Assignment 1 Report
z5225226

How to run my code

Run the code and input the name of image, the value of parameter M and N. M is 0 when max-filtering first and then min-filtering. On the contrary, M is 1. N is a free parameter for the filter window size.

```
# Please input the name of original image here
filename = input()

# Please input 0/1 here.
# If the background is dark and the objects are bright, choose 1
# If the background is bright and the objects are dark, choose 0
M = int(input())

# Please input the parameter N for N*N filter window
N = int(input())
```

Task 1

In the task one, the original input image is 'Particles.png' like Figure 1 below. It is a picture with bright background and dark object. We need to remove the background from the original image. Here I set parameter M as 0. (The reason will be mentioned in task three.)

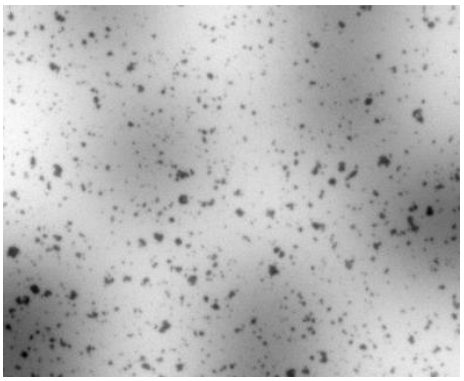
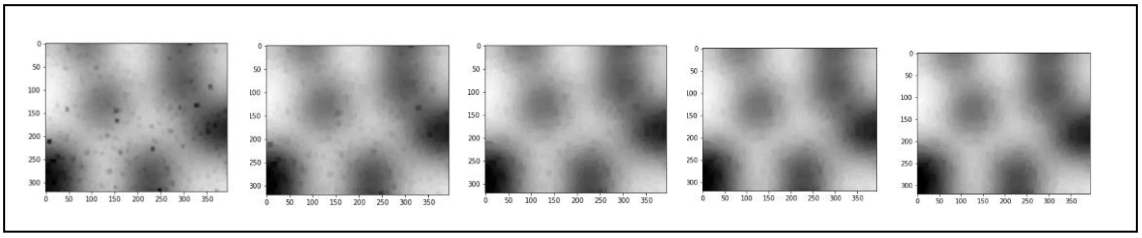


Figure 1

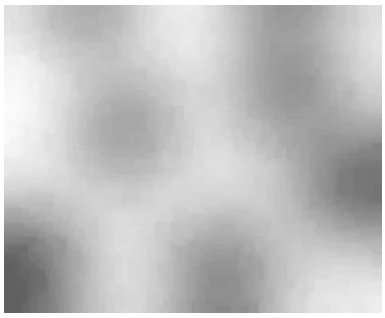
Firstly, I read this image as a grayscale image and create image A and image B as intermediate image here. For the picture with bright background, I use the N*N filter window to get the maximum gray value in the neighborhood for each pixel of original image and write it to image A. Then I use the same filter window to get the minimum gray value in the neighborhood for each pixel of image A and write it to image B. Finally, I get image B as the background.

This filter window is a N * N square with each pixel of the original image as center, which means all the pixels int(N-1/2) from the center are the neighbors. The filter window may exceed the boundary of image, so I reset it as the value of boundary if it exceeds. Because of int(N-1/2), N=4 is same as N=3 and I only consider odd number as N in the subsequent experiments.

To find the smallest N which causes the particles to disappear, I tried N=7,9,11,13,15. And the image Bs are below:



The smallest N I got is 15, because all the dark objects disappeared. When N = 15, the image B is below:



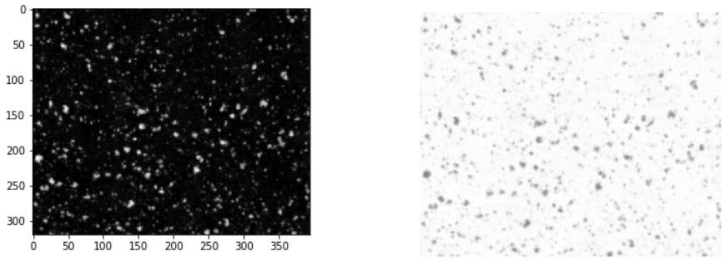
Array of Image B and Image B

It is obviously no pixel is particular different or much darker than its neighbors according to its array.

Task 2

In this task, I need to removing the background from the original image. I used subtract function in opencv3+. But what I got after calculating cv2.subtract(image B, original image) is a reversed picture like below, and using this function will make all negative values as 0. Missing value will sharp the image.

So I subtracted this image B and original image using regular '–' operator and added 255 to get the correct output.



Reversed output and output image O

Task 3

In the task, the original input image is 'Cells.png' like Figure 2 below. It is a picture with dark background and bright object.

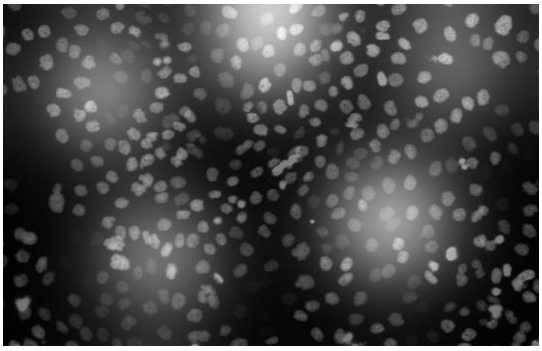


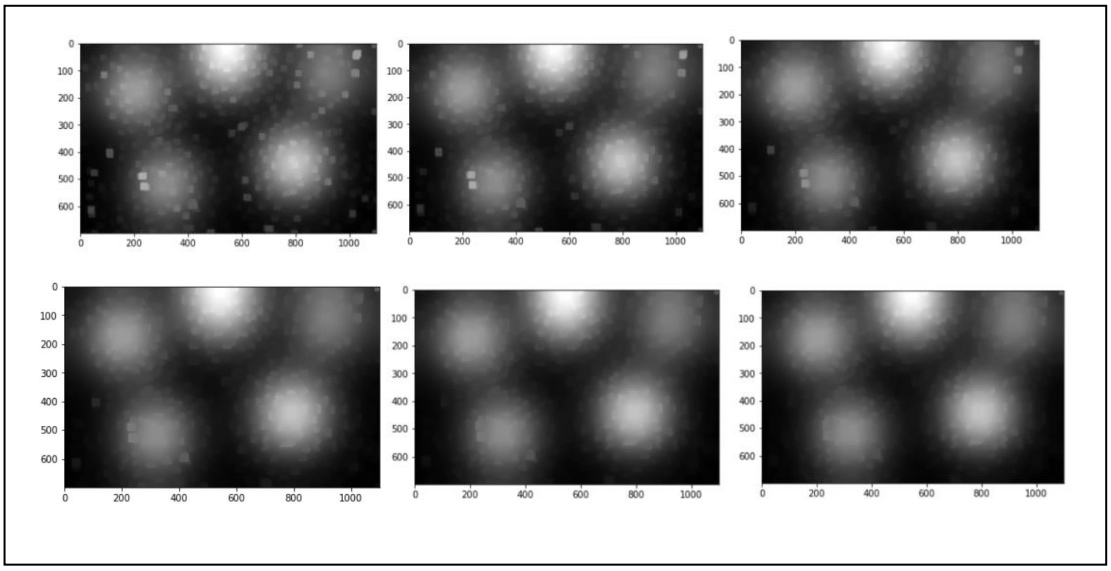
Figure 2

The way to remove background here is much similar to the task one. But here the parameter M needs to be set as 1 rather than 0 in task 1. Because in the task one, the background is bright and objects are dark. If I want to make the dark objects disappear, it is obviously to make the isolated objects bright like its neighbors. Therefore, I perform max-filtering here. However, after max-

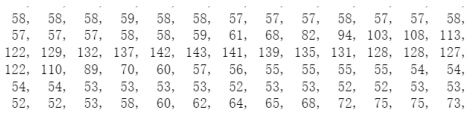
filtering, what I got will brighter than the original background. And then min-filtering needs to be used to make values much closer to the actual background.

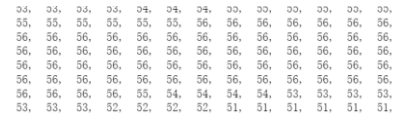
But since the background is dark, I need to do min-filtering first and then max-filtering. It is because if I want to get a dark background, I need to remove all bright objects. And min-filtering is a method to reduce the area of bright areas. Then the image A I got may darker than the original background, so I use max-filtering as a correction. In this way, the background I got will much similar to the original background.

To find the smallest N, I tried N from 20 to 31. And the image Bs I got when N is 21,23,25,27,29 and 31 are below:

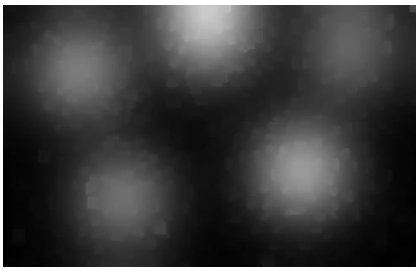


It seems like there is no changes between the image Bs when N=29 and N=31. Thus, the good value I chose for Cells.png is N=29. From the array of original image (first image below) and image B (second image below), it is obviously that there is no special pixel between the neighbors. All objects disappeared.





Then I did the subtraction (O = I - B) using regular '-' operator to get the image O. The background B and the output image O is below:



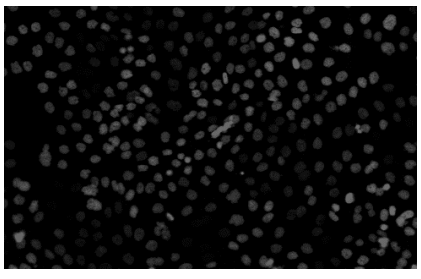


Image B and Image 0