DOMAIN FREKUENSI

PENGOLAHAN CITRA DIGITAL

GITA PRATIWI RUNDULEMO MANGKEY

F 551 20 109

C

PROGRAM STUDI S1 TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

FALKUTAS TEKNIK

UNIVERSITAS TADULAKO

2022

A. Tujuan

1. Mahasiswa mampu memahami apa itu Domain Frekuensi.
2. Mahasiswa mampu membuat program sederhana Perbaikan Citra Domain Frekuensi.
3. Mahasiswa mampu mengimplementasikan program Perbaikan Citra Domain Frekuensi.

B. Teori Dasar

Domain frekuensi adalah metode yang digunakan untuk menganalisis data. Ini mengacu pada analisis fungsi matematis atau sinyal berkenaan dengan frekuensi. Analisis domain frekuensi banyak digunakan di bidang-bidang seperti teknik sistem kontrol, elektronika dan statistik. Representasi domain frekuensi juga dapat mencakup informasi tentang pergeseran fasa yang harus diterapkan pada setiap sinusoida agar dapat menggabungkan kembali komponen frekuensi untuk memulihkan sinyal waktu asli. Analisis domain frekuensi sebagian besar digunakan untuk sinyal atau fungsi yang periodik dari waktu ke waktu. Ini tidak berarti bahwa analisis domain frekuensi tidak dapat digunakan dalam sinyal yang tidak periodik. Konsep yang paling penting dalam analisis domain frekuensi adalah transformasi. Transformasi dig nakan untuk mengubah fungsi domain waktu menjadi fungsi domain frekuensi dan sebaliknya. Transformasi yang paling umum digunakan dalam domain frekuensi adalah Transformasi Fourier.

Transformasi Fourier adalah alat pengolah gambar penting yang digunakan untuk menguraikan gambar menjadi komponen sinus dan kosinusnya. Keluaran dari transformasi merepresentasikan citra dalam domain frekuensi, sedangkan citra masukan merupakan ekivalen domain spasial. Pada citra domain frekuensi, setiap titik mewakili frekuensi tertentu yang terdapat pada citra domain spasial. Transformasi Fourier digunakan untuk mengubah sinyal dari bentuk apapun menjadi jumlah gelombang sinusoidal yang tak terbatas. Karena menganalisa fungsi sinusoidal lebih mudah daripada menganalisa fungsi berbentuk umum, metode ini sangat berguna dan banyak digunakan.

Dalam Transformasi Fourier terdapat dua jenis sampel diantaranya Discrete Fourier Transform (DFT) dan Fast Fourier Transform (FFT). Discrete Fourier Transform (DFT) adalah sampel Fourier Transform, sehingga tidak mengandung semua frekuensi yang membentuk gambar, tetapi hanya satu set sampel yang cukup besar untuk sepenuhnya

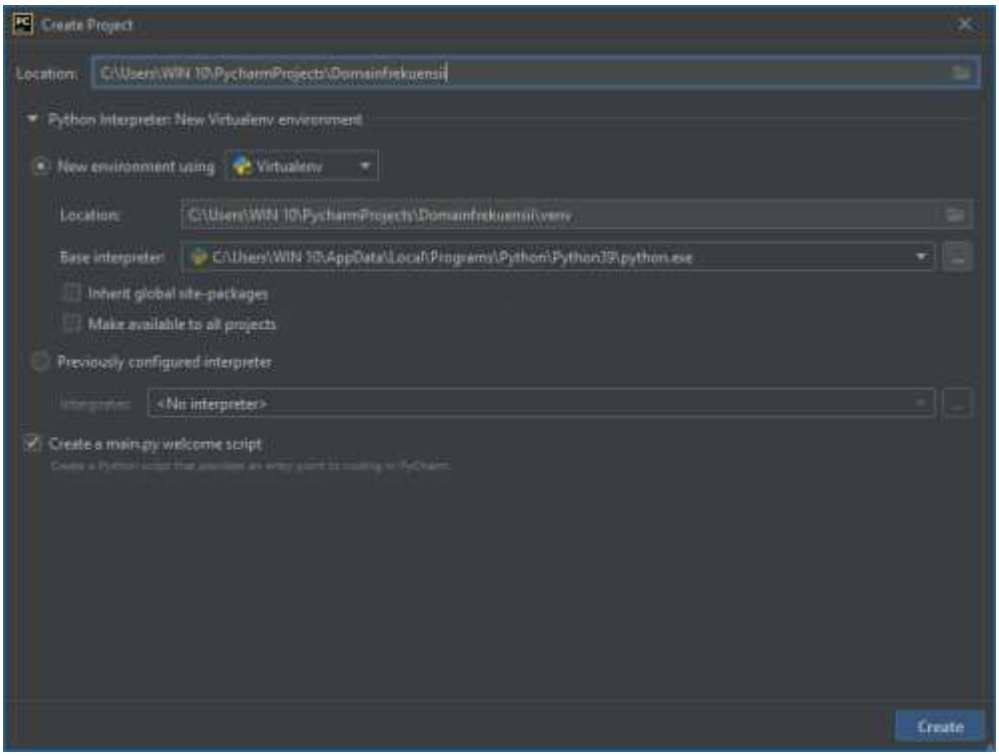
menggambarkan gambar domain spasial. Jumlah frekuensi sesuai dengan jumlah piksel pada citra domain spasial. Sedangkan Fast Fourier Transform (FFT) untuk menghitung DFT satu dimensi. Ini adalah peningkatan yang signifikan, khususnya untuk gambar besar. Ada berbagai bentuk FFT dan sebagian besar membatasi ukuran gambar input yang dapat ditransformasikan, seringkali ke persamaan $N = 2n$ di mana n adalah bilangan bulat

C. Praktikum

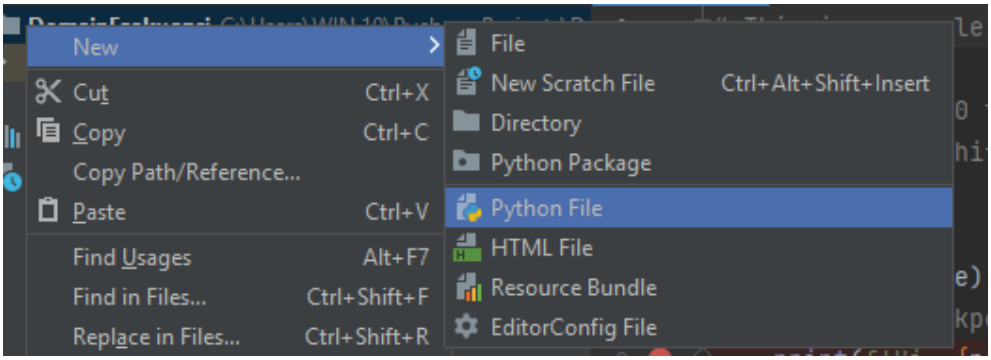
1. Sebelum memulai *coding*, pastikan sudah menginstal beberapa *library* yang diperlukan antara lain :
   a. *Opencv-Python.*
   b. *Numpy.*
   c. *Matplotlib*
2. Kemudian tambahkan sebuah gambar yang akan digunakan kedalam folder *img* pada program Domain Frekuensi. Serta perhatikan Ekstensi file gambar yang ditambahkan.

3. Selanjutnya masuk pada program Domain Frekuensi. Adapun Langkahlangkahnya sebagai berikut :

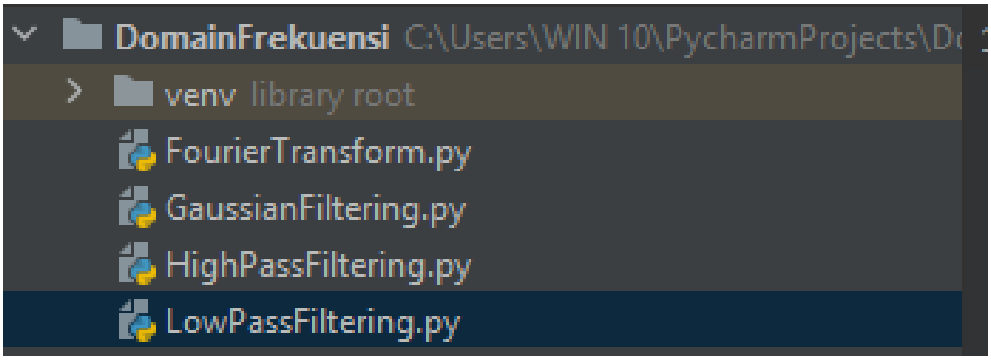
   a. Buat *project* baru dengan nama *Domain Frekuensi*



   b. Kemudian klik kanan pada folder *Domain Frekuensi – New – Python File*. Untuk membuat file baru.

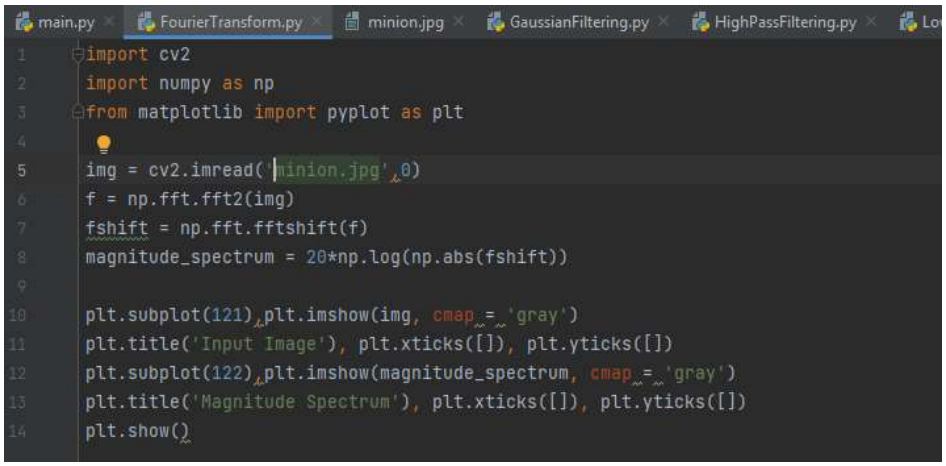


   c. File yang telah dibuat akan tampil seperti berikut



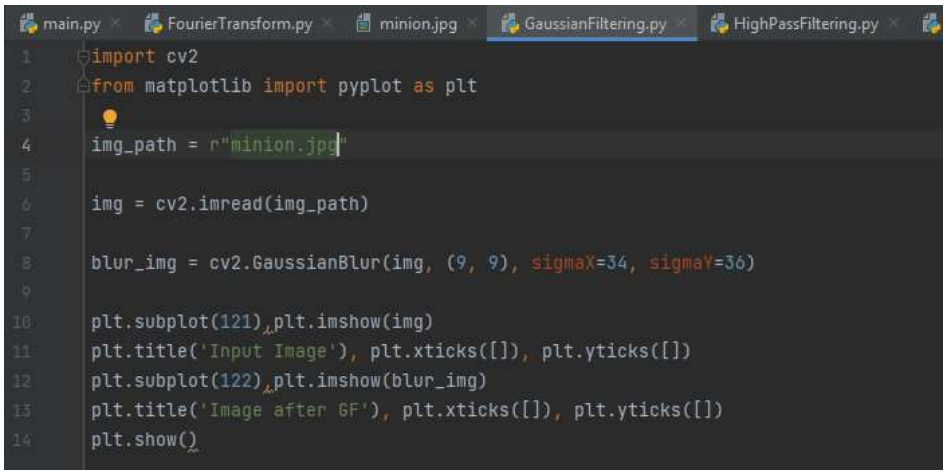
d. Selanjutnya ketikan kode program pada masing-masing file yang telah dibuat. Seperti berikut :

1.) *FourierTransform.py*



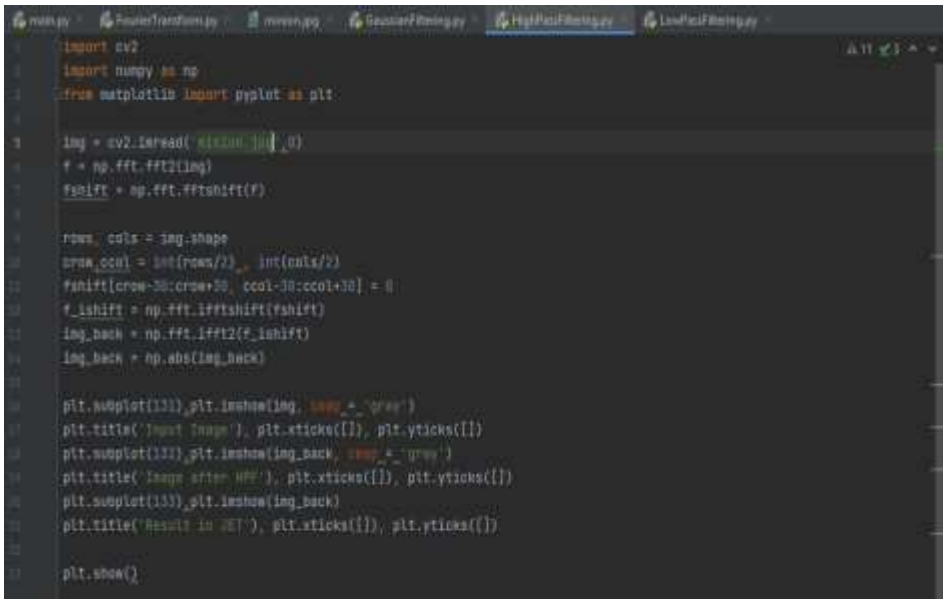
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('minion.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()
```

2.) *GaussianFiltering.py*



```
import cv2
from matplotlib import pyplot as plt

img_path = r"minion.jpg"

img = cv2.imread(img_path)

blur_img = cv2.GaussianBlur(img, (9, 9), sigmaX=34, sigmaY=36)

plt.subplot(121),plt.imshow(img)
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur_img)
plt.title('Image after GF'), plt.xticks([]), plt.yticks([])
plt.show()
```

3.) *HighPassFiltering.py*



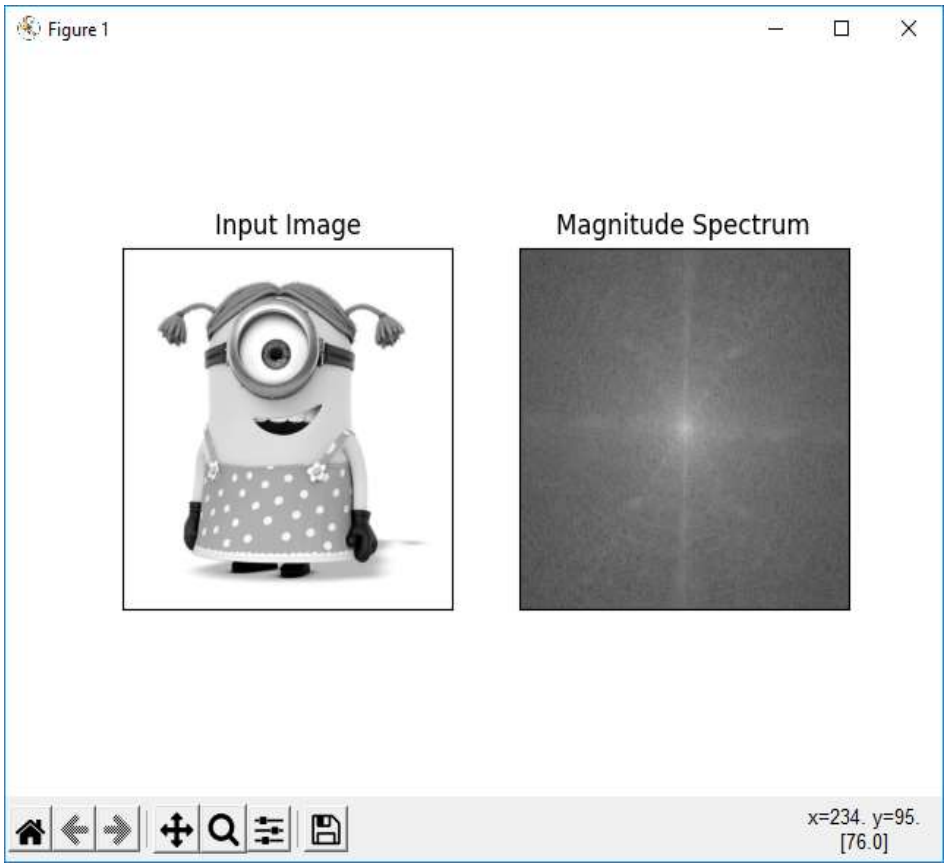
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('minion.jpg',0)
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)

rows, cols = img.shape
crow,ccol = int(rows/2) , int(cols/2)
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()
```

4.) *LowPassFiltering.py*

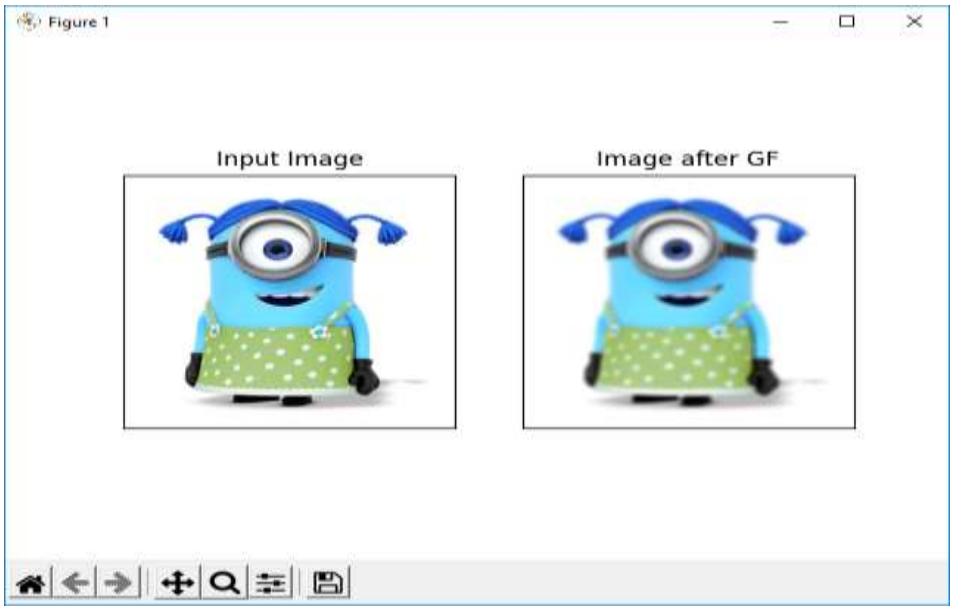
```python
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('minion.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 DFT
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('Image after LPF'), plt.xticks([]), plt.yticks([])
plt.show()
```

e. Kemudian Hasil dari program tersebut akan tampil seperti berikut :

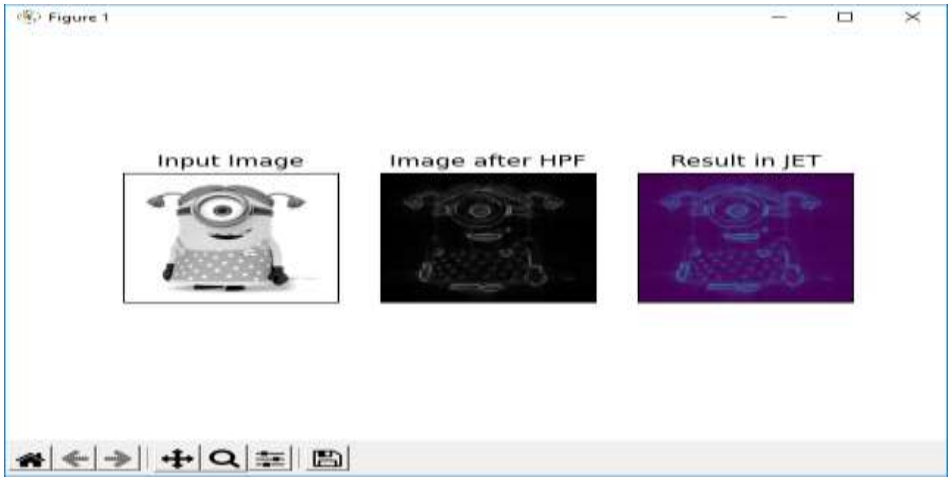
1.) *FourierTransform.py*



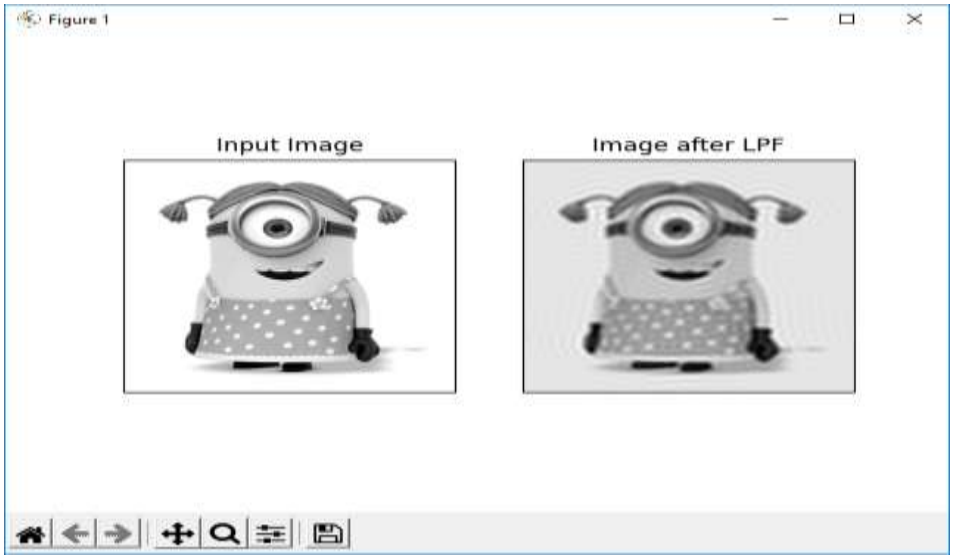
2.) *GaussianFiltering.py*



3.) *HighPassFiltering.py*



4.) *LowPassFiltering.py*



D. ANALISIS

Pada percobaan di atas dapat di analisis bahwa kita mmbuat *domain frekuensi* di mana kita akan membuat beberapa program. Yang pertama kita akan buat *fourier transform* dimana ini merupakan salah satu transformasi penting pada signal *processing* terutama pada *image processing*. Kode "*f = np.fft.fft2(img)*" digunakan untuk mengatur tampilan keseluruhan transformasi Fourier diskrit, dengan definisi dan konvensi yang digunakan. fshift = np.fft.fftshift(f) yang Menggeser suku frekuensi nol ke tengah larik. Untuk input dua dimensi, tukar kuadran pertama dan ketiga, dan kuadran kedua dan keempat. Dimana kode tersebut diatur untuk memberikan blurring pada gambar di two-dimensional. Sedangkan berikutnya, yaitu kode 20*np.log(np.abs(fshift)) merupakan pengaturan dimana diberikan paramete yang bisa digunakan pada fourier transformation. Berikutnya, tinggal mengatur citra gray pada gambar asli dan gambar hasil fourier transform yang akan ditampilkan dengan menggunakan subplot. Berikutnya, untuk gaussian filtering diberikan variabel blur_img untuk mengatur blur pada gambar. Kode tersebut lengkapnya adalah blur_img = cv2.GaussianBlur(img, (9, 9), sigmaX=34, sigmaY=36). Lalu, berikutnya mirip dengan kode sebelumnya yaitu pengaturan plt.subplot untuk mengatur subplot penampilan gambar. Olehnya hasil percobaan ini akan menampilkan gambar hasil yang berwarna dan blur. Kemudian, untuk high pass filter disini juga diberikan kode yang sama seperti fourier transform yaiitu pengaturan untuk blurring image two-dimensional dengan kode f = np.fft.fft2(img) dan fshift = np.fft.fftshift(f). Lalu, ada kode rows, cols = img.shape yang mengatur ketajaman gambar. Lalu, diberikan variabel img_back untuk hasil dari jet gambar daN diberikan fungsi untuk melakukan compute terhadap nilai absolut yaitu kode img_back = np.abs(img_back) yang sebelumnya didahului oleh kode img_back = np.fft.ifft2(f_ishift) ini, yang maksudnya adalah untuk memberikan blurring image pada img_backnya tadi. Untuk low pass filtering adalah proses filter yg mengambil citra dengan gradiasi intensitas yg halus dan perbedaan intensitas yg tinggi akan dikurangi atau dibuang. LPF dilakukan untuk menghilangkan ruang derau berfrekuensi tinggi dari sebuah gambar digital. Di sini terdapat kode cv2.dft(np.float32(img) yang digunakan untuk menemukan frekuensi domain. Lalu, ada flags = cv2.DFT_COMPLEX_OUTPUT) yang berfungsi untuk Ini mengembalikan hasil yang sama seperti sebelumnya, tetapi dengan dua saluran. Saluran pertama akan memiliki bagian nyata dari hasil dan saluran kedua akan memiliki bagian imajiner dari hasil. Gambar input harus dikonversi ke np.float32 terlebih dahulu. Lalu, diberikan juga

kode yang sama untuk blurring image dan juga img_shape untuk ketajaman gambarnya. Lalu, diberikan kode fshift = dft_shift*mask dan f_ishift = np.fft.ifftshift(fshift) untuk menerapkan mask dan inverse dari DFT. Dan kode img_back = cv2.idft(f_ishift) img_back = cv2.magnitude(img_back[:,:,0],img_back[:,:,1]) untuk mengembalikan hasilnya ke img_back.

E. Link Github

https://github.com/gitapratiwi20/F55120109_Gita-Pratiwi-Rundulemo-Mangkey_C_tugas2-pcd.git