Recognizing Cars

Louka Dlagnekov, Serge Belongie
Department of Computer Science and Engineering
University of California, San Diego, CA 92093
{ldlagnek,sjb}@cs.ucsd.edu

Abstract

License Plate Recognition (LPR) is a fairly well explored problem and is already a component of several commercially operational systems. Many of these systems, however, require sophisticated video capture hardware possibly combined with infrared strobe lights or exploit the large size of license plates in certain geographical regions and the (artificially) high discriminability of characters. In this paper, we describe an LPR system that achieves a high recognition rate without the need for a high quality video signal from expensive hardware. We also explore the problem of car make and model recognition for purposes of searching surveillance video archives for a partial license plate number combined with some visual description of a car. Our proposed methods will provide valuable situational information for law enforcement units in a variety of civil infrastructures.

1 Introduction

License plate recognition (LPR) is widely regarded to be a solved problem with many systems already in operation. Some well-known settings are the London Congestion Charge program in Central London, border patrol duties by the U.S. Customs, and toll road enforcement in parts of Canada and the United States. Although few details are released to the public about the accuracy of commercially deployed LPR systems, it is known that they work well under controlled conditions. However, they have two main disadvantages which we address in this paper.

Firstly, they require high-resolution and sometimes specialized imaging hardware. Most of the academic research in this area also requires high-resolution images or relies on geographically-specific license plates and takes advantage of the large spacing between characters in those regions and even the special character features of commonly misread characters.

Secondly, LPR systems by their nature treat license plates as cars' fingerprints. In other words, they determine a vehicle's identity based solely on the plate attached to it. One can imagine, however, a circumstance where two plates from completely different make and model cars are swapped with malicious intent, in which case these systems would not find a problem. We as humans are also not very good at reading cars' license plates unless they are quite near us, nor are we very good at remembering all the characters. However, we *are* good at identifying and remembering the appearance of cars, and therefore their makes and models, even when they are speeding away from us. In fact, the first bit of information Amber Alert signs show is the car's make and model and only then its license plate number, sometimes not even a complete number. Therefore, given the description of a car and a partial license plate number, the authorities should be able to query their surveillance systems for similar vehicles and retrieve a timestamp of when that vehicle was last seen along with archived video data for that time.

In this paper, we describe an LPR method that performs well without the need for expensive imaging hardware and also explore car make and model recognition (MMR). Because of the complementary nature of license plate and make and model information, the use of MMR can not only boost the LPR accuracy, but allow for a more robust car surveillance system.

1.1 Previous Work

Most LPR systems employ detection methods such as corner template matching [11] and Hough transforms [12] [27] combined with various histogram-based methods. Kim et al. [13] take advantage of the color and texture of Korean license plates (white characters on green background, for instance) and train a Support Vector Machine (SVM) to perform detection. Their license plate images range in size from 79×38 to 390×185 pixels, and they report processing low-resolution input images (320×240) in over 12 seconds on a Pentium3 800MHz, with a 97.4% detection rate and a 9.4% false positive rate. Simpler methods, such as adaptive binarization of an entire input image followed by character localization, also appear to work as shown by Naito et al. [15] and [3], but are used in settings with little background clutter and are most likely not very robust.

The most common custom OCR approach used by existing LPR systems is correlation-based template matching [16], sometimes done on a group of characters [6]. Sometimes, the correlation is done with principal component analysis (PCA) [9]. Others [22] apply connected component analysis on binarized images to segment the characters and minimize a custom distance measure between character candidates and templates. Classification of segmented characters can also be done using neural networks [17] with good results.

Instead of explicitly segmenting characters in detected plates, Amit et al. [2] use a coarse-to-fine approach for both detection and recognition of characters on license plates. Although they present high recognition rates, the license plate images they worked with were of high-resolution, and it is not clear whether their method will be as effective on the low-resolution images in our datasets.

To the best of our knowledge, MMR is a fairly unexplored recognition problem. Various work has been done on car detection in street scene images [10] [21] [19] and aerial photographs [20]. Dorko and Schmid [7] use scale invariant features to detect cars in images with 50% background on average. Agarwal et al. [1] automatically create a vocabulary of car parts, such as tires and windshields, from training images and detect cars by finding individual parts and comparing their spatial relations. Interestingly, most of the car detection literature only deals with side-views of cars, perhaps because from a large distance the side profile provides richer and thus more discriminating features.

The work of Ferencz et al. [8] is most closely related to our problem statement. Their work is helping develop a wide-area car tracking system and is not formulated as a recognition problem, but what they call an *object identification* problem. In our system we are interested in determining to which make and model class a new vehicle belongs, and although all classes consist of cars, there is fair amount of variation within each of the make and model classes. In contrast, Ferencz et al. are interested in determining whether two images taken at different times and camera orientations are of the exact same car, where there is really only a single example that serves as a model. They solve this problem by automatically finding good features on side views of cars from several hundred pairs of training examples, where good features refer to features that are good at discriminating between cars from many small classes.

1.2 Datasets

Video data for training and testing was 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. The video stream has a resolution of 640×480 and is sampled at 6 frames per second.

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

Using this video source, we sampled images of 419 cars, each at approximately the same location in the intersection, such that the size of the license plate was 104×31 pixels. The plates were then manually extracted from these images and split into a training set of 359 images and a test set of 60 images.

2 License Plate Detection

In any object recognition system, there are two major problems that need to be solved – that of detecting an object in a scene and that of recognizing it; detection being an important requisite. We approached the license plate detection problem as a text extraction problem [5]. The detection method can be described as follows. A window of interest, of roughly the dimensions of a license plate image, is placed over each frame of the video stream and its image contents are passed as input to a classifier whose output is 1 if the window appears to contain a license plate and 0 otherwise. The window is then placed over all possible locations in the frame and candidate license plate locations are recorded for which the classifier outputs a 1.

In reality, this classifier, which we shall call a strong classifier, weighs the decisions of many weak classifiers, each specialized for a different feature of license plates, thereby making a much more accurate decision. This strong classifier is trained using the AdaBoost algorithm, and the weak classifiers are considered weak since they only need be over 50% accurate. Over several rounds, AdaBoost selects the best performing weak classifier from a set of weak classifiers, each acting on a single feature.

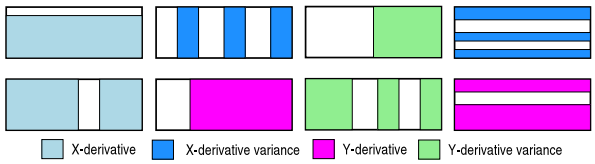

Figure 2: Types of features selected by AdaBoost. The sum of values computed over colored regions are subtracted from the sum of values over non-colored regions.

2.1 Feature Selection

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. Viola and Jones use Haar-like features, where sums of pixel intensities are computed over rectangular sub-windows [26]. Chen and Yuille argue that, while this technique may be useful for face detection, text has little in common with faces [5]. 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, whereas in typical face datasets, only a handful would be necessary.

A total of 2,400 features were generated as input to the AdaBoost algorithm. These were a variation of the Haar-like features used by Viola and Jones [26], but more generalized, yet still computationally efficient. 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 2.

The values of the regions of each window were the means of pixel intensities, derivatives, or variance of derivatives. None of the features actually selected by AdaBoost used raw pixel intensities, however, probably because of their poor discriminating ability with respect to wide illumination differences. Each weak classifier was a Bayes classifier, trained on a single feature by forming class conditional densities (CCD) from the training examples. When making a decision, 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.

2.2 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 only need to be correct 51% of the time.

At the start of training, each training example (x_i, y_i) 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

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

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

In addition to the 359 manually cropped positive training examples, we generated additional positive examples by extracting images from 10 random offsets (up to 1/8 of the width and 1/4 of the height of license plates) of each license plate location (for a total of 3,590). We found that this yielded better results than just using the license plate location for a single positive example per hand-labeled region. Of course, when the detector was in operation, it fired at many regions around a license plate, which we in fact used as an indication of the quality of a detection.

To generate negative examples, we picked 28 license plate-sized images from random regions known not to contain license plates in each positive frame, which resulted in 10,052 per set. We then applied a sequence of two bootstrap operations where false positives obtained from testing on the training data were used as additional negative examples for re-training the cascade, and obtained 9,948 additional negative examples.

2.3 Results

Scanning every possible location of every frame would be very slow were it not for two key optimization techniques introduced by Viola and Jones – integral images and cascaded classifiers [26]. The integral image technique allows

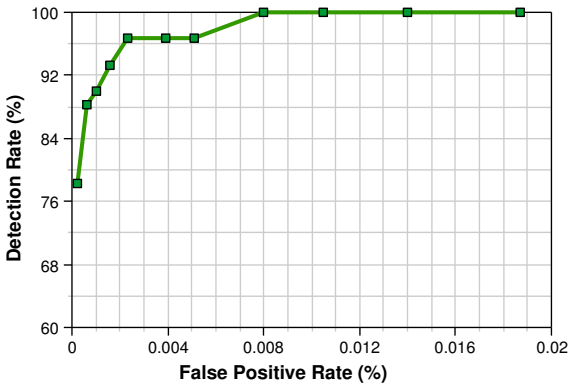

Figure 3: ROC curve for a 6-stage cascaded detector, with 2, 3, 6, 12, 40, and 60 features per stage respectively.

Figure 4: Examples of regions incorrectly labeled as license plates.

for an efficient implementation and the cascaded classifiers greatly speed up the detection process, as not all classifiers need be evaluated to rule out most non-license plate sub-regions. With these optimizations in place, the system was able to process 10 frames per second at a resolution of 640×480 pixels.

Figure 3 shows a receiver operating characteristic (ROC) curve for our cascaded detector. We did not achieve as low a false positive rate per detection rate on our datasets as either Chen and Yuille, or Viola and Jones, but the false positive rate of 0.002% for a detection rate of 96.67% in set 3 is quite tolerable. In practice, the number false positives per region of each frame is small compared to the number of detections around a license plate in the frame. Therefore, in our final detector we do not consider a region to contain a license plate unless the number of detections in the region is above a threshold.

Figure 4 shows a few examples of regions that our detector incorrectly labeled as license plates in our test dataset. Perhaps not surprisingly, a large number of them are text from advertising on city buses, or the UCSD shuttle. Those that contain taillights can easily be pruned by applying a color threshold.

3 License Plate Recognition

In this section, we present a process to recognize the characters on detected license plates. We begin by describing a method for tracking license plates over time and then describe our optical character recognition (OCR) algorithm.

3.1 Tracking

More often than not, the false positive detections from our license plate detector were erratic, and if on the car body, their position was not temporally consistent. We use this fact to our advantage by tracking candidate license plate regions over as many frames as possible. Then, only those regions with a smooth trajectory are deemed valid. The tracking of license plates also yields a sequence of samplings of the license plate, which can be used as input to a super-resolution pre-processing step before OCR is performed on them.

Numerous tracking algorithms exist that could be applied to our problem. Perhaps the most well-known and popular is the Kanade-Lucas-Tomasi (KLT) tracker [23]. The KLT tracker makes use of a Harris corner detector to detect good features to track in a region of interest (our license plate) and measures the similarity of every frame to the first allowing for an affine transformation. Sullivan et al. [25] make use of a still camera for the purposes of tracking vehicles by defining regions of interest (ROI) chosen to span individual lanes. They initiate tracking when a certain edge characteristic is observed in the ROI and make predictions on future positions of vehicles. Those tracks with a majority of accurate predictions are deemed valid. Okuma et al. [18] use the Viola and Jones [26] framework to detect hockey players and then apply a mixture particle filter using the detections as hypotheses to keep track of the players.

Although each of these tracking methods would probably have worked well in our application, we chose a far simpler approach which worked well in practice. Because detecting license plates is efficient we simply run our detector on each frame and for each detected plate we determine whether that detection is a new plate or an instance of a plate already being tracked. To determine whether a detected plate is new or not, the following conditions are checked:

- the plate is within T pixels of an existing tracker
- the plate is within T' pixels of an existing tracker and the plate is within θ degrees of the general direction of motion of the plates in the tracker's history

If any of these are true, the plate is added to the corresponding tracker, otherwise a new tracker is created for that plate. In our application T' was an order of magnitude larger than T. Our tracking algorithm was also useful for discarding false positives from the license plate detector. The erratic motion of erroneous detections usually resulted in the initiation of several trackers each of which stored few image sequences. Image sequences of 5 frames or fewer were discarded.

3.2 Character Recognition

It was our initial intent to apply a binarization algorithm, such as a modified version of Niblack's algorithm as used by Chen and Yuille [5], on the extracted license plate images from our detector, and then use the binarized image as input to a commercial OCR package. We found, however, that even at a resolution of 104×31 the OCR packages we experimented with yielded very poor results. Perhaps this should not come as a surprise considering the many custom OCR solutions used in existing LPR systems.

Unless text to be read is in hand-written form, it is common for OCR software to segment the characters and then perform recognition on the segmented image. The simplest methods for segmentation usually involve the projection of row and column pixels and placing divisions at local minima of the projection functions. In our data, the resolution is too low to segment characters reliably in this fashion, and we therefore decided to apply simple template matching instead, which can simultaneously find both the location of characters and their identity.

The algorithm can be described as follows. For each example of each character, we search all possible offsets of the template image in the license plate image and record the top N best matches. The searching is done using normalized cross correlation (NCC), and a threshold on the NCC score is applied before considering a location a possible match. If more than one character matches a region the size of the average character, the character with the higher correlation is chosen and the character with the lower correlation is discarded. Once all templates have been searched, the characters for each region found are read left to right forming a string. N is dependent on the resolution of the license plate image and should be chosen such that not all N matches are around a single character when the same character occurs more than once on a plate, and not too large so that not all possible regions are processed.

This method may seem inefficient, however, the recognition process takes on the order of half a second for a resolution of 104×31, which we found to be acceptable.

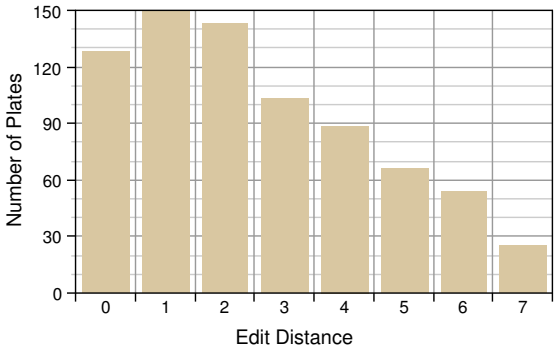

Figure 5: Character recognition results showing the number of license plates recognized with 0, 1, 2, 3, 4, 5, 6, and 7 mistakes.

3.3 Results

Training and test data for character recognition was obtained by running our license plate detector on several hours of video and extracting sequences of images for each tracked license plate. This process resulted in a total of 879 plate sequences each of which was labeled by hand. Of these, 121 were chosen at random to form an alphabet of characters for training. These 121 sequences contained the necessary distribution of characters to form 10 examples per character, for a total of 360 examples (26 letters and 10 digits). The remaining 758 plates were used for testing the OCR rate.

To test the accuracy of the character recognition, we used the edit distance to measure how similar our recognized text was to the labeled plates in the test set. Because certain characters are easily confused with others, even by humans, we applied a 'loose' character equality test whenever the edit distance algorithm compared two characters. The groups of characters {'O', '0', 'D', 'Q'}, {'E', 'F'}, {'I', 'T', '1'}, {'B', '8'}, and {'Z', '2'} were each considered of the same type and no penalty was applied for incorrect readings within the group. Figure 5 shows the number of license plates read with various numbers of mistakes with and without using the 'loose' comparison measure. Note that over half of the test set was recognized with two or fewer mistakes.

4 Make and Model Recognition

As with the license plate recognition problem, detecting the car is the first step to performing make and model recognition (MMR). To this end, one can apply a motion segmentation method to estimate a region of interest (ROI) containing

the car. Instead, we decided to use the location of detected license plates as an indication of the presence and location of a car in the video stream and to crop an ROI of the car for recognition. This method would also be useful for make and model recognition in static images, where the segmentation problem is more difficult.

4.1 Datasets

We automatically generated a database of car images by running our license plate detector and tracker on several hours of video data and cropping a fixed window of size 400×220 pixels around the license plate of the middle frame of each tracked sequence. This method yielded 1,140 images in which cars of each make and model were of roughly the same size. The crop window was positioned such that the license plate was centered in the bottom third of the image. We chose this position as a reference point to ensure matching was done with only car features and not background features. Had we centered the license plate both vertically and horizontally, cars that have their plates mounted on their bumper would have exposed the road in the image.

After collecting these images, we manually assigned make, model, and year labels to 790 of the 1,140 images. We were unable to label the remaining 350 images due to our limited familiarity with those cars. We often made use of the California Department of Motor Vehicles' web site to determine the makes and models of cars with which we were not familiar. The web site allows users to enter a license plate or vehicle identification number for the purposes of checking whether or not a car has passed recent smog checks. For each query, the web site returns smog history as well as the car's make and model description if available. The State of California requires all vehicles older than three years to pass a smog check every two years. Therefore, we were unable to query cars that were three years old or newer and relied on our personal experience to label them.

We split the 1,140 labeled images into a query set and a database set. The query set contains 38 images chosen to represent a variety of make and model classes, in some cases with multiple queries of the same make and model but different year in order to capture the variation of model designs over time. We evaluated the performance of each of the recognition methods by finding the best match in the database for each of the query images.

4.2 SIFT Matching

Scale invariant feature transform (SIFT) features recently developed by Lowe [14] are invariant to scale, rotation and even partially invariant to illumination differences, which makes them well suited for object recognition. We applied SIFT matching to the problem of MMR as follows:

1. For each image $\mathbf{d}$ in the database and a query image $\mathbf{q}$, perform keypoint localization and descriptor assignment.
2. For each database image $\mathbf{d}$:
   (a) For each keypoint $\mathbf{k}_q$ in $\mathbf{q}$ find the keypoint $\mathbf{k}_d$ in $\mathbf{d}$ that has the smallest L^2 distance to $\mathbf{k}_q$ and is at least a factor of α smaller than the distance to the next closest descriptor. If no such $\mathbf{k}_d$ exists, examine the next $\mathbf{k}_q$.
   (b) Count the number of descriptors n that successfully matched in $\mathbf{d}$.
3. Choose the $\mathbf{d}$ that has the largest n and consider that the best match.

4.3 Discussion

We found that a few types of keypoint matches resulting from the above algorithm did not contribute to the selection of a best car match. For example, some matching keypoints corresponded to entire groups of digits and letters on the license plates of a query image and a database image even though the the cars to which they belonged looked quite different. Since the best car match in the database is determined based on the number of matched keypoints, spurious matches should be ignored. We, therefore, applied the following keypoint pruning procedures:

- Limit horizontal distance between matching keypoints. This helps remove outliers when estimating an affine transformation between the query and database images.
- Ignore keypoints that occur in the license plate region.
- Do not allow multiple query keypoints to match to the same database keypoint.
- Compute an affine transformation from the query to the database image when there are more than three matching keypoints. If the scale, shear, or translation parameters of the transformation are outside a threshold, set the number of matching keypoints n to 0.

We used Lowe's implementation [14] of the keypoint localization part of the algorithm. Unlike in Lowe's implementation, the query's keypoint descriptors were compared with the keypoint descriptors of each image in the database. This means that the second best descriptor was not chosen for an object other than the current database image. Also, modifying the threshold from the 0.36 appearing in the published code to 0.60 (which is closer to the suggested in Lowe's

paper) increased the number of matches, but had little effect on the overall recognition rate – misclassified cars using one method were correctly classified with the other at the expense of different misclassifications.

When the number of matching descriptors between the query image and a database image is equal to that of another database image, we break the tie by selecting the database image with the smaller overall L^2 distance between all the descriptors. This only occurred when the best matches in the database had one or two matching descriptors, and applying the tie-break procedure had little effect on the overall recognition rate.

4.4 Results

The SIFT matching algorithm described above yielded a recognition rate of 89.5% on the query set. Recognition results for some of the queries in the test set are shown in Figure 6. The top 10 matches were all of the same make and model for some of the queries with over 20 similar cars in the database.

Most of the queries SIFT matching was not able to classify correctly had 5 or fewer entries similar to it in the database. Based on the results of queries corresponding to makes and models with many examples in the database, it is safe to assume that having more examples per make and model class will increase the recognition rate.

5 Summary and Conclusions

We have presented a useful framework for car recognition that combines LPR and MMR. Our recognition rates for both sub-problems are very promising and can serve as an important foundation to a query-based car surveillance system. Our LPR solution is real-time and works well with inexpensive camera hardware and does not require infrared lighting or sensors as are normally used in commercial LPR systems. Our MMR solution is also very accurate, however, further research is required to make it real-time.

Several possibilities exist that may help in that regard. Instead of comparing features in the query image with every single database image, it would be useful to cluster the database images into groups of similar type, such as sedan, SUV, etc. and perform a hierarchical search to reduce the number of comparisons.

A promising method that is applicable to our situation is the recent work by Sivic and Zisserman [24]. They formulate the object recognition problem as a text retrieval problem, which itself has been shown to be remarkably efficient based on our daily experiences with internet search engines. Future work on MMR should investigate the possibility of incorporating a similar approach.

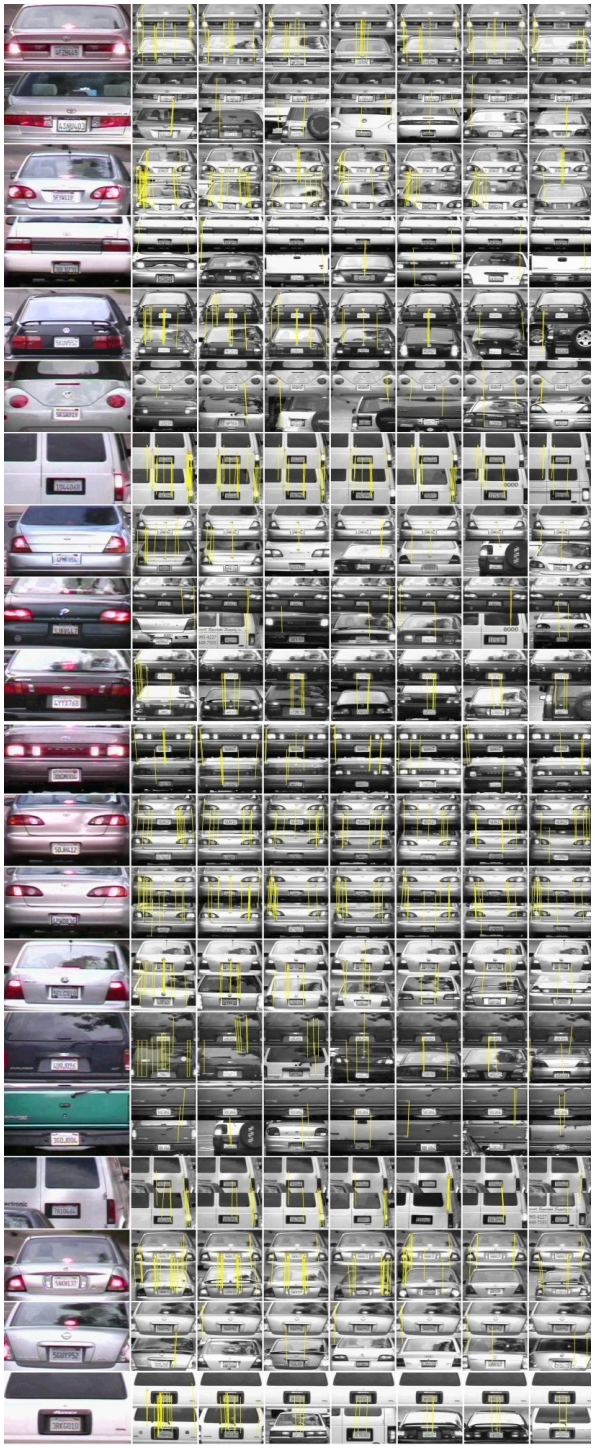

Figure 6: Recognition results on 20 query images in test set and their top 7 matches in the database. Yellow lines indicate correspondences between matched SIFT features.

In addition to searching the surveillance database for cars using some make and model description and a partial license plate, it would also be useful to be able to search for a particular color car as the make and model information may be incomplete. Various color- and texture-based image segmentation techniques used in content-based image retrieval such as [4] may be suitable for this purpose.

In our MMR work, we have not explored car pose variation beyond what normally occurs at the stop signs in our scenes. A robust MMR system should also work well in scenes where there is a large variation of poses. This could require the estimation of a car's 3-D structure to be used as additional input to the MMR algorithms.

References

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *PAMI*, 26(11):1475–1490, 2004.

[2] Y. Amit, D. Geman, X. Fan. A coarse-to-fine strategy for multiclass shape detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26, 1606–1621. 2004.

[3] G. Cao, J. Chen, J. Jiang, An adaptive approach to vehicle license plate localization. *Industrial Electronics Society*, 2003. IECON '03. Volume 2, pp 1786- 1791

[4] C. Carson, S. Belongie, H. Greenspan, J. Malik. Blobworld: color- and texture-based image segmentation using EM and its Application to image querying and classification. *PAMI*, 24(8):1026–1038, 2002.

[5] X. Chen, A. Yuille. Detecting and reading text in natural scenes. *CVPR*. Volume: 2, pp. 366–373, 2004.

[6] P. Comelli, P. Ferragina, M. Granieri, F. Stabile. Optical recognition of motor vehicle license plates. *IEEE Trans. On Vehicular Technology*, Vol. 44, No. 4, pp. 790–799, 1995.

[7] G. Dorko and C. Schmid. Selection of scale-invariant parts for object class recognition. *Proc. ICCV*, 2003.

[8] A. Ferencz, E. Miller, J. Malik. Learning hyper-features for visual identification. *NIPS*, 2004.

[9] N. Khan, R. de la Haye, A. Hegt. A license plate recognition system. *SPIE Conf. on Applications of Digital Image Processing*. 1998.

[10] B. Leung. Component-based Car Detection in Street Scene Images. *Master's Thesis*, Massachusetts Institute of Technology, 2004.

[11] H. Hegt, R. de la Haye, N. Khan. A high performance license plate recognition system. *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.

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

[13] K. Kim, K. Jung, and J. Kim, Color texture-based object detection: an application to license plate localization. Lecture Notes in Computer Science: International Workshop on Pattern Recognition with Support Vector Machines, pp. 293–309, 2002.

[14] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2(60):91–110, 2004.

[15] T. Naito, T. Tsukada, K. Yamada, K. Kozuka, S. Yamamoto, Robust license-plate recognition method for passing vehicles underoutside environment. *IEEE T VEH TECHNOL* 49 (6): 2309–2319 NOV 2000.

[16] T. Naito, T. Tsukuda, K. Yamada, K. Kozuka. Robust recognition methods for inclined license plates under various illumination conditions outdoors. *Proc. of IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems*, pp. 697702,1999.

[17] J. Nijhuis, M. Brugge, K. Helmholt, J. Pluim, L. Spaanenburg, R. Venema, M. Westenberg. Car license plate recognition with neural networks and fuzzy logic. *Proceedings of IEEE International Conference on Neural Networks, Perth, Western Australia*, pp 21852903. 1995.

[18] K. Okuma, A. Teleghani, N. de Freitas, J. Little and D. Lowe. A boosted particle filter: Multitarget detection and tracking, ECCV, 2004.

[19] C. Papageorgiou, T. Poggio. A trainable object detection system: car detection in static images. *MIT AI Memo*, 1673 (CBCL Memo 180), 1999.

[20] C. Schlosser, J. Reitberger, S. Hinz, Automatic car detection in high-resolution urban scenes based on an adaptive 3D-model. *Proc. IEEE/ISPRS Workshop on "Remote Sensing and Data Fusion over Urban Areas"*. 2003.

[21] H. Schneiderman, T. Kanade. A statistical method for 3D object detection applied to faces and cars. *IEEE CVPR*, 2000.

[22] V. Shapiro, G. Gluhchev. Multinational license plate recognition system: segmentation and classification. *Proc. ICPR* 1051–4651, 2004.

[23] J. Shi, C. Tomasi, Good Features to track. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR94), Seattle*, June 1994.

[24] J. Sivic, A. Zisserman. Video google: a text retrieval approach to object matching in videos. *Proc. ICCV*, 2003.

[25] G. Sullivan., K. Baker, A. Worrall, C. Attwood, P. Remagnino, Model-based vehicle detection and classification using orthographic approximations. *Image and Vision Computing*. 15(8), 649–654.

[26] P. Viola, M. Jones. 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, pp. 511–518. 2001.

[27] Y. Yanamura, M. Goto, D. Nishiyama, M. Soga, H. Nakatani, H. Saji. Extraction and tracking of the license plate using Hough transform and voted block matching. *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings*, pp.243–6. Piscataway, NJ, USA, 2003.