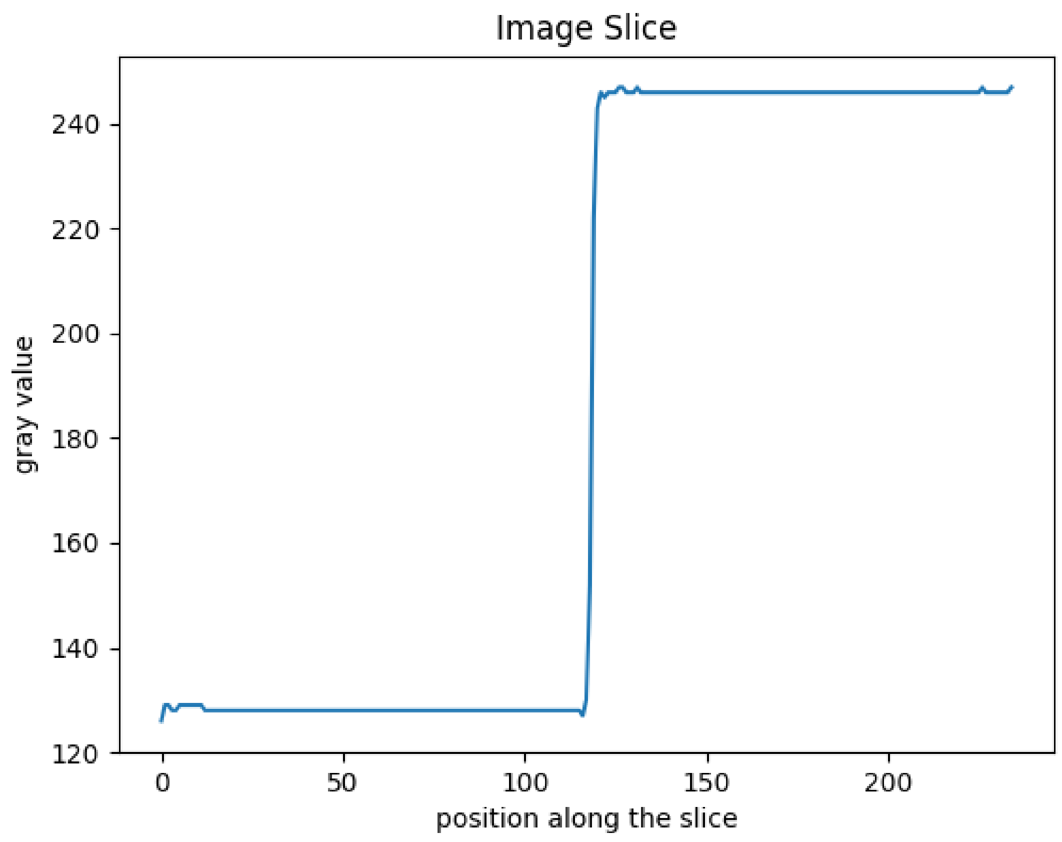
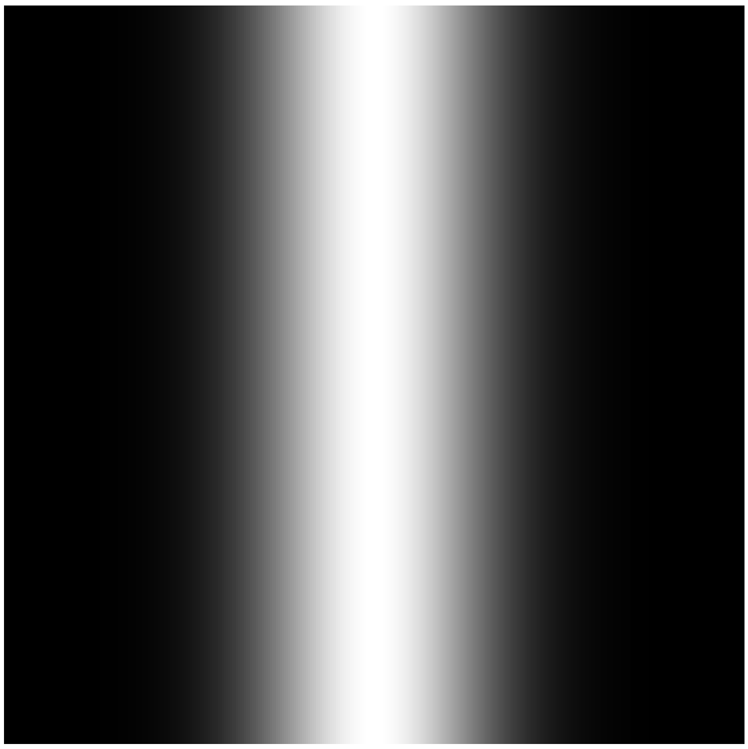
Edge detection

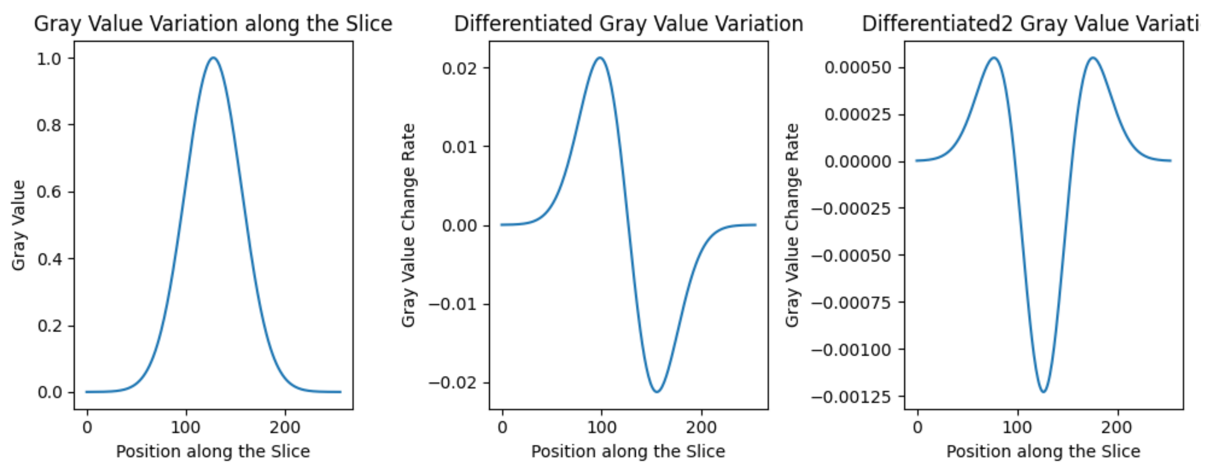
Edge detection on an image highlights the areas with significant changes. By applying horizontal and vertical derivatives to the image, the variations in the horizontal and vertical directions are emphasized, thereby achieving edge detection.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
image_path = 'image/01.png'
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
slice_index = image.shape[0] // 2
slice = image[slice_index, :]
plt.plot(slice)
plt.title('Image Slice')
plt.xlabel('position along the slice')
plt.ylabel('gray value')
plt.show()
```

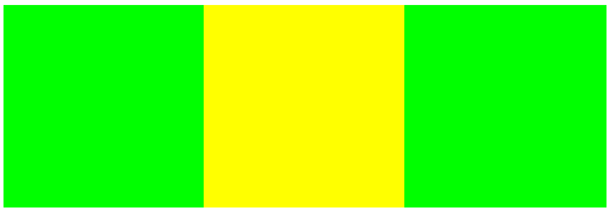


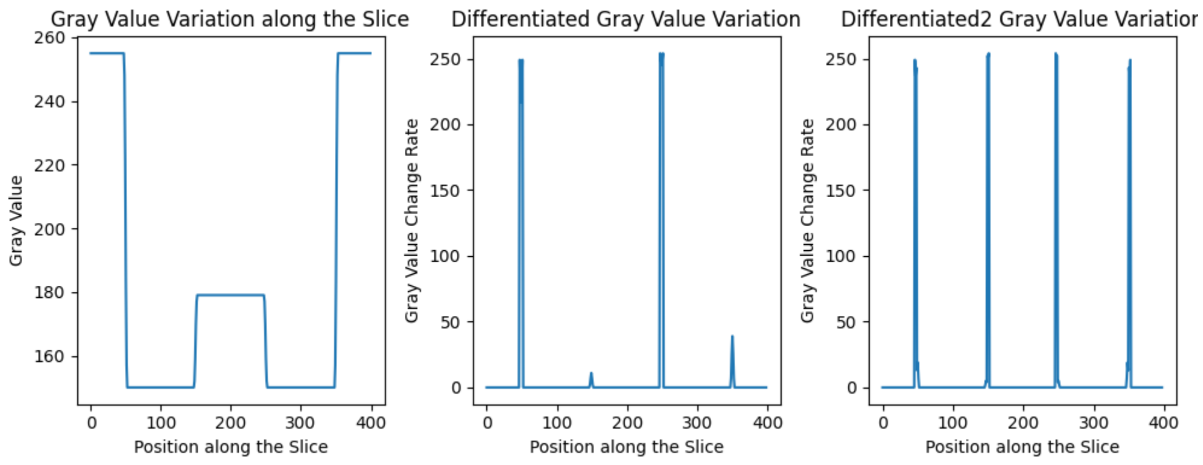
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
width, height = 256, 256
x = np.linspace(-1, 1, width)
gradient = np.exp(-x**2 * 10)
gradient_image = np.tile(gradient, (height, 1))
plt.imshow(gradient_image, cmap='gray', origin='upper')
plt.axis('off')
plt.show()
slice_index = gradient_image.shape[0] // 2
slice = gradient_image[slice_index,:]
diff_slice = np.diff(slice)
diff_slice2 = np.diff(diff_slice)
plt.figure(figsize=(10, 4))
plt.subplot(1, 3, 1)
plt.plot(slice)
plt.title('Gray Value Variation along the Slice')
plt.xlabel('Position along the Slice')
plt.ylabel('Gray Value')
plt.subplot(1, 3, 2)
plt.plot(diff_slice)
plt.title('Differentiated Gray Value Variation')
plt.xlabel('Position along the Slice')
plt.ylabel('Gray Value Change Rate')
plt.tight_layout()
plt.subplot(1, 3, 3)
plt.plot(diff_slice2)
plt.title('Differentiated2 Gray Value Variation')
plt.xlabel('Position along the Slice')
plt.ylabel('Gray Value Change Rate')
plt.tight_layout()
plt.show()
```



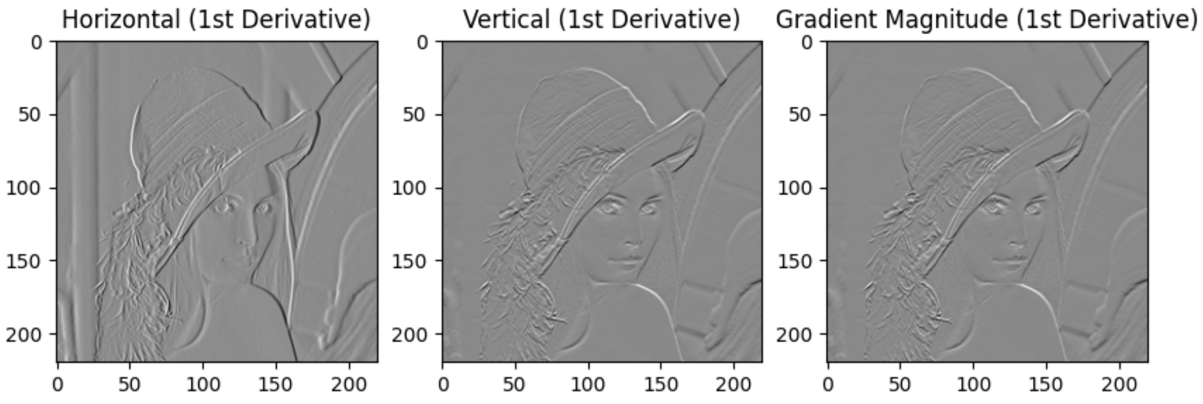


```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import numpy as np
import cv2

image = np.ones((200, 400, 3), dtype=np.uint8) * 255
color1 = (0, 255, 0)
color2 = (0, 255, 255)
color3 = (0, 255, 0)

cv2.rectangle(image, (50, 50), (150, 150), color1, -1)
cv2.rectangle(image, (150, 50), (250, 150), color2, -1)
cv2.rectangle(image, (250, 50), (350, 150), color3, -1)
cv2.imshow("Image with Squares", image)
image = cv2.cvtColor(image, cv2.COLOR_RGBA2GRAY)
image = cv2.GaussianBlur(image, (5, 5), 0)
slice_index = image.shape[0] // 2
slice = image[slice_index,:]
diff_slice = np.diff(slice)
diff_slice2 = np.diff(diff_slice)
plt.figure(figsize=(10, 4))
plt.subplot(1, 3, 1)
plt.plot(slice)
plt.title('Gray Value Variation along the Slice')
plt.xlabel('Position along the Slice')
plt.ylabel('Gray Value')
plt.subplot(1, 3, 2)
plt.plot(diff_slice)
plt.title('Differentiated Gray Value Variation')
plt.xlabel('Position along the Slice')
plt.ylabel('Gray Value Change Rate')
plt.tight_layout()
plt.subplot(1, 3, 3)
plt.plot(diff_slice2)
plt.title('Differentiated2 Gray Value Variation')
plt.xlabel('Position along the Slice')
plt.ylabel('Gray Value Change Rate')
plt.tight_layout()
plt.show()
```

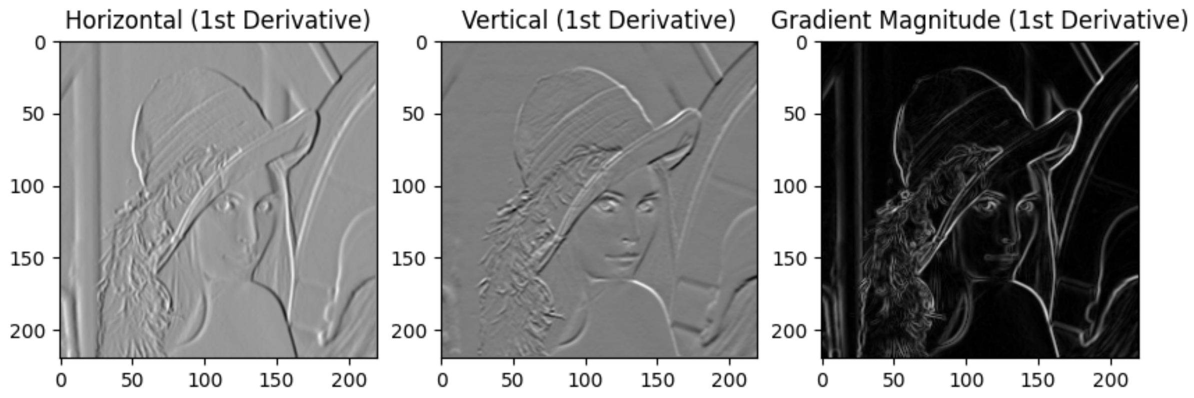




```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import numpy as np
import cv2

image = cv2.imread('image/lenna.png', cv2.IMREAD_GRAYSCALE)
cv2.imshow('image', image)
image = image.astype(np.float32) / 5.0
diff_image = np.zeros_like(image)
diff_image_x = np.zeros_like(image)
for i in range(image.shape[0]):
    diff_image[i, 1:] = np.diff(image[i, :])
    diff_image_x[i, 1:] = np.diff(image[i, :])
diff_image_y = np.zeros_like(image)
for j in range(image.shape[1]):
    diff_image[1:, j] = np.diff(image[:, j])
    diff_image_y[1:, j] = np.diff(image[:, j])
plt.figure(figsize=(10, 7))
plt.subplot(1, 3, 1)
plt.title('Horizontal (1st Derivative)')
plt.imshow(diff_image_x, cmap='gray')
plt.subplot(1, 3, 2)
plt.title('Vertical (1st Derivative)')
plt.imshow(diff_image_y, cmap='gray')
plt.subplot(1, 3, 3)
plt.title('Gradient Magnitude (1st Derivative)')
plt.imshow(diff_image, cmap='gray')
plt.show()
```

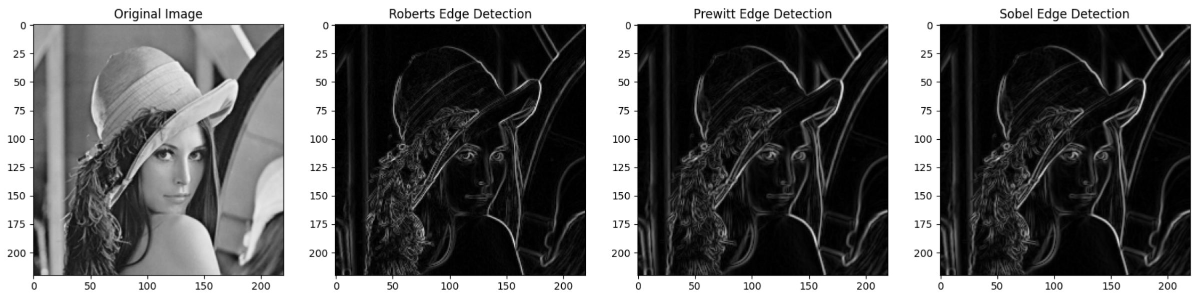

Horizontal (1st Derivative)
Vertical (1st Derivative)
Gradient Magnitude (1st Derivative)

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import numpy as np
import cv2
image = cv2.imread('image/lenna.png', 0)
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
gradient_magnitude = cv2.magnitude(sobel_x, sobel_y)
plt.figure(figsize=(10, 7))
plt.subplot(1, 3, 1)
plt.title('Horizontal (1st Derivative)')
plt.imshow(sobel_x, cmap='gray')
plt.subplot(1, 3, 2)
plt.title('Vertical (1st Derivative)')
plt.imshow(sobel_y, cmap='gray')
plt.subplot(1, 3, 3)
plt.title('Gradient Magnitude (1st Derivative)')
plt.imshow(gradient_magnitude, cmap='gray')
plt.show()
```

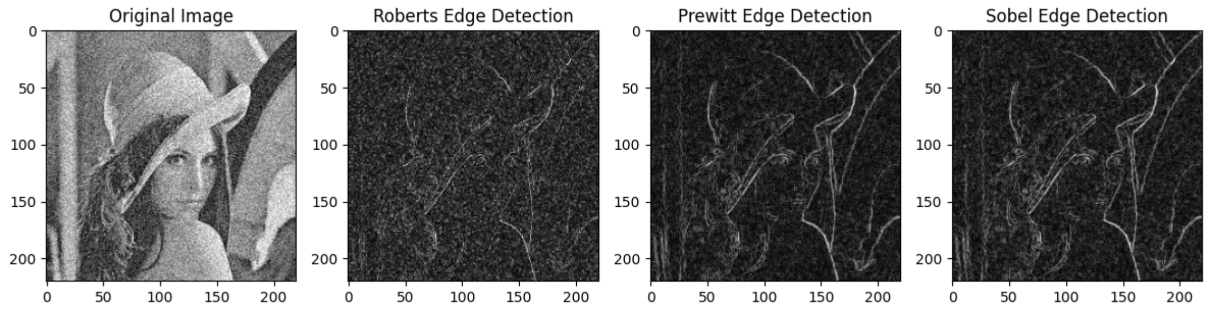

Horizontal (1st Derivative)
Vertical (1st Derivative)
Gradient Magnitude (1st Derivative)

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

image = cv2.imread('image/lenna.png', cv2.IMREAD_GRAYSCALE)
image = np.float32(image)
# Roberts
roberts_x = np.array([[1, 0], [0, -1]])
roberts_y = np.array([[0, 1], [-1, 0]])
roberts_edge_x = cv2.filter2D(image, -1, roberts_x)
roberts_edge_y = cv2.filter2D(image, -1, roberts_y)
roberts_edge = cv2.magnitude(roberts_edge_x, roberts_edge_
y)
# Prewitt
prewitt_x = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])
prewitt_y = np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])
prewitt_edge_x = cv2.filter2D(image, -1, prewitt_x)
prewitt_edge_y = cv2.filter2D(image, -1, prewitt_y)
prewitt_edge = cv2.magnitude(prewitt_edge_x, prewitt_edge_
y)
```

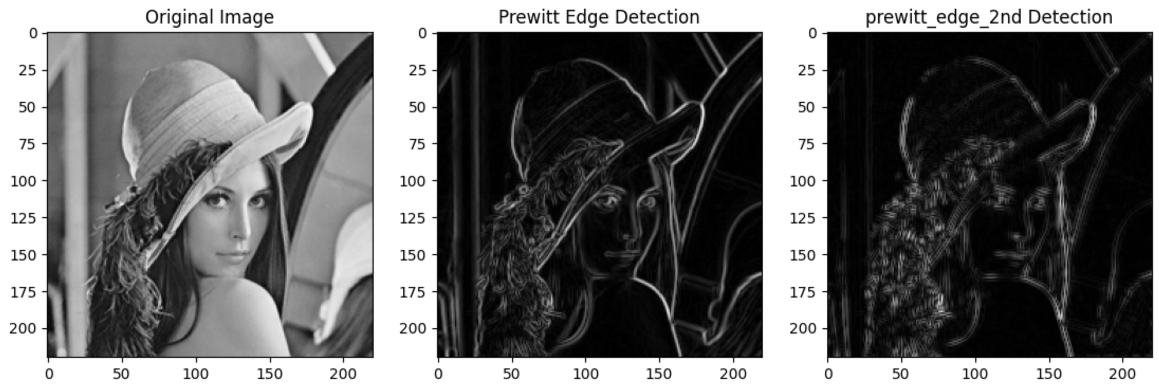
```
# Sobel
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
sobel_edge = cv2.magnitude(sobel_x, sobel_y)

plt.figure(figsize=(12, 12))
plt.subplot(1, 4, 1)
plt.title('Original Image')
plt.imshow(image, cmap='gray')
plt.subplot(1, 4, 2)
plt.title('Roberts Edge Detection')
plt.imshow(roberts_edge, cmap='gray')
plt.subplot(1, 4, 3)
plt.title('Prewitt Edge Detection')
plt.imshow(prewitt_edge, cmap='gray')
plt.subplot(1, 4, 4)
plt.title('Sobel Edge Detection')
plt.imshow(sobel_edge, cmap='gray')
plt.tight_layout()
plt.show()
```



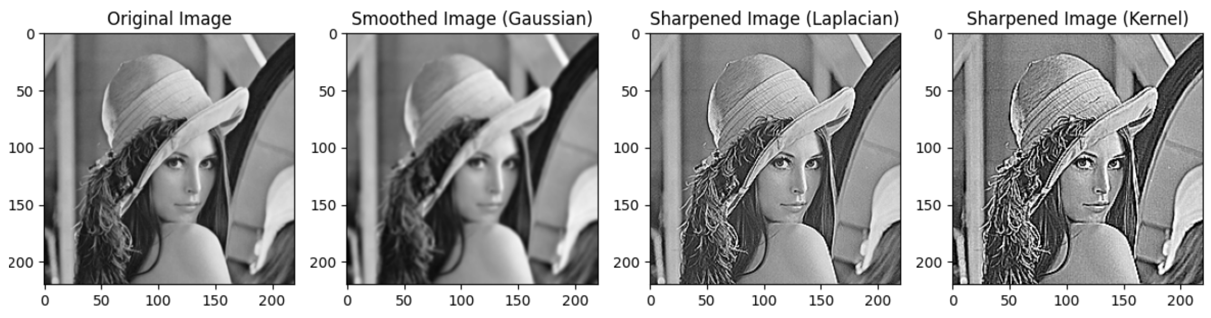
```
def add_gaussian_noise(image, mean=0, sigma=25):
    row, col = image.shape
    gaussian = np.random.normal(mean, sigma, (row, col))
    noisy_image = np.array(image, dtype=float) + gaussian
    noisy_image = np.clip(noisy_image, 0, 255)
    return noisy_image.astype(np.uint8)
# gaussian_noise
image = add_gaussian_noise(image)
```



```
prewitt_x = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])
prewitt_y = np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])
prewitt_edge_x = cv2.filter2D(image, -1, prewitt_x)
prewitt_edge_y = cv2.filter2D(image, -1, prewitt_y)
prewitt_edge = cv2.magnitude(prewitt_edge_x, prewitt_edge_y)

# 2nd
prewitt_x2 = np.array([[-1, 2, -1], [-1, 2, -1], [-1, 2, -1]]
prewitt_y2 = np.array([[-1, -1, -1], [2, 2, 2], [-1, -1, -1]]
prewitt_x2_edge = cv2.filter2D(prewitt_edge_x, -1, prewitt_x2
prewitt_y2_edge = cv2.filter2D(prewitt_edge_y, -1, prewitt_y2
prewitt_edge_2nd = cv2.magnitude(prewitt_x2_edge, prewitt_y2_
```



```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
image = cv2.imread('image/lenna.png', cv2.IMREAD_GRAYSCALE)

smoothed_image = cv2.GaussianBlur(image, (5, 5), 0)

laplacian = cv2.Laplacian(image, cv2.CV_64F)
sharpened_image = np.uint8(np.clip(image - laplacian, 0, 25
5))

sharpen_kernel = np.array([[-1, -1, -1], [-1, 9, -1], [-1,
-1, -1]], dtype=np.float32)
sharpened_image_kernel = cv2.filter2D(image, -1, sharpen_ke
rnel)

plt.figure(figsize=(12, 12))
plt.subplot(1, 4, 1)
plt.title('Original Image')
plt.imshow(image, cmap='gray')
plt.subplot(1, 4, 2)
plt.title('Smoothed Image (Gaussian)')
plt.imshow(smoothed_image, cmap='gray')
plt.subplot(1, 4, 3)
plt.title('Sharpened Image (Laplacian)')
plt.imshow(sharpened_image, cmap='gray')
plt.subplot(1, 4, 4)
plt.title('Sharpened Image (Kernel)')
plt.imshow(sharpened_image_kernel, cmap='gray')
plt.tight_layout()
plt.show()
```


Original Image
Smoothed Image (Gaussian)
Sharpened Image (Laplacian)
Sharpened Image (Kernel)
0
50
100
150
200