# Assignment 2: Automatic Thresholding and Color Conversion

Garrett Moncrief
Dept of Computer Science and Engineering
University of South Florida, Tampa, Florida, USA

## 1. INTRODUCTION

This assignment focuses on the use of auto thresholding in conjunction with smoothing techniques and color conversion. As used in the previous assignment we will use up to three regions of interest, each of which can have individual parameters set. Regions outside of a ROI will not have any changes made to them.

## 2. DESCRIPTION OF ALGORITHMS

In this section a description of the algorithms used will be provided.

ROI Selection - Using input parameters of x, y, sx, and sy while the image processes from i0, j0 to in, jn it will apply the changes when it reaches the region of x-sx, y-sy to s+sx, y+sy.

Automatic Thresholding - An initial threshold value is established (in this case either by taking the mean value of the ROI or allowing the user to establish a value) and the pixels in the area are divided into a background comprised of pixels less than or equal to the threshold and a foreground comprised of pixels greater than the threshold value. An average mean of the two new images is calculated and then an average of these two means is calculated. This value is subtracted from the initial value and if the result is less than a specified limit the process is finished. Otherwise the process is repeated with the new threshold value.

Conversion from RGB to HSI- the first step in converting RGB color space to HSI space is to normalize the RGB values. To do this we establish a r, g, b value each based respectively on R/G/B each divided by R+G+B. Then a value for h, s, and i need to be established which will be later normalized to the Hue, Saturation, and Intensity values.

To calculate hue value depends on whether b is greater than g. If so the calculation is:

$$h = 2\pi - \cos^{-1}\left\{\frac{0.5\cdot\left[(r-g)+(r-b)\right]}{\left[(r-g)^2+(r-b)(g-b)\right]^{\frac{1}{2}}}\right\} \qquad h \in [\pi, 2\pi] \text{ for } b > g$$

otherwise the calculation is:

$$h = \cos^{-1}\left\{\frac{0.5\cdot\left[(r-g)+(r-b)\right]}{\left[(r-g)^2+(r-b)(g-b)\right]^{\frac{1}{2}}}\right\} \qquad h \in [0, \pi] \text{ for } b \le g$$

To calculate the s value it is just 1 - (3 multiplied by whichever is the minimum value between r, g, or b). Finally the i value is (R+G+B)/(3*255)

Since Hue has a range of [0, 360], Saturation has a range of [0, 100] and Intensity has a range of [0, 255] the h, s, i values are normalized by letting H=h*180/pi, S=s*100, and I=i*255.

## 3. IMPLEMENTATION

The code was modified and developed in C++ in Visual Studio on a Windows 10 machine. The program was run directly through Visual Studio, using a parameter file that loaded images from an input directory and saved output files to an output directory.

On the parameter file the user must first note the number of images to be created. They then must specify the input and output paths. Then they must specify the operation and the number of ROI to be included, from 1 to 3. For the ROI the x, y, sx, and sy values must be specified. If more than one ROI is to be implemented the additional parameters must be included before beginning the next ROI.

Note: these procedures repeat for each ROI.

In my implementation the user can either specify an int for the initial threshold or if they insert a 0 an initial threshold is chosen as the value of pixels inside the ROI / the number of pixels inside the ROI.

These values are then used to divide the image into the foreground and background, with image values greater than the threshold being the foreground and less than the threshold being the background.

The average value of the background is added to the average value of the foreground and this number is divided by two. If the previous threshold - this value is under the user specified limit then this is the value that will be used for thresholding. Otherwise this value is set as the new threshold and it recalculates the foreground and background images.

The user can also specify if they want the images smoothed by entering 1 at the end of the parameters or not smoothed by entering 0.

For RGB<->HSI autothreshing the user specifies rgbautothresh, the ROI information, and enters 1, 2, 3 for hue, saturation, and intensity thresholding respectively.

The image is converted to the HSI information and then an average auto threshing value is calculated.

The image is parsed again picking out a foreground and a background based on the threshold value and the factor that it is being applied to.

Finally the image values are replaced, using MINRGB and MAXRGB for the ROIs the threshold is being applied to and using the original image colors elsewhere.

4. DESCRIPTION AND ANALYSIS OF RESULTS

This section illustrates the results of the algorithms used.

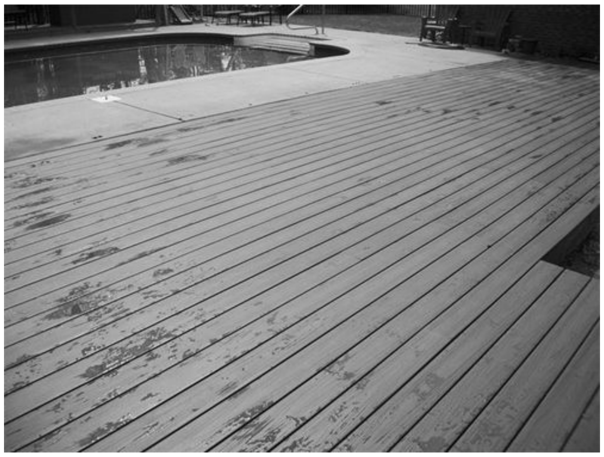
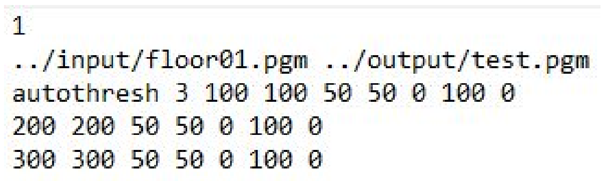
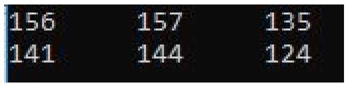
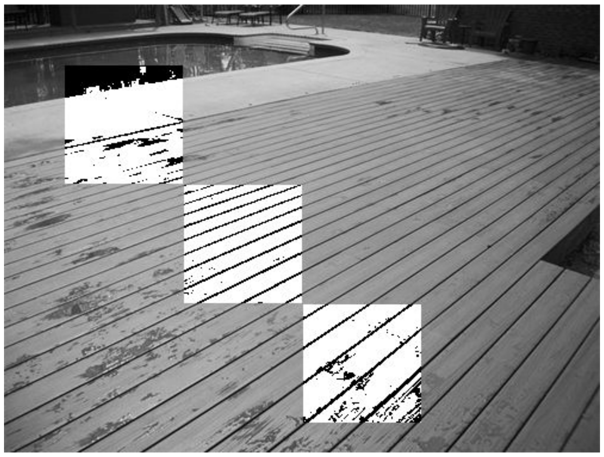
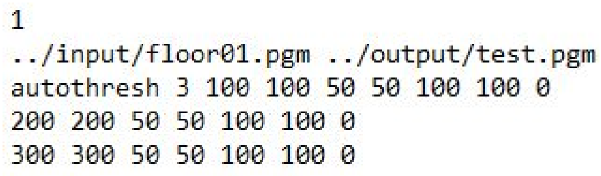


**Figure 1. The original image used for the auto thresholding functions.**

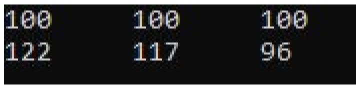

```
1
../input/floor01.pgm ../output/test.pgm
autothresh 3 100 100 50 50 0 100 0
200 200 50 50 0 100 0
300 300 50 50 0 100 0
```


```
156      157      135
141      144      124
```

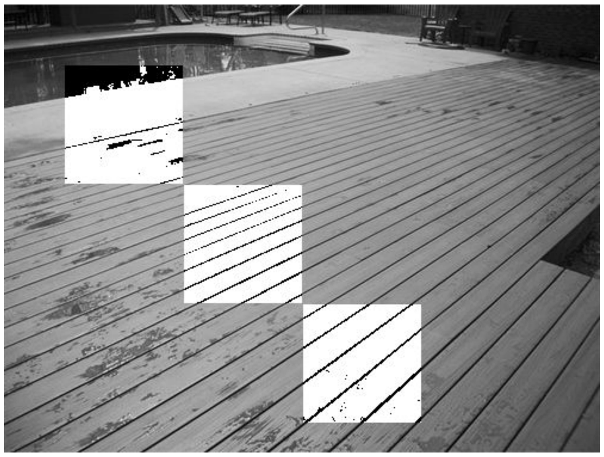

**Figures 2, 3, 4. The parameters entered for the auto thresholding function followed by the initial threshold calculated by the function and the threshold ultimately used on the image. The final image is the result of the function.**



```
1
../input/floor01.pgm ../output/test.pgm
autothresh 3 100 100 50 50 100 100 0
200 200 50 50 100 100 0
300 300 50 50 100 100 0
```



```
100       100       100
122       117       96
```



**Figures 5, 6, 7. The parameters entered for the auto thresholding function followed by the initial threshold value entered by the user and the value that was used for threshing. The final image is the resulting image generated.**

When the automatically generated initial threshold value is compared to the user entered initial value it is evident that he automatically generated value shows more detail in the image. It is assumed that the user could "play with" the numbers until finding one that worked well for the image, however the autotresholding function seems to offer a good and quick solution.

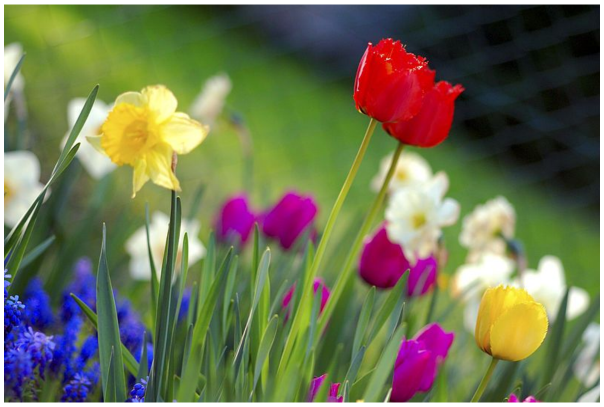
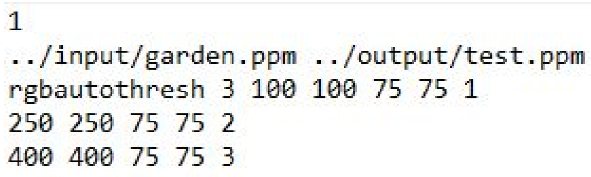
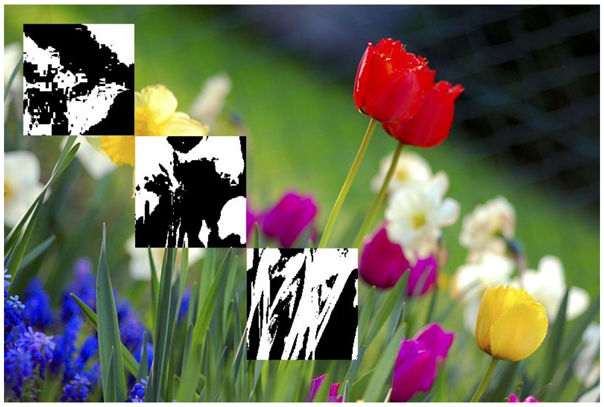


**Figure 8. The original image used for the rgb auto thresholding function.**



```
1
../input/garden.ppm ../output/test.ppm
rgbautothresh 3 100 100 75 75 1
250 250 75 75 2
400 400 75 75 3
```



**Figures 9 and 10. The parameters entered for rgb auto thresholding. The resulting image shows thresholding applied to the hue at top, saturation at center, and intensity at the bottom.**

## 5. CONCLUSION

In this assignment we added two new important tools to our kit: one to quickly and automatically generate threshold values and one that can apply threshold values to hue, saturation, and intensity channels rather than just to rgb color channels.