EE 610 - Image Processing
Assignment 1 -Basic Image Editor
Swapnil Bembde 14D070034

Approach to the assignment :

- firstly, I understood each manipulation that I am going to include in the assignment.
- Then, I implemented histogram of an image and it's equalization.
- Then, I learned Fourier transform of an image and it's shift in (u,v) domain.
- After this I implemented all the manipulations and display options.
- I wasn't comfortable with GUI in python, hence I started learning it.
- Building GUI and linking each function to GUI was my last goal of the assignment.

Main challenges faced:

- Building GUI was main challenge for me.
- Calculation of D(u,v).

Original Image -

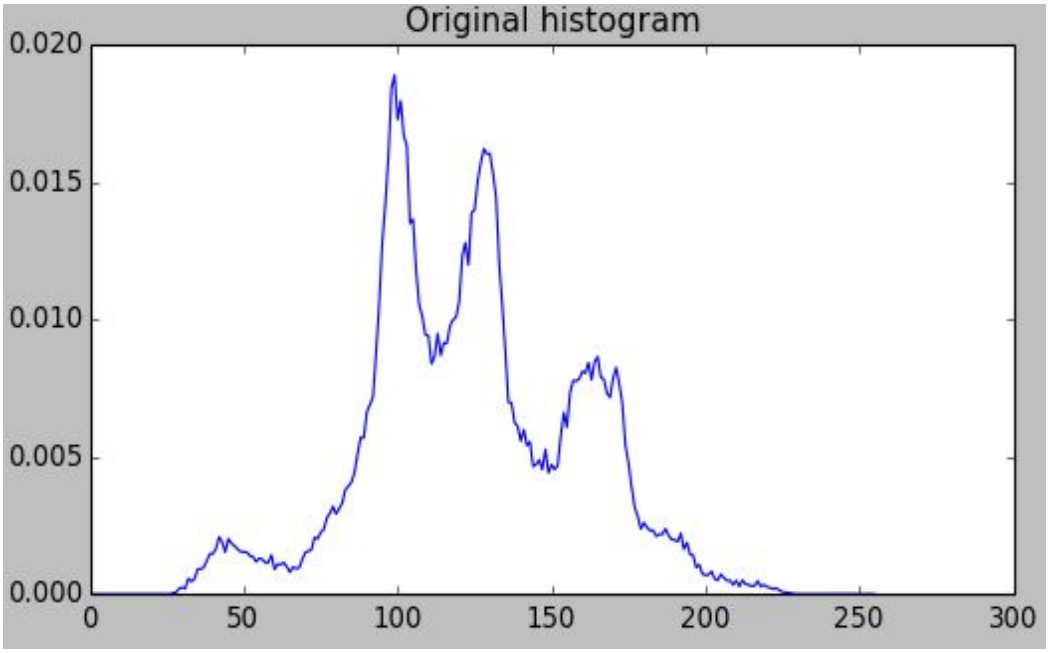
Three subimages -

1.png (256,256)

2.png(256,256)

3.png(256,256)

Histograms of subimages -
Y axis is divided by 256*256.
Images are shown in the order of 1,2,3 with their new histogram



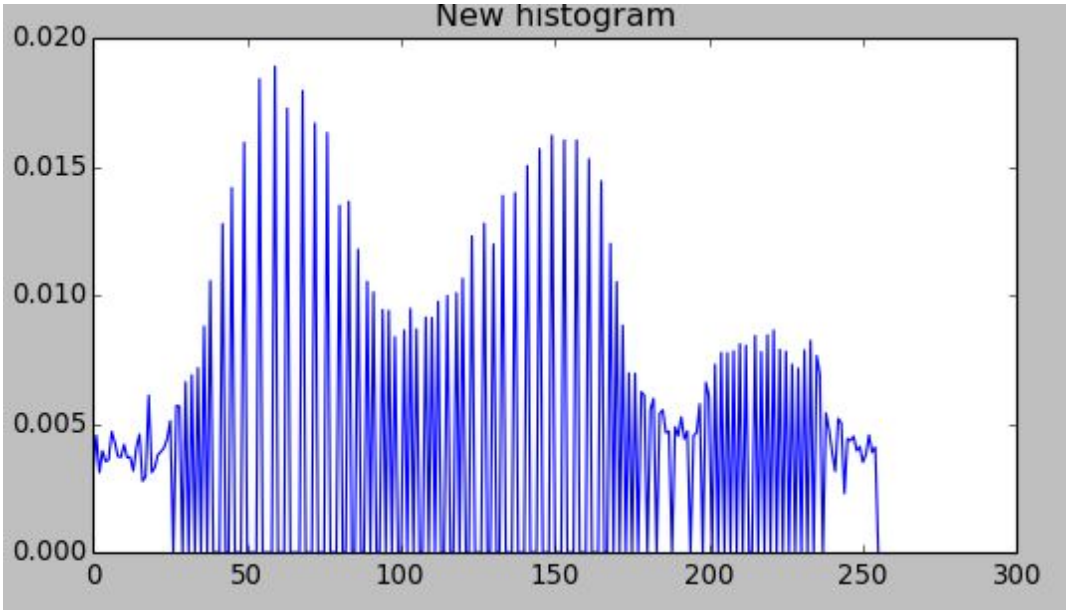
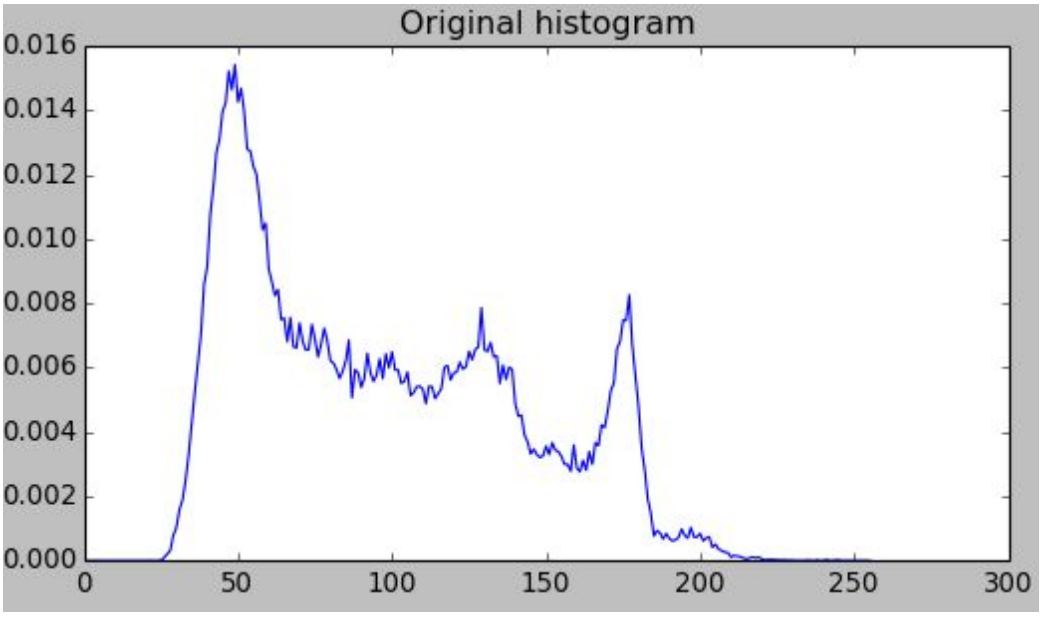
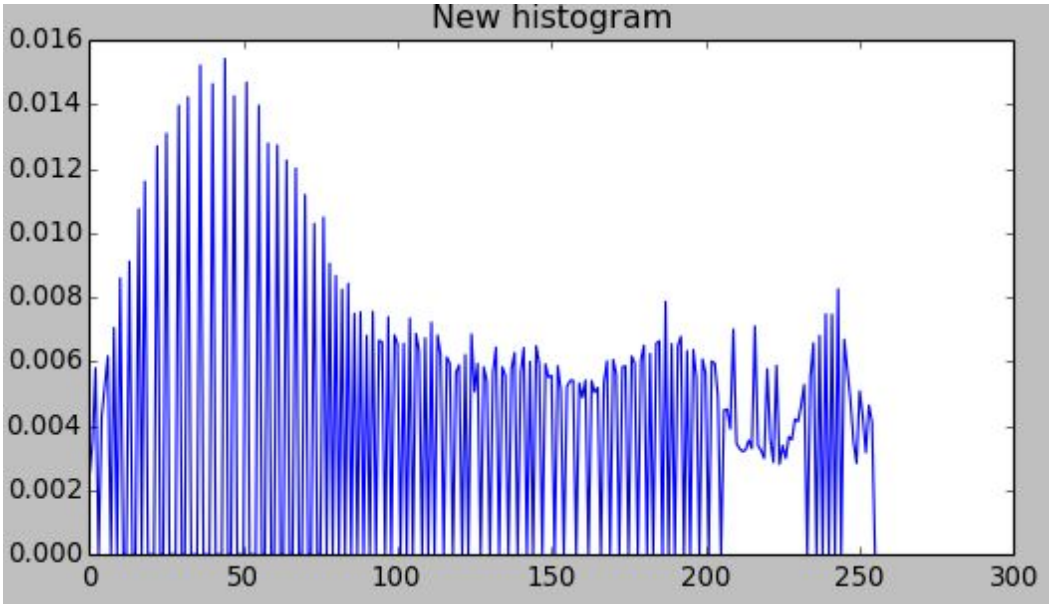
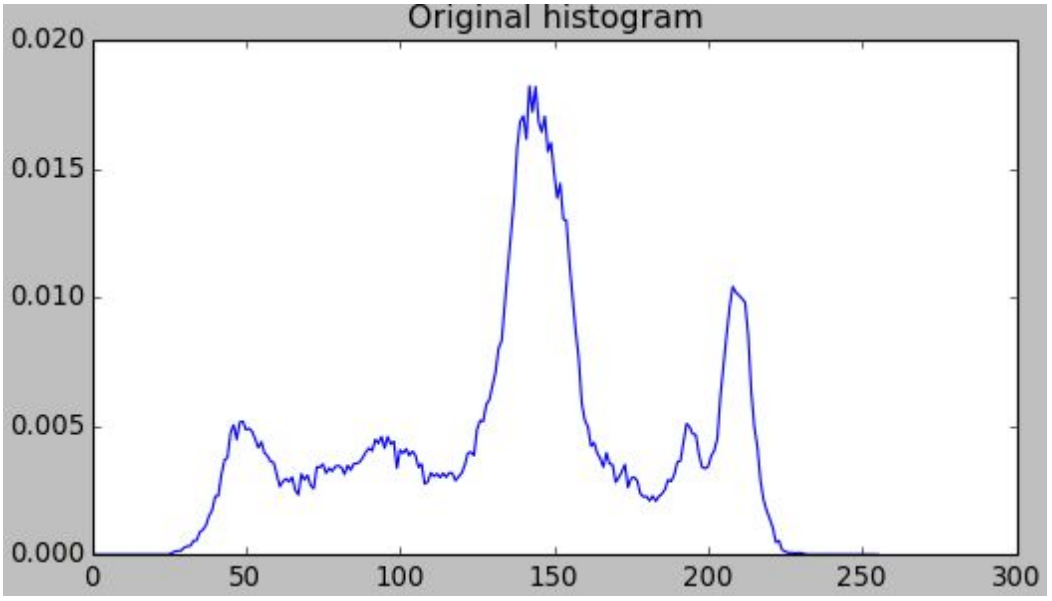

New histogram
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250
300

Image 1


Original histogram
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
50
100
150
200
250
300


New histogram
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
50
100
150
200
250
300


Original histogram
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250
300

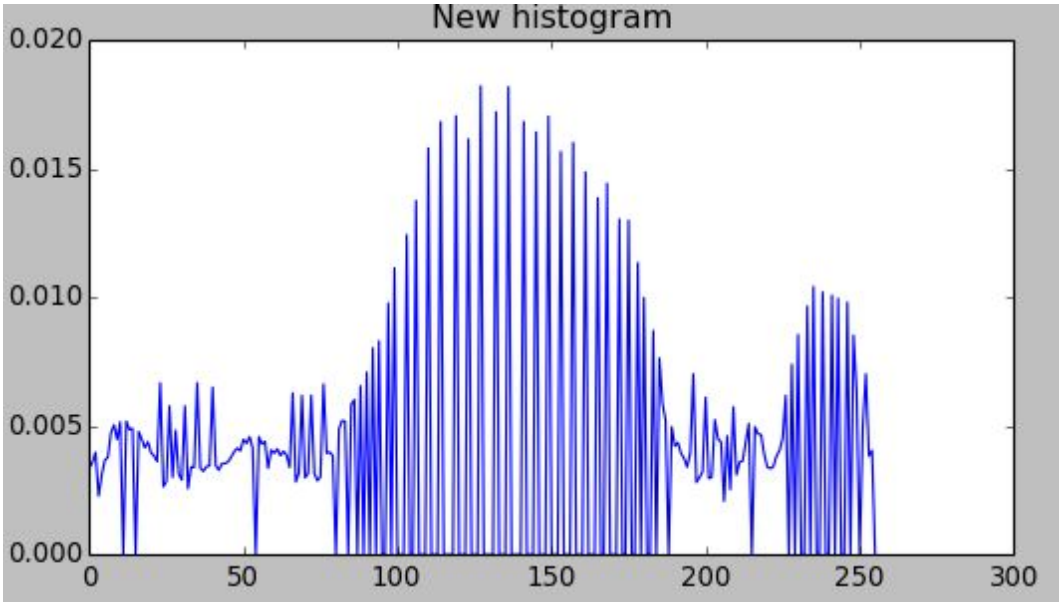
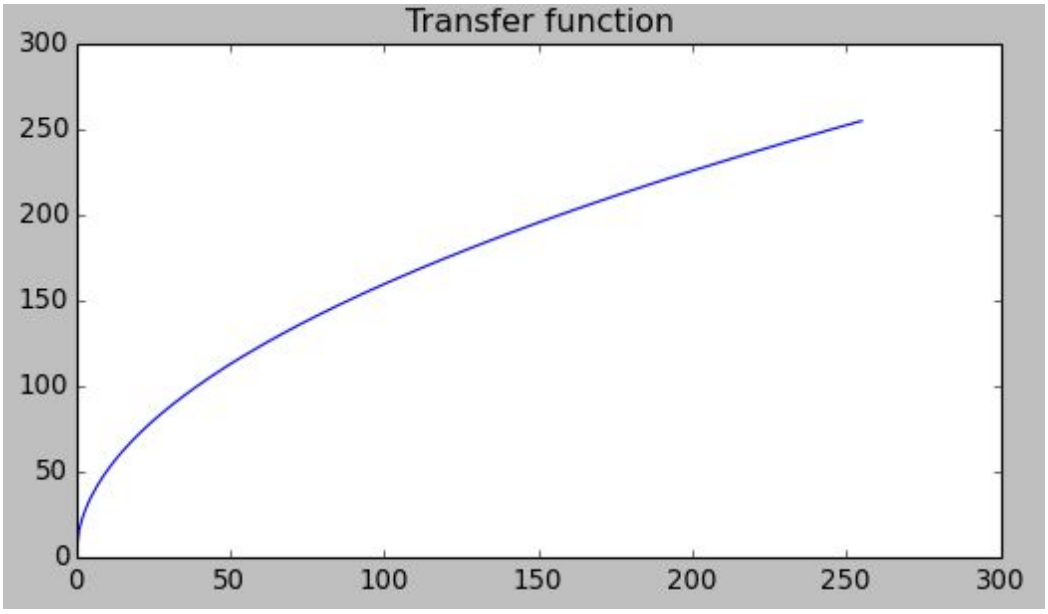

New histogram
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250
300

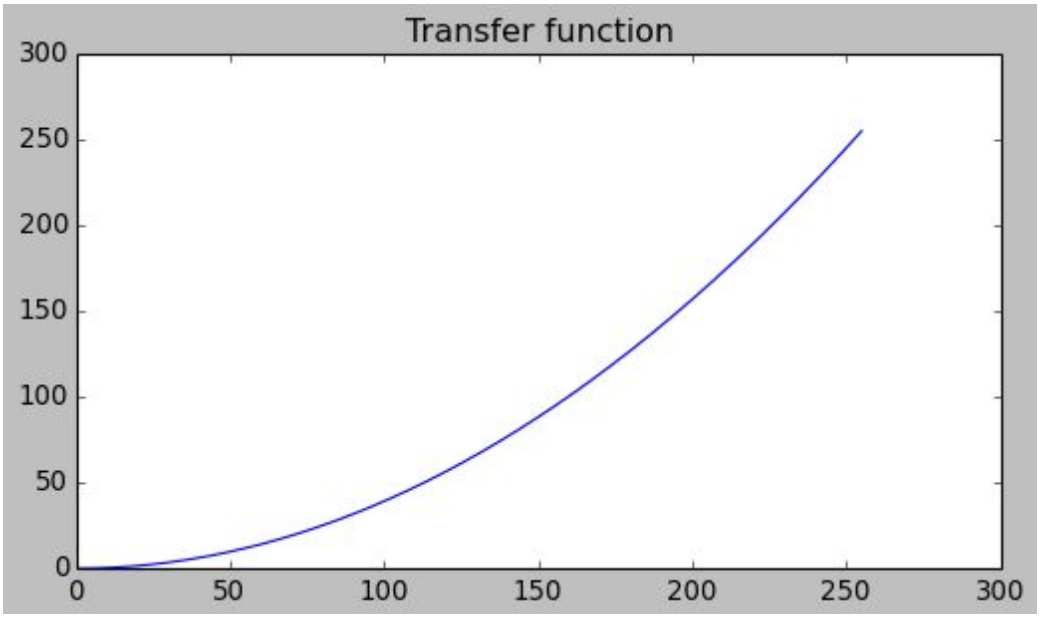
Image 3

Since all the subimages are different, each original histogram is different. Each intensity is different but the histogram shown here, looks like continuous. There should also be vertical lines In the original histograms. New histogram contains more information because it lies over greater range of intensity.

Gamma Correction -
With gamma = 0.5


Transfer function
300
250
200
150
100
50
0
0
50
100
150
200
250
300

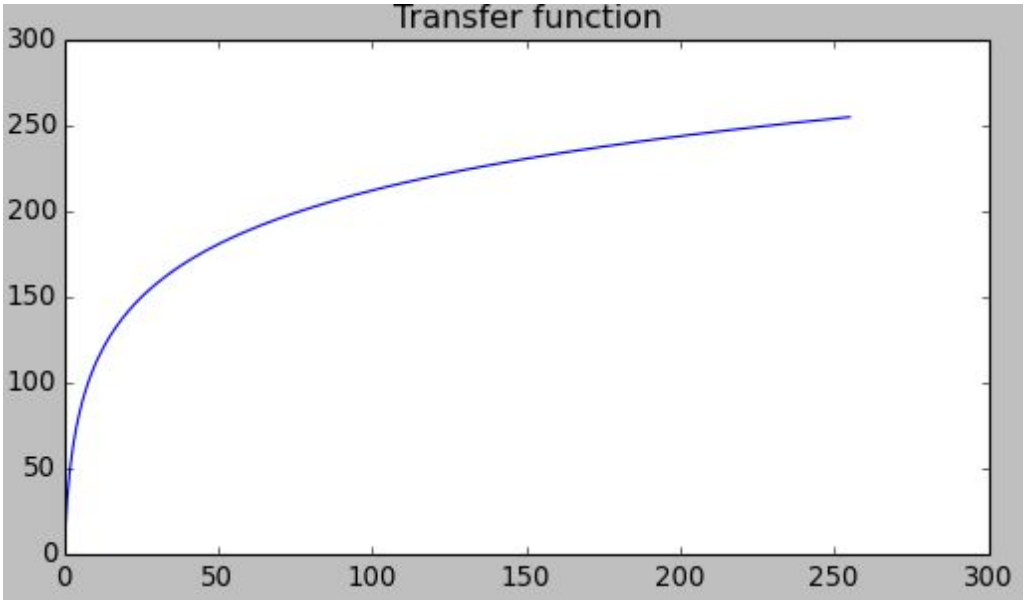
With gamma = 2.0



With gamma = 0.5, we can see that intensity of the pixels has increased (in general) as compared to the original image. This can also be observed by transformation of intensity plot.

With gamma = 2.0 , we can see that intensity of the pixels has decreased (in general) as compared to the original image. This can also be observed by transformation of intensity plot.

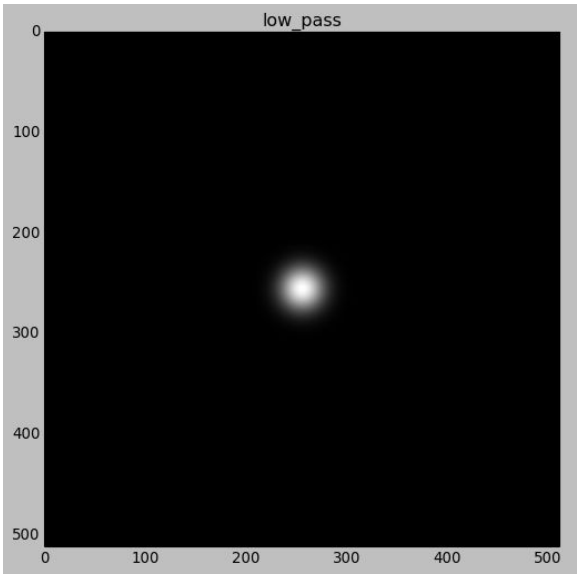
Log Transformation -



we can see that intensity of the pixels has increased (in general) as compared to the original images. This can also be observed by transformation of intensity plot.

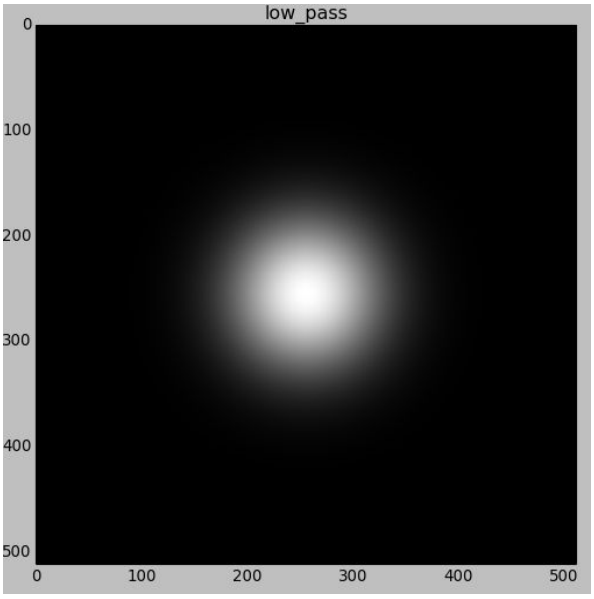
Gaussian Blur -

Blurred subimages with D0 = 15



Fourier Transform of Gaussian filter
D0 = 15

Blurred subimages with D0 = 50



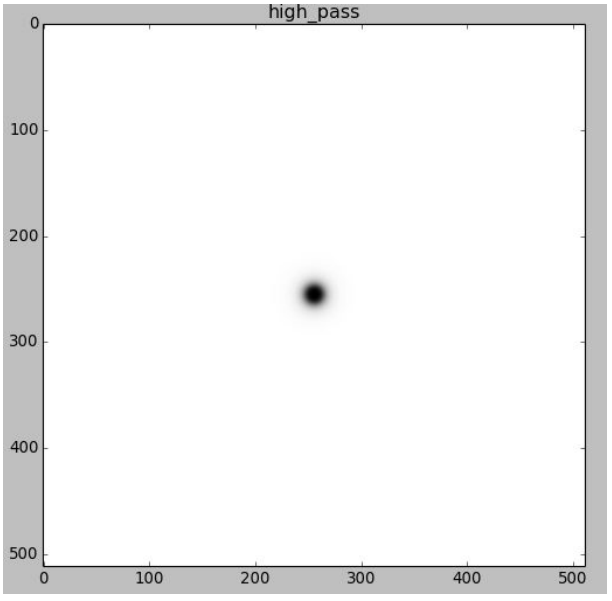
Fourier Transform of Gaussian filter

D0 = 50

As D0 increases, blurring of subimages decreases.

Butterworth Sharpening -

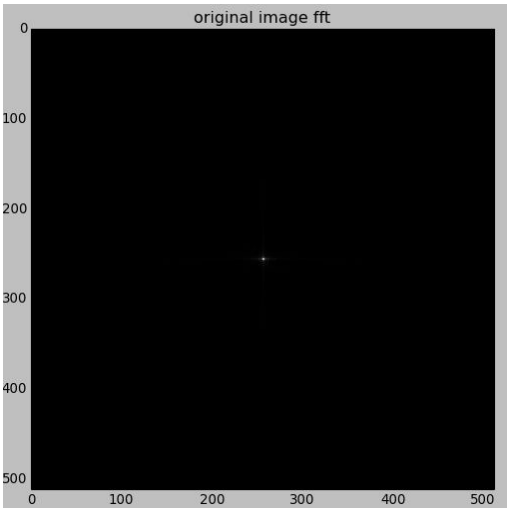
Sharpening is done with parameter values as D0 = 10, n (order =2) and amplification factor (50 %) .

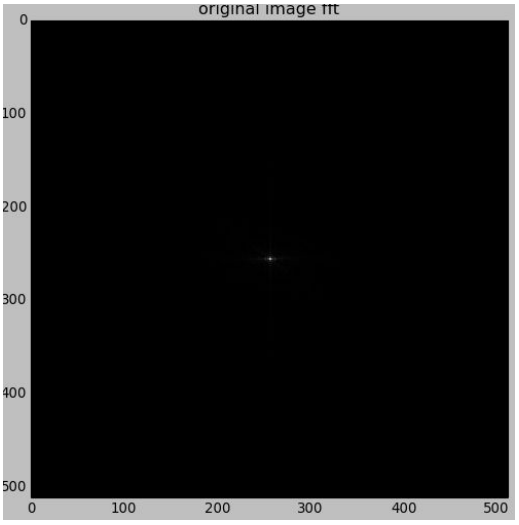


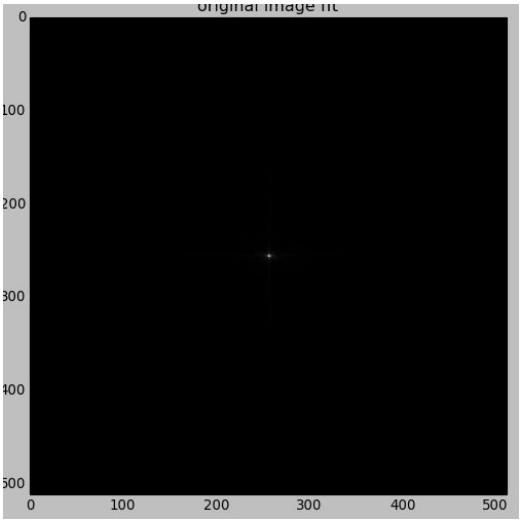
Fourier transform of Butterworth HPF
With D0 = 10

Transition into higher values of cutoff frequencies is much smoother with Butterworth highpass filter.

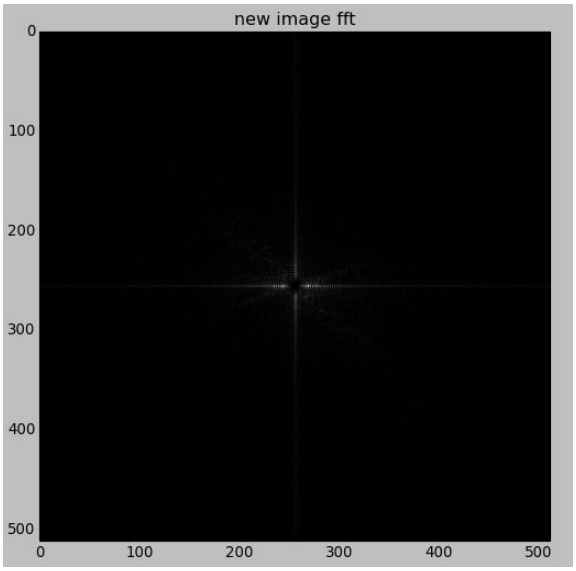
Display of fourier transforms of original and new subimages -

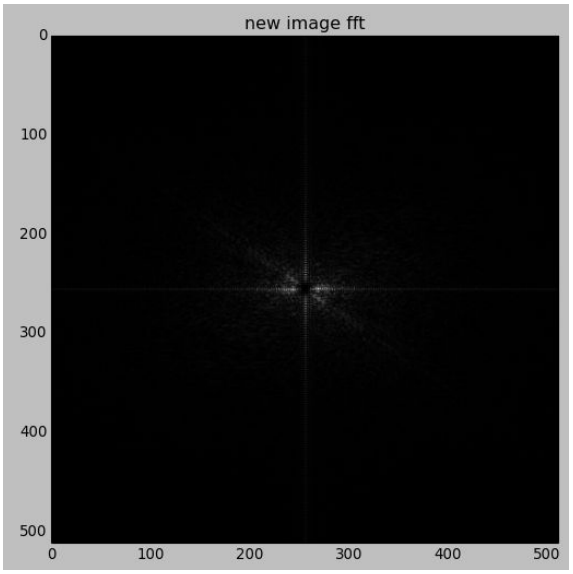


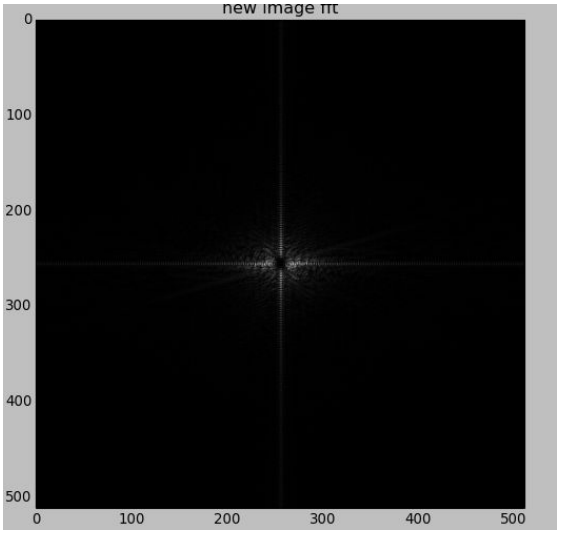




I tried capturing zoomed images but the resolution was not good enough. But I could tell each transform is different from other.
Fourier transform of new sub-images -







New transforms are more distributed over more distance from the center. By observing both, new and old transform of a subimage, we can tell new image is sharper than old image. Because there are more number of bright spots (pixels) in the new image transform.

Appendix-

```
#! /usr/bin/env python

import sys

try:
    from Tkinter import *
except ImportError:
    from tkinter import *

try:
    import ttk
    py3 = 0
except ImportError:
    import tkinter.ttk as ttk
    py3 = 1

import basic_support
from tkinter import filedialog
from tkinter.simpledialog import askfloat
import pylab as plt
import matplotlib.image as mpimg
import numpy as np
from PIL import ImageTk
from PIL import Image
from matplotlib import cm



def vp_start_gui():
    '''Starting point when module is the main routine.'''
    global val, w, root
    root = Tk()
    top = New_Toplevel_1 (root)
    basic_support.init(root, top)
    root.mainloop()

w = None
def create_New_Toplevel_1(root, *args, **kwargs):
    '''Starting point when module is imported by another program.'''
    global w, w_win, rt
```

```
    rt = root
    w = Toplevel (root)
    top = New_Toplevel_1 (w)
    basic_support.init(w, top, *args, **kwargs)
    return (w, top)

def destroy_New_Toplevel_1():
    global w
    w.destroy()
    w = None

#GUI is created using PAGE GUI builder
class New_Toplevel_1:
    def __init__(self, top=None):
        '''This class configures and populates the toplevel window.
           top is the toplevel containing window.'''
        _bgcolor = '#d9d9d9'  # X11 color: 'gray85'
        _fgcolor = '#000000'  # X11 color: 'black'
        _compcolor = '#d9d9d9' # X11 color: 'gray85'
        _ana1color = '#d9d9d9' # X11 color: 'gray85'
        _ana2color = '#d9d9d9' # X11 color: 'gray85'
        font10 = "-family {DejaVu Sans Mono} -size -12 -weight normal "  \
            "-slant roman -underline 0 -overstrike 0"
        font11 = "-family {Tibetan Machine Uni} -size 12 -weight bold "  \
            "-slant roman -underline 0 -overstrike 0"

        top.geometry("858x546+335+45")
        top.title("New Toplevel 1")
        top.configure(highlightcolor="black")

        #opens the local space
        def open_file():
            global orignl_img
            global currnt_img
            filename = filedialog.askopenfilename(filetypes =(("png format",
"*.png"),("tif format", "*.tif"),("All Files","*.*")),title = "Choose a file.")
            orignl_img = np.uint8(mpimg.imread(filename)*255.0)
            if orignl_img.shape.count(3) == 1:
                orignl_img = np.uint8((0.2126* orignl_img[:,:,0]) + \
                      np.uint8(0.7152 * orignl_img[:,:,1]) +\
                         np.uint8(0.0722 * orignl_img[:,:,2]))
            currnt_img = orignl_img
```

```
    # saves the image
    def save():
        global currnt_img
        save_filename = filedialog.asksaveasfilename(filetypes = (('png format',
'.png'),('All files', '*')),title='Save image as')
        mpimg.imsave(save_filename,np.uint8(currnt_img), cmap = cm.gray)

    #Undo all function
    def undo():
        global orignl_img
        global undo_img
        undo_img = ImageTk.PhotoImage(Image.fromarray(orignl_img))
        self.Canvas1.create_image(0, 0, anchor = NW,image = undo_img)

    #displays image for any manipulation
    def display_image():
        global currnt_image
        global display_img
        display_img = ImageTk.PhotoImage(Image.fromarray(currnt_img))
        self.Canvas1.create_image(0, 0, anchor = NW,image = display_img)

    #displays histogram of the current image
    def display_histogram():
        global currnt_img
        plt.hist(currnt_img.ravel(), 256, [0, 256])
        plt.show()

    def display_FFT_magnitude():
        global currnt_img
        f_img = np.fft.fft2(currnt_img)
        fshift = np.fft.fftshift(f_img)#shifting
        mag = 20*np.log(np.abs(fshift))
        plt.imshow(mag)
        plt.set_cmap('gray')
        plt.show()


    def display_FFT_phase():
        global currnt_img
        f_img = np.fft.fft2(currnt_img)
        fshift = np.fft.fftshift(f_img)
```

```
        phase = np.arctan(fshift.imag/fshift.real)#phase calculation
        plt.imshow(phase)
        plt.set_cmap('gray')
        plt.show()

    def log_trans():
        global currnt_img
        def log_transform(itnsity):

            return (255.0*(np.log(1+itnsity)/np.log(256))) #assuming max intensity in
the image = 255
        img = currnt_img
        h_out = np.zeros_like(img)
        m, n = img.shape
        for i in range(0, m):
            for j in range(0, n):
                h_out[i, j] = log_transform(img[i, j]) # generating  transformed image
        currnt_img = h_out

    def gamma_corr():
        global currnt_img
        Gamma = askfloat("Enter the value of gamma",'please enter gamma')
        def gamma_correction(itnsity, gamma):
            itnsity = itnsity/255.0
            itn2 = np.log(itnsity+1e-10)*gamma
            return (np.exp(itn2)*255.0)
        img = currnt_img
        h_out = np.zeros_like(img)
        m, n = img.shape
        for i in range(0, m):
            for j in range(0, n):
                h_out[i, j] = gamma_correction(img[i, j],Gamma)
        currnt_img = h_out

    # supporting function of histogram equalisation( histeq)
    def histo(img):
      # calculating number of pixels
        m, n = img.shape
        h = [0.0] * 256
        for i in range(m):
            for j in range(n):
                h[img[i, j]]+=1
```

```
        return np.array(h)

    def histeq():
        global currnt_img
        h = histo(currnt_img)
        #calculate Histogram
        m, n = currnt_img.shape # shape of image
        tf_sum = np.array([sum(h[:i+1]) for i in range(len(h))]) # transformation
        cum_tf = np.uint8(255 * tf_sum/(m*n)) # cummulative sum
        h_out = np.zeros_like(currnt_img)
        # new values
        for i in range(0, m):
            for j in range(0, n):
                h_out[i, j] = cum_tf[currnt_img[i, j]]
        currnt_img = h_out # equalized image

    def glpf():
        global currnt_img
        D0 = askfloat("Enter the value of D0",'please enter D0')
        if(D0 == None):
            D0 = 3.0

        # crops the given image into (m/2,n/2) image
        def crop(image):
            lx, ly = image.shape
            return image[:lx/2, :ly/2]

        def lpf(shape,d0=3.0):
            m,n = shape
            u = np.linspace(-0.5, 0.5, n) *n
            v = np.linspace(-0.5, 0.5, m) *m
            d = np.sqrt((u**2)[np.newaxis] + (v**2)[:, np.newaxis]) # calculating d
            filt = np.exp(-d*d/(2*d0*d0)) # gaussian function
            return filt
        img = currnt_img
        m, n = img.shape
        new_img = np.zeros((2*m,2*n))
        new_img[:img.shape[0],:img.shape[1]] = img #reshaping image and padding

        fft_orig = np.fft.fftshift(np.fft.fft2(new_img)) # fourier transform of original
image with shift
        low_pass = lpf(new_img.shape,D0)#filter
```

```
        new_filt = fft_orig*low_pass# fft of modified image
        recon_image = np.abs(np.fft.ifft2(np.fft.ifftshift(new_filt))) #inverse shifting
and inverse fft
        currnt_img = crop(recon_image) # croping

    def bhpf():
        global currnt_img
        D0 = askfloat("Enter the value of D0",'please enter D0')
        if(D0 == None):
            D0 = 3.0
        Order = askfloat("Order of Butterworth HPF",'please enter an integer')
        if(Order == None):
            Order = 2
        # crops the given image into (m/2,n/2) image
        def crop(image):
            lx, ly = image.shape
            return image[:lx/2, :ly/2]
        def hpf(shape,d0=3,order=2):
            d0  =float(d0)
            m, n = shape
            u = np.linspace(-0.5, 0.5, n) *n
            v = np.linspace(-0.5, 0.5, m) *m
            d = np.sqrt((u**2)[np.newaxis] + (v**2)[:, np.newaxis]) # calculating d
            filt = 1 /(1.0 + (d0/d)**(2*order)) # butterworth filter
            return filt

        img = currnt_img
        m, n = img.shape
        new_img = np.zeros((2*m,2*n))
        new_img[:img.shape[0],:img.shape[1]] = img #reshaping image and padding

        fft_orig = np.fft.fftshift(np.fft.fft2(new_img))# fourier transform of original
image with shift
        high_pass  = hpf(new_img.shape,D0,Order)#filter
        new_filt = fft_orig*high_pass# fft of modified image
        recon_image = np.abs(np.fft.ifft2(np.fft.ifftshift(new_filt)))
        currnt_img = crop(recon_image)#crop

        # creating buttons, image space and labels for GUI
    self.Button1 = Button(top,command = open_file) # with command linking of
perticular function
    self.Button1.place(relx=0.07, rely=0.04, height=27, width=58)
```

```
self.Button1.configure(activebackground="#d9d9d9")
self.Button1.configure(text='''Open''')

self.Button2 = Button(top,command = save)
self.Button2.place(relx=0.22, rely=0.04, height=27, width=55)
self.Button2.configure(activebackground="#d9d9d9")
self.Button2.configure(text='''Save''')

self.Button3 = Button(top,command =histeq)
self.Button3.place(relx=0.01, rely=0.18, height=27, width=142)
self.Button3.configure(activebackground="#d9d9d9")
self.Button3.configure(text='''Equalize Histogram''')

self.Button4 = Button(top,command =gamma_corr)
self.Button4.place(relx=0.16, rely=0.33, height=27, width=139)
self.Button4.configure(activebackground="#d9d9d9")
self.Button4.configure(text='''Gamma Correction''')

self.Button5 = Button(top,command =log_trans)
self.Button5.place(relx=0.01, rely=0.26, height=27, width=113)
self.Button5.configure(activebackground="#d9d9d9")
self.Button5.configure(text='''Log Transform''')

self.Button6 = Button(top,command =glpf)
self.Button6.place(relx=0.2, rely=0.26, height=27, width=108)
self.Button6.configure(activebackground="#d9d9d9")
self.Button6.configure(text='''Guassian Blur''')

self.Button7 = Button(top,command = undo)
self.Button7.place(relx=0.01, rely=0.33, height=27, width=75)
self.Button7.configure(activebackground="#d9d9d9")
self.Button7.configure(text='''Undo all''')

self.but46 = Button(top,command =bhpf)
self.but46.place(relx=0.2, rely=0.18, height=27, width=96)
self.but46.configure(activebackground="#d9d9d9")
self.but46.configure(text='''Sharpening''')

self.Label1 = Label(top)
self.Label1.place(relx=0.09, rely=0.11, height=35, width=122)
self.Label1.configure(activebackground="#f9f9f9")
self.Label1.configure(font=font11)
```

```
        self.Label1.configure(text='''Manipulations''')

        self.Button9 = Button(top, command = display_image)
        self.Button9.place(relx=0.05, rely=0.46, height=27, width=64)
        self.Button9.configure(activebackground="#d9d9d9")
        self.Button9.configure(text='''Image''')

        self.Button10 = Button(top, command = display_histogram)
        self.Button10.place(relx=0.17, rely=0.46, height=27, width=88)
        self.Button10.configure(activebackground="#d9d9d9")
        self.Button10.configure(text='''Histogram''')

        self.Button11 = Button(top,command = display_FFT_magnitude)
        self.Button11.place(relx=0.05, rely=0.53, height=27, width=90)
        self.Button11.configure(activebackground="#d9d9d9")
        self.Button11.configure(text='''Magnitude''')

        self.Button12 = Button(top, command = display_FFT_phase)
        self.Button12.place(relx=0.17, rely=0.53, height=27, width=61)
        self.Button12.configure(activebackground="#d9d9d9")
        self.Button12.configure(text='''Phase''')

        self.Label2 = Label(top)
        self.Label2.place(relx=0.1, rely=0.38, height=35, width=66)
        self.Label2.configure(activebackground="#f9f9f9")
        self.Label2.configure(font=font11)
        self.Label2.configure(text='''Display''')

        self.Canvas1 = Canvas(top)  # for image space
        self.Canvas1.place(relx=0.38, rely=0.04, relheight=0.94, relwidth=0.6)
        self.Canvas1.configure(background="#e3beff")
        self.Canvas1.configure(borderwidth="2")
        self.Canvas1.configure(relief=RIDGE)
        self.Canvas1.configure(selectbackground="#c4c4c4")
        self.Canvas1.configure(width=514)

# for mainloop
if __name__ == '__main__':
    vp_start_gui()
```

Supporting file for GUI :-

```
#! /usr/bin/env python
#


import sys

try:
    from Tkinter import *
except ImportError:
    from tkinter import *

try:
    import ttk
    py3 = 0
except ImportError:
    import tkinter.ttk as ttk
    py3 = 1


def init(top, gui, *args, **kwargs):
    global w, top_level, root
    w = gui
    top_level = top
    root = top

def destroy_window():
    # Function which closes the window.
    global top_level
    top_level.destroy()
    top_level = None

if __name__ == '__main__':
    import basic
    basic.vp_start_gui()
```