CSE585/EE555: Digital Image Processing I

Computer Project # 1:

Mathematical Morphology: Hit-or-Miss Transform

Aishwarya Mallampati, Mrunmayi Ekbote, Wushuang Bai

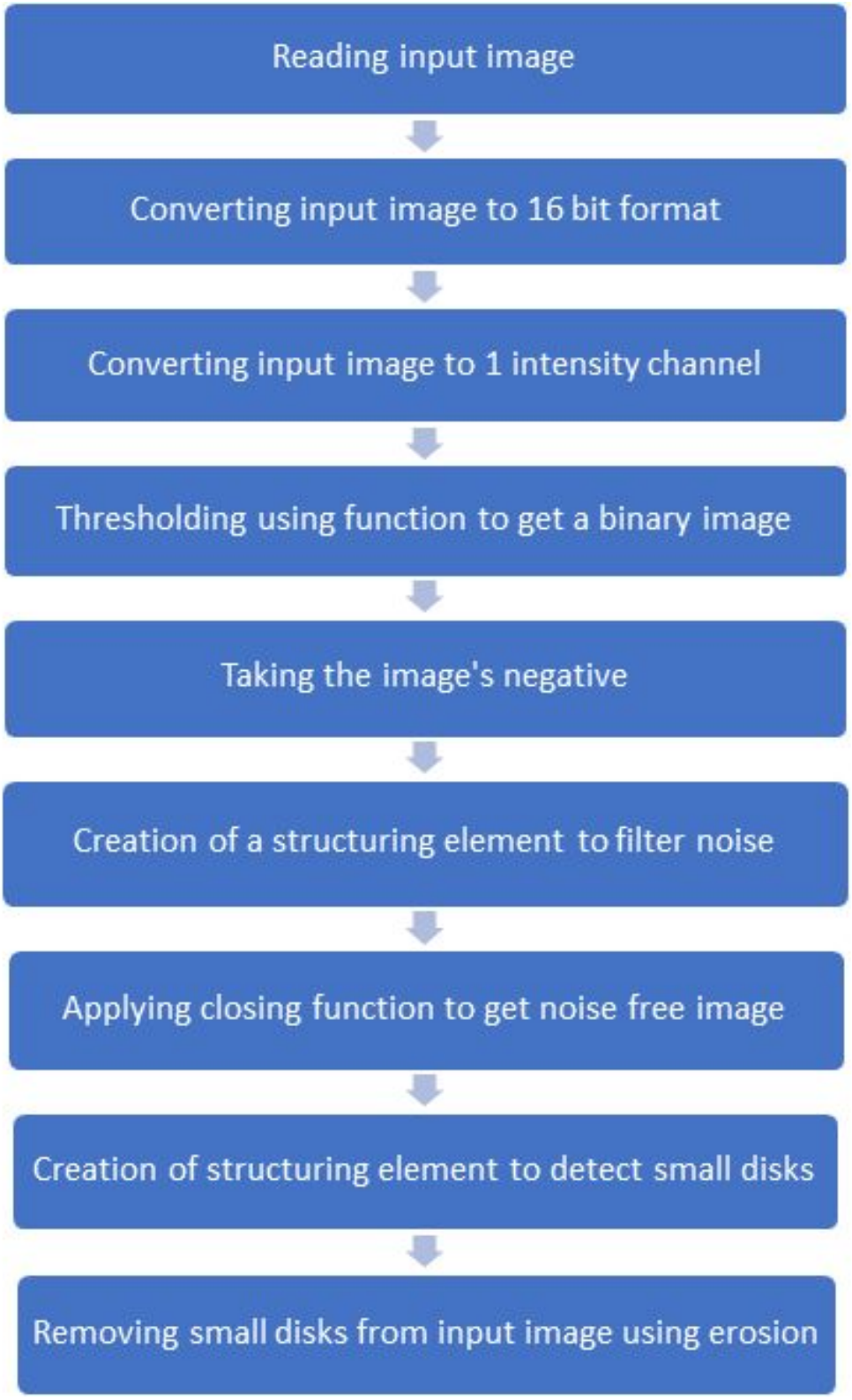
Date: 1/31/2020

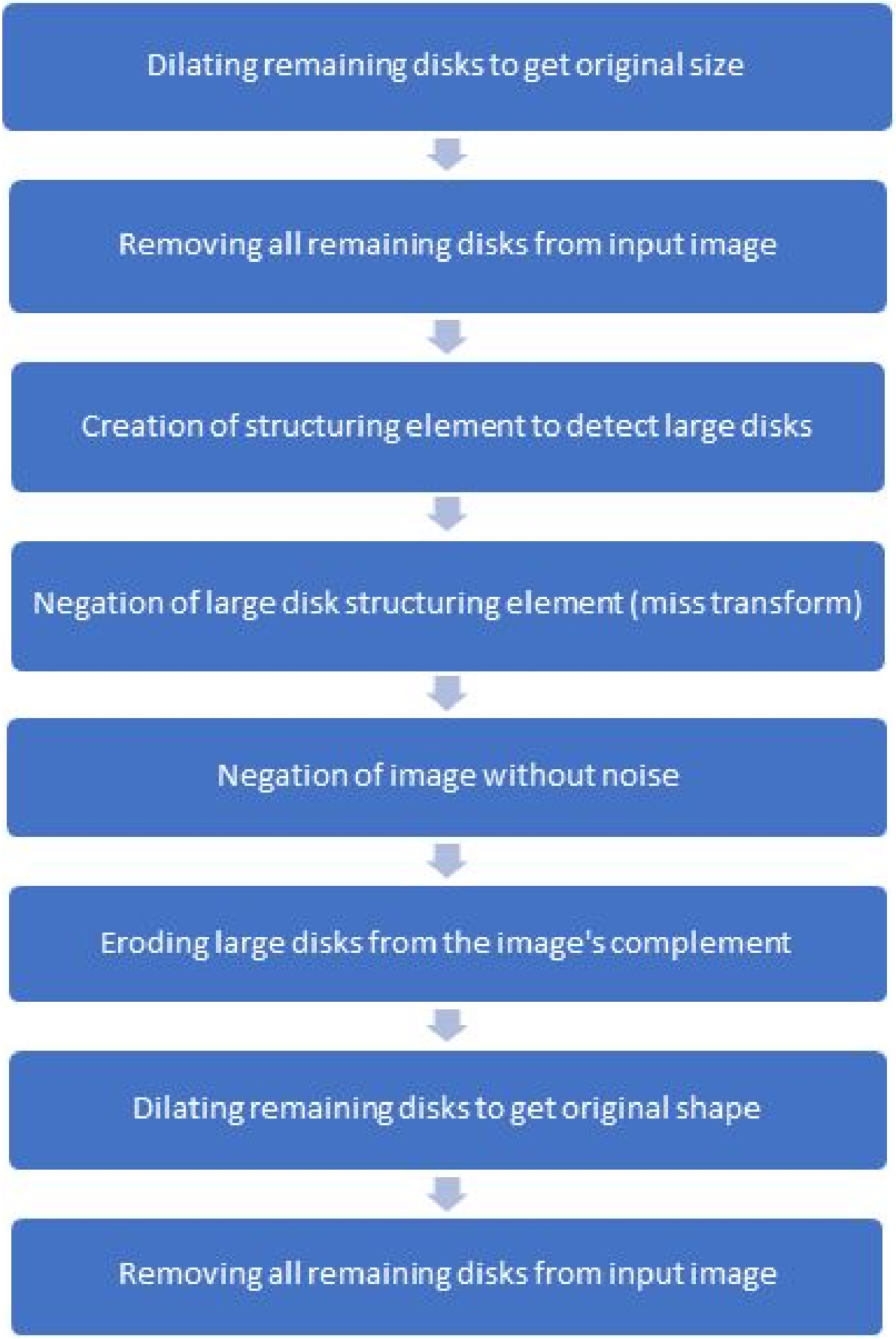
A. Objectives

- The objective of this project is to apply the hit-or-miss transform on a digital image and observe the results.
- In this project, we have studied and implemented various mathematical morphological operations such as erosion, dilation, opening, closing, etc. using MATLAB software.
- We have filtered the given image using opening and closing to remove the salt-and-pepper noise present in it.
- We have designed a Hit-or-miss transform that detects the smallest and largest disks in the given image.

B. Methods

The flowchart below describes the overall workflow of the project:




Dilating remaining disks to get original size
Removing all remaining disks from input image
Creation of structuring element to detect large disks
Negation of large disk structuring element (miss transform)
Negation of image without noise
Eroding large disks from the image's complement
Dilating remaining disks to get original shape
Removing all remaining disks from input image

The code of this project was divided into one main function and several other functions which were used for particular tasks.

Function **main.m**

Firstly, we read the input image using the inbuilt imread function. Since this image was in the 8 bit format, we converted it to a higher precision format (16 bit) using im2uint16 function. This image had 3 intensity channels (R,G,B), so it was converted to one intensity channel (BW).

Then the function 'pixelwisethresholding' was called to threshold the obtained image in order to convert it into a binary image.

Function **pixelwisethresholding.m**

This function has the parameters (image, no of rows, no of columns).

Firstly, a matrix was created that consists of all 0s, having M rows and N columns, using the function 'zero'. After this, two for loops were used to cycle through an MXN array and if the value of the pixel was less than half the value of the pixel with maximum value, it was set to 0 (black); else it was set to 1(white).

Function **zero.m**

This function has the parameters (M(no of rows) N(no of columns) x(x coordinate of a pixel) y(y coordinate of a pixel)). It was used to create an image full of zeroes. It used two for loops to cycle through an MXN array and assign 0 to all values.

Negating input image to match standard format

Ater thresholding, the given image was complemented it considers black as the foreground and white as the background, which is opposite to general conventions.

Creation of structural element to remove salt and pepper noise

Then a structuring element was created for the removal of salt and pepper noise present in the image using opening/closing. The element used was a straight line of length 3 pixels and thickness 1 pixel. This element was used to perform the closing operation by calling the closing function.

Function **closing.m**

This function was used to close the image for removing the salt and pepper noise. It had two parameters (image, structuring element). Closing operation is performed by first dilating the given image and then eroding the dilated image. For this the dilation_erosion function was used.

Function **dilation_erosion.m**

This function has three parameters(image, structuring element, flag). In this function, we first two matrices having the same size as our image and structuring element respectively. Then we created an output image of the same size as the input image and set it to all 0s using the function zero.m. The structuring element was moved along the image pixel wise using two for loops and the output was changed accordingly. The flag parameter was used to decide the operation to be performed- flag= 1 indicates dilation and flag= 2 indicates erosion.

After performing closing operation, a noise free image was obtained.

Hit Transform: Detection of smallest disks:

To detect the smallest disks present in the input image, a structuring element approximately of the same size and shape of the small disks is required. To create structuring element, the input image was analyzed using imshow and the pixel values of the boundaries of the smallest disk were noted. Using the boundary pixel values, a small disk sub-matrix was extracted from the input image. After this, the small disks were removed from the input image by erosion using the dilation_erosion function. After this, the remaining disks were dilated to restore their original shape and size using the dilation_erosion function. Then, all the remaining disks were removed from the input image to get an output image containing only the smallest disks.

Miss Transform: Detection of largest disks:

Similarly, to detect the largest disks, the boundaries of one of the largest disks in the input image were noted using imshow(). Using these boundary pixel values, a large disk submatrix was extracted from the input image. A miss transform needs to be applied which misses all the other disks in the input image. So, the structuring element should be the background image of the largest disk. The large disk matrix is negated to generate the required structuring element. For miss transform, the complement of the input image was used. After this, the largest disks were eroded from the complement of the input image. The remaining disks were dilated to retrieve their previous shape and size, using the

dilation_erosion function. Finally, all the remaining disks were removed from the input image to get an output image containing only largest disks.

As per our original image, disks should be in black color, so the output image was negated.

The final output was obtained by combining the two output images with only the smallest disks and only the largest disks.

Function **main_without_noiseremoval.m**

This function was used to find the output image by applying hit-or-miss transform to the input image without filtering the salt and pepper noise. Similar steps as the main function are followed here, only the closing operation is skipped.

C. Results

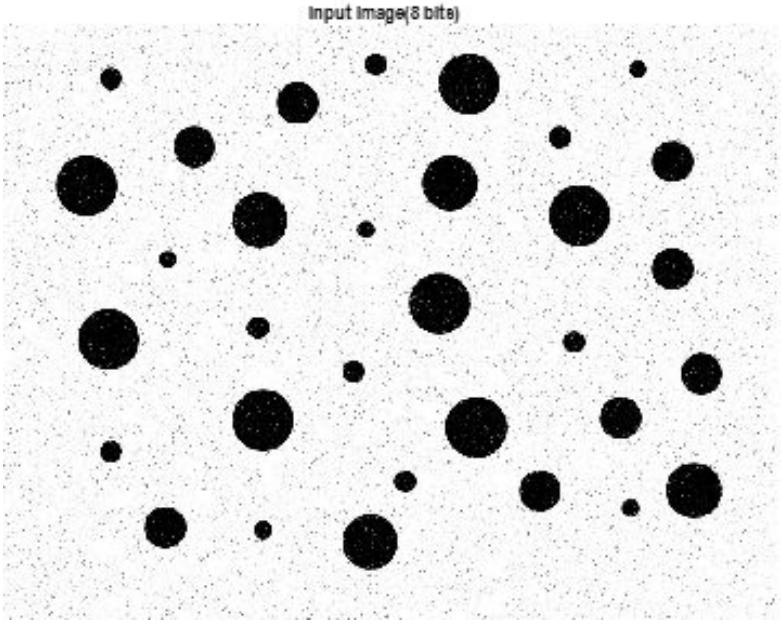


Fig 1 Input Image (8 bits) 'Random Disks'

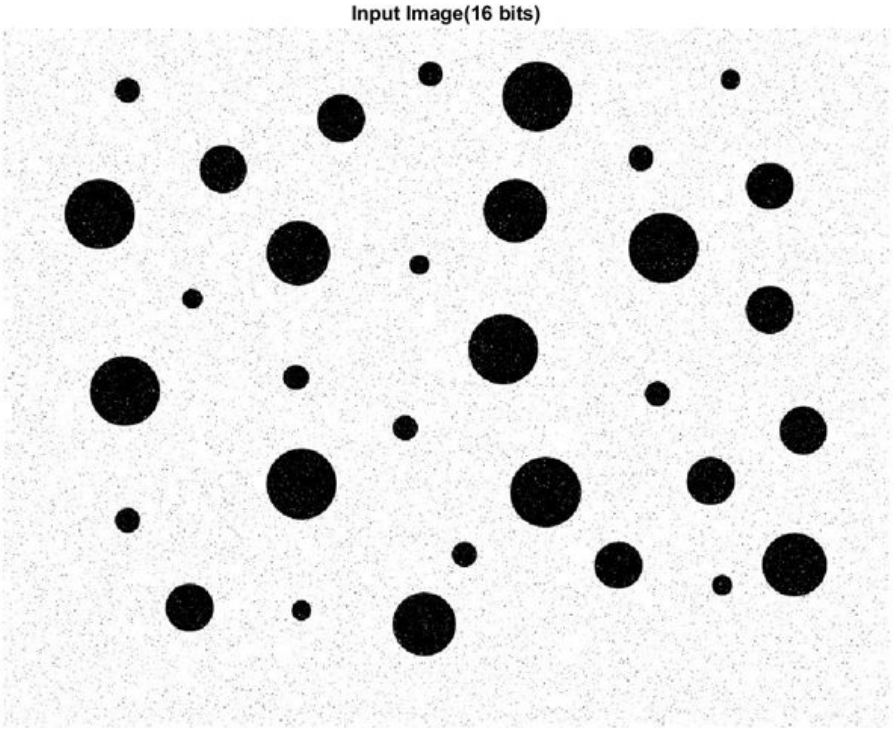


Fig 2 Input image converted to 16 bits

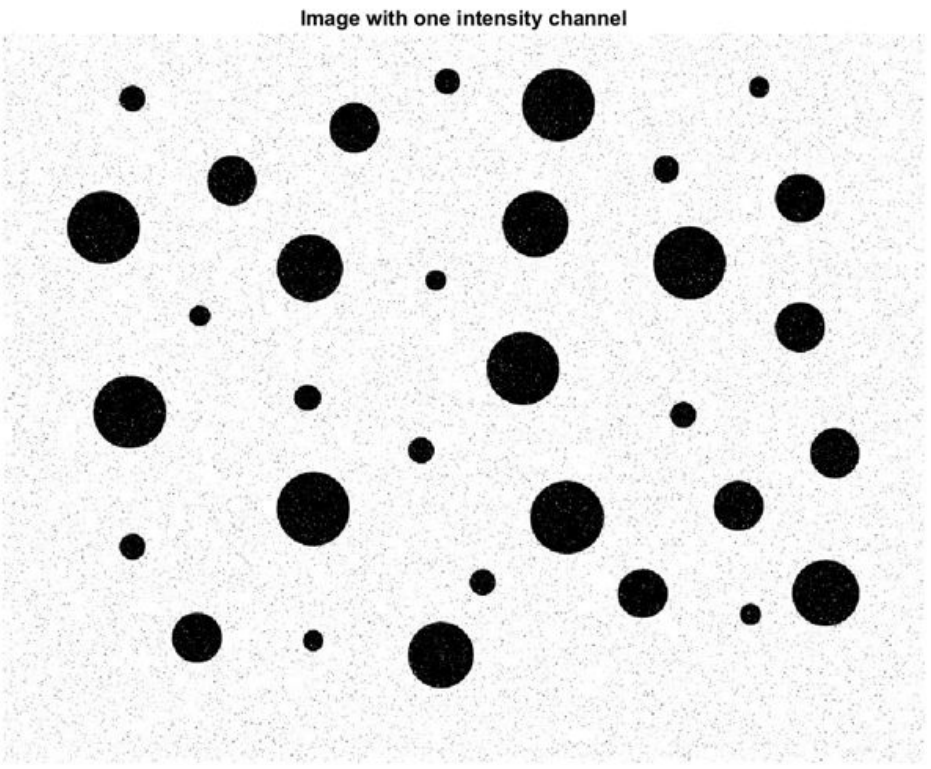


Fig 3 Input image with 3 intensity channels converted to 1 intensity channel

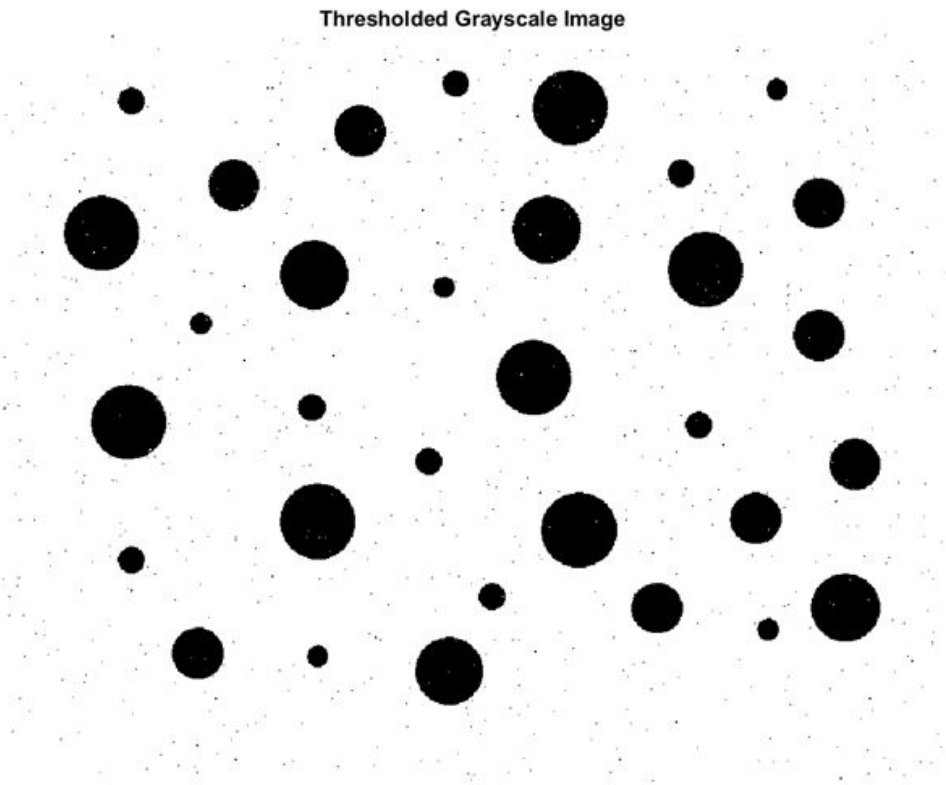


Fig 4 Thresholded image

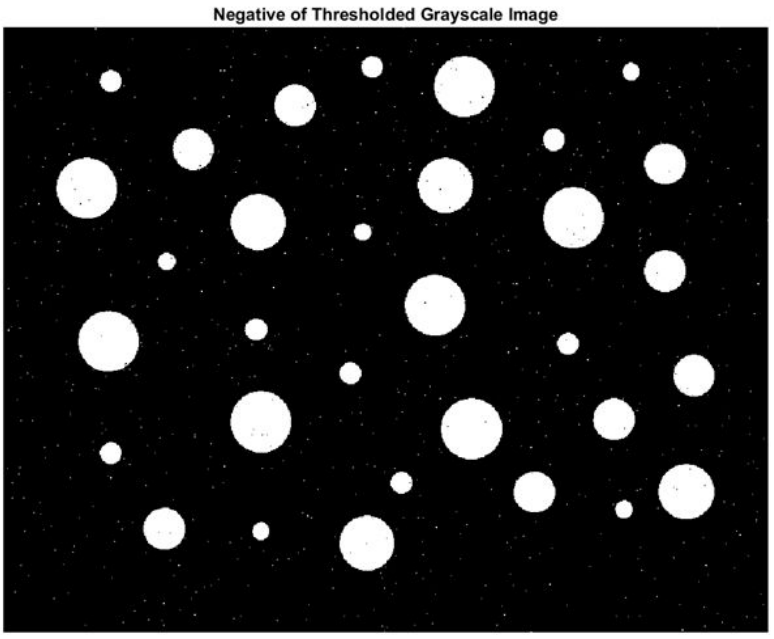


Fig 5 Negative of thresholded image

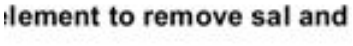


Fig 6 Structuring element to remove salt and pepper noise (straight line of length 3 pixels and thickness 1 pixel)

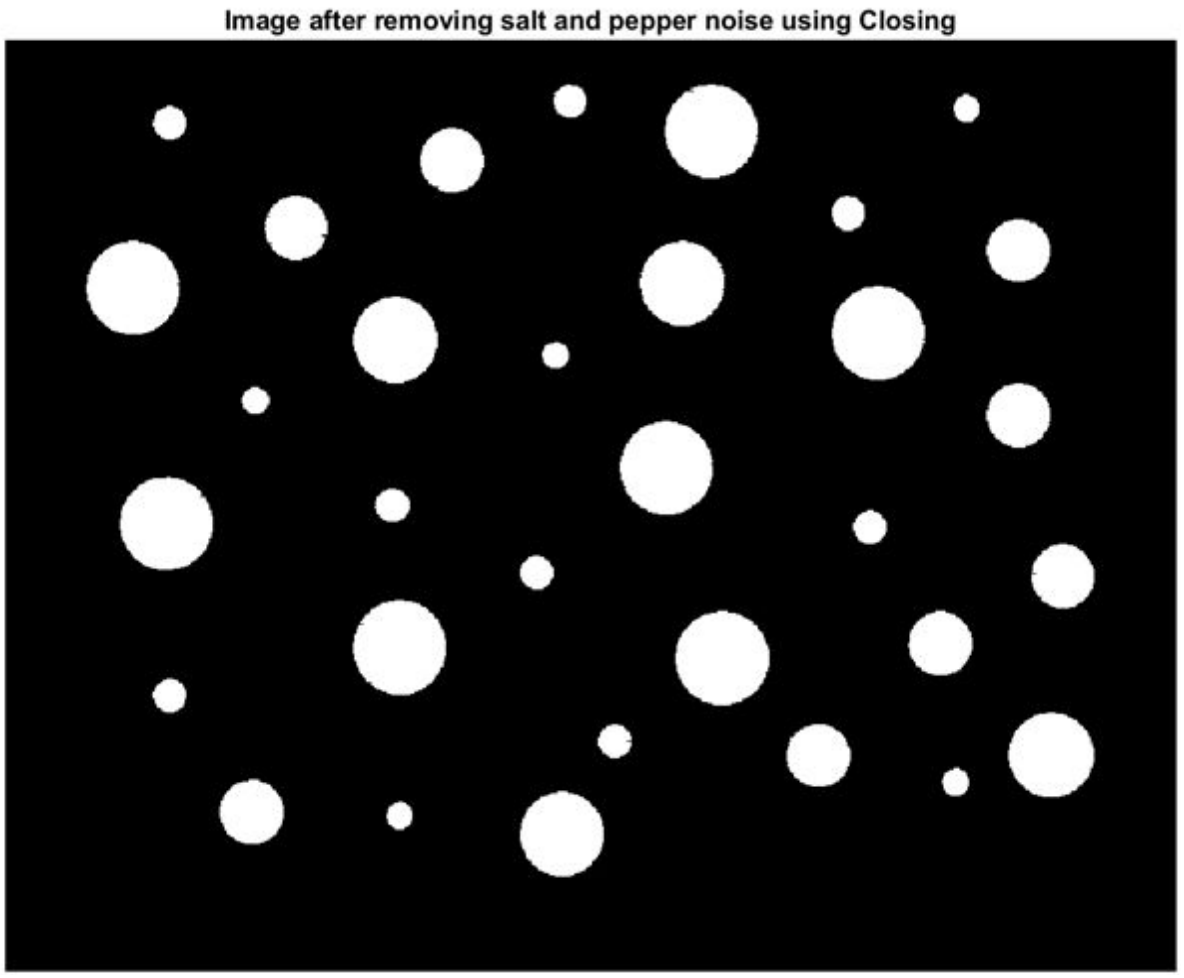


Fig 7 Image after removing salt and pepper noise using closing operation

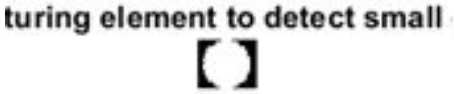


Fig 8 Structuring element to detect the smallest disks

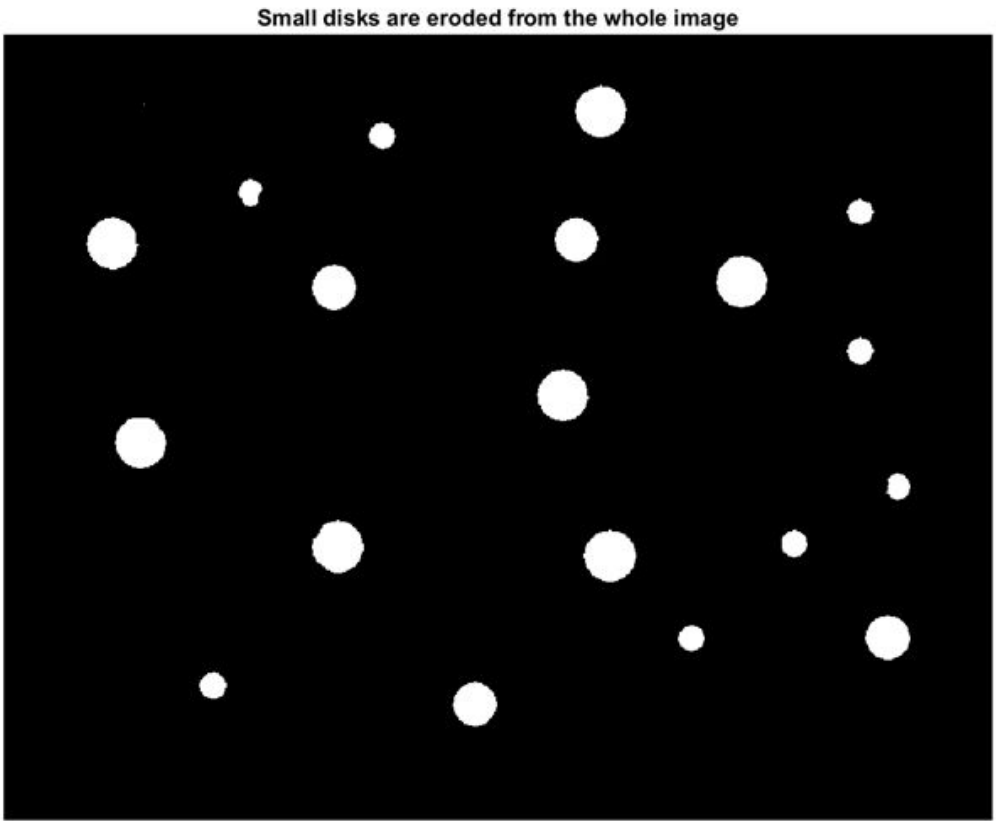


Fig 9 Image after erosion of smallest disks

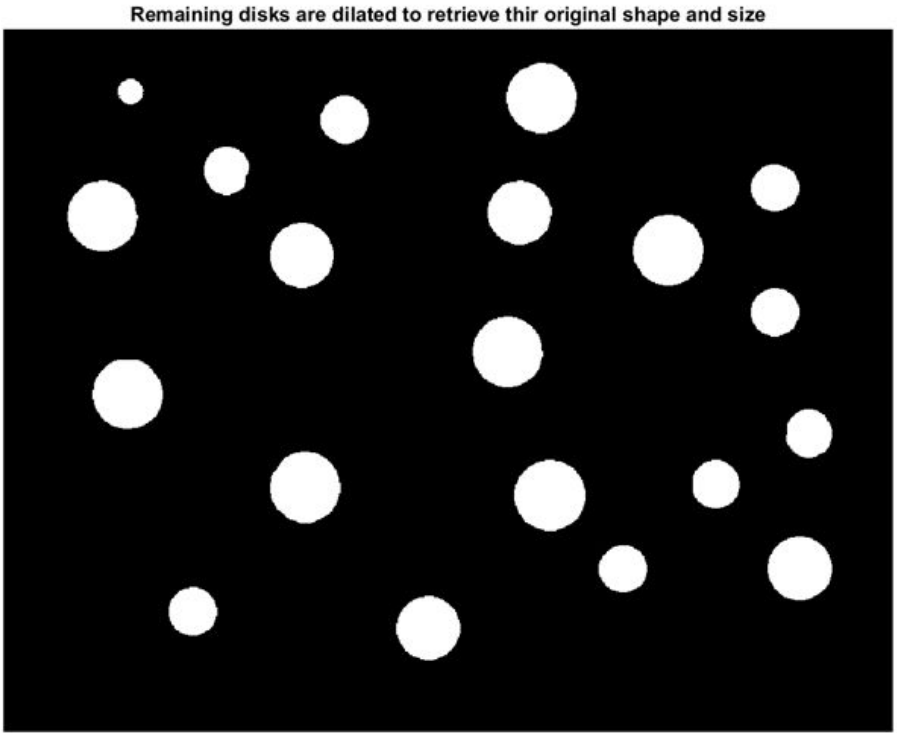


Fig 10 Dilation of remaining disks to retrieve original shape and size

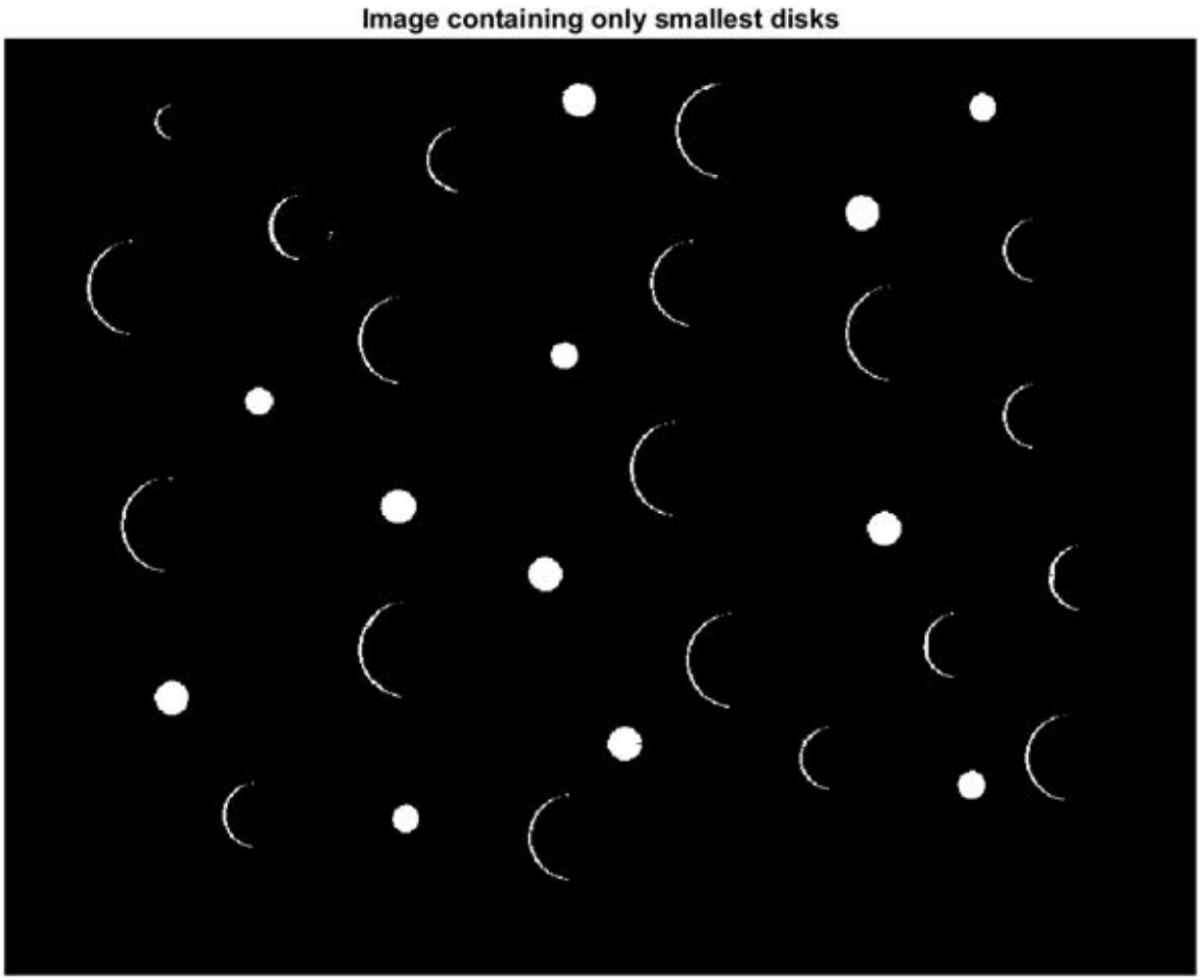


Fig 11 Image containing only the smallest disks(Hit Transform result)

Fig 12 Largest disk

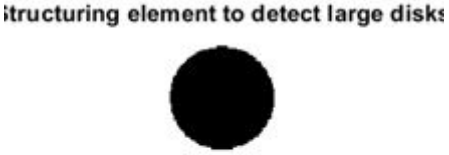


Fig 13 Structuring element to detect the largest disks

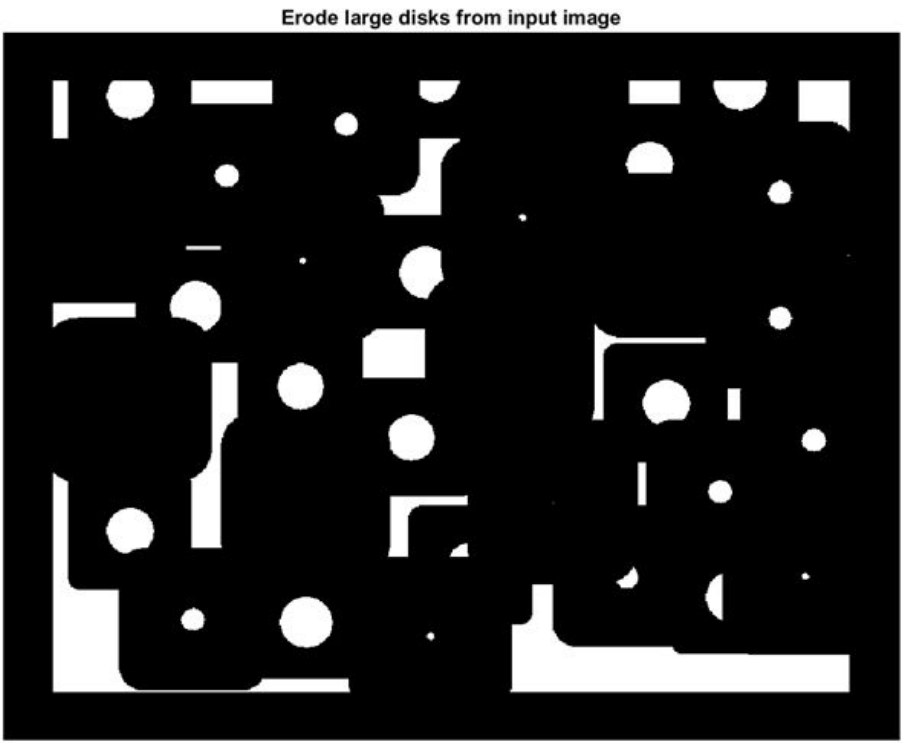


Fig 14 Erosion of largest disks from the input image

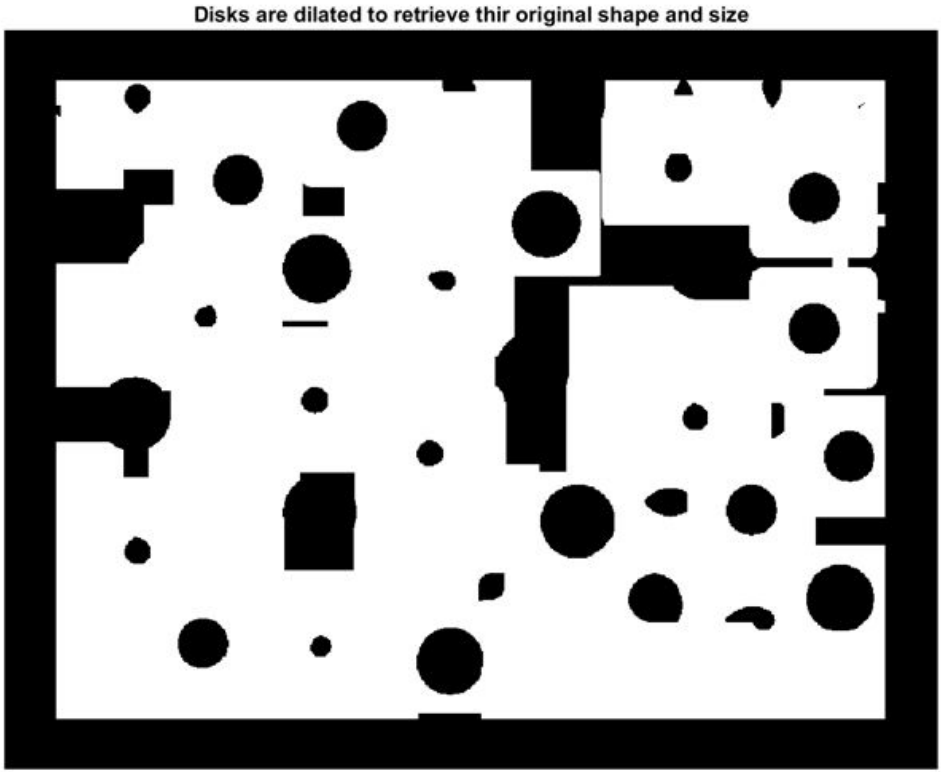


Fig 15 Dilation of disks to retrieve their original shape and size

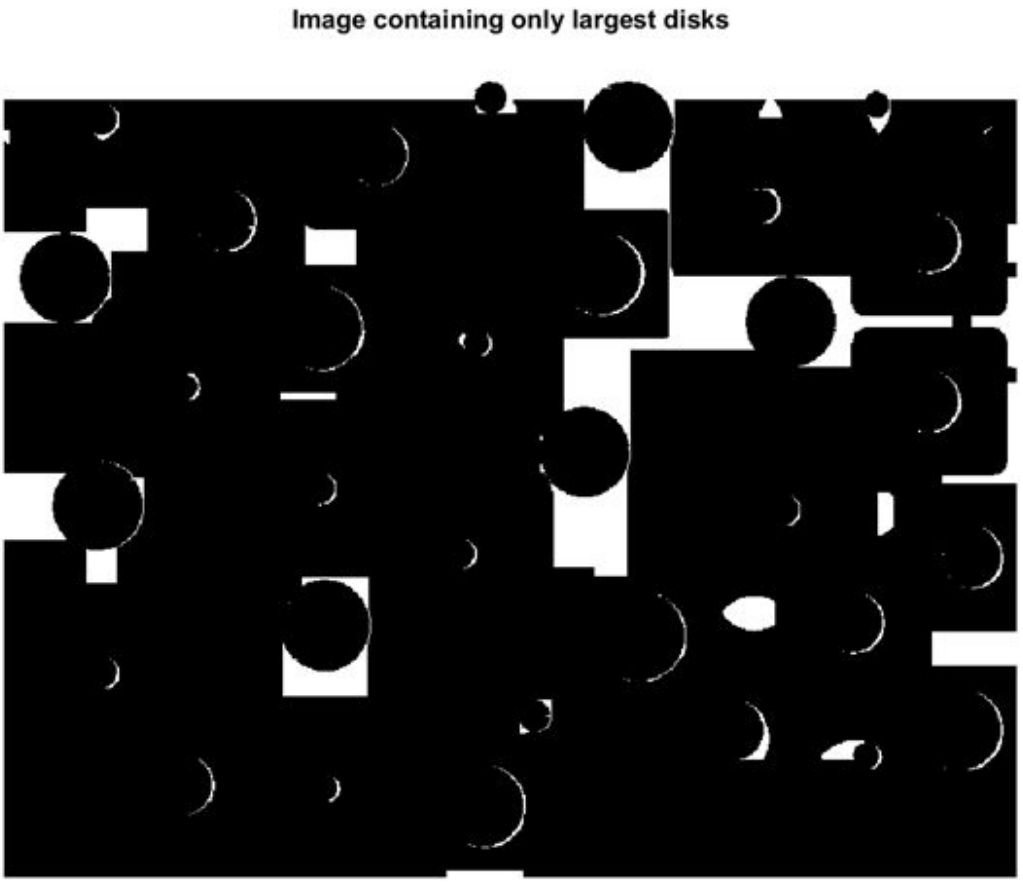


Fig 16 Image containing only the largest disks(Miss Transform Result)

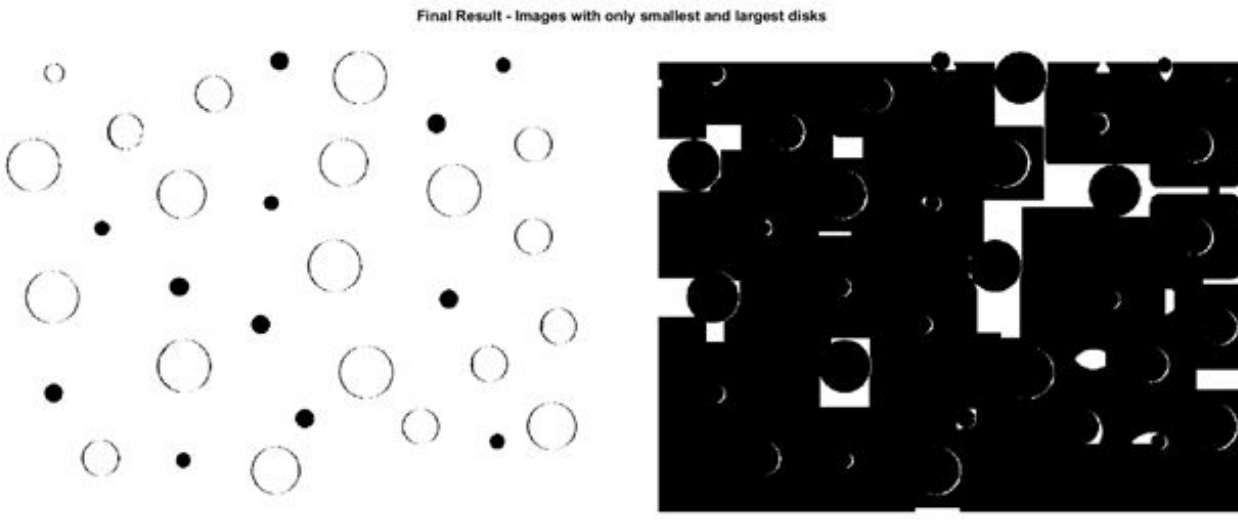


Fig 17 Final Result- images containing only the smallest and largest disks

(a)Discuss exactly how you reduced the salt-and-pepper noise.

In order to remove the salt and pepper noise, we used the closing operation. This operation consisted of dilation followed by erosion using the same structuring element.

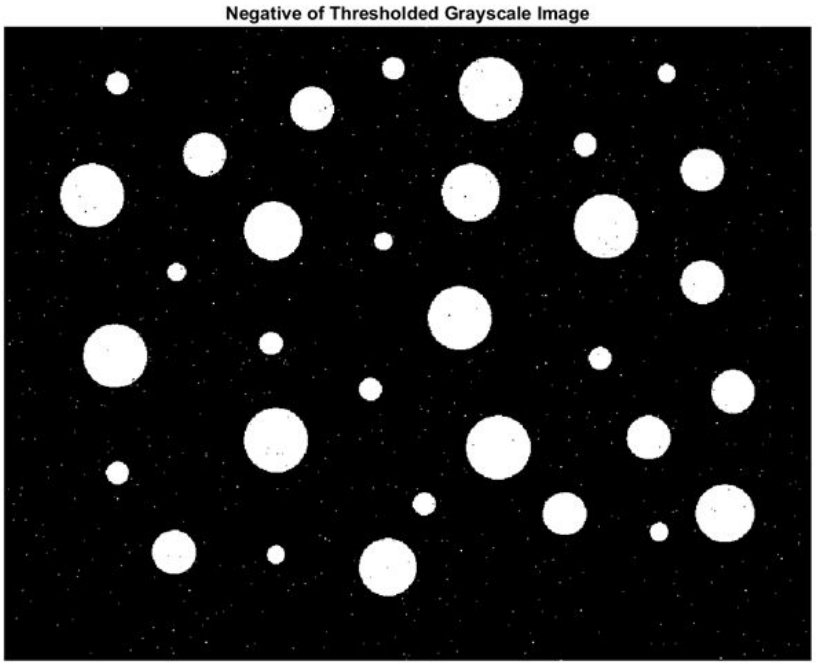


Fig: Original thresholded grayscale image with salt-and-pepper noise

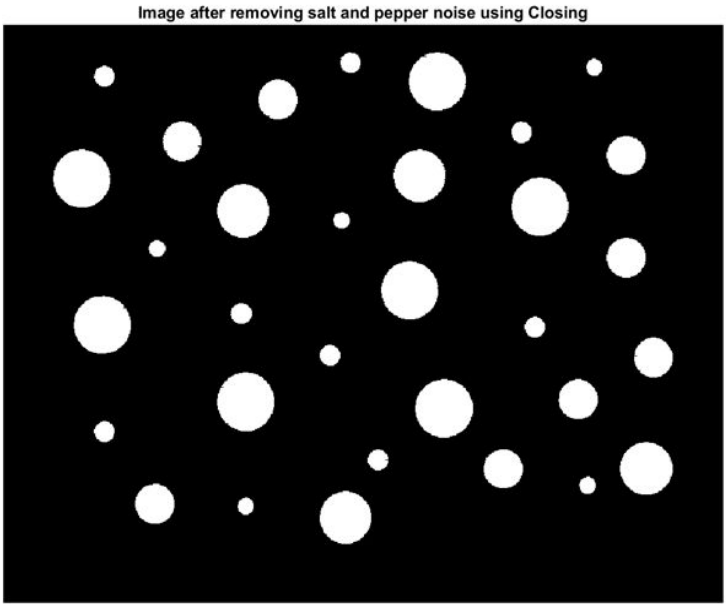


Fig: Image after noise removal by applying closing

(b) Discuss how you selected structuring elements, A and B, for the hit-or-miss transform.

To detect small disks present in the input image, a structuring element approximately of the same size and shape of the small disks is required. To create such a structuring element, the input image was analyzed using imshow and the pixel values of the boundaries of the small disk were noted. Using the boundary pixel values, the small disk sub-matrix was extracted from the input image.

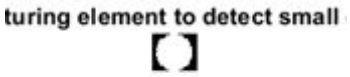


Fig: Structuring element to detect small disks

Similarly, to detect large disks, the boundaries of one large disk in the input image were noted using imshow(). Using these boundary pixel values, a large disk submatrix was extracted from the input image. In order to detect large disks, a miss transform needs to be applied which misses all the other disks in the input image. So, the structuring element should be the background image of the large disk. The large disk matrix was negated to generate the required structuring element.

Fig: Large disk

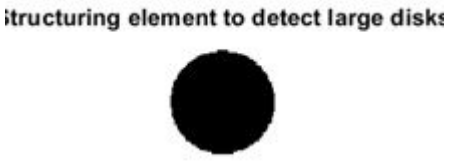


Fig: Structuring element to detect large disks

(c) Suppose you do not apply the small close/open as suggested. Will your hit-or-miss transform work? Demonstrate this and discuss your results.

If the opening and closing is not applied, the hit-or-miss transform will not work properly. The noise will not allow the algorithms to detect the required shape properly

The result of this operation can be observed in the figure below.

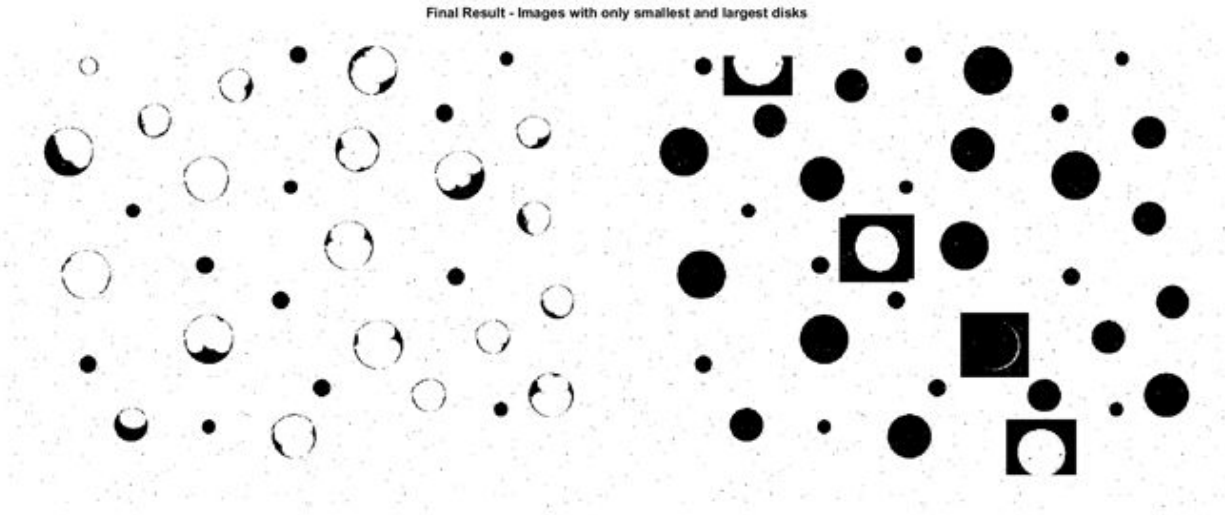


Fig Final result of hit or miss transform without noise removal

D. **Conclusions**

From the observed results, it can be concluded that to apply hit or miss transform to a digital image, it is necessary to select proper structuring elements to get the correct output. Also the presence of any noise in the input image will lead to an incorrect output, hence the noise must be filtered out using appropriate means.

How to run source code:

- Just run the “main.m” file for the actual results. All the images generated during the program execution are stored in “output_images” folder.
- To view output without removing salt and pepper noise, run “main_without_noiseremoval.m” All the images generated during the program execution are stored in “output_images_without_noiseremoval” folder.