Image Processing with Fourier Transform

Finite and Discrete Input and Output

The *Discrete Fourier Transform* transforms a sequence of N complex numbers
$\{x_n\} = x0, x1, \ldots . x_{n-1}$ into another sequence of complex numbers
$\{X_k\} = X_0, X_1, \ldots . X_{N-1}$, which is defined by

One dimension

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N} kn}$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{i2\pi kn/N}$$

Two dimension

$$X[s,t] = \sum_n \sum_m x[n,m] e^{-j2\pi mk/M} e^{-j2\pi nk/N}$$

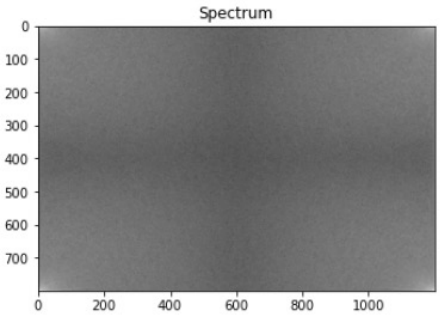
$$x[n,m] = \sum_n \sum_m X[s,t] e^{j2\pi mk/M} e^{j2\pi nk/N}$$

We're using FFT – Fast Fourier Transform (FFTs are faster ways of doing DFTs)

1. fft2 - fft.fft2(a, s=None, axes=(- 2, - 1), norm=None)

This function computes the n-dimensional discrete Fourier Transform over any axes in an M-dimensional array by means of the Fast Fourier Transform (FFT). By default, the transform is computed over the last two axes of the input array, i.e., a 2-dimensional FFT.]

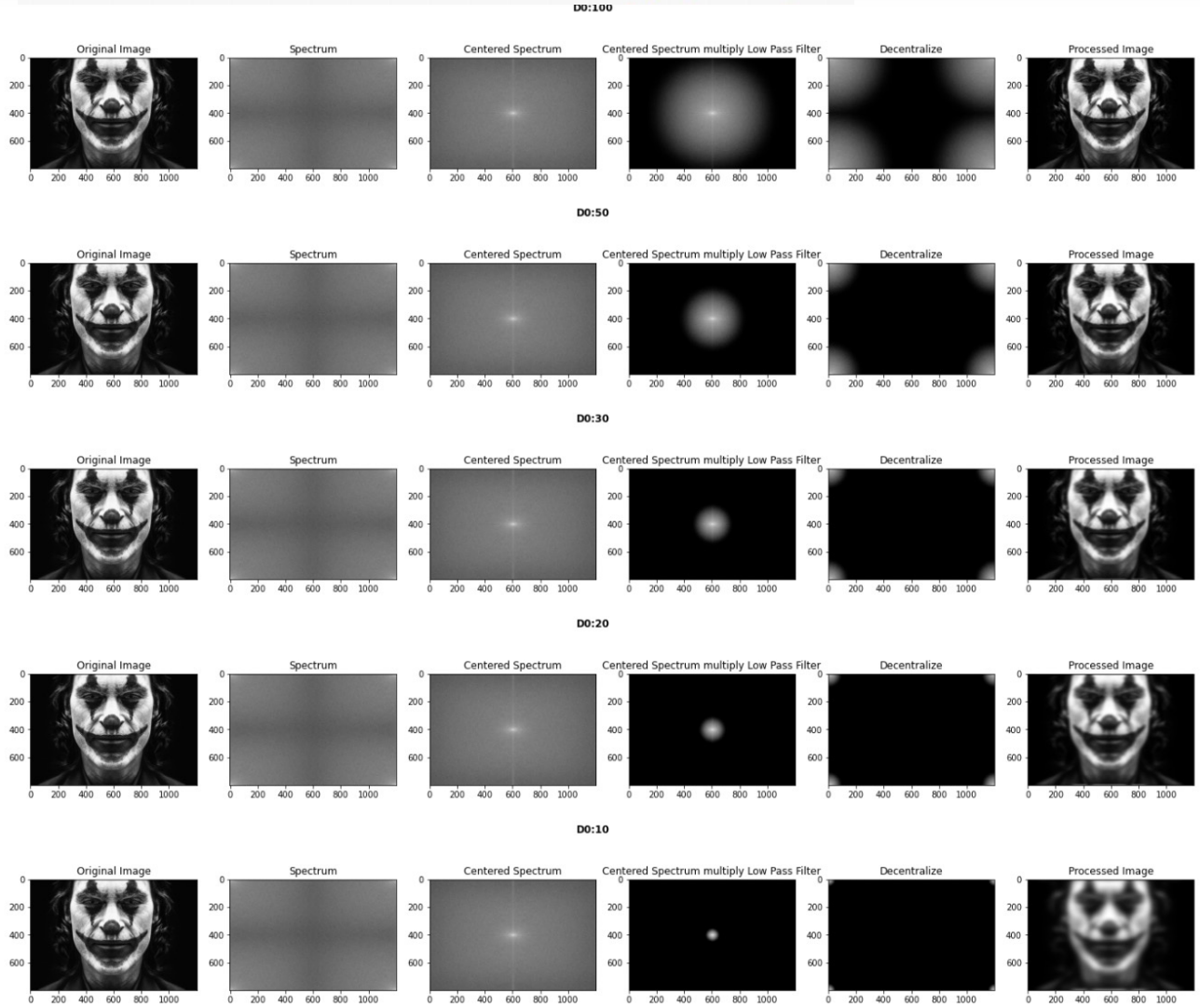
```
original = np.fft.fft2(img)
plt.subplot(152), plt.imshow(np.log(1+np.abs(original)),'gray'), plt.title("Spectrum")
```



Blurring using Low Pass Filter

Low Pass Filter only allow low frequencies to pass

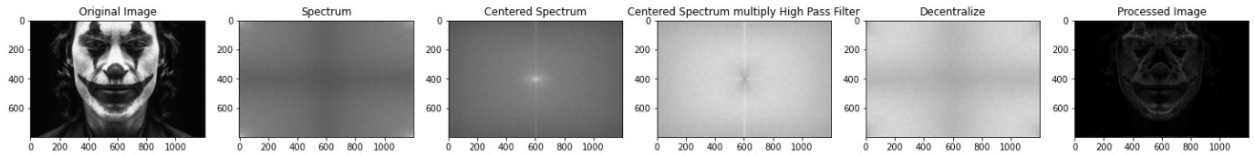
```
base[y,x] = exp(((-distance((y,x),center)**2)/(2*(D0**2))))
```



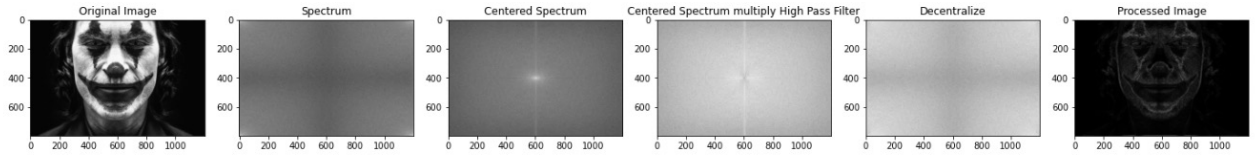
Edge Detection using High Pass Filter

High Pass Filter, on the contrary, only allow higher frequencies to pass

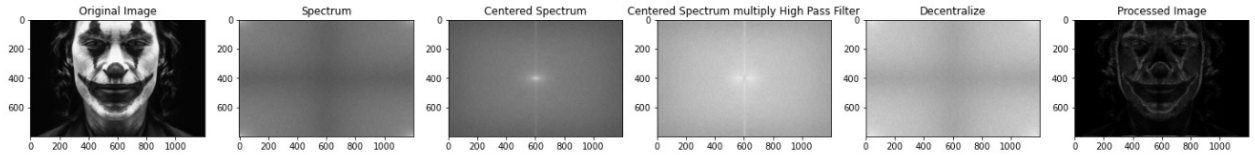
```
base[y,x] = 1 - exp(((-distance((y,x),center)**2)/(2*(D0**2))))
```



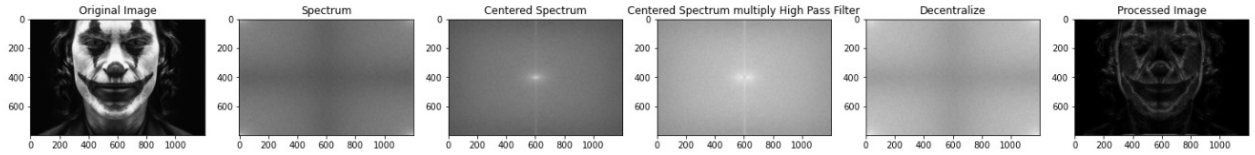
D0:50



D0:30



D0:20



D0:10

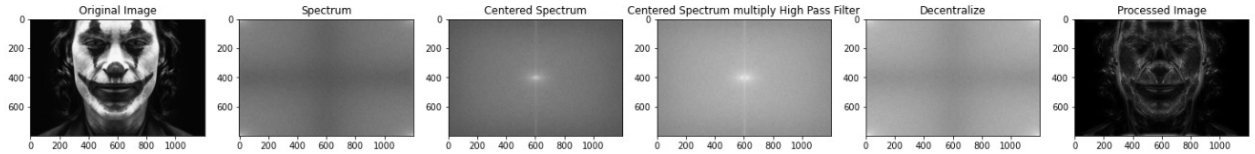
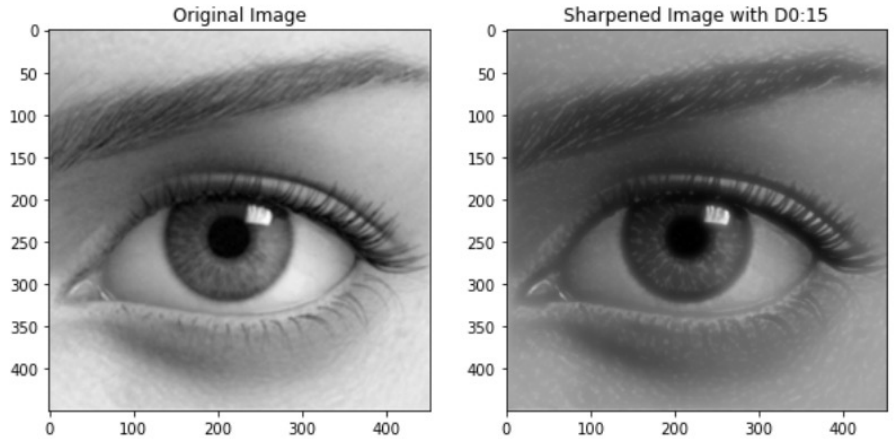


Image Sharpening using High Pass Filter

Image sharpening is done by adding to the original image a signal proportional to a high-pass filtered version of the image.

```
img = cv2.imread('image_sharpen.jpeg', 0)
# perform FFT
original = np.fft.fft2(img)
# get spectrum
center = np.fft.fftshift(original)
# center spectrum for high pass
HighPassCenter = center * gaussianHP(d0,img.shape)
# perform high pass
HighPass = np.fft.ifftshift(HighPassCenter)
# reverse FFT -> IFFT
inverse_HighPass = np.fft.ifft2(HighPass)
# add filter to original image
sharpened = img+np.abs(inverse_HighPass)
```

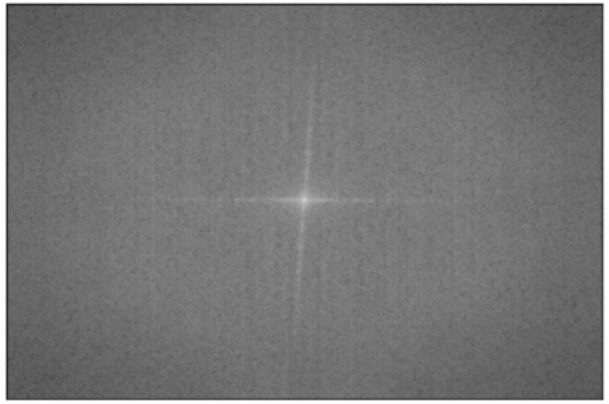


Noise Suppression using Low Pass Filter

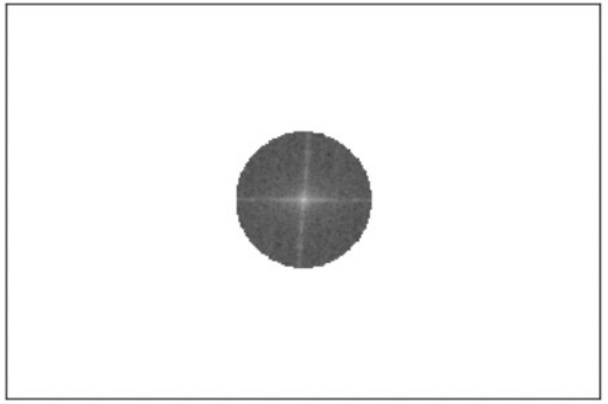
A low pass filter (LPF) is used to remove high frequency noise from the signal and preserves the low frequency components in the signal

Input Image

After FFT



FFT + Mask



After Noise Suppression