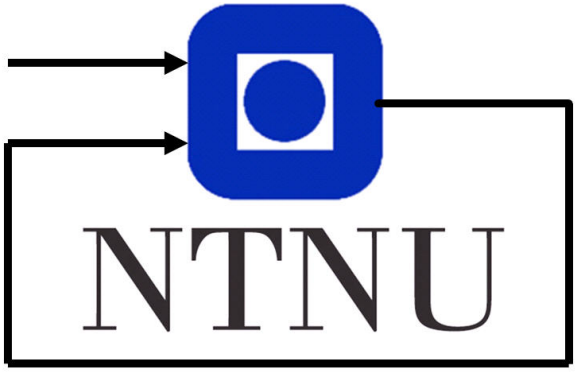
TTK4255 Robotic vision - HW1

Håkon Haugann

January 19, 2021



Department of Engineering Cybernetics

Part 1 - Theory

Task 1.1

Brightness and contrast adjustment are the only transformation among the others that are *point operators*.

Task 1.2

A convolution with the kernel

$$\begin{bmatrix} -0.5 & -1.0 & -0.5 \\ -1.0 & +7.0 & -1.0 \\ -0.5 & -1.0 & -0.5 \end{bmatrix}$$

does not alter an image with uniformly distributed values, but seem to amplify the difference of a step-edge. By these observations it seems this linear filter highlights edges by increasing contrast along the edge border.

Task 1.3

Given the standard deviation σ and a threshold ϵ, we want to find h such that

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

For the given values $\sigma = 3$ and $\epsilon = \frac{1}{256}$, we find that

$$h > \sqrt{2 \cdot 3^2 \ln\left(\frac{256}{3\sqrt{2\pi}}\right)} - 1 \approx 6.97,$$

meaning h should be at least 7. Hence, the filter kernel should be of size 15×15.

Part 2 - Basics

Task 2.1

Using *matplotlib.pyplot* we can easily import images and extract their dimensions. The resulting console output shown in Figure 1 below.

```
[4] print("Image height: {}\nImage width: {}".format
    (h,w))

    Image height: 720
    Image width: 1280
```

Figure 1: Image dimensions for *grass.jpg*

Task 2.2

Plotting the channels separated in three plots we get the results shown in Figure 2:

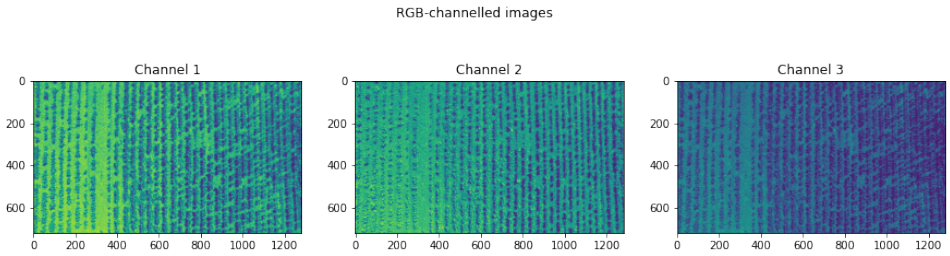


Figure 2: The RGB channels of *grass.jpg* split into three plots.

Channel 3 is ruled out immediately as it darkens in the regions of green in the original image. The clear edges between the earth and the plants in channel 1 indicate it is not the green channel. Alhough channel 2 seem to light up more than expected in the originally brown earth parts of the image, it seems to be the best fit for the green channel.

Task 2.3

Setting a threshold on the green channel for pixel values under 190 gives the results shown in Figure 3: This does not immediately strike me as success, as one would expect an image

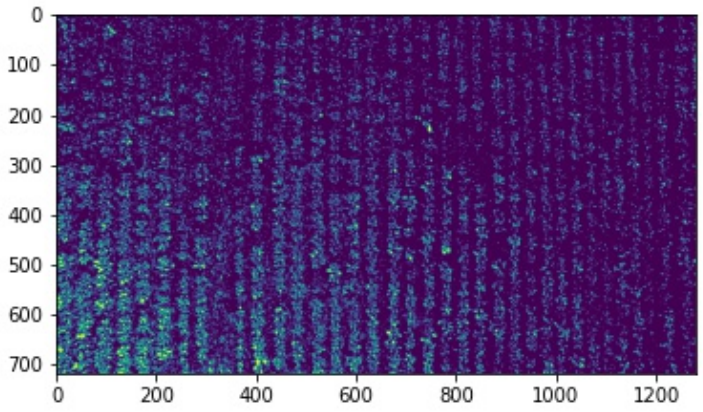


Figure 3: The green channel of *grass.jpg* thresholded for pixel values under 190.

resembling the middle image in Figure 2 of the task sheet.

Task 2.4

Normalizing each channel by dividing by the sum of the pixel values across all three channels using `grass.sum(axis=2)`, gives a better indication of the presence of color for each channel. This is shown in Figure 4:

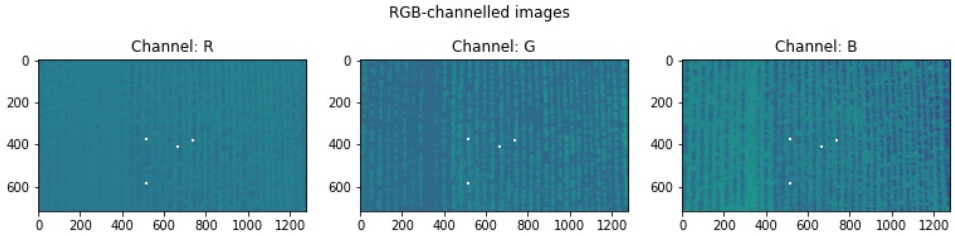


Figure 4: Each channel is normalized such that each pixel is how much of the given color is in the corresponding pixel in the original image.

Task 2.5

This seems to work much better, as seen in the results in Figure 5. The threshold was set to 0.4, which seems to be the lowest possible without including non-plant articles in the image.

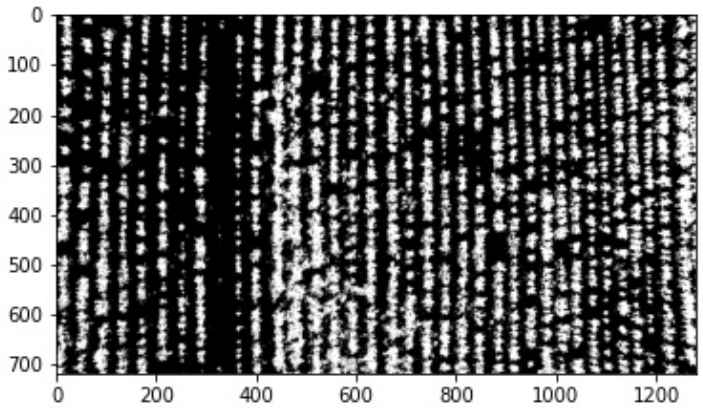


Figure 5: Binary image where pixels lights up where the green channel value is greater than 0.4.

Part 3 - Edge detection

Task 3.1 - 3.4

Implementation tasks in Python. As it is not asked for, I will not include the code in this report.

Task 3.5

Figure 6 illustrates the obtained results of this task. Tuning the blur-parameter σ and the threshold seems to make it possible to filter out possibly unwanted edges, such as the net in the top right of the image.

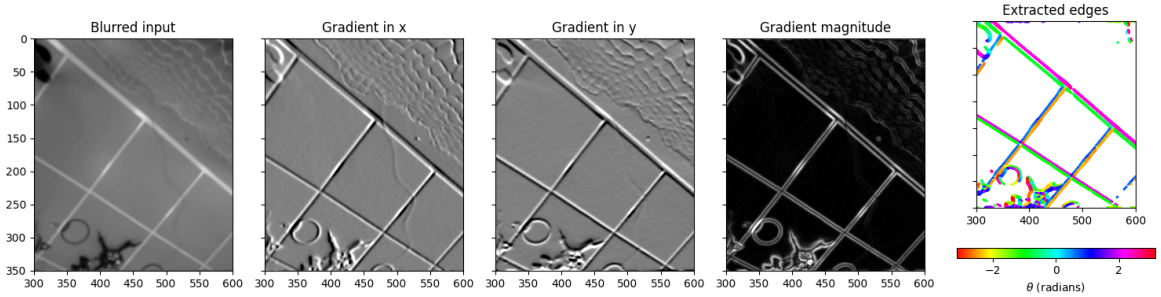


Figure 6: Showing the Gaussian blurred input image, the gradients and the detected edges. Results are shown for $\sigma = 1.5$ and thresh $= 0.04$