Image Processing, assignment 3

Tijmen van ter Beek*, Kasper Nooteboom†

November 11, 2018

1 Object class

Our first plan was to detect gaming controllers. After many tries using Gaussian filters, edge detection and thresholding, it turned out that in almost all natural situations, these objects casted a shadow, making detecting edges between the object and the surface it is placed upon nearly impossible, seeing that most gaming controllers are black, and the edges that we did find were, compared to both their direct surroundings and the rest of the image, too weak to be properly processed using thresholding, Hough transformations, moments etc. We also tried using a larger Sobel kernel when detecting edges, but this also didn't show any positive effects. This made us choose to detect another kind of object.
We chose to detect highway signs (the large blue things with city names and distances on them) because they should have a strong contrast with the background, seeing that they should be easily visible for drivers, making the project more achievable. The context table is shown below.

Criterium	**Value**
Minimum / Maximum size	Minimum: decent size sign compared to total image and at least 30×30px. Maximum: 512×512px.
Lighting variations	Most highway signs are outside.
Rotation variations	All rotations.
Occlusion	No occlusion. It should technically be possible, but practically, it isn't. Yet.
Other	Object not viewed under 'weird' angles; so close to frontal. Also, minimum noise.

2 Explanation and discussion

We started with detecting edges by first increasing the contrast in the image, then applying a Gaussian filter, followed by a Sobel kernel. This resulted in far better results than when we tried to detect gaming controllers. We decided to try to find the contours of the signs using a Hough transformation, but there were quite a few complications.
The background often contained trees, and with that, a lot of contrast, which caused the trees to also contain some of the "lines" found by the Hough transformation. We tried to solve this problem by dilating, then automatically thresholding using the Maximum Entropy method, and finally thinning using Golay Hit-or-Miss elements. However, dilating caused the lines to morph, making them less straight and less recognizable. (See Figure 1) Removing the dilation made results slightly better, so we kept it that way. (See Figure 2)
However, the problem of the amount of background edges was not solved by this. We tried to reduce this problem by applying image opening at different moments in the pipeline, or closing before thinning, but this all led to worse results. We also tried exchanging the Maximum Entropy thresholding for a local Bernsen threshold, removing pixels with zero or one neighbouring pixels, and applying the Hough transformation without thresholding. All of these options did not lead to better results.
Finally, we were able to detect rectangles in the positive results we did get, by finding two sets of parallel lines, with the pairs being at an angle of $\frac{1}{2}\pi$ compared to each other (so that they're perpendicular to each other). After trying many thresholds for a couple values, we succeeded in detecting a highway sign correctly in one image. (See Figure 3) The robustness of our solution is worthless however, but after the many tries we did to get to this point, we no longer have enough time to improve our detection. We will, however, discuss some possible improvements and current problems that we thought of.

*5961564
†5845866

Final pipeline in order: increasing contrast, applying Gaussian filter, detecting edges, applying automatic Maximum Entropy threshold, thinning using Golay elements, applying Hough transformation, finding rectangles from the found lines.
Problems in our detection:

- The parameters of the some functions (e.g. error margins in line angles) are synthetic, and are thereby very sensitive and specific to an image. Because of this, different values are needed for different images, causing the program to not be able to detect sings by itself in most cases. This could be solved by making these parameters dependent on some properties of the image, similar to the thresholding functions. For example, the minimum length of a line found by the Hough transformation could depend on the total number of edge pixels. The current version is already dependent on the maximum amount of pixels on a line can be inside the image, as suggested during the lectures.

- Background objects, but also letters on the sign, etc., cause a large amount of edges, even after thinning with Golay elements. (See Figures 4 and 5) We could not think of a trustworthy way of filtering these out, but the problem should be solved to extending the Hough transformation. We explain later in this document how we plan this possible adjustment should have a positive effect.

- All rectangular objects, or all lines perpendicular to each other, get grouped together and will be considered a rectangle; start- and end points of the lines are not taken into account. For example, lampposts in combination with a horizon or a sidewalk are currently seen as a rectangle. This should also be solved with our extended Hough transformation.

Our proposed extension for the Hough transformation:

In a normal Hough transformation, the accumulator gets simply increased by 1 (or the edge strength) per possible line. We would extend this, so that the pixels resulting in that line get saved in a list. These pixels would get sorted in the order in which the line passes through them. Of this, we take the median, because we do not expect this pixel to be a noise-pixel. From this median, we go along the line in both directions, to find the start and end point. We do this by taking the squared distance between subsequent pixels. This gives us a measure for pixel intensity. When this intensity get too high for a couple subsequent pixels ("too high" has to be empirically defined) we conclude that the line has ended, and we find the actual end of the line by finding where in the last bit the distance between pixels was above some k times the mean or the standard deviation for the first time. We also use a minimum length for the lines.

Pros:
This method ensures that lines that consist of many disjointed pixels (noise caused by the background, text on the sign, etc.) aren't considered. It also provides an estimation of a start and end point, which can be used to distinguish between rectangles and other lines.
Cons:
This method is very computationally expensive, and complex to implement. We also do not know for sure whether or not it works, and we have not yet concretized some of its aspects.

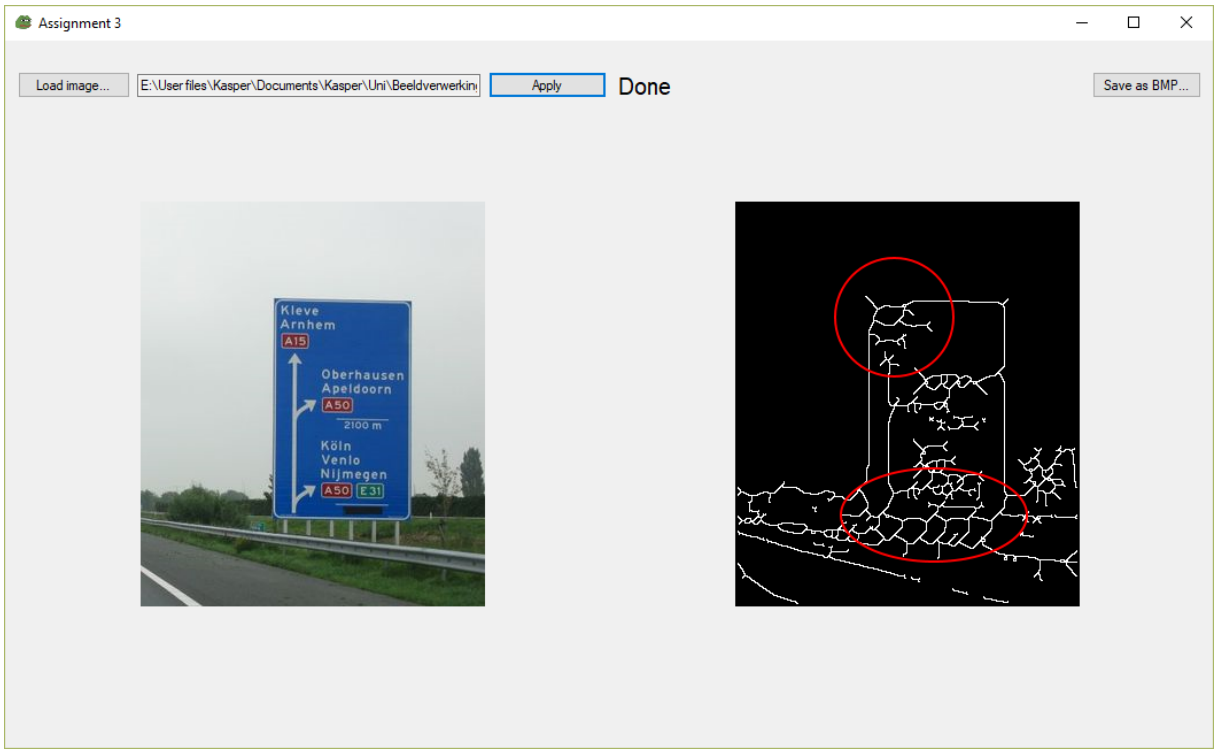

Figure 1: Sign with dilated edges

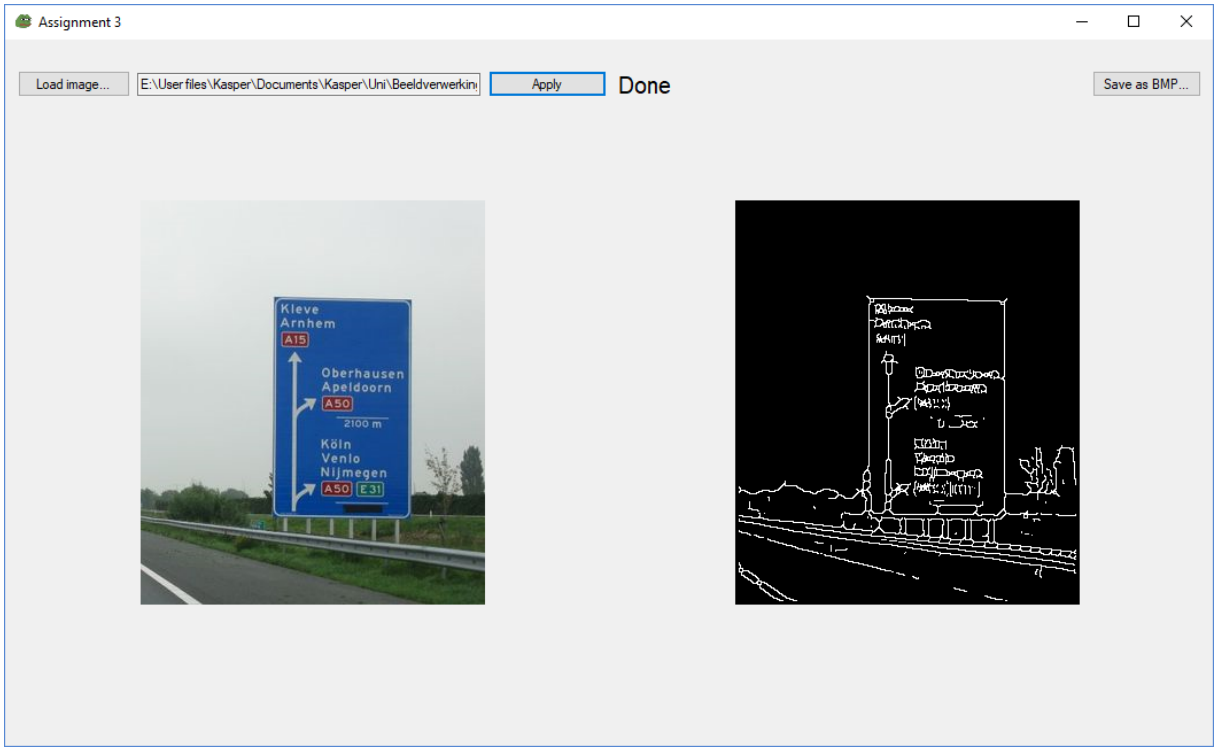

Figure 2: Sign with non-dilated edges

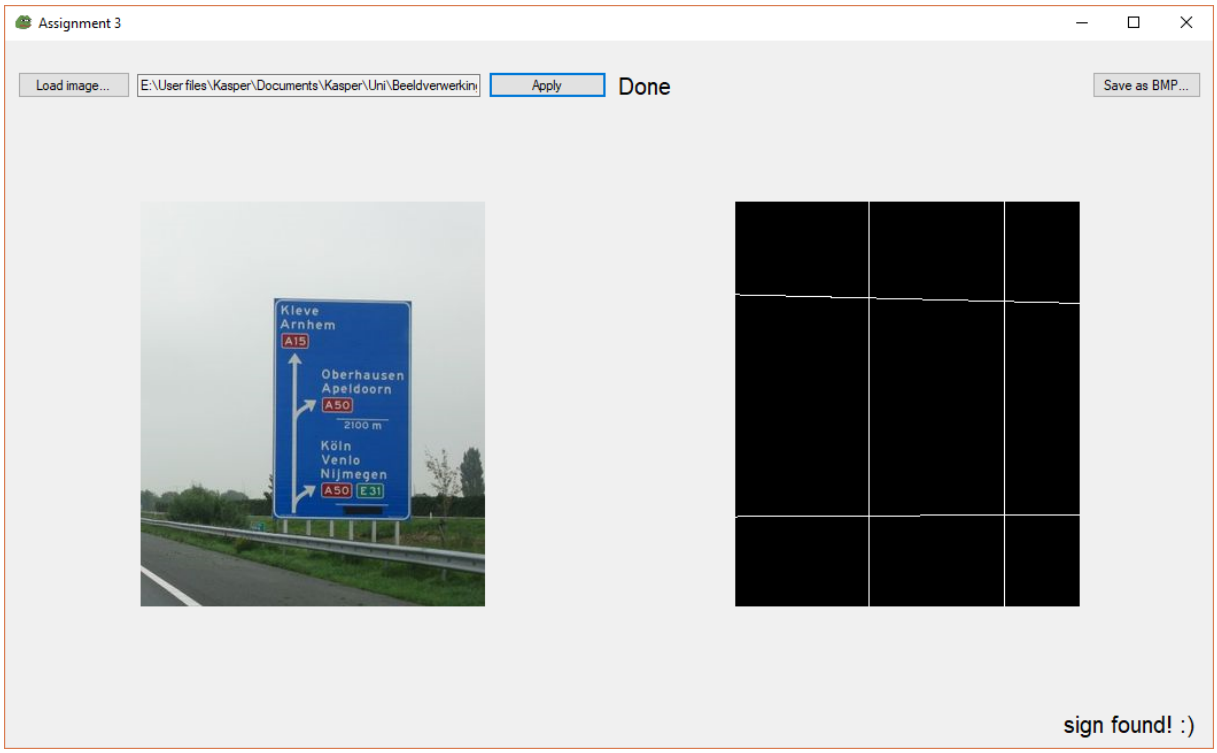

Figure 3: Highway sign correctly identified

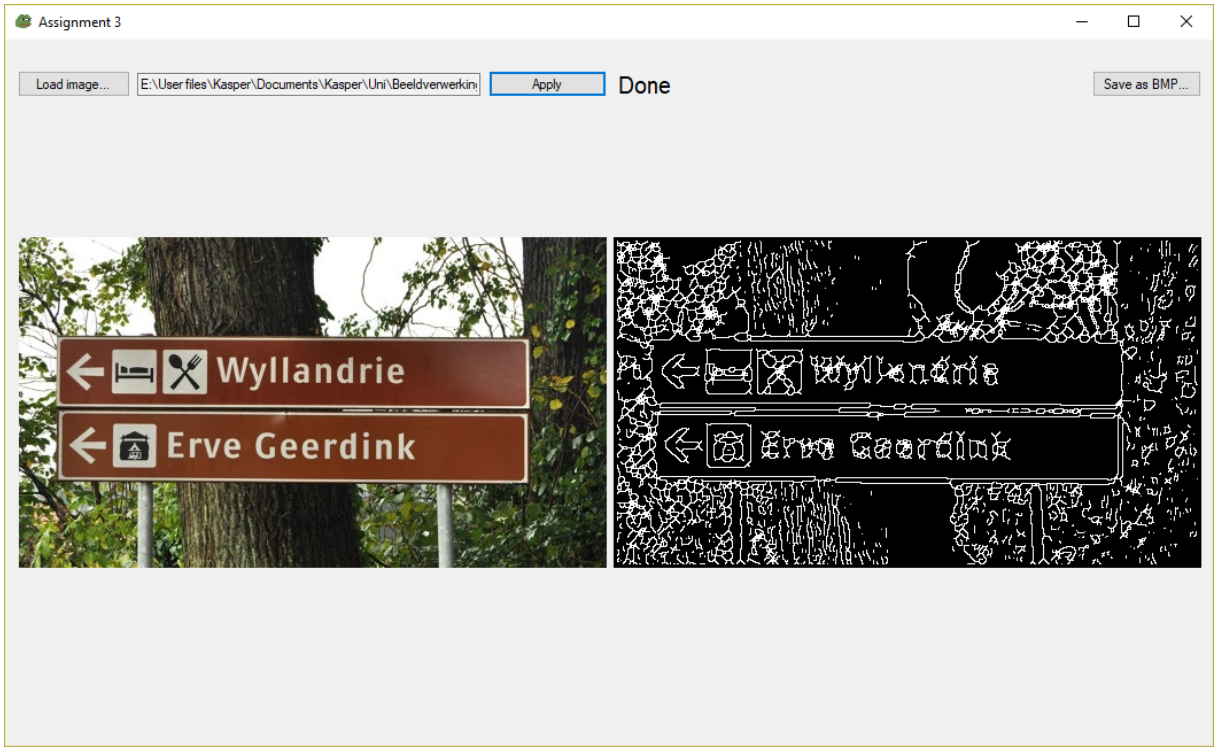

Figure 4: Edges found in the left image

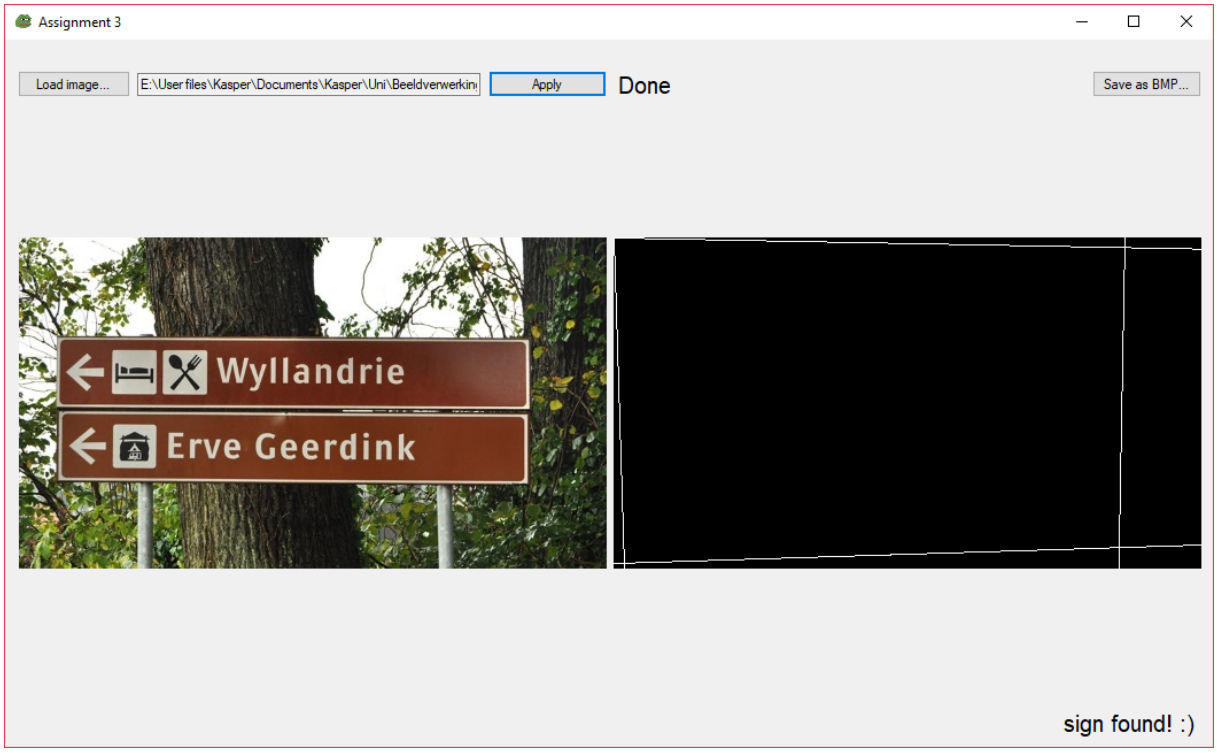

Figure 5: False positive: rectangle caused by leaves etc.