Rongtao Shen
z5178114

Assignment Report

Task 1

There are two steps in this task. First, it should create a max-filtered image from the input image. Then, using the min-filtering method to make a correction on the max-filtered image and create a min-filtered image.

As for max-filtering, I write a function called max_filtered and there are two parameters in this function, one is the input image and the other is the size of neighborhood. In this function, I firstly get the height and width of the input image. Then I create a new image with the same size as the input image. Next, we need to add paddings around the input image because there are no enough neighbors for the pixels in edge if we do not add paddings. Therefore, I create a new image with paddings by using cv2.copyMakeBorder. The borderType I use in this method is cv2.BORDER_DEFAULT which has a better performance than using cv2.BORDER_CONSTANT because it will use the mirror pixel as the padding pixel, which can decrease the difference with the original image. In addition, the padding size can be computed by the size of the neighborhood, which is N // 2. Finally, I use two loops based on the height and width to traverse all the pixels in the padding image, using numpy.max() method to get the maximum gray value in neighborhood (include the middle pixel) for each pixel and assign this value to the corresponding pixel in the new image, which can get a max-filtered image.

As for min-filtering, I also write a function called min-filtered. The difference with the max-filtered function is that we need to find the minimum gray value in a neighborhood around each pixel in the image by using numpy.min() method. The purpose of this min-filtered function is to make a correction on the image after max-filtering because max-filtering causes the gray values in image A to be higher than the actual background values in image I. Therefore, doing the min-filtering after max-filtering could get an accurate background image.

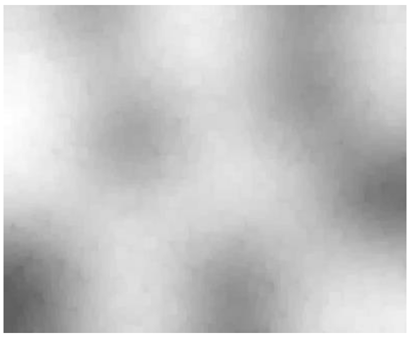


Image A

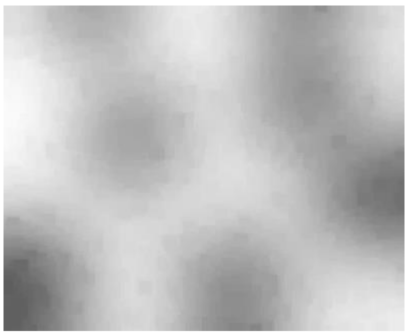
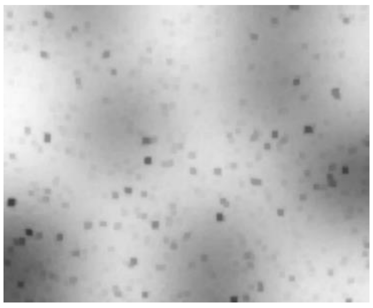
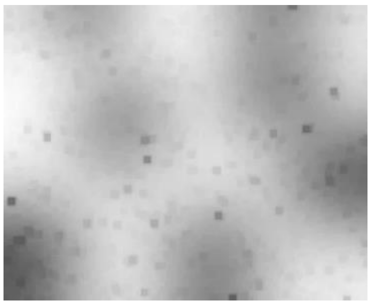


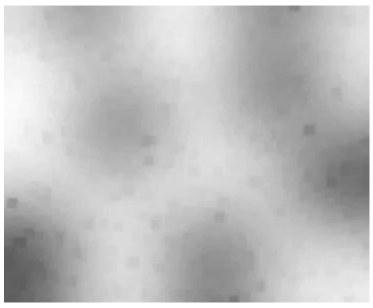
Image B

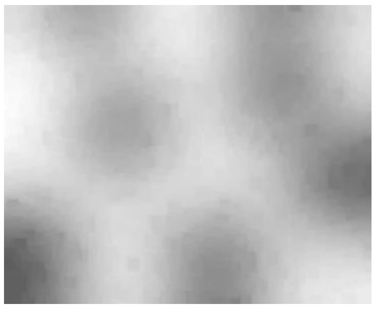
We can see in these two images above some shadings have been removed after max-filtering because the shading is dark and easily filtered by the max-filtering. Therefore, in order to get an accurate background image, we need to do the min-filtering after max-filtering to solve the problem of the gray values in image A to be higher than the actual background values in image I.

According to experiment with different values of N, the smallest value of N that causes the dark particles in I to disappear altogether in image A is 11. In the experiment, I started to set a small size of the neighborhood like 5 at the beginning, the dark particles do not all disappeared (we can see the image below). Then when we increase the value of the size of the neighborhood, the dark particles disappear obviously and when the value of N is 11, the dark particles almost disappear.


N=5

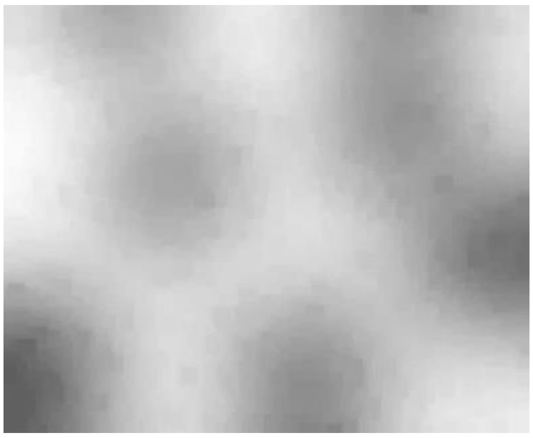

N=7


N=9


N=11

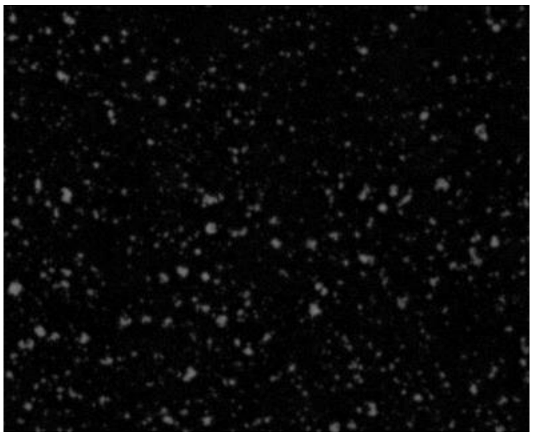
The reason why this value of N causes the particles to disappear is because the value of N affects the value we get in the neighborhood. If we set a small size of the neighborhood, it is hard to eliminate the black particles to get the background because there are many dark particles whose pixel value is small in the input image. Instead, when we increase the size of the neighborhood, it is easy to eliminate the black particles under the maximum filter because the input image is mainly based on a brighter background, which we can get a relatively big value for each pixel.

The image B computed from Particles.png is below:

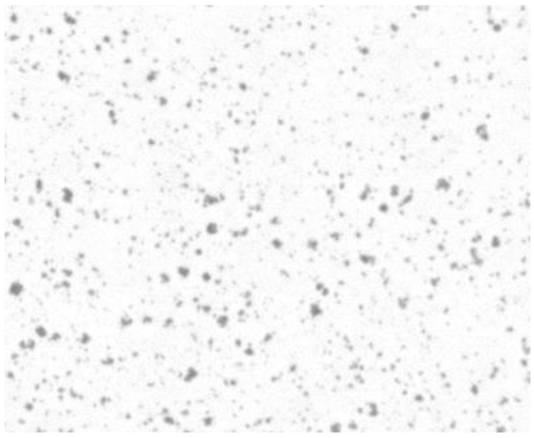


Task 2

After max-filtering and min-filtering, the algorithm can estimate the background B of the input image I. Since this is an image with bright background and dark objects, if we directly subtract the background image pixel by pixel from the input image, it will produce negative values, which will get a dark image with all pixel value is 0. To handle this problem, I change the type of the matrices to a signed integer type (numpy.int32) and then use I-B+255 to get the image O. The reason to add 255 is because we need to change the negative values to the positive values, which we can get the no shading image with bright background and dark particles. Finally, I change the image O back to the type numpy.uint8 and get the final image O. Another method to get the image O is that we can firstly subtract the image I from the image B pixel by pixel to get an image with dark background and bright particles, which can be called image_subtract. Then use an image with all pixel value is 255 to subtract the image_subtract and finally get the image O.
We can see image_subtract below, which has the dark background and bright particles.



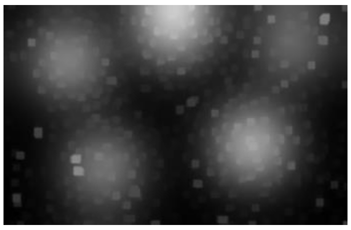
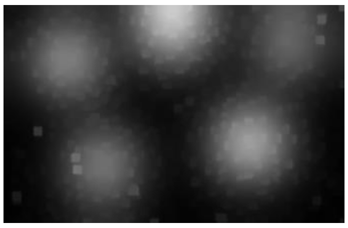
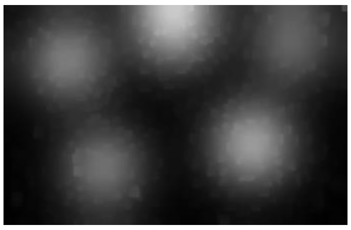
The image O computed from Particles.png is below:



Task 3

In this task, it is similar to use max-filtered and min-filtered function mentioned in task 1. But the main difference between the task 3 and task 1 is that we need to reverse the max-filtering and min-filtering. In task 3, we should use min-filtered function first to get a new image, which can remove the bright objects and get the background image. Then, we use max-filtered function to make a correction and get a good background image. After getting a good background image, we subtract the background image from the input image pixel by pixel and finally get the image O with no shading.

The reason why M=0 or M=1 for the different images is mainly because these two images have different color of the objects and background, in Cells.png the objects are bright and the background is dark, whereas in Particles.png the objects are dark, and the background is bright. As for the Cells.png, in order to get a good background image, we need to remove the objects from the input image to get a new image. Since the objects in Cells.png is bright, which have the high gray value, therefore, we need to do min-filtering first to retain the dark background and remove the bright objects. In addition, since the input image exists some relatively bright shading, which may be partly removed by the min-filtering, therefore, we should also use max-filtered function to make a correction after min-filtering. However, in Particles.png, since the particles are dark, which have the low gray value, therefore, we need to do the max-filtering first to remove the dark particles and get a good background image. The best value for parameter N that I choose in task 3 is 29. Since the sizes of the images are different and the sizes of the objects between Cells.png and Particles.png are also different, the Cells.png is larger than the Particles.png, therefore, dealing with larger image and larger objects in images is a matter of changing the value of the neighborhood parameter N. If we set a similar value of N with the task 1, the performance of removing objects to get a good background image is bad. Therefore, we need to increase the neighborhood parameter N because the image size and the size of objects is larger, setting a small value of neighborhood parameter N is not easy to get a small value of pixel to remove the bright objects. So, after some experiments, we can find a suitable value which is 29.

N=21 N=25 N=29

As the image show above, we can see when N is set to 21 the background image we can get is not good and the image with N equals to 25 is better than the image with N equals to 21. And when we set N to 29, we can get an accurate background image.

The images B and O computed from Cells.png is below:

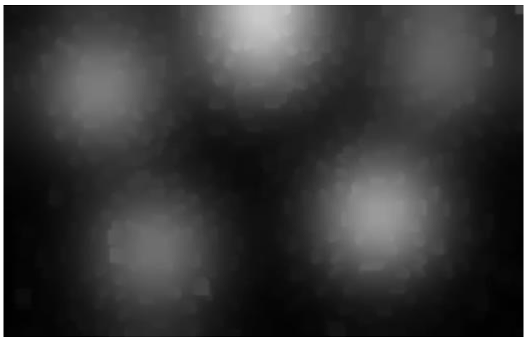


Image B

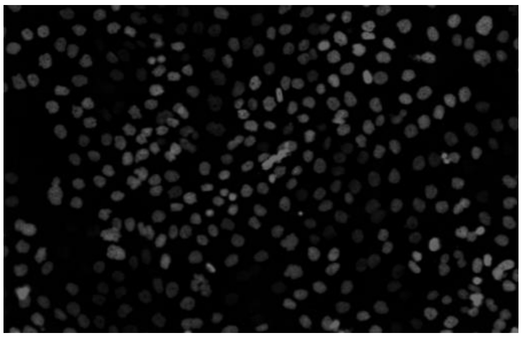


Image O