Assignment1 : Adaptive thresholding in a ROI for gray scale and color images

John Doe
Department of Computer Sc. and Engg.
University of South Florida, Tampa, Florida, USA

1. Introduction and overall description

This assignment focuses on eff cient implementation of image manipulation techniques like thresholding in gray scale and color images[1, 2]. Image thresholding/binarization compares image pixel value with a user def ned value. It is naturally associated with segmentation of a scene into foreground and background. Conventially, all foreground pixels are set to white and all background pixels are set to black or vice versa. As often knowledge about the position of a object(foreground) is known beforehand, supporting a region of inerest(ROI) and operating within the ROI is provided in this assignment. The report is organized as follows. In Section 2, the basic algorithms used in this assignment are described. Section 3 describes the implementation details of the assignment. In Section 4 the results for the assignment is presented and f nally we conclude in Section 5.

2. Description of algorithms

In this section the basic algorithms used in this assignment are described. First the algorithm of thresholding in gray scaleimages is described. The algorithm provided is that of a adaptive thresholding and hence takes care of image characteristics in a local neighbourhood. Also as this process is computationally expensive a incremental technique is used for computation and hence the algorithm runs in a time proportional to the input image. A algorithm for thresholding in color images is described next.

2.1. Adaptive Thresholding

Binarization of an image into forground and background based on a single user def ned threshold is usually diff cult. This is because of illumination variation across the object leading to noise in the true intensity value of the object. Also guessing a optimum threshold value is diff cult for the user. Hence adaptive thresholding by considering the local statistics of a pixel in its neighbourhood is expected to improve segmentation. Thus given a window size $W \times W$ and a user def ned value T, the adaptive threshold is calculated as $threshold = mean_{window}(W) + T$. Offcourse, the operation is performed inside a used-def ned ROI specif ed by the postion of the top corner of the ROI (X, Y) and size (S_x, S_y).

Computation of the window mean (averaging) is a separable operation. Hence the sum of all pixel values in the window can be computed by f rst computing the row-wise sum of each row inside the window mask for the entire image and then adding the resultant in a column wise manner inside the window. Thus for a window size of $M \times M$, this leads to $2M$ operations which is much better than the brute force method consisting of N^2 operations. We can actually do better than this by considering that the consecutive sum can be obtained from the previous sum by subtracting the value of the left most pixel value in the previous window and adding the right-most pixel value in the current window for the separable row operation. Similarly, the column operation on the result can be computed by subtracting the value of the top most pixel value in the previous window and adding the bottom-most pixel value in the current window. Thus each 1-dimensional operator takes 2 operations for calculating the new mean from the previous mean (leading to 4 operations for each mean from the previous mean). It is to be noted that this is independent of the window size. Hence the incremental approach is must better computationally than the predecessors described.

As at the corners of the ROI where window-mean cannot be determined because the window goes out of the ROI, we set mean at those pixel locations to the global mean in the ROI.

2.2. Color Image Thresholding

Thresholding in color images is complicated by the fact that human perception of color is non-linear and does not follow the triplet R-G-B form traditionally used for display of color images. Hence simple thresholding on each component does not give correct results. Color schemes like YUV, and HSI models have been developed that model the human perception of color. Hence ideally, thresholding a color image should first lead to a conversion of the image from RGB to HSI domain and then using the intensity component of the HSI domain for thresholding. But as RGB to HSI is computationally very intensive because of non-linear trigonometric functions, in this algorithm a different approach is proposed. Here we accept from user a color value C and distance from this color. Thus if the color value of the pixel is at a higher distance than this from the user defined color C, we classify it as foreground else it is a background pixel. We can think of it as considering all pixels as foreground, which have color values falling in the sphere centred on the user defined color C in the 3-dimensional RGB color cube.

3. Description of implementation

The entire code is developed in C++ language on a solaris workstation. The code for reading and writing of images was provided as part of a Image class file. This Image class file is enhanced by adding member functions for image scaling and thresholding. The frontend just creates a Image object, initializes the object with data from images and calls appropriate member functions. The member functions return a resultant Image object which is obtained after operating on the input image object, which is saved for offline viewing. The code reads a parameter file containing the input and output image names, the function to be performed on it and also the ROI selection by specifying pixel location (X, Y) and ROI size (S_x, S_y). For grey scale thresholding operation the parameter file provides the user defined values T and the symmetric window size W. For color thresholding the parameter file provides a triplet representing the user defined color C and distance d from this user defined color. The format of the parameter file required for supporting the functionality of the assignment is provided in the submitted ”readme” file.

4. Description and Analysis of Results

4.1. Description of Results

This section illustrates the results of the algorithms used. The results for grey scale adaptive thresholding is shown first and is compared with normal thresholding. Then the results of the color thresholding is provided on sample images.

Figures [3-4] show the results of adaptive thresholding on grey scale images shown in Figure 1. We can see as window sizes are made larger better segmentation is obtained

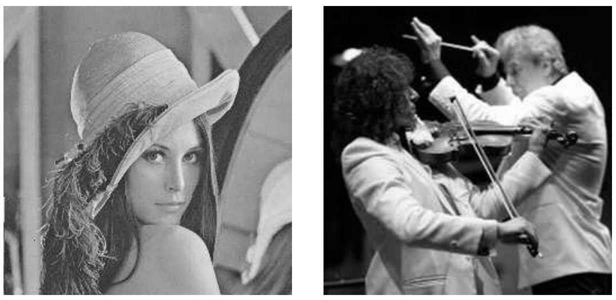

Figure 1. Original images used for experiments on Adaptive Thresholding on Gray Scale Images

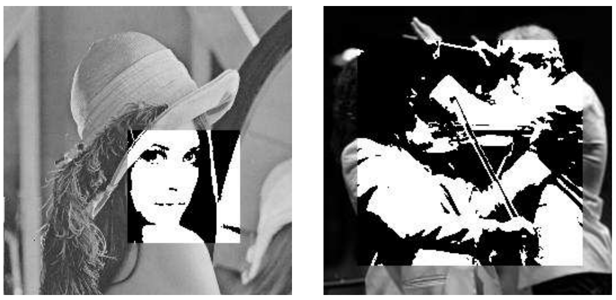

Figure 2. Results of normal thresholding on the ROI for images in Figure 1 with threshold value of 128

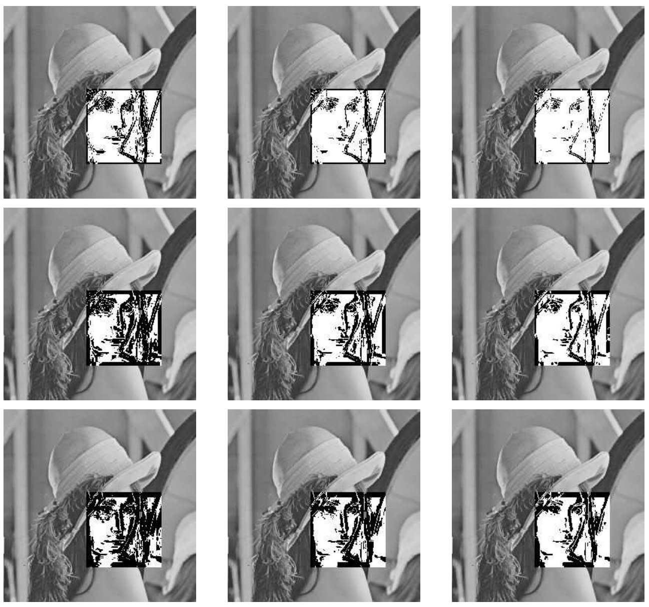

Figure 3. Top row shows adaptive thresholding for window size 5 with thresholds in 10, 15 and 20 for left image of Figure 1. Middle row shows the result for window size 10. Bottom row shows results for window size 15.

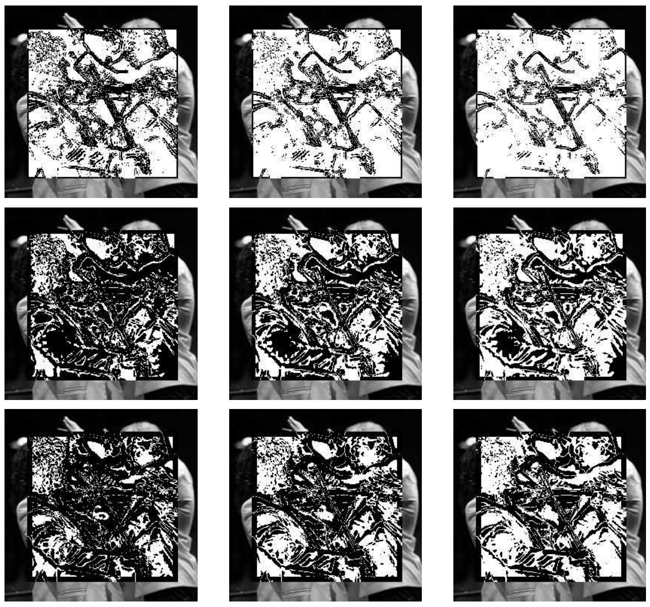

Figure 4. Top row shows adaptive thresholding for window size 5 with thresholds in 10, 15 and 20 for right image of Figure 1. Middle row shows the result for window size 10. Bottom row shows results for window size 15.

for relatively large objects. Comparing adaptive thresholding with global thresholding using user defined value provides comparable or better results for the ”violin” image. However much better segmentation is obtained is obtained using adaptive thresholding for the ”lenna” image as much greater detail is obtained in the binarized ROI of the face. Thus the ability of adaptive thresholding producing better segmentation does depends upon the image.

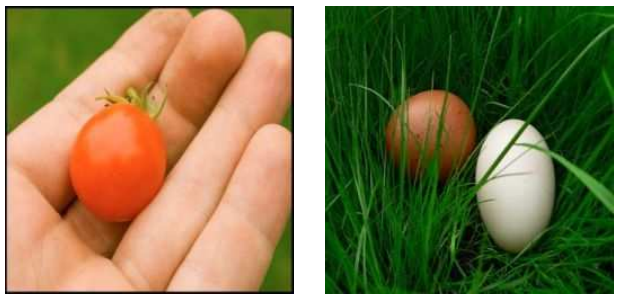

Figure 5. Original images used for experiments of Thresholding on Color Images

Figure [6] shows results of color thresholding on images in Figure [5]. As seen segmentation does improve if threshold value of distance increases till a range. After that segmentation quality greatly decreases. This is expected as we are using a sphere in the RGB color space for thresholding,

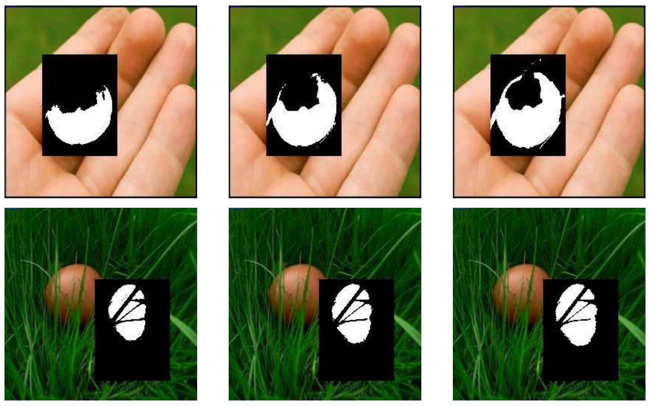

Figure 6. Top row shows color thresholding with distance in 40, 50 and 60 for left image of Figure 1 for user provided value of (225,65,20) to segment orange color tomato. Bottom row shows color thresholding with distance in 40, 50 and 60 for left image of Figure 1 for user provided value of (225,225,225) to segment the white egg.

but actually the shape is non-symmetric and complicated as human perceptual model is related non-linearly to RGB color space.

4.2. Performance Evaluation and Analysis of Results

Basic thresholding in ROI offcourse work in $O(N^2)$. Normal adaptive thresholding redoes calculation and calculates mean repeatedly which runs in $O(N^2M^2)$ where M is the size of the window. However as we use a two pass incremental separable operation for mean calculation as described before, the computation is of time $O(N^2)$ again. (Here we mean N is the size of the image). As at the corners

Adaptive thresholding does improve segmentation quality in some images but in general is not always superior to user defined basic thresholding. As at the corners of the ROI where window-mean cannot be determined because the window goes out of the ROI, we set mean at those pixel locations to golbal mean in the ROI. Hence as window sizes become large a false border inside the ROI is often observed. Color thresholding produce good segmentation only within small distances from user specified color in the 3-dimensional RGB color space.

5. Conclusions

In this assignment adaptive thresholding was performed to improve segmentation on images with varying illumination. Particular attention was paid to make Adaptive thresholding work in $O(N^2)$. This would signif cantly boost performance of segmenation when N or M is large and particularly when large number of images are to be processed. The color thresholding method of selecting pixels in the 3D neighbourhood of the user specif ed color works for a limited range of distances after which the human perceptual model can no longer be approximated. This technique is quite suitable close disntances and neighbourhood colors as the non-linear transformation from RGB to HSI space is avoided here.

References

[1] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Pearson Education, 2002.

[2] V. H. M. Sonka and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Brooks/Cole, 1998.