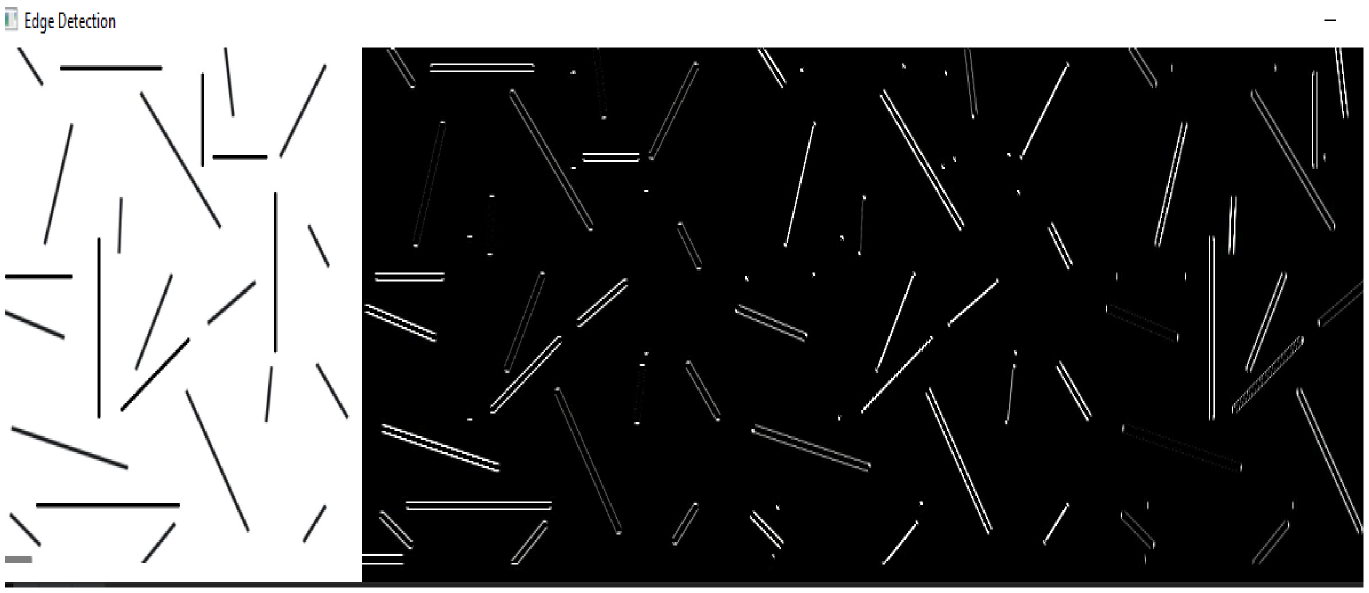


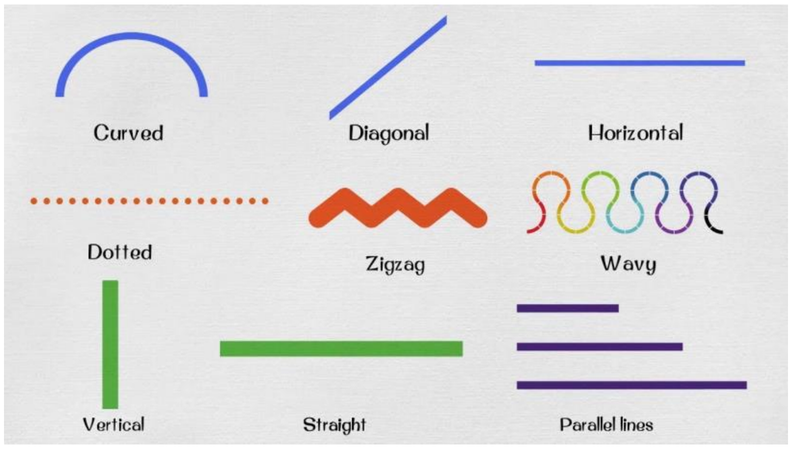
Procedure:

```
import cv2
import numpy as np

img = cv2.imread("lines.jpeg")
img = cv2.resize(img, (0, 0), None, .5, .5)

HorizKn = np.array(([[-1, -1, -1], [2, 2, 2], [-1, -1, -1]]), np.float32)
Ng45Kn = np.array(([[2, -1, -1], [-1, 2, -1], [-1, -1, 2]]), np.float32)
VertiKn = np.array(([[-1, 2, -1], [-1, 2, -1], [-1, 2, -1]]), np.float32)
Ps45Kn = np.array(([[-1, -1, 2], [-1, 2, -1], [2, -1, -1]]), np.float32)

HorizBlur = cv2.filter2D(src=img, kernel=HorizKn, ddepth=-1)
Ng45Blur = cv2.filter2D(src=img, kernel=Ng45Kn, ddepth=-1)
VertiBlur = cv2.filter2D(src=img, kernel=VertiKn, ddepth=-1)
Ps45Blur = cv2.filter2D(src=img, kernel=Ps45Kn, ddepth=-1)
```

```
horizStck = np.concatenate((img, HorizBlur, Ng45Blur, VertiBlur, Ps45Blur), axis=1)

cv2.imwrite("Output.jpg", horizStck)

cv2.imshow("Edge Detection", horizStck)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

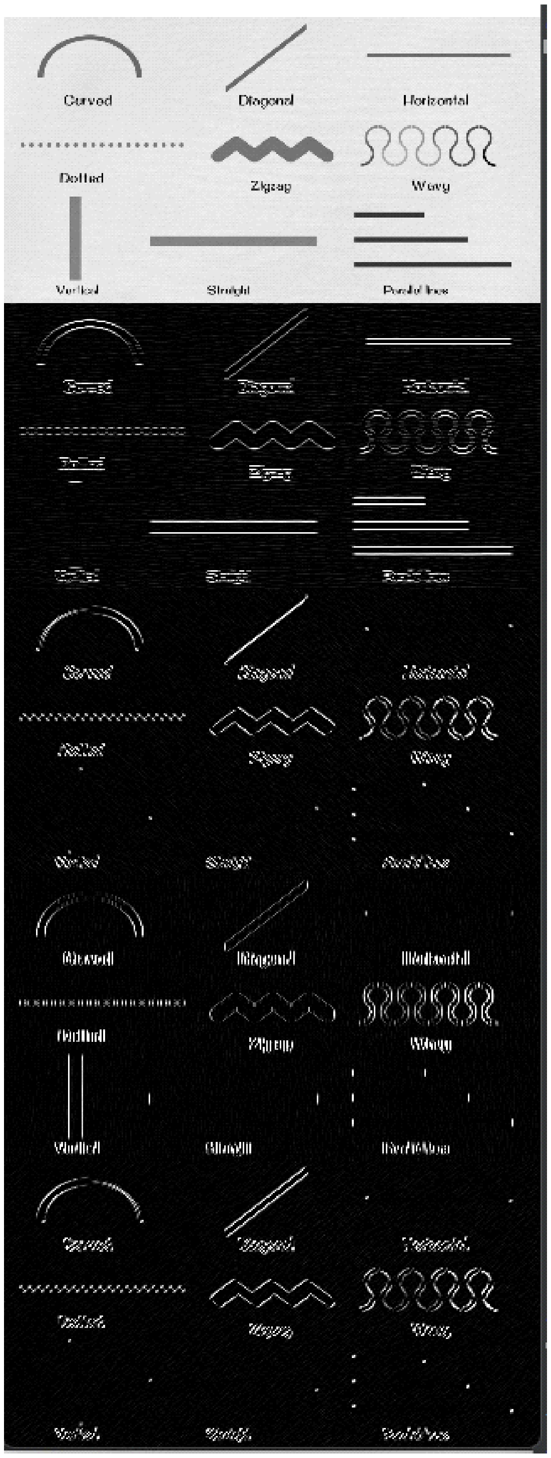




Procedure:

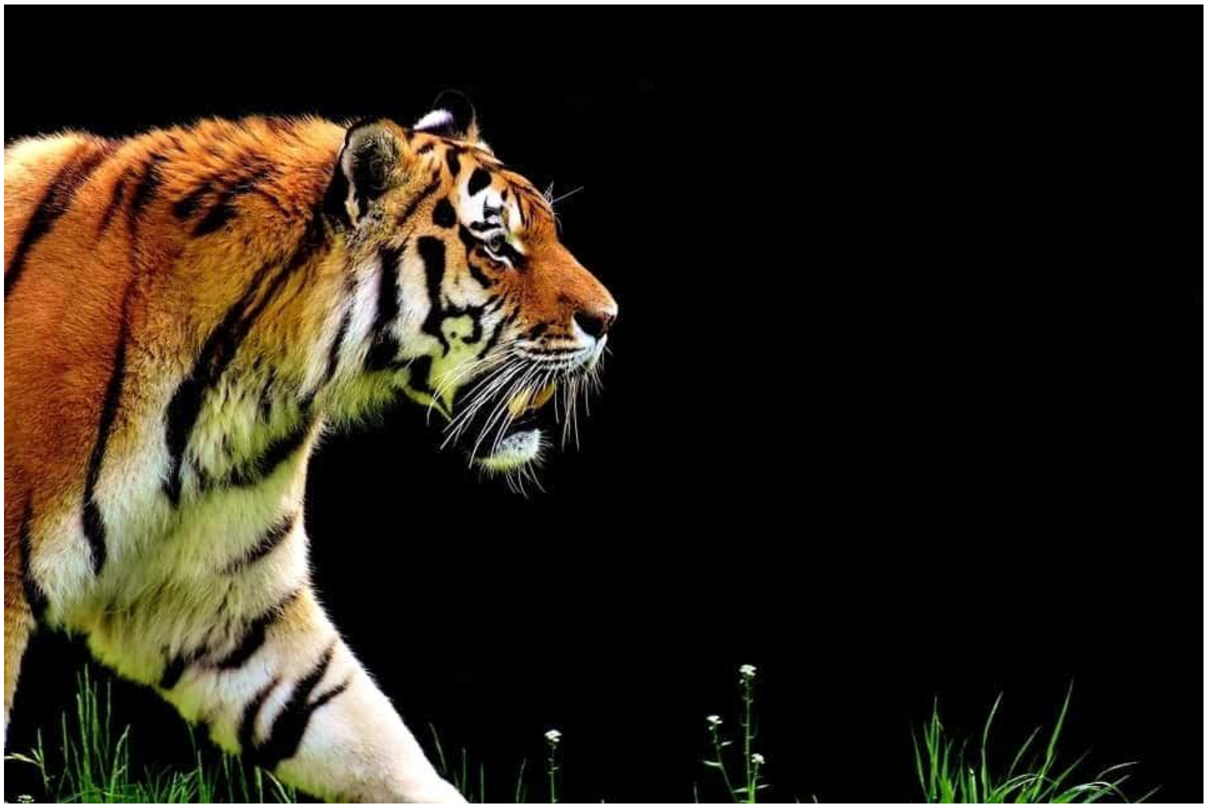
```
import cv2
import numpy as np

img = cv2.imread("curves.jpeg")
img = cv2.resize(img, (0, 0), None, .25, .25)
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

HorizKn = np.array(([[-1, -1, -1], [2, 2, 2], [-1, -1, -1]]), np.float32)
Ng45Kn = np.array(([[2, -1, -1], [-1, 2, -1], [-1, -1, 2]]), np.float32)
VertiKn = np.array(([[-1, 2, -1], [-1, 2, -1], [-1, 2, -1]]), np.float32)
Ps45Kn = np.array(([[-1, -1, 2], [-1, 2, -1], [2, -1, -1]]), np.float32)

HorizBlur = cv2.filter2D(src=img, kernel=HorizKn, ddepth=-1)
Ng45Blur = cv2.filter2D(src=img, kernel=Ng45Kn, ddepth=-1)
VertiBlur = cv2.filter2D(src=img, kernel=VertiKn, ddepth=-1)
Ps45Blur = cv2.filter2D(src=img, kernel=Ps45Kn, ddepth=-1)

horizStck = np.concatenate((img, HorizBlur, Ng45Blur, VertiBlur, Ps45Blur), axis=0)

cv2.imwrite("Output.jpg", horizStck)

cv2.imshow("Edge Detection", horizStck)
```

Output:


Curved
Diagonal
Horizontal
Dotted
Zigzag
Wavy
Vertical
Straight
Parallel lines

Task 2

Find the horizontal and vertical edges in the following picture. Display both horizontal, vertical and combined edges. Use a sobel filter. Implement both using the filter and the built-in function.

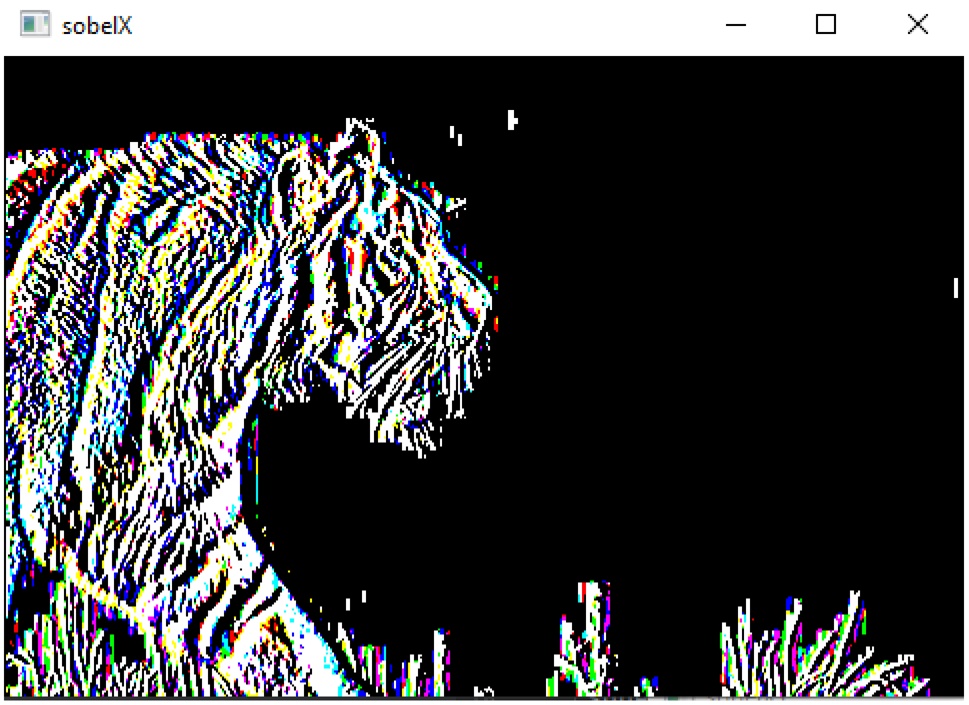


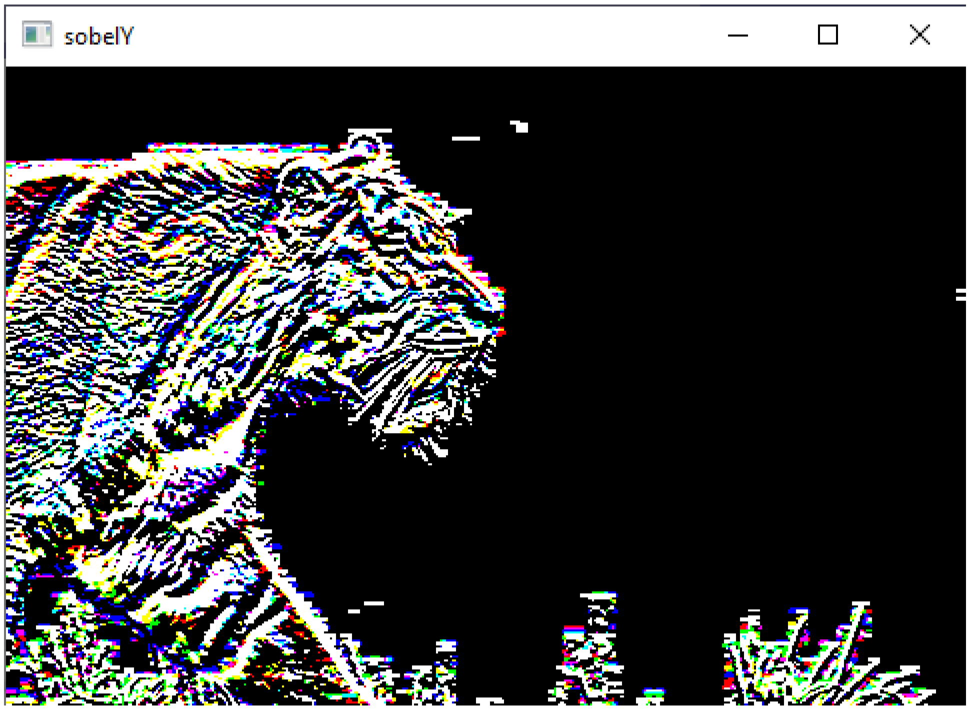
Procedure:

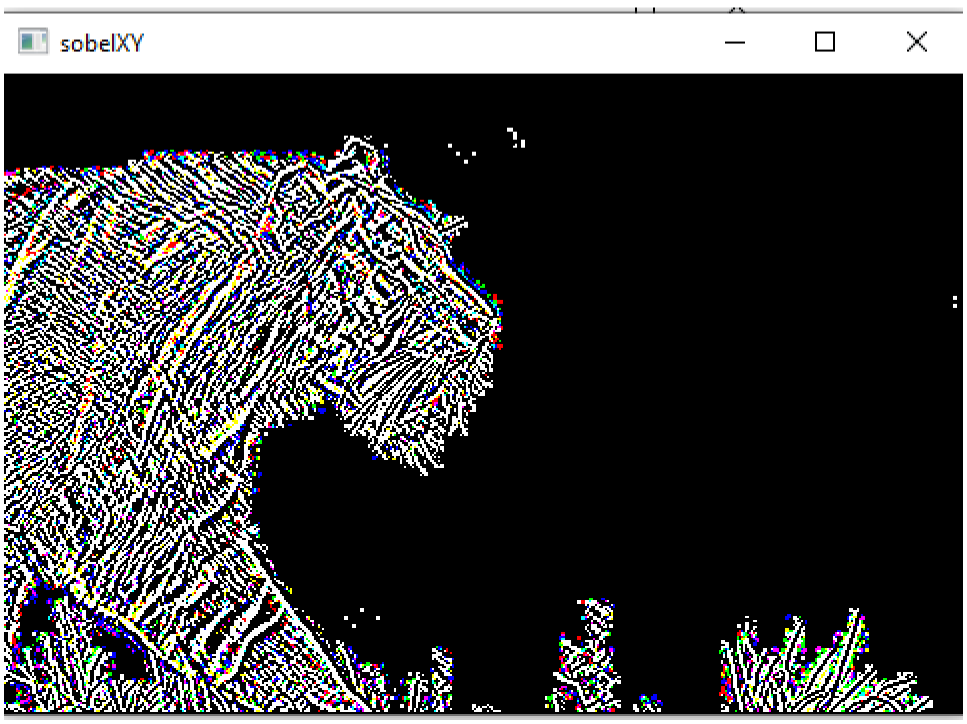
```
import cv2
import numpy as np

img = cv2.imread("tiger.jpg")
img = cv2.resize(img, (0, 0), None, .5, .5)
HorizKn= np.array(([[-1, -1, -1], [2, 2, 2], [-1, -1, -1]]), np.float32)
Ng45Kn = np.array(([[2, -1, -1], [-1, 2, -1], [-1, -1, 2]]), np.float32)
VertiKn = np.array(([[-1, 2, -1], [-1, 2, -1], [-1, 2, -1]]), np.float32)
Ps45Kn = np.array(([[-1, -1, 2], [-1, 2, -1], [2, -1, -1]]), np.float32)
SobelX = cv2.Sobel(img, cv2.CV_32F, dx=1, dy=0, ksize=3)
SobelY = cv2.Sobel(img, cv2.CV_32F, dx=0, dy=1, ksize=3)

cv2.imshow("sobelX", SobelX)
cv2.imshow("sobelY", SobelY)
SobelXY = cv2.Sobel(img, cv2.CV_32F, dx=1, dy=1, ksize=3)
cv2.imshow("sobelXY", SobelXY)
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

Output:



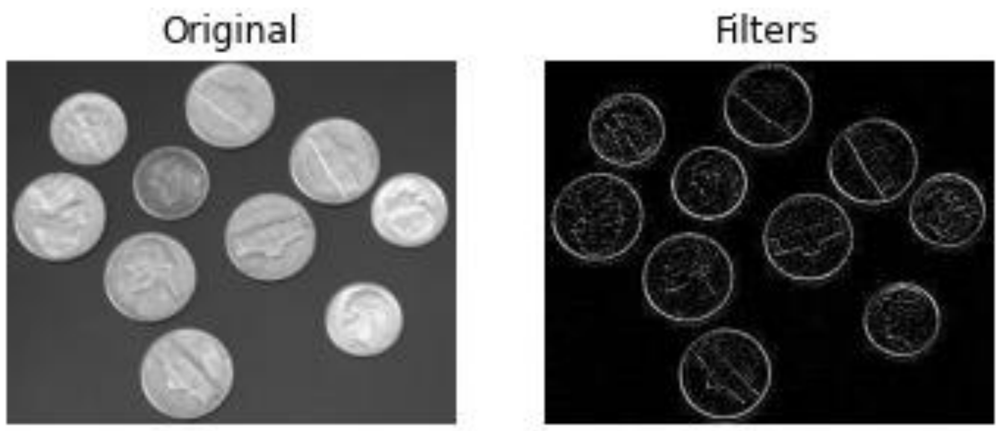




Task 3:

Read the image 'coin', you have to get the required output image shown in figure below. Analyze the input image and make the decision yourself what to do to get the required output.

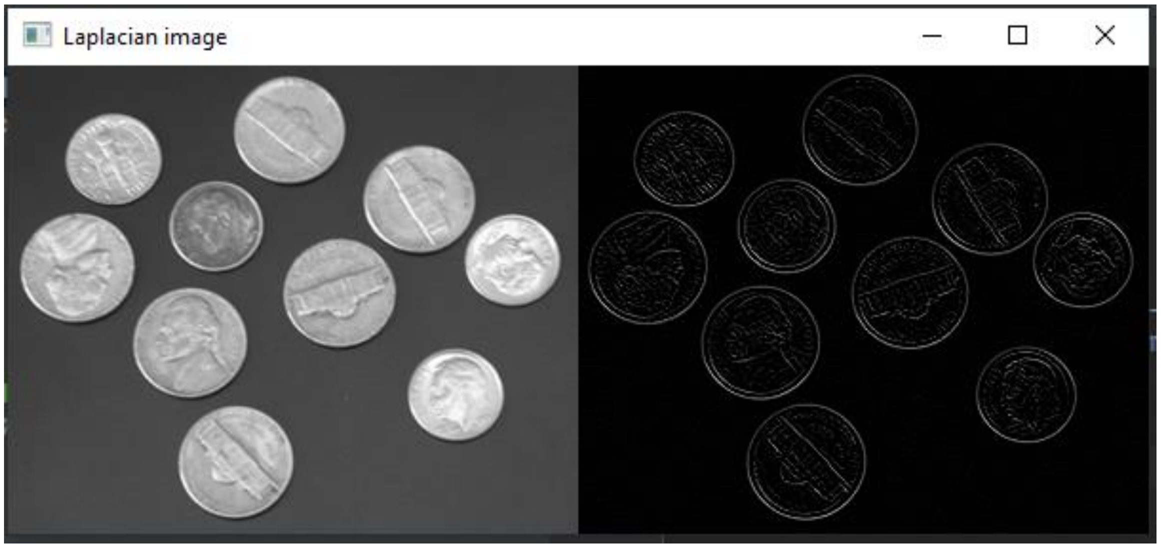
Hint: Choose kernel/mask values yourself to get required output.



Procedure:

```
import cv2
import numpy as np

img = cv2.imread('coins.png')
Lap_filter = cv2.Laplacian(img, ddepth=-1)
img1 = np.concatenate((img, Lap_filter), axis=1)
cv2.imshow("Laplacian image", img1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:


Laplacian image