CS - 3712
Image Processing Application

SEMESTER PROJECT

140284J
P.K.K.D. Kanakanamge

18-07-2017

Contents

OVERVIEW

Image processing is the subject of analyzing and manipulation of digital images to improve its quality. Image processing is connected with several fields such as photography, medicine and military.

This report contains the basic techniques and operations of image processing used within the semester project.

This application is capable of delivering the basic image processing operations such as image point operations, image filtering operations and edge detection capabilities. This application also includes a Histogram view for the image to graphical representation of intensities distributed within the image along with their relevant pixel count.

Each of the operations within the application are divided into separate sections so that the user can clearly identify what each operations outputs to the original image. Each section also contains an Undo operation where each operation can be reversed to obtain the original image.

Main intension of this image processing application is to make user identify what each operation is capable of doing for a given image.

OPERATIONS LIST

Point Operations

Point operations of an image refers to changes done by considering individual pixels at a time. This image processing application is capable of operations such as image rotation, transpose, dithering, negative and mirror or vertical flip functionality.

Image Rotation

Image rotation is a point operation which makes the original image rotate by 180 degrees. Code is similar to Image Transpose which is given below.

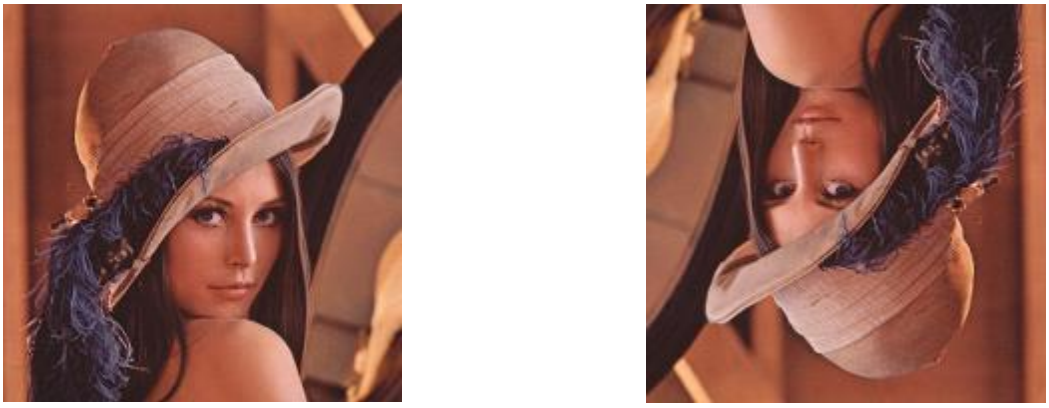


Figure 1 Image Rotation

Transpose of the Image

Transpose is similar to image rotation where image gets rotated by 90 degrees. This is also a point operation where each RGB value of individual pixel is considered.

Code:

```
tempImage = new BufferedImage(image.getWidth(), image.getHeight(),
image.getType());

        for (int i = 0; i < image.getWidth(); i++) {

            for (int j = 0; j < image.getHeight(); j++) {

                tempImage.setRGB(i, j,
image.getRGB(image.getHeight() - 1 - j, i));

            }

        }
```

A copy of the original image is created as tempImage and each original image RGB value is set to the tempImage as in the code.

Dithering an Image

Dithering represent each pixel value of the original image by a combination of black and white pixel values. Image gets 2x size as each individual pixel RGB value of the original image gets replaced by corresponding 4 pixels having black or white values. doDither() function returns the percentage of black and white that needs to be present within the replacing pixel.

```
tempImage = new BufferedImage(image.getWidth() * 2, image.getHeight() * 2,
image.getType());


      for(int i=0;i<tempImage.getWidth();i+=2){

         for(int j=0;j<tempImage.getHeight();j+=2){

            doDither(tempImage,getGreyLevel(image.getRGB(i/2, j/2)),i,j);
```

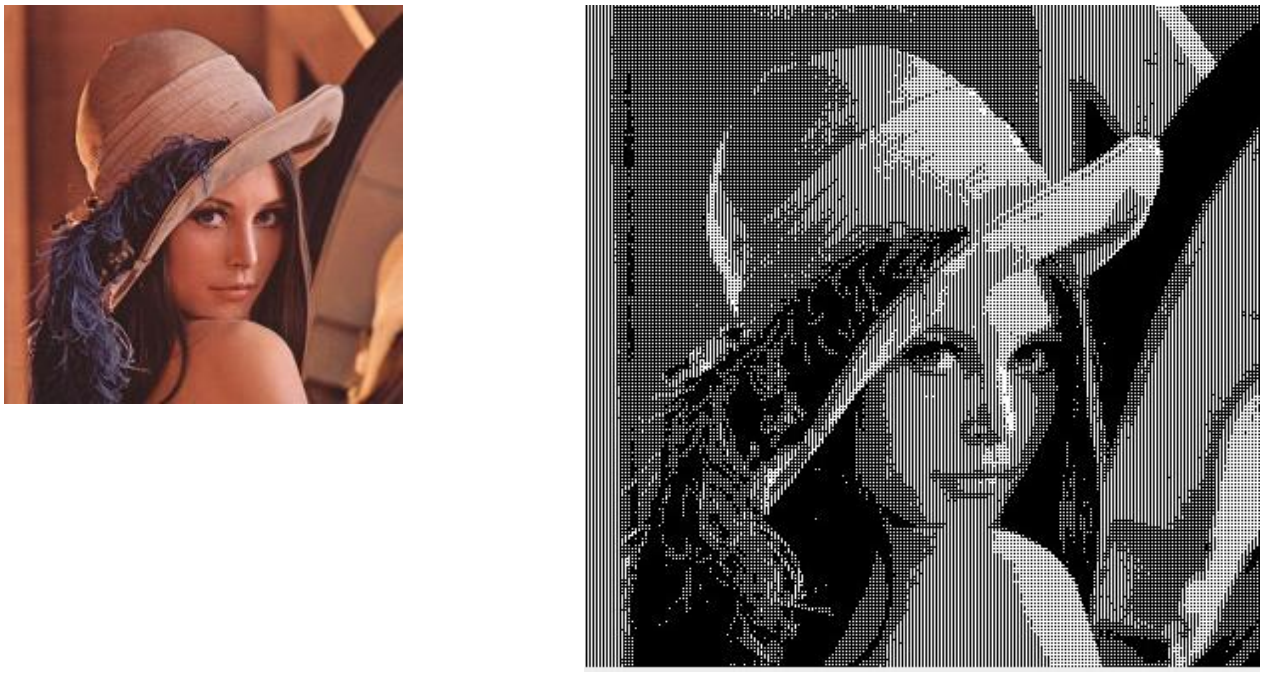


Figure 2 Image Dithering

Negative of an Image

Negative of an image is also obtained by performing a point operation on the image pixels. Each image pixel intensity is reduced from 255 to obtain the new intensity value.

Code:

```
tempImage = new BufferedImage(image.getWidth(),
image.getHeight(), image.getType());

    for (int i = 0; i < image.getWidth(); i++) {

        for (int j = 0; j < image.getHeight(); j++) {

            tempImage.setRGB(i, j, 255 - image.getRGB(i, j));

        }

    }
```

tempImage is the temporary image formed to do the operation from the original image.

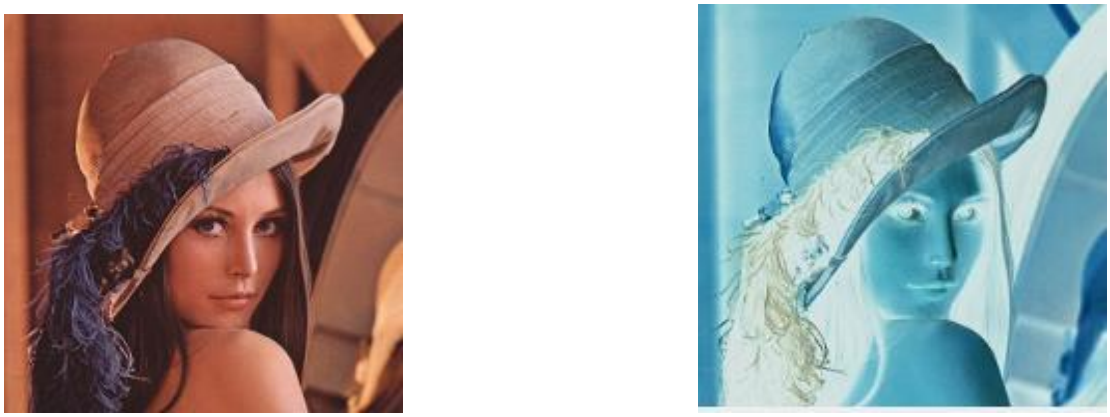


Figure 3 Negative of an Image

Vertical Flip (Mirror) of an Image

Vertical flip forms the mirror image of the original image. This is also a point operation. Changing the position of each pixel from left to right is done to achieve mirror effect.

Code:

```
tempImage = new BufferedImage(image.getWidth(), image.getHeight(), image.getType());
    for (int i = 0; i < image.getWidth(); i++) {
      for (int j = 0; j < image.getHeight(); j++) {
        tempImage.setRGB(image.getWidth() - 1 - i, j, image.getRGB(i, j)); }
    }
```

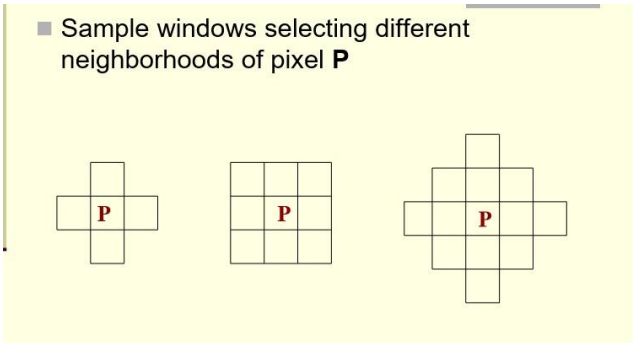
Figure 4 Mirror Operation

Image Filters

Image filters are neighborhood operations where several pixels within a selected area is considered when deciding the new pixel intensity for replacement. Filters are used to reduce the noise within an image. A suitable mask is chosen and relevant operation is done accordingly to obtain the filtered image.

This application can apply mean and median filters to an input image. Filter mask is hardcoded to be a 3x3 matrix.

Sample mask selections as follows.



Mean Filter

Mean filter uses the pre-defined matrix and compute the mean value of the pixel intensities within the neighboring pixels of a selected pixel. Then the pixel surrounded by the neighboring pixels is replaced by the newly computed pixel intensity value.

This result is blur within the image.

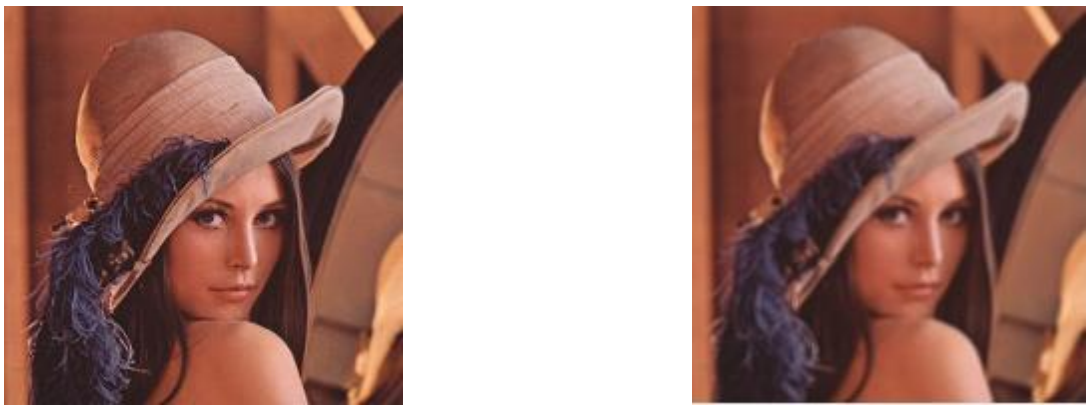


Figure 5 Mean Filter

Median Filter

Median filter also uses the pixels within the neighborhood of a selected pixel. Rather than calculating the mean, median filter first sorts the pixel intensity values and the median pixel intensity value is taken to replace the original pixel intensity value.

This results in less blur in the filter applied image.

This filter results in shifting of edges as median intensity is selected to replace the original pixel value.

Figure 6 Median Filter

Image Resizing

This is also known as image resampling. Application supports nearest neighbor and linear interpolation methods to resize a given image. Image resampling is done by allocating new pixel locations and approximating new intensity values when enlarging images and removing available pixels when reducing image size.

Nearest Neighbor method

Image resampling by nearest neighbor method duplicates every other column and row within the original image to obtain a new image of size 2x of the original image. Size reduction will eliminate every other row and column within the original matrix.

Images can only be resized to a multiple of two.

This method introduces blur when increasing the size of the image as same rows and columns are duplicated. And also this method induces a loss of image details when enlarged.

In the application, it requires a scale factor and this scale factor determines how much the image should enlarge.

Code:

*Source code provided.

Linear Interpolation Method

Linear interpolation method approximate the intensity value of the newly added pixel from the pixels around the newly added location. Rather than duplicating the pixel rows and columns this method preserves image details.

By linear interpolation method, image will not get the blur effect.

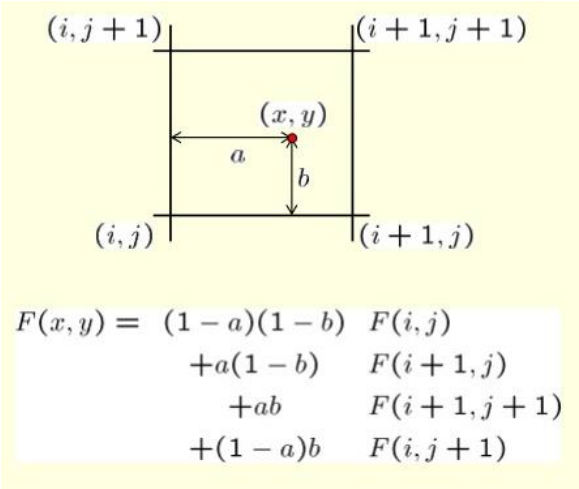
This Software gets a scale factor from user and scales accordingly by using the linear interpolation algorithm.

Code:

*Source code provided

Figure 7 Linear Interpolation Method

Linear interpolation, pixel intensity approximation is done as follows,



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

F(x,y) represent the pixel value to be approximated by surrounding neighbors.

Edge Detection

Edge detection is the process of identifying sudden changes or discontinuities within the image texture. These edges are classified as horizontal edges, vertical edges and diagonal edges.

This software supports edge detection techniques of Prewitt operator, Sobel operator, Laplacien inwards edge detection and Laplacien outwards edge detection.

Threshold value must be provided by the user, this threshold value determines the sharpness of the detected edge.

Sobel Operator

Sobel operator can be used to detect horizontal and vertical edges. This operator uses a mask to detect edges. A sample mask is given below.

Following is the vertical Mask of Sobel Operator:

-1	0	1
-2	0	2
-1	0	1

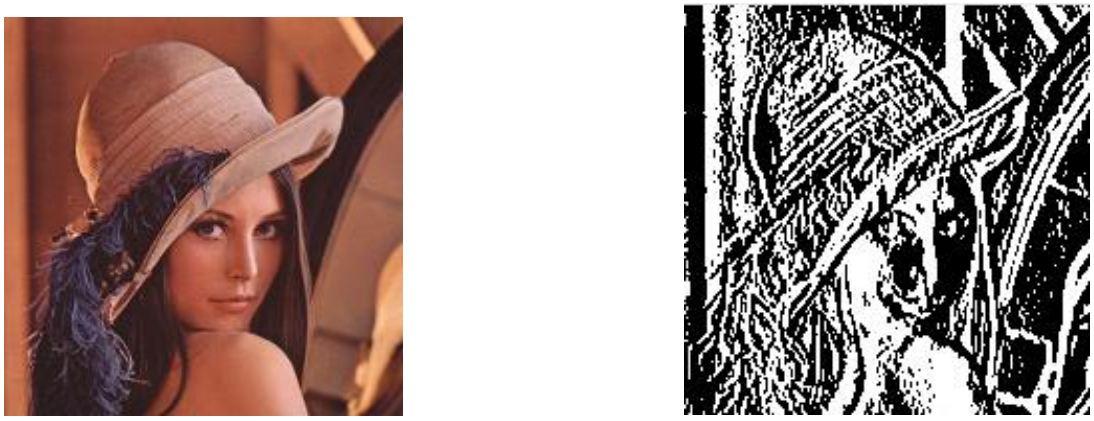


Figure 8 Sobel Operator

Prewitt Operator

Prewitt operator is also similar to Sobel operator with a fixed mask values. This also can be used to detect horizontal and vertical edges. Edge clearness is defined by the threshold value given by user.

Figure 9 Prewitt Operation

Laplacien Operator

Laplacien Operator is much advanced edge detection operator. Rather than detecting horizontal and vertical edges, this operator detects inwards and outwards edges with relevant masks

Major difference of this operator when compared to other operators is that this operator uses a second order derivative mask to detect edges.

0	1	0
1	-4	1
0	1	0

Positive Laplacian Operator is use to take out outward edges in an image.

0	-1	0
-1	4	-1
0	-1	0

Negative Laplacian operator is use to take out inward edges in an image

Here are the screenshots of an image when laplacien operator is used within the software.

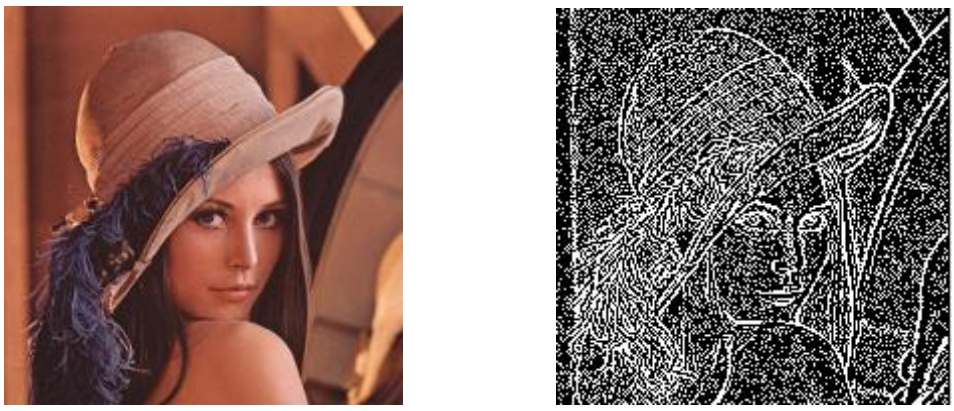


Figure 10 Inwards Laplacien Operator

Figure 11 Outwards Laplacien Operator

HISTOGRAM

Image histogram is the graph showing distribution of intensities of the image along with the number of pixels having each intensity value.

Histogram values can be used to manipulate the image so as to change brightness and contrast within the image.

This software can show the histogram distribution of each input image.

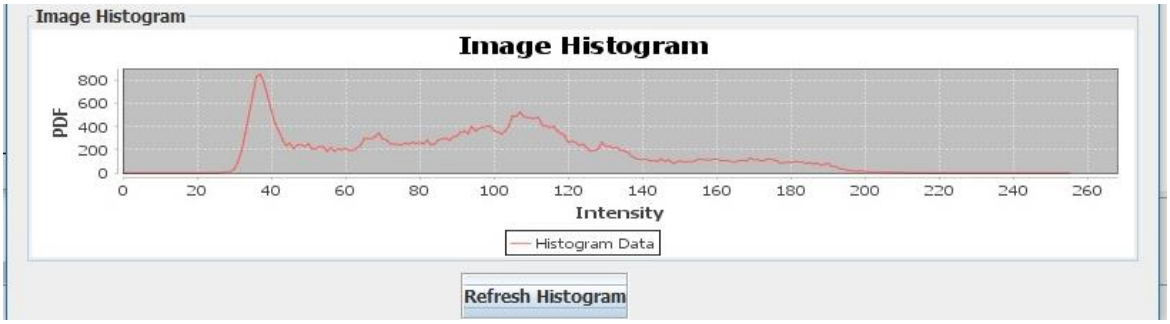


Figure 12 Image Histogram