SPL-1 Project Report

# Basic Image Processing Tool


**Submitted by:**

***Hasnain Sheikh***

**BSSE Roll No: BSSE 1408**

**BSSE Session: 2021-2022**

**Supervised by:**

***Dr. Emon Kumar Dey***

*Designation: Associate Professor*

*Institute Of Information Technology*


Dr. Emon Kumar Dey
Supervisor


**Institute of Information Technology**

**University of Dhaka**


[17-12-2023]

# Table of Contents

# 1. Introduction

Basic Image Processing Tools is an image processing software that consists of a number of image processing tools. The tools in this software are helpful in editing an image very efficiently. The software also gives option to save the processed images to use the image for different purposes of the user. User can input a bitmap image to the software and use the tools on it. I tried to implement nine image processing tools which are more often used in image processing related fields. The tools are: Edge Detection, Greyscale, Negation, Smoothing, Brightening, Blurring, Angle Calculation, Histogram and Sharpening. The results obtained are perfectly reusable and can also be re-used in the software to edit the edited image again.

This software is a small-scale implementation of a large image processing software that we often use in our regular life. Many more tools can be added to the software to upscale the usage of software.

## 1.1 Background Study

To implement this project, some prior study was necessary –

### 1.1.1 Digital Image Pixels

A pixel is generally thought of as the smallest single component of a digital image. The more pixels used to represent an image, the closer the result can resemble the original. The number of pixels in an image is sometimes called the resolution. The number of distinct colors that can be represented by a pixel depends on the number of bits per pixel.

### 1.1.2 Bitmap File Format

I had to study about the bitmap file format before using it in my software. It was one of the most challenging partof my project since I had to study the file format and store the data of a bitmap image in separate reusable variables. The bitmap image file consists of fixed-size structures (headers) as well as variable-sized structures appearing in a predetermined sequence.

The bitmap file header consists of the following parts: -

| **Structure Name** | **Size** | **Purpose** |
|---|---|---|
| Bitmap File Header | 14 Bytes | To store general information about Bitmap image file |
| DIB Header | Fixed size (7 different version exists) | To store details information about Bitmap image and define the pixel format |
| Extra bit mask | 3 or 4 words<br>(12 or 16 bytes) | To define the pixel format |
| Color Table | Variable size | To define colors used by bitmap image data |
| Gap1 | Variable size | Structure alignment |
| Pixel array | Variable size | To define the actual value of the pixels |
| Gap2 | Variable size | Structure alignment |
| ICC color profile | Variable size | To define the color profile for color management |

### 1.1.3 Greyscale

Grey scaling was one of the prominent features that I had to implement since many other tools in my project use grey scaled image as a part of the process. grayscale image is one in which the value of each pixel is a single sample representing only an amount of light; that is, it carries only intensity information. Grayscale images, a kind of black-and-white or grey monochrome, are composed exclusively of shades of grey. The contrast ranges from black at the weakest intensity to white at the strongest. In the 24-bit color images, we can make all the pixels look like grey scaled by making the RGB values in a pixel, equal. The average of the RGB values of a pixel is then, the new value for all the RGB values of that pixels in a grey scaled image.

### 1.1.4 Sobel's Operator

Sobel's Operator, also known as Sobel – Feldman operator is a matrix that is used in different image processing algorithms, mostly in the edge detection algorithms to create a well outlined image with highlighted edges. The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives – one for horizontal changes, and one for vertical. If we define A as the source image, and Gx and Gy are two images which at each point contain the horizontal and vertical approximations respectively, the computations are as follows:

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Fig. 1.1: Horizontal & Vertical Gradient of the given Image

The x-coordinate is defined here as increasing in the "right"-direction, and the y coordinate is defined as increasing in the "down"-direction. At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

Fig. 1.2: Resultant Gradient Approximation

### 1.1.5 Median Filter

The median filter is a commonly used filter that is used to implement different image processing algorithms related to image smoothing. The filter is the value of the median of a sorted matrix at any position of an image pixel array. The median filter slides over the image using a 3 by 3 matrix and changes the values of the pixels with median of matrix at that particular pixel.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Fig. 1.3: Median Filter

### 1.1.6 Unsharp Masking

Unsharp masking (USM) is an image sharpening technique, often available in digital image processing software. Its name derives from the fact that the technique uses a blurred, or "unsharp", negative image to create a mask of the original image. The unsharp mask is then combined with the original positive image, creating an image that is less blurry than the original. Here I have used the following mask matrix to obtain a blurry image of the given image and then compare the original and the blurred image to get a sharpened image as a result:

Sharpen filter $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

Fig. 1.4: Unsharp Mask Filter Matrix

## 2. Challenges

There are a number of challenges that I needed to face during the implementation of software. The process was both confusing and overwhelming. Some of the challenges that I faced and solved are: -

➢ Working with a project in C with multiple files and functions
➢ Handling delicate data for pixels and headers
➢ Implementing multiple loops with exact limits
➢ Working with the matrices in programming
➢ Handling the delicate errors that were often caused by type conversion
➢ Handling the recursive and inclusive calls of different functions
➢ Reading and writing the image data from the files

➢ Image Learning –
  - Bitmap Images and its header info
  - Pixel data storage and RGB values information
  - Gradients and intensity of the image based on the pixel values

➢ Image processing algorithms –
  - Sobel's edge detection algorithm
  - Unsharp Masking
  - Brightness Algorithm
  - Median Filter Algorithm
  - Gaussian Blur Algorithm

## 3. Objectives

By the end of this project, I will be able to:-

➢ Deliver a fully functional software with interactive user interface

➢ Edit any 24-bit bitmap image of any height and width

➢ Apply nine image processing algorithms to an image

➢ Save an edited image for future purposes

➢ Read and write any type of files in C/C++ and store the data in separate variables for further usage in the future

➢ Work with an image to the pixel level and manipulate it

## 4. Scopes

The scopes of my project are: -

➢ Creating an image processing system that takes an image as input and outputs an edited image

➢ Implementing the following image processing algorithms: -

- Edge Detection
- Grey Scaling
- Smoothing
- Negation
- Histogram
- Angle calculation
- Brightening
- Blurring
- Sharpening

➢ Using C/C++ as a programming language to implement the given algorithms and processes

➢ Using internal C libraries to complete the project (Avoid using any external library)

# 5. Project Description

There are basically two parts in my project –

- ➢ Read and Write Image
- ➢ Process the image using different algorithms

## 5.1 Read and Write Image

The primary goal of the software is to read an image from the user input path and output the edited image to another user defined path. The read function of the software takes the file path as an user input. It then reads the bitmap image file header into a structure for future use. Then it reads the pixels of the bitmap image files in a pixel structure where each pixel contains three 8bit character values i.e., RGB values. The processing is done with the pixel array 8 extracted from reading the image file. After completion, the previously recorded bitmap header info is written into another new file which will be the output result image. The pixel array is then written into the output image file. The output image file will be stored in the source code folder if a specific path is not given by the user. The output file can also be used as an input file in the software again, if necessary.

## 5.2 Processing the Image

The input image can be processed according to the choice of user with the help of eight functions provided in the user menu. The description of the functions is given below: -

### 5.2.1 Grey Scaling

Grey scaling is done by making each pixel between the values 0 -255. Since our 24-bit image contains three values in pixels. I took the average of all the red, green and blue values and

replace all the values in that pixel with the average value. In this way, a pixel will of a single value between 0 -255. Hence, the output image will be grey scaled properly.

### 5.2.2 Edge Detection

The image pixels are distributed throughout the image in different intensities. The edges are created where an area of larger intensity merges with an area of lighter intensity. Thus, our primary goal in this part is to find this type part and make the merged pixels lighter and set the other pixels off. I implemented the following algorithm with the help of Sobel's operator, a 3 by 3 matrix that helps in determining the derivatives vertically and horizontally over a pixel. Then, I calculate the resultant derivative. If the result is greater than a certain threshold then pixel value is set to 255 and if the value is below the threshold, that pixel is set to 0. In this way, only the pixels that has gradients approximately near to the edges will only be lightened and other area of the image will be darkened. A grey scaled image is necessary for the edge detection process, so I used the already implemented grey scaling function to grey scale a color image and then start detecting the edges.

### 5.2.3 Smoothing

Smoothing an image can be done with many filtering algorithms. In this process, the pixels values are changed according to the neighboring pixels to avoid any extreme valued pixel, therefore creating a smoothly distributed pixels through the image. I have used median filtering algorithm to smoothen the image. Algorithm takes the median value of the pixel and its nearby pixels and change the value of that pixel to the median value. It requires the pixel values to be sorted to find the appropriate median. Since a pixel contains three internal values, the algorithm is applied to each of red, green and blue values of the pixel.

### 5.2.4 Brightening

Image brightening is a comparatively easy process in which the user inputted threshold is added to all the pixel values to brighten it to a certain level. If the threshold + current value exceeds 255, then the algorithm sets that pixel value to 255. Similarly, if user inputs negative value to reduce the brightness of the image and current value – threshold becomes negative, it is rounded to 0, making a darkened image.

### 5.2.5 Blurring

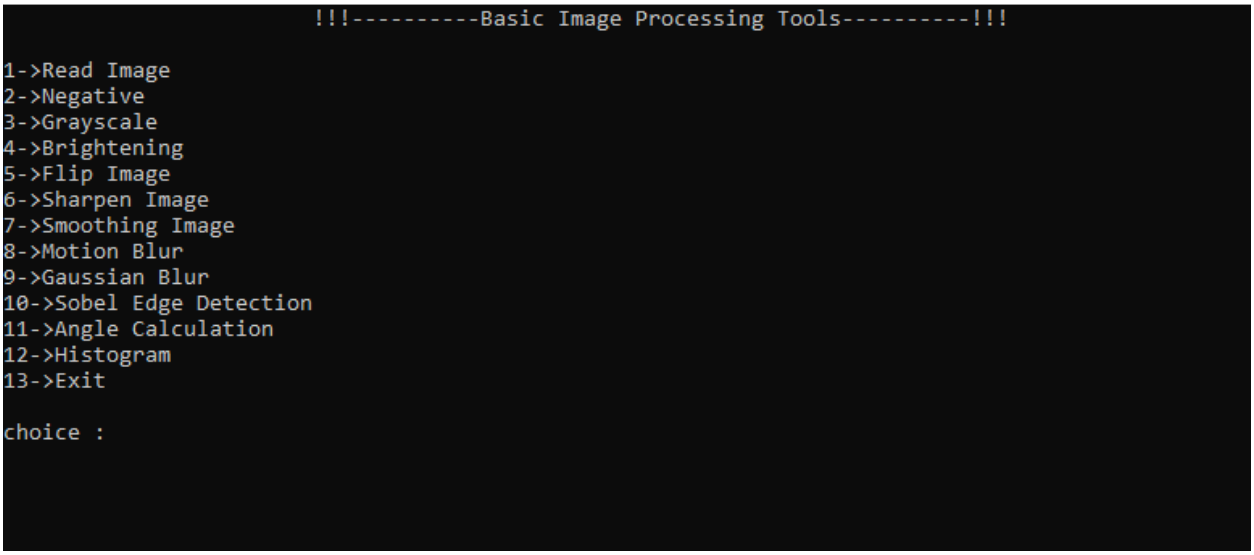
Blurring uses an averaging algorithm where I use a 3 by 3 matrix through the pixel array of the image and set the central pixel value to the average value of that matrix. This makes all the pixels extremely smoothened with respect to the neighboring pixels. This process done three times on the three values of a pixel in our input image.

### 5.2.6 Sharpening

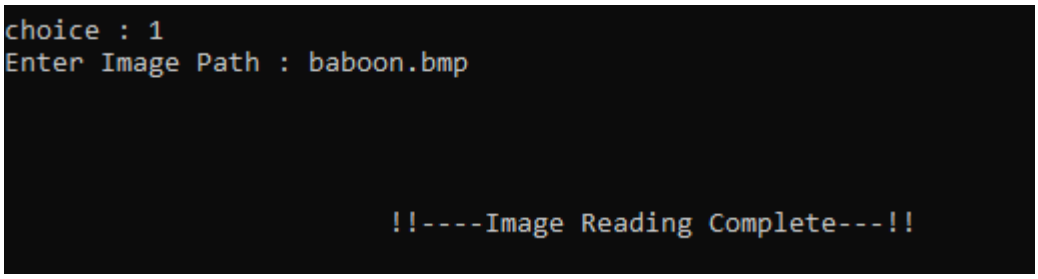
Image sharpening is the process of making an image less blurry or sharpening the edge pixels values to a certain point to make them look less blurry. The unsharp masking algorithm is applied to this project. This algorithm creates a blurry image of the original image and then merges the two images to get a less blurry original image. Sharpening an image usually makes an image clearer. In this project, a 3 by 3 unsharp mask matrix is used to create a sharpened image. If greater masks are used in this process, the more sharpened image can be created with the algorithm. The algorithm implementation is done by using the given 3 by 3 unsharp mask to multiply it with the pixel's matrix repeatedly at every pixel value.

# 6.User Manual

➢ The user will see the following interface after running the program




➢ User has to input path of a bmp file format image in the indicated place of command line. The initial preview of the image will be shown when enter is pressed.




➢ The user will be shown a small preview at the top of the menu, then he can enter the tools section without any error popup to start using the image processing tools.

## Edge Detect

On clicking the edge detect option, an edge detected output image will be shown with the option to save it.

Fig. 6.4: Edge Detected Image of Baboon

## Grey Scale

On clicking the grey scale option, a grey scaled image of the input image will popup as the desired output.

Fig. 6.5: Grey Scaled Image of Baboon

## Smoothing

On clicking the smoothing option, the input image will be smooth by using median filter. Output image will be saved as SmoothImage.bmp.

Fig. 6.6: Smoothened Image of Baboon

## Brightening

Brightening option gives the user option to input the value between -255 to +255. If a user chooses a positive value, the brightness of the image will increase by that value, if a user chooses a negative value, the brightness will decrease. The figure uses a brightness value of + 128

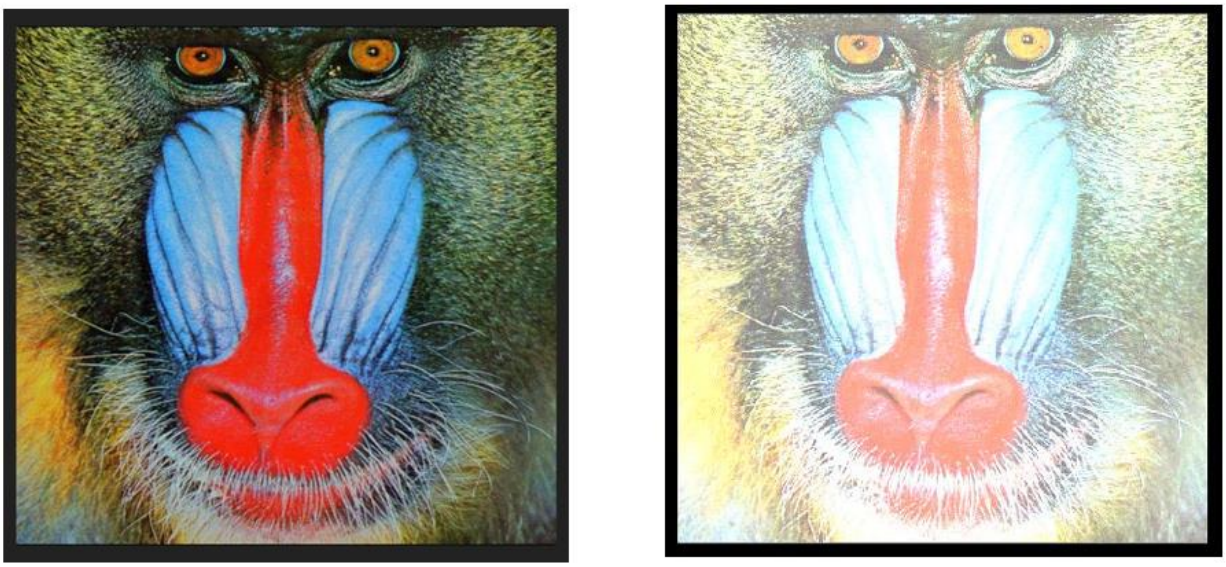


Fig. 6.7 : Brightened Image of Baboon

## Blurring

The image will be blurred by Gaussian Blur operation. I use (3x3) and (5x5) Gaussian filter for blurring the input image.

Fig. 6.8: Blurred Image of Baboon (Original vs Edited)

## Sharpening

An image is sharpened by using sharping filter. Here I use 4 types of sharping filters.

They are -

- ✓ Laplacian filter
- ✓ Strong Laplacian filter
- ✓ High Boost Filter
- ✓ Strong High Boost Filter

Fig. 6.11: Sharpened Image of Baboon

➢ After having the desired output, if user clicks the save option, he/she will be asked to name the file to save. Image will be saved when enter is pressed. The "Output image" of the main menu contains the most recently edited version of the input image.

➢ The manual option contains the user manual of the software.

➢ The "Exit" button shows a popup and closes the software.

## 7. Conclusion

This software is a very powerful, fast and easy to use software that comes with some popular image processing tools. There are many users who use the image processing tools for various purposes. Since this software is implemented by keeping its broader aspects in mind, it can be expanded in future with loads of extra tools and features. I have used my knowledge in reading and writing files, writing a code as a project, using custom headers to make this project successful. I have learned to use the algorithms that are used to implement the tools, the structure of a bitmap and manipulating an image at pixel levels. I hope to add more features to this software in the future. This software will also play a huge role in learning for my purpose of making a bigger image manipulation software.

# 8.References


1. https://www.tutorialspoint.com/cprogramming/c_file_io.htm
2. Digital Image Processing Basics - GeeksforGeeks
3. Introduction to image processing | Digital Image Processing (ut.ee)
4. *Fulton, Wayne (1997–2010). "A few scanning tips, Sharpening - Unsharp Mask". Scantips.com. Archived from the original on 2019-04-27. Retrieved 1 October 2019.*

5. Philippe Cattin (2012-04-24). "Image Restoration: Introduction to Signal and Image Processing". MIAC, University of Basel. Retrieved 11 October 2013.

6. Digital Image Processing 4th Edition by Rafael Gonzalez (Author), Richard Woods (Author)

7. Image Processing in C by Dwayne Philips

8. Campbell, Alastair. The Designer's Lexicon. ©2000 Chronicle, San Francisco.