Zhihao Cao
CSCI 55700
Mar. 6 2016

Report

Description

- **Histogram Equalization**

First, create accumulated normalized histogram for the input image as a look up table. Then, use the value in each pixel to look up new value in the look up table.

- **Logarithm mapping**

This one is fairly simple. I apple the formula c*log(pixel value) to each pixel. And form a new image by that.

- **Rotation**

For the input byte array, I hardly find a way to rotate the image in the array. I load it into BufferedImage and so that I can use AffineTransform for rotation. However, when I tried to simply rotate an image, it is not rotate about the center point. I have to create a new image and draw the rotated pixel in it. The size of new image is define by

```
w = (int) (d * Math.abs(Math.cos(angle)));
h = (int) (d * Math.abs(Math.sin(angle)));
if (w>h) h=w; //always use the long edge as image edge
else w=h;
```

Then, use AffineTransformOp to load the rotated image.

- **Gaussian filter**

First, Use Random.nextGaussian() to create noise for the image.
Then, create the filter array using sigma and size input parameters.
Then, for each pixel, loop the neighbor pixels in filter size and use the filter array to generate a new value. And create a new image by doing this to all pixels.

- **Median filter**

First, create salt-pepper-noise

```
if (value<noise) //0 < noise < 10
        if(value<noise*.5)
            image_matrix_median[i][j] = 255;
        else
            image_matrix_median[i][j] = 0;
else
        image_matrix_median[i][j] = image_matrix[i][j];
```

Then, for each pixel, push neighbor pixels (3x3) to an array. (Need boundary detection to avoid special case error) Then, sort the array and return the median value of the array for this pixel. Do this for all pixels to form a new image.

Result

Building.pnm

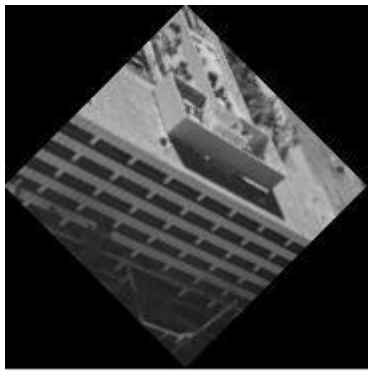

Original Histogram Equalization

Log mapper (constant)



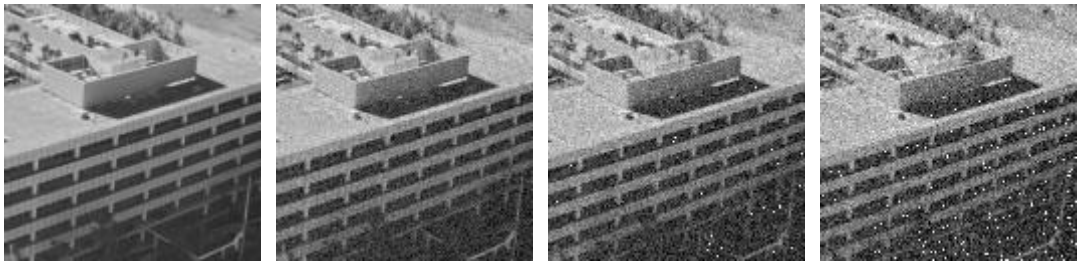

2 5 10 15

Rotation (degree)


10 45 90

Gaussian Noise (noise level)

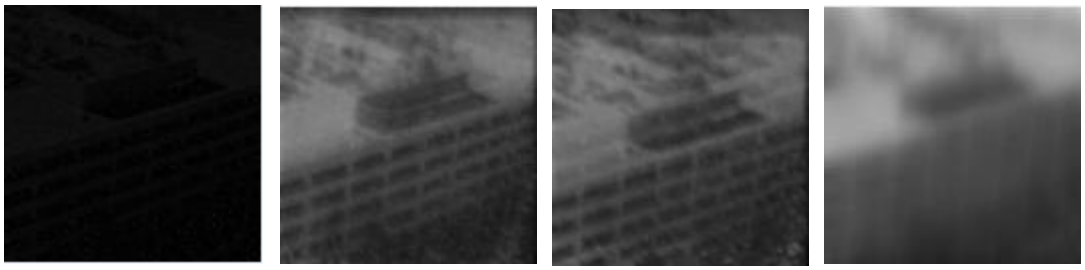


1 10 15 20

Gaussian filter (noise level, sigma, size) -> changing sigma

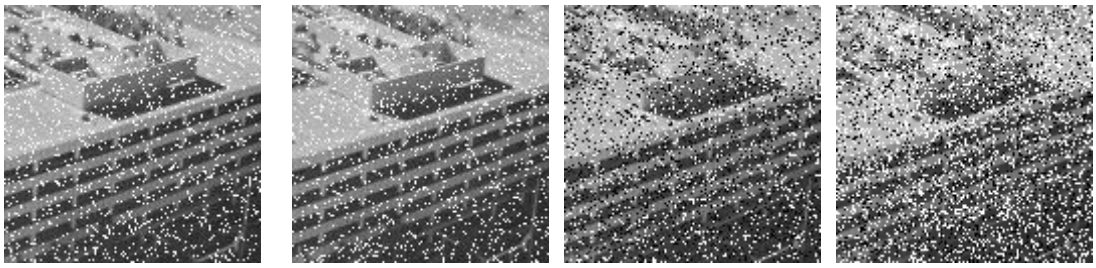
(20, 1, 9) (20, 5, 9) (20, 10, 9) (20, 20, 9)

Gaussian filter (noise level, sigma, size) -> changing size



(20, 5, 4) (20, 5, 16) (20, 5, 32) (20, 5, 49)

Salt-pepper Noise (0 < noise level < 10)



0.01 1 2 3

Median filter (3x3)

Analysis

1. Histogram Equalization

It can adjust image intensities to enhance contrast. For dark image, it can make it brighter. For too bright image, it can balance the darkness and brightness. Basically, it stretches the gray scale values of original image and so that for the new image, the gray scale values can spread from 0 to 255. So, it would be useful when the image's intensities are focus on some regions. I have done a simple comparison with Photoshop; they pretty much are the same results, but PS has more options for the tuning.

2. Logarithm mapping

It can enhance the low intensity pixel values. After applying this function, it will reveal more details in dark regions. For many dark images, this can be an effective algorithm to be applied. I have done a simple comparison with Photoshop; they pretty much are the same results, but PS has more options for the tuning.

3. Rotation

The explanation is simple, just to rotate the image. However, to implement the rotation is a bit tough that I expected. I have to create new image and apply AffineTransform to each pixel. Besides, I have to calculate the new size of the image so that all pixels can fix in the new image. In our daily life, image rotation is everywhere. It helps us recognize images, design things, etc. Comparing the result

with Photoshop, my implementation is less professional. Mine can't get rid of the black background part. And the border is like zigzag. PS's result is very smooth in the border.

4. Gaussian filter

It can reduce image noise and smooth sharp region. However, it will remove details from image. This filter is widely applied in in our daily life. Sometime, we may just want to make pictures to be blurred and increase their attraction.

5. Median filter

It can effectively reduce the salt-and-pepper noise. It is fairly low cost algorithm to reduce noise. In blur image, this function would be less useful.

Discussion

Since I use java to implement the functions, I encountered a lot of challenges. I had paid a lot of efforts to solve many issues, but some of them couldn't be solved. For example, it works perfect for square type image (building.pnm). However, for rectangular images, only part of the image can be processed. I had paid a lot of effort to solve this problem, but it seems like a problem for pnm format. I tried to use jpg file to run the functions; there is no such problem occurred. One tricky thing should be noticed is that java reads byte as signed byte [-128,127]. As you know gray value in a byte should be [0,255]. This feature caused that my output image is messed. After I did some research and I found out this feature. I use: data = byte[i] `& 0xff` to solve problem.