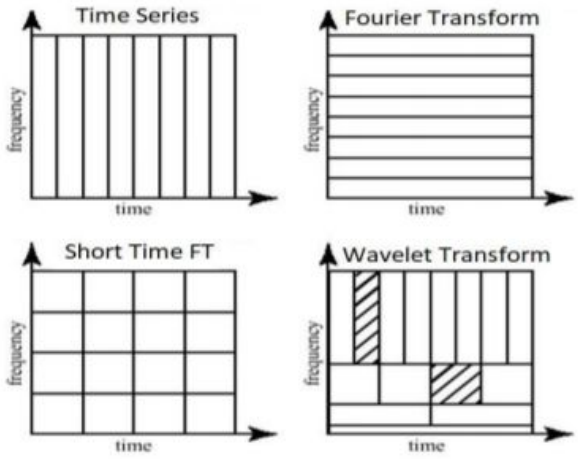
TRANSFORMASI FOURIER DAN WAVELET

Gambar dibawah ini menjelaskan perbedaan antara transformasi Fourier dan Wavelet.



a. Jelaskan perbedaan Fourier dan Wavelet berdasarkan gambar tersebut ?

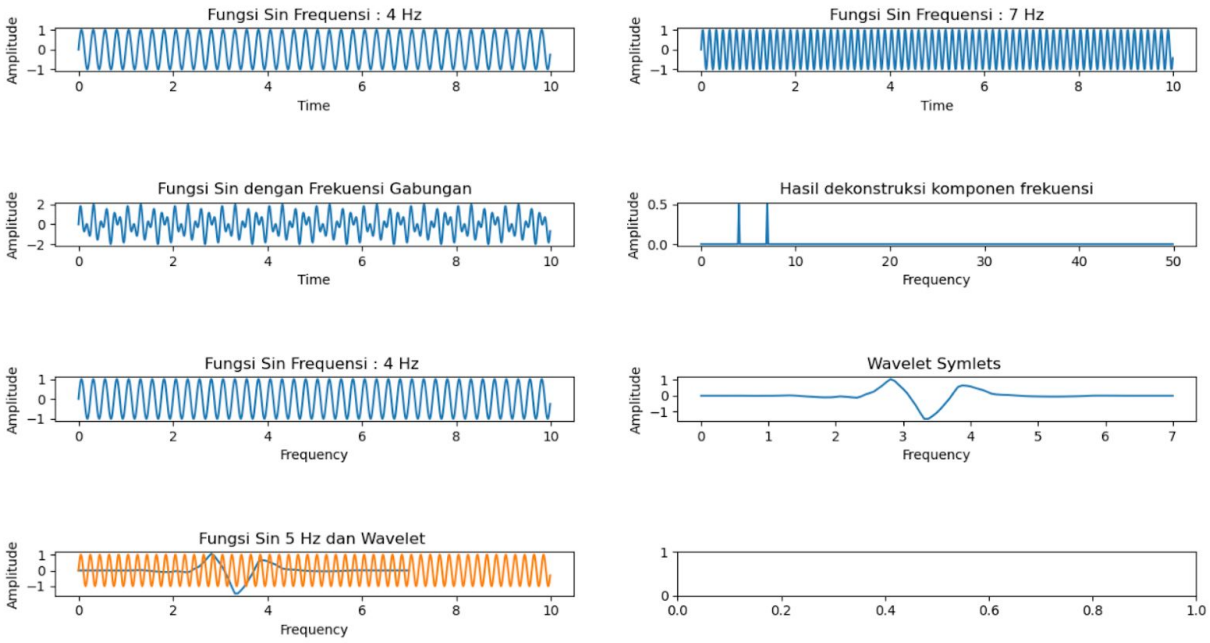
   Gambar di atas merupakan grafik waktu terhadap frekuensi. Kotak-kotak yang terdapat dalam grafik mengindikasikan seberapa besar suatu fitur bisa kita bedakan antara frekuensi dan waktu nya. Misalnya pada grafik pertama (Time Series) , kita bisa membedakan domain-domain waktu tetapi kita tidak memperoleh informasi tentang frekuensi yang dimiliki fitur tersebut sehingga yang dapat dibagi kotak per kotak hanyalah satuan waktu. Pada gambar Fourier Transform, karakteristik yang terjadi berkebalikan dengan Time Series. Gambar fourier transform dapat menjelaskan frekuensi yang menjadi konstruksi suatu fitur tetapi kita tidak memperoleh informasi tentang kapan terjadinya (waktu) dari frekuensi tersebut sehingga yang dapat dibagi hanyalah domain frekuensi. Selain itu pada gambar Short Time FT (Fourier Transform) , Fungsi tersebut bisa membagi domain waktu dan domain frekuensi secara bersamaan. Hal ini karena Short Time FT merupakan bentuk fourier transform yang memiliki batasan saat pengaplikasiannya. Fungsi asli akan dipecah per segmen lalu dianalisis satu persatu oleh Short Time FT. Namun hal tersebut menyebabkan lebar window (segmen) haruslah tetap. Wavelet Transform mampu mengatasi beberapa masalah sebelumnya. Pada wavelet, fitur yang memiliki ketergantungan terhadap waktu akan mempunyai resolusi yang tinggi pada domain waktu dan fitur yang memiliki ketergantungan frekuensi akan mempunyai resolusi tinggi pada domain frekuensinya.

b. Tulis kode program yang sederhana menggunakan python untuk menjelaskan perbedaan tersebut.

```
import pywt
```

```
import numpy as np
import matplotlib.pyplot as plt

freq1 = 4
freq2 = 7
time = np.arange(0, 10, 0.01)

func1 = np.sin(2*np.pi*freq1*time)
func2 = np.sin(2*np.pi*freq2*time)

figure, axis = plt.subplots(4, 2)
plt.subplots_adjust(hspace=3)

axis[0, 0].set_title('Fungsi Sin Frequensi : 4 Hz')
axis[0, 0].plot(time, func1)
axis[0, 0].set_xlabel('Time')
axis[0, 0].set_ylabel('Amplitude')

axis[0, 1].set_title('Fungsi Sin Frequensi : 7 Hz')
axis[0, 1].plot(time, func2)
axis[0, 1].set_xlabel('Time')
axis[0, 1].set_ylabel('Amplitude')

func_merge = func1 + func2

axis[1, 0].set_title('Fungsi Sin dengan Frekuensi Gabungan')
axis[1, 0].plot(time, func_merge)
axis[1, 0].set_xlabel('Time')
axis[1, 0].set_ylabel('Amplitude')

fourierTransform = np.fft.fft(func_merge) / len(func_merge)
fourierTransform = fourierTransform[range(len(func_merge) // 2)]
tpCount = len(func_merge)
values = np.arange(tpCount // 2)
timePeriod = tpCount / 100
frequencies = values / timePeriod

axis[1, 1].set_title('Hasil dekonstruksi komponen frekuensi')
axis[1, 1].plot(frequencies, abs(fourierTransform))
axis[1, 1].set_xlabel('Frequency')
axis[1, 1].set_ylabel('Amplitude')
```

```
axis[2, 0].set_title('Fungsi Sin Frequensi : 4 Hz ')
axis[2, 0].plot(time, func1)
axis[2, 0].set_xlabel('Frequency')
axis[2, 0].set_ylabel('Amplitude')

[_, psi, xx] = pywt.Wavelet('sym4').wavefun(level=4)

axis[2, 1].set_title('Wavelet Symlets ')
axis[2, 1].plot(xx, psi)
axis[2, 1].set_xlabel('Frequency')
axis[2, 1].set_ylabel('Amplitude')

t = np.arange(0, 10, 0.01)
y = np.sin(2 * np.pi * 5 * t)
axis[3, 0].set_title('Fungsi Sin 5 Hz dan Wavelet ')
axis[3, 0].plot(xx, psi)
axis[3, 0].plot(t, y)
axis[3, 0].set_xlabel('Frequency')
axis[3, 0].set_ylabel('Amplitude')

plt.show()
```



Fourier mampu mendekonstruksi gabungan frekuensi kedalam komponen satuannya. Sedangkan wavelet melakukan dekonstruksi perbagian segmen.

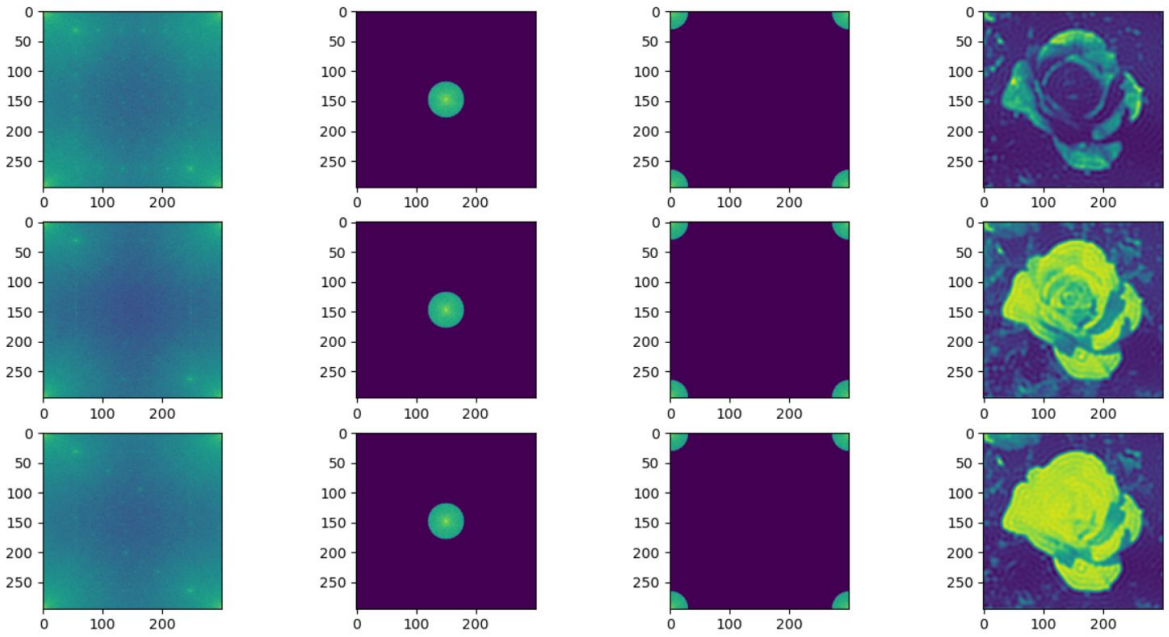
c. Buat program dan bandingkan kinerja kedua metode transformasi tersebut pada kasus menghilangkan noise pada gambar berikut (file image terlampir). Mahasiswa dapat menggunakan fungsi-fungsi pada library opencv.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from math import exp, sqrt
from skimage import img_as_float
from skimage.util import random_noise
from skimage.restoration import denoise_wavelet, estimate_sigma


def distance(x, y):
    return sqrt((x[0]-y[0])**2 + (x[1]-y[1])**2)


def gaussian_low_pass(size):
    rows, cols, *_ = size
    mask = np.zeros(size, dtype=np.uint8)
    center = (rows // 2, cols // 2)
    for x in range(cols):
        for y in range(rows):
            mask[y, x] = exp(((-distance((y, x),
center)**2)/(2*(50**2))))
    return mask


def generate_mask(size):
    row, col, *_ = size
    crow, ccol = row // 2, col // 2
    mask = np.zeros(size, dtype=np.uint8)
    r = 30
    x, y = np.ogrid[:row, :col]
    filter_area = (x - crow) ** 2 + (y - ccol) ** 2 <= r ** 2
    mask[filter_area] = 1
    return mask
```

```
def fourier_transform(image):
    image_fft = np.fft.fft2(image.astype("float32"))
    image_fshifted = np.fft.fftshift(image_fft)
    mask = generate_mask(image.shape)
    image_fshifted = image_fshifted * mask
    image_ishifted = np.fft.ifftshift(image_fshifted)
    image_back = np.fft.ifft2(image_ishifted)

    result = [np.log(1+np.abs(image_fft)),
np.log(1+np.abs(image_fshifted)),
              np.log(1+np.abs(image_ishifted)), np.abs(image_back)]
    return result


def do_fourier(image):
    image_r = fourier_transform(image[:, :, 0])
    image_g = fourier_transform(image[:, :, 1])
    image_b = fourier_transform(image[:, :, 2])

    plt.subplot(3, 4, 1), plt.imshow(image_r[0])
    plt.subplot(3, 4, 5), plt.imshow(image_g[0])
    plt.subplot(3, 4, 9), plt.imshow(image_b[0])

    plt.subplot(3, 4, 2), plt.imshow(image_r[1])
    plt.subplot(3, 4, 6), plt.imshow(image_g[1])
    plt.subplot(3, 4, 10), plt.imshow(image_b[1])

    plt.subplot(3, 4, 3), plt.imshow(image_r[2])
    plt.subplot(3, 4, 7), plt.imshow(image_g[2])
    plt.subplot(3, 4, 11), plt.imshow(image_b[2])

    plt.subplot(3, 4, 4), plt.imshow(image_r[3])
    plt.subplot(3, 4, 8), plt.imshow(image_g[3])
    plt.subplot(3, 4, 12), plt.imshow(image_b[3])

    plt.show()

    image[:, :, 0] = image_r[3]
    image[:, :, 1] = image_g[3]
    image[:, :, 2] = image_g[3]
```

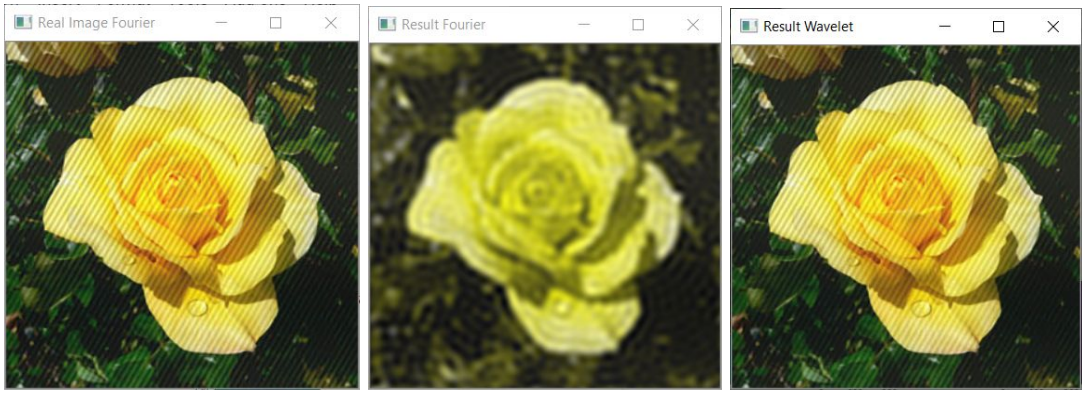
```
    return image


def do_wavelet(image):
    image = img_as_float(image)
    sigma_est = estimate_sigma(image, multichannel=True,
average_sigmas=True)
    result = denoise_wavelet(image, multichannel=True,
convert2ycbcr=True,
                             method='VisuShrink', mode='soft',
                             sigma=sigma_est/4, rescale_sigma=True)
    return result


if __name__ == "__main__":
    plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

    image = cv2.imread("image-soalno-3.jpg")

    result_fourier = do_fourier(image.copy())
    result_wavelet = do_wavelet(image.copy())

    cv2.imshow("Real Image Fourier", image)
    cv2.imshow("Result Fourier", result_fourier)
    cv2.imshow("Result Wavelet", result_wavelet)

    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Ruang Fourier pada 3 Channel :



Pada gambar diatas diimplementasikan metode Low Pass Filter

Hasil dari 3 metode :



Hasil yang didapat pada fourier menunjukan pengurangan dari noise, tetapi color space sedikit terganggu karena pengaruh konversi nilai unsigned int 8 bit ke float untuk perhitungan transformasinya. Sedangkan pada wavelet tidak terlihat perubahan yang signifikan.