Locate My Plate
A License Plate Localisation System

F. Huizinga [0418862], T. Kostelijk [0418889]
{folkerthuizinga,mailtjerk}@gmail.com

June 24, 2009

1 Introduction

This report describes the implementation of a robust, real-time License Plate Localisation system (LPL) [1, 2]. We analyze which characteristic features are important for license plate localisation. Using supervised learning, the system generates a cascading classifier which consists of layers that each hold one strong classifier. A strong classifier is a linear function of several weak classifiers obtained by boosting. Each weak classifier is a feature which describes characteristics of a license plate. Section 2 describes the data used for our experiments. Section 3 describes the features and how they are generated. Next the training and classification of weak-, strong- and cascading classifiers are both explained. Finally the results are shown and we come to the conclusions.

2 Dataset

The dataset used is obtained from [3]. It contains 246 car images with a resolution of approximately 640×480. The images are annotated on location and size of the license plate and were rescaled by 50%. The dataset is divided in a train-, test- and validation set with 159, 40 and 47 images respectively.

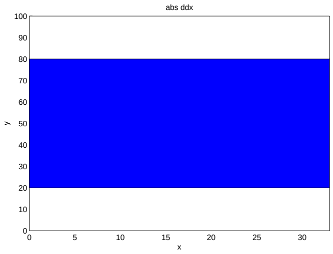
3 Features

The core of the LPL system consists of features as described by [1, 2, 4]. Features are image filters applied on a certain type of image (e.g. an x-derivative) see Figure 1. Formally a feature f is a tuple $\langle i, B, o \rangle$ defined as:

- i, an index corresponding to the image type.
- B, the set of blocks where each block $b \in B$ contains a sign $b_s \in \{-1, 1\}$ indicating subtraction or addition of that block and positions $b_{pa}, b_{pb} \in [0, 1]$, determine the relative block position within the feature.
- o, the orientation of the feature blocks: horizontal or vertical.

The feature value $f : x \mapsto \mathbb{R}$ of an image x is shown in Algorithm 1.

(a) The original image.



(b) The feature, internally represented as binary 01110.

(c) The feature applied.

Figure 1: A horizontal, second order x-derivative feature with binary code 01110, consisting of three blocks applied to an image.

Algorithm 1 featureValue(f, x, w, h): Returns the image $V = f(x)$

Require: The feature $f = \langle i, B, o \rangle$, the image x, the width w and height h of the feature.

Initialize V as an image with dimensions $D(x) - [w, h]$ consisting of zeros.
Let I be the i^{th} image type of x.
if o is horizontal **then**
  $I \leftarrow I^{\top}$
  Swap w and h
end if
Let B' be the set of blocks with rescaled coordinates using w, h.
for all $b \in B'$ **do**
  Let X be the result of applying b to I while respecting b_{pa}, b_{pb} alignment.
  $V \leftarrow V + b_s \cdot X$
end for
return V

3.1 Image Types

For this report, the following image types were used.

- 1st order derivative in both x and y directions.
- 2nd order derivative in both x and y directions.

Before applying the feature, the above image types are passed through an absolute filter. By calculating an integral image per image type, the featureblocks can be calculated very efficiently as each block calculation requires four array access instructions [5].

3.2 Generation

By representing a feature as a binary string, the set S of possible features can be easily calculated:

$$S = \{b(x, n) | \forall x \in \{1, \ldots, (2^n - 2)/2\}\},$$

where $b(x, n)$ represents x as a binary string of length n as the number of segments. Note that binary strings $0_1, \ldots, 0_n$, $1_1, \ldots, 1_n$ and the inverse binary strings are ignored. Each element in the binary string $s \in S$ represents the position and the sign of a feature segment. Adjacent segments that share the same sign are merged together and are called a feature block.

This set of features is duplicated for each image type. The horizontal features are generated using the transpose of the vertical features and by swapping the dimensions.
The height and width of the feature are normalised, i.e. set to 1 so it can be generaly rescaled to different license plate sizes. An assumption of the size of the license plate is made and the normalised feature is rescaled to this. For this project only one scale is used, more on this in future work.

4 Training

The overall cascading classifier consists of three types of training. The first type is the training of the weak classifiers using features. The second type is a linear combination of one or more weak classifiers into a strong classifier using a boosting algorithm. The third type is a cascading classifier with a strong classifier on each layer.

4.1 Weak Classifier

A weak classifier consists of a feature, a threshold $t \in \mathbb{R}$ and an operator $\circ \in \{<, >\}$ which separates positive and negative samples according to the trainings set. After training, the weak classifier C constructs a binary image $B = t \circ f(x)$, where x is the image and f the function that returns the value of the feature as described in Section 3. The locations of the ones in B correspond to the location of possible license plates.

4.2 Strong Classifier

A strong classifier is constructed according to the boosting algorithm described by [5]. There is no time and space to elaborate on the techniques used in this paper, but it is the core of the project. Understanding thinks one a more deep level would consider reading [5].

In short the algorithm selects the next 'best' features with their respective weight, α, by re-weighting the positive and negative samples after greedy selection of the weak classifier. Classification is performed as follows:

$$C(x) = \begin{cases} 1 & \sum_{i=1}^{N} \alpha_i \big(t_i \circ_i f_i(x)\big) \geq \tau \sum_{i=1}^{N} \alpha_i \\ 0 & \text{otherwise} \end{cases}$$

where N is the number of weak classifiers as selected by the boosting algorithm and $\tau \in [0, 1]$ a threshold which allows for changing the false positive- and detection rate.

4.3 Cascading Classifier

The cascading classifier is the final classifier. This classifier is trained as described in [5]. By specifying a false positive rate per layer, a detection rate per layer and a false positive rate goal, the algorithm constructs a cascade of strong classifiers using the training- and validation set. Algorithm 2 shows the classification of an image using a trained cascading classifier. The strong classifier $c_s \in C$ classifies according to Section 4.2. This results in a binary image B' which is logically 'anded' with the previous binary image B resulting in less false positives after each iteration.

5 Results

For our experiments we trained the cascading classifier using $fp = 0.99, d = 0.999, fp_{min} = 0.001$ for the false positive rate, detection rate and minimal false positive rate respectively. The final cascading classifier contains five layers with $1, 4, 14, 7, 10$ features respectively. An example of the

Algorithm 2 cascadingClassify(C, x, w, h): Returns the binary image B of x

Require: C the cascading classifier, x the image, w, h the dimensions of the features

Initialize B as an image with dimensions $D(x) - [w, h]$ consisting of ones.
for all $c_s \in C$ **do**
    $B' \leftarrow c_s(x|B)$
    $B \leftarrow B \wedge B'$
end for
return B

cascading classifier is displayed in Figure 5. To illustrate how a layer is constructed, the strong classifier of layer 2 is explained in Figure 2
The confusion matrix on the test set can be found in Table 1.

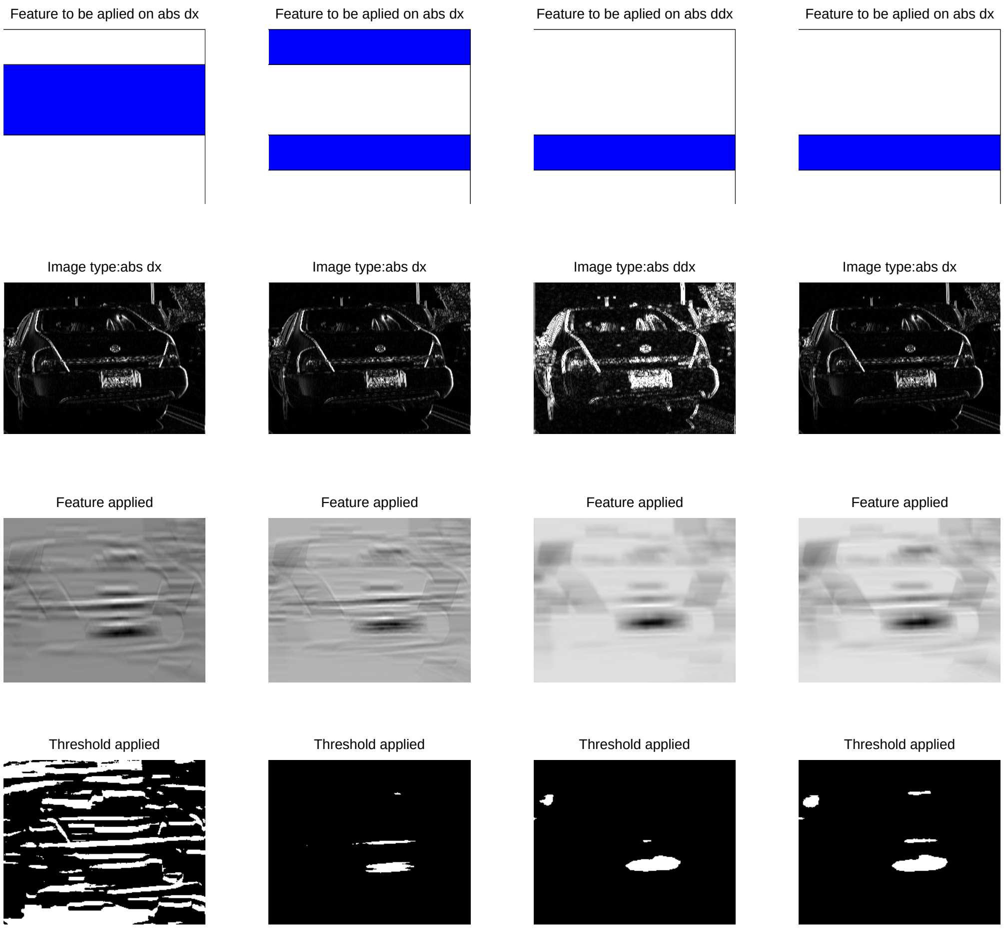


Figure 2: Example of the strongclassifier in layer 2 with 4 features.

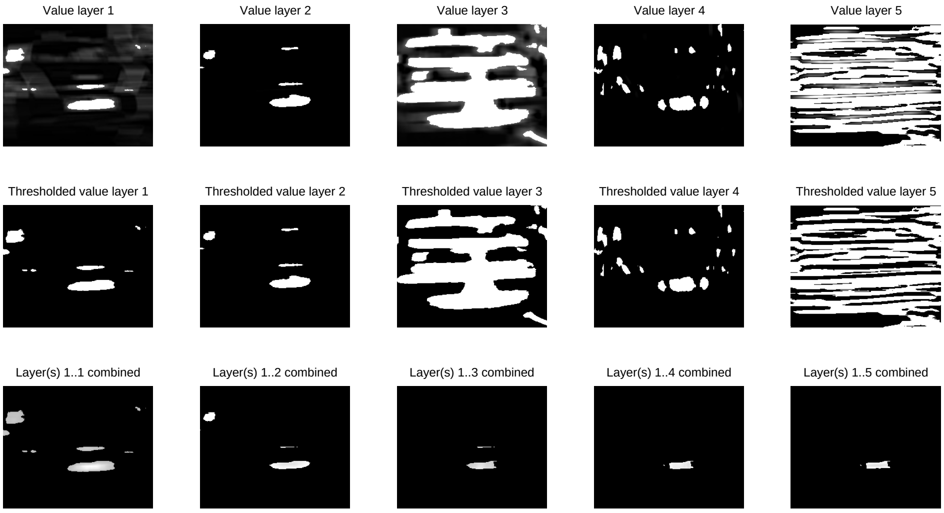


Figure 3: Example of the resulting cascading classifier with 5 layers.

37	166186
3	2399245

Table 1: The confusion matrix of the test set.

An overall detection rate of 0.925 and false positive rate of 0.0648 was achieved. Figure 4 shows the averaged false positive rate per layer in the cascading classifier. Note that we experimented with just four image types as described in Section 3.1, using more advanced image types as described by [4] would decrease the false positive rate even further. A final result of the applied cascader can be found in Figure 5.

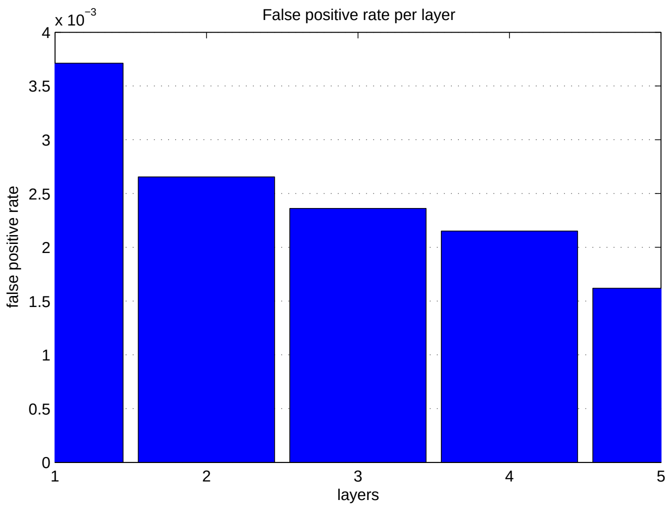


Figure 4: The false positive rate per layer, averaged over the test set.

(a) The original image

(b) The cascader applied, every white pixel is the upper left prediction of the license plate

Figure 5: An original image of a car and the resulting binary image after cascading.

6 Conclusions

Given just four types of images, the cascading classifier performs very well. With a detection rate of 0.925 and false positive rate of 0.0625 we can conclude that License Plate Localisation can be effectively achieved using this method. Even when no more sophisticated image types are used, the cascader could be used by a License Plate Recognition system as the remaining false positives can be filtered out by optical character recognition and the characteristics of a license plate.

7 Future work

7.1 Image types and layer optimalization

Because of lack of time and computational resources the features where only applied to four image types. The results would increase if more sophisticated image types where used. Some images which would TODO PRESTEREN good are

- The variance in x and y direction
- Images that display color variance explicit, for example blue-yellow for dutch license plates
- TODO jij had iets bedacht met gaussians ??, weet je er verder nog meer??

The layers in front of the cascade needs to evaluate lots of locations in the image. It would be efficient to use computational cheap image types, like the derivative images, in the first layers of the cascade. The image types that are more computational expensive could be used in layers that are near the end.

7.2 Multiple scales

An assumption of the license plate dimensions is made. In a real world system the license plate size could be anything. A solution for this would be the multiscale approach. Use different scales of the license plate and start with the biggest. Next compare the values to the learned thresholds and optionally look for a smaller scale.

7.3 What's next

Our program comes with an image with multiple license plate location predictions. This needs to be transfered to the number of license plates present in the image, in our examples 1. An idea to

do this is to fit a rectangle to the found locations and use the upper left pixel.
When the exact location is found the license plate needs to be transfered, this could be done using existing techniques like Optical Character Recognition.

7.4 Future research questions

What todo with an image with multiple license plates? This would conflict with the multiscale approach, e.g. if 3 license plates are found the total value will be so high that a wrong scale could be picked.

References

[1] L. Dlagnekov, "Video-based car surveillance: License plate, make, and model recognition," Master's thesis, University of California, San Diego, 2005.

[2] H. Zhang, W. Jia, X. He, and Q. Wu, "2006, learning-based license plate detection using global and local features," *Proceedings International Conference on Pattern Recognition, page to appear*, 2006.

[3] L. Dlagnekov and S. Belongie, "Ucsd/calit2 car license plate, make and model database," Tech. Rep., 2005.

[4] X. Chen and A. L. Yuille, "Detecting and reading text in natural scenes," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2, 2004, pp. II–366–II–373 Vol.2. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2004.1315187

[5] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000013087.49260.fb