Project 1: Gray Scale/Color Binarization and 1D/2D Uniform Smoothing using ROI

Daniel Sawyer U3363-7705
CAP5400 Image Processing Fall 2019
University of South Florida, Tampa

I. 1. Introduction and overall description

This assignment focuses on basic image manipulation using thresholding/binarization as well as 1D and 2D uniform smoothing within a specified ROI. Image thresholding/binarization compares pixel values with a user defined threshold. In this case, if the pixel value is less than or equal to the threshold, it is set to white. If it is above the threshold, its set to black. The 1D and 2D smoothing is basically identical when it comes to the output image but 1D is significantly faster. The smoothing is done by averaging all the pixels around it in a "window" that is specified by the user. This reduces noise but also makes the image appear blurry. In Section 2, the basic algorithms used in the assignment are described. Section 3 describes the implementation details of the assignment. In Section 4 the results for the assignment is presented and finally we conclude in Section 5.

II. Description of algorithms

In this section the basic algorithms used in this assignment are described. The algorithm for gray scale thresholding/binarization, color thresholding/binarization, 2D smoothing, and then 1D smoothing are described as follows.

A. 2.1 Gray Scale Thresholding

This algorithm just runs through all of the pixels within the ROI and checks their value against a user defined threshold Tmin and Tmax. If pixel value is in between Tmin and Tmax, then the pixel value is set to white/255. If it is not, the value is set to black/0.

B. 2.2 Color Thresholding

This algorithm goes through every pixel and calculates a distance using each channel of the pixel RGB(Red Green Blue). The user defines a threshold(TC), Color Red(Cr), Color Green(Cg), and Color Blue(Cb). You then take the distance = $\sqrt{(RED - Cr)^2 + (GREEN - Cg)^2 + (BLUE - Cb)^2}$ and compare it to the TC value. If it is less than or equal to the TC value, set all channels(RGB) to white/255. If it is greater than TC, set all channels(RGB) to black/0.

C. 2.3 2D Smoothing

This algorithm takes the average of pixel values within a Window Size(WS) which is an odd number 3 or greater squared. A WS value of 3 would make a window 3x3 in size for a total of 9 pixels. In order to read these pixels, a value M is calculated where M=(WS-1)/2. An outter double for loop goes through all the pixels i,j within the ROI specified by the user. Then an inner double for loop goes through all the pixels in the window k,l. The inner for loops are looped through by this formula, average = $\frac{1}{WS^2}\sum_{k=i-M}^{i+M}\sum_{l=j-M}^{j+M} I(k,l)$. The average is then placed into the value of pixel I(i,j). This performance can be increased by using the 1D algorithm which is explained next.

D. 2.4 1D Smoothing

Like 2D smoothing, 1D smoothing takes the average of all the pixels in a window in order to reduce noise while also causing the image to become blurred. The smoothing also takes in an odd number for window size(WS) and uses that as the window to average the pixel values. M is also calculated the same, M=(WS-1)/2. The difference comes in the way you go about averaging these pixels. Unlike 2D smoothing, 1D smoothing goes through all the horizontal pixels and vertical pixels in separate loops while using a rolling window. The algorithm uses double for loops that go through every pixel i,j. For the horizontal, the first for loop goes through all the rows i and the second loop goes through all the columns j. The inner loop goes through the first WS group of horizontal pixels j like this: average = $\frac{1}{WS}\sum_{k=j-M}^{k+M} I(i,k)$. After that, average = average - I(i,j-M-1) + I(i,j+M) becuase those are the only 2 values that change for the rest of the row. These values are saved into an intermediate array and then used instead of the source image for the vertical process. The vertical process is the same process as the horizontal but is repeated with i and j interchanged in order to read through the pixels vertically. This produces an almost identical image as the 2D smoothing but does it much much faster than the 2D smoothing.

III. 3.Description of implementation

The entire code is developed in C++ language on Ubuntu 16.04.1 and USF's FSPrime linux server. The code for reading and writing the ppm and pgm image files is provided as part of the image class file. The utilities class file contains all 4 process that were described in this report above. The

main is in project1.cpp and reads the parameters file, which contains all the info on the input image, output image, and process to run on the image with their ROIs. It then saves the resulting image(s). For gray scale thresholding, the parameters are the threshold T. For color thresholding the parameters are threshold TC, Cr, Cg, Cb. For 1D and 2D smoothing the parameters are windows size WS. All processes also have an ROI that uses the following format that is put in after the process parameter(s): startX startY SizeX SizeY. The README includes a more detailed explanation.

IV. 4.Description and Analysis of Results

A. 4.1. Description of Results

Fig. 1. Binarization w/ 5 ROIs, 2 overlap

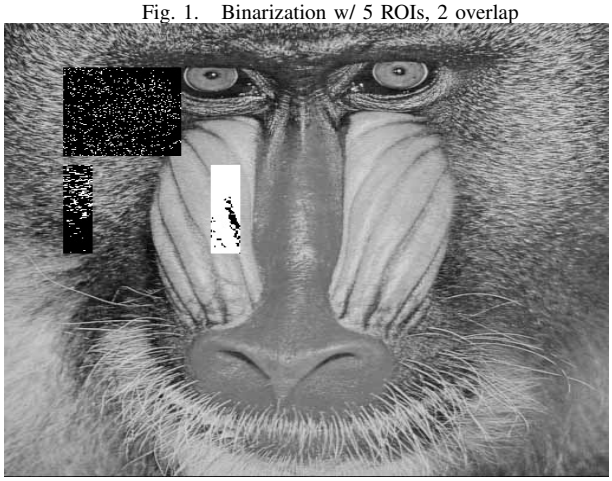
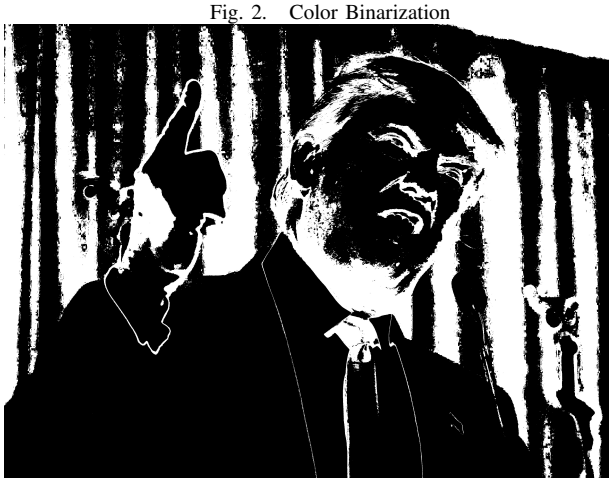


Fig. 2. Color Binarization



The first image figure 1 is the use of thresholding with a gray scale image. From left to right, the threshold value Tmin/Tmax = 120/150, then 50/100, and finally 100/200. As you can see, the higher the T values, the more white there is, the lower the T values, the darker it becomes since the image isnt very dark.

The second image figure 2 is the use of color thresholding. From right to left you have TC Cr Cg Cb values as follows: 128 200 200 0, 200 255 0 0, 150 0 200 200. You can see that depending on the Cr Cg Cb values that certain colors are highlighted more than others.

Fig. 3. 1D Adaptive Uniform Smoothing

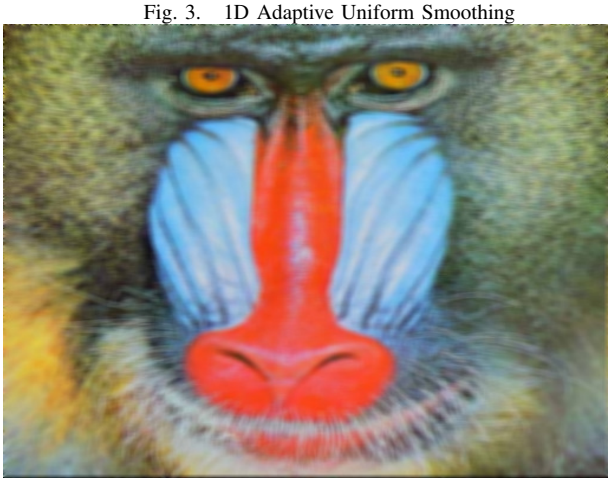
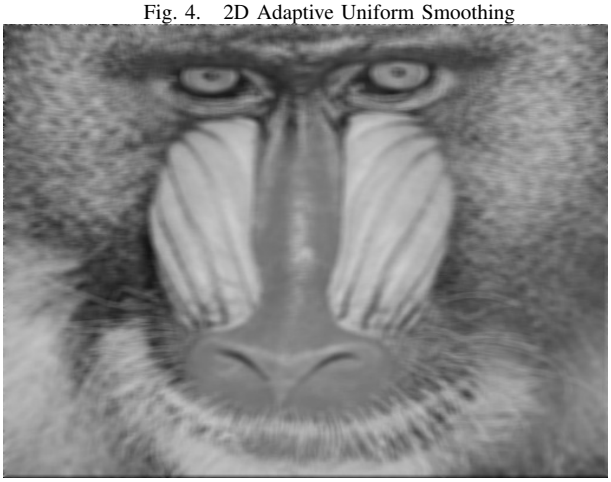


Fig. 4. 2D Adaptive Uniform Smoothing



The third image figure 3 is an example of 1D Smoothing. From left to right, the values of window size WS are different and are set as follows: 5, 9, 7. As you can see, the larger the WS, the more noticeable the change. As the WS increases, the more the image is blurred.

The final image figure 4 is an example of 2D smoothing. It looks identical to the previous figure 3 which uses 1D smoothing. The WS values are the same as above from left to right. The only difference between this and 1D is the speed. This happens to be much slower.

B. 4.2. Performance Evaluation and Analysis of Results

Both gray scale and color thresholding run at O(n^2), where n^2 is the size of the 2D pixel array or i x j. 1D Smoothing runs at O(n^2) as well, this is due to it running through every pixel in the image, same as above. 2D Smoothing runs at O(n^2m^2) where m is the windows size. This is significantly slower than 1D Smoothing, especially with large images.

V. 5.Conclusion

This assignment was used as a way to learn how to perform basic image manipulation. The basics of thresholding/binarization and uniform 1D/2D Smoothing are simple manipulations that are at the core of image processing. The thresholding can be used to bring certain parts of the image

or colors to the forefront while the smoothing is good for noise reduction. All of these processes help build our basis for future image processes that will come in this class. It was also intriguing to see how 1D Smoothing produces the same image with quite a bit less complexity. The Understanding of these basic techniques are important for learning more complex processes to come.

REFERENCES

[1] Example Project Code and Example Report from TA website.