Optical Character Recognition without Neural Networks

Using MATLAB GUIDE

Nedeljko Tešanović
RA-181/2017

Task and motivation

Optical character recognition (OCR) allows computers to read text coming from an image and is an essential part of the emerging technological advancements in the fields of Computer Vision and Machine Learning. It has found applications in all sorts of fields/jobs, from security checkpoints and parking ticket system to the medical sector where it's used to read text for people with bad/no eyesight.

Neural Networks are the usual way of doing OCR, but in this example, I've looked into the possibility and effectiveness of an OCR algorithm for Licence Plate recognition without the use of Neural Networks.

The algorithm in question wasn't supposed to automatically find a licence plate on an image of a car, therefore a simple cropping solution was implemented to allow users to crop out the licence plate from bigger images (Although it can do it in certain cases, and expanding upon it would be rather easy(

Why not use just NNs like normal people?

The main motivation behind an NN-free approach is their black-box nature:
It is hard to tell how a Neural Network came to a conclusion in terms of character recognition, and what rules govern it’s behaviour.

A non-neural algorithm might have worse characteristics but understanding, debugging and modifying it will be a lot easier and a lot more manageable.

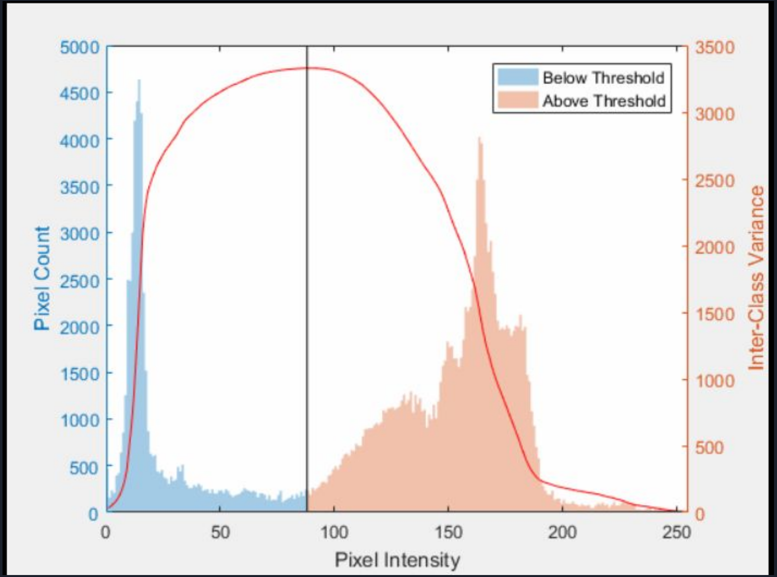
Proposed solution

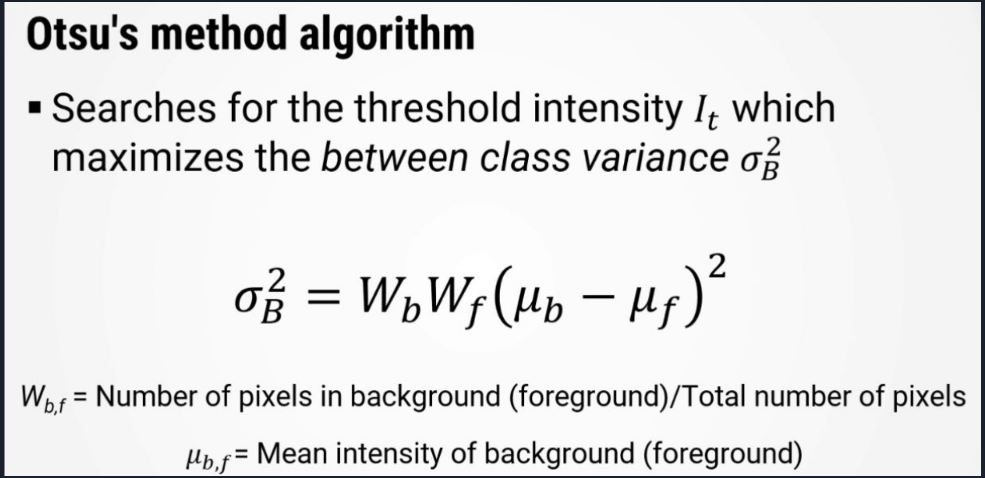
The main algorithm is based on the very basic idea of pixels holding different values depending on their brightness. With that in mind, we can comb the image and inspect sums of row/column values to find the pixels. This algorithm was tweaked to look for white characters (img[x,y] = 255)

1. Load the templates into memory
   Done only during the first reading run to save performance
2. Load the target image into memory
3. (Optional) Modify the target image
   Crop, Sharpen, Filter, Adjust contrast
4. Preprocess
   Image binarization with a auto/manual threshold
   If the threshold is greater than 0.5, it is most likely that the characters themselves are black, so the image will be inverted
5. Detect and isolate the Region of Interest (ROI)
   By comparing sums of white pixels per row with a threshold, we can isolate individual groups of white pixels. The widest one is most likely to be the licence plate text
6. Detect and isolate individual characters
   Same principle as step 5, but this time combing through columns instead. Groups that are too narrow are discarded as artefacts
7. Compare the character candidates and templates
   After equalizing their dimensions, the Euclidean distance between their pixels is calculated
8. Choose the most likely letter/number
   The template with the minimum distance is most likely our character. Since the templates are named XY.png, with X being the uppercase letter of the character that is in that template, and Y being the number of the template, we can just isolate X as the character’s value (for example A1.png, A2.png, A3.png, B1.png, C1.png - > A, A, A, B, C)
9. Concatenate the results and return the read string

Otsu's method - Calculating the binarization threshold

The threshold is determined by minimizing intra-class intensity variance, or equivalently, by maximizing inter-class variance.





Otsu's method algorithm

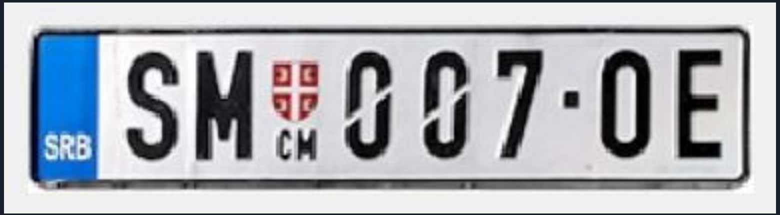
- Searches for the threshold intensity I_t which maximizes the *between class variance* σ_B^2

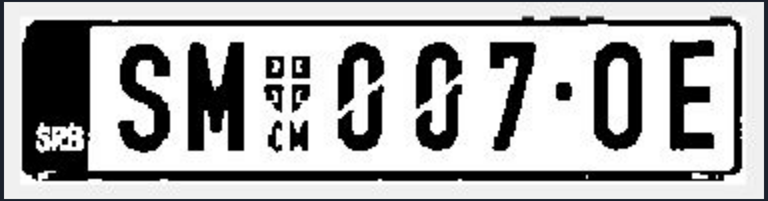
$$\sigma_B^2 = W_b W_f \left(\mu_b - \mu_f\right)^2$$

$W_{b,f}$ = Number of pixels in background (foreground)/Total number of pixels

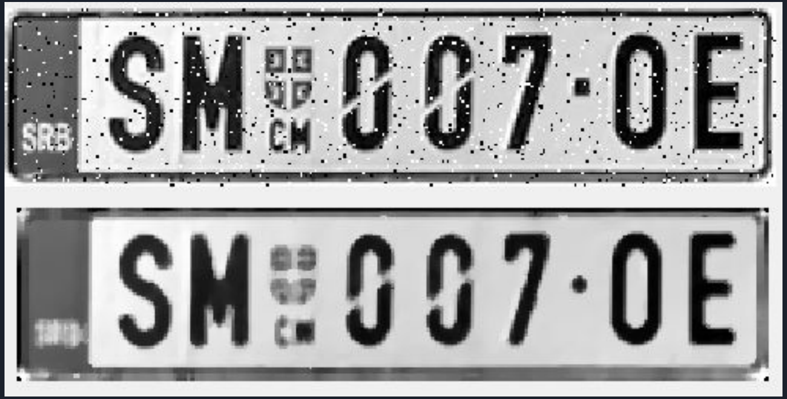
$\mu_{b,f}$ = Mean intensity of background (foreground)

Otsu’s Method Output: 0.53725





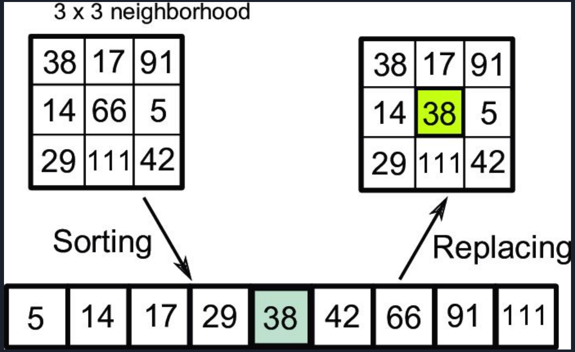
Median Filter



The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighboring entries.

Very useful for Salt and Pepper noise.

As can be seen by seen in the resulting image, the image gets filter, but also becomes blurred. That is not an issue, as additional sharpening can be easily done.



Wiener filter

The Wiener filter is a filter used to produce an estimate of a desired or target random process by linear time-invariant (LTI) filtering of an observed noisy process, assuming known stationary signal and noise spectra, and additive noise. The Wiener filter minimizes the mean square error between the estimated random process and the desired process.

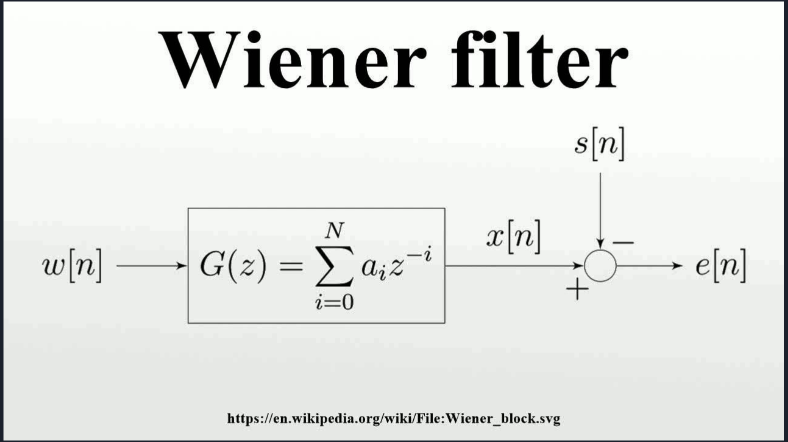

Wiener filter
$s[n]$
$w[n]$
$G(z) = \sum_{i=0}^{N} a_i z^{-i}$
$x[n]$
$-$
$+$
$e[n]$
https://en.wikipedia.org/wiki/File:Wiener_block.svg

Image with Gaussian noise

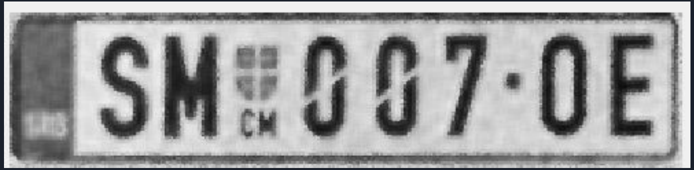
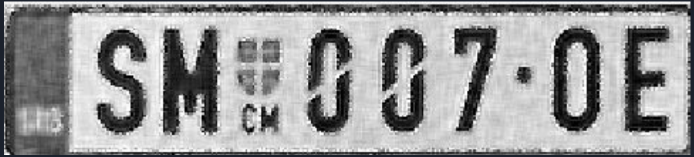


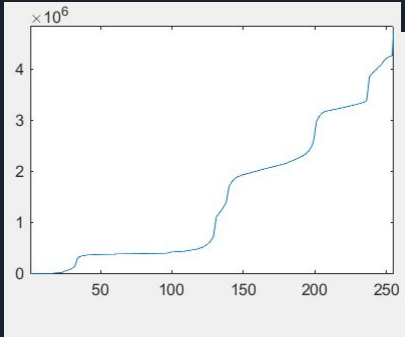
Image filtered with a 5x5 Wiener filter



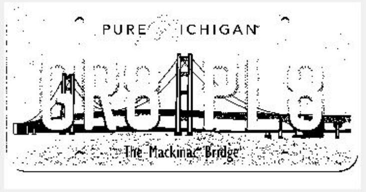
Sharpened Image

Histogram equalisation

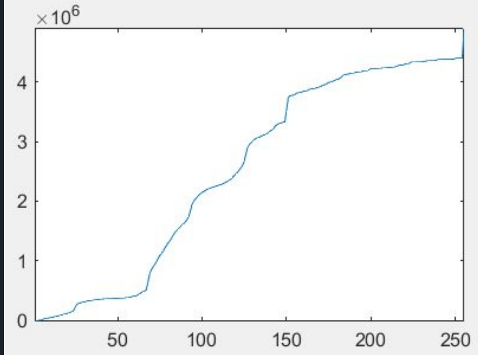
Histogram equalisation improves contrast between different shades , making the white/black text pop out more, improving image detection, as it can be observed in the pictures on the right.

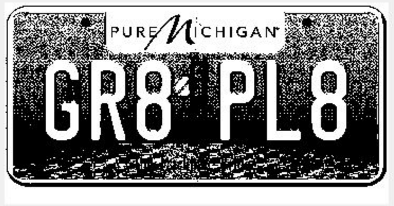


Cumulative histogram



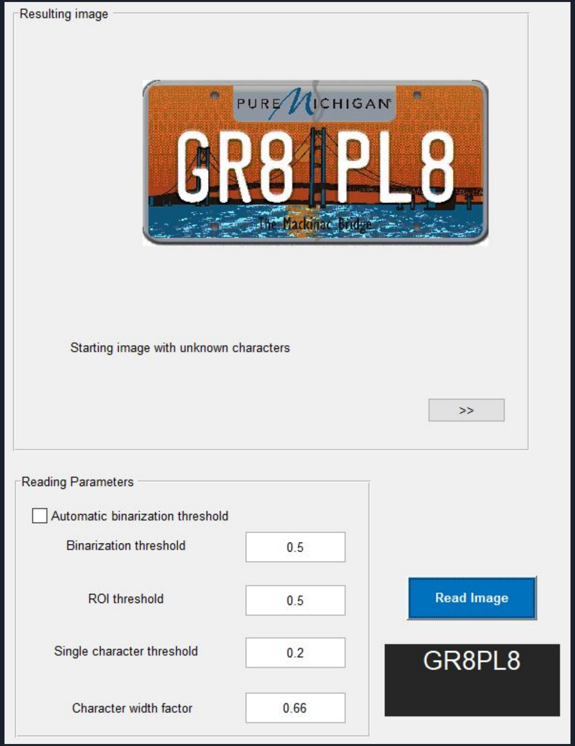
Binarisation completed with threshold: 0.49





Binarisation completed with threshold: 0.49

Reading results


Resulting image

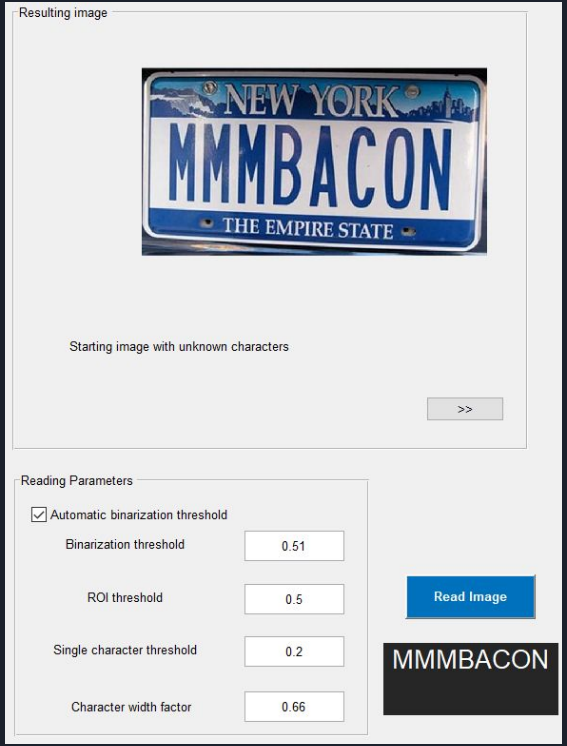
Starting image with unknown characters

Reading Parameters

☐ Automatic binarization threshold

Binarization threshold	0.5
ROI threshold	0.5
Single character threshold	0.2
Character width factor	0.66

GR8PL8


Resulting image

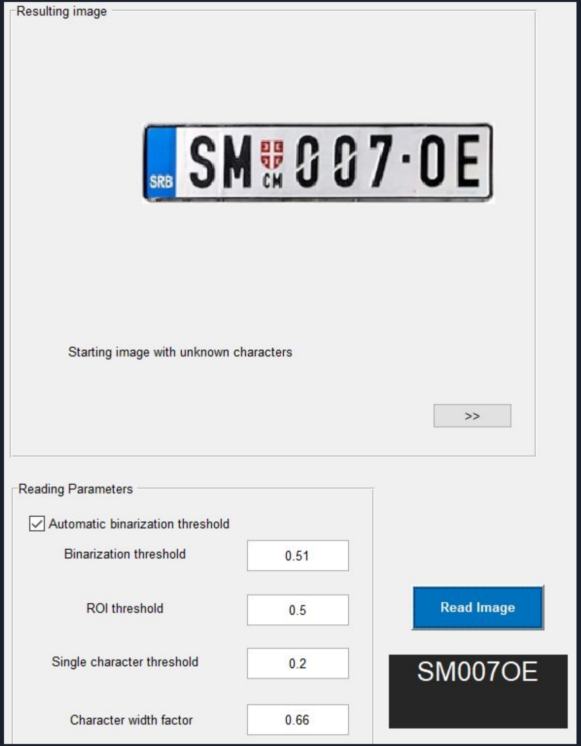
Starting image with unknown characters

Reading Parameters

☑ Automatic binarization threshold

Binarization threshold	0.51
ROI threshold	0.5
Single character threshold	0.2
Character width factor	0.66

MMMBACON


Resulting image

Starting image with unknown characters

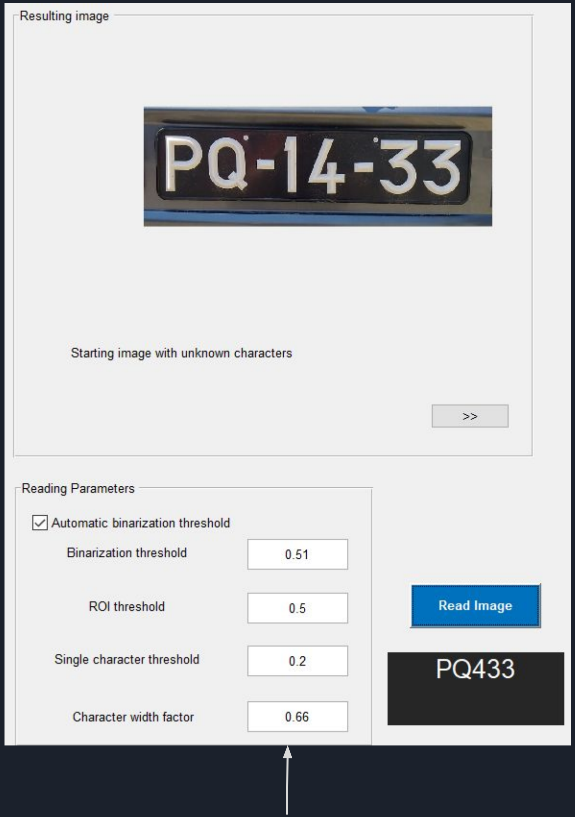
Reading Parameters

☑ Automatic binarization threshold

Binarization threshold	0.51
ROI threshold	0.5
Single character threshold	0.2
Character width factor	0.66

SM007OE

Effect of bad parameters


Resulting image

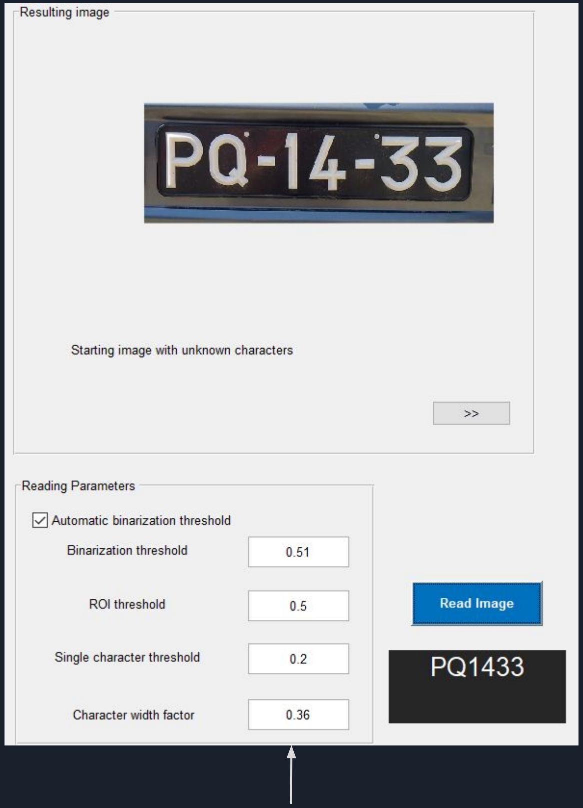
Starting image with unknown characters

Reading Parameters

[✓] Automatic binarization threshold

Binarization threshold	0.51
ROI threshold	0.5
Single character threshold	0.2
Character width factor	0.66

PQ433


Resulting image

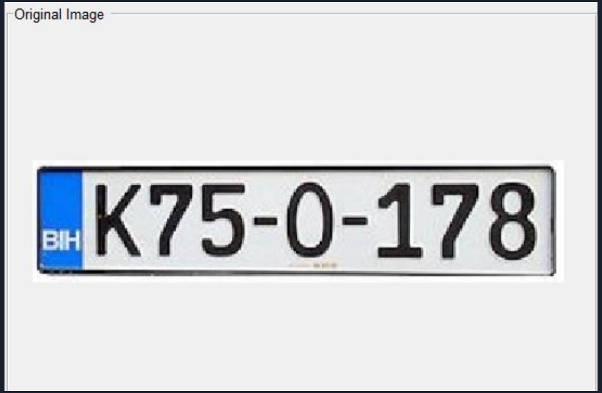
Starting image with unknown characters

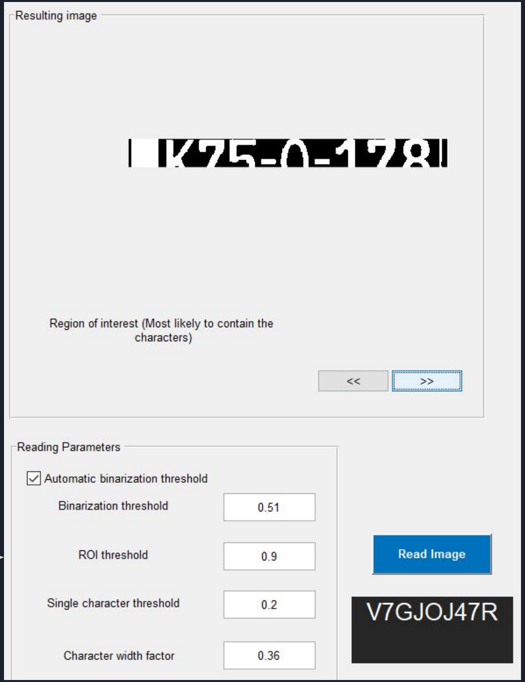
Reading Parameters

[✓] Automatic binarization threshold

Binarization threshold	0.51
ROI threshold	0.5
Single character threshold	0.2
Character width factor	0.36

PQ1433


Original Image


Resulting image

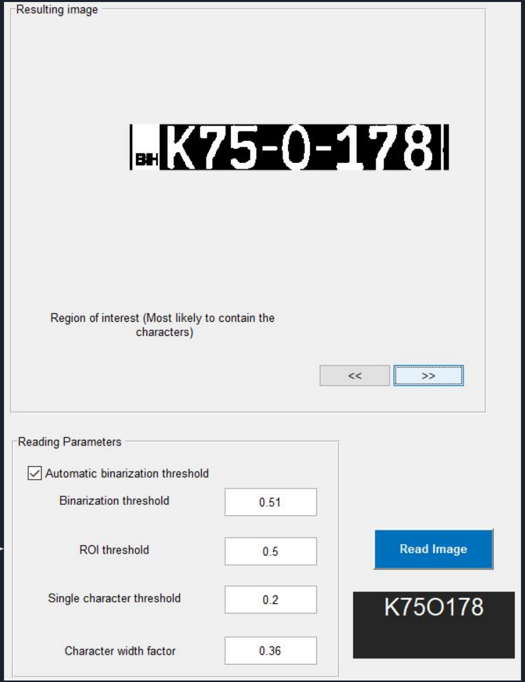
Region of interest (Most likely to contain the characters)

Reading Parameters

[✓ Automatic binarization threshold]

Binarization threshold	0.51
ROI threshold	0.9
Single character threshold	0.2
Character width factor	0.36

```
V7GJOJ47R
```


Resulting image

Region of interest (Most likely to contain the characters)

Reading Parameters

[✓ Automatic binarization threshold]

Binarization threshold	0.51
ROI threshold	0.5
Single character threshold	0.2
Character width factor	0.36

```
K75O178
```

Encountered problems and room for improvement

This approach has an issue with the unintentional binary image inversion.

The binary image will be inverted if it’s binarization threshold is greater than 0.5, which works well for most cases, but not all. This could be improved by replacing the threshold comparison with white/black pixel count comparisons.

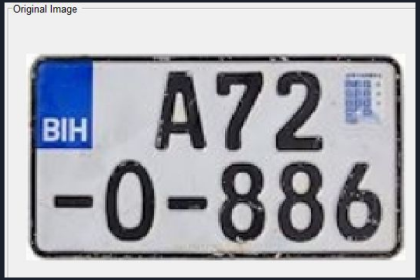
The second drawback of this approach is it’s dependency upon templates, as the time and space complexity are less than stellar. Multiple types of templates are necessary as sometimes, certain characters might differ in height, which will easily confuse the algorithm, so it would be beneficial to eliminate any unnecessary top/bottom space from individual characters before comparing them with the templates.

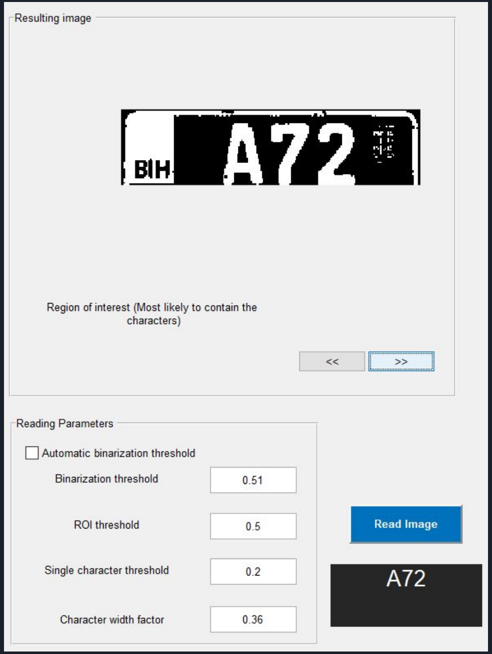
And the last, and by far biggest, drawback of this algorithm is it’s limitation to single-row text.

For example, truck/motorbike plates are usually two-lined, unlike regular single-line car plates.

This will result in either only one row being picked, like in the picture, or the whole plate being picked (if the threshold is low enough), making the individual characters unisolable.

Fixing this would require extensive changes to the algorithm, such as account for multiple-lined plates and so on, but the complexity of such an algorithm would quickly rise up to the point that Neural Networks would be a far better choice.





Region of interest (Most likely to contain the characters)

- [ ] Automatic binarization threshold

Binarization threshold	0.51
ROI threshold	0.5
Single character threshold	0.2
Character width factor	0.36

A72

Conclusion

Even though it has its flaws, this algorithm performs far better than expected, and if modified as suggested, is a solid alternative to a NN approach.