CAP- 5400

DIGITAL IMAGE PROCESSING
FALL '09

ASSIGNMENT 1

SATRAJIT BASU
U-09538502

DATE: SEP. 22, 2009

Content:

Introduction:

The objective of the assignment was to implement the basic image manipulation techniques and experiment with image binarization for both grey-level as well as color-images.

Thresholding / Binarization:

Thresholding, also known as binarization, involves reading each pixel of the image, comparing it with a clearly defined threshold. If less, it replaces the intensity with zero, else with one. Threshold usually, comes from a visualizing the histogram of the image. Thresholding is especially useful if there is a foreground object and the rest of the be eliminated. If the histogram has a clear peak for the foreground, it can be easily segmented. Thresholding finds use in reading and manipulation of medical images such as CT scans and MRI.

Variable Thresholding:

It is difficult to Binarize an image into foreground and background based on a single user defined threshold. This is because of the illumination variations across the object which leads to noise in the true intensity value of the object. Also it is difficult for the user to guess a value for the threshold. Hence arises the concept of Variable Thresholding. It involves considering the local statistics of a pixel in its neighborhood to improve segmentation. Hence for a given window size $W \times W$ and a user defined threshold value T, the Variable Threshold is calculated as threshold = window_mean(W) + T . The operation in this assignment is performed within a user-defined ROI. The ROI is specified by the position of the top corner of the ROI (X, Y) and size (Sx, Sy).

Color Binarization:

A Color images is represented using three bytes per pixel, with each byte corresponding to one of the three primary colors- Red, Green and Blue. In general, operations for gray-scale images are implemented for color images by individually applying it to each of the bytes of the color image. But this cannot be done for thresholding operations, because adopting that would lead to falsely bringing out an edge where there is a lack of a particular color. Hence thresholding of color images is done using the distance of a given pixel from a user specified color. The distance of the pixel from the user defined color is calculated and if the result is found to be within the threshold, the pixel is considered to be in the foreground and is set to 1.Pixels, which fall outside this color space, are set to 0 and is considered to be the background. For an user defined threshold TC and an user defined color C in RGB space, all pixels within TC distance is set to white and the rest are set to black.

Let,
User defined threshold = TC; User defined color C=(Cr,Cg,Cb) ; Pixel color P=(R,G,B);
Distance is calculated as follows:
Distance=|Cr-R|+|Cg-G|+|Cb-B|

```
IF   Distance<TC
    P= (255,255,255)
ELSE
    P= (0, 0, 0)
```

Smoothing:

Image smoothing is a set of local pre-processing methods whose predominant use is the use suppression of image noise. Smoothing makes use of redundancy in the image data. The new value is calculated by averaging the brightness values in some neighborhood. Smoothing can often lead to blurring of sharp edges of an image. Hence edge-preserving smoothing is considered. Edge-preserving smoothing is based on the idea that the average is computed only from those points in the neighborhood which have similar properties to the point being processed. Consider the following conditions:

Edge-preserving uniform smoothing of “size” SM is being applied to a grey-scale image. The user defined threshold is TSM. Then for SM=3

1	**1**	**1**
1	**1**	**1**
1	**1**	**1**

Uniform Smoothing window

a00	**a01**	**a02**
a10	**a11** **f(i,j)**	**a12**
a20	**a21**	**a22**

image

Calculating average:

g(i,j)=int(1*a00+1*a01+1*a02+1*a10+1*a11+1*a12+1*a20+1*a21+1*a22)/9

Then Binarizing using:

```
IF |f (i,j)-g(i,j)|<TSM
        f (i,j)=g(i,j)
ELSE
        f (i,j)=f(i,j)
```

Implementation:

a. **main( ) function:** In the main function of the provided framework for the assignment, I have implemented function calls to the class ipTools. In the main( ) the argument variable is used to identify the operation to be performed on the image. argv[1] value is compared to choose the type of operation for a particular image.

b. **The functions called in the ipTools class are:**

OutputROI( ): parameters: source image, roi file, target image.
this function scans the entire image and places only those pixels in the target image which falls within the roi parameter file which contains the pixel and the size of the region of interest.

binarize( ): parameters: source image, target image, threshold value and roi file.
the binarize function performs basix thresholding operation on the image within the region of interest.

GreyThresh( ): parameters : source image, target image, threshold, windowsize
this function performs variable thresholding on gray-scale images.

Colorbinarize( ): parameters: source image, target image, threshold, roi file, user defined color (Cred, Cgreen, Cblue)
This function performs calor thresholding on color images (*.ppm)

Smoothing( ): parameters: source image, roi, taget image, threshold and size
this function performs edge-preserving smoothing operation on grey-map images.

In this implementation only a single operation is performed on the image at a time in all the ROIs. Though it has to be said further manipulation of the same program can be done to achieve implementation of different operations in different ROI for the same image.

OUTPUT:

Grey-level Thresholding:

	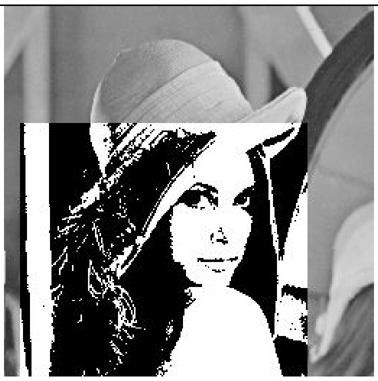	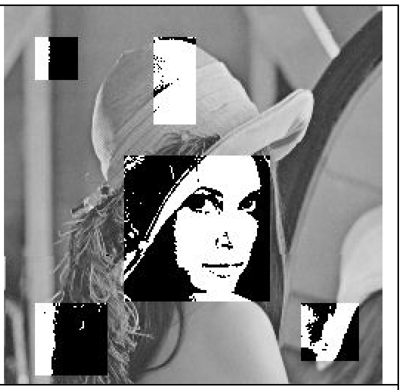
a.original image	**b. binarization with threshold 140**	**c. in 5 different ROI**

Fig. 1. Simple Thresholding

In simple grey-level thresholding the grey-scale image is binarized by comparing pixel values with the user defined threshold value, which in this case is 140. First output image(b) shows the effect of binarization in a single ROI. The second output image(c) shows the same operation been done on 5 different ROIs.

Variable Thresholding:

In the implementation of variable thresholding the window size and threshold was provided by the user. The mean of the pixel values within the window were compared with the user defined threshold. If the value exceeded the threshold the pixel value was put as black else it was made white.

The resultant output of this operation is given in Fig. 2

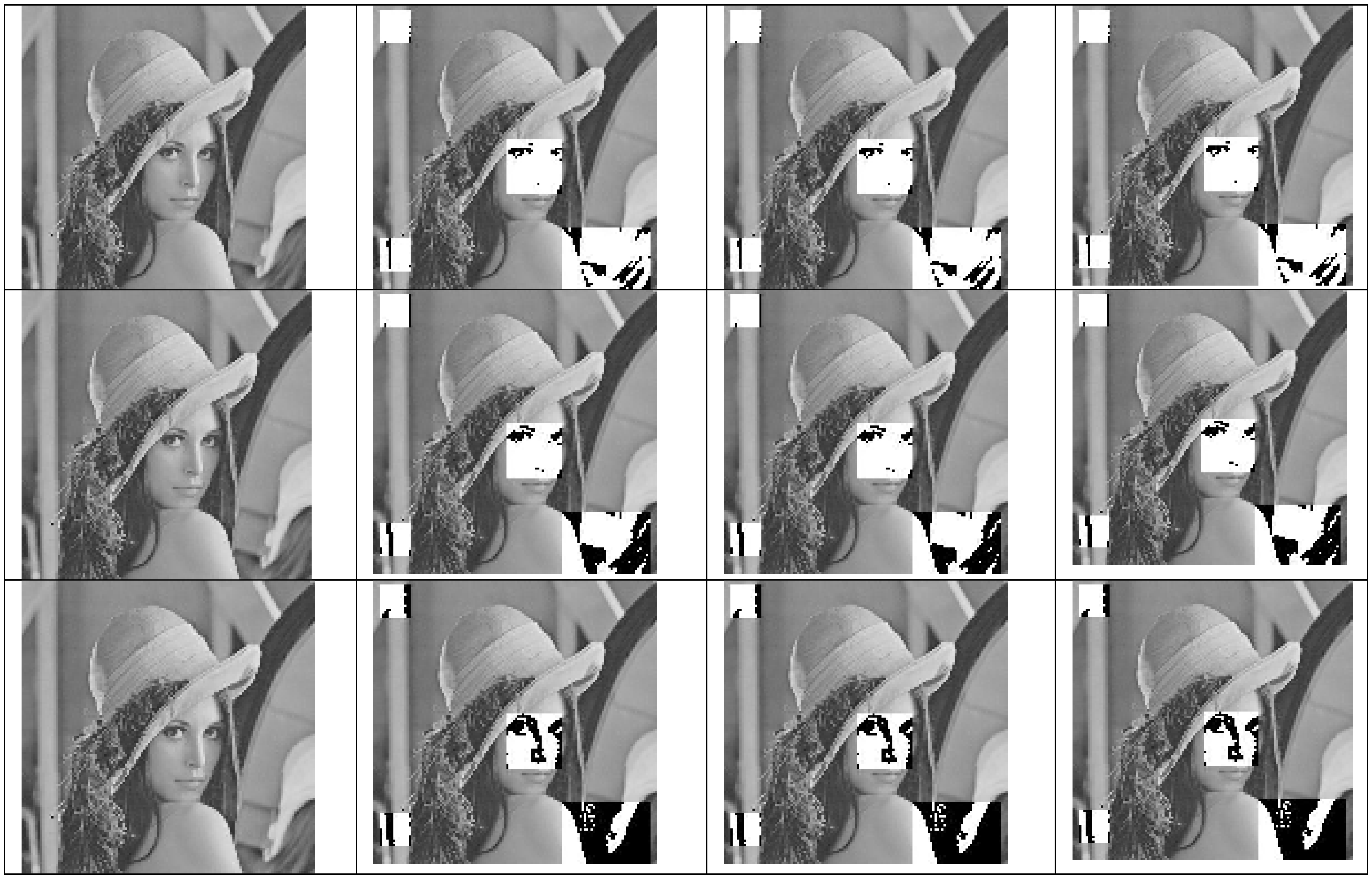

Fig.2. Variable Thresholding for grey-scale images.

The results above show Variable-Thresholding on the same image and same ROIs for three different threshold values and three different window sizes.

Top Row:	Threshold = 100; Window Size = 3, 5, 8
2nd Row:	Threshold = 120; Window Size =3, 5, 8
3rd Row:	Threshold=150; Window Size = 3, 5, 8

Color Binarization:

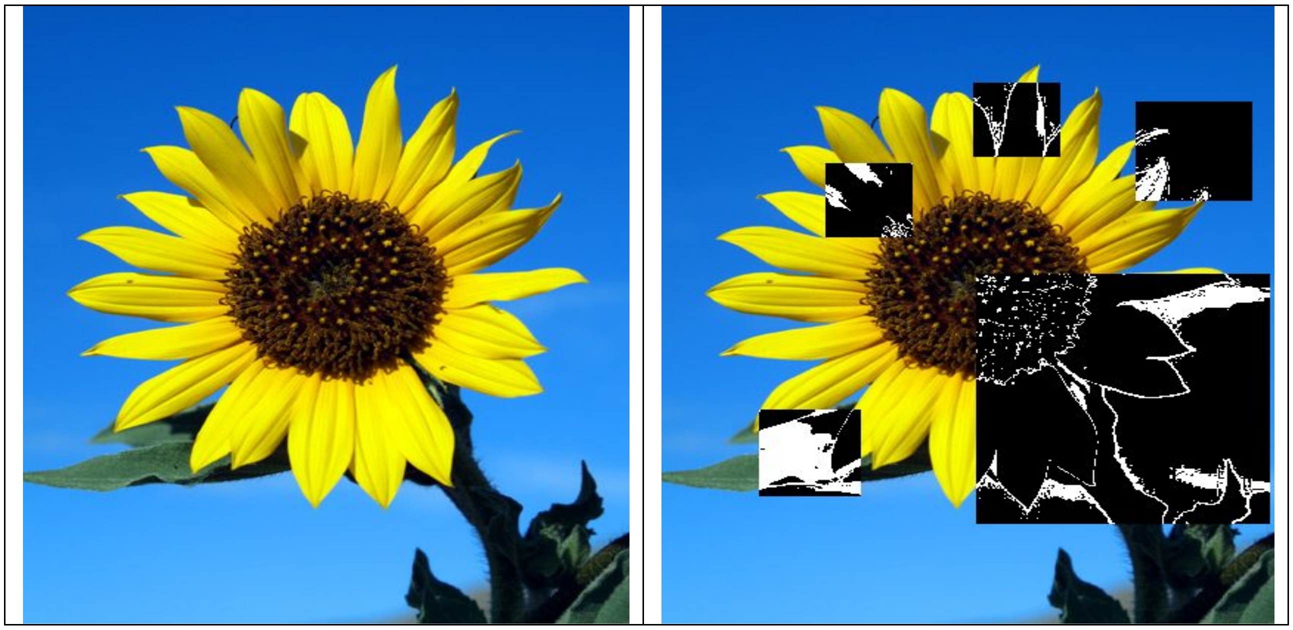

Fig. 3. Color Binarization

In Color-Binarization, the distance of a pixel from the user defined color is first calculated. The absolute value of the distance if less than the user defined threshold leads to the pixel being assigned (255,255,255). Else the pixel is set the value (0,0,0).

Edge- Preserving Smoothing:

	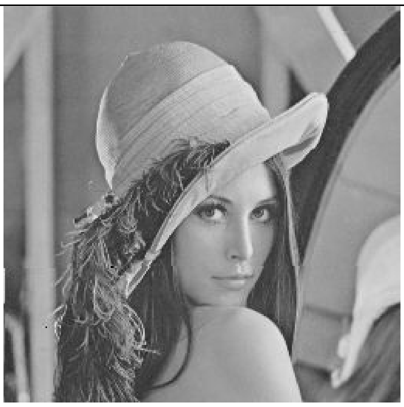	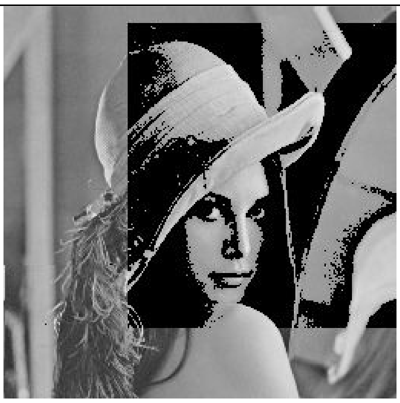
Original Image	**Smoothing with threshold 15**	**Smoothing with threshold 150**

Fig.4. Edge-Preserving smoothing

Edge-preserving smoothing is implemented here by choosing a sliding window the size of which is user-defined. The approximation made here is that for pixels, which are at a distance less than half the window size from the edge of the ROI, are not smoothed. Smoothing is done only on pixels within the ROI on which the smoothing window can be placed without it being outside the ROI. Better implementation of Edge preserving smoothing can be done using the likes of Directional Smoothing. Also the smoothing done on the image chosen here is not very prominent for lower values of threshold. There should be other images on which this operation can be carried out to get better results.

References:

1> Image Processing, Analysis and Machine Vision, Milan Sonka et al