License Plate Detection Using AdaBoost

Louka Dlagnekov
Department of Computer Science & Engineering
UC San Diego
La Jolla, CA 92093-0114

Abstract

License Plate Recognition (LPR) is a fairly well explored problem with many successful solutions. Though most of these solutions are reasonably fast, there can be increased benefits to making them even faster, such as multiple recognition stages in video frames in a real-time video stream for improved accuracy. The goal of this project is to evaluate how well object detection methods used in text extraction[1] and face detection [2] apply to the problem of LPR. A strong classifier is trained by the AdaBoost algorithm and is used to classify parts of an image within a search window as either license plate or non-license plate.

1. Introduction

In any object recognition system, there are two major problems that need to be solved – that of detecting the object in a scene and that of recognizing it. This project will mainly focus on the detection mechanism and rely on third-party OCR software for recognition. Most LPR systems employ detection methods such as corner template matching [3] and Hough transforms [4] [5] combined with various histogram-based methods. Viola and Jones have proposed very efficient object detection methods [2] that have been applied successfully to text extraction [1]. It is reasonable to presume that these methods will also apply well to the problem of detecting license plates. This was, indeed, found to be the case.

The methods used involve training a strong classifier using the AdaBoost algorithm. Over several rounds, AdaBoost selects the best performing weak classifier from a set of weak classifiers, each acting on a single feature, and, once trained, combines their respective votes in a weighted manner. This strong classifier is then applied to sub-regions of an image being scanned for likely license plate locations.

An optimization introduced by Viola and Jones involves a cascade of strong classifiers, each with specifically designed false-positive and false-negative rates, which greatly speeds up the scanning process, as not all classifiers need be evaluated to rule out most non-license plate sub-regions. However, due to time constraints, a cascaded classifier was not implemented nor was the actual OCR process on the text on the license plates.

After training, 100 weak classifiers were selected, and since no cascade classifier was implemented, all 100 classifiers were evaluated at every sub-region during the scanning process. This yielded a detection rate of 95.6% with a false positive rate of 5.7%. The 5.7% false positive rate was reduced greatly by clustering matched regions into groups, and ruling out groups with a low number of members. Further filters were applied based on color content, as most false positives included vegetation, which contains a high green color component.

The motivation for this project was to create an elaborate vehicle recognition system, which is able to identify the make and model of a vehicle, as well as its license plate. An algorithm to detect the location of the license plate is an important first step. In parts of the detection process this context plays a large role.

2. Datasets

Training and test data in this project consist of manually extracted license plate images and selected frames from video captured from a digital video camera mounted on a street lamp pole overlooking a stop sign. Figure 1 shows a typical frame captured from the camera. This camera, along with 15 others, were set up in the Regents parking lots of UCSD as part of the RESCUE-ITR (Information Technology Research) program by the UCSD Police Department. The video stream has a resolution of 640 $\times$ 480 and sampling is done at 10 frames per second.

Due to the fairly low volume of cars passing through this intersection (several thousand in an 8 hour recorded period), some sort of automation was necessary to at least scan through the video stream to find frames containing cars. An application was written for this purpose, which searches frames (in a crude, but effective method of red color component thresholding, which catches cars' taillights) for cars and facilitates the process of extracting license plates by providing an interface for hand-clicking on points.

Using this process, over 1,500 training examples were extracted as shown on Figure 2(a). In the figure, time flows

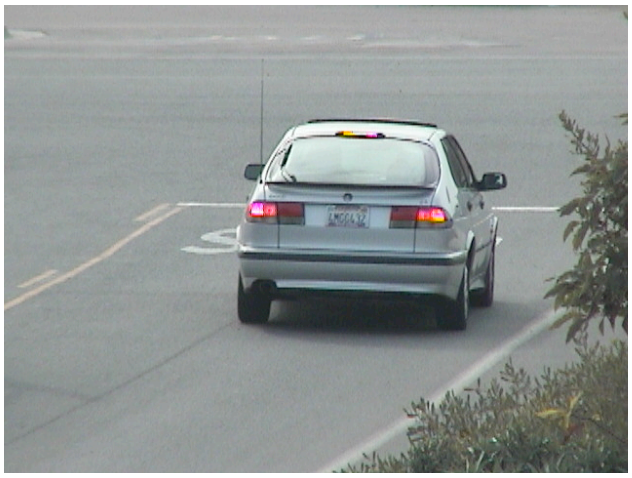


Figure 1: A frame from the video stream used for extracting training sets and test sets.

sideways, such that the top left license plate image was captured at 8am and the bottom right at 4pm. Note the dark areas in the image – this is most likely a result of cloud cover, and this illumination change can be accounted for by variance normalizing the images as shown in Figure 2(b). Although this variance normalization technique does improve the consistency of license plate examples, it had little effect on the overall results and was thus not used as it is fairly computationally costly, even when using the integral and square-integral image techniques.

Negative training examples were obtained by randomly sampling 45 $\times$ 15 windows of parts of frames known not to contain any license plates, for a total of 5,431 examples, initially. This set was increased in subsequent trainings of AdaBoost to include areas falsely identified as license plates, which brought the total of negative examples to just under 10,000.

A training test set of 158 frames was used, each frame containing exactly one instance of a license plate, and each of which was used for evaluating the performance of the trained classifier.

3. Feature Selection

The AdaBoost algorithm is used to select a small number of weak classifiers from a very large set of weak classifiers and construct a strong classifier, which makes classifications based on the sum of the weighted hypotheses of the selected weak classifiers. Each weak classifier is designed to produce a binary response on a single feature and need only be slightly more accurate than random guessing. The features to which the weak classifiers respond are important in terms of overall accuracy and should be chosen to discriminate well between license plates and non-license plates.

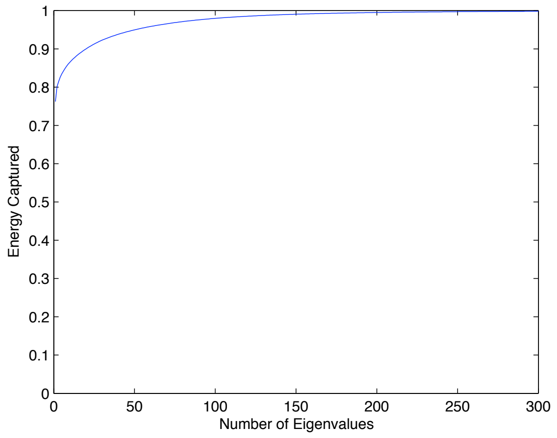
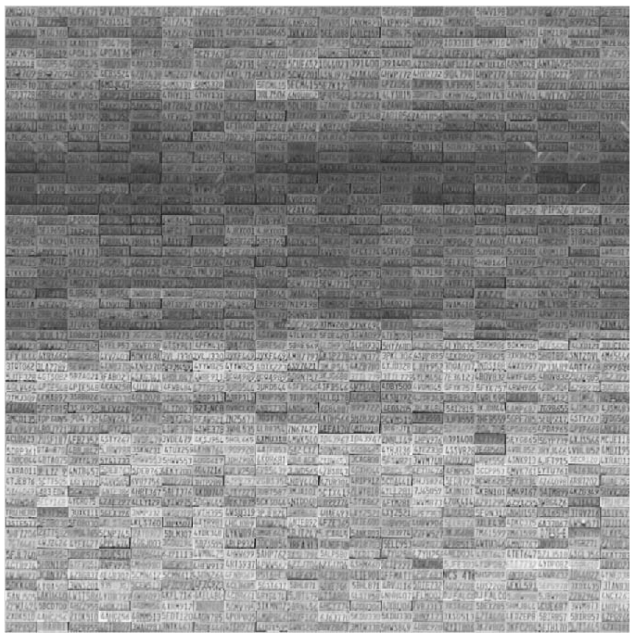


Figure 3: PCA on all 1,520 positive license plate examples with a window of interest of 0-300. Note that 30 components are required to capture 90% of the energy.

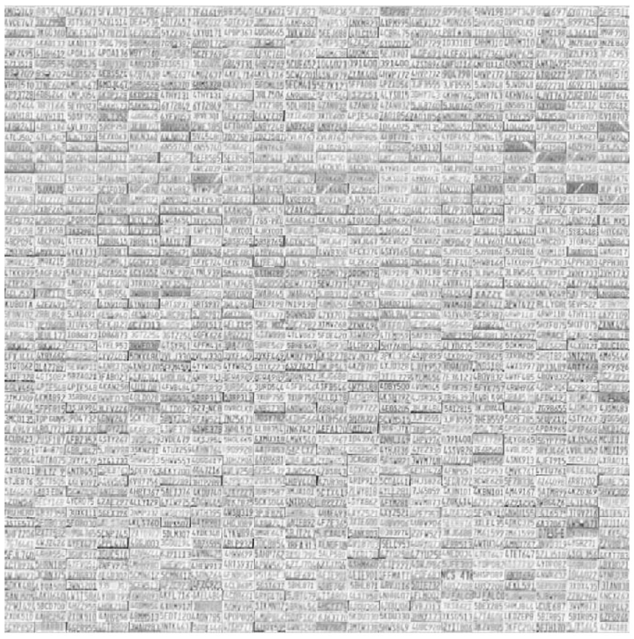
Viola and Jones use Haar-like features, where sums of pixel intensities are computed over rectangular subwindows [2]. Chen and Yuille argue that, while this technique may be useful for face detection, text has little in common with faces [1]. To support their assumption, they perform principal component analysis (PCA) on their training examples and find that about 150 components are necessary to capture 90 percent of the variance, where as in typical face datasets, only a handful would be necessary. To investigate whether this was the case with license plates, a similar plot was constructed, shown in Figure 3. Unlike the text of various fonts and orientations with which Chen and Yuille were working, license plates require much fewer components to capture most of the variance. However, an eigenface-based approach for classification yielded very unsatisfactory results, and is extremely expensive to compute over many search windows. Fisherface-based classification, which is designed to maximize between-class scatter to within-class scatter, also yielded unsatisfactory results.

It is desirable to select features, which produce similar results on all license plate images and are good at discriminating between license plates and non-license plates. After pre-scaling all training examples to the same 45 $\times$ 15 size and aligning them, the sum of the absolute values of their x and y derivatives exhibit the pattern shown in Figure 4. The locations of the 7 digits of a California license plate are clearly visible in the y derivative and y derivative variance. Although, the x derivative and x derivative variance show the form which Yuille and Chen report for text images, the y derivative and y derivative variance is quite different and yields a wealth of information.

A total of 2,400 features were fed into the AdaBoost al-

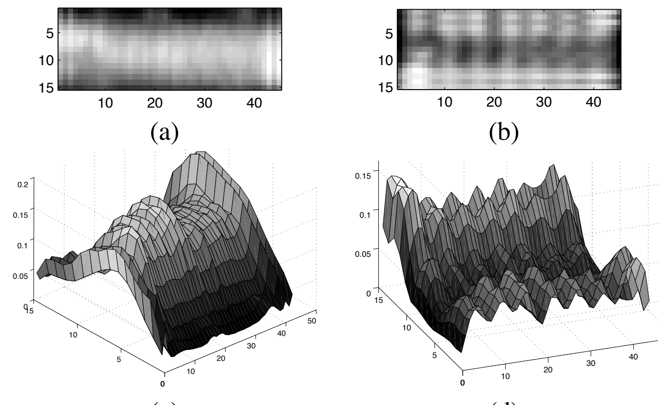


(a)



(b)

Figure 2: (a) 1,200 of 1,520 training examples. (b) Same images variance normalized.



(a) (b) (c) (d)

Figure 4: The means of the absolute value of the (a) x derivative, and (b) y derivative, and the variance of the (c) x derivative, and (d) y derivative.

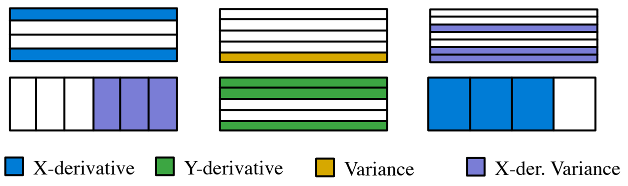


Figure 5: Types of features selected by AdaBoost. The sum of values computed over colored regions are subtracted from non-colored regions.

gorithm. These were a variation of the Haar-like features used by Viola and Jones [2], but more generalized, yet still computationally simple. A scanning window was evenly divided into between 2 and 7 regions of equal size, either horizontal or vertical. Each feature was then a variation on the sum of values computed in a set of the regions subtracted from the sum of values in the remaining set of regions. Therefore, each feature applied a thresholding function on a scalar value. Some of these features are shown in Figure 5.

The values of the regions of each window were either the means of pixel intensities, derivatives, or variance of derivatives. Each weak classifier was trained on a single feature by forming class conditional densities (CCD) from the training examples. The CCD for a typical weak classifier is shown in Figure 6. When making a classification, regions where the license plate CCD is larger than the non-license plate CCD are classified as license plate, and vice-versa, instead of using a simple one dimensional threshold.

4. AdaBoost Training

In its original form, AdaBoost is used to boost the classification accuracy of a single classifier, such as a perceptron, by combining a set of classification functions to form a strong classifier. As applied to this project, AdaBoost is used to select a combination of weak classifiers to form a strong classifier. The weak classifiers are called weak because they

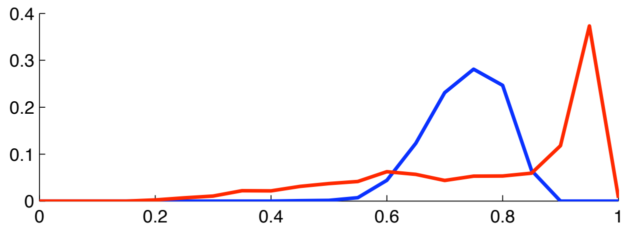


Figure 6: Typical class conditional densities for weak classifier features. There is clearly a large amount of error that cannot be avoided when making classifications, however this error is much smaller than the 50% AdaBoost requires to be effective.

only need to be correct 51% of the time.

At the start of training, each training example $(x_1, y_1)...(x_n, y_n)$ is assigned a weight $w_i = \frac{1}{2m}$ for negatives and $w_i = \frac{1}{2l}$ for positives, where $y \in \{0, 1\}$, m is the number of negatives, and l is the number of positives. The uneven initial distribution of weights leads to the name "Asymmetric AdaBoost" for this boosting technique.

Then, for $t = 1, ..., T$ rounds, each weak classifier h_j is trained and its error is computedd as $\epsilon_t = \sum_i w_i \left| h_j(x_i) - y_i \right|$. The h_j with lowest error is selected, and the weights are updated according to:

$$w_{t+1,i} = w_{t,i} \left(\frac{\epsilon_t}{1 - \epsilon_t} \right)$$

if x_i is classified correctly, and not modified if classified incorrectly. This essentially forces the weak classifiers to concentrate on "harder" examples that are most often misclassified.

After T rounds, T weak classifiers are selected and the strong classifier makes classifications according to

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where, $\alpha_t = ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.

This method was applied to the dataset described above for $T = 100$ rounds. Figure 5 shows the first few types of features selected by AdaBoost. A total of 37 variance based (16 x-derivative, 21 mean pixel intensity), 40 x-derivative, 18 y-derivative, 5 mean pixel intensity features were selected in the $T = 100$ rounds of final training.

Figure 7 shows the classification error of the strong classifier on the training set as a function of the number of rounds. The 10,000 training examples are quickly learned, and by the 50th round, no positive examples are misclassified. It is interesting to note that the error rate of the individual weak classifiers remains at about 0.35 for much of the process, which indicates that their capability to discriminate the two classes should be unaffected to a certain extent by applying more negative examples, particularly the false positives from the test set. This procedure of feeding false positives as negative examples for retraining was done once leading to great improvement, and it seems this can be applied many more times.

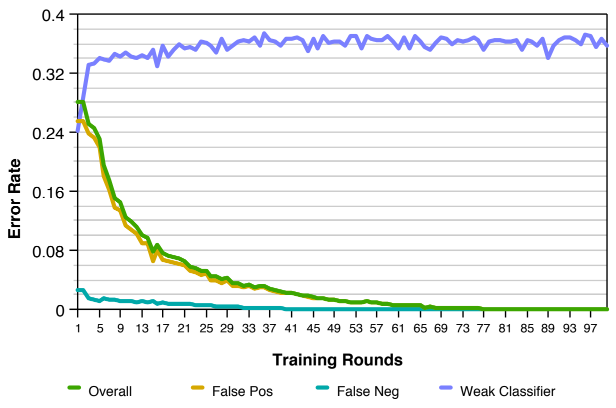


Figure 7: Error rates for 100 training rounds.

To avoid necessary computation at each scanning window location, the integral images for pixel intensities, derivative intensities, and square-of-derivative intensities were used. Those features, which use the variance of derivatives in their classification, access derivative and square-of-derivative intensity integral images to quickly compute the variance

$$\sigma^2 = m^2 - \frac{1}{N} \sum x^2$$

where m is the mean and x is the derivative intensity at a particular pixel.

When sliding the search window across the frame to be analyzed, several matches are found near license plates. To handle multiple matches and yield a single "yes" location that is closest to the true location of the license plate, a form of clustering was used to group detected points that are close to each other and use the mean of these points as the detected location. This also yields a certain measure of confidence at each location – a higher number of points in a group means a more likely location for a license plate.

Since cars do not travel too large of a distance in front of the camera, the size of the license plate in the image remains roughly the same, therefore scanning at various scales was not necessary.

5. Results

The trained strong classifier was applied to both the 158-image training test set and live video streams. A detection rate of 95.6% with a false positive rate of 5.7% was achieved

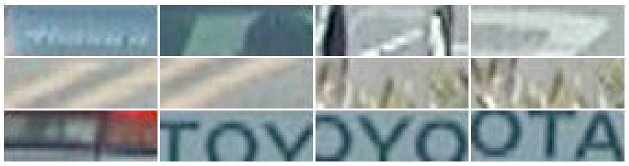


Figure 8: Examples of image windows misclassified as license plates.

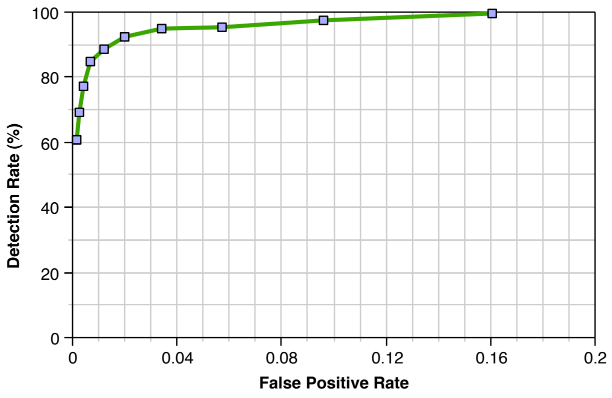


Figure 9: Receiver Operating Characteristic (ROC) curve for the 100 feature classifier.

in the former using the default threshold of 0.5 in the strong classifier function $h(x)$. By varying this threshold, different detection rates and false positive rates were achieved, as shown in the Receiver Operating Characteristic (ROC) curve in Figure 9.

Most miss-classifications were on parts of the image containing vegetation and road markings, as shown in Figure 8. However, some included the names of vehicle models that are commonly attached to the rear of cars ("4Runner"), and others included text markings on the backs of some trucks.

The vegetation-containing false positives can easily be ruled out by applying simple tests on color components of license plate candidate windows. Vegetation patches tend to have a much higher green color component compared to the rest of the images. Figure 10 shows a detection example after color thresholding is applied. Other methods for dealing with the false positives were also investigated, including distance from candidate license plate location to detected taillight locations. Taillights make for easy and computationally efficient objects to detect, and for the majority of cars, license plates are positioned equi-distant from taillights.

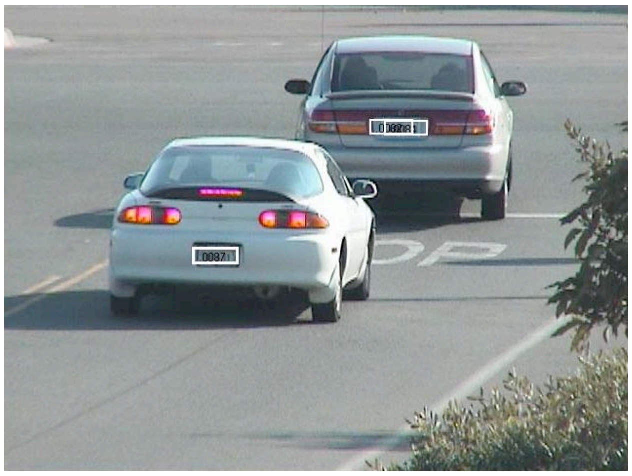


Figure 10: Detected license plate locations on a frame from a live video stream.

6. Conclusions & Future Work

The AdaBoost algorithm has shown itself to be promising for the task of license plate detection. Clearly, more work needs to be done, but this seems like a good start.

A simple and easy way to improve the ROC curve would be to use even more classifiers. Viola and Jones [2] use a 200 feature classifier. Further classes of features, including histogram tests, edge-linking, and color-based thresholds, should also be investigated as they play a large role in text detection according to Chen and Yuille [1]. The features currently used are also not as independent as they should be.

The detected images of license plates being 45 $\times$ 15 are not very large and performing OCR on them will most likely not be very effective without first applying some sort of super-resolution technique that uses several frames from the video data. Once a higher resolution image is obtained, the image will have to pass through an adaptive binarization filter to aid in the OCR process. The OCR process may need to be custom created to specifically work well with the text found on license plates. In addition, Amit et al.'s shape detection method should be investigated.

References

[1] Xiangrong Chen; Yuille, A.L. Detecting and reading text in natural scenes. *Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Proceedings of the 2004 IEEE Computer Society Conference on , Volume: 2 , 27 June - 2 July 2004 pp. 366-373

[2] Viola P., Jones M. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, 2001. CVPR 2001.* Proceedings of the 2001 IEEE

Computer Society Conference on , Volume: 1, 8-14 Dec. 2001 Pages:I-511 - I-518 vol.1

[3] Hegt HA, de la Haye RJ, Khan NA. A high performance license plate recognition system. [Conference Paper] *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics* (Cat. No.98CH36218). IEEE. Part vol.5, 1998, pp.4357-62 vol.5. New York, NY, USA.

[4] Kamat V., Ganesan S. An efficient implementation of the Hough transform for detecting vehicle license plates using DSP'S. [Conference Paper] Proceedings. *Real-Time Technology and Applications Symposium (Cat. No.95TH8055). IEEE Comput. Soc. Press. 1995*, pp.58-9. Los Alamitos, CA, USA.

[5] Yanamura Y., Goto M., Nishiyama D, Soga M, Nakatani H, Saji H. Extraction and tracking of the license plate using Hough transform and voted block matching. [Conference Paper] *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683). IEEE. 2003*, pp.243-6. Piscataway, NJ, USA.

[6] Amit Y., Geman D., Fan X. A course-to-Fine Strategy for Multiclass Shape Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence. 2004*, pp. 1606-1621