Basic Image Editor

Assignment-1

submitted by

Sarthak Nijhawan
150070037,
Electrical Engineering Department,
IIT Bombay

Harshit Khariwal
15D070026,
Electrical Engineering Department,
IIT Bombay

***Abstract*—This paper focuses on the implementation and design of a GUI based Basic Image Editor that incorporates and exhibits the performance of various tools and techniques borrowed from the domain of Image Processing. This electronic document presents a MATLAB based GUI that employs various standard image enhancement techniques ranging from intensity transformations in spatial domain to filtering in the frequency domain. It also tends deploy various secondary support features such as UNDO, RESET etc. reinforcing easy-to-use nature of the GUI for the user.**

***Index Terms*—Image Processing, Image Enhancement, Frequency Domain Filtering, MATLAB, Fast Fourier Transform**

.

I. Introduction

The main objective of this assignment is to build a robust, efficient and optimized GUI based platform to evaluate the performance of various Image Processing operations on images. The GUI consists of two categories of features, namely, *Primary Features* and *Secondary Support Features*. The former set of features consists of image enhancement techniques while the latter focuses on efficient handling of those operations over images in the GUI such as *Import Image, Save Image, UNDO, RESET etc*.

Apart from the basic requirements asked in the assignment we have successfully introduced more exciting features in the editor. The extra set of features include:

- Primary Features
  1) Adaptive Thresholding
  2) Bit Plane Slicing
  3) Geometric Transformations
  4) Common Frequency Domain Filters
  5) Radix-2 DIT FFT Algorithm for 2D-DFT
- Secondary Features
  1) UNDO *(more than 1 undo allowed)*
  2) Sharpness Slider
  3) Blurrness Slider

In addition to features mentioned before, colored images are also handled accordingly in the editor. Owing to the fact that most of the spatial domain and filter domain techniques apply to the intensity maps of images, these transformations are applied on intensity map equivalent of a colored image in *HSV* space, i.e. the *Value(V) Channel*. After the processing, the enhanced maps are then used to reconstruct and revert the image back to *RGB* space for further processing, storage and display.

Moving further, following up in this paper, Section-2 talks about some background reads and findings we employed before tackling the problem. Section-3 discusses our approach in detail. Section-4 presents some of the results we got on various sets of images. On the ending note, Section-5 discusses the future possibilities and extensions for the assignment, in particular, provided with more time. It also addresses some of the challenges we faced during its development.

II. Background Read

We came across various tools and techniques during the process, took note of many and implemented them as and when needed. Class notes and lecture slides were our first source of information in addition to the vast online repositories on the *Internet*. Cutting back to the chase, following are some of the topics/areas/parts of implementation which we went through in detail:

A. GUI

With no prior hands on experience we naively started looking for options in developing and ended up using the GUI feature in MATLAB for our final implementation. The *GUIDE* feature was an easy-to-use interface that enabled us to build the GUI using its drag-and-drop feature along with the auto-generated basic level code to start with. Documentation and online tutorials proved consequential in further development process.

B. Color Spaces

With just the basic idea of some color models, we delve into the details of the two color models, namely *RGB* and *HSV*. RGB model is represented in a 3D space with each color forming one of the orthogonal basis. Each tuple in the space represents the amount/intensity of each color at that point. While on the other hand, *HSV* is represented in cylindrical

coordinates in 3D space with axes: {Hue, Saturation, Value }. The Value channel lies in the range of $[0, 1]$ and serves as the grayscale equivalent of an intensity map for each color. Saturation on the other hand indicates various shades of a particular color depending upon the amount of blackish tint it inherits. Hue forms the main color component in the space. in our case, RGB color model is converted to HSV color model prior to applying any of the techniques on colored images.

C. FFT Radix-2 DIT Algorithm

Naive 2D DFT algorithm takes a lot of time to run. It has a worst case running time complexity of $\mathcal{O}(M^2N + MN^2)$. This takes a huge amount of time to compute in the forward direction. For instance, in our case, it took approximately 36 seconds to compute the DFT of a 256x256 sized image as opposed to the FFT Radix-2 algorithm which took not more than 1(0.8) second to compute. Following this up we turned our attention to the *Radix-2 Decimation In Time Algorithm* which worked out really well in practice due to its impressive running time complexity of $\mathcal{O}(MN \log N + NM \log M)$; where [M,N] is the size of the image. It uses the *Divide and Conquer* approach in its implementation. The algorithm to the same follows up in the approach section.

D. Adaptive Thresholding

Adaptive thresholding is an image segmentation algorithm that appears quite resistent to varying lighting conditions. This is a technique used in image processing to segregate images into foreground and background efficiently, without affecting the structure of the objects in the image. We came across various techniques to do so, namely, *Mean thresholding, Median thresholding, Gaussian thresholding, Otsu binarization*. But due to lack of time, we successfully implemented only the *Mean thresholding*. The algorithm follows up in the approach section.

E. Frequency Domain Filtering

This involves filtering of images in the frequency domain through the application of masks over the magnitude response of the 2D DFT. For instance, Smoothening (Blurring) is achieved in the frequency domain by high frequency attenuation i.e. Low pass filtering. While Sharp edges or sharpened image is achieved by low frequency attenuation also known as High Pass filtering. Our approach to this problem is discussed in the next section.

F. Intensity Transformations

This forms an integral part of all the Image Enhancement techniques in Image Processing. Each image is seen as a function of intensity over spacial coordinates as $f : X, Y \rightarrow [0, 255]$. The transformation function $T_f : [0, 255] \rightarrow [0, 255]$ is a mapping from a range of intensities to itself.

$$f_T[x, y] = T(f[x, y]) \qquad \forall x, y \tag{1}$$

; x, y are the spatial coordinates of the image in 2D space and f_T is the enhanced image.

III. Approach

This section focuses on the tools and techniques we applied during the development of various features in the GUI. Each of the following subsection discusses our approach and algorithms deployed in implementation of the features in detail:

A. Colored Images

For colored images, the intensity map obtained from the $Value(V)$ channel of the image, in HSV Color Model, is mapped to $[0, 255]$ for the application of a common algorithm on the similar lines of a grayscale intensity map. Post application of the transformation function the final (enhanced) intensity map is then incorporated along with the other 2 layers, namely $Saturation(S)$ and $Hue(H)$ in HSV space and finally, converted back to the *RGB* color model for further processing, display and storage.

B. GUI

The GUI is made over MATLAB's GUI feature. It includes two axes for comparison between the original input image and the current edited image, which is placed on the top of the *State-Stack* of the GUI, which in turn forms the integral part in the functioning of the UNDO feature, which can be called multiple times as opposed to un-doing just the last change. The GUI is designed to handle the import and saving of images from and to the local disk, respectively. It also supports the *UNDO* and *UNDO-all* (in our case, the *Reset*) feature too, where the former can be applied a multiple number of times till you reach the first operation applied over the imported image. Apart from these, the primary features implemented in the background are interfaced properly using push buttons and sliders to ensure the best exhibition of output images in real time. In addition to these features, the GUI is also able to handle both Colored and Grayscale images effectively. The screen shot the of the GUI showing all the features is as shown below:

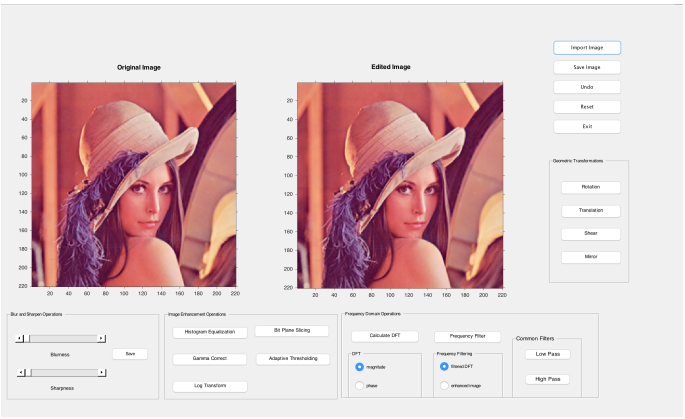


Fig. 1: Layout of GUI

C. Histogram Equalization

This is a contrast enhancement technique which increases the global contrast of images by ensuring uniform distributivity

of pixels of the image over the range of intensities. This is an intensity transformation enhancement technique where the transformation function itself comes out to be the *CDF* of the image over the range of intensities. Given an intensity map (image) X, pdf is calculated as

$$p_x(i) = p(x = i) = \frac{n_i}{n}, \quad 0 \leq i < L \tag{2}$$

where, L being the total number of intensity levels in the image (typically 256), n_i being the number of pixels with intensity i,n being the total number of pixels in the image, and $p_x(i)$ being in fact the image's histogram for pixel value i, normalized to [0,1]. Besides this, cdf is calculated as

$$cdf_x(i) = \sum_{j=0}^{i} p_x(j) \tag{3}$$

which is also the image's accumulated normalized histogram. The transformation function $T_x(i)$ is then given as:

$$T_x(i) = \lfloor (L-1) * cdf_x(i) \rfloor \tag{4}$$

This Transformation ensures that the pixels spread out uniformly over the range of intensities $i.e.[0, L-1]$.

D. Log Transform

This is an intensity transformation which maps dark pixels in an image to higher intensities as per the logarithmic function behaviour. Given an intensity map I, the transformation function $T_x(i)$ is calculated as:

$$T_x(I(i,j)) = c * log(1 + I(i,j)) \tag{5}$$

where, c is the normalization factor and $I(i,j)$ is the intensity value of the pixel at the position (i,j). Also note that 1 is added in the transformation to avoid cases of $log(0)$.

E. Gamma Correction

Also known as the *Power-Law Transform* this is also an intensity transformation technique which is applied as follows. Given an intensity map I, it is first normalized to ensure that the intensities lie in the range $[0, 1]$. the transformation function $T_x(i)$ is then calculated as:

$$T_x(I(i,j)) = c * I(i,j)^{\gamma} \tag{6}$$

where, c is the normalization factor, $I(i,j)$ is the intensity value of the pixel at the position (i,j) and γ is the power law parameter.

F. Adaptive Thresholding

The algorithm is as follows

- A window of size nxn (n taken as input by the user) is decided upon
- Using the sliding window method in 2D, at each pixel a threshold is calculated by taking the mean of the neighborhood intensities
- Based on the calculated thresholds each pixel is binarized, accordingly

In our case, we basically implemented a Mean thresholding based algorithm where the threshold at each pixel is calculated by taking mean of the intensities of the window around it. The input other than image and window size(n), is a constant parameter C to maintain the contrast and noise level in the thresholded image.

G. Blurring

In our implementation of a blurring an image, we applied a Gaussian kernel using the window sliding method, which is equivalent of taking a convolution in spatial domain (due to the symmetry of the filter, otherwise just correlation). The algorithm is as follows:

- Size of the Gaussian kernel (n, an odd integer) and its standard deviation (σ) are taken as inputs
- Using the sliding window method, at each pixel the gaussian weighted average of neighboring intensities is taken as the new intensity value.

Running time complexity of this method is $\mathcal{O}(MNn^2)$, where $[M, N]$ is the size of the intensity map and nxn is the size of the Gaussian kernel in 2D.

H. Sharpening

This feature intends to sharpen the input image by protruding the edges more than the normal level. The algorithm of this feature is as follows:

- Three inputs are needed: size of the blurring kernel n (set equal to 21), std dev of the gaussian blur kernel σ (set equal to 2) and the amount of unsharp masking which is controlled by the parameter k (taken as input from the user)
- The input image is first blurred using Gaussian kernel in spatial domain using the previous state approach
- Post computation of the blurred image, a sharpening mask is computed by subtracting the blur image from the original image
- Output image is computed by taking a weighted average of the sharpening mask and the original image as shown below
- Due to the decrease in the contrast of the final image, we also perform a histogram equalization on the image before displaying/storing

Given an image I and its blurred image B, the sharpened image, S is given by

$$\begin{aligned} Mask &= OriginalImage(I) - BlurredImage(B) \\ S &= (1-k) * OriginalImage(I) + k * Mask \end{aligned} \tag{7}$$

where $k \epsilon [0, 1]$ is an input parameter that controls the amount of sharpening.

I. Geometric Transformation

This feature involves a total of four sub features which result in geometric (affine) transformation of the input image in one or the other way.

1) Translation: The two inputs other than the image gives the extent of translation (in pixels) in each direction. The coordinates of each pixel is then shifted according to the inputs given, using the below stated formula. The pixels which map to the coordinates out of the range of the original image's size are discarded.

$$\begin{aligned} x' &= x + x_0 \\ y' &= y + y_0 \end{aligned} \tag{8}$$

2) Rotation: In addition to the image, the function takes as input the angle θ by which the image is to be rotated about the center of the image matrix. The pixels which map to the coordinates out of the range of the original image's size are discarded.

$$\begin{aligned} x' &= cos(\theta) * x - sin(\theta) * y \\ y' &= sin(\theta) * x + cos(\theta) * y \end{aligned} \tag{9}$$

3) Shear: The input alpha and beta represents the factor of shear in x and y direction respectively. The center is first shifted to the center and then the simple equations of shear is applied to each coordinate of the input image.

$$\begin{aligned} x' &= x + \alpha * y \\ y' &= y + \beta * x \end{aligned} \tag{10}$$

This transformation gives output as a sheared image in both x and y direction. Again the pixels mapping to the coordinates out of the range of the original image's size are discarded.

4) Flip Image: This feature flips the image in horizontal direction i.e. the output image is the mirror image of the input image about the vertical axis passing through the center of the image. Each y coordinate is flipped while the x coordinate remains same and the transformation is given by equations:

$$\begin{aligned} x' &= x \\ y' &= N - y \end{aligned} \tag{11}$$

where N is the length of image in y axis.

J. Bit Plane Slicing

This feature takes input an image and an integer in the range [1,8]. The integer value indicates the plane number of the spliced image which is to be displayed on the axes. All the bit planes are segregated from the image, considering plane number 1 as the MSB plane and 8 as the LSB plane. The output plane is binarized to the levels 0 and 255 in the intensity domain for display.

K. 2D-DFT

The algorithm *Radix-2 Decimation in Time FFT* is as follows:

- (Only for colored images) Extract the intensity map of the image by converting it into the *HSV* space.
- After extracting the intensity map, the dimensions are rounded off to the nearest power of 2 (hence the name, Radix-2) by padding the image with zeros, only on one side of each dimension.
- Due to the variable separability property of 2D-DFT in each dimension, FFT on the image intensity map is performed in 2 rounds, once in each dimension.
- While iterating over one dimension, 2D FFT is calculated by performing multiple 1D FFTs on the other dimension

This variable separability property of DFT is shown by the following set of equations:

$$\begin{aligned} F[u,v] &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f[m,n] * e^{-i*2\pi*(um/M+vn/N)} \\ F[u,v] &= \sum_{m=0}^{M-1}(\sum_{n=0}^{N-1} f[m,n] * e^{-i*2\pi*vn/N}) * e^{-i*2\pi*um/M} \end{aligned} \tag{12}$$

Clearly, the equations (12) can also be written as:

$$\begin{aligned} P[m,v] &= \sum_{n=0}^{N-1} f[m,n] * e^{-i*2\pi*vn/N} \\ F[u,v] &= \sum_{m=0}^{M-1} P[m,v] * e^{-i*2\pi*um/N} \end{aligned} \tag{13}$$

It is clear from the equations (13) that $F[u,v]$ can be calculated in 2 rounds

1) Calculate $P[m,v]$ by performing 1D FFT in the second dimension while iterating over the first dimension for every point (m,v) where $m\epsilon[0,M-1]$
2) Now iterating over the second dimension, perform 1D FFT on the first to calculate the final $F[u,v]$ at each point $[u,v]$

Hence, the running time complexity of the algorithm comes out to be $\mathcal{O}(MNlog(MN))$ as opposed to the naive 2D DFT with complexity $\mathcal{O}(MN(M+N))$.

The final centralised version of the 2D DFT is then stored for further display and processing.

L. Filter Domain Masking

Apart from the import of frequency masks from the local disk, we also incorporated the GUI with some common filter masks for easy and instant evaluation. To ensure proper application of masks, they should be centralized to the center of the image and must have dimensions same as that of the DFT of the image (in power of 2). After the application of the masks Inverse DFT is calculated using the in-built *ifft2* function in MATLAB to ensure fast reconstruction.

Common filters supported in the GUI are circular *Low-Pass* and *High Pass filters*, where radius is given as an input to the GUI in each case.

IV. Testing and Results

In order to test all these features, it is important to test them on the images where the effect will be most significant. This section demonstrates each feature of the image by the means of test images and also describes the reason to select a particular test image.

A. *Blurring*

This feature can be applied to any sharp image and does not have a specific purpose. Different extent of Blurring is shown below for both gray scale and color images. Extent of blurring will increase progressively.

Fig. 2: Increasing Blurrness in gray scale image

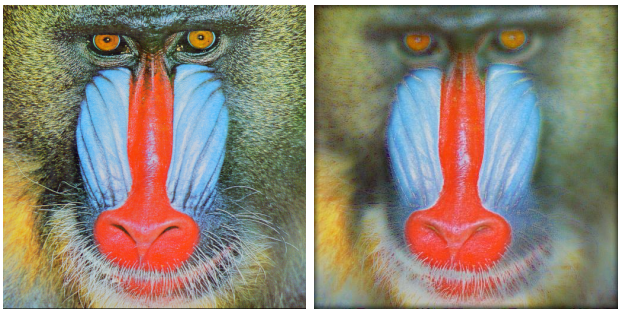


Fig. 3: Blurring a colored Image

B. *Sharpening*

Sharpening is usually used to highlight edges and fine details of an image. Any image with some edges is a goof test image for sharpening. Different extent of sharpening is shown below for both gray scale and color images. This image has lot of fine edges which gets sharpen prominently.

Fig. 4: Sharpening in colored Image

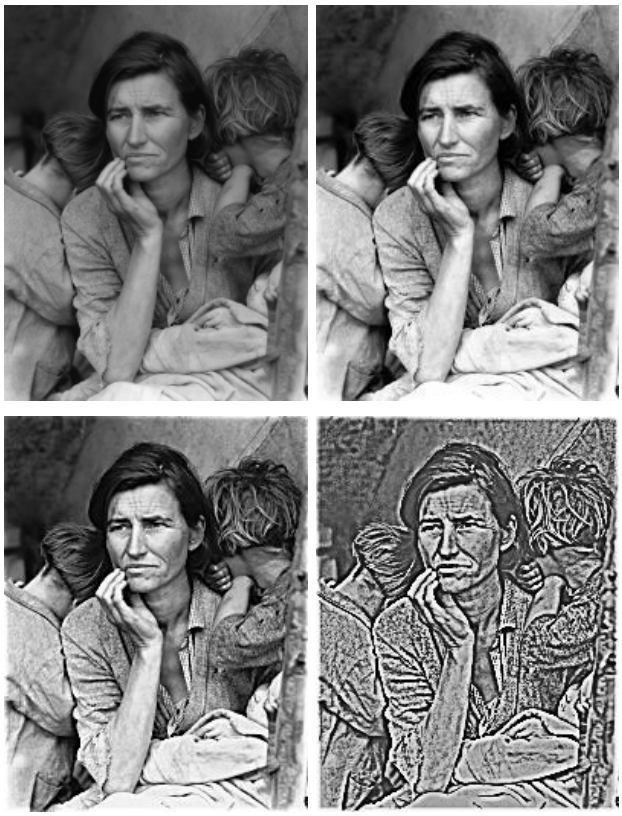


Fig. 5: Increased Sharpness in gray scale image

C. *Histogram Equalization*

Histogram equalization provides better global contrast. To test this feature we need an image which has most of the details in either darker range or lighter range of pixels. The images here have a lot of parts which are not visible in darker levels. So these are good images to test this feature on.

Fig. 6: Original Image

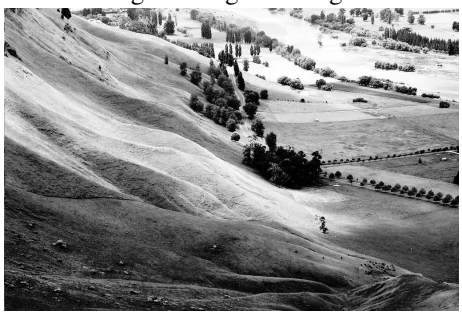


Fig. 7: Histogram Equalized Image

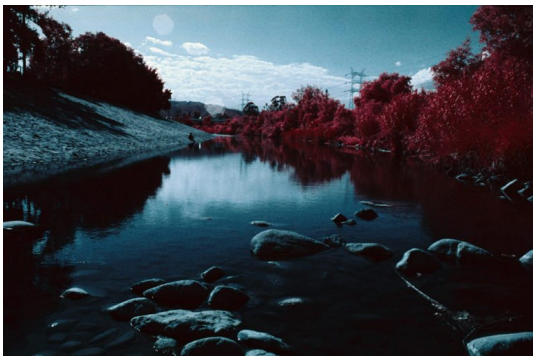


Fig. 8: Original Image

Fig. 9: Histogram Equalized Image

D. Log Transform

Log transform expands the low intensity levels and compresses the higher intensity levels. The darker values are enhanced as compared to lower values. Log Transform is best seen on gray scale images which has both darker and lighter regions.

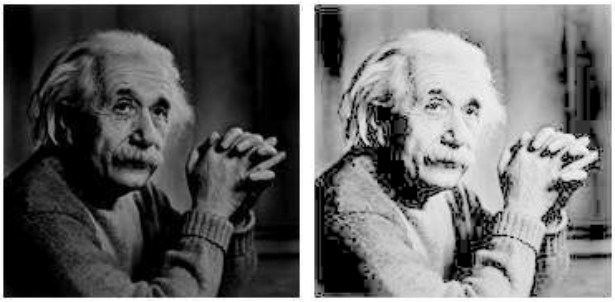


Fig. 10: Original image vs Log transformed image

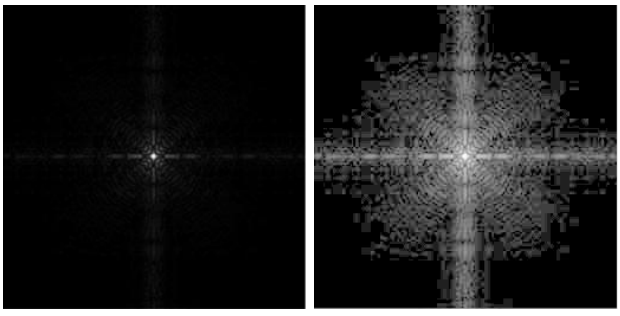


Fig. 11: Log Transformation of DFT

E. Gamma Correction

Gamma correction enhances the contrast of image depending on the value of γ. If $\gamma > 1$ then the darker region of the image gets compressed and brighter side gets enhanced while if $\gamma < 1$, then vice versa. Any gray scale image will show clearly show the effect of Gamma Correction.

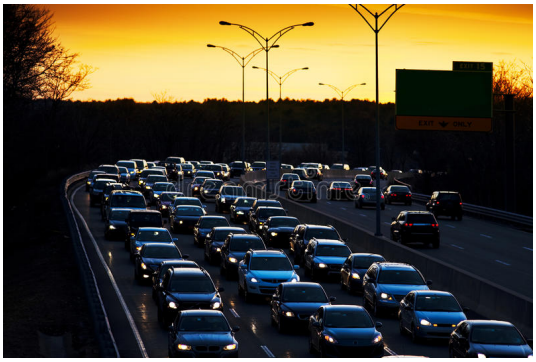


Fig. 12: Original Image

Fig. 13: Gamma Value = 0.45

Fig. 14: Gamma Value = 2

F. Adaptive Thresholding

This feature segments an image in small windows by setting all pixels whose intensity values are above a threshold to a foreground value and all the remaining pixels to a back ground value. Types of images used in this feature must have gradients of intensity. The text in the page is clearly visible after Adaptive Thresholding.

Fig. 15: Original Image

Fig. 16: Output Image

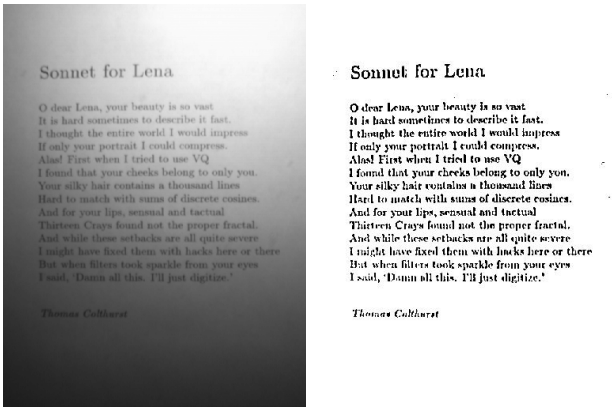
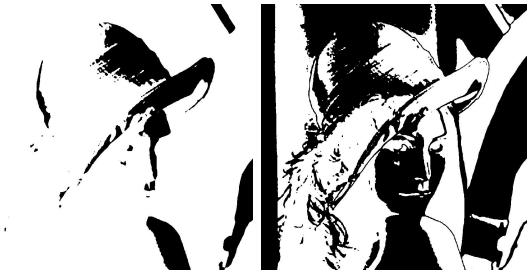


Fig. 17: Adaptive Thresholding result

G. Bit Plane Slicing

This feature highlights the contribution made to total image appearance by specific bits. Any color image or gray scale image which has significant shades of pixels can be used to test this feature.



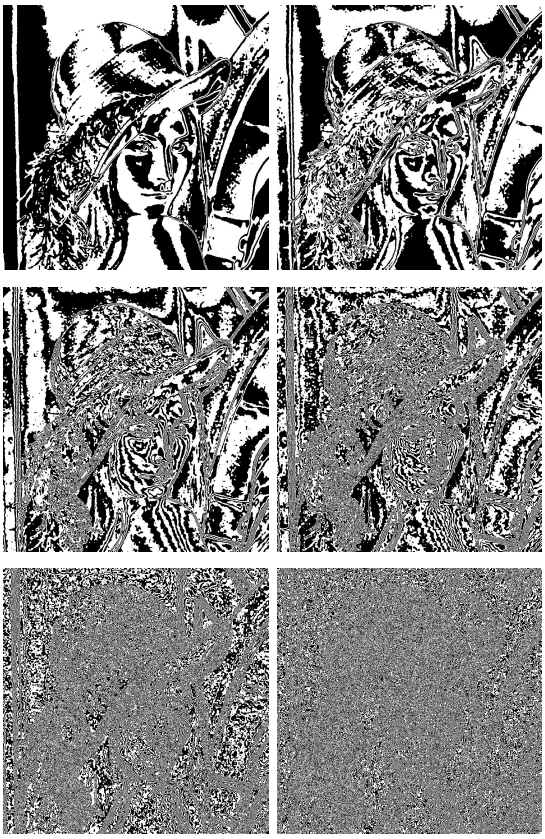


Fig. 18: Plane1(MSB) to Plane8(LSB)

H. Geometric Transformations

These transformations can be tested on any image. It is useful if the image is in wrong orientation.The four Transformations are shown on the image.

Fig. 19: Original Image

1) Rotation: The output of the above image on rotation of 30 degrees and 80 degrees is shown below:

Fig. 20: Rotation

2) Translation: Here the part of the image is cropped by shifting the image. This feature also can be used to crop the image. The output of the image on translation of 50 pixels in x direction and 100 pixels in y direction is shown below:

Fig. 21: Translated Image

3) Shear: In this feature, each pixel of the image is displaced in fixed direction by an amount proportional to its signed distance from a line that is parallel to that direction. The output of the image after shearing by a factor of 0.3 in x direction and by a factor of 0.6 in y direction is shown below:

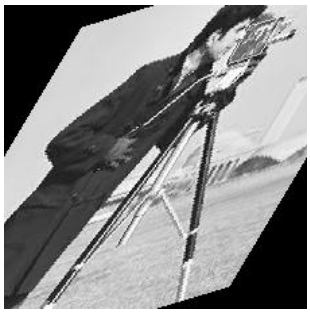


Fig. 22: Sheared Image

4) Flip: In this feature the image is flipped about the vertical axis, i.e. the mirror image is taken as output about the vertical axis. The output is shown below:

Fig. 23: Mirrored Image

I. 2D-DFT

This feature decomposes an image into its sine and cosine components. Magnitude and Phase response is best seen in images where there are sharp changes in intensity. The Magnitude response is Log transformed to have a better view of frequencies.

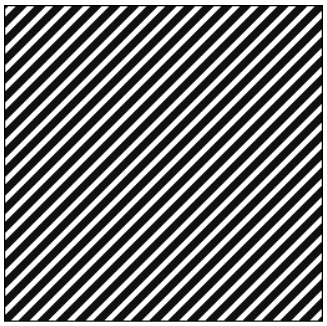


Fig. 24: Original Image

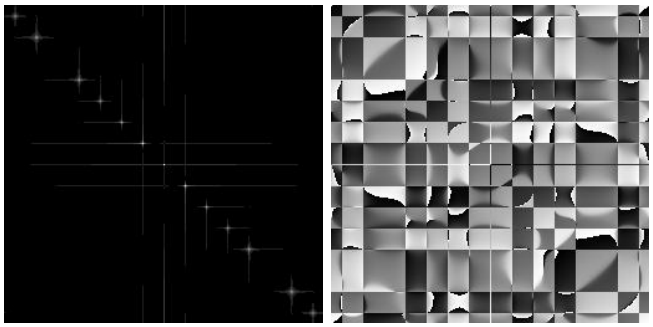


Fig. 25: Magnitude and Phase Response of Fig. 24

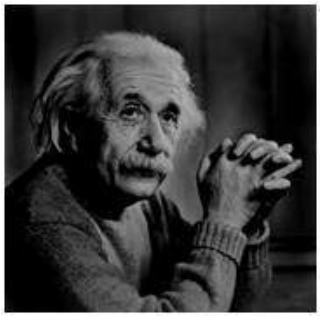


Fig. 26: Original Image

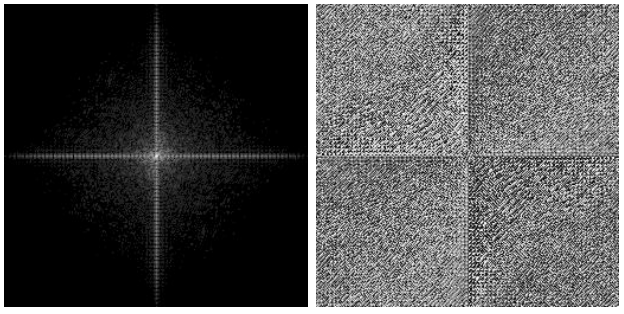


Fig. 27: Magnitude and Phase Response of Fig. 26

J. Frequency Domain Filtering

This section includes Low Pass and High Pass filtering of Images. Low Pass filter makes the images smooth or blur. And High pass filter shows sharpens the image and shows the dark edges. This feature is applied to two images as below.

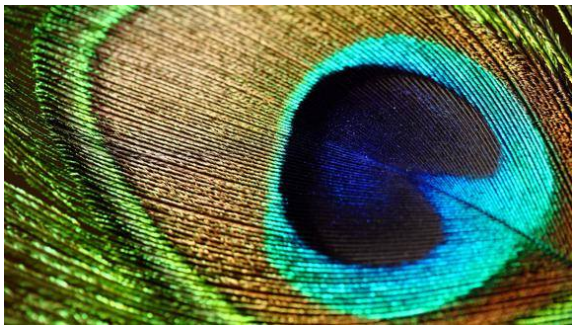


Fig. 28: Original Image

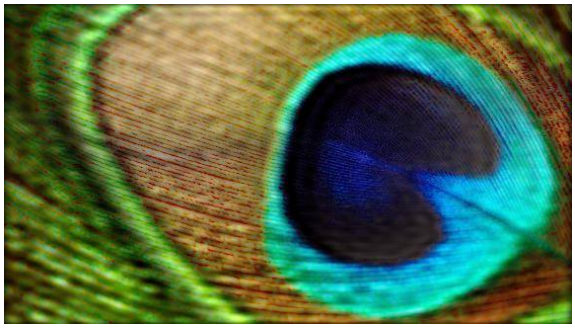


Fig. 29: Low Pass Filter Output

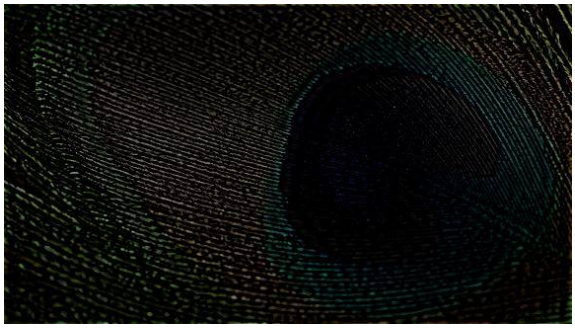


Fig. 30: High Pass Filter Output

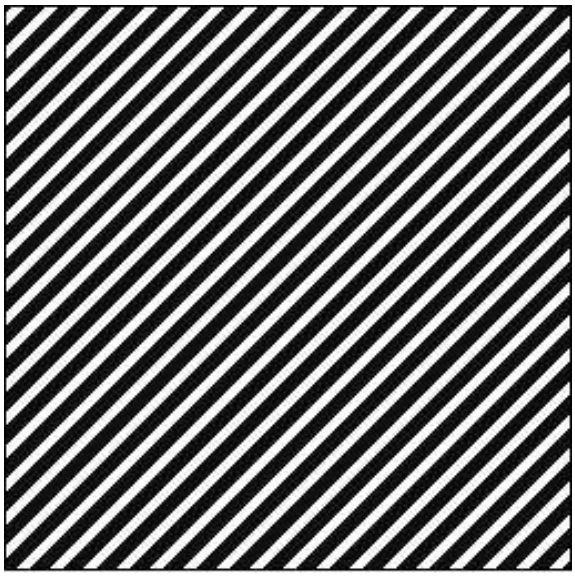


Fig. 31: Original Image

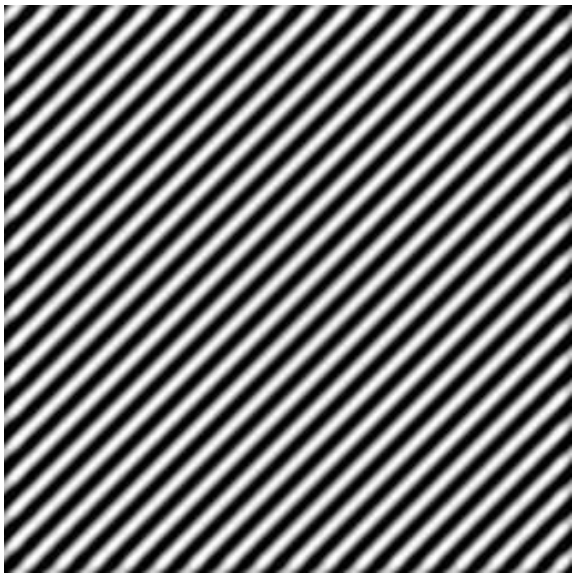


Fig. 32: Low Pass Filter Output(Slightly Blurred)

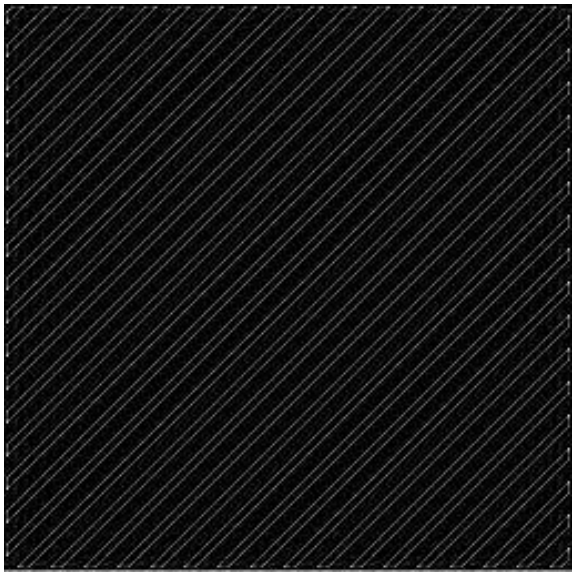


Fig. 33: High Pass Filter Output

V. Discussions

The challenges faced in the implementation of GUI are:

- The major challenge was to implement GUI as we didn't have any prior hands on experience with GUI development in MATLAB.
- Direct DFT implementation took long to calculate Magnitude and Phase Response. So implementing 2D-DFT by FFT algorithm was a difficult task to perform as the number of pixels in images needs to be in power of 2.
- Implementation of UNDO feature was a tough task. It required formation of a linear stack which stores all the stages of edits and gets updated by the latest one.
- Sharpening lead to decrease in contrast of images which was then resolved by performing Histogram Equalization on sharped images.

The improvements and features that can be added further in the GUI are:

- There is a scope to introduce many other frequency domain features.
- If we had more time, we would have implemented CLAHE which is advancement of AHE(Adaptive Histogram Equalization).
- There are various of other frequency domain filters which could be implemented in future.
- The frequency masking feature in the GUI could be extended by taking FFT of output post frequency masking.

VI. Reference

- Class Notes and Lecture Slides
- Adaptive Thresholding
- Coordinate Transform
- Image Sharpening
- Frequency Filtering
- FFT Implementation
- General Resource - Wikipedia