CS5500 – Digital Image Processing

Cal Poly Pomona

Homework 2

Fall 2023

Description:

Historgram Equalization

Spatial Filters

Bit Planes

Name: Fidelis Prasetyo

Email: (fprasetyo@cpp.edu)

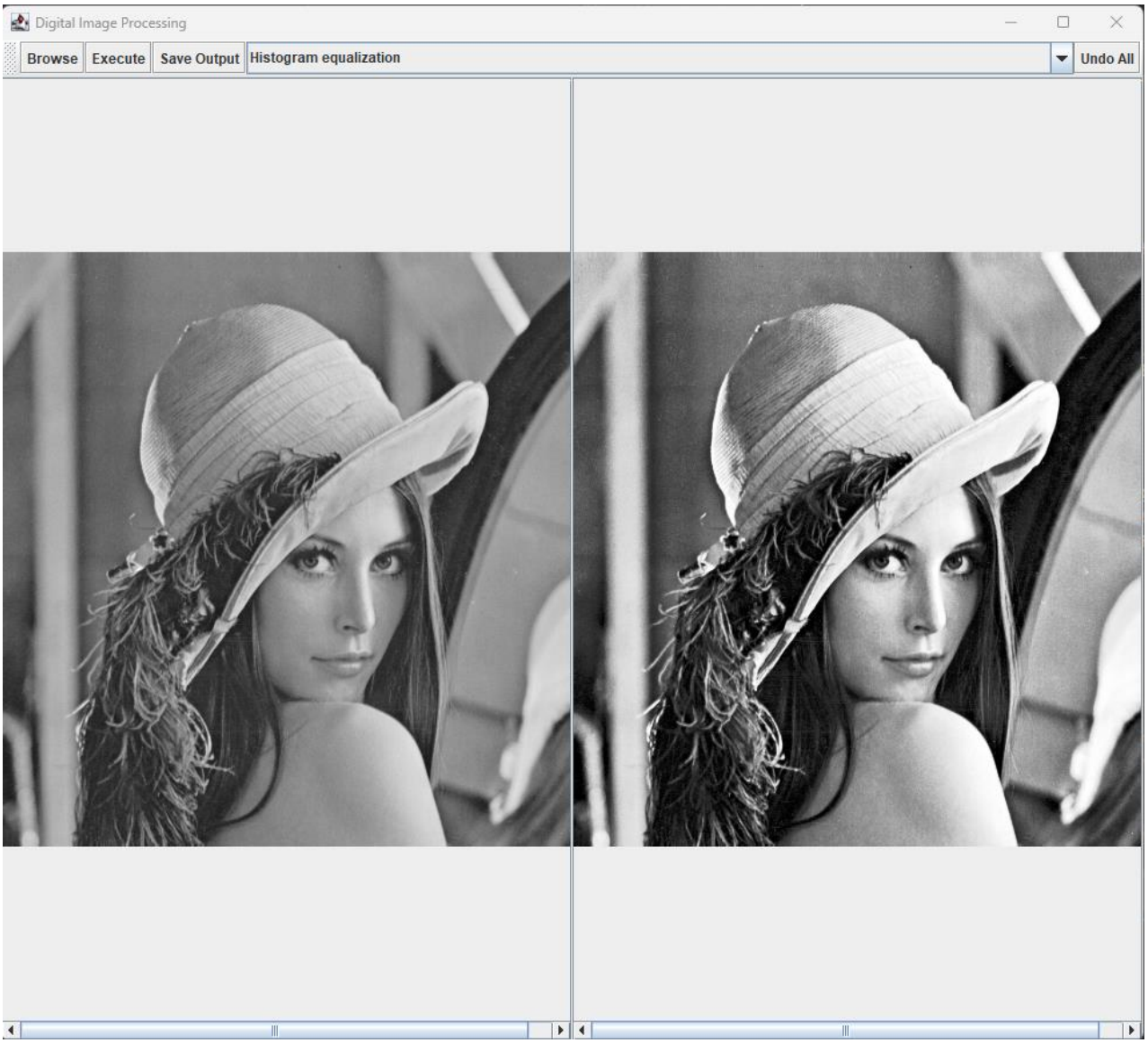
BroncoID: 015765555

Github & Source code:

https://github.com/fidelisprasetyo/DigitalImageProcessing

Program Description

Preview of GUI of the program:



Description:

- Left image: the original image.
- Right image: the processed image.
- Browse button: to open the desired image file.
- Execute button: apply the chosen action.
- Save output: save the processed image (right image) to a file.
- Undo all: revert all changes to the original image.

New Implemented Features:

- Histogram Equalization: equalize the left (input) image. The user can choose between global equalization or local equalization with the desired mask size.
- Spatial Filters: apply the chosen spatial filtering to the input image and the user can put their desired mask resolution:
    - Smoothing filter
    - Median filter
    - Sharpening Laplacian filter
    - High-boosting filter
- Remove bit-planes: The user can choose the desired bit plane to be removed from the image.

Program Demonstration

Operation	Parameter	Original	Result
Global histogram equalization	n/a		
Local histogram equalization	Mask = 3x3		
	Mask = 5x5		

	Mask = 7x7		
	Mask = 9x9		
Smoothing Filter	Mask = 3x3		

	Mask = 5x5		
	Mask = 7x7		
	Mask = 9x9		
Median Filter	Mask = 3x3		

	Mask = 5x5		
	Mask = 7x7		
	Mask = 9x9		
Sharpening Laplacian Filter	Mask = 3x3		

	Mask = 5x5		
	Mask = 7x7		
	Mask = 9x9		
High-boosting Filter	Mask = 3x3 A = 2		

	Mask = 3x3 A = 9		
	Mask = 7x7 A = 2		
	Mask = 7x7 A = 9		
Remove bit-plane	Bit plane = 0		

	Bit plane = 7		

Homework Answers

1. Source code for histogram equalization:
    a. Global Histogram Equalization

```
public static int[] getHistogram(BufferedImage inputImage) {
    int height = inputImage.getHeight();
    int width = inputImage.getWidth();
    int[] histogram = new int[256];
    for (int y = 0; y < height; y++) {
        for (int x = 0; x < width; x++) {
            int grayValue = ImageUtil.getGrayValue(inputImage,x,y);
            histogram[grayValue]++;
        }
    }
    return histogram;
}
public static BufferedImage globalEqualization(BufferedImage inputImage) {
    int height = inputImage.getHeight();
    int width = inputImage.getWidth();
    int[] originalHist = getHistogram(inputImage);

    double[] equalizedCdf = new double[256];
    int[] cdf = new int[256];
    double mn = width*height;

    cdf[0] = originalHist[0];
    for(int i = 1; i < 256; i++) {
        cdf[i] = cdf[i-1] + originalHist[i];
    }

    for(int i = 0; i<256; i++) {
        equalizedCdf[i] = (double) 255 * cdf[i] / mn;
    }

    BufferedImage equalizedImage = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
    for(int y = 0; y < height; y++) {
```

```
        for(int x = 0; x < width; x++) {
            int pixel = ImageUtil.getGrayValue(inputImage,x,y);
            int newPixel = (int) equalizedCdf[pixel];
            int rgb = ImageUtil.convertGrayToRGB(newPixel);
            equalizedImage.setRGB(x,y,rgb);
        }
    }
    return equalizedImage;
}
```

b. Local Histogram Equalization

```
public static BufferedImage localEqualization(BufferedImage inputImage, int
maskSize) {
    int height = inputImage.getHeight();
    int width = inputImage.getWidth();
    BufferedImage equalizedImage = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
    BufferedImage paddedImage = ImageUtil.extrapolateImage(inputImage,
maskSize);

    int padding = maskSize/2;
    for(int y = padding ; y < height + padding ; y++) {
        for(int x = padding ; x < width + padding ; x++) {
            int rgb = getEqualizedPixel(x,y, paddedImage, maskSize);
            equalizedImage.setRGB(x - padding, y - padding, rgb);
        }
    }
    return equalizedImage;
}

private static int getEqualizedPixel(int X, int Y, BufferedImage paddedImage,
int maskSize) {
    int[] cdf = new int[256];
    int[] histogram = new int[256];
    double[] equalizedCdf = new double[256];

    for(int y = Y - maskSize/2; y < Y + maskSize/2; y++) {
        for(int x = X - maskSize/2; x < X + maskSize/2; x++) {
            int grayValue = paddedImage.getRGB(x,y) & 0xFF;
            histogram[grayValue]++;
        }
    }

    cdf[0] = histogram[0];
    for(int i = 1; i < 256; i++) {
        cdf[i] = cdf[i-1] + histogram[i];
    }

    for(int i = 0; i < 256; i++) {
        equalizedCdf[i] = (double) 255 * cdf[i]/(maskSize*maskSize);
    }

    int pixel = ImageUtil.getGrayValue(paddedImage, X, Y);
```

```
    int newPixel = (int) equalizedCdf[pixel];
    int rgb = ImageUtil.convertGrayToRGB(newPixel);

    return rgb;
}
```

2. Bit-plane removal effect:
    a. What effect would setting to zero the lower-order bit planes have on the histogram of an image in general? Please do this implementation and show it and then comment on it.

       Answer: Removing lower-order bit planes doesn't give any noticeable change in general, as lower-order bit planes hold less significant data of the image. For example, in the result above, where I removed the bit-plane 0 from the image, the output is almost identical with the input. Some of the information is lost for sure, but the losses are not visible with naked eye.

    b. What would be the effect on the histogram if we set to zero the higher-order bit planes instead? Please do this implementation and show it and then comment on it.

       Answer: Removing higher-order bit planes on the other hand, affects the image significantly, as the more significant bits hold more significant data of the image. In the demonstration above, I removed the bit plane 7 (msb) from the image and as the result, the image changed significantly.

Source Code & Supporting Files

The entire source code, this pdf file, and output images can be obtained from this GitHub repository:

https://github.com/fidelisprasetyo/DigitalImageProcessing