Result

1. Input image loading and binary image conversion

We first loaded the 8-bit original image and extracted the grayscale image out (Figure 1). For further processing, the image was converted to binary image with 0.5 as the threshold level (Figure 2). To be specific, pixels with value bigger than 255 * 0.5 will be changed to 0, while those whose values are smaller than 255 * 0.5 will be set to 1. Note that in the binary image, we inverted the color of the original images, so that the color of disks is white (represented by 1) and the background is black (represented by 0), because disks are our interest areas, which had better be set to 1.

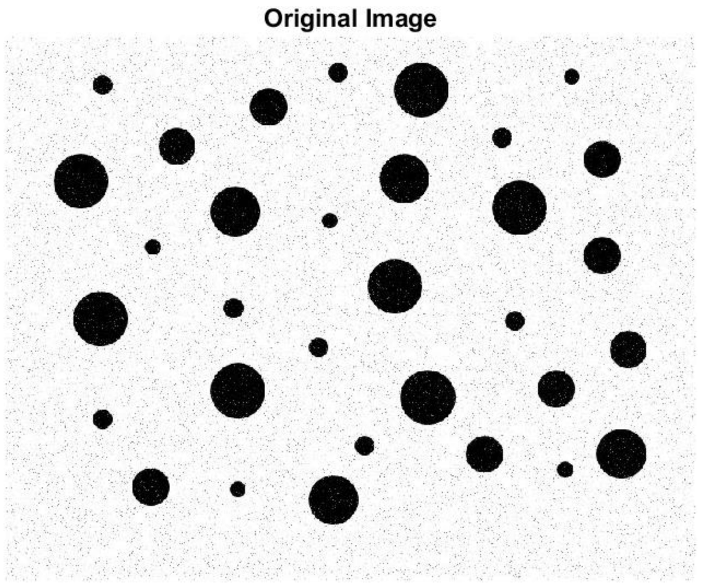


Figure 1 Grayscale image of the original image

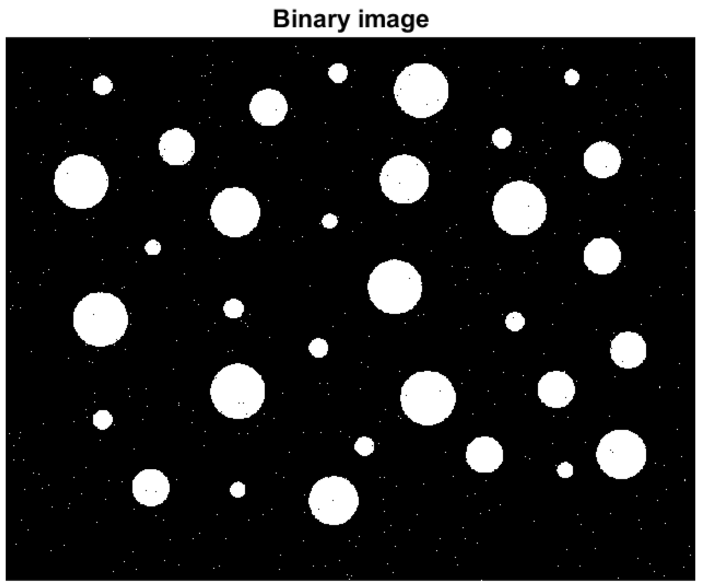


*Figure 2 Binary image converted from the original image with threshold 255*0.5*

2. Remove the salt-and-pepper noise

As shown by Figure 1 and Figure 2, there were salt-and-pepper noise at the background and inside the disks. The noise will severely affect the identification of disks with hit-or-miss transform. Therefore, we filtered such noise out with small open and close filters. We first applied the {(-1, -1), (-1,0)} open filter to the binary image (*Figure 3*). Opening is involved erosion followed by dilation, which is capable of removing object smaller than the structure element. *Figure 3* demonstrated the small white noise at the background were successfully gone after open filter. In addition to the white noise, there are also small cavities inside disks to be filled. We achieved this by further applying close filter {(-1, -1), (-1, 0), (0, 0), (0, -1)} to the image after opening. Any holes smaller than the close filter will be filled. From *Figure 4*, we can see, all the cavities were filled and all disks became solid.

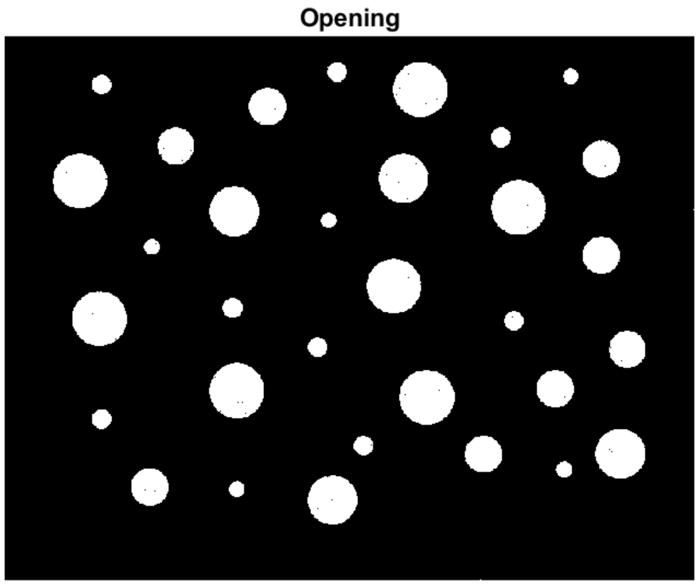


Figure 3 Binary image after {(-1, -1), (-1,0)} open filter

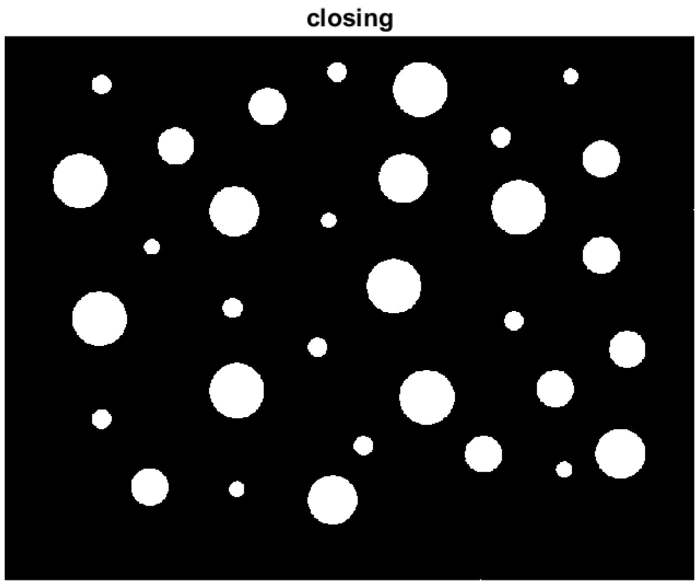


Figure 4 Binary image after further closing with structural element {(-1, -1), (-1, 0), (0, 0), (0, -1)}

3. Identify radius of the smallest and the biggest disks

Having preprocessed the image, we next identified the radius of all the disks. Especially, we need to figure out radius of the smallest and biggest disks. As described in the method and shown in Figure 5, imfindcircles function enabled us to find both centers and radii of all circles within given radius range. As shown in Table 1, the radii of all disks fell into five groups ($r1 \approx 9.0$, $r2 \approx 11.5$, $r3 \approx 22$, $r4 \approx 29$, $r5 \approx 32$). Therefore, the smallest radius is around 9, and the biggest radius is around 32.

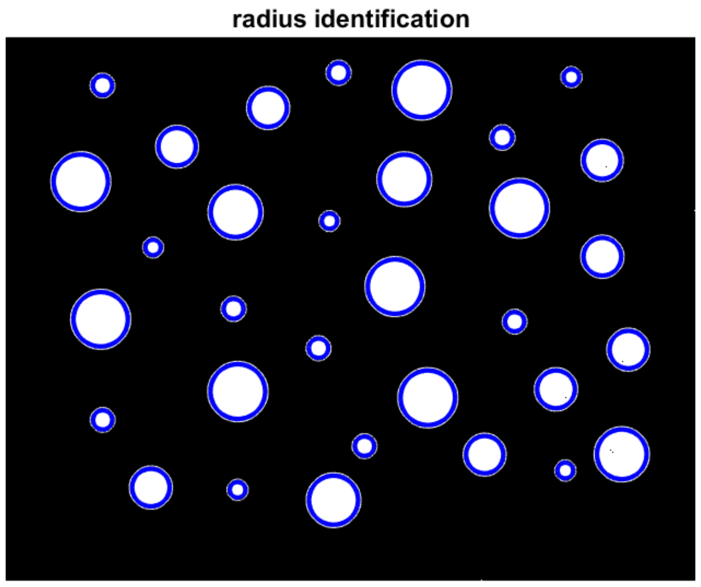


Figure 5 Circle capture with imfindcircles

r1	r2	r3	r4	r5
9.021061	11.26251	21.48988	28.70378	31.68008
9.043704	11.26987	21.81422	28.83285	31.81814
9.155122	11.28027	21.81422	29.00218	31.81995
9.18997	11.29771	21.90512	29.05537	31.82826
9.229171	11.32621	21.96168		31.83494
	11.53776	21.96443		31.9037
	11.54774	22.04442		32.02316
	11.66112	22.05914		

Table 1 All radius identified with imfindcircles

4. Detect the smallest disks

With the information of the smallest and the largest radii, we constructed two structural elements Figure 6. A-smallest is a disk with radius 8, while B is a square having a hole, the radius of which is 10. To detect the smallest disks, we first computed erosion of the preprocessed image with respect to A-smallest. Disks smaller than A were removed at this step. Then with the complimentary image of the preprocessed image, we computed erosion with respect to B. This step will eliminate all disks having larger radii than 10. Lastly, the two resulted images were intersected with each other. In summary, the three steps detect disks with radius ranging from 8 to 10. Figure 7 below demonstrated that we successfully detected 5 disks. By checking corresponding positions in the original image (Figure 8), we further confirmed that the 5 identified positions were indeed the where the 5 smallest disks located.

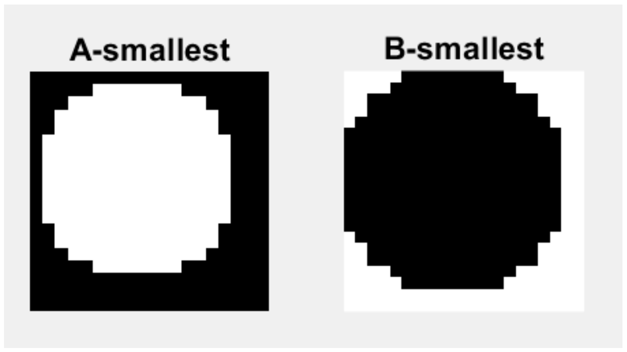


Figure 6 Structural element for finding the smallest circles

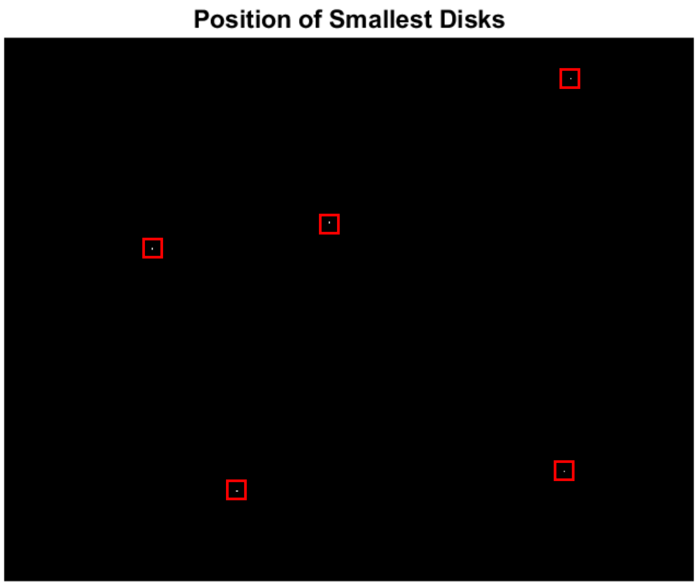


Figure 7 Positions of smallest disks detected (highlighted with red box)

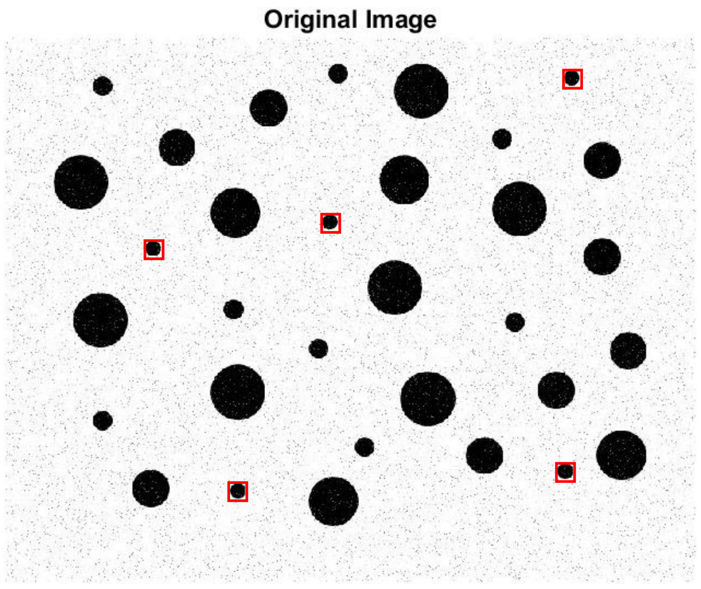


Figure 8 Corresponding smallest disks in the original image

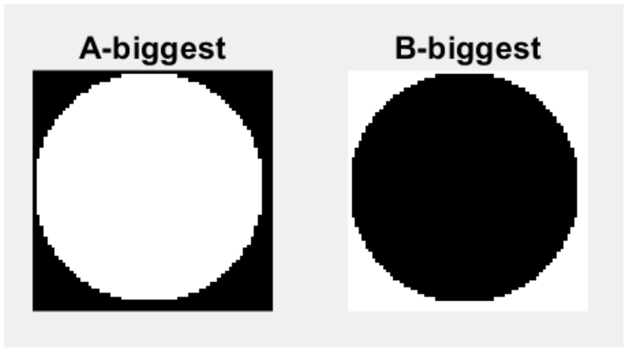


Figure 9 Structural element for finding the largest disks

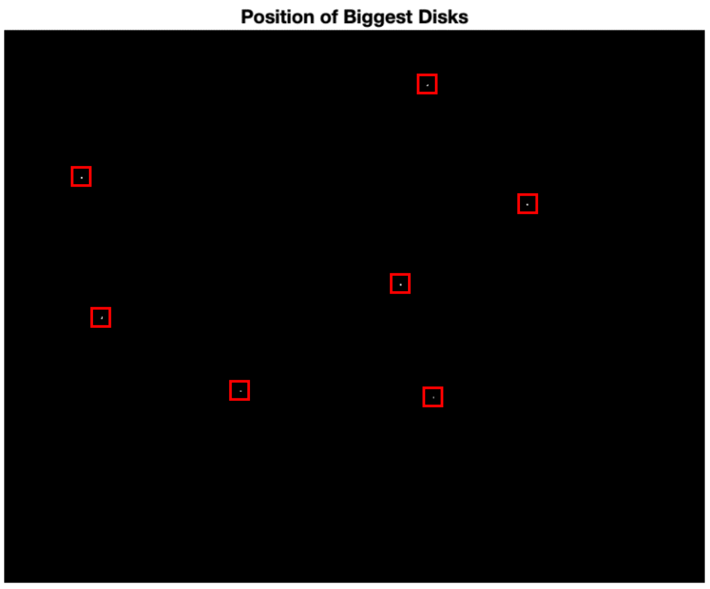


Figure 10 Positions of biggest disks identified

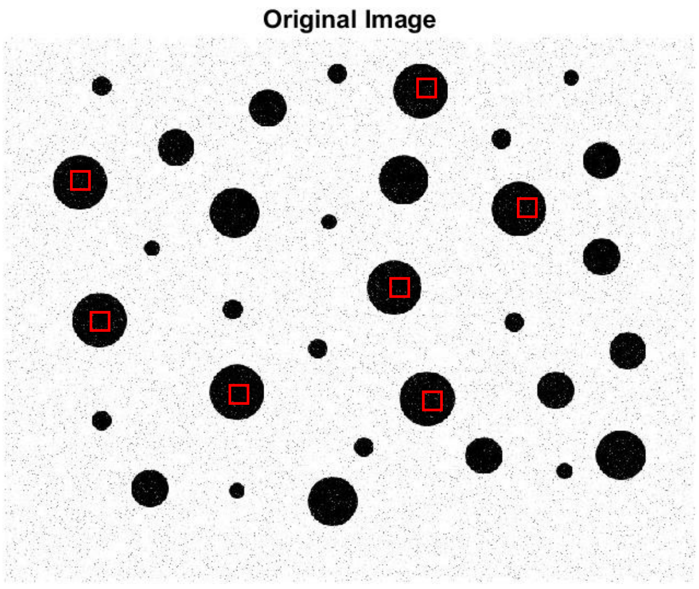
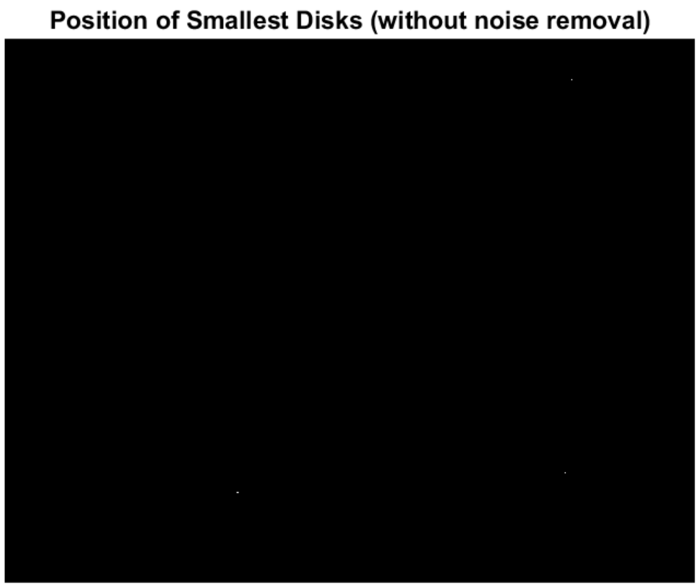


Figure 11 Corresponding biggest disks in the original image



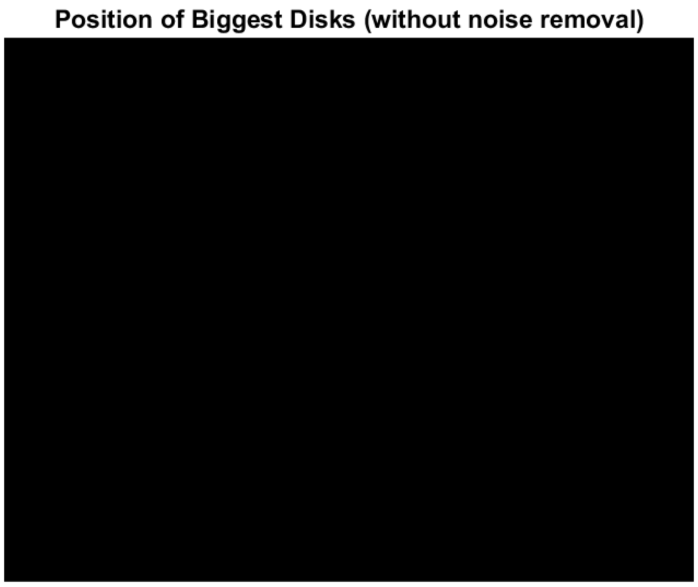


Figure 12 Positions of smallest disks without noise removal

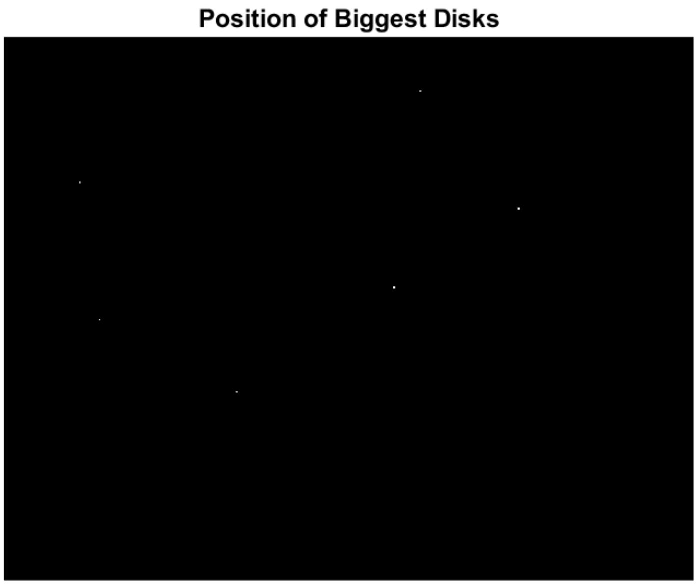


Figure 13 Positions of largest disks without noise removal