Table of Contents

Index No : 140106T

Name : K.G.S. De Silva

ProjectRepository : https://github.com/GayanSandaruwan/SimIMMan

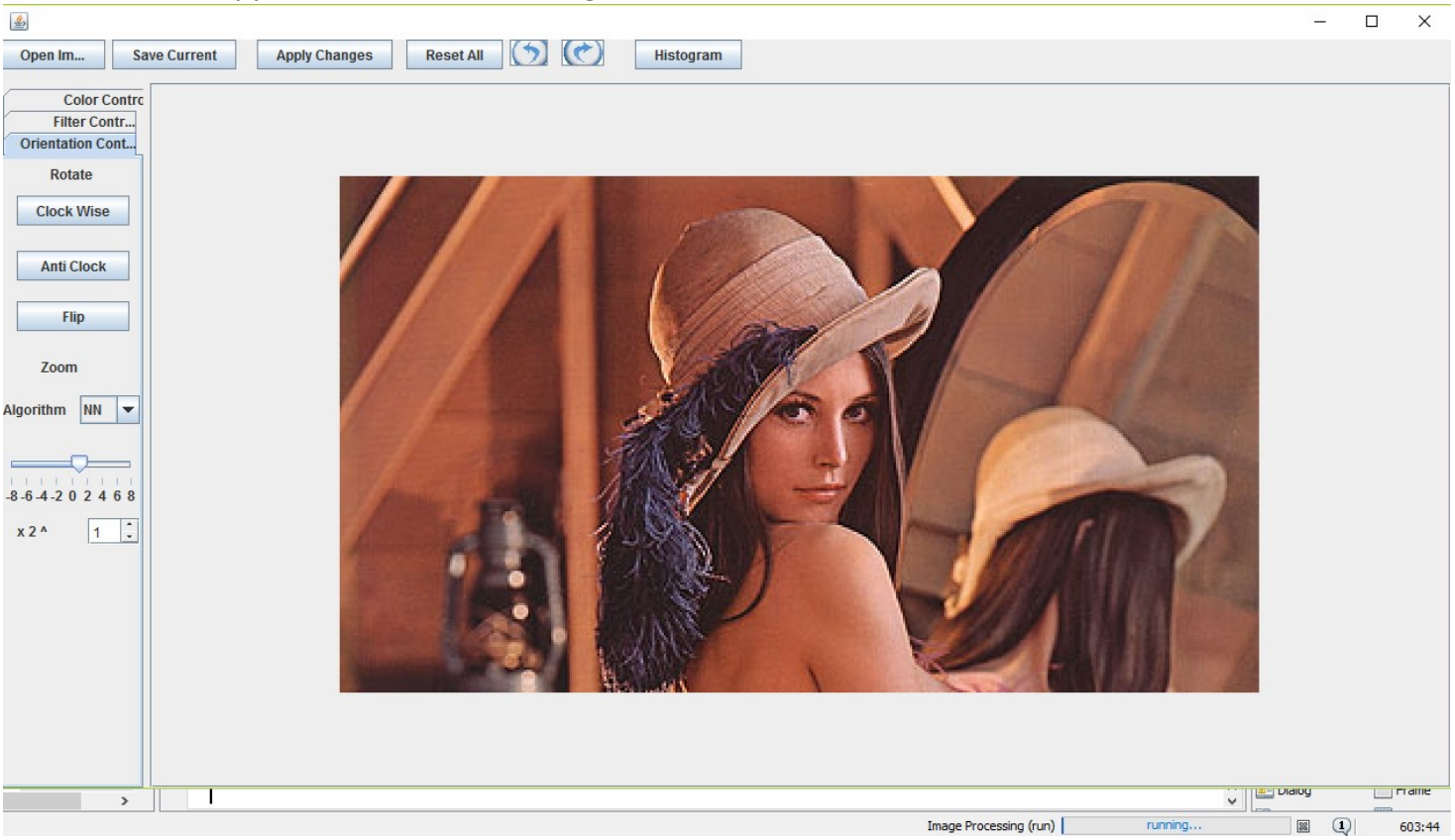
Introduction

Tool was developed using Java SE language. It is capable of performing simple image processing tasks as described in a below.

Images can be added to the program using Open button and the associated File Explorer UI with it (Or can use the Lena's imaged by default added to the tool for inspection).

In the tool bar menu there are

- ✓ Open Image : Open Image followed by a File Explorer.
- ✓ Save Current : Save The current Image in "JPEG" format in a user specified location
- ✓ Apply Changes : Apply the most recent Changes to the image.
- ✓ Reset All : Roll Back to the original version of the image
- ✓ Undo : Applied Changes.
- ✓ Redo: Can be used after a Undo. Forward the selection.
- ✓ Histogram : Draws Intensities of R,G,B values separately of the last changes applied version of the Image.



Options as described above.

How to Use

As displayed in the above Screen Shot a collapsed editing menu is used in the left-hand size giving more space for the image preview.

collapsed tab window in the left corner has three basic image manipulation categories as,

- ✓ Color Controls: Gray Scaling, Brightness Adjusting, Contrast Adjusting, Negative Image
- ✓ Filter Controls: Mean Filter, Median Filter, Gaussian Filter, Edge Detection (Sobel & Robert)
- ✓ Orientation Controls: Rotation (Clock wise, Anti-Clock wise, Flip), Resizing (Nearest Neighbor, Bi Linear Interpolation)

Can select each tab by clicking on the name of the respective tab.

For the Resizing, resizing algorithm needs to be selected from the drop-down menu.

- ✓ NN – Nearest Neighbor
- ✓ BI LI – Bi Linear.

None of the work except in the orientation tab is automatically saved. When you need to add more than one effect, press “Apply Changes” in the tool bar and continue work.

Applied changes can be undone using the Undo button, it is capable to traverse until the initial Image using Undo or Reset all will quickly load the initial image.

Histogram button will be displayed the Histogram of the image currently visible in the interface.

Implementation Approach

BufferedImage Digital image model is used which is built in with Java language.

it uses a 2D matrix of RGB values of the image.

RGB value of each point can be accessed using the getRGB(x,y) method in the BufferedImage class.

A separate class “imageProcessor” was implemented with all the required image manipulation algorithms in a matrix manipulation manner.

Applied changed are temporarily in “ImageStore” Class. Histogram is drawn using the “Histogram” Class separately.

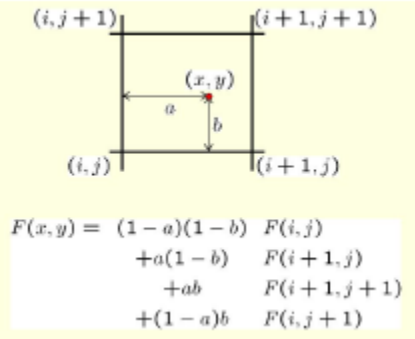
Implemented Techniques

Image Resampling

Image resampling is done basically using two algorithms. Bi Linear Algorithm and Nearest Neighbor algorithm.

In Nearest neighbor method pixel lines are added or removed as lines. When adding same pixels lines are doubled.

By linear method is as described in the image below.



$$F(x,y) = \begin{array}{ll} (1-a)(1-b) & F(i,j) \\ +a(1-b) & F(i+1,j) \\ +ab & F(i+1,j+1) \\ +(1-a)b & F(i,j+1) \end{array}$$

Original

Nearest Neighbour method scale

Bi-linear method scale

Point Operations

Intensity of the pixels is changed without a effect from other pixels around it.

Vertical Flipping, Image Rotation (Clock Wise & anti-Clock wise), Negative Image, Grayscale point operations are available with the tool.

Vertical Flipping

Output and input example of Vertical Flipping

Original Image

Flipped Image

Vertical Flipping Algorithm

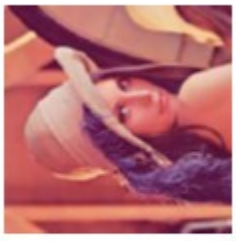
```
int height = image.getHeight();
int width = image.getWidth();

for(int i=0; i<width; i++){
    for(int j=0; j<height; j++){
        tempImage.setRGB(i, j, image.getRGB(width-1-i, j));
    }
```

Image Rotation

Output and input of Image Rotation

Original

Rotated 90 degrees clockwise



Rotated 90 degrees anti-clockwise

Algorithm Used for Rotation

```
int height = image.getWidth();
int width = image.getHeight();

BufferedImage tempImage=new BufferedImage(width,height,
        BufferedImage.TYPE_INT_RGB);

for(int i=0; i<width; i++){
    for(int j=0; j<height; j++){
        if(clockwise){
            tempImage.setRGB(i, j, image.getRGB(j, width-1-i));
        }
        else{
            tempImage.setRGB(i, j, image.getRGB(height-1-j, i));
        }
    }
}
```

Grayscale Operation

Input and Output of the Gray Scaling function.

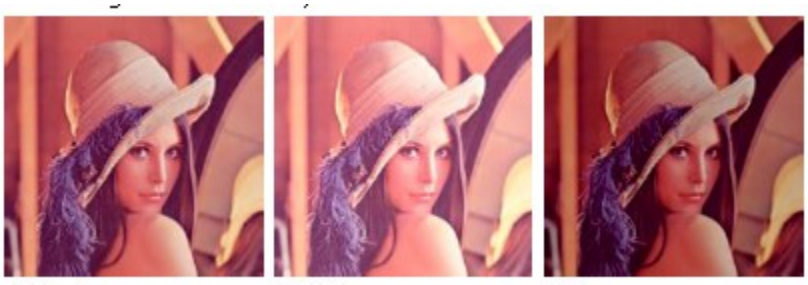
Original

Grayscale

Grayscale value is obtained by getting the mean of the both 3 RGB values.

$$gray\ value = (r + g + b) / 3$$

Brightness Operation



Original Brighten Darken

At each iteration intensity of the R,G,B values are reduced or increased depending on Darkening od Brightening.

Noise Reduction

Mean Filter, Median Filter and Gaussian Filter functions are available with the Tool.

Mean Filter

Input and Ootput of the Mean Filter function.

Noisy Image

Mean Filtered Image

Algorithm

$$g(i,j) = h(i,j) \star f(i,j) = \sum_{m,n=-M/2}^{m,n=M/2} h(m,n) f(i-m, j-n)$$

Convolution is taken with h matrix is all 1/9 s in 3X3 Matrix.

Median Filter

Input and Output of the Median Filter function.

Original Image

Median Filtered Image

Algorithm

$$R(\mathbf{x}) = \{z_1, z_2, \ldots, z_N\}, \text{ where } z_i \leq z_{i+1}$$

$$g(\mathbf{x}) = \text{med}(R(\mathbf{x}))$$

R(x) is a neighboring pixels array, sorted in ascending order.

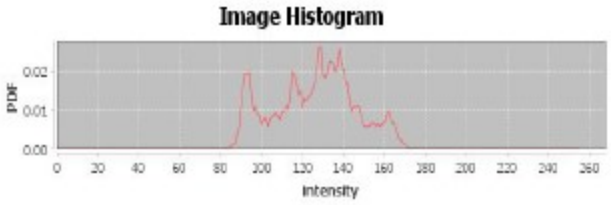
Contrast Adjusting

Input and Outputs for the Contrast Adjusting Function

Original Image

High Contrast Image

Low Contrast Image



Algorithm

A transition function is used as follows.

$$s = \begin{cases} \alpha r & 0 \le r < a \\ \beta(r-a) + s_a & a \le r < b \\ \gamma(r-b) + s_b & b \le r < L \end{cases}$$

Edge Detection

A sudden change or discontinuity of a image is known as an edge. Edge detection is available with two different Edge detection algorithms known as Sobel and Robert.

Sobel

Input and Output to Sobel function

Original Image

Sobel Horizontal
Edge Detection

Sobel function Masks

```
double[][] horizontal = new double[][]{{1, 0, -1}, {2, 0, -2}, {1, 0, -1}};
double[][] vertical = new double[][]{{1, 2, 1}, {0, 0, 0}, {-1, -2, -1}};
```

Robert

Input and output to the

Before Robert Added

Robert Filter Added

Algorithm Mask

```
double[][] horizontal = new double[][]{{0, 0, 0}, {0, 1, 0}, {0, 0, -1}};
double[][] vertical = new double[][]{{0, 0, 0}, {0, 0, 1}, {0, -1, 0}};
```

****