TTK4255: Assignment 1

Due date 24.01.2022

Ivan Gushkov and Christopher Strøm

Problem 1

Task 1.1

a) Convolution is a linear operation since the resultant value of the pixel depends on the value of the pixels around it. So not a point operation.

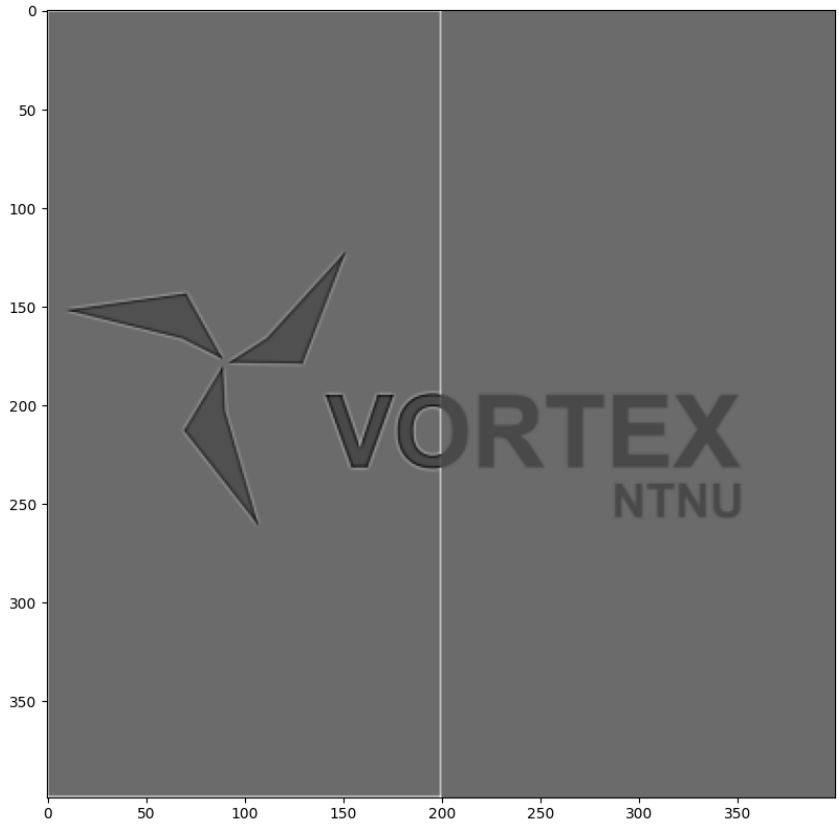
b) Again the pxl value depends on the neighbourhood of pixels.

c) Brightness and contrast adjustment is a point operator as the resultant pixel value can be written as g(x) = h(f(x)) where the new pixel value only depends on the old pixel value.

d) Here we can consider that a pixel is mapped from one place to another in the image. Denoting the start position as a and position of arrival as b, we can see that the value of b after the transformation depends on a pixel in some opportune neighborhood of b, that is a. So this is not a point transformation.

Task 1.2

It can be seen from figure 1 that the kernel has a sharpening effect. This was determined experimentally in python.



Figur 1: The results from applying the kernel to a logo of the best student organization on campus! The filter is applied to only the left half of the image, resulting in a shappened image.

Task 1.3 a) To derive the expression for the half-width h s.t. the value of g outside the window is less than ε is equivalent to solving $g(h) < \varepsilon$, where g is given by (1) in the homework description.

$$\begin{aligned}
\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{h^2}{2\sigma^2}\right) &< \varepsilon \\
\ln\left(\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{h^2}{2\sigma^2}\right)\right) &< \ln(\varepsilon) \\
\ln\left(\frac{1}{\sqrt{2\pi}\sigma}\right)+\ln\left(\exp\left(-\frac{h^2}{2\sigma^2}\right)\right) &< \ln(\varepsilon) \\
-\frac{h^2}{2\sigma^2} &< \ln(\varepsilon)-\ln\left(\frac{1}{\sqrt{2\pi}\sigma}\right) \\
\frac{h^2}{2\sigma^2} &> \ln\left(\frac{1}{\varepsilon\sqrt{(2\pi\sigma^2)}}\right) \\
h &> \sigma\sqrt{\left(2\ln\left(\frac{1}{\varepsilon\sqrt{(2\pi\sigma^2)}}\right)\right)}
\end{aligned}$$

Since h is required to be an integer, the final expression is thus

$$h = \left\lceil \sigma\sqrt{\left(2\ln\left(\frac{1}{\varepsilon\sqrt{(2\pi\sigma^2)}}\right)\right)}\right\rceil \tag{1}$$

b) Using equation 1 h can be calculated to be

$$h = \left\lceil 3\sqrt{\left(2\ln\left(\frac{256}{\sqrt{(2\pi 3^2)}}\right)\right)}\right\rceil = 8 \tag{2}$$

Problem 2

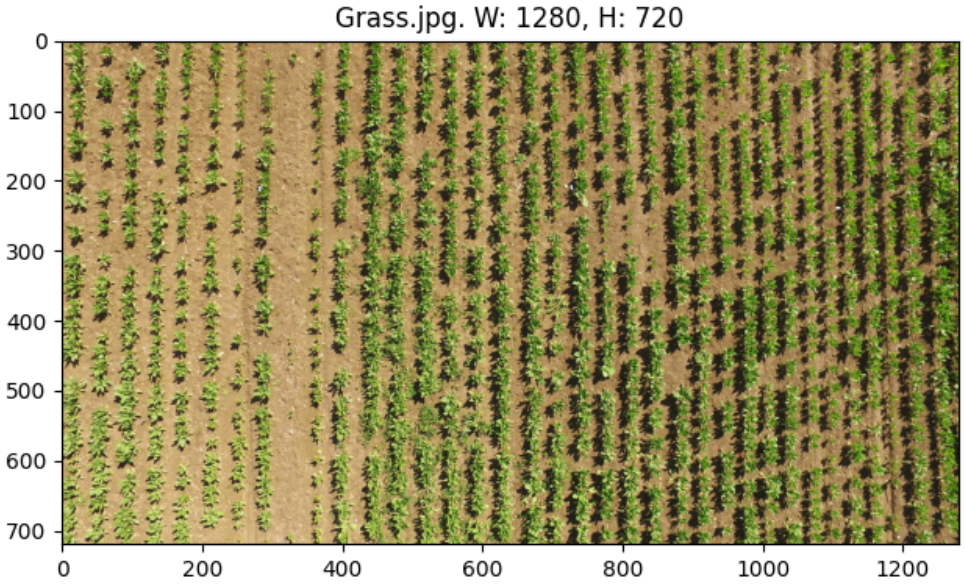
2.1: Figure 2 shows the included grass.jpg image displayed together with the image width and height, extracted using .shape.

2.2 Figure 3 shows the three individual channels of the included image, displayed using a grayscale colormap. It is not immediately clear which channel corresponds to green. However, looking at the zoomed in image in figure 4 the plants are the brightest in channel 1 (0-indexed). It is thus reasonable to assume that channel 1 holds the green values for the image. This would also correspond to the ordering *R-G-B* for the three color channels.

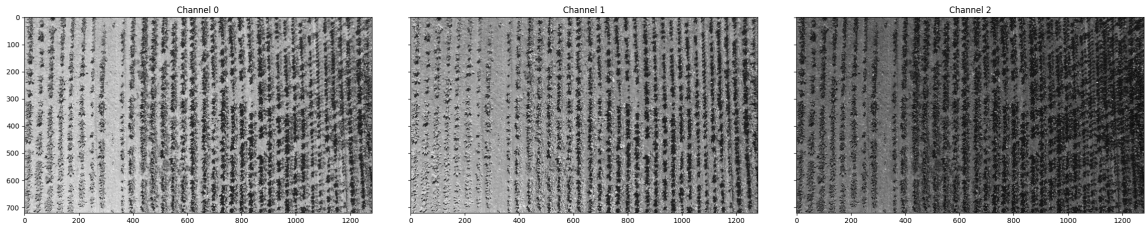
2.3 Figure 5 shows an attempt at isolating the pixels belonging to the sugar beet leaves. Adjusting the threshold higher leads to very few leaf pixels being recognized as 1's, while a lower threshold leads to the shadows of the leaves becoming part of the segmentation. Also note that this segmentation performs somewhat OK in the right half of the image, but quickly breaks down for the left part of the image.

2.4 Figure 6 shows the three individual color channels of the image after RGB-normalization. From this image it is much easier to identify that channel 1 corresponds to green, at least comparing to figure 3.

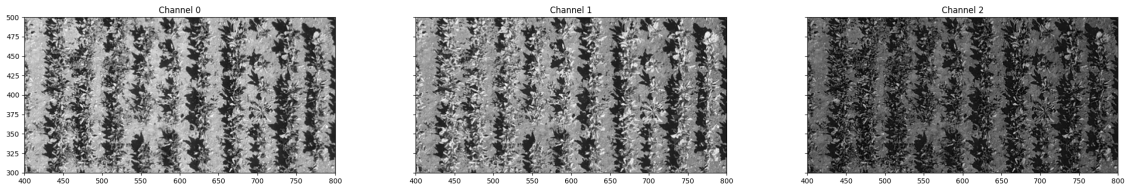
2.5 Figure 7 shows a much better segmentation of the sugar beet leaves, compared to the initial naive approach.



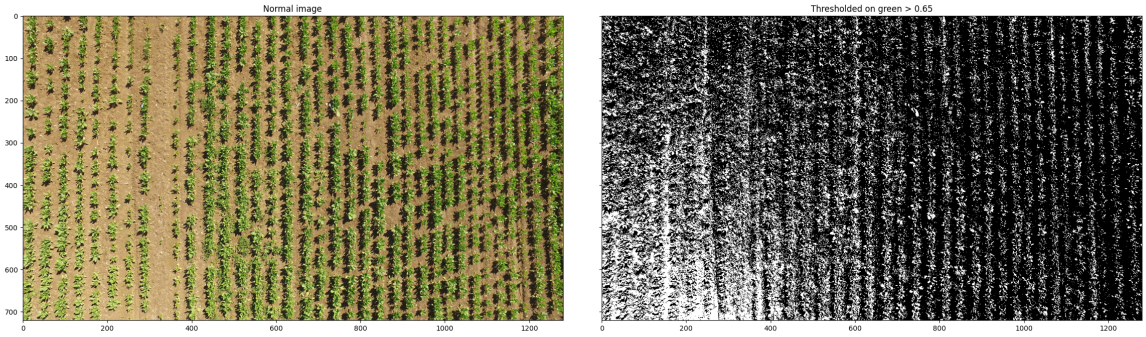
Figur 2: Task 2.1



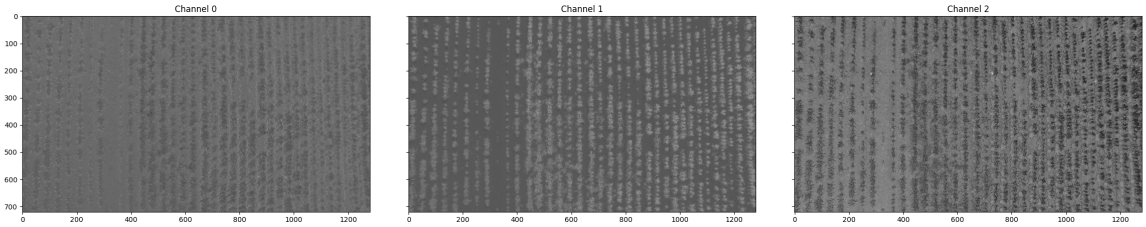
Figur 3: Task 2.2, the three individual channels of the included image.



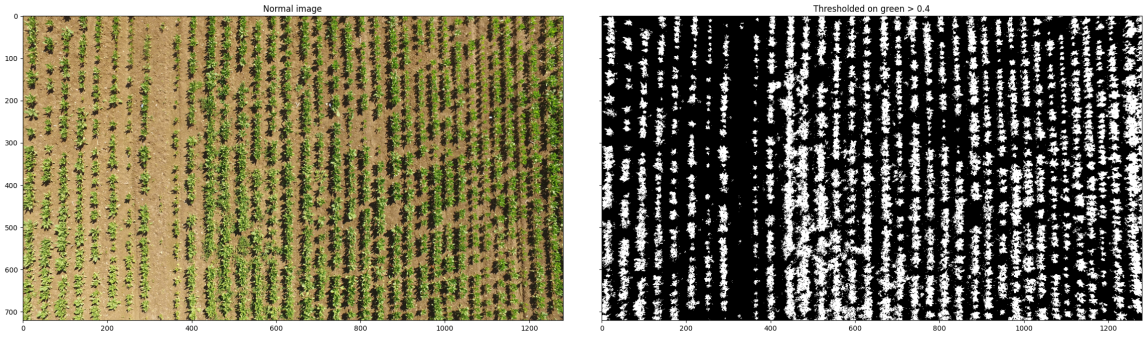
Figur 4: Task 2.2, figure 3, but zoomed in over a smaller region.



Figur 5: Task 2.3, naive thresholding on green to isolate beets.



Figur 6: Task 2.4, the three color channels after normalization.



Figur 7: Task 2.5, thresholding the RGB normalized image on green.

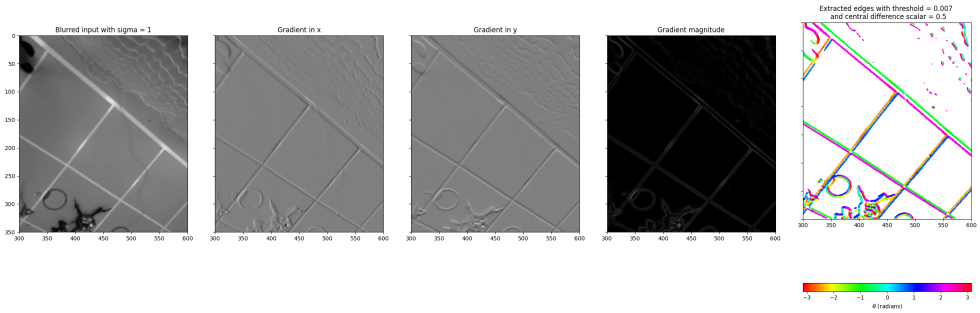
Problem 3

3.5 The code for problem **3.1** - **3.4** was implemented in order to produce the following figures. Figure 8 and 9 shows edge detection performance under different values for σ in the gaussian filter kernel when every other parameter is left untouched. Increasing σ leads to the edge detector failing to detect any edges, and seems reasonable given that a larger σ corresponds to more smoothing, i.e. attenuating edges in the image. However, figure 10 shows that by decreasing the edge detection threshold yields better performance for the higher σ. Do note that the extracted edges now appear wider, which is an artifact from the gaussian filter.

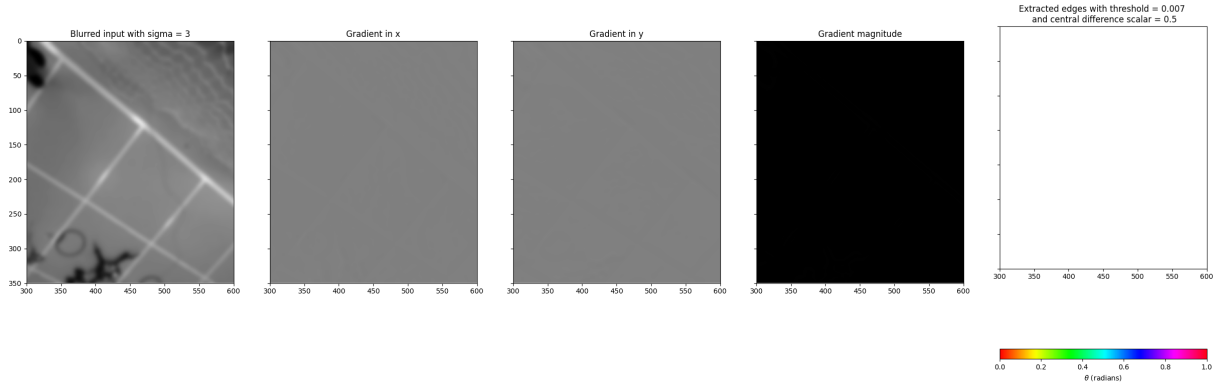
The gradients in both directions is calculated using a central difference filter of length 3, where the middle element is 0, and the first and last elements are -$k$and k respectively. As shown in figure 11, the edge detector can be tuned using k instead of the threshold value.

Lastly, figure 12 demonstrates the effect of the gaussian filter - by removing it, the gradient computation effectively differentiates noise. The effects of this can be seen by the noise that is propagated into the edge detector.

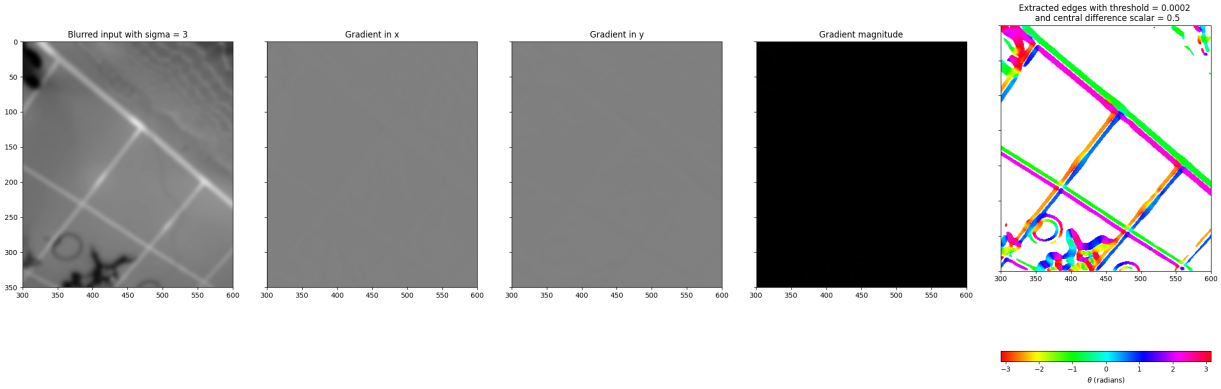
Problem 3 continued on next page...



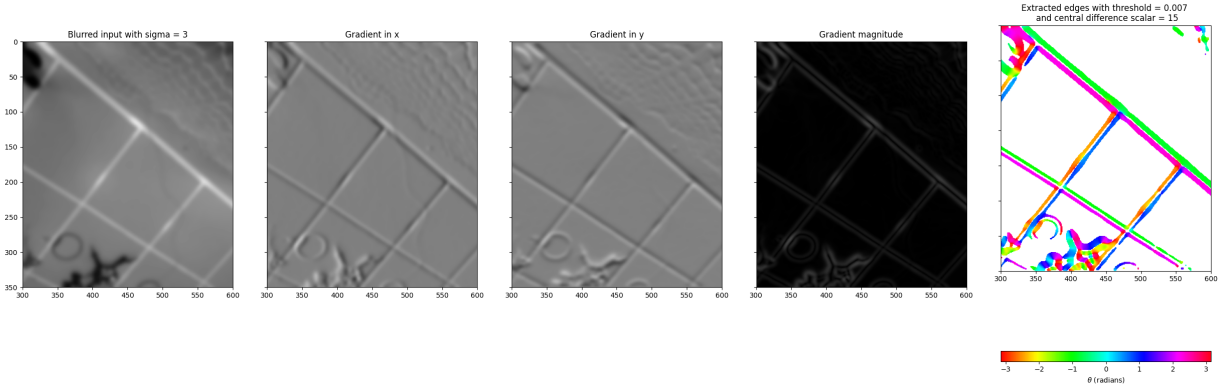
Figur 8: Task 3.5, edge detection using $\sigma = 1.0$ edge detection threshold $= 0.007$ and central difference scale $= 0.5$



Figur 9: Task 3.5, edge detection using $\sigma = 3.0$, edge detection threshold $= 0.007$ and central difference scale $= 0.5$

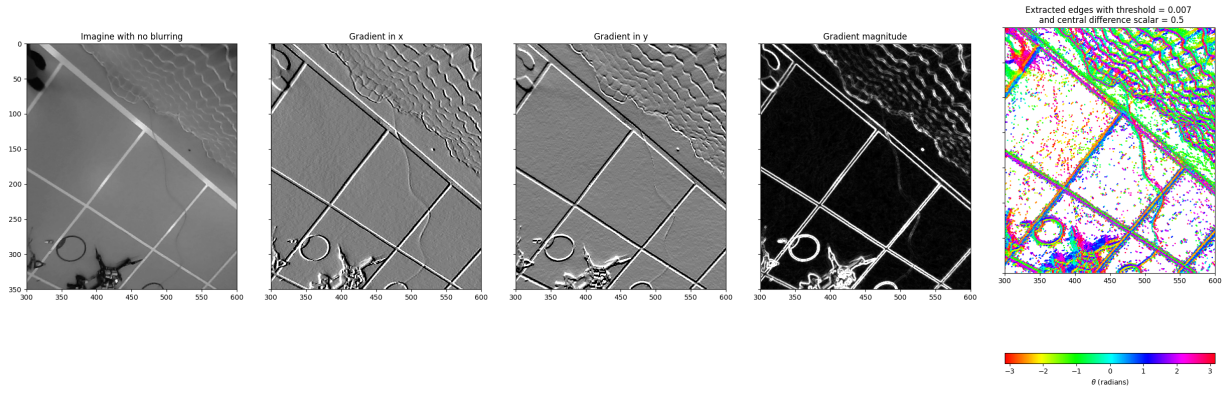


Figur 10: Task 3.5, edge detection using $\sigma = 3.0$, edge detection threshold $= 0.0002$, and central difference scale $= 0.5$



Figur 11: Task 3.5, edge detection using $\sigma = 3.0$ edge detection threshold $= 0.007$ and central difference scale $= 15$

Problem 3 continued on next page...



Figur 12: Task 3.5, edge detection using no gaussian blur, edge detection threshold = 0.007 and central difference scale = 0.5