Task1

1. implementation approach

Task 1 asks us to do a max/min filter. First used cv2.imread with zero as a parameter to read the gray image.

Then I need to pad the image to make sure that the filter works well in the boundary.

I use cv2. copyMakeBorder function to extend the boundary which length is N//2 and try each way of making a border.

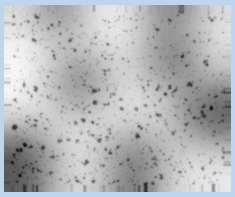
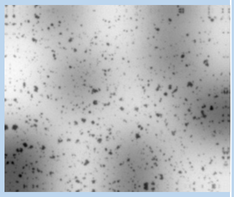
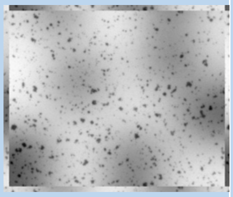
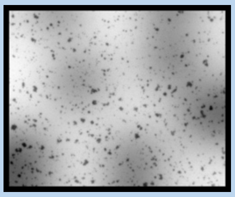
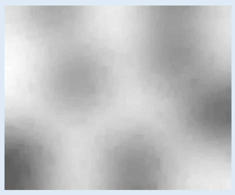
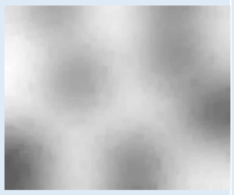
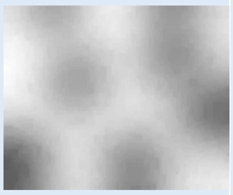
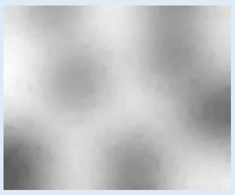
After many attempts, I found that repeating the last pixel on the boundary works best.

After that, I do the max/min filter by checking the maximum and the minimum gray value of each pixel's N*N neighborhoods and replace the gray value by the maximum or minimum so I can get image_A and image_B.

2. Intermediate results and experiences

I attempt to work with these kinds of padding types and I think to REPLICATE and REFLECT should work the same because the maximum and minimum value should be the same in this situation, and BORDER_WRAP and BORDER_CONSTANT may cause deviation on maximum or minimum.

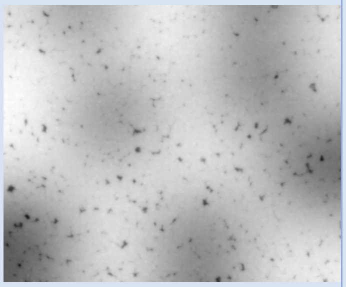
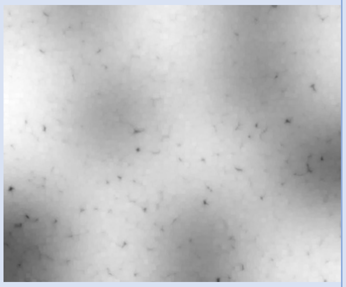
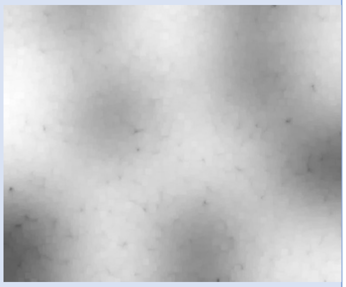
I chos4.ed BORDER_REPLICATE because the build-in cv2.medianBlur function in OpenCV uses the same way.

Padding way	BORDER_REPLICATE	BORDER_REFLECT	BORDER_WRAP	BORDER_CONSTANT
After padding				
After max-min filter				

3. The best value for parameter N

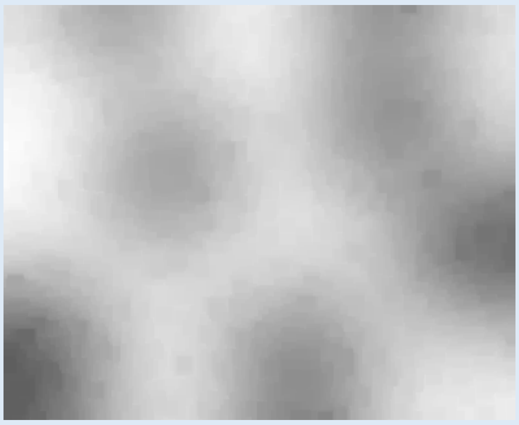
After attempt N from 3 to 41, 13 is the smallest value of N that causes the dark particles in I to disappear altogether in image A.

Because there is no significant change after increasing N over 13 but it will increase compute time, so I choose N as 13.

N	3	5	7
A			
N	9	11	13
A			
N	15	17	19
A			

4. Background estimate image B

Image_B: (N=13)



Task2

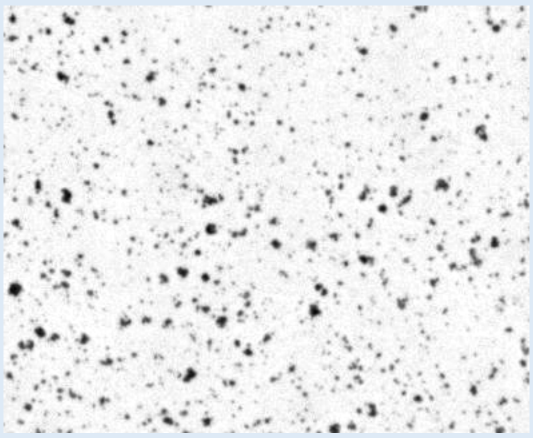
1. implementation approach

Because the direct subtraction on uint8 datatype will cause overflow problems such as 100-200=0.

I convert the data type to int32 to avoid this overflow and then use contrast stretching to keep the range of gray value between 0 and 255 and convert it back to unsigned int8.

2. Output image O

Image_O



Task3

1. implementation approach

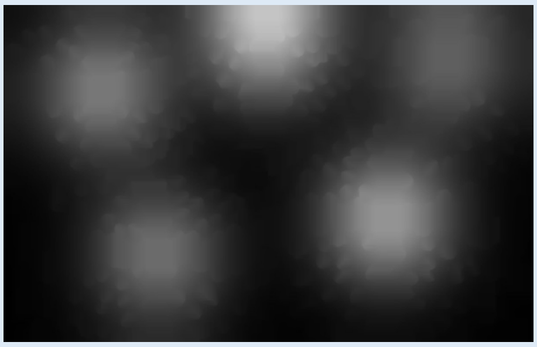
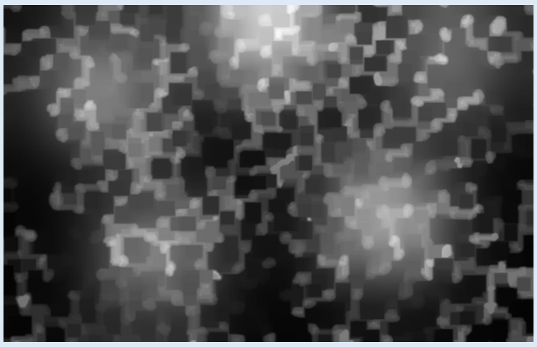
The function of task3 is similar to task 1 and I just modified the function max_min_max_filtered() with parameter M. If M=0, this function will return min_filtered(max_filtered(img,n),n), if M=1, this function will do min filter first and max filter second.

2.Image with different value of M

Obviously, we cannot estimate the background when using parameter M=0.

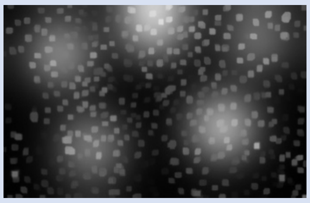
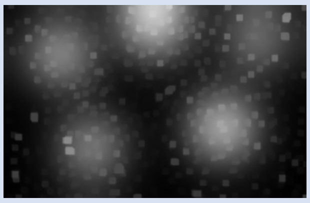
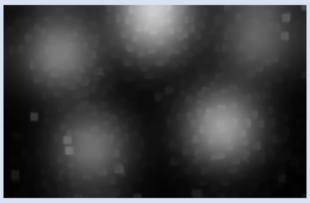
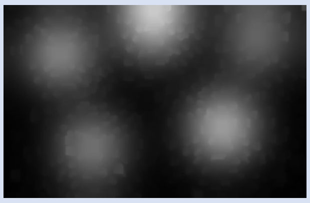
The max filter treats the dark dots as noise and the bright part as background but the min filter bright dots as noise and the dark part as background.

So we should set parameter M equal to 0 when the image's background is bright and noise dots is dark such as Particles.png and set to 1 when opposite such as Cells.png.

M=1	M=0
	

3. The best value for parameter N

I think the best value for parameter N is 30 because the increase of N cannot improve the quality of image_B but costing more time.

N	15	20	25
A			
N	30	35	40
A			

4. Background image B and output image O

Image_B
Image_O