```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
path = 'lena.png'
img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)

height, width = img.shape
kernel_size = 3
kernel = np.ones((kernel_size, kernel_size), np.float32) / (kernel_size * kernel_size)
st = kernel.shape[0]//2
ed = kernel.shape[1]//2

mean_img = np.zeros((height, width))
for row in range(st, height-st):
  for col in range(ed, width-ed):
    # mean filtera
    for i in range(-st, st+1):
      for j in range(-ed, ed+1):
        mean_img[row, col] += img[row+i, col+j]*kernel[i+st, j+ed]

plt.title('mean filter')
plt.imshow(mean_img, cmap='gray')
plt.show()
```

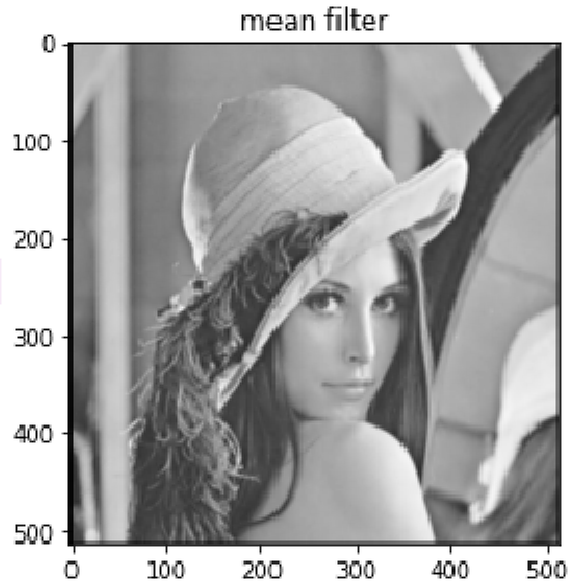


fig 2.1 : mean filter

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

path = 'lena.png'
img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
height, width = img.shape
kernel_size = 3
kernel = np.ones((kernel_size, kernel_size), np.float32) / (kernel_size * kernel_size)
st = kernel.shape[0]//2
ed = kernel.shape[1]//2
mean_img = np.zeros((height, width))
for row in range(st, height-st):
  for col in range(ed, width-ed):
    # median filter
    arr_list = list()
    arr_list.append(img[row-st:row+st+1, col-ed:col+ed+1])
    arr_list = np.array(arr_list)
    arr_list = arr_list.flatten()
    arr_list.sort()
    mean_img[row, col] = arr_list[len(arr_list)//2]

plt.title('median filter')
plt.imshow(mean_img, cmap='gray')
plt.show()
```

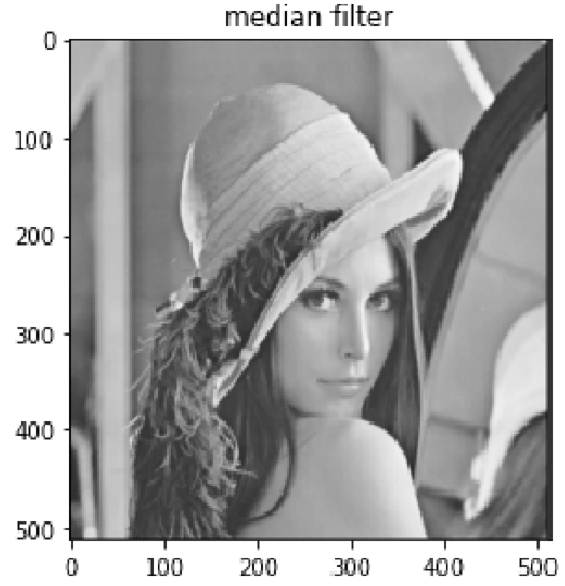


fig 2.2 : median filter

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

path = 'lena.png'
img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
height, width = img.shape
kernel_size = 7
sigma = (kernel_size - 1) // 6
kernel = np.array([[0, -1, 0], [-1, 4, -1], [0, -1, 0]])
st = kernel.shape[0]//2
ed = kernel.shape[1]//2
lapalce_img = np.zeros((height, width))
for row in range(st, height-st):
  for col in range(ed, width-ed):
    img_portion = img[row-st:row+st+1, col-ed:col+ed+1]
    img_portion = img_portion[::-1, ::-1]
    lapalce_img[row][col] = np.sum(img_portion*kernel)

from skimage.exposure import rescale_intensity
lapalce_img = rescale_intensity(lapalce_img, in_range=(0, 255))

plt.imshow(img, cmap='gray')
plt.show()
plt.imshow(lapalce_img, cmap='gray')
```

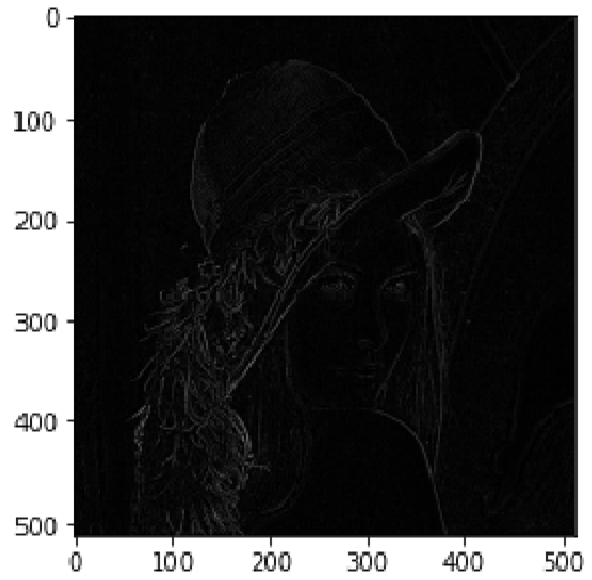
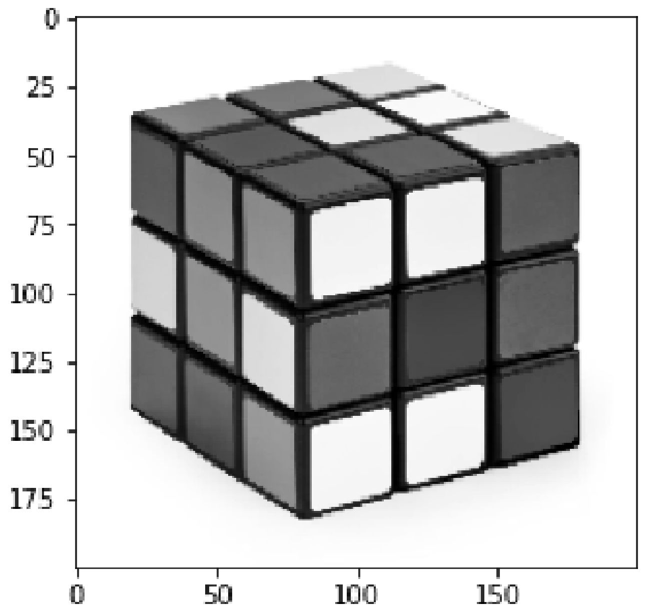


fig 3 : laplace filter

```
path = 'rubs cube.jpg'
img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
img = cv2.resize(img, (200, 200))
height, width = img.shape
kernel_size = 5
sigma = kernel_size/5
kernel = np.zeros((kernel_size, kernel_size))
st = kernel.shape[0]//2
ed = kernel.shape[1]//2

def gaussian_filter(x, y, sigma):
  return math.exp(-(x**2+y**2)/(2*sigma**2))
for i in range(-st, st+1):
  for j in range(-ed, ed+1):
    kernel[i+st][j+ed] = gaussian_filter(i, j, sigma)
kernel = kernel/kernel.sum()

def second_kernal(x, y):
    sigma=1
    kernal2 = np.zeros((kernel_size, kernel_size))
    for i in range(-st, st+1):
        for j in range(-ed, ed+1):
            intensity_diff = abs(img[x,y] - img[x+i,y+j])
            kernal2[st+i][ed+j] = math.exp(-(intensity_diff**2)/(2*sigma**2))
    return kernal2

result = np.zeros((height, width), dtype=np.float32)
for x in range(st, height-st):
  for y in range(ed, width-ed):
     final_kernal = kernel*second_kernal(x, y)
     final_kernal = final_kernal/final_kernal.sum()
     imagePixel = img[x-st:x+st+1, y-ed:y+ed+1]
     result[x,y] = np.sum(imagePixel*final_kernal)
from skimage.exposure import rescale_intensity
result = rescale_intensity(result, in_range=(0, 255))
plt.imshow(img, "gray")
```



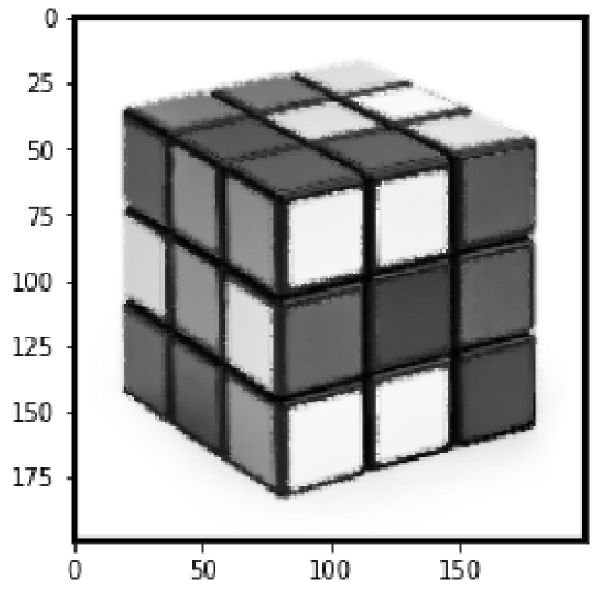


fig 4 : bilateral filter

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage.exposure import rescale_intensity
path = 'lena.png'
img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
height, width = img.shape
horizontal_mask = np.array([[1, 0, -1], [2, 0, -2], [1, 0, -1]])
vertical_mask = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])
st = horizontal_mask.shape[0]//2
ed = horizontal_mask.shape[1]//2
Gx = np.zeros((height, width))
Gy = np.zeros((height, width))
for row in range(st, height-st):
  for col in range(ed, width-ed):
    Gx[row, col] = np.sum(img[row-st:row+st+1, col-ed:col+ed+1] * horizontal_mask)
for row in range(st, height-st):
  for col in range(ed, width-ed):
    Gy[row, col] = np.sum(img[row-st:row+st+1, col-ed:col+ed+1] * vertical_mask)
G = np.sqrt(Gx**2 + Gy**2)
plt.title('Gx')
plt.imshow(Gx, cmap='gray')
plt.show()
plt.title('Gy')
plt.imshow(Gy, cmap='gray')
plt.show()
plt.title('sobel filter')
plt.imshow(G, cmap='gray')
```

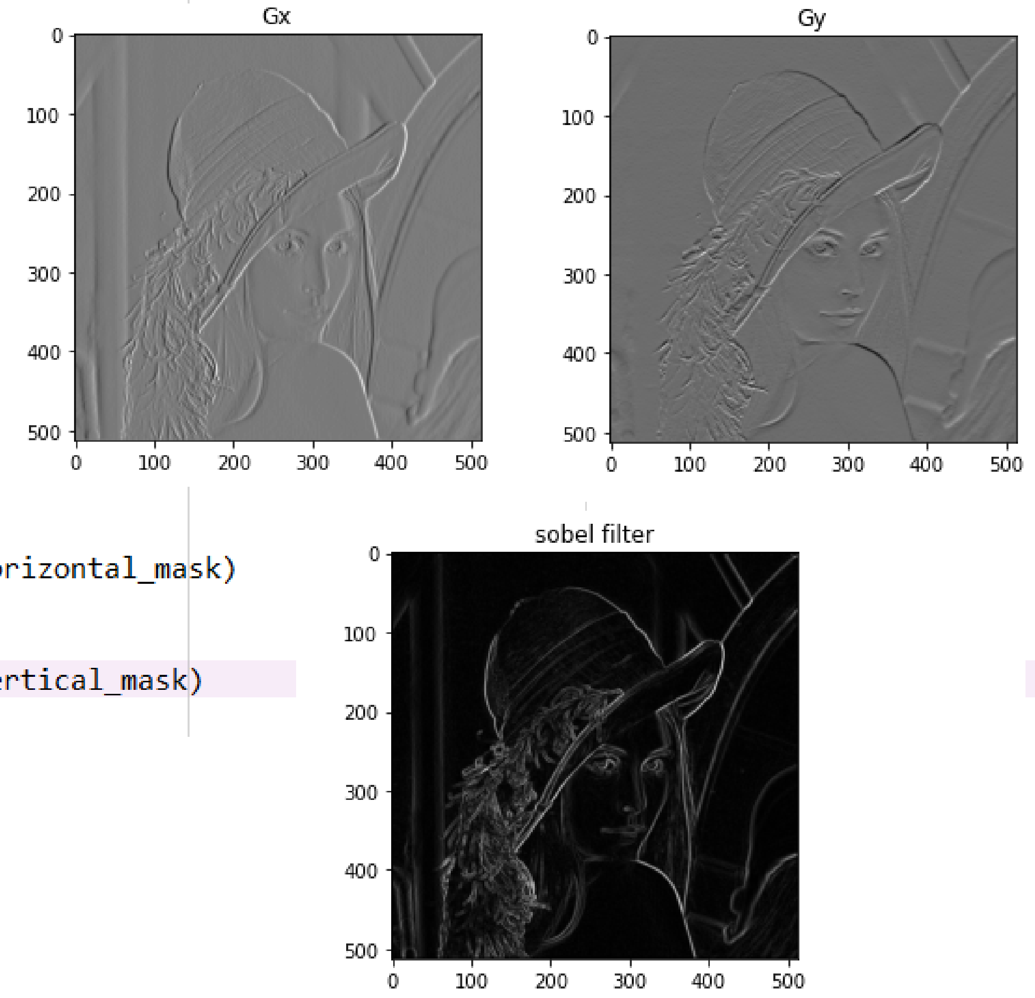
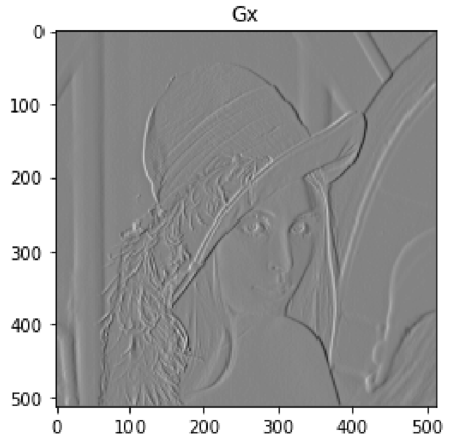
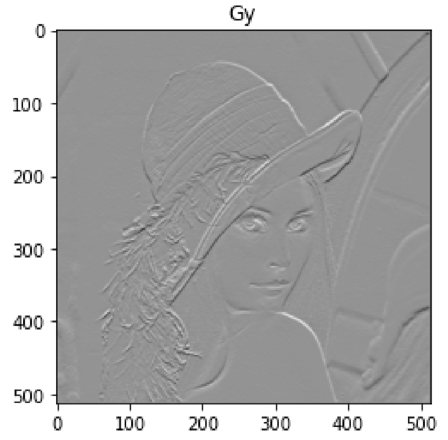
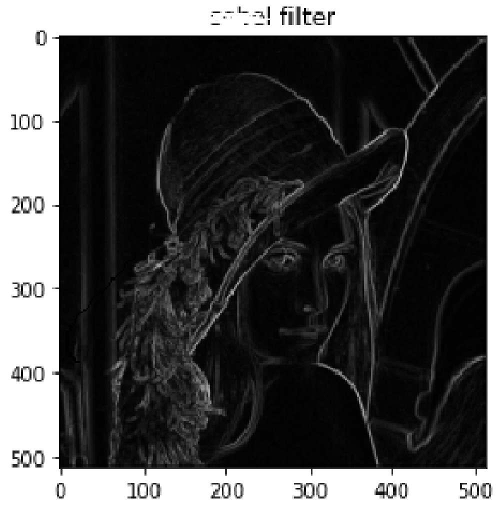


fig 5 : sobel filter

```python
path = 'lena.png'
img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
height, width = img.shape
horizontal_mask = np.array([[-3, 0, 3], [-10, 0, 10], [-3, 0, +3]])
vertical_mask = np.array([[-3, -10, -3], [0, 0, 0], [3, 10, 3]])
st = horizontal_mask.shape[0]//2
ed = horizontal_mask.shape[1]//2
Gx = np.zeros((height, width))
Gy = np.zeros((height, width))

for row in range(st, height-st):
  for col in range(ed, width-ed):
    Gx[row, col] = np.sum(img[row-st:row+st+1, col-ed:col+ed+1] * horizontal_mask)

for row in range(st, height-st):
  for col in range(ed, width-ed):
    Gy[row, col] = np.sum(img[row-st:row+st+1, col-ed:col+ed+1] * vertical_mask)
G = np.sqrt(Gx**2 + Gy**2)
plt.title('Gx')
plt.imshow(Gx, cmap='gray')
plt.show()
plt.title('Gy')
plt.imshow(Gy, cmap='gray')
plt.show()
plt.title('sobel filter')
plt.imshow(G, cmap='gray')
```







```python
path = 'lena.png'
img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
height, width = img.shape
horizontal_mask = np.array([[1, 0], [0, -1]])
vertical_mask = np.array([[0, 1], [-1, 0]])
st = horizontal_mask.shape[0]//2
ed = horizontal_mask.shape[1]//2
Gx = np.zeros((height, width))
Gy = np.zeros((height, width))

for row in range(st, height-st):
  for col in range(ed, width-ed):
    Gx[row, col] = np.sum(img[row-st:row+st, col-ed:col+ed] * horizontal_mask)

for row in range(st, height-st):
  for col in range(ed, width-ed):
    Gy[row, col] = np.sum(img[row-st:row+st, col-ed:col+ed] * vertical_mask)

G = np.sqrt(Gx**2 + Gy**2)
plt.title('Gx')
plt.imshow(Gx, cmap='gray')
plt.show()
plt.title('Gy')
plt.imshow(Gy, cmap='gray')
plt.show()
plt.title('robert filter')
plt.imshow(G, cmap='gray')
```

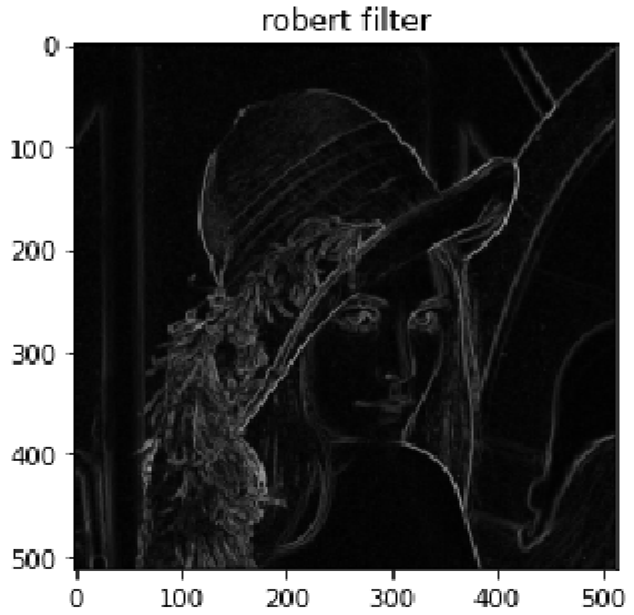


fig 7: robert filter

```
path = 'lena.png'
img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
height, width = img.shape
horizontal_mask = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])
vertical_mask = np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])
st = horizontal_mask.shape[0]//2
ed = horizontal_mask.shape[1]//2
Gx = np.zeros((height, width))
Gy = np.zeros((height, width))

for row in range(st, height-st):
  for col in range(ed, width-ed):
    Gx[row, col] = np.sum(img[row-st:row+st+1, col-ed:col+ed+1] * horizontal_mask)

for row in range(st, height-st):
  for col in range(ed, width-ed):
    Gy[row, col] = np.sum(img[row-st:row+st+1, col-ed:col+ed+1] * vertical_mask)

G = np.sqrt(Gx**2 + Gy**2)
plt.title('Gx')
plt.imshow(Gx, cmap='gray')
plt.show()
plt.title('Gy')
plt.imshow(Gy, cmap='gray')
plt.show()
plt.title('prewitt filter')
plt.imshow(G, cmap='gray')
```

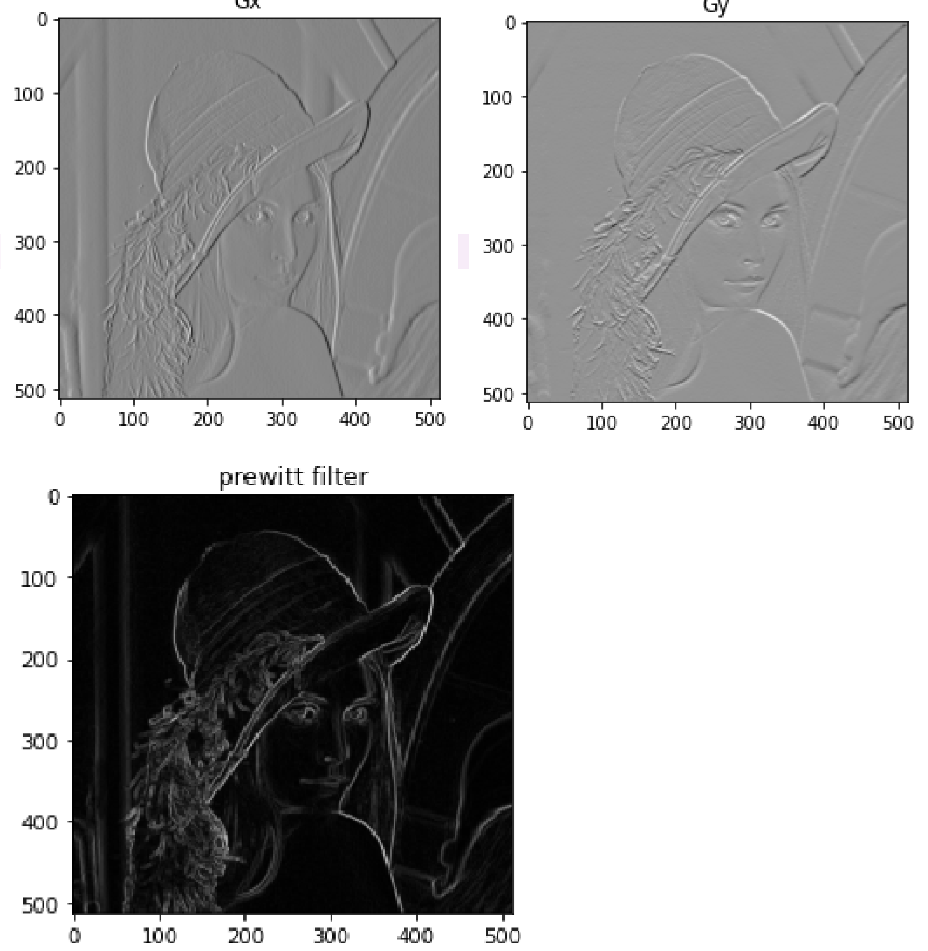
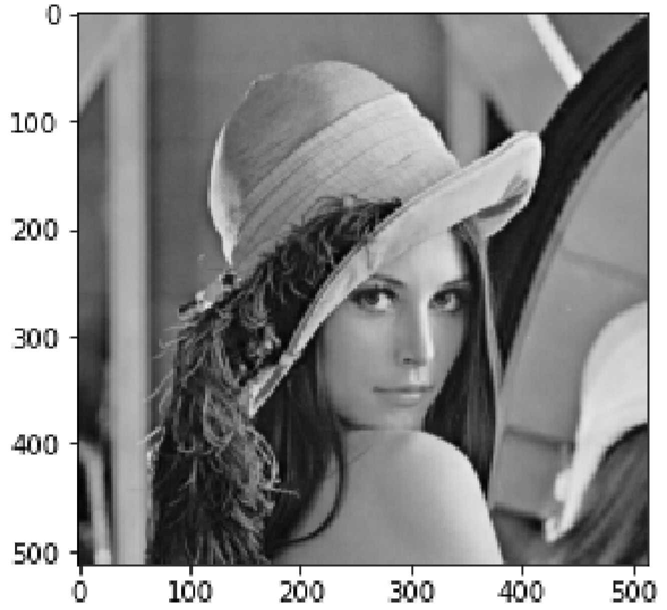


fig 8: prewitt filter

```
path = 'lena.png'
img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
height, width = img.shape
kernel_size = 7
sigma = (kernel_size - 1) // 6
kernel = np.zeros((kernel_size, kernel_size))
st = kernel.shape[0]//2
ed = kernel.shape[1]//2

import math
def gaussian_filter(x, y, sigma):
  PI = math.pi
  return (1/(2*PI*sigma**2))*math.exp(-(x**2+y**2)/(2*sigma**2))

for i in range(-st, st+1):
  for j in range(-ed, ed+1):
    kernel[i+st][j+ed] = gaussian_filter(i, j, sigma)

gaussian_img = np.zeros((height, width))

for row in range(st, height-st):
  for col in range(ed, width-ed):
    img_portion = img[row-st:row+st+1, col-ed:col+ed+1]
    # img_portion = img_portion[::-1, ::-1]
    gaussian_img[row][col] = np.sum(img_portion*kernel)

plt.imshow(img, cmap='gray')
plt.show()
plt.imshow(gaussian_img, cmap='gray')
plt.show()
```



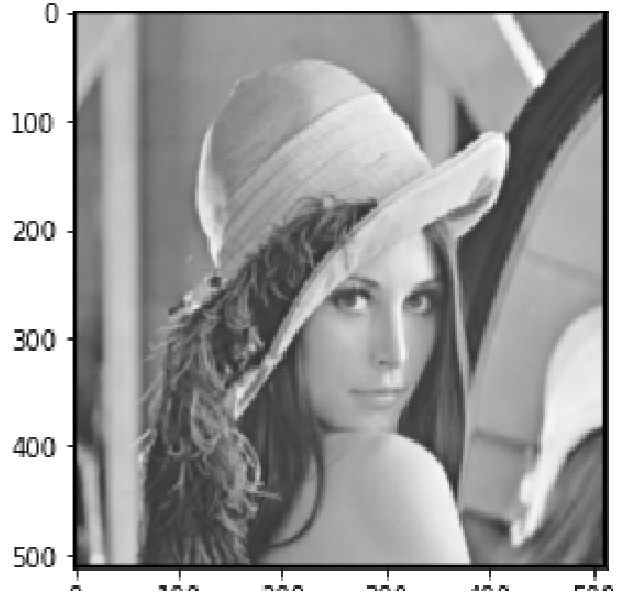


fig 9: gaussian filter