Assignment 1

Image Manipulation Techniques

REPORT

Prepared by Pragnesh Patel

for

Prof. Dmitry Goldgof

Submitted on September 16, 2011

Table of Contents

Introduction

The purpose of this assignment included getting acquainted with various image manipulation techniques. Given a basic image processing toolbox, we were to add 4 basic filters that manipulated images in specific ways. Keeping in mind both the data structures and algorithm efficiency, the four functions :GrayThreshold, Arithmetic, ColorThreshold and AdaptiveThreshold were implemented to allow batch image processing. A major part of the assignment also included defining regions of interest, which allows for multiple manipulations in the different parts of an image. This report first looks at the algorithms of each function and then continues on to cover ipToolBox implementation and test results.

1. Algorithms and Concepts

1.1 Region of Interest:

Defining a region of interest (ROI) is a crucial concept in image processing as it allows for multiple manipulation on the same image. The ipToolBox allows user to define more than one ROI per image with different parameters. The information about each ROIs were provided in a simple text file, which included the center pixel location, horizontal stretch, vertical stretch and other operation specific parameters (threshold, window size, etc).

1.2 Gray Threshold:

Gray Thresholding is a Point type image operation that follows simple binarization concept, where each pixel of an image is compared with the user specified threshold. The pixels above the threshold value are changed to white color and the rest to black. Gray Threshold takes on $O(N^2)$, since it only involves traversing through a 2D array of image pixels.

$$\text{if } (I_{(j,k)} > T) \quad \hat{I}_{(i,j)} = 255;$$
$$\text{else} \quad \hat{I}_{(i,j)} = 0;$$

1.3 Arithmetic:

Arithmetic is also a Point type of image manipulation. This type of functions involves simple mathematical operation on individual pixel, such as increasing or decreasing the intensity by the specified value. This algorithm also leads to N^2 operations.

$$\hat{I}_{(i,j)} = I_{(j,k)} \pm a ;$$ (a is a user specified value)

1.4 Color Thresholding:

Color Thresholding is different than Gray Thresholding due to its requirement of dealing with all three RGB channels. For our assignment, we were to calculate the distance between each pixel in ROI and user specified color C. The regions are then binarized by comparing calculated regions with the provided threshold. Gray and Color Thresholding are examples of fixed variable threshold algorithm. In such algorithms, no information from neighboring pixels is required and thus have no impact on the resulting output.

Distance $(D_{(j,k)}) = \sqrt{(I_{(j,k)red} - C_{red})^2 + (I_{(j,k)green} - C_{green})^2 + (I_{(j,k)blue} - C_{blue})^2}$

$$\text{If} \quad (D_{(j,k)} > T) \quad \hat{I}_{(i,j)} = 0;$$
$$\text{else} \quad \hat{I}_{(i,j)} = 255;$$

Adaptive Thresholding: This algorithm is differs from the others in its categorization of operation type. Unlike the once discussed above, Adaptive Thresholding relies on neighboring pixel information to calculate the threshold. The threshold equation is as follows,

```
Threshold = window_mean(WS)*W + T
```

where, WS, W and T are user defined window size, weight and threshold, respectively.

The window size provided should be an odd number, as it makes it possible to define the given pixel as the center of the window. The mean can be calculated by first initiating two loops to add values of each pixel in

```
I (j – w/2, k – w/2) to I (j + w/2, k + w/2)
```

and then dividing it by w^2 (total pixels in window).

The threshold is then used for normal binarization in given regions. This step still takes on $O(N^2)$, but the windows_mean function adds M^2 to the function run time. Since window mean is calculated for each pixel, the whole function leaves us with $O(N^2M^2)$. However, we can do better by dividing the window_mean function into two 1-D operations (first row-wise then column-wise).

$$I * [Arr1]_{(j,1)} \rightarrow I2 * [Arr2]_{(1,k)} \rightarrow \hat{I}$$

Thus leaving us with $O(N^2M)$ for the whole operation.

2. Implementation

Important characteristics of ipToolBox include:

- ability to process multiple images at once
- ability to accept multiple ROIs with different parameters for each ROI
- error checking
- execution on UNIX environments (C++)

Batch Processing:

In order to allow for multiple image processing, simple shell scripts were written for each filter. The shell scripts iterate over each .ppm or .pgm (depending on the filter) in a given folder and outputs the results.

ROI Class

The ROI file for each filter has to be defined differently (refer to README for detailed information regarding the format of each). In order to hold the information regarding each ROI, a class containing array of structs was implemented. Not only does this allow to hold information regarding different type of ROIs, but it also allowed for direct access to each piece of information, thus making function implementation efficient.

Errors

The program also contains basic error checking for input validation, parameter validation and inbound checks. However, it is required that all the arguments in ROI files should be in the format described in README.

Unix Compilation

To compile this program on UNIX system,

1) browse to the project directory
2) execute `sh compile.sh`
3) Depending on the filter execute `sh exeme_FILTER.sh ipToolBox`
   (FILTER : GrayThresh, ColorThresh, AdaptThresh, Arithmetic
4) Information regarding output files is displayed.

Example sh file

exeme.sh

```
sh compile.sh
sh exeme_GrayThresh.sh ipToolBox
sh exeme_ColorThresh.sh ipToolBox
```

Note: Must change directories in exeme_FILTER.sh to point to the pictures.

3. Results

3.1 GrayThresh

Figure 1 shows the original and after product of GrayThresh filter on a grayscale image. As its visible and expected, with increasing threshold, the number of black pixels outputted also increases. The advantage of allowing multiple regions with different threshold sizes is clearly visible through this example of a microscopic image. The ROI file for this filter should be in following format:

(x, y, sx, sy, Threshold)

3.2 Color Threshold

Figure 2.b contains two color threshold regions. As we can see that when a high contrast image is compared to high contrast pixel, the segmentation becomes sharper as the provided distance increases. White regions in original image are more darkened in the resulting image. The ROI file format for color thresh is

(x, y, sx, sy, Distance, red, green, blue)

Figure 1.a: Original

Figure 1.b: Gray Thresholding.
(TOP: left = 25, right = 75,
BOTTOM: left = 125, right = 200)

Figure 2.a: Original Color

Figure 2.b: Color Thresholding.
(LEFT: D=100, Color (255 ,155, 0)
(RIGHT: D=200, Color (255 ,155, 155)

3.3 Adaptive Threshold

ColorThresh and GrayThresh are both examples of fixed threshold algorithms. Adaptive Threshold on other hand is a variable threshold algorithm, because for every pixel in the given ROI there is a different threshold value. Using such kind of algorithm allows for image smoothing, since noise at certain pixel can be reduced by using information from neighboring pixels. Another application of the adaptive threshold is that of edge detection. Setting the window size to a reasonably large number, would cause the edge to differ from the neighboring pixels, thus generating a white border around the object. Using such algorithms can help identify specific regions in images, especially when working with microscopic organisms.

ROI file format: (x, y, sx, sy, T, winsize, weight)

3.4 Arithmetic

Like GrayThresh, Arithmetic is a simple filter that increments the intensity of each pixel with the value provided by user. In the given sample, 4 different arithmetic are applied to different regions of the image. With increasing value the image becomes brighter and in the last box its almost completely white, reaching maximum value of 255.

ROI file format: (x, y, sx, sy, value)

Figure 3.a: Original Color

Figure 3.b: Color Thresholding.
(TOP: left=25, right =75)
(BOTTOM: left=175, right = 200)

4. Conclusion

Implementing ipToolBox has helped me understand the basic image manipulation techniques and algorithms involved in it. Even though the framework for the assignment was provided, understanding how the image storage works and the data structures required for it, proved to be one of the most crucial tasks of this assignment.

Understanding how such operations are implemented also helps in developing their applications. Using these functions to brighten/darken an image, perform edge detection, image smoothing and extracting important information from an 2D image is now possible.

Exploring how these functions perform in 3D space (movie) is the next step.