Frankfurt University of Applied Sciences

A Project Report

On

Binarization Filter

Under the Guidance of Mr. Damir Dobric

Presented by,

Suraj Bharadwaj

Mat.Nr: 1224681

CONTENTS

1. Introduction

1.1 What is Binarization?

Binarization is the process of converting a pixel image to a binary image.
Separating the foreground out from the background of an image is an important preprocessing in image analysis. Its purpose is to acquire some useful information in the image for higher level image processing. Binarization or thresholding is such a widely-used method and generally, its process is to first determine a gray threshold according to some objective criteria and then assigns each pixel to one class (such as the foreground) if its gray level or gray value is greater than the determined threshold and otherwise to the other class (such as the background).

A binary image is produced by quantization of the image gray levels to two values, usually 0 and 1. Binarization can be used in recognizing text and symbols, e.g. document processing. Identifying objects with distinctive silhouettes, e.g. components on a conveyor in a manufacturing plant, and determining the orientation of objects are some other examples of binarization applications.

Binarization generally involves two steps including determination of a gray threshold according to some objective criteria and assigning each pixel to one class of background or foreground. If the pixels intensity is greater than the determined threshold then it belongs to foreground class and otherwise to the background.

The main problem in binarization is the choice of thresholding technique. The selection of most optimal binarization algorithm is difficult, because different binarization algorithm gives different performance on different data sets. This is especially true in the case of historical documents images with variation in contrast and illumination.

The algorithms divide into two categories:

a) Global Binarization- Uses single threshold value for whole image.
b) Local Binarization-The threshold value is calculated locally pixel by pixel or region by region.

1.2 Problem Description

Thresholding is a technique used for segmentation, which separates an image into two meaningful regions: foreground and background, through a selected threshold value T. If the image is a grey image, T is a positive real number in the range of [0,….,K], Where, K ∈R . So, thresholding may be viewed as an operation that involves tests against a value T. In this work we obtained the optimum value of T by comparing of two novels Binarization algorithms that we have considered. The segmentation procedure is represented by the following equation:

$$G_B(x,y) = \begin{cases} 1, & if\ G(x,y) > T \quad .....(1) \\ 0, & if\ G(x,y) \le T \quad .....(2) \end{cases}$$

G(x, y) represents the intensity value of pixel at (x, y) location in the grey image G, where, G∈ R. GB represents the obtained segmentation result such that GB ∈R .If GB (x, y) = 1, then pixel location (x, y) in the bimodal image G is classified as a foreground pixel, otherwise it is classified as a background pixel. By comparing Binarization algorithms assign the best optimum threshold value for T.

2. Project Description

2.1 Global Thresholding

Global binarization algorithms use a single global threshold value for a whole document image. The global threshold is used to separate image pixels and background image pixels of objects.
A sample image histogram is shown in Fig. 1. A sample image histogram with global threshold is shown in Fig. 2.

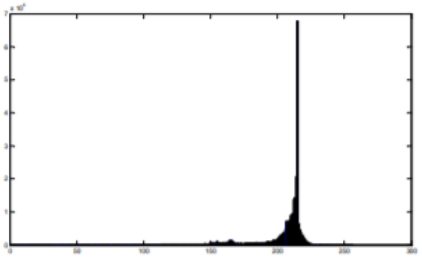


Fig. 1. A sample image histogram.

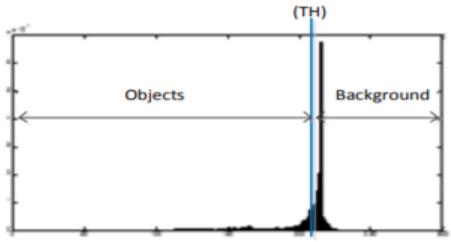


Fig. 2. A sample image histogram with global threshold.

The gray-level histogram is normalized by the following relation:

$$Z = \sum_{i=1}^{L} p_i = \sum_{i=1}^{L} \frac{n_i}{N} = 1, \qquad (1)$$

$$0 \le p_i \le 1 \quad and \quad i = 1,2,...,L,$$

where:

n_i - number of pixels at level i;
N - total number of pixels;
L - number of gray-levels, typically 255.

2.2 Local Thresholding

2.2.1 Otsu Threshold

Otsu's method is the most successful local thresholding method. It automatically performs histogram shape-based image thresholding for the reduction of a gray-level image to a binary image. The algorithm assumes that the image for thresholding contains two classes of pixels (e.g., foreground and background) and then calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal. It exhaustively searches for the threshold that minimizes the intra-class variance, defined as the weighted sum of variances of the two classes.

2.2.2 Steps in Otsu Threshold:

The weighted within-class variance is $\sigma^2_w(t) = q_1(t)\sigma^2_{\ 1}(t) + q_2(t)\sigma^2_{\ 2}(t)$ where the class probabilities of different gray level pixels are estimated as:

$$q_1(t) = \sum_{i=1}^{t} P(i) \qquad q_2(t) = \sum_{i=t+1}^{I} P(i)$$

And the class means are given by:

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \qquad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$$

Finally, the individual class variances are:

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

Now, we could actually stop here. All we need to do is just run through the full range of *t* values [1,256] and pick the value that minimizes

2.2.3 A Faster Approach

By a bit of manipulation, we can calculate what is called the *between class* variance, which is far quicker to calculate. Luckily, the threshold with the maximum *between class* variance also has the minimum *within class* variance. So it can also be used for finding the best threshold and therefore due to being simpler is a much better approach to use.

After some algebra, we can express the total variance as...

$$\sigma^2 = \underbrace{\sigma_w^2(t)}_{\text{Within-class, from before}} + \underbrace{q_1(t)[1-q_1(t)][\mu_1(t)-\mu_2(t)]^2}_{\text{Between-class,}}$$

Since the total is constant and independent of t, the effect of changing the threshold is merely to move the contributions of the two terms back and forth.
So, minimizing the within-class variance is the same as maximizing the between-class variance.
The nice thing about this is that we can compute the quantities in σ^2 B(t) recursively as we run through the range of t values.

3. Enhancement Techniques:

3.1 Median Filter

Median filtering is a nonlinear method used to remove noise from images. It is widely used as it is very effective at removing noise while preserving edges. It is particularly effective at removing 'salt and pepper' type noise. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image).

The median filter works by moving through the image pixel by pixel, replacing each value with the median value of neighboring pixels. The pattern of neighbors is called the "window", which slides, pixel by pixel over the entire image 2 pixel, over the entire image.
The median is calculated by first sorting all the pixel values from the window into numerical order, and then replacing the pixel being considered with the middle (median) pixel value**.**

If the window has an odd number of entries, then the median is simple to define: it is just the middle value after all the entries in the window are sorted numerically.
For an even number of entries, there is more than one possible median.

Typically, by far the majority of the computational effort and time is spent on calculating the median of each window. Because the filter must process every entry in the signal, for large signals such as images, the efficiency of this median calculating is a critical factor in determining how fast the algorithm can run.

3.1.1 Principle of Median Filter

The source code defines the Median Filter extension method targeting the Bitmap class. The matrix Size parameter determines the intensity of the Median Filter being applied.

The Median Filter extension method iterates each pixel of the source image. When iterating image pixels we determine the neighboring pixels of the pixel currently being iterated. After having built up a list of neighboring pixels, the List is then sorted and from there we determine the middle pixel value. The final step involves assigning the determined middle pixel to the current pixel in the resulting image, represented as an array of pixel color component bytes.

3.1.2 Applying a Median Filter

The median filter like mean filter is also a sliding-window spatial filter, but it replaces the center value in the window with the median of all the pixel values in the window. As for the mean filter, the kernel is usually square but can be any shape. An example of median filtering of a single 3x3 window of values is shown below:

unfiltered values

6	2	0
3	97	4
19	3	10

In order: 0, 2, 3, 3, **4**, 6, 10, 15, 97

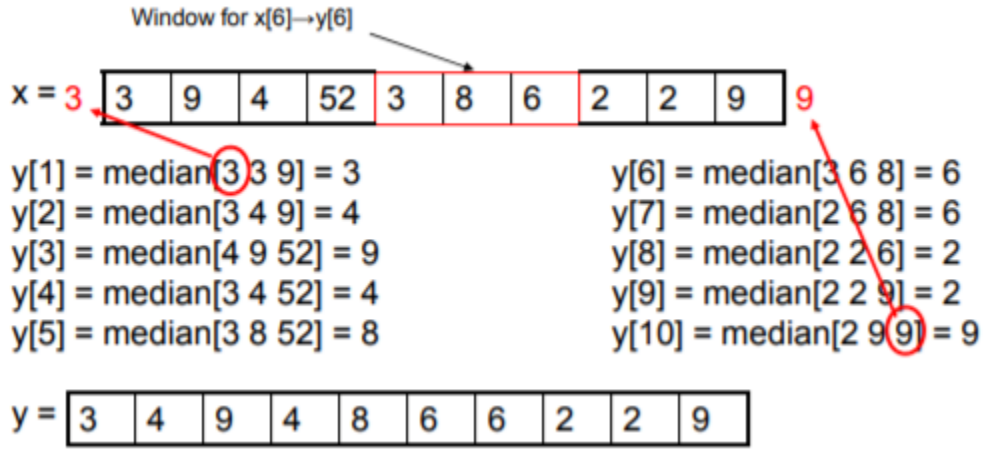
median filtered

*	*	*
*	4	*
*	*	*

Center value (previously 97) is replaced by the median of all nine values (4).

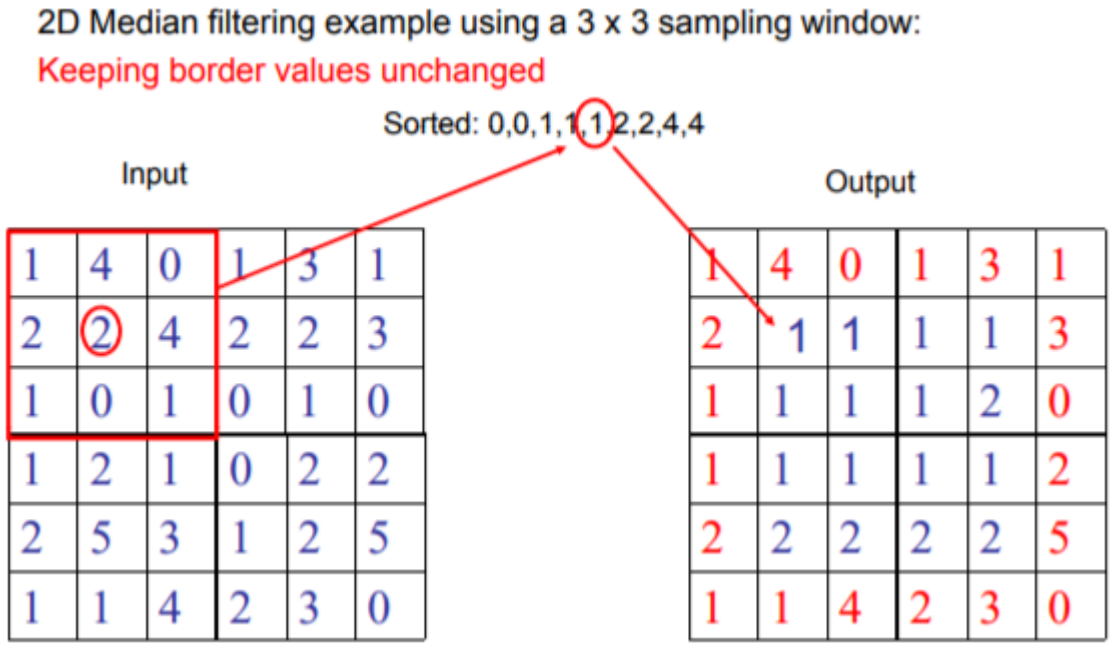
Note that for the first (top) example, the median filter would also return a value of 5, since the ordered values are 1, 2, 3, 4, **5**, 6, 7, 8, 9. For the second (bottom) example, though, the mean filter returns the value 16 since the sum of the nine values in the window is 144 and 144 / 9 = 16. This illustrates one of the celebrated features of the median filter: its ability to remove 'impulse' noise (outlying values, either high or low). The median filter is also widely claimed to be 'edge-preserving' since it theoretically preserves step edges without blurring. However, in the presence of noise it does blur edges in images slightly.

The following second example shows the application of a median filter to a simple one dimensional signal. A window size of three is used, with one entry immediately preceding and following each entry.



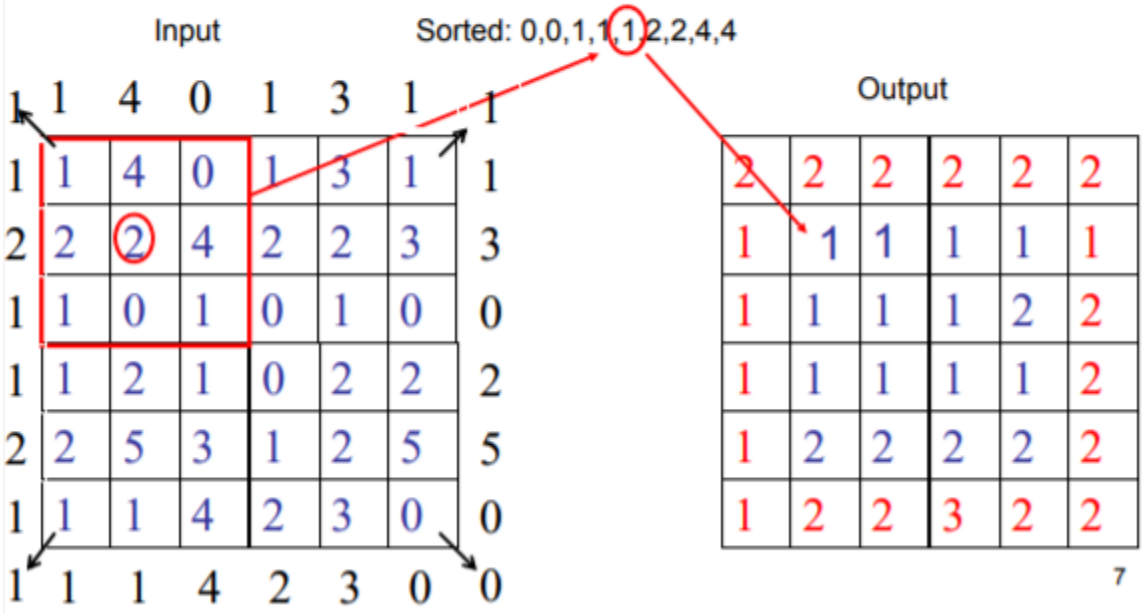
For y[1] and y[9], extend the left-most or right most value outside the boundaries of the image same as leaving left-most or right most value unchanged after 1-D median.

In the previous example, because there is no entry preceding the first value, the first value is repeated (as is the last value) to obtain enough entries to fill the window



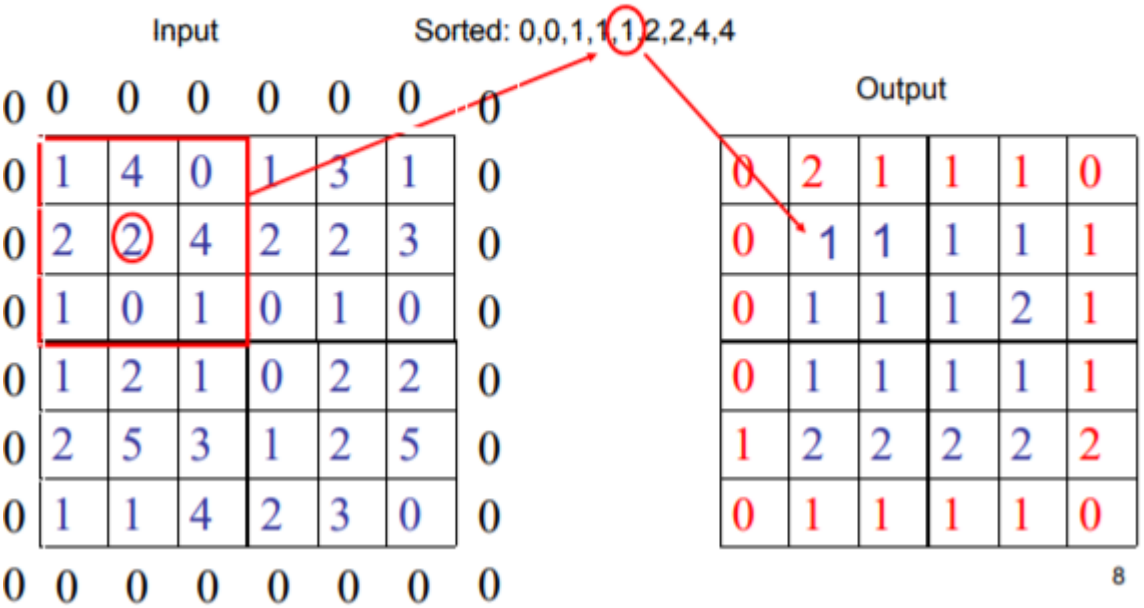
2D Median filtering example using a 3 x 3 sampling window:

Extending border values outside with values at boundary



2D Median filtering example using a 3 x 3 sampling window:

Extending border values outside with 0s



3.2 Convolution Filter

Convolution is the treatment of a matrix by another one which is called “kernel”. The Convolution Matrix filter uses a first matrix which is the Image to be treated. The image is a bi-dimensional collection of pixels in rectangular coordinates. The used kernel depends on the effect you want

3.2.1 Principle of Convolution Filter

The filter studies successively every pixel of the image. For each of them, which we will call the “initial pixel”, it multiplies the value of this pixel and values of the 8 surrounding pixels by the kernel corresponding value. Then it adds the results, and the initial pixel is set to this final result value.

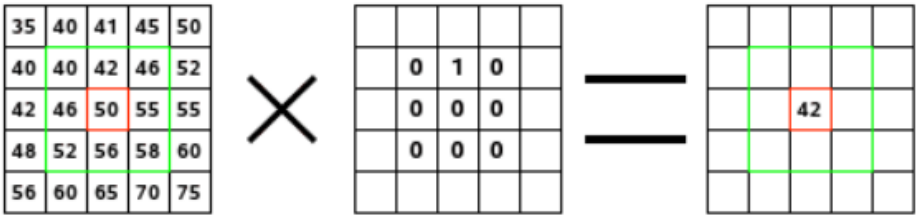
- Convolution is a general purpose filter effect for images.
- Is a matrix applied to an image and a mathematical operation comprised of integers.
- It works by determining the value of a central pixel by adding the weighted values of all its neighbors together.
- The output is a new modified filtered image
- A convolution is done by multiplying a pixel’s and its neighboring pixels color value by a matrix
- Kernel: A kernel is a (usually) small matrix of numbers that is used in image convolutions.
- Differently sized kernels containing different patterns of numbers produce different results under convolution.
- The size of a kernel is arbitrary but 3x3 is often used

Why convolve an image?

Blurring, Smoothing, Edge Detection, Sharpening and Embrossing. The resulting filtered images still bears a relation to the input source image.

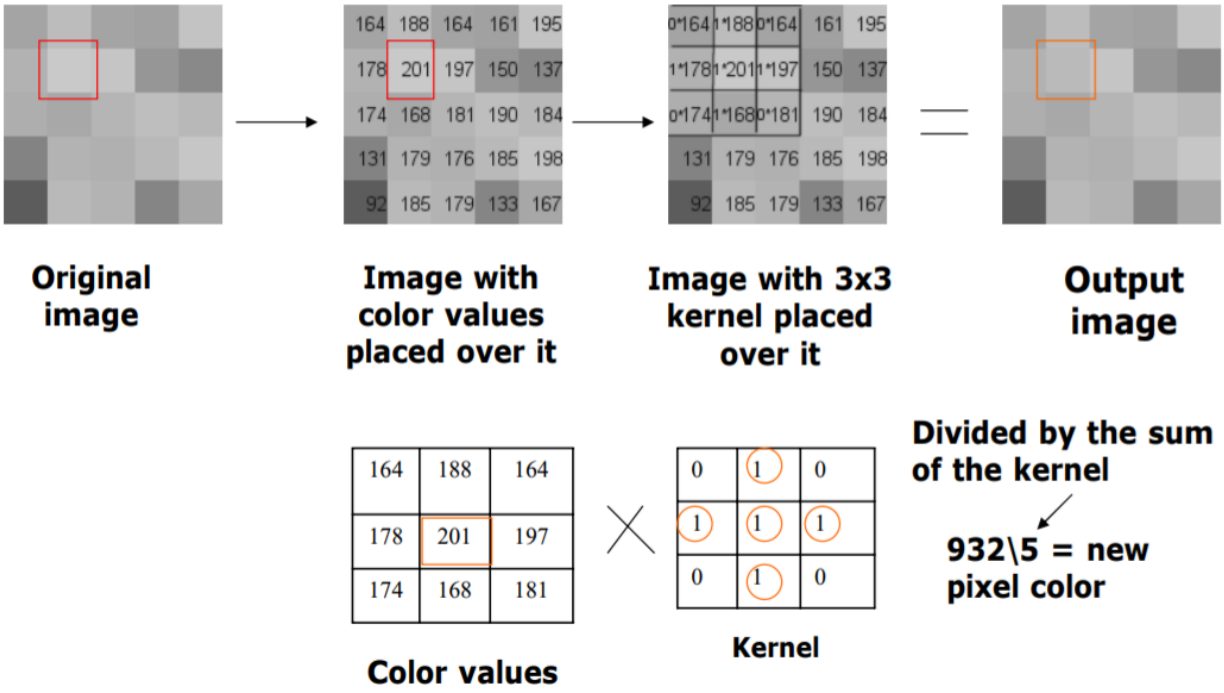
3.2.2 Applying Convolution Filter

A simple example:



On the left is the image matrix: each pixel is marked with its value. The initial pixel has a red border. The kernel action area has a green border. In the middle is the kernel and, on the right is the convolution result. Here is what happened: the filter read successively, from left to right and from top to bottom, all the pixels of the kernel action area. It multiplied the value of each of them by the kernel corresponding

value and added results. The initial pixel has become 42: (40*0)+(42*1)+(46*0) + (46*0)+(50*0)+(55*0) + (52*0)+(56*0)+(58*0) = 42. (The filter doesn't work on the image but on a copy). As a graphical result, the initial pixel moved a pixel downwards.



Kernel Examples:

0	0	0
-1	1	0
0	0	0

Edge enhance

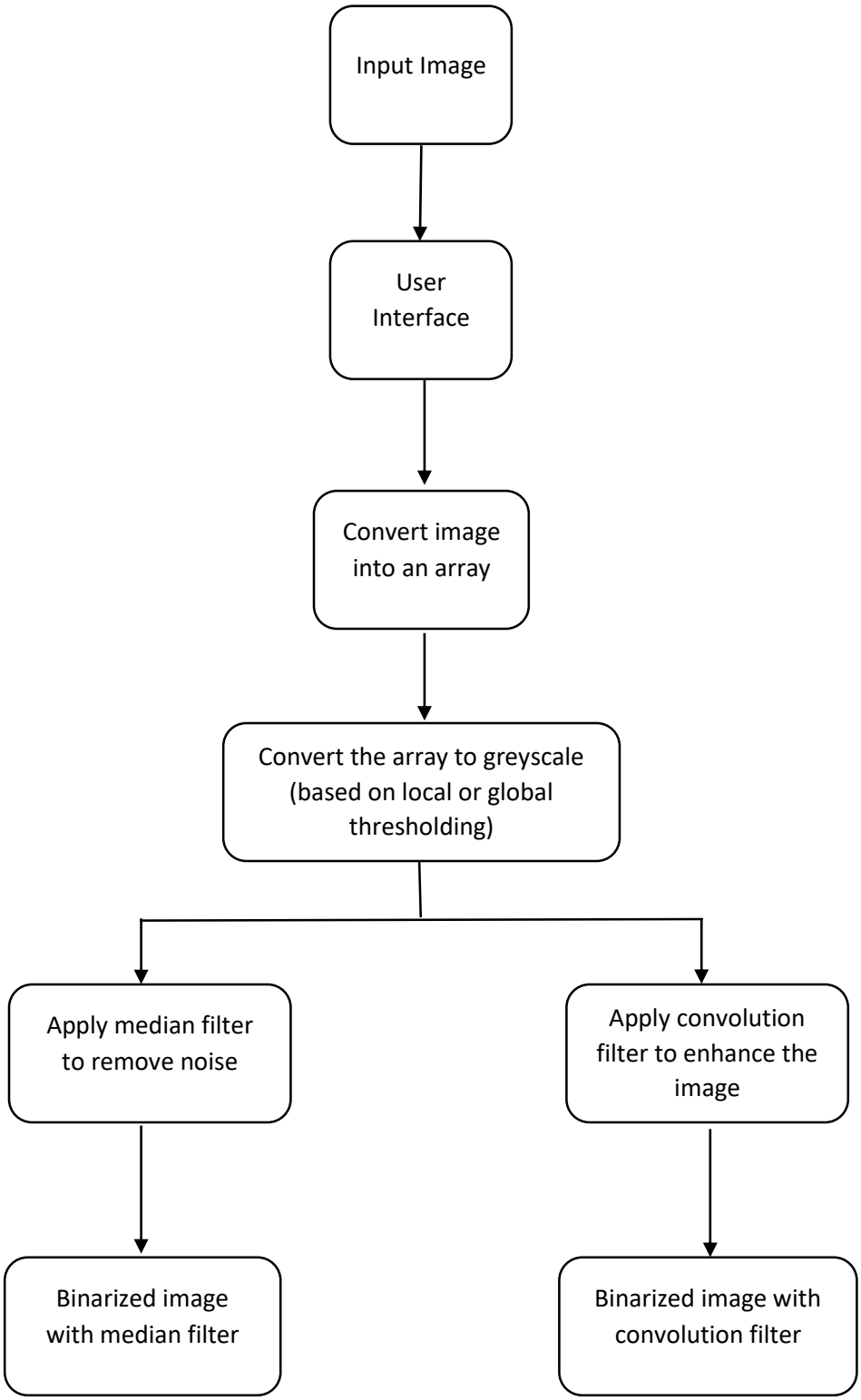
0	1	0
1	-4	1
0	1	0

Edge Detect

-2	-1	0
-1	1	1
0	1	2

Embross

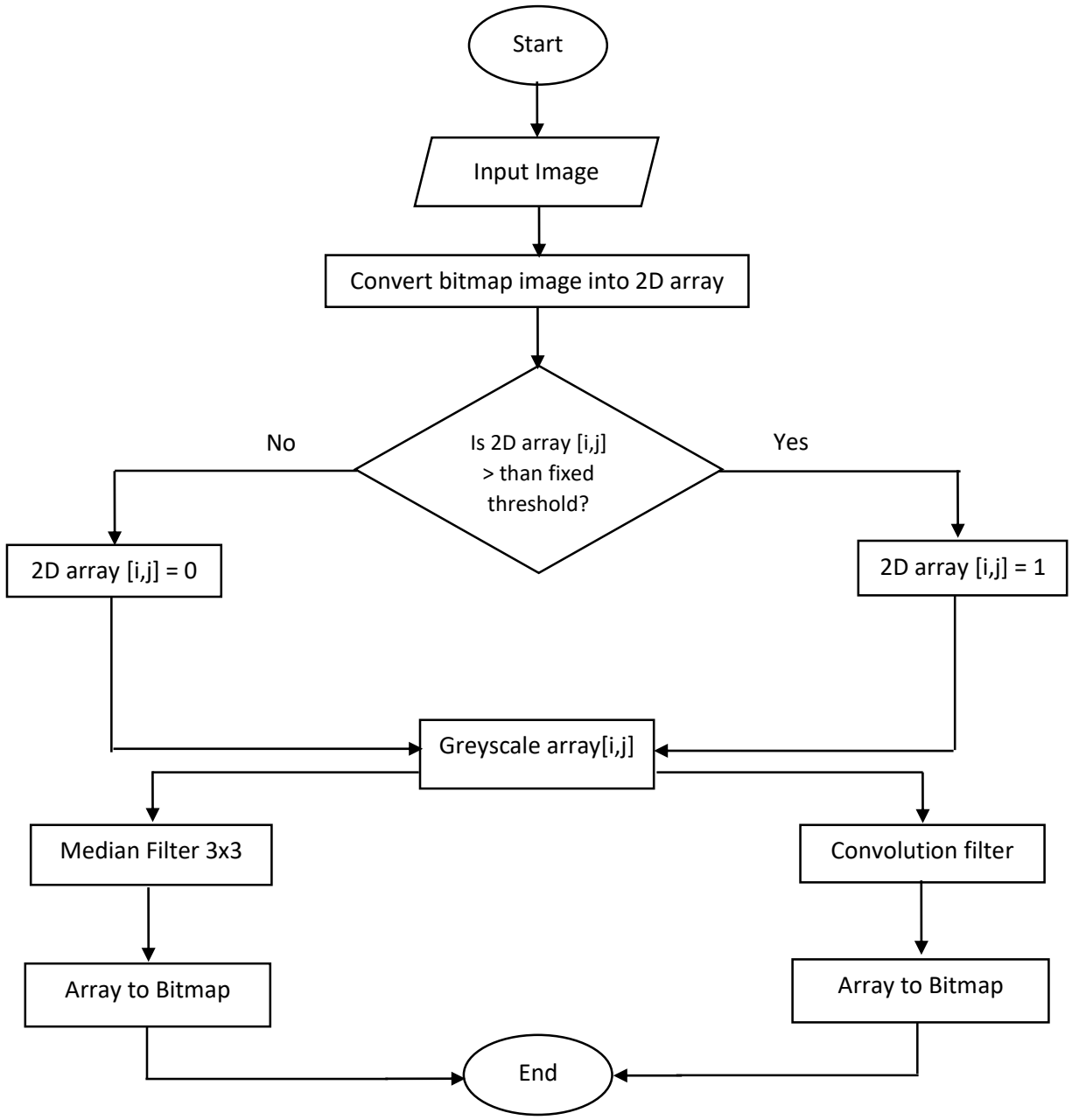
4. Binarization Filter Block Diagram



5. Global Threshold Binarization Filter (Fixed Threshold):

Steps involved:

1. The input image in Bitmap is converted into a 2D array which contains combined value of R, G and B values.
2. Considering the predefined threshold, the resulting array is converted into a greyscale array where the pixel values greater than the threshold are set to white and the pixel values lesser than threshold are set to black.
3. For further enhancement and removal of noise either median filter or convolution filter can be applied.
4. The result of the filter applied will then be converted to bitmap.
5. The obtained bitmap result is then saved in the desired format.



6. Local Threshold Binarization Filter (Dynamic Threshold):

Steps involved:

1. The input image in Bitmap is converted into a multidimensional array which contains -alpha, -R, -G, -B, -ARGB and –last component which is a 32 bit integer containing alpha, R, G, B in its consecutive 8 bit positions.

2. Dynamic threshold is being calculated by Otsu thresholding technique where histogram of the image is obtained.

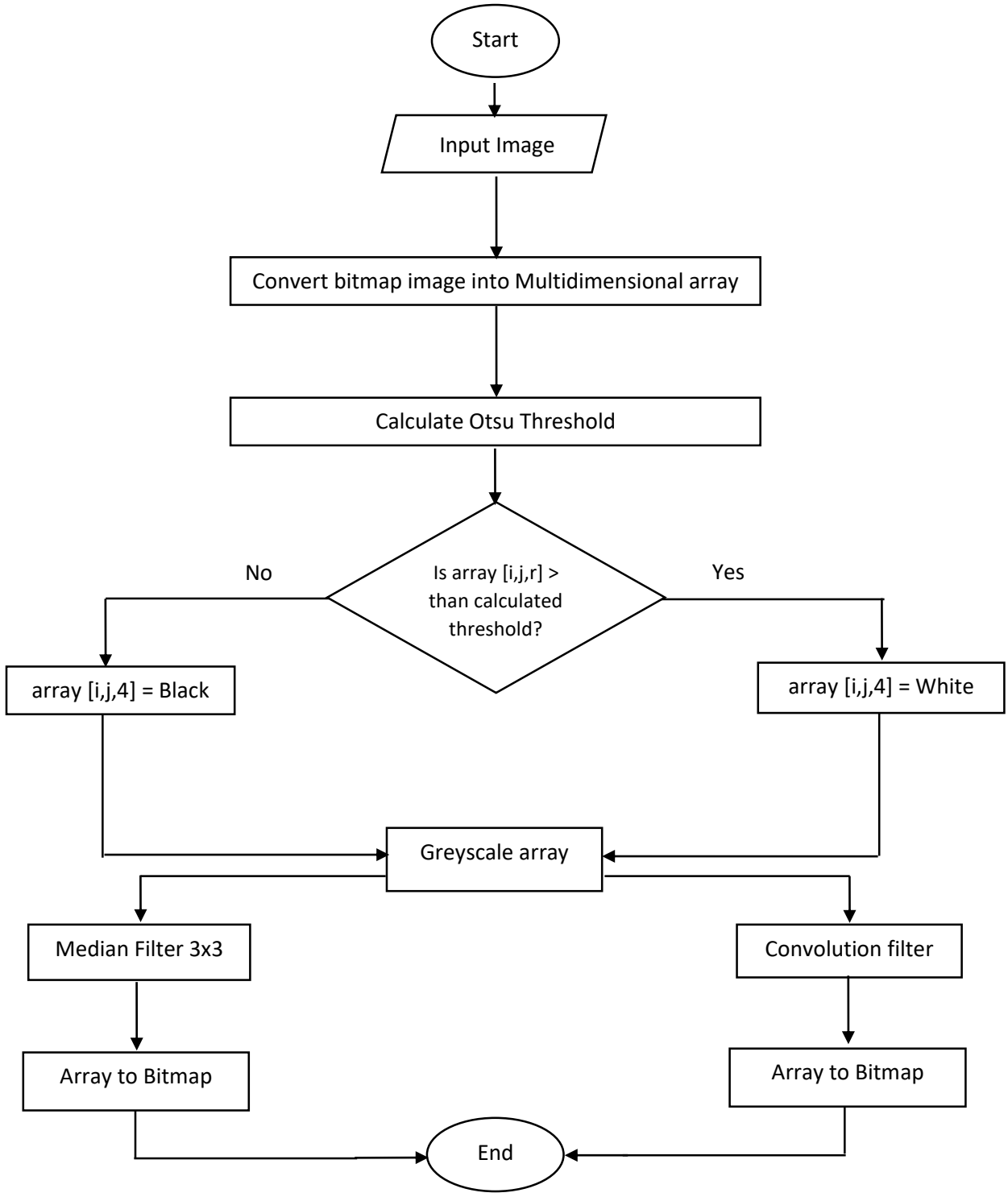
3. Considering the calculated threshold, the resulting array is converted into a greyscale array where the R value is checked and if it is greater than the threshold then RGB is set to white(255,255,255) and if the R value is lesser than threshold then RGB is set to black(0,0,0). Only the last component is recalculated using the new R, G, B values.

4. For further enhancement and removal of noise either median filter or convolution filter can be applied.

5. The result of the filter applied will then be converted to bitmap.

6. The obtained bitmap result is then saved in the desired format.

Local Thresholding Flow Chart



7. Outputs

Binarization Filter using fixed threshold:

Original Image

Binarized fixed threshold image (th=127)

Median filter Output

Convolution Filter Output

Binarization Filter using Dynamic threshold:

Original Image

Binarized Dynamic threshold image

Median filter Output

Convolution Filter Output

8. Conclusion

In this project an approach for image binarization has been presented. Given a piece of image of a fixed size, the model outputs a selectional value for each pixel of the image depending on the confidence whether the pixel belongs to the foreground of the document. These values are eventually thresholded to yield a discrete binary result.

Global thresholding method is better approach for calculate the threshold values of a grey scale images. But it doesn't give good results for colored image and under intensity illumination. Therefore Otsu Threshold is an effective method to calculate the threshold of an image dynamically as Otsu is used to, automatically perform clustering-based image thresholding.

In a world saturated with technology and information, digital signals are literally everywhere. Much of the technology that exists today would not be possible without a means of extracting information and manipulating digital signals. One such method for processing digital images, called filtering, can be used to reduce unwanted signal information (noise) or extract information such as edges in the image. The converted original image to grayscale contains a certain amount of noise in it. Depending upon the requirement we can use median filter, convolution filter or other filtering techniques.

9. References

1. https://softwarebydefault.com/2013/05/18/image-median-filter/
2. https://www.codeproject.com/Articles/16859/AForge-NET-open-source-framework
3. https://www.markschulze.net/java/meanmed.html
4. https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Image%20Filtering_2up.pdf
5. http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html
6. https://www.researchgate.net/publication/318107356_A_selectional_auto-encoder_approach_for_document_image_binarization
7. Binarization Techniques used for Grey Scale Images-International Journal of Computer Applications (0975 – 8887) Volume 71– No.1, June 2013

Link to Repository:
https://daenet.visualstudio.com/SE-2017-2018/SE-2017-2018%20Team/_git/SE-2017-2018?path=%2FSE%2FMyProject%2FBinarization%20Filter&version=GBSurajBharadwaj&_a=contents