**Assignment1: Thresholding and Smoothing in ROIs for gray scaleand color images**

Yu Liang

Department of Computer Sc. and Engineering. University of South Florida, Tampa, Florida, USA

**1. Introduction**

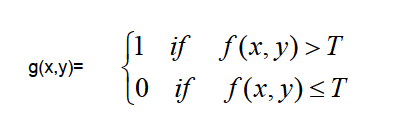
This assignment focuses on efficient implementation of image manipulation techniques like thresholding and smoothing in gray scale and color images using programs wrote in C++. Image thresholding/binarization is to set up a threshold value by the user and compares image pixel value with the threshold value. If the image value is larger than threshold value, the image value is set to be white, otherwise it is set to be black. We will implement two kinds of binarization: graythresholding and colorthresholding. In graythresholding, we compare the gray value with the threshold. In colorthresholding, we calculate the distance between the RGB image value with the user defined RGB value. If the distance is larger than a user defined threshold than it is set to be white, vise versa. In the other section, we will implement three smoothing: 1D smoothing, 2D smoothing and edge preserving smoothing. We will compare the result from each smoothing and discussion the performance. All implementation will be applied on Region

Of Interest (ROI).

**2. Description of algorithms**

**2.1. Gray Image Thresholding**

Image thresholding/binarization is to set up a threshold value by the user and compares image pixel value with the threshold value. If the image value is larger than threshold value, the image value is set to be white, otherwise it is set to be black.



**2.2. Color Image Thresholding**

In colorthresholding, we calculate the distance between the RGB image value with the user defined RGB value. If the distance is larger than a user defined threshold than it is set to be white, vise versa.

**2.3. 1D Smoothing**

We use smoothing to reduce the noise in the image. In 1D Smoothing, we reset the image value using the average of the window size image value. It is called 1D because we only consider the values in the same row with the image pixel. For example, if the window size is 3 and the image pixel values are 2, 5, and 6, then the pixel value 5 will be replaced by the average of these three values, which is 4.

**2.4. 2D Smoothing**

2D smoothing is similar with 1D smoothing. The difference is that in 2D smoothing, the window is 2D, meaning that it’s

an odd number N \* N windows. The image value will be replaced with the average of the window size image value.

**2.5. Edge Preserving Smoothing**

During smoothing, we also want the edge of the pictures to be preserved. In Edge Preserving Smoothing, we set a threshold based on 2D smoothing. We first calculate the average of the window size value and then compare the value with our threshold. If the difference is large which means the image pixel is highly likely on the edge, then we preserve the image value. Only when the average of the window size value is smaller than the threshold, we may replace the value with the average

**3. Description of implementation**

The entire code is developed in C++ language on a FSprime server. The code is based on the sample code for Assignment 0 (sample project). We will show the result of 1. Binarizaiton: Gray thresholding, Color thresholding. 2. Smoothing: 1D smoothing, 2D smoothing and Edge preserving smoothing. Implement rectangular region of interest (ROI) selection by specifying pixel location (X,Y) of the left-top pixel of ROI and ROI size (Sx,Sy). Modify basic thresholding function to operate only within specified ROI(s). Implement uniform smoothing filter operation using square odd window size WS. Program two implementations: (a) regular 2D smoothing, (b) separable 1D incremental implementation of.

Add "edge-preserving" uniform smoothing of "size" WS (first input parameter) to options of gray level image processing. Let threshold TWS be the second user defined input parameter.

Add color binarization option to your image processing toolbox. Let threshold TC be user defined input parameter. For color images, consider TC be a distance from user defined color C(CR,CB,CG) in RGB space. Set all pixels within TC distance to "white" and the rest to "black".

**4. Results**

Figures 1 show the results of gray thresholding on grey scale images. We can see as the threshold become larger, there are more white space than black space.



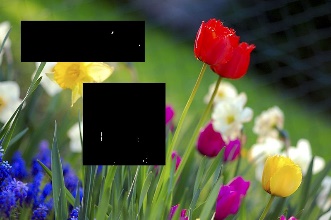




**Figure 1. Original images of a tree and two on gray Thresholdings on Gray Scale Images. The middle image is set with threshold 50 and the bottom image is set with threshold 150 with 2 ROIs.**

Figures 2 show the results of Color thresholding on color scale images. We can see the result varied as we change the 'threshold(TC)' 'Red(CR)' 'Green(CG)' 'Blue(CB)'.





**Figure 2. Original images of a garden and three on Color Thresholdings on color Images. The upper right image is set with threshold 150, RGB ( 10,10,10). The bottom left right image is set with threshold 150, RGB ( 0,255,0). The bottom right image is set with threshold 200, RGB ( 0,255,0).**

Figures 3 show the results of 1D smoothing and 2D smoothing on Gray scale images. We can see the 2D smoothing reduce more noises but also make the picture vague.

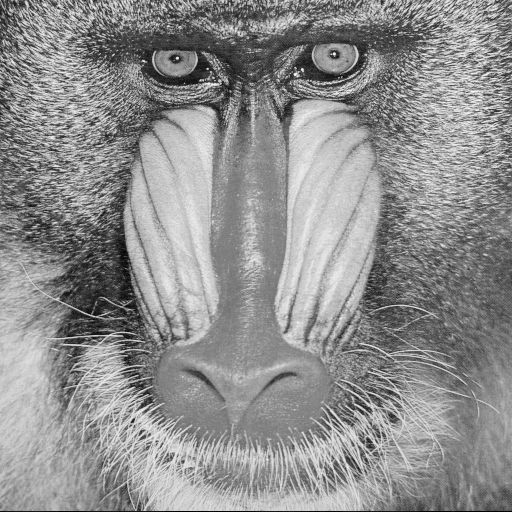


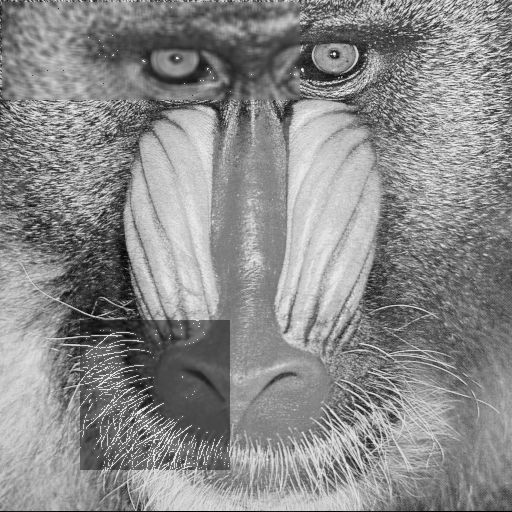
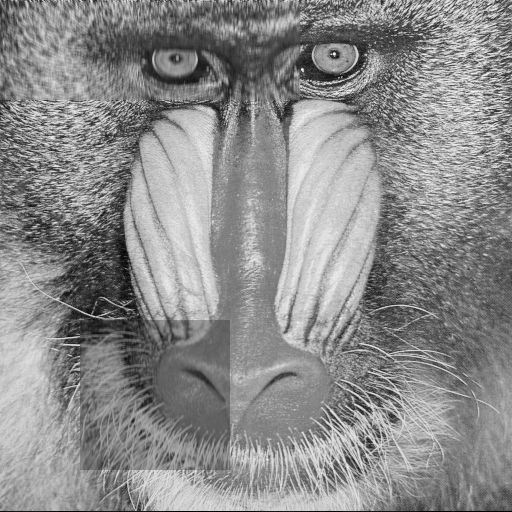




**Figure 3. Original images of lena and two on smoothing Images. The middle image is 1D smoothing and the bottom image is 2D smoothing.**

Figures 4 show the results of Edge Preserving Smoothing on Gray scale images. We can see the whisker of the baboon is preserved.



**provid value of (225,225,** **225) to segment the white egg.**

**Figure 4. Original images of baboon and two on smoothing Images. The middle image is 1D smoothing and the bottom image is Edge Preserving smoothing.**

**5. Discussion**

Discuss results and performance (including execution speed of two implementations) of uniform

smoothing operation: the 1D smoothing takes shorter time than the 2D smoothing because it goes through less pixel to calculate the average.

Discuss results of edge preserving smoothing operation: The edge preserving smoothing did a good job to preserve the edge of the image. In our example, the whisker of the baboon was preserved.

Discuss results of color binarization: Color binarization can be complexed and it takes more tries to find the best solution.

**6. Conclusions**

This assignment shows the basic implementation of the binarization and smoothing. It highlighted the good performance of edge preserving smoothing.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Readme:

Yu Liang

This software is architectured as follows. This software can work on grad server.

iptools -This folder hosts the files that are compiled into a static library.

image - This folder hosts the files that define an image.

utility- this folder hosts the files that students store their implemented algorithms.

lib- This folder hosts the static libraries associated with this software.

project- This folder hosts the files that will be compiled into executables.

bin- This folder hosts the binary executables created in the project directory.

\*\*\* INSTALATION \*\*\*

On Linux

Enter the project directory in terminal and run make

As a result you should get iptool in project/bin directory.

\*\*\* FUNCTIONS \*\*\*

1. Add intensity: add

Increase the intensity for a gray-level image.

2. gray Binarization: binarize1

Binarize the pixels with the gray threshold.

3. color Binarization: binarize2

Binarize the pixels with the color threshold.

4. 1D smoothing : smooth1

5. 2D smoothing : smooth2

6. edge preserving smoothing : edgesmooth

7. Scaling: Scale

Reduce or expand the heigh and width with two scale factors.

Scaling factor = 2: double height and width of the input image.

Scaling factor = 0.5: half height and width of the input image.

ROI\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 # Number of ROI

0 0 100 300 100 # Parameters for ROI1: x, y, sx, sy, T/WS

320 80 150 150 200 # Parameters for ROI2: x, y, sx, sy, T/WS

\*\*\* PARAMETERS FILE \*\*\*

There are for parameters:

1. the input file name;

2. the output file name;

3. the name of the filter. Use "add", "binarize1 or binarize2", "smooth1, smooth2, edgesmooth" and "scale" for your filters;

4. the value for adding intensity, threshold value for binarize filter, or the scaling factor for scale filter.

\*\*\* Run the program: ./iptool parameters.txt

parameters.txt Example:

baboon.ppm baboon\_colorthre.ppm binarize2 125 10 10 10

baboon.pgm baboon\_graythre.pgm binarize1 125

baboon.pgm baboon\_add50.pgm add 50

baboon.pgm baboon\_smooth2d.pgm smooth2 20

baboon.pgm baboon\_smooth1d.pgm smooth1 20

baboon.pgm baboon\_edgesmooth.pgm edgesmooth 100