Image Processing - Assignment 3

Task 1: Theory [1 point]

a) **[0.2 points] Define opening and closing in terms of erosion and dilation. What happens when open and closing are applied multiple times on the same image?**
Opening is when one performs an erosion, and afterwards a dilation with the same structuring element to an image that is:
$A \circ B = (A \theta B) \oplus B$

Closing is the same thing, only in opposite order (dilation -> erosion).
$A \bullet B = (A \oplus B) \theta B$

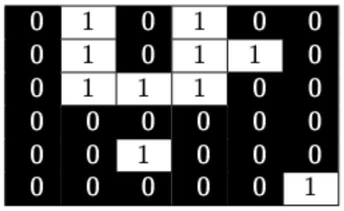
The output will change after the first operation (either erosion og dilation), but all subsequent operations will not change the output, because the corners are already rounded.

b) **[0.1 points] Smoothing of an image is often done before performing edge detection. What is the purpose of smoothing the image before edge detection?**
When performing edge detection we take the spatial derivative to detect change. But noise will cause the output to be useless, because noise has a lot of change. When smoothing before detecting edges, the noise gets smoothed out, and the output becomes useful.

c) **[0.2 points] The Canny edge detector uses a method called hysteresis thresholding. Shortly explain how hysteresis thresholding works.**
Canny uses hysteresis thresholding to figure out what edges are actually real edges. There two thresholds resulting in 3 partitions:
1. Edges with greater values than the **high threshold** are sure edges
2. Edges with lower values than the **low threshold** are not edges.
3. Edges between the two thresholds are edges only if they are connected to sure edges.

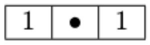
d) **[0.2 points] Why do we use hysteresis thresholding instead of a single threshold?**
Because it is very difficult to find the correct threshold value which divides all the true edges from the non-edges. If we pick a value which is high, we might only get the dominant edges, but this could also lead to some holes where the edges are not strong enough. If we chose a too low threshold, we could end up with all the desired dominant edges, but other undesired edges and useless noise.

e) **[0.3 points] Determine the dilation A $\oplus$ B of the binary image in Figure 1a. Use the structuring element in Figure 1b.**



0	1	0	1	0	0
0	1	0	1	1	0
0	1	1	1	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	1

(a) A 6×6 binary image.



(b) A 1×3 structuring element

Figure 1: A binary image and a structuring element. The foreground is colored white and given the symbol 1. The background has the symbol 0 and is colored black. The reference pixel in the structuring element (b) is indicated by a black circle.

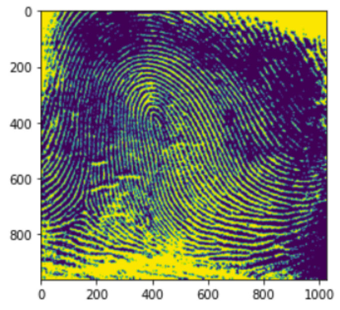
The output pixels are set to 1 if the structuring element hits image pixels

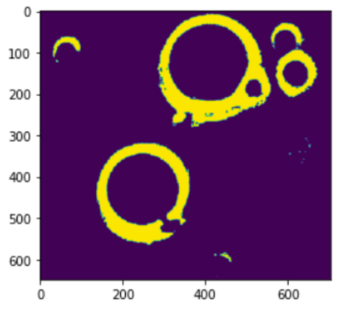
1	**1**	**1**	**1**	**1**	**0**
1	**1**	**1**	**1**	**1**	**1**
1	**1**	**1**	**1**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**
0	**0**	**0**	**0**	**1**	**1**

Task 2: Segmentation [2 points]

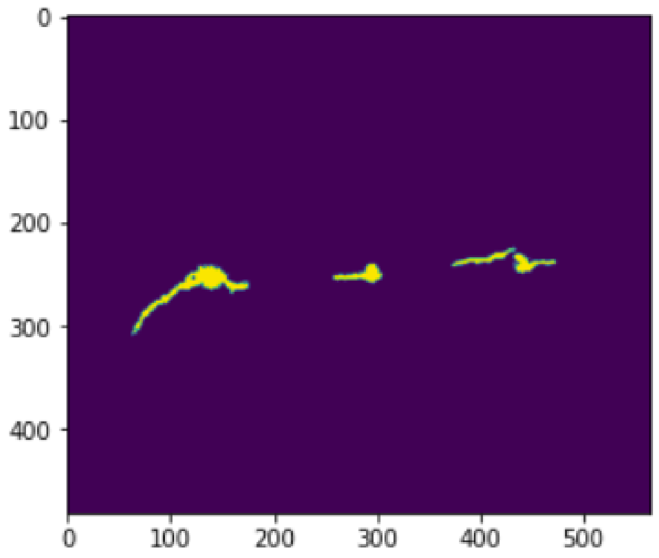
a) **[1.0 points] Implement a function in task2a.py/task2a.ipynb that implements Otsu's algorithm for thresholding, and returns a single threshold value. Segment the images thumbprint.png and polymercell.png, and include the results in your report.**

```
Found optimal threshold: 153
Saving image to: image_processed\thumbprint-segmented.png
Reading image: images\polymercell.png
Found optimal threshold: 181
Saving image to: image_processed\polymercell-segmented.png
```





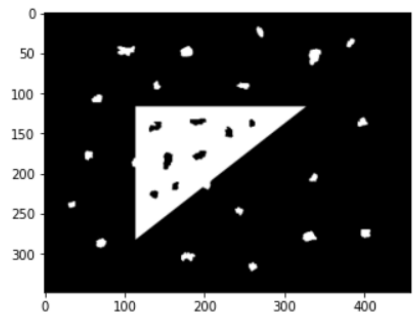
b) **[1.0 points] Implement a function in task2b.py/task2b.ipynb that segments a grayscale image using the region growing method outlined above. Use a Moore neighborhood (8-connectedness) to expand your set of candidate pixels around each seed point. Apply it on the image defective-weld.png and show the result in your report. Use the 4 seed points given in the starter code and use a pixel intensity threshold of 50. Hint: To make sure you implement the code correct, you should test your algorithm with a pixel intensity threshold of 90.**

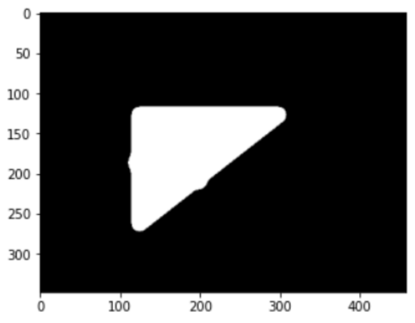


Task 3: Morphology [2 points]

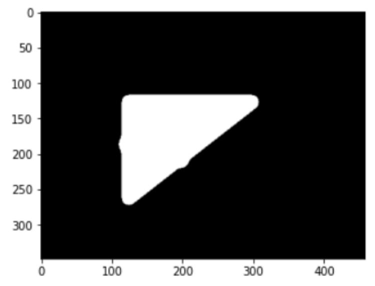
a) **[0.6 points] Use what you know about erosion, dilation, opening, and closing to remove the noisy elements from the image in Figure 3. Implement this in the file task3a.py/task3a.ipynb. Your result should look something like the one in Figure 3b. Include the resulting image in your report and shortly explain how you removed the noise.**
First I created a disk, and used it as the structure on both a n opening and a closing of the original image. I did this because opening is like filtering the inside of the triangle by rolling the structure around the inside of the shape. Closing has the same intuition, but it rolls on the outside of the shape.

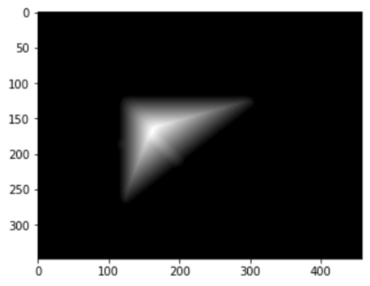
```
Reading image: images\noisy.png
Saving image to: image_processed\noisy-filtered.png
```





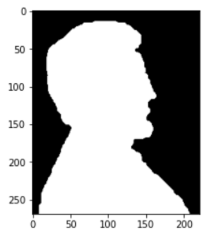
b) **[0.6 points] Implement the distance transform using the erosion method explained above, in the file task3b.py/task3b.ipynb.ipynb. You can use a 3 × 3 structuring element of all ones to get chessboard distance. Test the function on the noise free binary image you got from task (a) and show the result in your report**

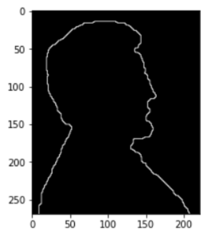




c) [0.3 points] Implement a function that extracts the boundary from a binary image, as defined in Equation 1, in the file task3c.py/task3c.ipynb. You can use a 3 × 3 structuring element of all ones. Show the boundary on the image lincoln.png and include the result in your report.

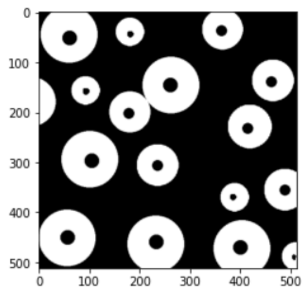
```
Reading image: images\lincoln.png
Saving image to: image_processed\lincoln-boundary.png
```

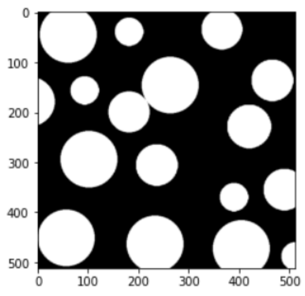




d) [0.5 points] Implement a function that takes in a binary image and a set of starting points indicating position of holes, and applies the hole filling algorithm outlined in Algorithm 1. Implement this in the file task3d.py/task3d.ipynb. Apply the function on the image balls-with-reflection.png and include the resulting image in your report. The position of the holes are given in the starter code. Use 30 iterations (K = 30), and a 3 × 3 structuring element (B) of all 1's.

```
Reading image: images\balls-with-reflections.png
Saving image to: image_processed\balls-with-reflections-filled.png
```


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500