License plate recognition

Matej Havelka and Patrik Barták

Input

The plate recognition system expects a certain type of input in order to function well. The image must be cropped to only include the plate, and the characters must not be significantly rotated or skewed. That means localization must perform a custom-coded perspective transformation. It must also be of a sufficient resolution in order to give the recognition system enough detail. All of this is provided by our localization function, and an example is shown below.

Preprocessing

A number of preprocessing steps must be taken to remove unnecessary information from the plate. First, we convert the image to grayscale and perform contrast stretching in order to highlight more clearly the text from the background. The image is then binarized using a threshold manually tuned on the training set (described in more detail later).

These steps already give quite a good result, but the plate frame often remains and must be removed in order to not interfere with segmentation. At this stage, virtually all noise, including the frame, is connected to the edge of the image. For this reason, we will now describe our custom-coded denoising flooding operation.

The figure below shows a plate after pre-processing.

Stage 1 – Flooding

The flooding operation is done by creating an array of seed points. We chose 8 points located in the corners of the image and the midpoints of the edges of the image. If these points have value 0, nothing happens. If these points have value 1, they are set to 0 and this procedure repeats for all neighbors.

This removes any shapes connected to the seed points. Thanks to this procedure, we are able to isolate the characters against a black background. An example is shown below:

Stage 2 – Segmentation

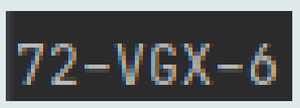
Once the image is prepared, our algorithm performs segmentation. This is done by summing all pixels along a vertical axis and then using an algorithm to split the image into individual characters where the projected sum is lower than a certain threshold. This threshold was tuned manually using the training set. Each individual character is then resized to a standardized 60 * 85 pixel size to prepare for template matching. The image below shows this.

Stage 3 – Template matching

Once individual characters have been segmented, we perform recognition using template matching. We bitwise XOR each character with the set of all possible character templates and score the match based on the fraction of pixels that remain. This is our distance metric. In order to improve the accuracy of recognition even when characters are not perfectly formed, we extended the set of templates with sheared ones. This is shown on the right.

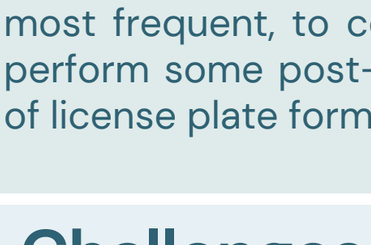
Results

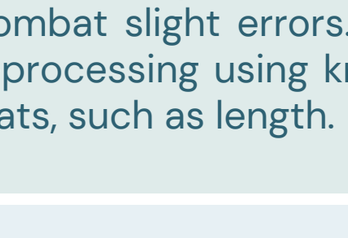
Now that we have explained how we get to the results, lets take a look at some results. For some plates from the training set this works very nicely. Our algorithm seems to properly work on all plates from Category I and II (both training and test set). At the moment we use a simple majority vote to determine the true license plate, that is we output the license plate that was the most frequent, to combat slight errors. We also perform some post-processing using knowledge of license plate formats, such as length.



Challenges

During the development of this algorithm we have faced many challenges. Errors compound, and any small error that might happen prior to recognition might grow to a much more significant error. One of the challenges we faced were rotations, when a character we receive is slightly tilted, the pixel overlap decreases very quickly, especially when it comes to thin characters like L, I or 1. Another challenge we had to overcome are slight variations in the output, some frames output incorrect results, and we needed to ensure that those get squished by the amount of correctly specified license plates. Lastly, we had to ensure the system had enough detail to distinguish letters such as S and 5, and B and 8, which was challenging.





You can see that as the "X" found in the image is slightly tilted, when compared to a normal X, it does not provide a perfect match.

Numerical evaluation

To properly evaluate the data we needed to properly split the input videos into a training and testing sets. We have taken 60% of the given videos from a category as training data, on with which we worked to make our algorithms better and more robust. Then with the remaining 40% we tested the data. That is we did not use those license plates during the development, thus we could see how effective the algorithm is on newly found data, and how much is our algorithm taylored for our training set. We used the 60-40 split because we needed enough training data to cover as many edge cases as possible, but also not too much data such that we would overfit on that data. We use accuracy as a metric and it's compared manually with hand-annotated data. When we run our algorithm we manage to correctly identify 100% license plates from the test set from Category I and II. In Category III it is bounded by the Localization, more precisely 50%, and In Category IV we only manage to get 25%, however that is because there are 4 videos in the test set, which so there is only 1 correctly identified license plate. As mentioned in previous sections, parameters such as thresholds and kernels were manually tuned against the training set by observing both the effect they have on the plate image, and the effect they have on our numerical evaluation.