Pengolahan Citra

5. Filtering LPH dan HPF in domain frequency

Dosen Pengampu

Hero Yudo Martono ST, MT

Disusun Oleh :

Nama : M. Faza Nur Husain

Nrp : 3121550004

D3 PJJ AK TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

TAHUN AKADEMIK 2021/2022

Source Code :

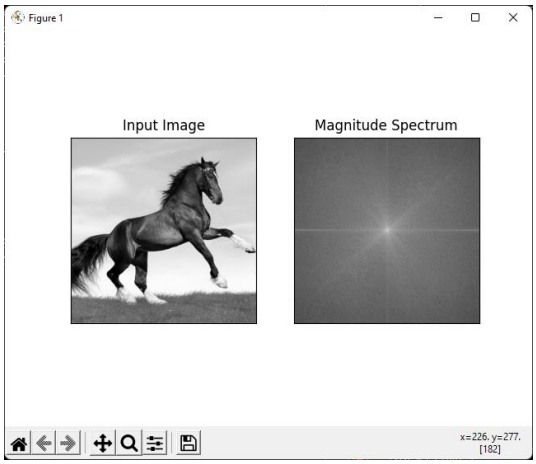
```
def spektrum():
    img = cv2.imread('gambar/kuda.jpg', 0)
    img_float32 = np.float32(img)
    dft = cv2.dft(img_float32, flags=cv2.DFT_COMPLEX_OUTPUT)
    dft_shift = np.fft.fftshift(dft)
    magnitude_spectrum = 20 * \
        np.log(cv2.magnitude(dft_shift[:, :, 0], dft_shift[:, :,
1]))
    plt.subplot(121), plt.imshow(img, cmap='gray')
    plt.title('Input Image'), plt.xticks([]), plt.yticks([])
    plt.subplot(122), plt.imshow(magnitude_spectrum, cmap='gray')
    plt.title('Magnitude Spectrum'), plt.xticks([]),
plt.yticks([])
    plt.show()
    return


def spektrum2():
    img = cv2.imread('gambar/kuda.jpg', 0)
    f = np.fft.fft2(img)
    fshift = np.fft.fftshift(f)
    magnitude_spectrum = 20*np.log(np.abs(fshift))
    plt.subplot(121), plt.imshow(img, cmap='gray')
    plt.title('Input Image'), plt.xticks([]), plt.yticks([])
    plt.subplot(122), plt.imshow(magnitude_spectrum, cmap='gray')
    plt.title('Magnitude Spectrum'), plt.xticks([]),
plt.yticks([])
    plt.show()
    return


def afterhpfjet():
    img = cv2.imread('gambar/kuda.jpg', 0)
    f = np.fft.fft2(img)
    fshift = np.fft.fftshift(f)
    magnitude_spectrum = 20*np.log(np.abs(fshift))
    rows, cols = img.shape
    crow, ccol = int(rows/2), int(cols/2)
    print(crow, ccol)
    fshift[crow-30:crow+30, ccol-30:ccol+30] = 0
    f_ishift = np.fft.ifftshift(fshift)
    img_back = np.fft.ifft2(f_ishift)
    img_back = np.abs(img_back)

    plt.subplot(131), plt.imshow(img, cmap='gray')
    plt.title('Input Image'), plt.xticks([]), plt.yticks([])
    plt.subplot(132), plt.imshow(img_back, cmap='gray')
    plt.title('Image after HPF'), plt.xticks([]), plt.yticks([])
    plt.subplot(133), plt.imshow(img_back)
```

```
    plt.title('Result in JET'), plt.xticks([]), plt.yticks([])
    plt.show()
    return


def spektrum3():
    img = cv2.imread('gambar/kuda.jpg', 0)
    dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
    dft_shift = np.fft.fftshift(dft)
    rows, cols = img.shape
    crow, ccol = int(rows/2), int(cols/2)
    # create a mask first, center square is 1, remaining all zeros
    mask = np.zeros((rows, cols, 2), np.uint8)
    mask[crow-30:crow+30, ccol-30:ccol+30] = 1
    # apply mask and inverse DF1
    fshift = dft_shift*mask
    f_ishift = np.fft.ifftshift(fshift)
    img_back = cv2.idft(f_ishift)
    img_back = cv2.magnitude(img_back[:, :, 0], img_back[:, :, 1])
    plt.subplot(121), plt.imshow(img, cmap='gray')
    plt.title('Input Image'), plt.xticks([]), plt.yticks([])
    plt.subplot(122), plt.imshow(img_back, cmap='gray')
    plt.title('Magnitude Spectrum'), plt.xticks([]),
plt.yticks([])
    plt.show()
    return


def lapsobel():
    img = cv2.imread("gambar/kuda.jpg", 0)
    laplacian = cv2. Laplacian(img, cv2.CV_64F)
    sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)
    sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=5)

    plt.subplot(2, 2, 1), plt.imshow(img, cmap='gray')
    plt.title('Original'), plt.xticks([]), plt.yticks([])
    plt.subplot(2, 2, 2), plt.imshow(laplacian, cmap='gray')
    plt.title('Laplacian'), plt.xticks([]), plt.yticks([])
    plt.subplot(2, 2, 3), plt.imshow(sobelx, cmap='gray')
    plt.title('Sobel X'), plt.xticks([]), plt.yticks([])
    plt.subplot(2, 2, 4), plt.imshow(sobely, cmap='gray')
    plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])
    plt.show()
    return


def hpffilter():
    # simple averaging filter without scaling parameter
    mean_filter = np.ones((3, 3))
    # creating a guassian filter
    x = cv2.getGaussianKernel(5, 10)
```

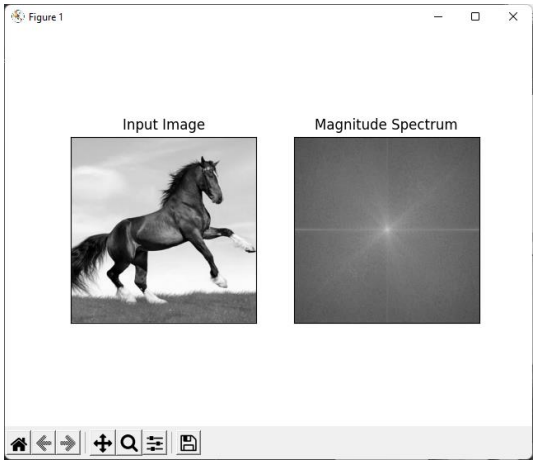
```
    gaussian = x*x.T

    # different edge detecting
    # scharr in x-direction
    scharr = np.array([[-3, 0, 3],
                       [-10, 0, 10],
                       [-3, 0, 3]])
    # sobel in x direction
    sobel_x = np.array([[-1, 0, 1],
                       [-2, 0, 2],
                       [-1, 0, 1]])
    # sobel in y direction
    sobel_y = np.array([[-1, -2, -1],
                       [0, 0, 0],
                       [1, 2, 1]])
    # :Laplacian
    laplacian = np.array([[0, 1, 0],
                          [1, -4, 1],
                           [0, 1, 0]])

    filters = [mean_filter, gaussian, laplacian, sobel_x, sobel_y,
scharr]
    filter_name = ['mean filter', 'gaussian', 'laplacian',
'sobel_x',
                   'sobel_y', 'scharr_x']
    fft_filters = [np.fft.fft2(x) for x in filters]
    fft_shift = [np.fft.fftshift(y) for y in fft_filters]
    mag_spectrum = [np.log(np.abs(z)+1) for z in fft_shift]

    for i in range(6):
        plt.subplot(2, 3, i+1), plt.imshow(mag_spectrum[i],
cmap='gray')
        plt.title(filter_name[i]), plt.xticks([]), plt.yticks([])

    plt.show()
    return
```

Output :

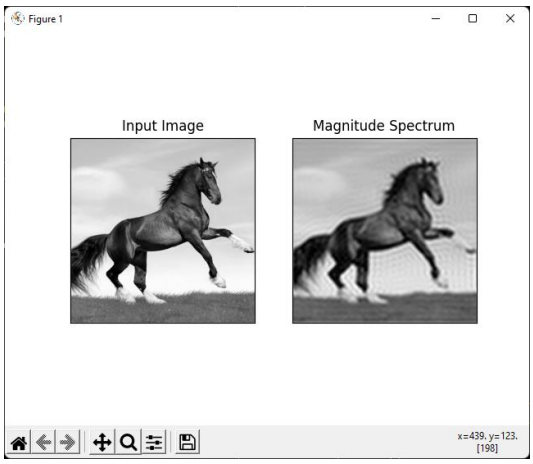
spektrum()



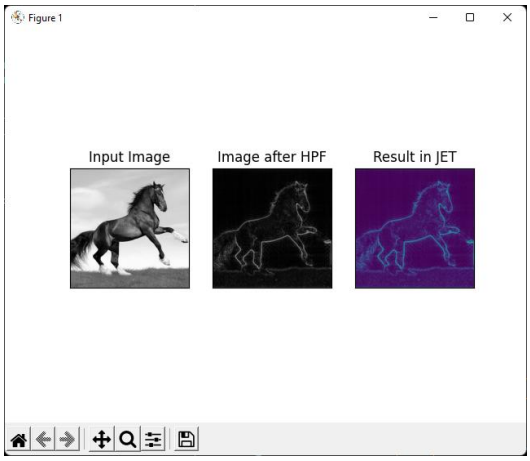
spektrum2()



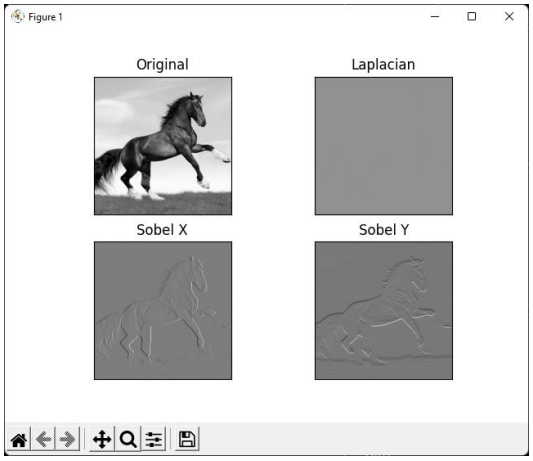
spektrum3()



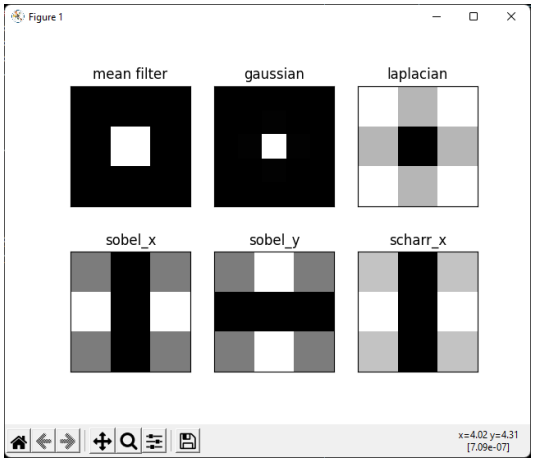
afterhpfjet()



lapsobel()



hpffilter()



https://github.com/FazaZas/pengolahan_citra.git