INDIAN INSTITUTE OF TECHNOLOGY

KHARAGPUR

DIGITAL IMAGE PROCESSING LABORATORY

A REPORT ON

EXPERIMENT 04

Frequency Domain Filtering

24.02.2021

Group No. 16

Name:	Gambhira Sirish	Shubham Sahoo
Roll No:	**17EC35009**	**17EC35023**

DEPT OF ELECTRONICS AND ELECTRICAL COMMUNICATION ENGINEERING

VISUAL INFORMATION AND EMBEDDED SYSTEMS

Table of Contents

Introduction

In order to perform frequency domain filtering, we need to convert the image into frequency space. Discrete Fourier Transform (DFT) is used to convert the discrete image into frequency domain and vice-versa. The DFT is obtained by decomposing a sequence of values into components of different frequencies. This operation is too slow to be implemented from definition.

Fast Fourier Transform (FFT) rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors. The overall time complexity of each frequency component can be reduced from O (n * n) to O (n * logn) using FFT algorithm, where n is the data-size. It uses a divide - and - conquer approach in order to achieve the same. Since, the inverse of a DFT is the same as the DFT, FFT can be easily adapted.

Frequency domain filtering is used to sharpen or smoothen the image by removing the high/ low frequency components. It is different from spatial filtering as it focuses on the frequency component of the images.

Low Pass Filters:

Low pass filter removes the high frequency components that means it keeps low frequency components. It is used for smoothing the image. It is used to smoothen the image by attenuating high frequency components and preserving low frequency components.

High Pass Filters:

High pass filter removes the low frequency components that means it keeps high frequency components. It is used for sharpening the image. It is used to sharpen the image by attenuating low frequency components and preserving high frequency components. The basic model for filtering is $G(u,v) = H(u,v)F(u,v)$, where F (u,v) represents the fourier transform, H (u,v) represents the transfer function and G (u,v) represents the required transform. The filtered image is obtained by inverse transform of G (u, v)

For most of the cases $H_{high\ pass}(u,v) = 1 - H_{low\ pass}(u, v)$

Algorithm

FFT 2D:

Step 1: Rewrite DFT equation as:

$$\begin{aligned} F(u,v) &= \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)e^{-2\pi i(xu+yu)/N} \\ &= \frac{1}{N}\sum_{x=0}^{N-1} e^{-2\pi ixu/N}\sum_{y=0}^{N-1} f(x,y)e^{-2\pi iyu/N}. \end{aligned}$$

The right hand sum is basically just a one-dimensional DFT if *x* is held constant. The left hand sum is then another one-dimensional DFT performed with the numbers that come out of the first set of sums.

So, we can compute a two-dimensional FFT by [2]:

- We apply row wise FFT for each row of the image - let it be fft_row
- We then apply column-wise FFT on fft_row of the image to get the 2D FFT

This requires a total of 2-*N* one dimensional transforms, so the overall process takes $O(N^2 \log N)$ time.

1D FFT:

The 1D FFT can be computed by first calculating FFT of even-indexed inputs and FFT of odd-indexed inputs, and later combining both to give FFT of sequence [1].

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N}(2m+1)k}$$

From the periodicity of complex exponential, $X_{k\,+\,N/\,2}$ can be obtained directly from this even and odd FFTs. Final expression:

$$\begin{aligned} X_k &= E_k + e^{-\frac{2\pi i}{N}k} O_k \\ X_{k+\frac{N}{2}} &= E_k - e^{-\frac{2\pi i}{N}k} O_k \end{aligned}$$

FFT Shift

Inorder to ease our filtering in frequency domain, the frequency components are shifted to centre before applying filters.

Filter Implementation

Filter	Low Pass	High Pass
Ideal	$H(u,v)=\begin{cases}1 \text{ if } D(u,v)\le D_0\\0 \text{ if } D(u,v)>D_0\end{cases}$	$H(u,v)=\begin{cases}0 \text{ if } D(u,v)\le D_0\\1 \text{ if } D(u,v)>D_0\end{cases}$
Gaussian	$H(u,v)=e^{-D^2(u,v)/2D_0^2}$	$H(u,v)=1-e^{-D^2(u,v)/2D_0^2}$
Butter-worth	$H(u,v)=\frac{1}{1+[D(u,v)/D_0]^{2n}}$	$H(u,v)=\frac{1}{1+[D_0/D(u,v)]^{2n}}$

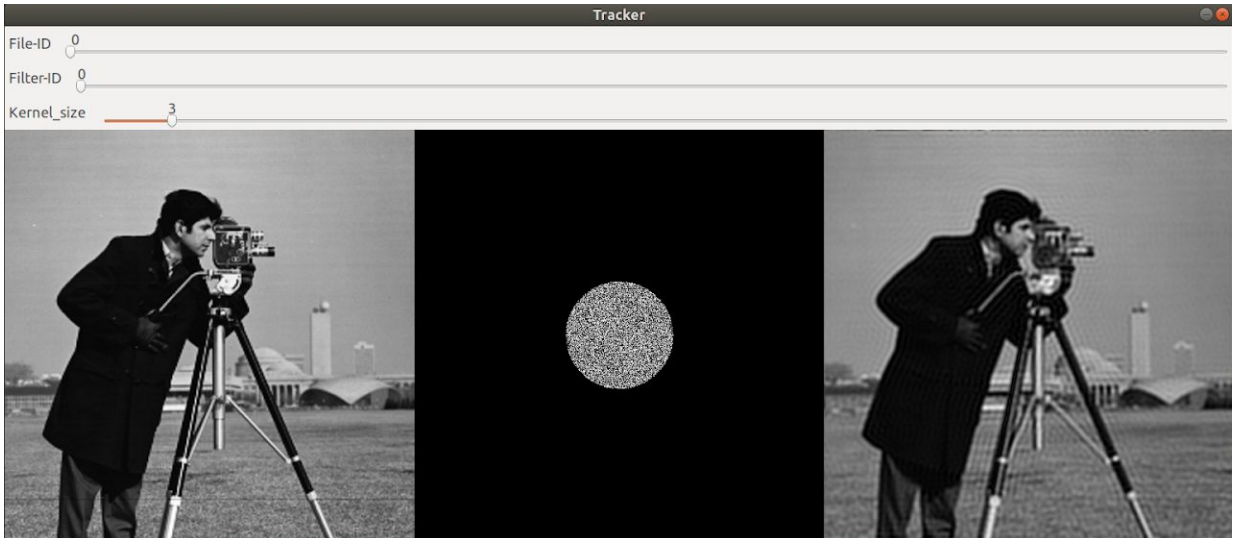
where $D(u,v)=[(u-M/2)^2+(v-N/2)^2]^{1/2}$

The filter order for butter-worth is taken as 5 (constant). The cutoff is multiplied by a factor of sqrt(n) for better visualisation.

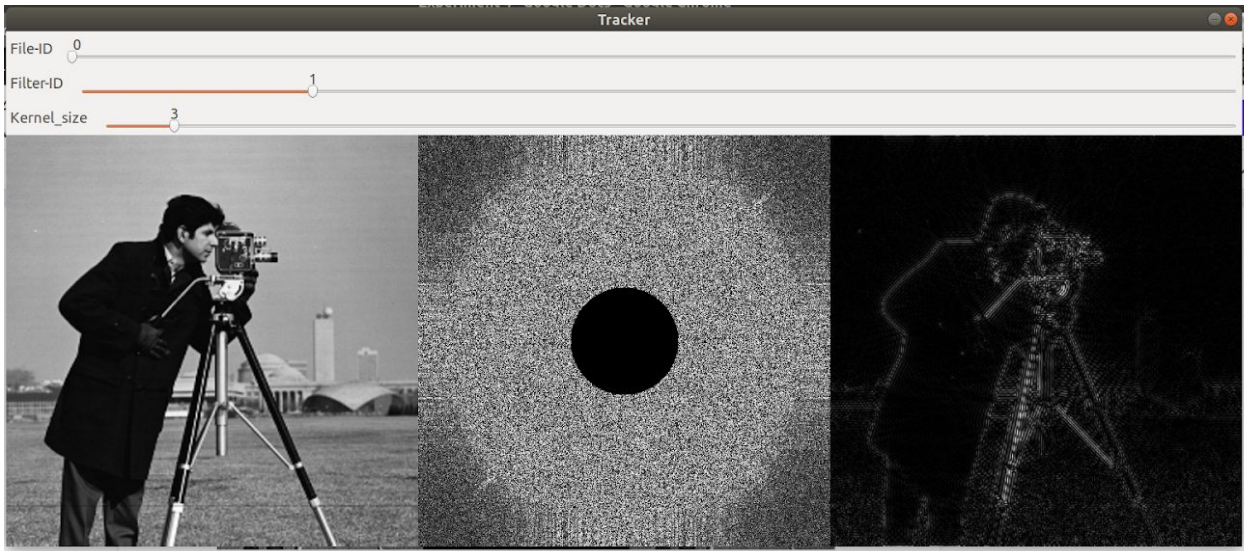
Results

Option Type = Separation

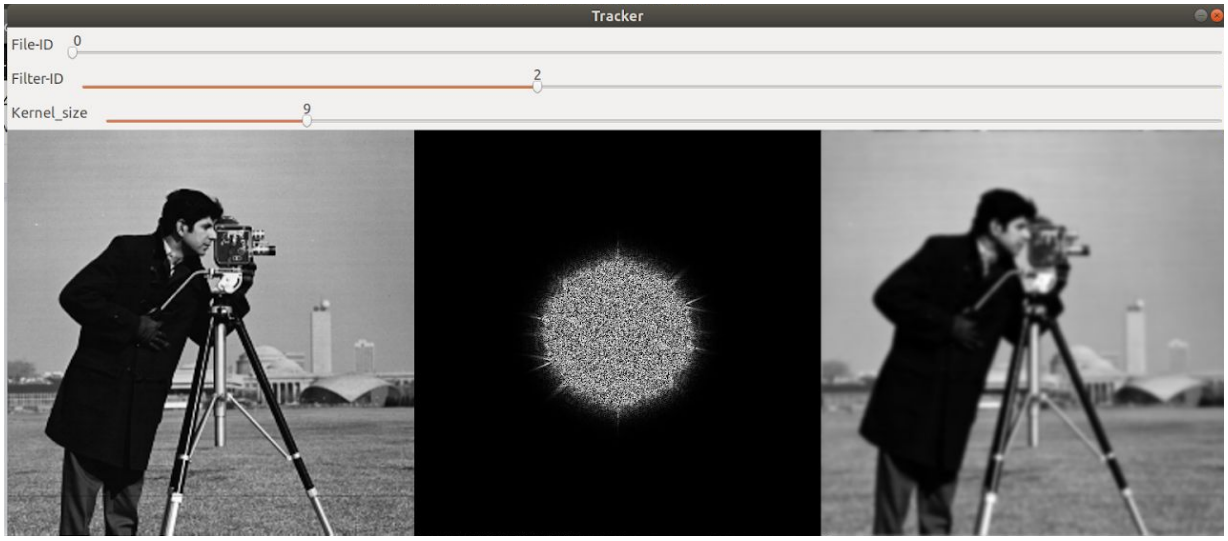
Ideal LPF



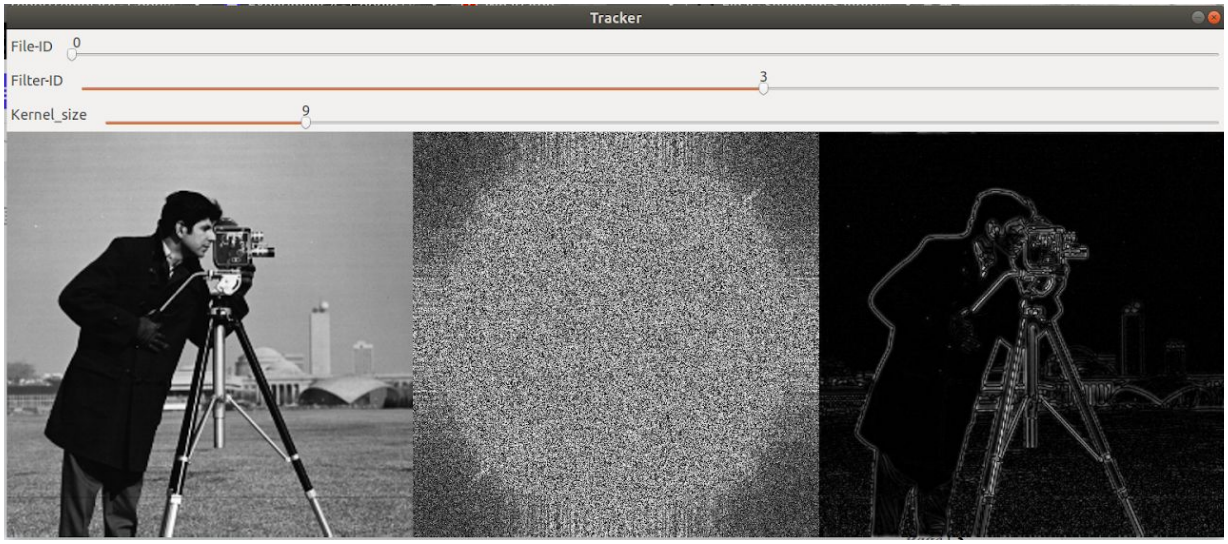
Ideal HPF



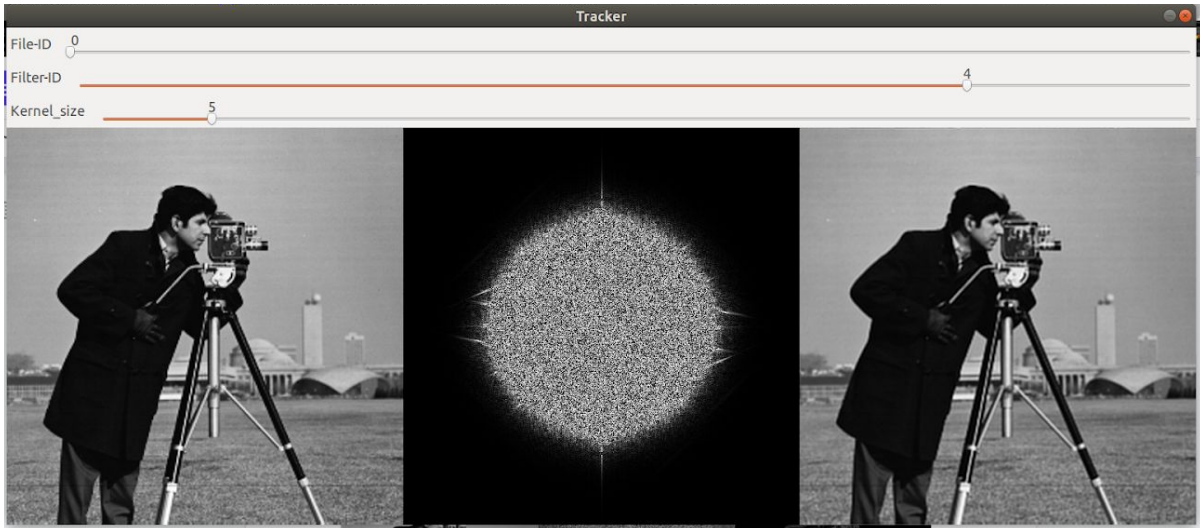
Gaussian LPF



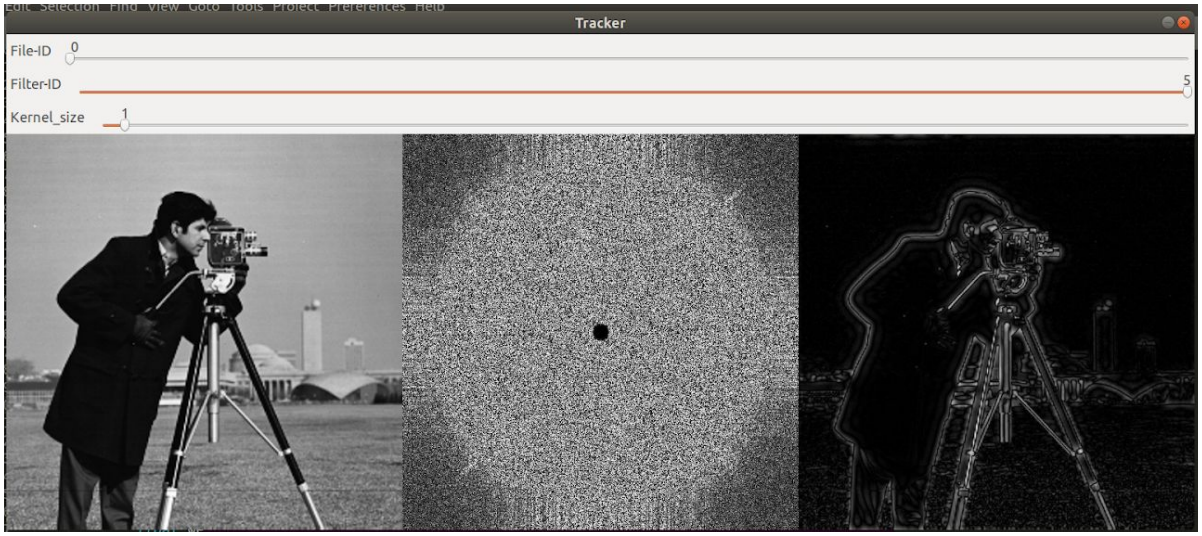
Gaussian HPF



Butterworth LPF

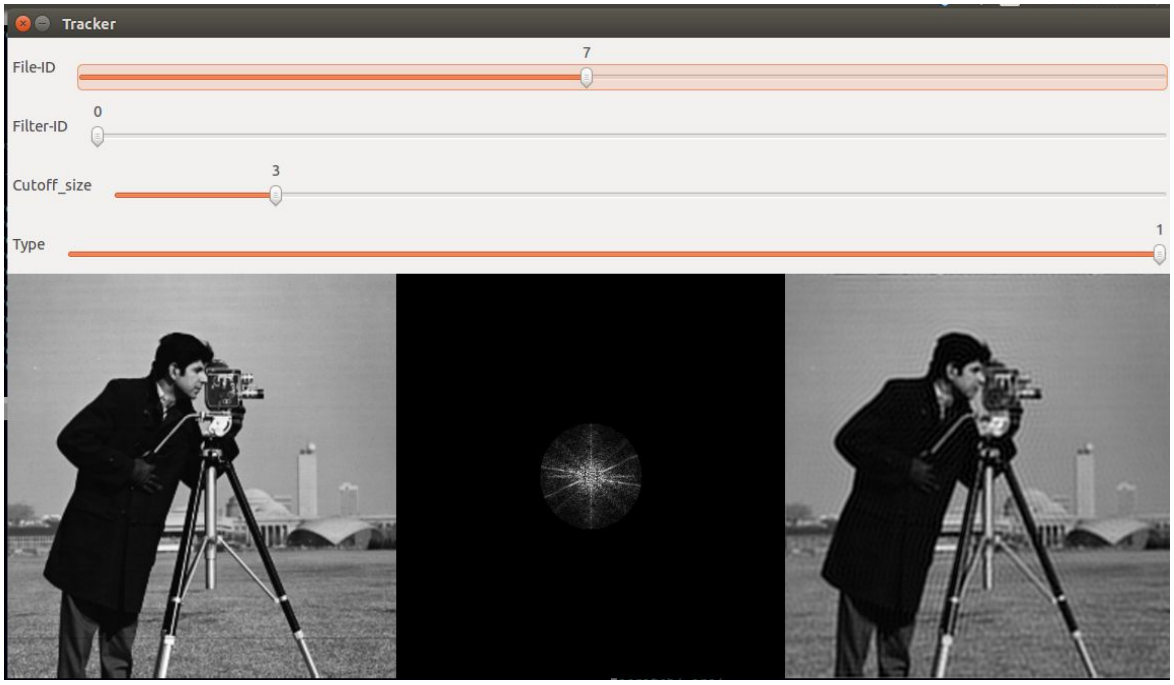


Butterworth HPF

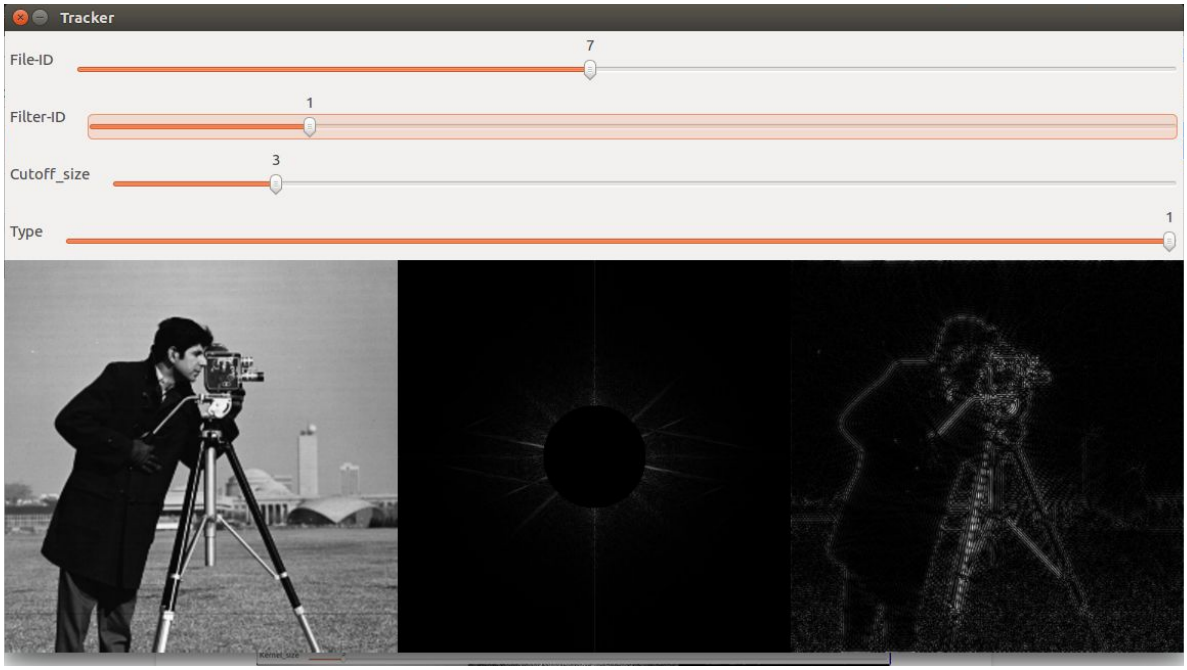


Option Type = Variation

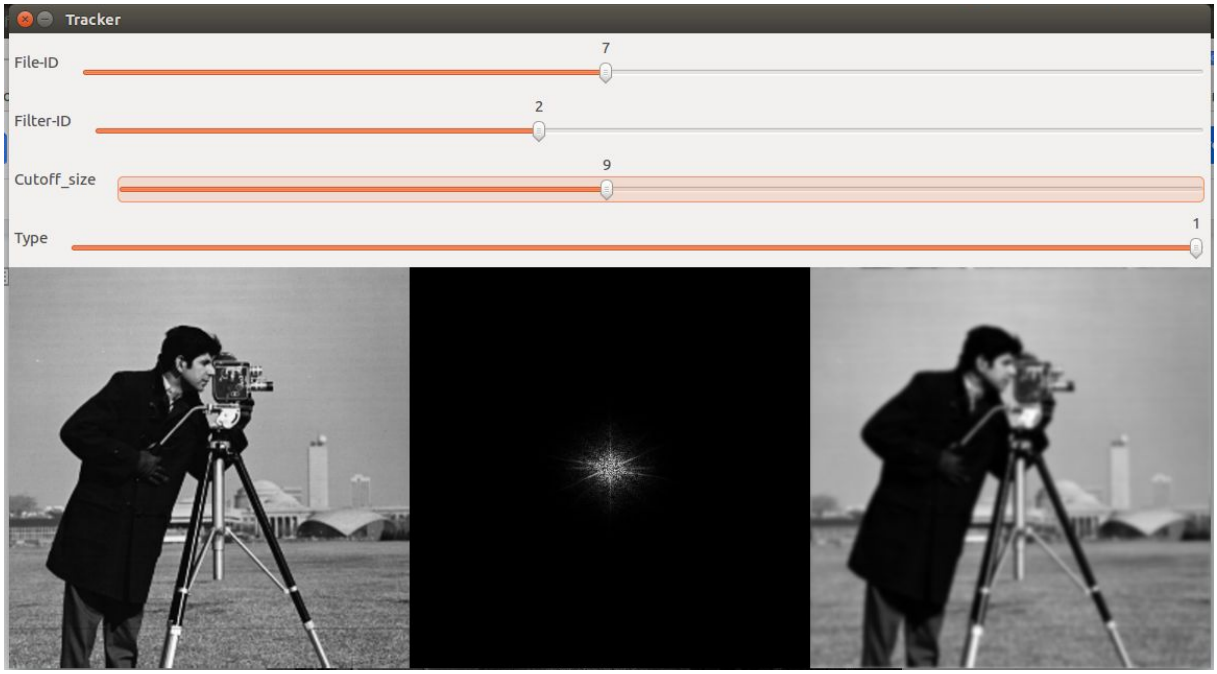
Ideal LPF



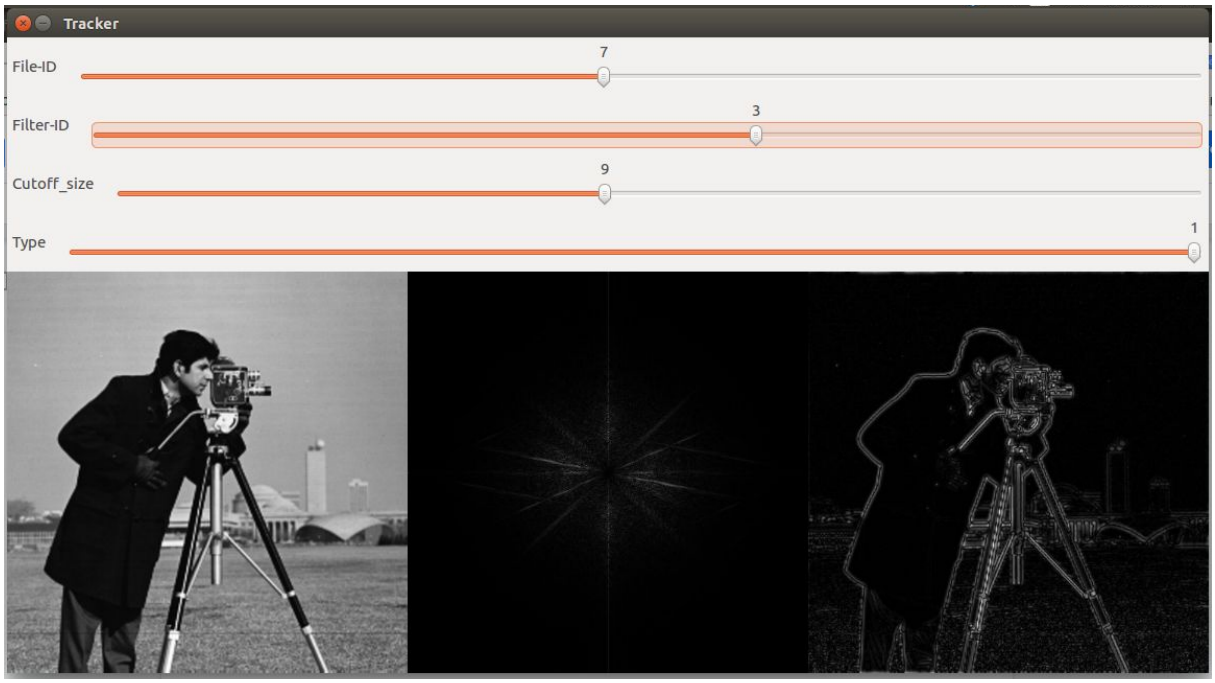
Ideal HPF



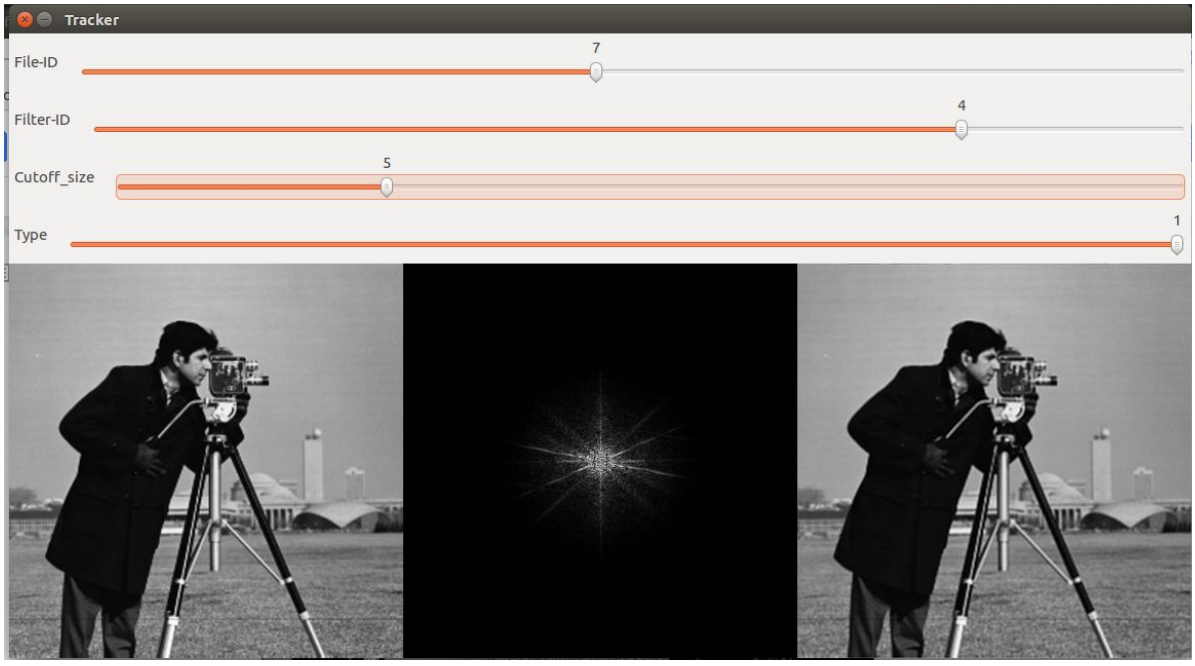
Gaussian LPF



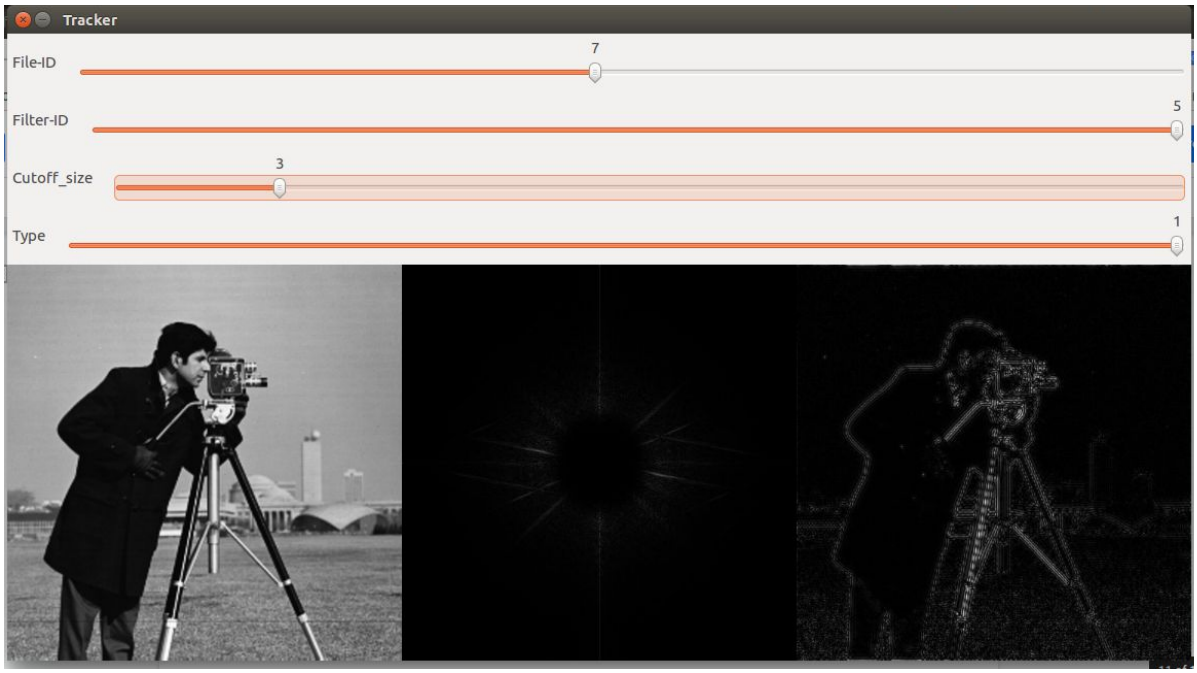
Gaussian HPF



Butterworth LPF



Butterworth HPF



Analysis

- Frequency filtering is a general process as it processes the entire image at once and ignores the presence of any object regions in the image.
- Multiplication operation is easier to perform than convolution. Hence frequency domain filtering is easier to perform.
- We see that the low pass filtered output in Gaussian and Butterworth LPF is better than ideal LPF because the mask used in case of the ideal LPF is very abrupt, whereas in the other the higher frequencies are gradually attenuated.
- We can clearly see the "ringing effect" in case of ideal filters as their inverse fourier transform is a sinc function.
- In HPF also, we see that the edges are more prominent in the outputs and hence they are quite useful in edge detection.
- LPFs are mainly used to remove noise and remove details in the image.
- If HPF is applied to a noisy image, then the noise gets amplified. So various denoising techniques should be performed first before applying HPF to it.
- We were unable to visualise the transfer function of gaussian HPF. The effect of scaling on gaussian HPF was different when compared to other filters.
- The butterworth filter response depends on the order of the filter. We took the butterworth order as n = 5. As the order of butterworth filter increases it tends towards the ideal filter and hence the ringing effect increases.
- We used the same FFT shift function to shift the FFT back and forth.
- The cutoff of the filters are multiplied by sqrt(n) times for better visualisation
- The sigma for the gaussian filter is calculated by dividing the cutoff by 6 [4] to visualise the results better.
- We have used the parameter "type" for scaling and visualizing the filter response output.

Contributions:

Shubham: Ideal Filters, Addon to Gaussian, FFT Implementation

Sirish: Gaussian, Butterworth, FFT Implementation

References

[1] https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm

[2] http://fourier.eng.hmc.edu/e101/lectures/Image_Processing/node6.html

[3] https://universe.bits-pilani.ac.in/uploads/JNKDUBAI/ImageProcessing7-FrequencyFiltering.pdf

[4] https://towardsdatascience.com/fast-fourier-transform-937926e591cb