Computer Vision

COMP9517

Assignment 1

T2, 2020

Kovid Sharma (z5240067)

Table of Contents

Assignment 1
Kovid Sharma - z5240067

Introduction

For this assignment I've prepared a video showing how the background is computed and subtracted from the input image and the output improves and then degrades with increasing N value.

The video can be found in the zip folder and on these YouTube link:

- Particles : https://youtu.be/rtDDlY_Ev4w
- Cells : https://youtu.be/-d0YQ52EycA

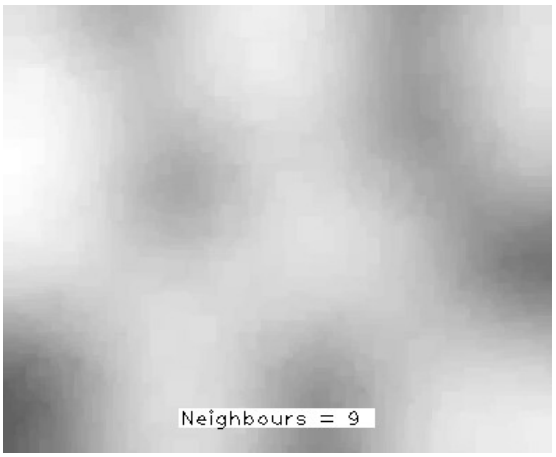
The images shown in the assignment have a label 'Neighbours = N' that is put on the images using cv2.putText. For better readability the background for the 'text label' is made white by using this calculation : Image[yval-12:yval+2, xval-1:xval+140] = 255 where,

xval = int((cols/2) - 70)

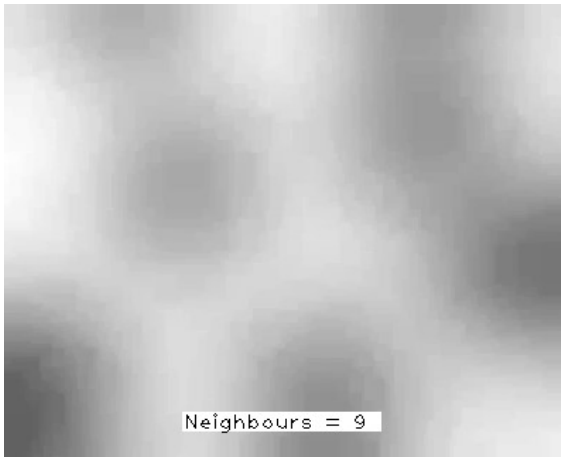
yval = int(rows - 20)

Task 1

Background is computed by finding min filtering then max filtering. #see python code



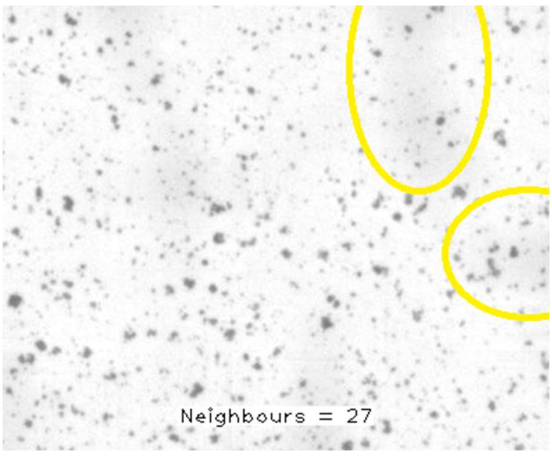
Min Filtered A

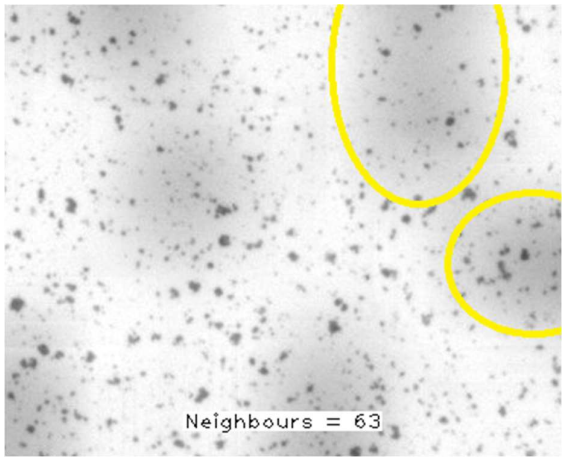


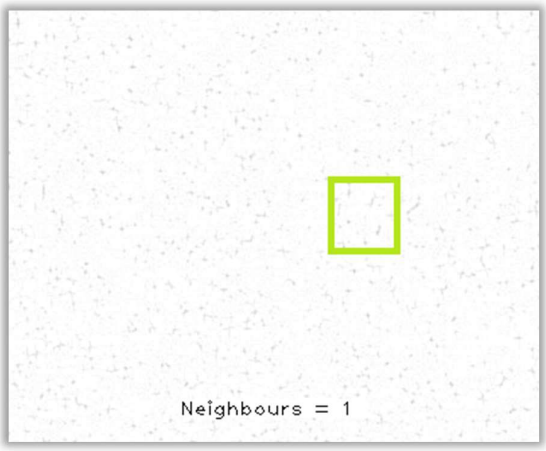
Max Filtered B

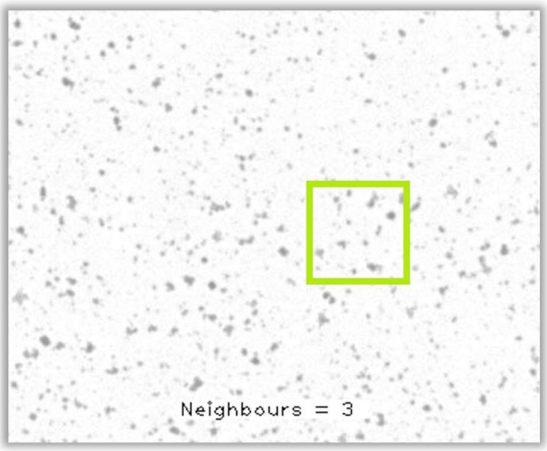
We use min-filtration first and then max since the background is white dominant.

The black dots disappear more & more but background creeps back for a higher value of N:





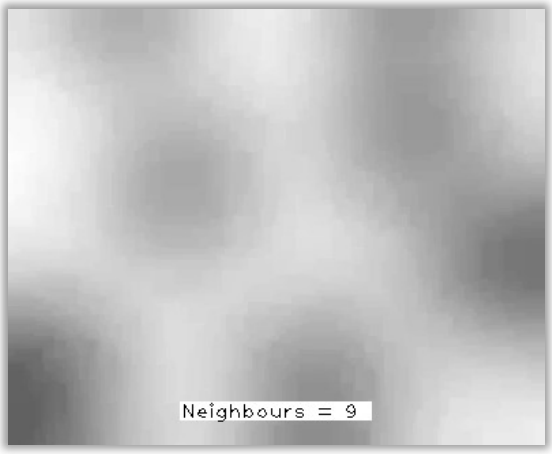




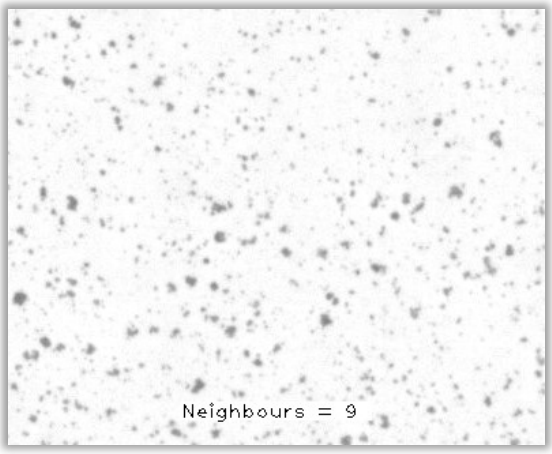
If the value of N is too low, the particles objects are not that dark, meaning the background has a lot of foreground information that is being lost.
As seen in the two images above and below, going anything below or way higher the desired N value results in the image not being clear(low N) or the background returning(high N).

Task 2

We can easily do arithmetic and logical operations between 2 images. Images are nothing but matrices. NumPy arrays can be easily subtracted, point by point using '-' operator.
Since we have the background sorted, the output image is simply calculated by subtracting the background from the original image.



Background Image



Final Image

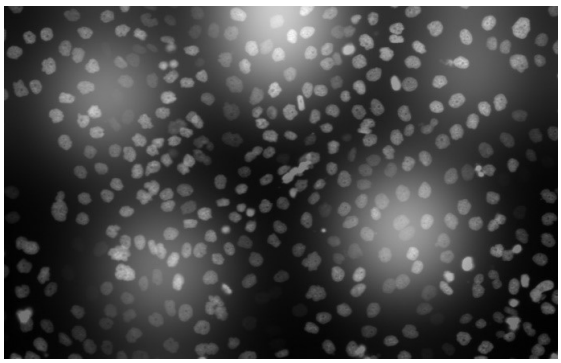
I've done this calculation in the python code under the comment '#removing background'.
If the objects are black and the background is white, O is calculated by I - B + 255
If the objects are white like in cells.png and the background is black, O = I – B.
To automate this, I've written a function under the tag '#find percentage of black background and choose M automatically' which automatically detects the background chooses which formula to use for best results.

Task 3

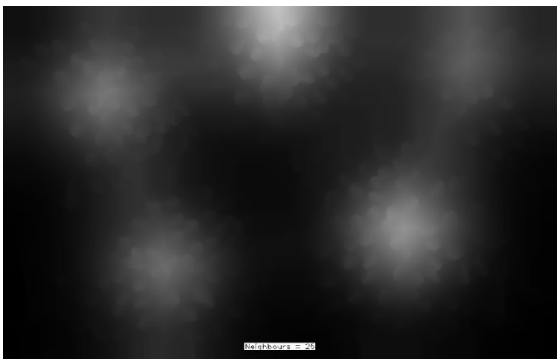
Particles.png shape is 700x1100 pixels. That's a total of 770,000 pixels. **Cells.png** shape is 394x320 pixels. That's equal to 126,080 pixels. Therefore **Cells.png** is more than 6 times bigger than **Particles.png.** Dealing with different images is a matter of changing the breadth and width of the image. This can be found using .shape function which returns the rows and cols of an image/matrix.

Particles.png requires M = 0, and **Cells.png** requires M = 1 :

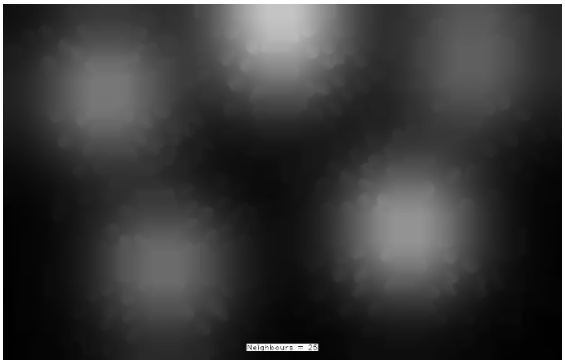
This is because the object pixels are of different value and the background is opposite in both images. Black pixels correspond to 0 value and white pixels are 255.



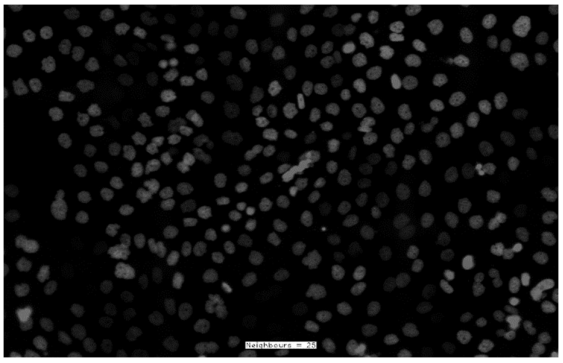
Input Image



A



B

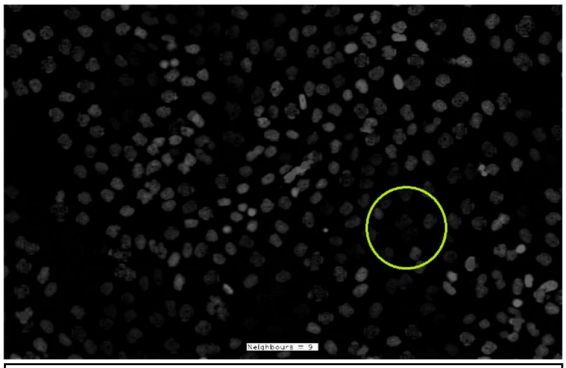


Output Image

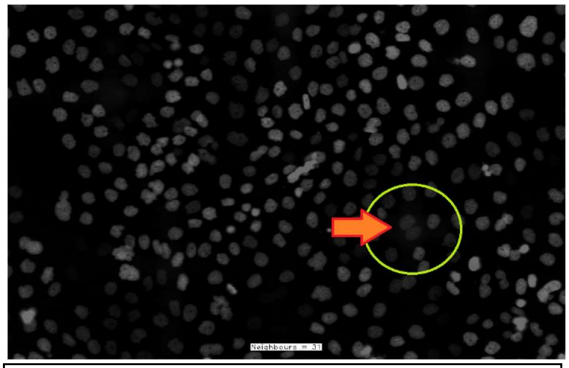
The objects in **Particles.png** are small, it needs a smaller N value to filter out the background. Since the objects in **Cells.png** are bigger, a bigger N value is required to filter out the background.

Image	Particles.png	Cells.png
Image Size	126,080 pixels	770,000 pixels
Object Size	11x10 pixels	21x24 pixels
N Value (Neighbours)	9	25
Runtime	3.2 secs	26.2 secs

Here we can see there is a direct co-relation between the size of the objects in the image and the number of neighbours needed to remove the background.



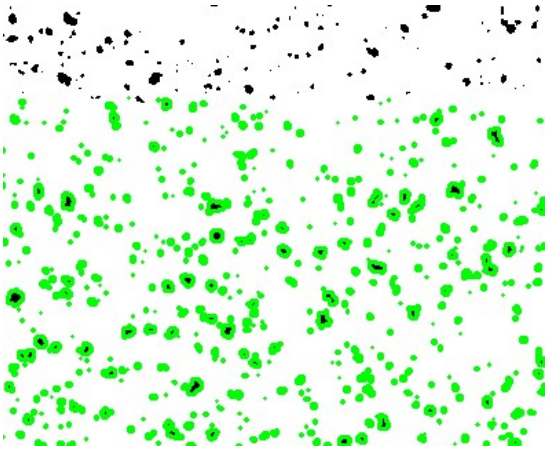
When the N value is too low, as is here at N = 9, the darker cells are not that visible as the pixel value is quite low. The size of the cell is also quite large compare to the objects in Particles.png



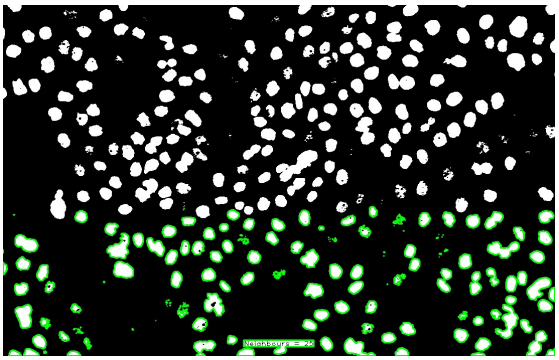
If the N value is set too high, as is here at N = 31, the darker cells are quite visible but the noise around the cells also increases.

Finding the size of the objects in images:

Using in-built libraries in OpenCV, I was able to find the area occupied by the objects. This would give us a fair estimate of the value of N.



Pixel Area: 119660
Image Area: 126080
Image - Pixel Area 6420
Total number of black shapes: 464
Object Dimensions: (9.816389065738598, 11.816389065738598)
I calculated for Particles.png that the average object was around 10x11 pixels.
Functions used - cv2.drawContours, cv2.threshold, cv2.countNonZero



Pixel Area: 134463
Image Area: 770000
Image - Pixel Area 635537
Total number of black shapes: 424
Object Dimensions: (21.212439984269187, 24.212439984269187)
Similarly, I calculated for Cells.png that the average object was roughly around 21x24 pixels.

Note: If the images are not clearly visible, here is a Google Drive link for all these images, videos and python code.
Link :
https://drive.google.com/drive/folders/1FxZur3hTLb5bGYmpB_lx87GqbTItVycq?usp=sharing

References

1. Matplotlib https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.title.html
2. OpenCV https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_core/py_basic_ops/py_basic_ops.html
3. NumPy https://numpy.org/devdocs/user/quickstart.html
4. Computer Vision: Algorithms and Applications http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf