

DOMAIN FREKUENSI

PENGOLAHAN CITRA

DIGITAL



PARHAN PAUZI SALIM

F55120095

C

**PROGRAM STUDI S1 TEKNIK
INFORMATIKA JURURSAN TEKNOLOGI
INFORMASI FAKULTAS TEKNIK
UNIVERSITAS
TADULAKO
2022**

I. TUJUAN

1. Mahasiswa mampu mempelajari domain frekuensi.
2. Mahasiswa mampu memahami domain frekuensi
3. Mahasiswa mampu mengimplementasikan domain frekuensi.

II. ALAT DAN BAHAN

1. Laptop
2. *Pycharm*
3. *Python*
4. Citra gambar

III. TEORI DASAR

Kualitas citra merupakan hal yang paling krusial dalam semua bidang yang berbasis gambar. Citra yang berkualitas tinggi akan memudahkan pembacanya dalam memahami informasi yang terkandung didalamnya, sehingga mengurangi kesalahan yang mungkin terjadi. Berbicara mengenai kualitas citra, akan merujuk pada proses pengolahan citra, dimana pengolahan citra merupakan suatu proses yang dilakukan dengan masukan berupa citra dan hasilnya juga berupa citra dengan tujuan untuk meningkatkan kualitas dari citra tersebut. Salah satu metode yang dapat digunakan dalam proses perbaikan citra adalah Super Resolusi, Super Resolusi merupakan suatu teknik yang digunakan untuk membangun citra beresolusi tinggi dari sekumpulan citra yang memiliki resolusi rendah.

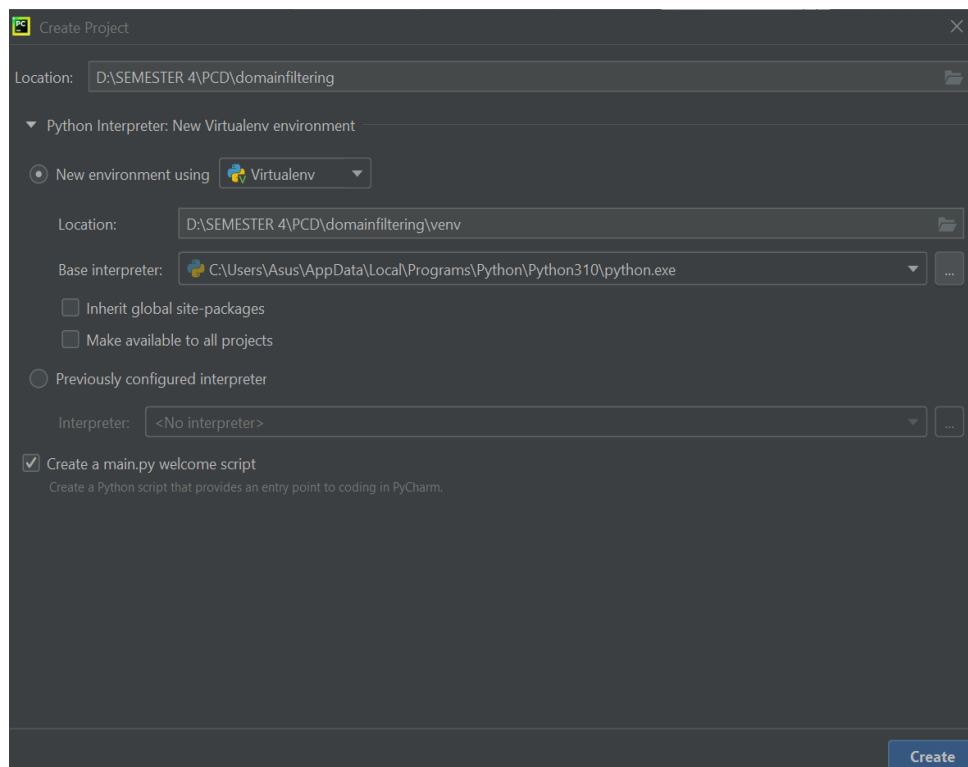
Teknik Super Resolusi mempunyai dua langkah utama, yaitu registrasi citra dan rekonstruksi citra. Registrasi merupakan langkah terpenting dalam teknik Super Resolusi karena registrasi citra menentukan baik buruknya hasil dari perbaikan citra menggunakan Super Resolusi. Registrasi citra dapat diartikan sebagai proses untuk mendapatkan nilai pergeseran diantara citra beresolusi rendah yang melibatkan dua citra atau lebih yang memiliki objek sama namun diperoleh pada waktu, sudut pandang, dan sensor yang berbeda. Dalam Tugas Akhir ini, akan diterapkan salah satu metode dari registrasi citra

yaitu metode *Power Cepstrum* yang bekerja dalam domain frekuensi. *Power Cepstrum* dapat digunakan untuk mendeteksi parameter rotasi dan translasi yang lebih efisien dan cukup akurat. Metode ini diperkenalkan pertama kali oleh *Bogert et al*, dimana awalnya metode ini digunakan untuk menganalisis data yang mengandung echoes.

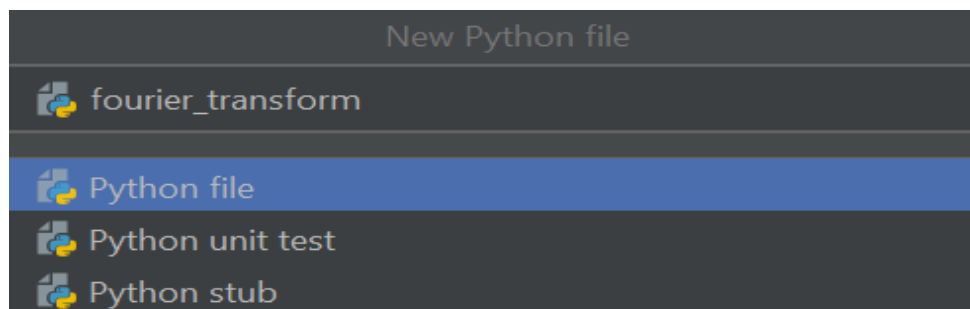
Power Cepstrum diperoleh dari logaritma *spectrum* sinyal yang kemudian ditransformasi menggunakan Transformasi *Fourier*. Metode ini mendapatkan nilai pergeseran dengan cara memproses nilai intensitas kedua citra referensi dan citra uji. Tujuan dari penelitian ini adalah membangun sebuah program yang dapat mendeteksi besar pergeseran diantara dua citra menggunakan metode *Power Cepstrum* dengan asumsi tidak ada beda rotasi diantara kedua citra. Program yang akan dibangun diharapkan dapat berkontribusi untuk menentukan pergeseran titik piksel citra dalam teknik Super Resolusi, sehingga bermanfaat dalam proses perbaikan citra.

IV. LANGKAH KERJA

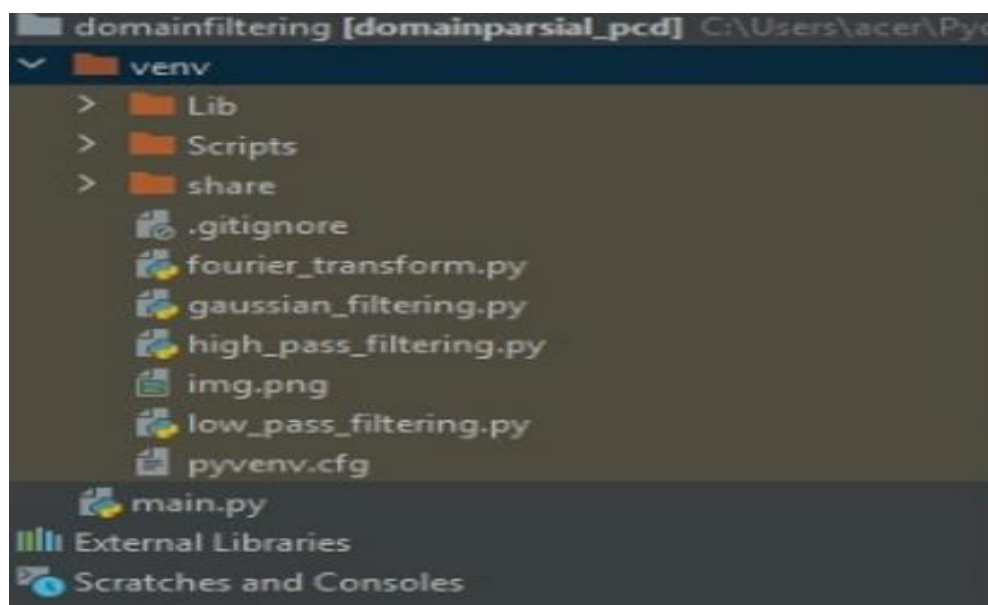
1. Membuka aplikasi *pycharm*, Kemudian membuat *project* baru.



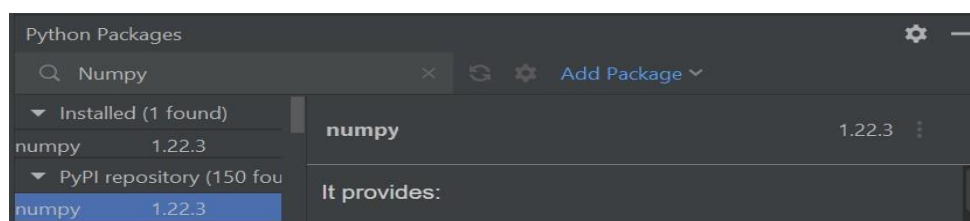
2. Membuat *python file* baru dengan nama “*fourier_transform.py*”.



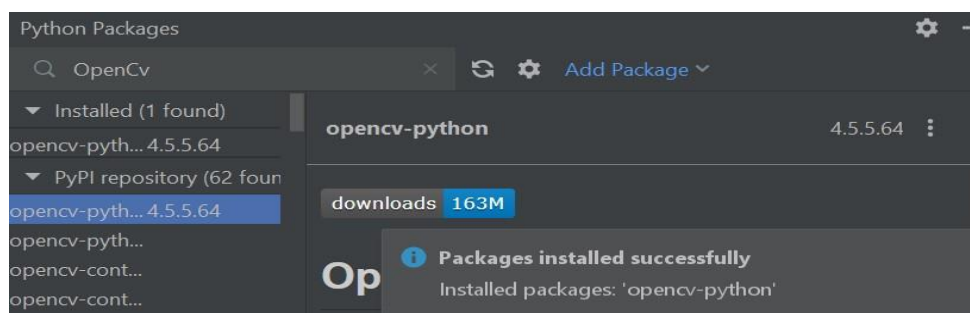
3. Menginputkan gambar ke dalam folder yang sama dengan *python file*.



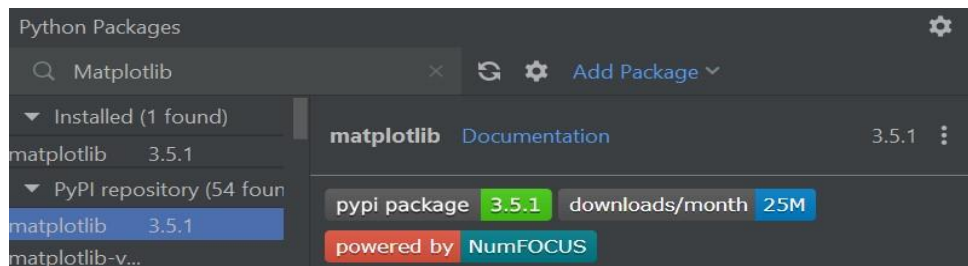
4. Menginstal *packages numpy*.



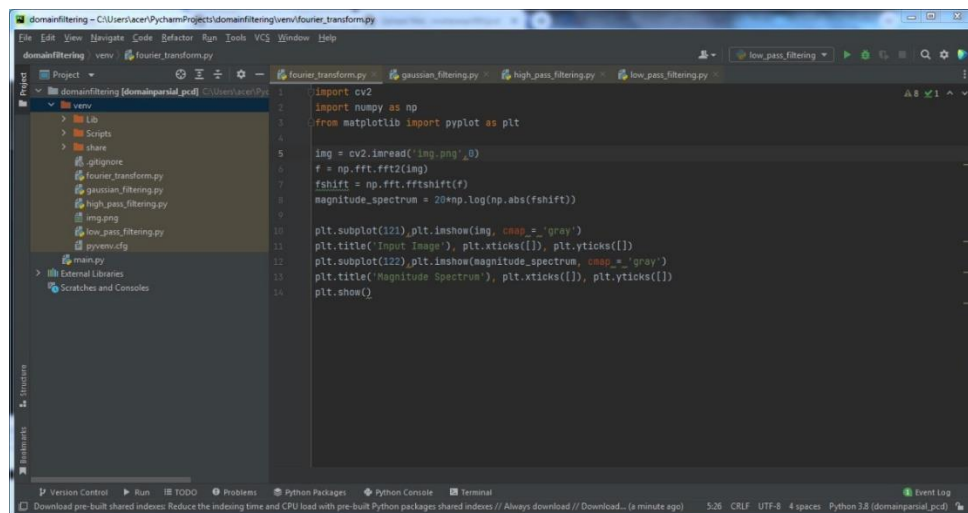
5. Menginstal *packages openCV*.



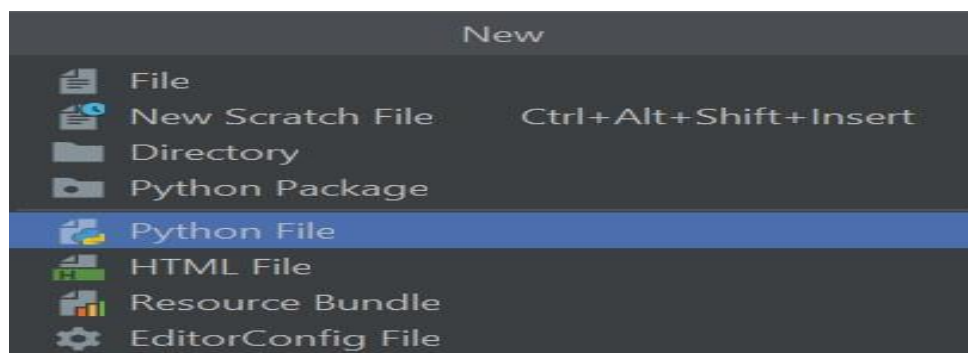
6. Menginstal *packages matplotlib*.



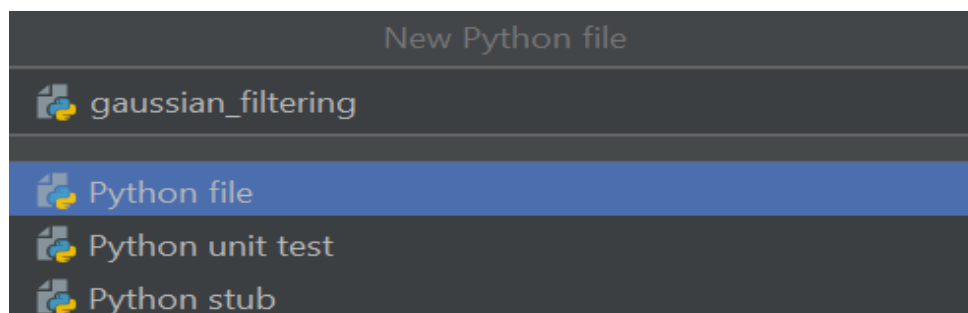
7. Menambahkan kode seperti berikut pada “*fourier_transform.py*”.



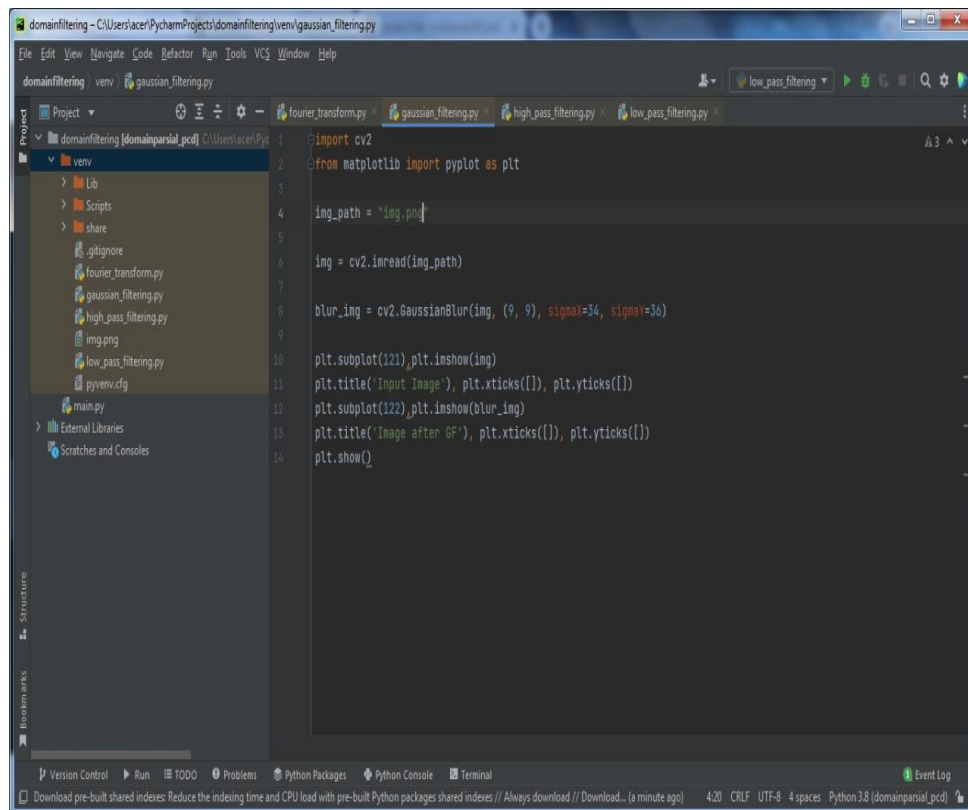
8. Membuat *file python* baru lagi dengan menekan “*file>new>python file*”.



9. Memberi nama *file* dengan nama “*gaussian_filtering.py*”

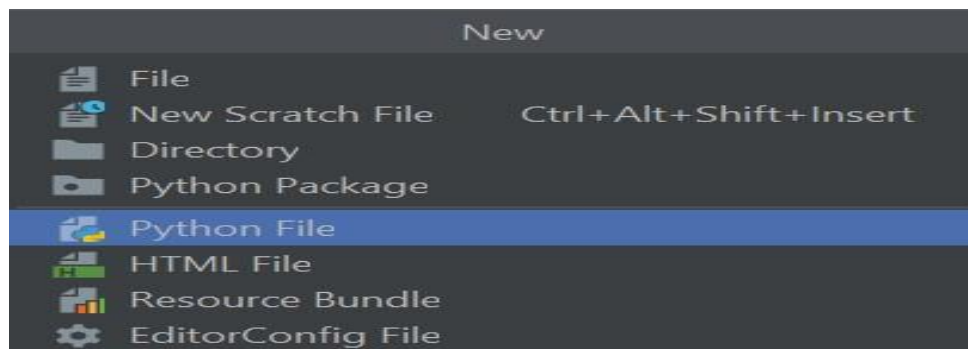


10. Menambahkan kode seperti berikut pada “*gaussian_filtering.py*”

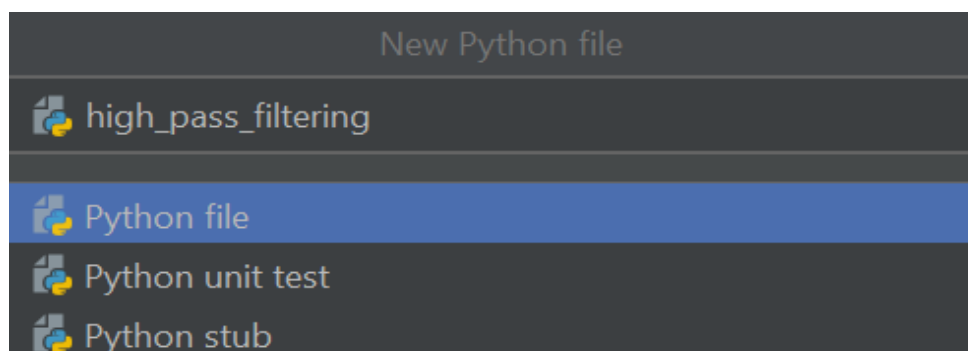


```
1 import cv2
2 from matplotlib import pyplot as plt
3
4 img_path = 'img.png'
5
6 img = cv2.imread(img_path)
7
8 blur_img = cv2.GaussianBlur(img, (9, 9), sigmaX=34, sigmaY=36)
9
10 plt.subplot(121), plt.imshow(img)
11 plt.title('Input Image'), plt.xticks([], plt.yticks([]))
12 plt.subplot(122), plt.imshow(blur_img)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

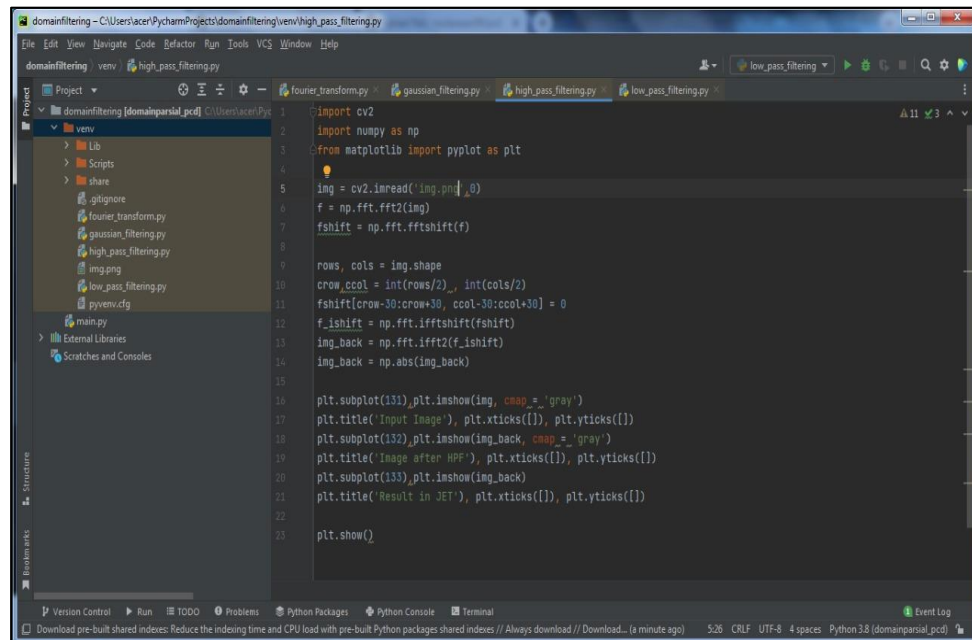
11. Membuat *file python* baru lagi dengan menekan “*file>new>python file*”.



12. Memberi nama *file* dengan nama “*high_pass_filtering.py*”

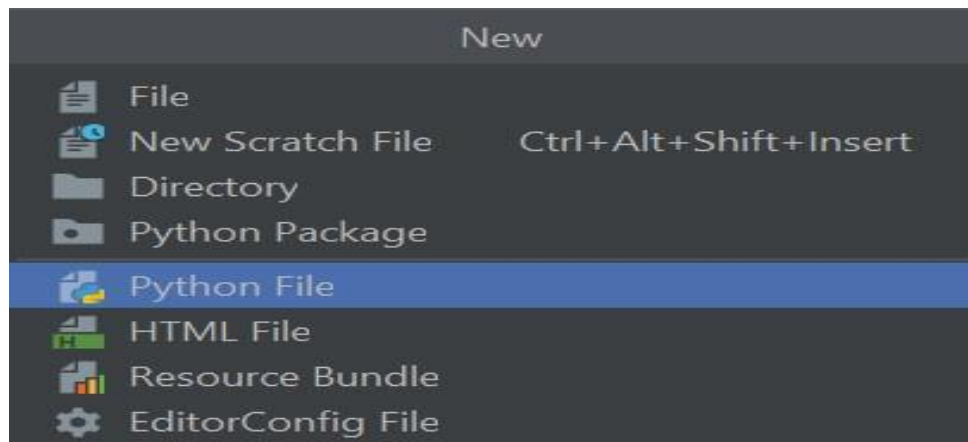


13. Menambahkan kode seperti berikut pada “*high_pass_filtering.py*”

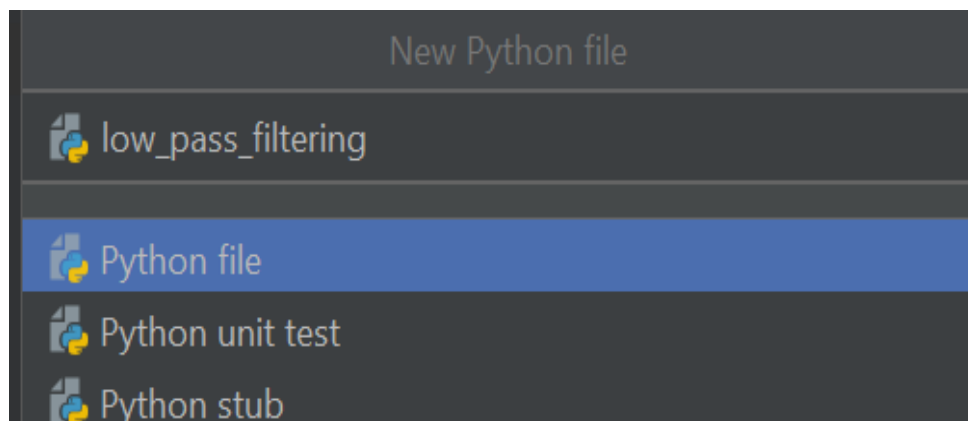


```
1 import cv2
2 import numpy as np
3 from matplotlib import pyplot as plt
4
5 img = cv2.imread('img.png', 0)
6 f = np.fft.fft2(img)
7 fshift = np.fft.fftshift(f)
8
9 rows, cols = img.shape
10 crow, ccol = int(rows/2), int(cols/2)
11 fshift[crow-30:crow+30, ccol-30:ccol+30] = 0
12 f_ishift = np.fft.ifftshift(fshift)
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()
```

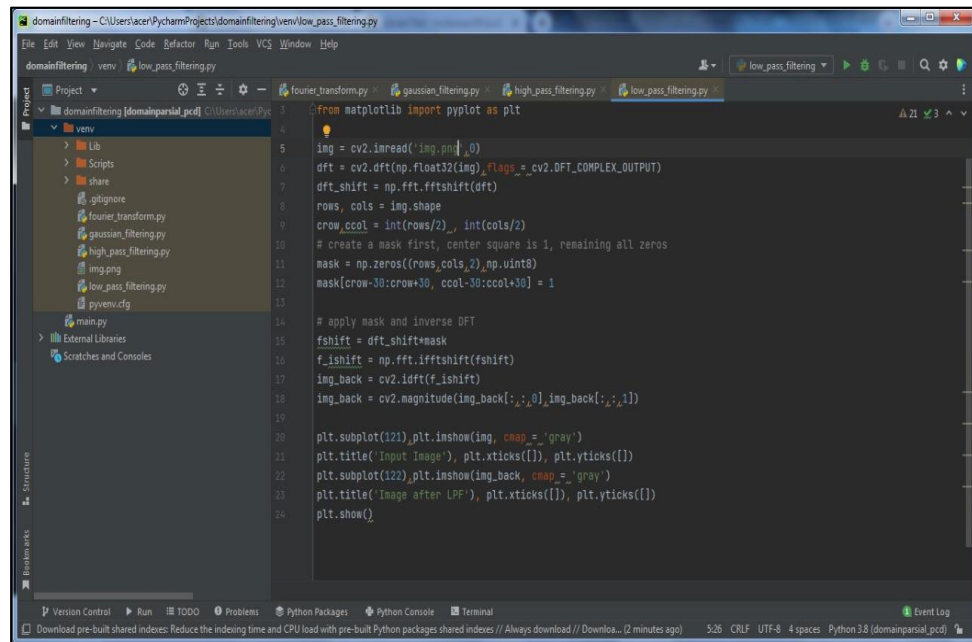
14. Membuat *file python* baru lagi dengan menekan “*file>new>python file*”



15. Memberi nama *file* dengan nama “*low_pass_filtering.py*”.



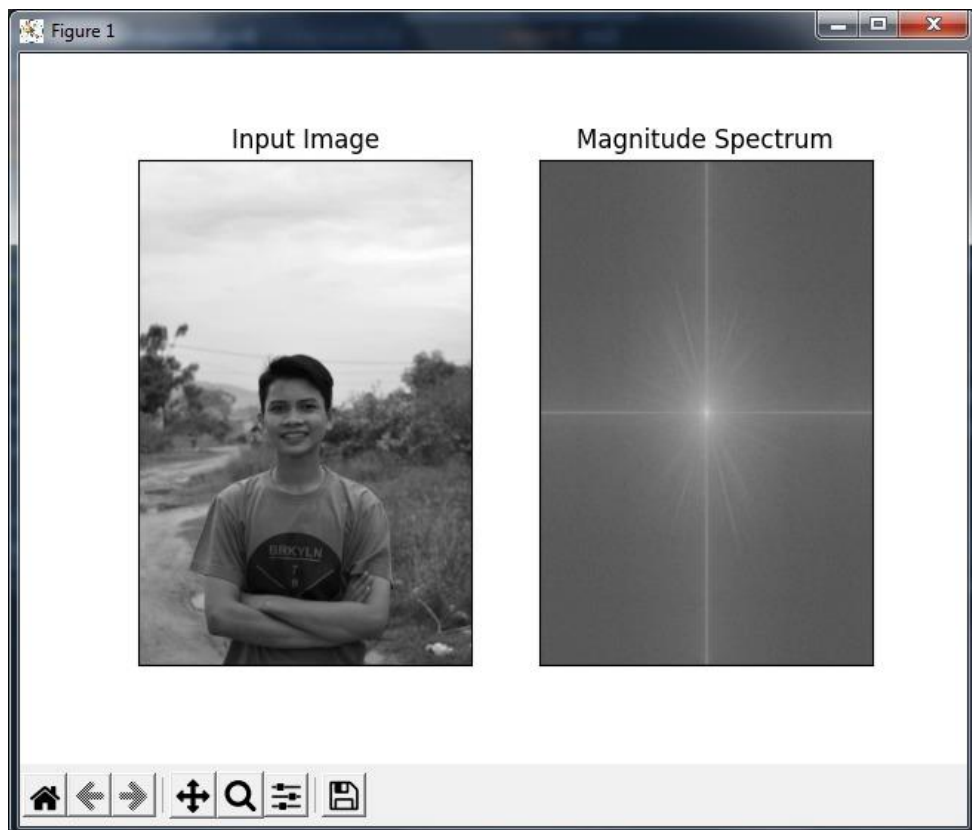
16. Menambahkan kode seperti berikut pada “*low_pass_filtering.py*”.



```
1 from matplotlib import pyplot as plt
2
3 img = cv2.imread('img.png', 0)
4
5 dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
6 dft_shift = np.fft.fftshift(dft)
7 rows, cols = img.shape
8 crow, ccol = int(rows/2), int(cols/2)
9
10 # create a mask first, center square is 1, remaining all zeros
11 mask = np.zeros((rows,cols,2),np.uint8)
12 mask[crow-crow*30:crow+crow*30, ccol-ccol*30:ccol+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()
```

V. HASIL PERCOBAAN

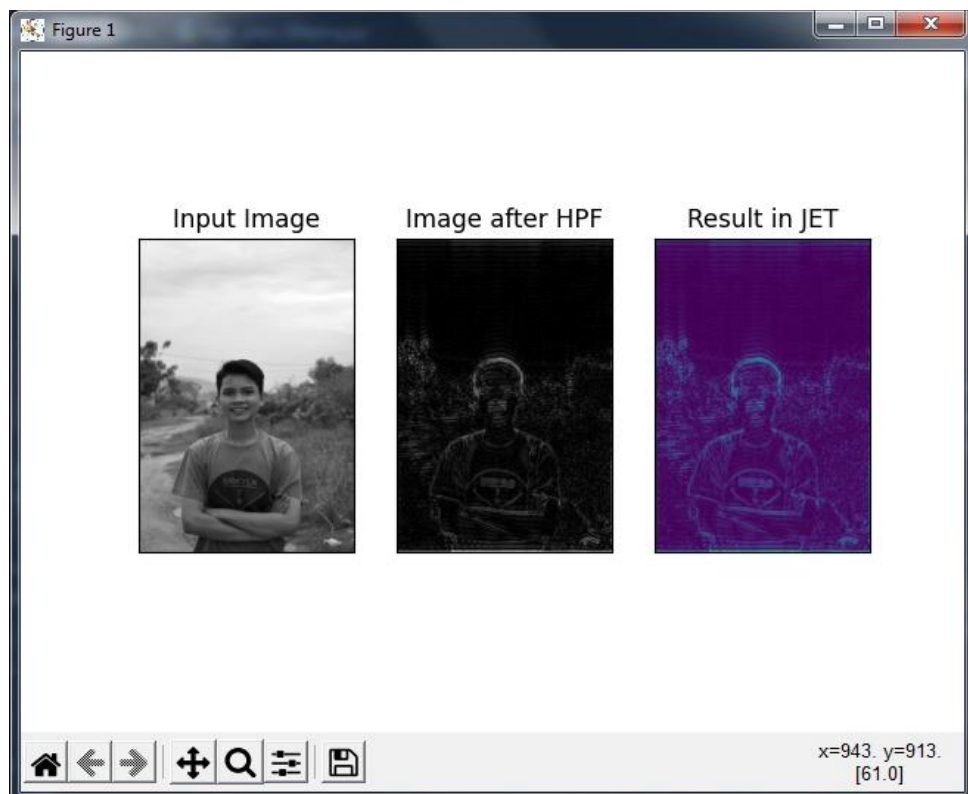
1. Hasil *Fourier Transform*



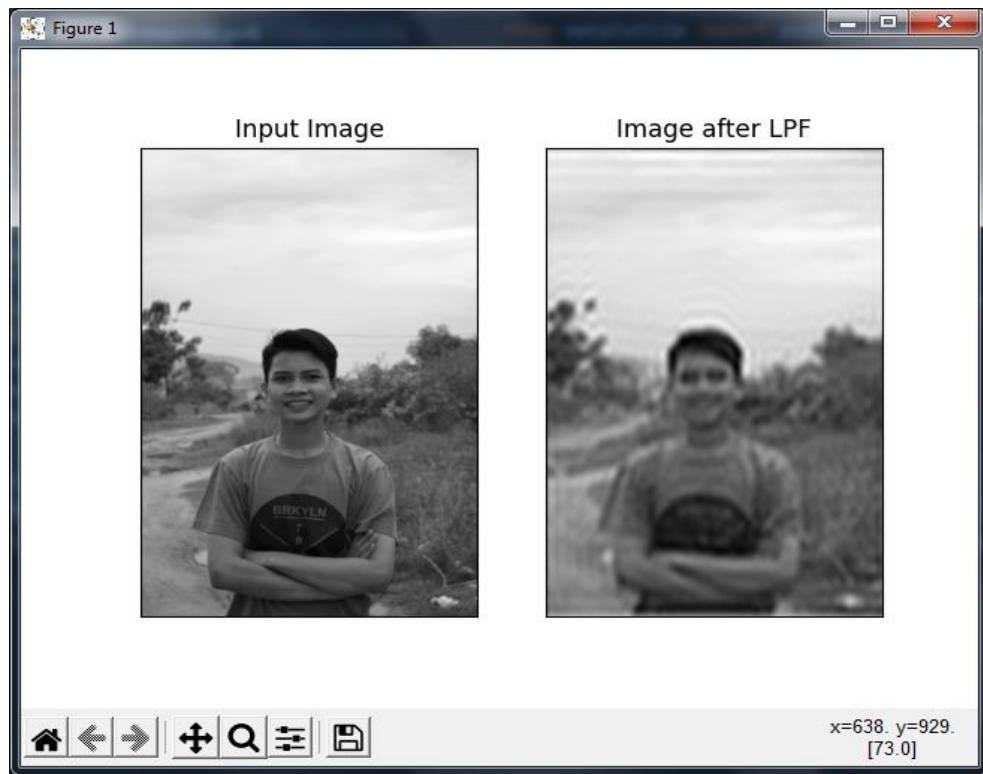
2. Hasil *Gaussian Filtering*



3. Hasil *High Pass Filtering*



4. Hasil *Low Pass*



VI. ANALISIS

pada percobaan diatas, dapat dikeahui bahwa frekuensi adalah jumlah latihan dalam periode tertentu. Frekuensi adalah ulangan gerak berapa kali atlet harus melakukan gerak setiap giliran. Frekuensi tinggi berarti ulangan gerak banyak sekali dalam satu giliran, sedang frekuensi rendah artinya ulangan gerak sedikit dalam satu giliran. Sebelum melakukan programan, kita harus menginstall library yang di gunakan, yaitu numpy, cv2, matplotlib. NumPy merupakan salah satu library Python yang berfungsi untuk proses komputasi numerik. NumPy memiliki kemampuan untuk membuat objek N-dimensi array. Array merupakan sekumpulan variabel yang memiliki tipe data yang sama. Cv2 adalah fungsi pengikat keyboard. Maksudnya adalah waktu dalam milidetik. Jika Kita menekan sembarang tombol di keyboard, program akan terus berjalan. Jika 0 dilewati, ia menunggu tanpa batas waktu untuk terkena stroke kunci. Hal ini juga dapat diatur untuk mendeteksi menekan tombol tertentu seperti, jika tombol *a* ditekan. Cv2 adalah fungsi untuk

menghancurkan atau menutup jendela. Jika Kita ingin menutup jendela tertentu. Fungsi dari kode `"img=cv2.imread('...')"` berfungsi untuk membaca gambar. `"plt.subplot"` digunakan untuk membuat multiple plots yang memiliki weight dan height yang sama. Sedangkan matplotlib sendiri adalah suatu library atau package yang paling populer di bahasa python untuk melakukan visualisasi data seperti membuat plot grafik untuk satu sumbu atau lebih. Setiap sumbu memiliki sumbu horizontal (x) dan sumbu vertikal (y).

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`.

VII. LINK GITHUB

<https://github.com/ParhanSalim095/Dfrekuensi.git>