

**DOMAIN FREKUENSI**  
**PENGOLAHAN CITRA DIGITAL**



**MUTIARA SARI**

**F55120091**

**C**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**  
**JURURSAN TEKNOLOGI INFORMASI FAKULTAS**  
**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 digunakan 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

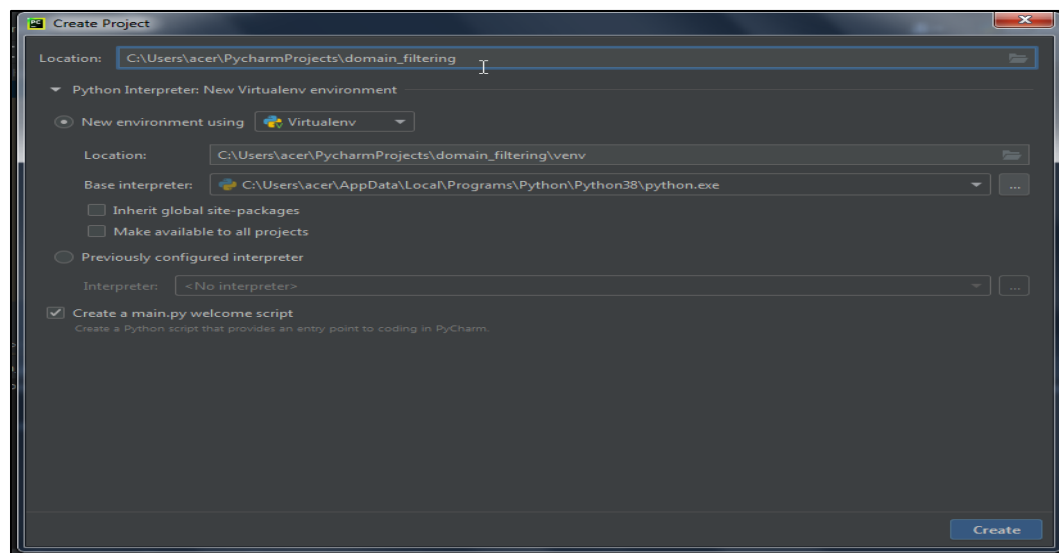
yang membentuk gambar, tetapi hanya satu set sampel yang cukup besar untuk sepenuhnya menggambarkan gambar domain spasial. Jumlah frekuensi sesuai dengan jumlah piksel pada citra domain spasial. Sedangkan Fast Fourier Transform (FFT) untuk menghitung DFT satu dimensi. Ini adalah peningkatan yang signifikan,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

### C. PRAKTIKUM

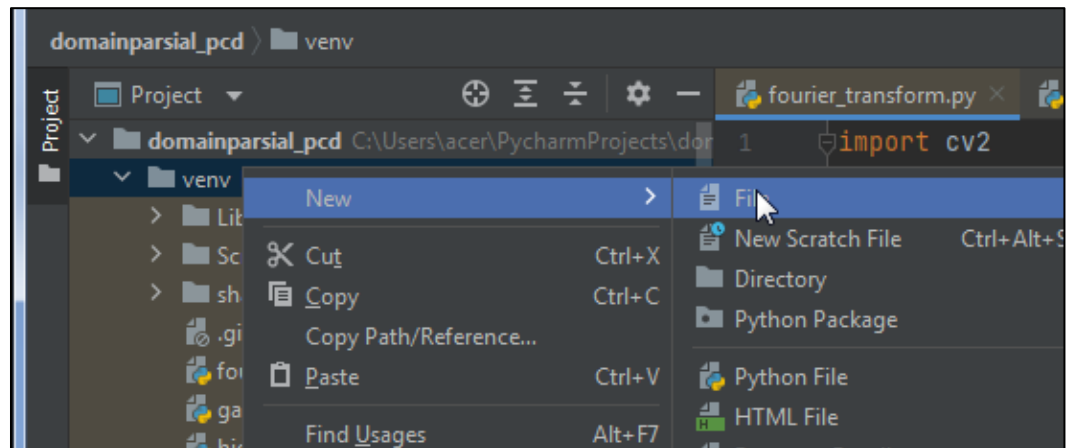
1. Sebelum memulai coding, pastikan sudah menginstal beberapa package yang diperlukan yaitu : opencv-python, numpy, dan matplotlib
2. Kemudian tambahkan gambar ke dalam folder project yaitu gambar berikut ini



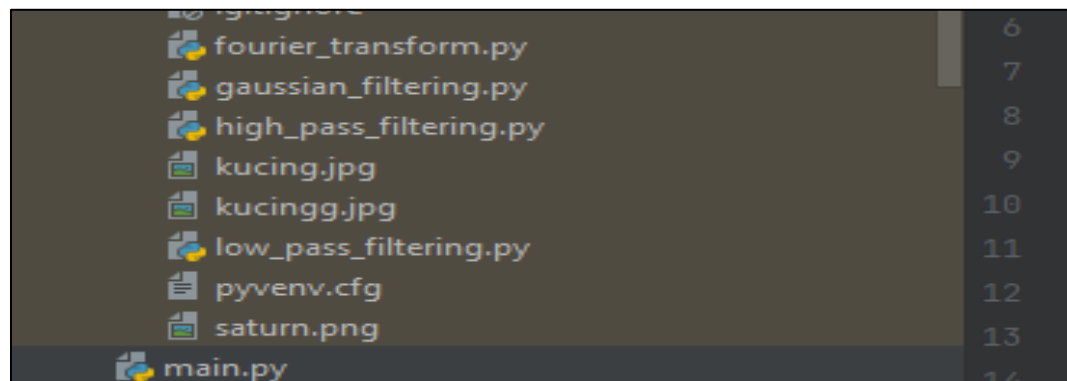
3. Selanjutnya, masuk pada program yang dibuat dengan cara seperti ini :
  - a. Buat project baru dengan nama domainfiltering



- b. Buat file baru

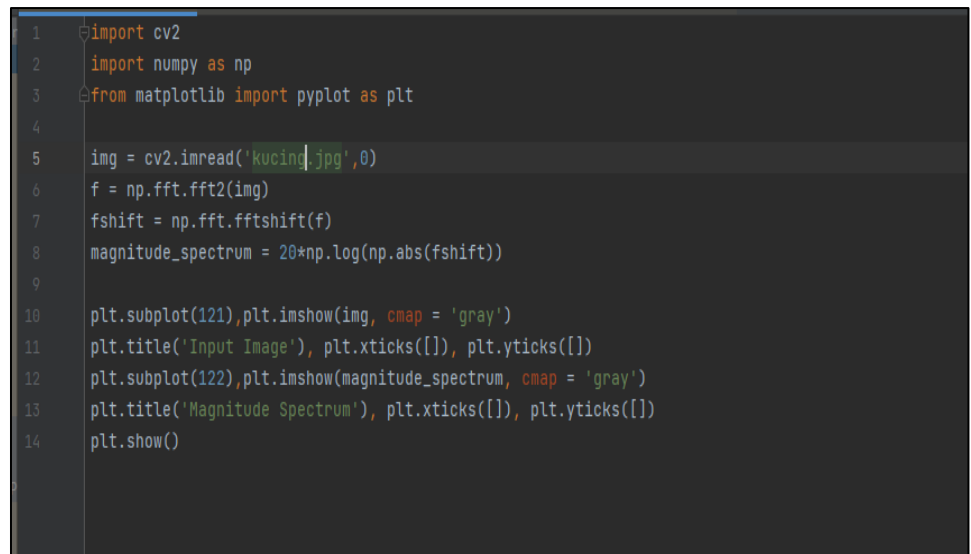


- c. File yang telah dibuat akan tampil seperti berikut



- d. Selanjutnya, ketikkan kode program pada masing-masing file yang telah dibuat.  
Seperti berikut ini :

- Fourier\_transform.py



- Gaussian\_filtering.py

```
fourier_transform.py × gaussian_filtering.py × high_pass_filtering.py × low_pass_filtering.py ×
1 import cv2
2 from matplotlib import pyplot as plt
3
4 img_path = "kucing.jpg"
5
6 img = cv2.imread(img_path)
7
8 blur_img = cv2.GaussianBlur(img, (9, 9), sigmaX=34, sigmaY=36)
9
10 plt.subplot(121), plt.imshow(img)
11 plt.title('Input Image'), plt.xticks([]), plt.yticks([])
12 plt.subplot(122), plt.imshow(blur_img)
13 plt.title('Image after GF'), plt.xticks([]), plt.yticks([])
14 plt.show()
```

- High\_pass\_filtering.py

```
fourier_transform.py × gaussian_filtering.py × high_pass_filtering.py × low_pass_filtering.py ×
1 import cv2
2 import numpy as np
3 from matplotlib import pyplot as plt
4
5 img = cv2.imread('kucing.jpg', 0)
6 f = np.fft.fft2(img)
7 fshift = np.fft.fftshift(f)
8
9 rows, cols = img.shape
10 crow, ccol = int(rows/2), int(cols/2)
11 fshift[crow-30:crow+30, ccol-30:ccol+30] = 0
12 f_ishift = np.fft.ifftshift(fshift)
13 img_back = np.fft.ifft2(f_ishift)
14 img_back = np.abs(img_back)
15
16 plt.subplot(131), plt.imshow(img, cmap = 'gray')
17 plt.title('Input Image'), plt.xticks([]), plt.yticks([])
18 plt.subplot(132), plt.imshow(img_back, cmap = 'gray')
19 plt.title('Image after HPF'), plt.xticks([]), plt.yticks([])
20 plt.subplot(133), plt.imshow(img_back)
21 plt.title('Result in JET'), plt.xticks([]), plt.yticks([])
22
23 plt.show()
```

- Low\_pass\_filtering.py

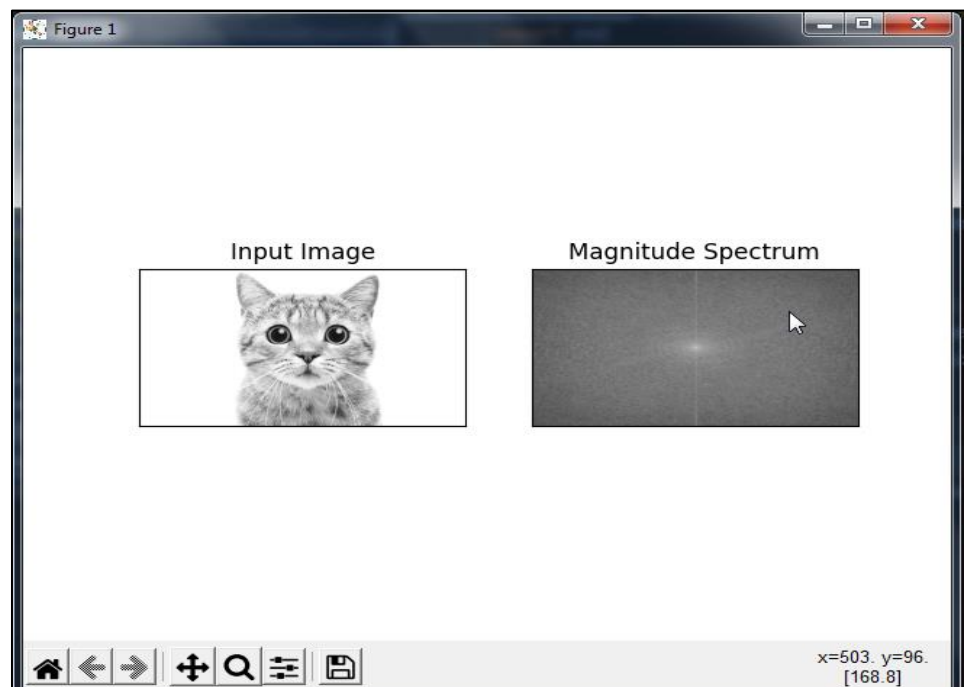
```

1 import numpy as np
2 import cv2
3 from matplotlib import pyplot as plt
4
5 img = cv2.imread('kucing.jpg',0)
6 dft = cv2.dft(np.float32(img),flags = cv2.DFT_COMPLEX_OUTPUT)
7 dft_shift = np.fft.fftshift(dft)
8 rows, cols = img.shape
9 crow,ccol = int(rows/2) , int(cols/2)
10 # create a mask first, center square is 1, remaining all zeros
11 mask = np.zeros((rows,cols,2),np.uint8)
12 mask[crow-30:crow+30, ccol-30:ccol+30] = 1
13
14 # apply mask and inverse DFT
15 fshift = dft_shift*mask
16 f_ishift = np.fft.ifftshift(fshift)
17 img_back = cv2.idft(f_ishift)
18 img_back = cv2.magnitude(img_back[:,0],img_back[:,1])
19
20 plt.subplot(121),plt.imshow(img, cmap = 'gray')
21 plt.title('Input Image'), plt.xticks([], plt.yticks([]))
22 plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
23 plt.title('Image after LPF'), plt.xticks([], plt.yticks([]))
24 plt.show()

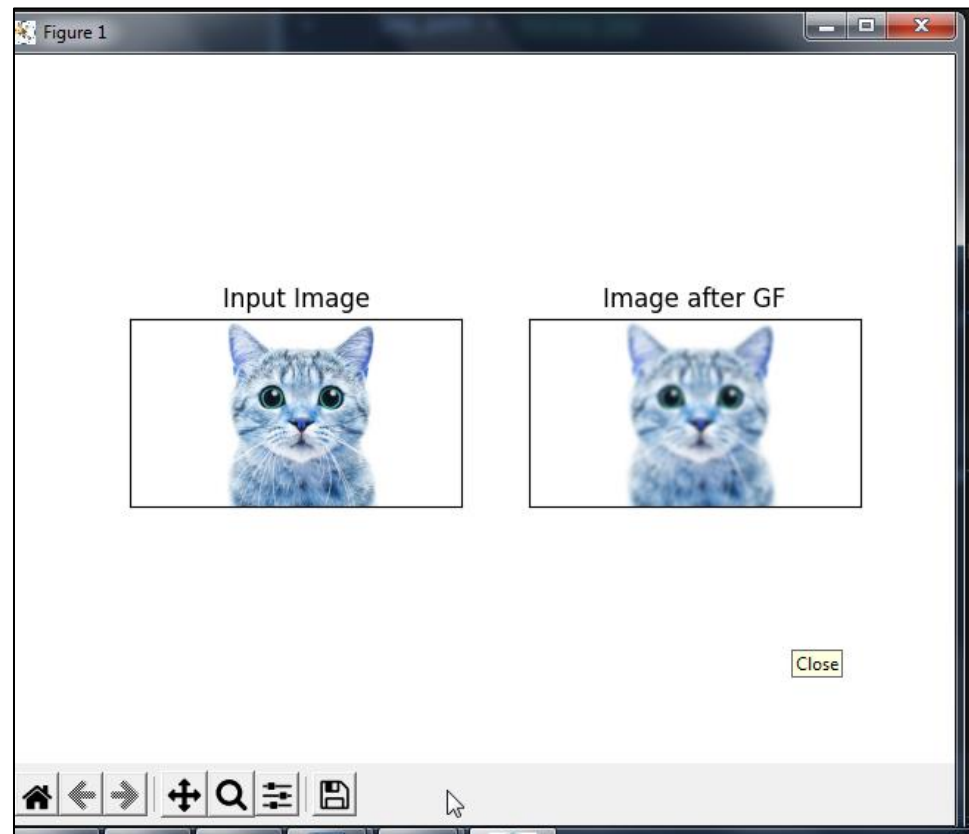
```

e. Kemudian hasil program tersebut akan tampil seperti berikut :

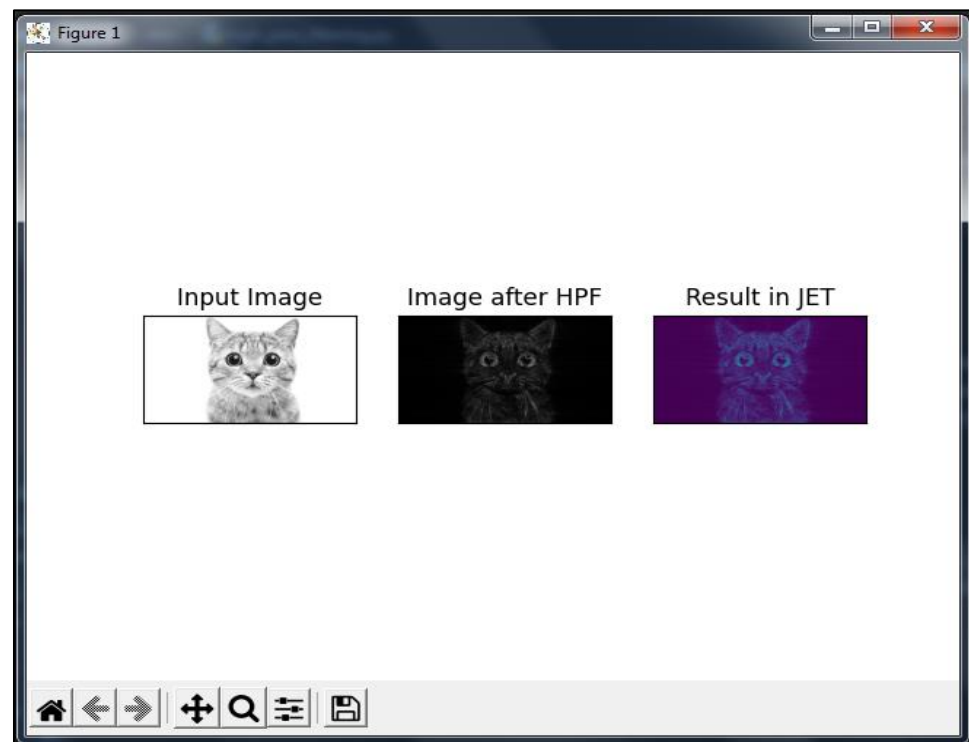
- Fourier\_transform.py



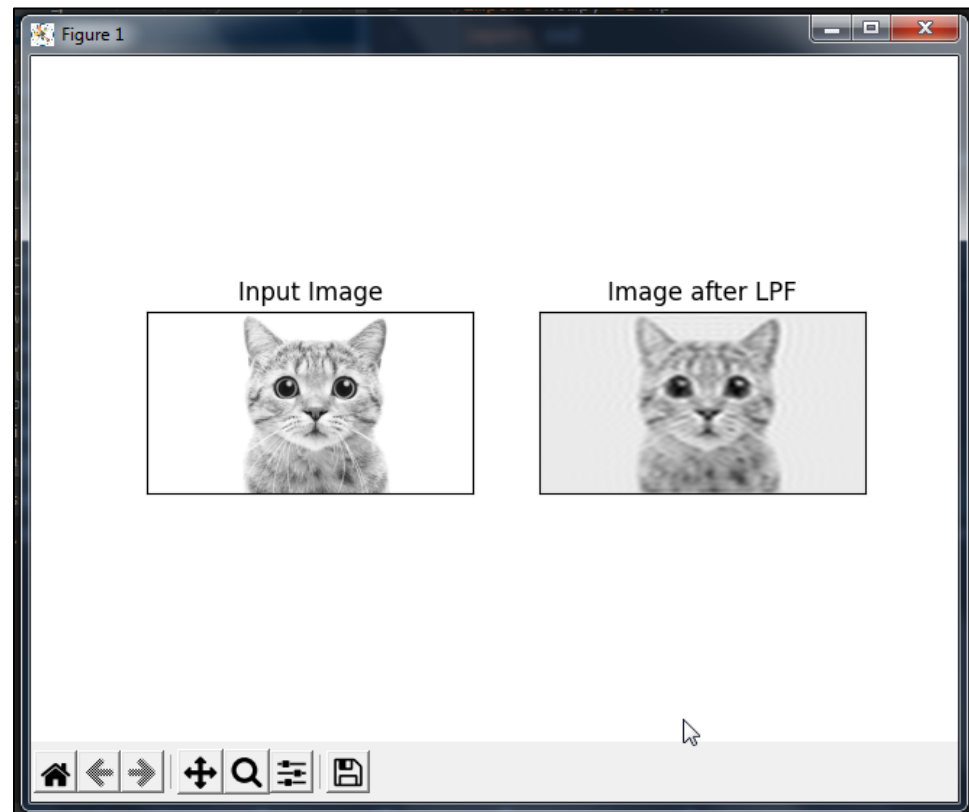
- Gaussian\_filtering.py



- High\_pass\_filtering.py



- Low\_pass\_filtering.py



#### D. ANALISIS

File pertama yang dibuat adalah fourier transform dimana ini merupakan salah satu transformasi penting pada signal processing terutama pada image processing. Kode `f = np.fft.fft2(img)` akan 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 yaitu 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_back`nya 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/mutiarasari99/pcd>