UNSW
SYDNEY

COMP9517 Computer Vision

Assignment 1

Report

z5240221 Jatin Gupta

Table of Contents

1 Task 1

1.1 Explanation of the chosen implementation approach

The implementation uses NumPy and cv2 functions to read and manipulate the picture. The assignment specification instructed us to read an image in greyscale and apply max filter on it. Max-Filter the process of getting each pixel value of the image and finds the maximum grey value among its N nearest neighbours including itself. Here N is a user-defined parameter which is given by command-line argument for this implementation.

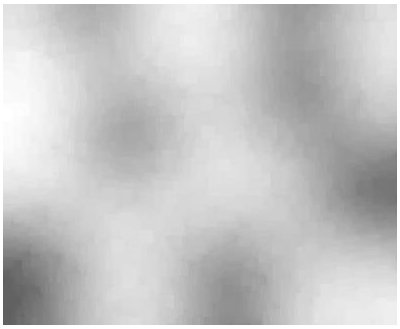
To run the task1.py file:

1. User needs to make sure if the required requirements are installed. If not, then requirements can be installed by requirements.txt file. Since I have used conda env, hence it has more than required libraries.
2. Run command in the directory "python task1.py N" where N should be an odd integer N > 1.
3. The output will be saved in the root as (for N = 13)
    a. task1_A_N_13.jpg. (Image A as per spec)
    b. task1_B_N_13.jpg. (Image B as per spec)

The max-filter is applied by copying the image into another image with (N-1)/2 pixels zero paddings around the image. We selected 0 as the values of the padding pixel as we are doing Max Filtering, it will not affect the results because the pixel value nearby will be either 0 or greater than zero. Then we iterate through all the pixels one by one and find the maximum value grey neighbour pixel value and assign to the pixel which is in question. We find the max using Numpy function called amax(). This function efficiently converts the 2d array to 1d array and find its max value in O(n) time complexity. Finally, we check the image for any overflow as a grey image can only have datatype uint8 and values between 0 to 255 and return the image.

For Min-filter image, we follow the same process except we do not pad the image with 0 pixels. We pad it with 255-pixel value because we have to find the min of the neighbours. If we apply 0 as the padding, we will get 0 as the min value always when finding min.

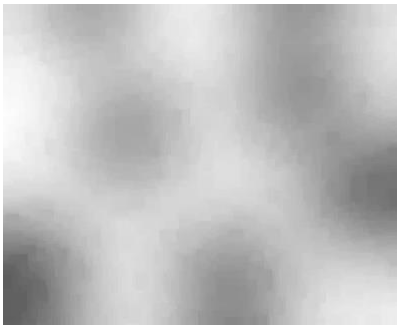
1.2 Discussion of intermediate results and experiences and the best value for parameter N



I ran a loop and saved all images from N value 3 to 50. And analyzed all the images. I saw with each increasing value of N from 3 the black spots were disappearing. I found out that from N = 11, we start getting clear images. But we still see one visible black spot. Hence, we move one value up to **N = 13** where we cannot see any black dots available. Hence, we take that value as the final value for task 1. We came to this conclusion as most of the dots/spots are 13x13 pixel wide.

Task 1 result image A at N = 13

1.3 Background estimate image B



For Min-filter image, we follow the same process except we do not pad the image with 0 pixels. We pad it with 255-pixel value because we must find the min of the neighbours. If we apply 0 as the padding, we will get 0 as the min value always when finding min.

Task 1 result image B at N = 13

2 Task 2

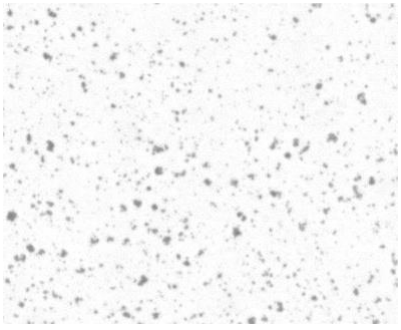
2.1 Explanation of the implementation approach

For task 2 implementation, I decided to import function from task1 and use them for task 2. I used max-filter function then resultant of that function is passed to min-filter. The resultant of the min-filter is subtracted from the original image. Since the min-filter produces an image which is close to higher values of 255, hence the solution will be negative in many pixels. To bring back negative pixels to positive range we add 255 to the whole image to get the output. I have also added a version to normalize the image using cv2.normalize function.

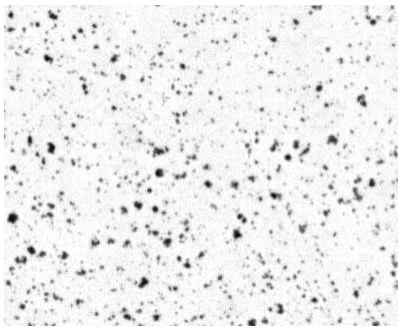
To run task 2:

1. Run command in the directory "python task2.py N" where N should be an odd integer N > 1.
2. The output will be saved in the root as (for N = 13)
    a. task2_O_N_13.jpg.
    b. task2_O_normalized_N_13.jpg.
    c. task2_background_N_13.jpg.

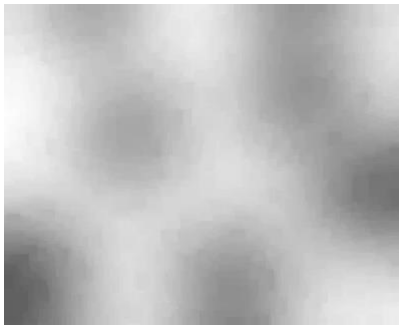
2.2 Output image O



Task 2 result image O at N = 13



Task 2 result image normalized with cv2.normalize function, O at N = 13



Task 2 image background at N=13

3 Task 3

3.1 Explanation of the generalization approach

I have tried to keep the implementation which can handle any size of the image. It dynamically takes in the height and width from the input image. I have tried to make the code modular and we can use min and max filter methods independently. Additionally, I have made min_max and max_min function which performs min filter then max filter and max filter then min filter respectively. All the functions take in free parameter N. I have hardcoded the image files for this implementation for ease of marking, but it can be changed to command line arguments as well with 2 lines of code. For this implementation, we have taken manual values for M and N via command line, but we can extend this program to get background colour of the image and selecting a value of M automatically.

3.2 Explanation why M = 0 or M = 1 for the different images

Images with lighter backgrounds should prefer M = 0 as it will apply max filter and then min filter. This is preferred because when the background is light, most of the pixel values in the background will be close to 255. Hence when we apply max filter it will try to make the image more towards the whiter side. And then when we apply min filter which will intensify the shaded regions. And then we result from min filter (B) is subtracted it from I it will get rid of the shaded regions and restore black dots.

And opposite works for the dark background image where we apply min filter. Min filter masks over the darker spots and make the area darker. And then we apply the max filter on it to enhance white shading. Then when we subtract the result of the max filter from the original image I to remove the white shading.

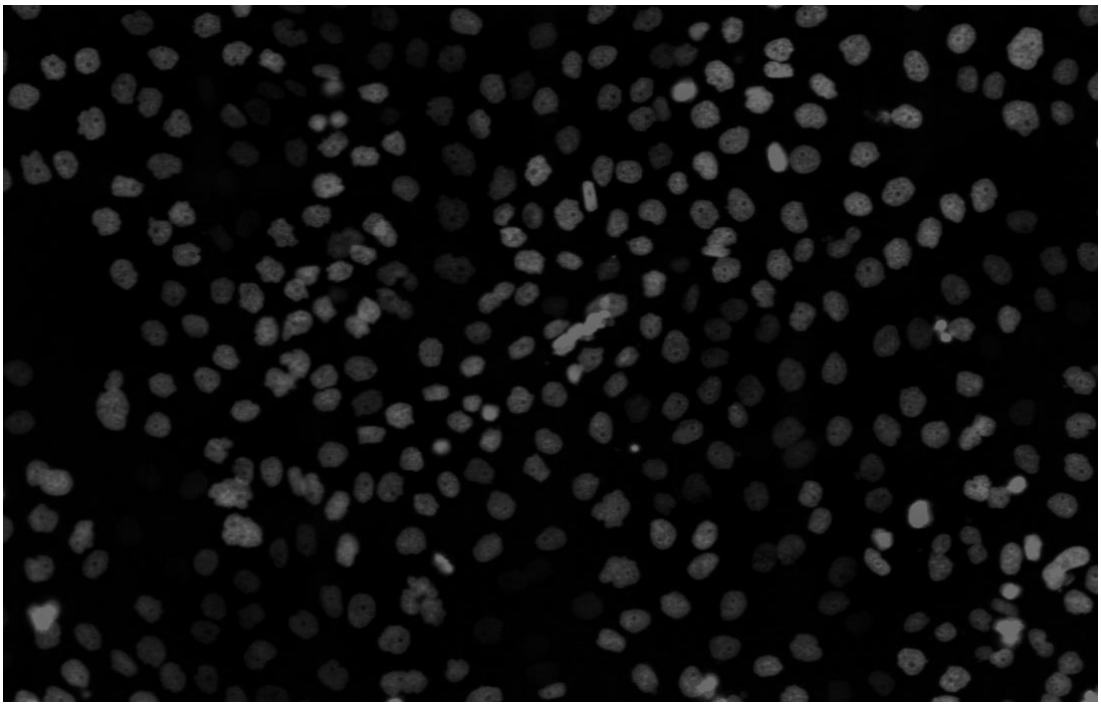
3.3 Discussion of the best value for parameter N

For task 3 since the particles were bigger and took more pixels hence the N value was higher than the previous task. At N = 33, we get decent image but at N = 35 we illuminate the right-top cell. Hence, I feel **N = 35** is a decent parameter for this image.

3.4 Background image B and output image O

To run task 3:

1. Run command in the directory “python task3.py N M” where N should be an odd integer N > 1 and M is either 0 or 1.
2. The output will be saved in the root as (for N = 37 and M = 1)
    a. task3_N_35_M_1.jpg.
    b. task3_N_35_M_1_background.jpg.



Task 3 result image O at N = 35, M = 1

Task 3 background image at N = 35, M = 1