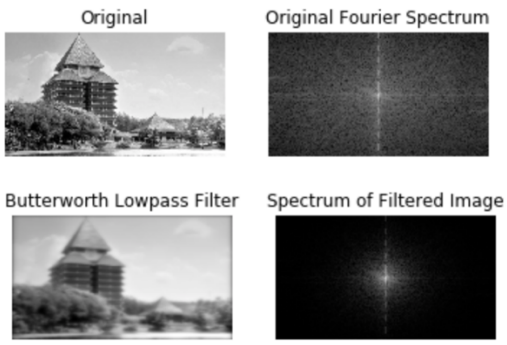
Tugas 2
Pengolahan Citra A
Muhammad Irfan Amrullah / 1706039585

1) A. dan B.

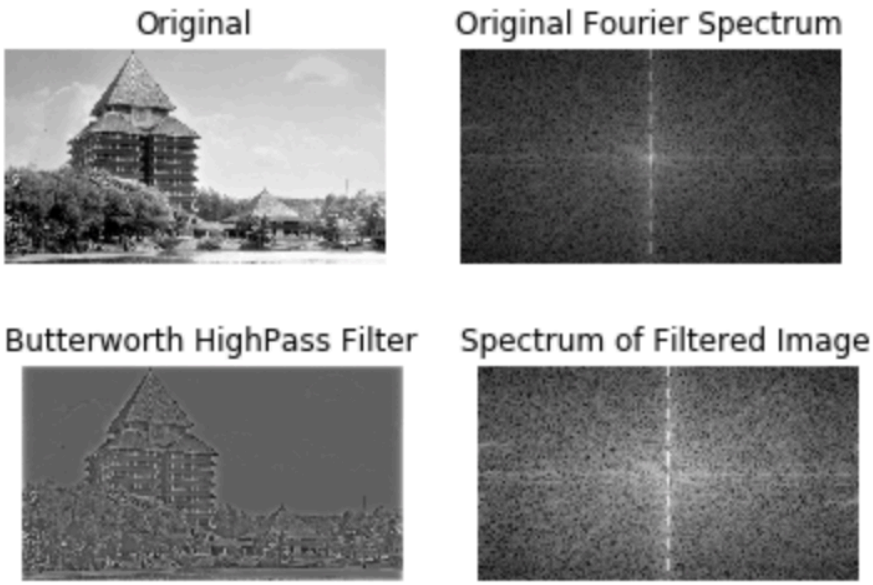
In [7]:
```
i1 = color.rgb2gray(io.imread('rektorat_ui.jpeg'))
w, l = paddedsize(i1.shape[0], i1.shape[1])
# membuat gaussian lowpass filter
# fungsi lpfilter terdapat di file helper.py
h = lpfilter('btw', w, l, 0.05 * w)
# menghitung DFT citra
f = fp.fft2(i1,(w,l))
# Apply lowpass filter
LPFS_lena = h*f
# convert ke domain spasial
LPF_lena = fp.ifft2(LPFS_lena).real
LPF_lena = LPF_lena[:i1.shape[0],:i1.shape[1]]

# Menampilkan fourier spectrum
Fc = fp.fftshift(f)
Fcf = fp.fftshift(LPFS_lena)
# fungsi abs untuk menghitung magnitude
S1 = np.log(1+abs(Fc))
S2 = np.log(1+abs(Fcf))
plt.subplot(1,2,1); plt.imshow(i1, cmap='gray')
plt.title("Original"); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(S1, cmap='gray')
plt.title("Original Fourier Spectrum"); plt.axis("off")
plt.show()

plt.subplot(1,2,1); plt.imshow(LPF_lena, cmap='gray')
plt.title("Butterworth Lowpass Filter"); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(S2, cmap='gray')
plt.title("Spectrum of Filtered Image"); plt.axis("off")
plt.show()

#Semakin tinggi nilai D0 semakin turun tingkat smoothnya
```


```
# Menampilkan fourier spectrum
Fc = fp.fftshift(f)
Fcf = fp.fftshift(LPFS_lena)
# fungsi abs untuk menghitung magnitude
S1 = np.log(1+abs(Fc))
S2 = np.log(1+abs(Fcf))
plt.subplot(1,2,1); plt.imshow(i1, cmap='gray')
plt.title("Original"); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(S1, cmap='gray')
plt.title("Original Fourier Spectrum"); plt.axis("off")
plt.show()

plt.subplot(1,2,1); plt.imshow(LPF_lena, cmap='gray')
plt.title("Butterworth Lowpass Filter"); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(S2, cmap='gray')
plt.title("Spectrum of Filtered Image"); plt.axis("off")
plt.show()

#Semakin tinggi nilai D0 semakin turun tingkat smoothnya
```

C.

In [8]:
```
i1 = color.rgb2gray(io.imread('rektorat_ui.jpeg'))
w, l = paddedsize(i1.shape[0], i1.shape[1])
# membuat gaussian lowpass filter
# fungsi lpfilter terdapat di file helper.py
h = hpfilter('btw', w, l, 0.05 * w)
# menghitung DFT citra
f = fp.fft2(i1,(w,l))
# Apply lowpass filter
LPFS_lena = h*f
# convert ke domain spasial
LPF_lena = fp.ifft2(LPFS_lena).real
LPF_lena = LPF_lena[:i1.shape[0],:i1.shape[1]]

# Menampilkan fourier spectrum
Fc = fp.fftshift(f)
Fcf = fp.fftshift(LPFS_lena)
# fungsi abs untuk menghitung magnitude
S1 = np.log(1+abs(Fc))
S2 = np.log(1+abs(Fcf))
plt.subplot(1,2,1); plt.imshow(i1, cmap='gray')
plt.title("Original"); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(S1, cmap='gray')
plt.title("Original Fourier Spectrum"); plt.axis("off")
plt.show()

plt.subplot(1,2,1); plt.imshow(LPF_lena, cmap='gray')
plt.title("Butterworth HighPass Filter"); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(S2, cmap='gray')
plt.title("Spectrum of Filtered Image"); plt.axis("off")
plt.show()

#Semakin tinggi nilai D0 semakin tajam gambarnya
```



D. D0 pada lowpass semaking tinggi nilai D0, semakin turun tingkat smoothnya. D0 pada highpass semakin tinggi nilai d0 semakin tajam gambarnya.

E.

Pada proses pengerjaannya, keduanya hampir sama. Pertama-tama citra dibuat grayscale dan dihitung DFTnya dengan menjalankan method fft2(). Kemudian pada citra

tersebut diterapkan *filter* yang ingin dipakai (BLPF atau BHPF) dengan cara mengkalikan DFT nya dengan fungsi filtering tersebut. Kemudian citra yang *filtered* tersebut diconvert ke domain spasial dengan method ifft2().

Butterworth high pass filter mempertahankan frequency diluar radius D0 dan membuang yang dibawah radius D0. Dia memiliki transisi gradual dari 0 sampai 1 untuk mengurangi ringing artifacts. Kesimpulannya gambar menjadi tajam karena yang blur di cut.

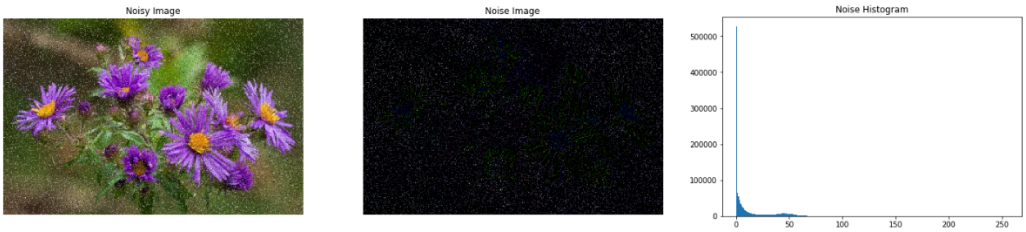
Sedangkan low pass filter mengcut semua frequency diluar d0. Kesimpulannya gambar menjadi lebih smooth karena yang tajam di cut.

Gambar ini

2) A.

In [32]:
```
img2_noisy=io.imread('flowers_noisy.jpg')
img2=io.imread('flowers.jpg')
noise= np.subtract(img2_noisy, img2, dtype='int16')

noise_hist= util.img_as_ubyte(noise)

plt.figure(figsize=(25,5))
plt.subplot(131), plt.imshow(img2_noisy),plt.title('Noisy Image'),plt.a
plt.subplot(132), plt.imshow(noise),plt.title('Noise Image'),plt.axis('
plt.subplot(133), plt.hist(noise_hist.flatten(), 256, range=(0,256)),pl
plt.show()
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1
] for floats or [0..255] for integers).
```



Dari histogram yang kita dapatkan apabila kita samakan dengan Histogram noise, maka akan didapatkan bahwa jenis noise tersebut adalah Impulse Noise. Yaitu spesifiknya adalah salt and paper.

Jenis filter yang saya gunakan adalah btw

In [13]:
```
from skimage import io, color, util
import cv2

bunga = io.imread('flowers_noisy.jpg')

blur = cv2.medianBlur(bunga, 5)
plt.subplot(); plt.imshow(blur)
plt.title("After median filtered"); plt.axis("off")
plt.show();
```

B.

In [33]:
```
import numpy
import math
import cv2
original = bunga
contrast = blur

def psnr(img1, img2):
    mse = numpy.mean( (img1 - img2) ** 2 )
    if mse == 0:
        return 100
    PIXEL_MAX = 255.0
    return 20 * math.log10(PIXEL_MAX / math.sqrt(mse))

d=psnr(original,contrast)
print('PSNR:' + str(d))
print('MSE:' + str(numpy.mean( (original - contrast) ** 2 )))
```

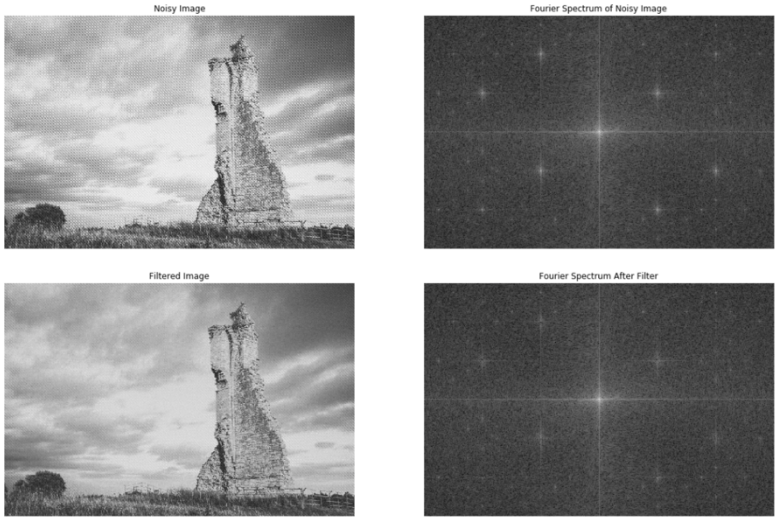
```
PSNR:30.505755988586888
MSE:57.87683344694388
```

C. Untuk filter saya menggunakan median filtering yang teknisnya adalah merata2 data yang hilang dan mengisinya dengan pixel disekitarnya. Dari nilai PSNR didapatkan 30.5. PSNR digunakan untuk mengukur kualitas antar gambar original dan gambar yang sudah di kompres. Semakin tinggi psnrnya semakin bagus kualitas gambar yang di kompres atau yang di restorasi. Di soal ini karena hanya 30 berarti bisa dibilang psnr gambar saya masih kurang berkualitas.

Nilai MSE digunakan untuk menghitung kumulatif error dari gambar yang di compress dan gambar original. Semakin rendah, semakin rendah pula errornya. Pada gambar ini error yang saya dapatkan adalah 57.

3)

In [37]:
```
football = io.imread('ruin.jpg')
w, l = paddedsize(football.shape[0],football.shape[1])
F = fp.fft2(util.img_as_float(football),(w,l))
Fc = fp.fftshift(F)
S1 = np.log(1+abs(Fc))

plt.figure(figsize=(20,8))
plt.subplot(121); plt.imshow(football, cmap='gray')
plt.title('Noisy Image'); plt.axis("off")
plt.subplot(122); plt.imshow(S1, cmap='gray')
plt.title('Fourier Spectrum of Noisy Image'); plt.axis("off")
plt.show()

# fungsi notch terdapat pada file helper.py
H1 = notch('btw', w, l, 10, 249, 330)
H2 = notch('btw', w, l, 10, 500, 168)
H3 = notch('btw', w, l, 10, 249, -168)
H4 = notch('btw', w, l, 10, 500, -330)
H5 = notch('btw', w, l, 10, -249, 170)
H6 = notch('btw', w, l, 10, -500, 330)
H7 = notch('btw', w, l, 10, -249, -330)
H8 = notch('btw', w, l, 10, -500, -168)
FS_football = F*H1*H2*H3*H4*H5*H6*H7*H8
F_football = fp.ifft2(FS_football).real
F_football = F_football[:football.shape[0],:football.shape[1]]
Fcf = fp.fftshift(FS_football)
S2 = np.log(1+abs(Fcf))

plt.figure(figsize=(20,8))
plt.subplot(121); plt.imshow(F_football, cmap='gray')
plt.title('Filtered Image'); plt.axis("off")
plt.subplot(122); plt.imshow(S2, cmap='gray')
plt.title('Fourier Spectrum After Filter'); plt.axis("off")
plt.show()
```



Kita menggunakan 1 kali pemrosesan yaitu notch filter. Kita menghapus satu2 noise yang terlihat pada fourier spectrum dengan memasukkan koordinat satu persatu.

Untuk logikanya kita pertama menghitung dft citra. Lalu kita tampilkan koefisien frekuensi-nol ditengah. Hitung magnitude untuk menampilkan fourier. Pakai notch untuk menghilangkan titik-titik sesuai koordinat. Convert ke domain spasial, tampilkan frekuensi 0 ditengah untuk menampilkan fourier spectrum, hitung abs.nanti hasilnya akan berkurang noisenya

B. Notch filter digunakan untuk menghilangkan noise “Spectral” yang berulang2 pada gambar. Notch itu seperti highpass filter yang kecil tapi fungsinya menaikkan frequency

selain komponen dc. Mengurangi frequency terpilih dan meninggalkan frekuensi lain di fourier transform tidak berubah. Ini sangat berguna apabila foto kita memiliki noise yang berulang.