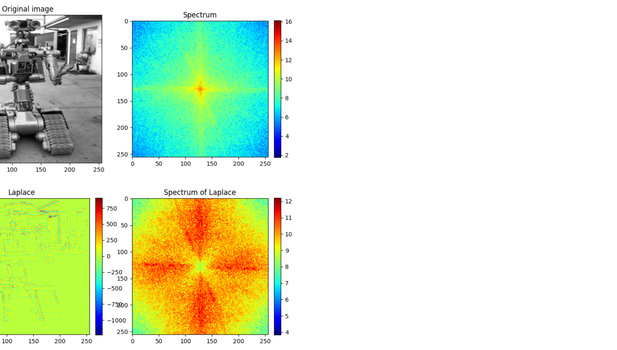
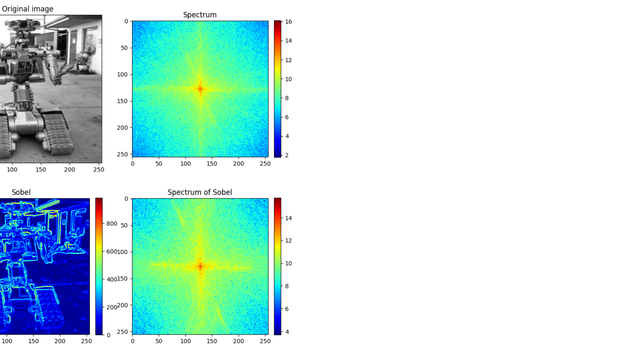
In [ ]
```python
import numpy as np
import matplotlib.pyplot as plt
import cv2

robot_img = cv2.imread("cv06_robot.bmp")
robot_img = cv2.cvtColor(robot_img, cv2.COLOR_BGR2RGB)

def D2_DFT(image):
    """
    2D Discrete Fourier Transform.
    :param image: vstupní obraz
    :return: 2D DFT obrazu
    """
    fft2 = np.fft.fft2(image)
    # shift zero-frequency component to the center of the spectrum
    fft2_shift = np.fft.fftshift(fft2)
    return fft2_shift, fft2


def plot_spectrum(image):
    """
    Vykreslení spektra obrazu.
    :param fft2_shift: spektrum obrazu
    """
    fft2_shift = D2_DFT(image)[0]
    log_real = np.log(np.abs(fft2_shift))
    plt.imshow(log_real, cmap="jet")
    plt.colorbar(fraction=0.046, pad=0.04)


def plot_histogram(image):
    """
    Vykreslení histogramu obrazu.
    :param image: vstupní obraz
    """
    plt.hist(cv2.calcHist([image], [0], None, [256], [0, 256]))
```

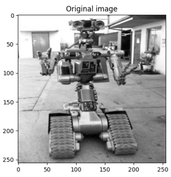
In [ ]
```python
def laplace_edge_detector(img):
    """Laplace edge detector with both first and second derivatives
    :param img: input image
    :return: edge image
    """

    g = img.astype(np.float32)

    # Filter version 2
    h1 = np.array([[2, -1, 2], [-1, -4, -1], [2, -1, 2]])
    h2 = np.array([[-1, 2, -1], [2, -4, 2], [-1, 2, -1]])

    # Apply filters to image
    filtered_img1 = np.zeros_like(g)
    for i in range(1, g.shape[0] - 1):
        for j in range(1, g.shape[1] - 1):
            filtered_img1[i, j] = np.sum(h1 * g[i - 1 : i + 2, j - 1 : j + 2])

    filtered_img2 = np.zeros_like(g)
    for i in range(1, g.shape[0] - 1):
        for j in range(1, g.shape[1] - 1):
            filtered_img2[i, j] = np.sum(h2 * g[i - 1 : i + 2, j - 1 : j + 2])

    # Combine filtered images to obtain edge image
    filtered_img = filtered_img1 + filtered_img2
    # threshold = np.max(np.abs(filtered_img)) * 0.125
    # edge_img = np.zeros_like(filtered_img)
    # edge_img[np.abs(filtered_img) > threshold] = 255

    return filtered_img


robot_gray = cv2.cvtColor(robot_img, cv2.COLOR_RGB2GRAY)

plt.figure(figsize=(10, 10))

plt.subplot(2, 2, 1)
plt.title("Original image")
plt.imshow(robot_gray, cmap="gray")

plt.subplot(2, 2, 2)
plt.title("Spectrum")
plot_spectrum(robot_gray)

robot_lap = laplace_edge_detector(robot_gray)

plt.subplot(2, 2, 3)
plt.title("Laplace")
plt.imshow(robot_lap)
plt.imshow(robot_lap, cmap="jet")
plt.colorbar(fraction=0.046, pad=0.04)


plt.subplot(2, 2, 4)
plt. title("Spectrum of Laplace")
plot_spectrum(robot_lap)
```

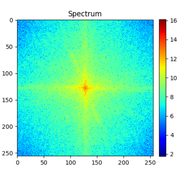


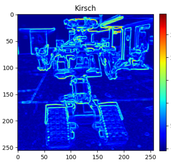
In [ ]
```python
def sobel_edge_detector(img):
    """Sobel edge detector
    :param img: input image
    :return: edge image
    """

    g = img.astype(np.float32)

    # Horizontal and vertical Sobel filters
    h1 = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])
    h2 = np.array([[0, 1, 2], [-1, 0, 1], [-2, -1, 0]])
    h3 = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])
    h4 = np.array([[2, 1, 0], [1, 0, -1], [0, -1, -2]])
    h5 = np.array([[1, 0, -1], [2, 0, -2], [1, 0, -1]])
    h6 = np.array([[0, -1, -2], [1, 0, -1], [2, 1, 0]])
    h7 = np.array([[-1, -2, -1], [-2, 0, 2], [-1, 2, 1]])
    h8 = np.array([[-2, -1, 0], [-1, 0, 1], [0, 1, 2]])

    filters = (h1, h2, h3, h4, h5, h6, h7, h8)

    # Apply filters to image
    filtered_imgs = []
    for filt in filters:
        filtered_img = np.zeros_like(g)
        for i in range(1, g.shape[0] - 1):
            for j in range(1, g.shape[1] - 1):
                filtered_img[i,j] = np.sum(filt * g[i - 1 : i + 2, j - 1 : j + 2])
        filtered_imgs.append(filtered_img)

    # find the maximum value fr filtered images
    max_val = np.max(filtered_imgs, axis=0)


    return max_val


plt.figure(figsize=(10, 10))

plt.subplot(2, 2, 1)
plt.title("Original image")
plt.imshow(robot_gray, cmap="gray")

plt.subplot(2, 2, 2)
plt.title("Spectrum")
plot_spectrum(robot_gray)

robot_sobel = sobel_edge_detector(robot_gray)

plt.subplot(2, 2, 3)
plt.title("Sobel")
plt.imshow(robot_sobel)
plt.imshow(robot_sobel, cmap="jet")
plt.colorbar(fraction=0.046, pad=0.04)

plt.subplot(2, 2, 4)
plt.title("Spectrum of Sobel")
plot_spectrum(robot_sobel)
```

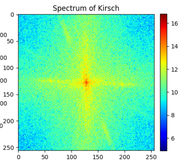


In [ ]
```python
def kirsch_edge_detector(img):
    """Kirsh edge detector
    :param img: input image
    :return: edge image
    """

    g = img.astype(np.float32)

    # Horizontal and vertical Sobel filters
    h1 = np.array([[3, 3, 3], [3, 0, 3], [-5, -5, -5]])
    h2 = np.array([[3, 3, 3], [-5, 0, 3], [-5, -5, 3]])
    h3 = np.array([[-5, -5, -5], [3, 0, 3], [3, 3, 3]])
    h4 = np.array([[-5, -5, 3], [-5, 0, 3], [3, 3, 3]])
    h5 = np.array([[3, -5, -5], [3, 0, -5], [3, 3, 3]])
    h6 = np.array([[3, 3, 3], [3, 0, -5], [3, -5, -5]])
    h7 = np.array([[3, 3, 3], [3, 0, 3], [-5, -5, -5]])
    h8 = np.array([[3, 3, 3], [-5, 0, 3], [-5, -5, -5]])
    filters = (h1, h2, h3, h4, h5, h6, h7, h8)

    # Apply filters to image
    filtered_imgs = []
    for filt in filters:
        filtered_img = np.zeros_like(g)
        for i in range(1, g.shape[0] - 1):
            for j in range(1, g.shape[1] - 1):
                filtered_img[i, j] = np.sum(filt * g[i - 1 : i + 2, j - 1 : j + 2])
        filtered_imgs.append(filtered_img)

    # find the maximum value fr filtered images
    max_val = np.max(filtered_imgs, axis=0)

    return max_val


plt.figure(figsize=(10, 10))

plt.subplot(2, 2, 1)
plt.title("Original image")
plt.imshow(robot_gray, cmap="gray")

plt.subplot(2, 2, 2)
plt.title("Spectrum")
plot_spectrum(robot_gray)

robot_kirsch = kirsch_edge_detector(robot_gray)

plt.subplot(2, 2, 3)
plt.title("Kirsch")
plt.imshow(robot_kirsch)
plt.imshow(robot_kirsch, cmap="jet")
plt.colorbar(fraction=0.046, pad=0.04)

plt.subplot(2, 2, 4)
plt.title("Spectrum of Kirsch")
plot_spectrum(robot_kirsch)
```


Original image
0
50
100
150
200
250
0
50
100
150
200
250


Spectrum
0
50
100
150
200
250
0
50
100
150
200
250
16
14
12
10
8
6
4
2


Kirsch
0
50
100
150
200
250
0
50
100
150
200
250
2500
2000
1500
1000
500
0


Spectrum of Kirsch
0
50
100
150
200
250
0
50
100
150
200
250
16
14
12
10
8
6