Real-Time License Plate Recognition on an Embedded DSP-Platform

Clemens Arth, Florian Limberger, Horst Bischof
Graz University of Technology
Institute for Computer Graphics and Vision
Inffeldgasse 16/2, 8010 Graz, Austria
{arth,bischof}@icg.tu-graz.ac.at, florian.limberger@aon.at

Abstract

In this paper we present a full-featured license plate detection and recognition system. The system is implemented on an embedded DSP platform and processes a video stream in real-time. It consists of a detection and a character recognition module. The detector is based on the AdaBoost approach presented by Viola and Jones. Detected license plates are segmented into individual characters by using a region-based approach. Character classification is performed with support vector classification. In order to speed up the detection process on the embedded device, a Kalman tracker is integrated into the system. The search area of the detector is limited to locations where the next location of a license plate is predicted. Furthermore, classification results of subsequent frames are combined to improve the class accuracy.

*The major advantages of our system are its real-time capability and that it does not require any additional sensor input (*e.g. *from infrared sensors) except a video stream. We evaluate our system on a large number of vehicles and license plates using bad quality video and show that the low resolution can be partly compensated by combining classification results of subsequent frames.*

1. Introduction

There is a need for intelligent traffic management systems in order to cope with the constantly increasing traffic on todays roads. Video based traffic surveillance is an important part of such installations. Information about current situations can be automatically extracted by image processing algorithms. Beside vehicle detection and tracking, *identification* via license plate recognition is important for a variety of applications. These include, *e.g.* automatic congestion charge systems, access control, tracing of stolen cars, or identification of dangerous drivers.

Deploying *smart cameras* for the purpose of video based traffic surveillance has the advantage of allowing direct on site image processing tasks. The information is extracted from the input data and sent in compressed form to a central node, hence decreasing demands on both communication and computation infrastructure. Furthermore, *embedded systems* are cheaper than general purpose computers and suitable for deployment in harsh environments due to their physical robustness.

Most license plate recognition systems in operation today use special hardware like high resolution cameras or infrared sensors to improve the input quality and they operate in controlled settings. A different solution, as proposed in this work, is the continuous analysis and consideration of subsequent frames. However, this implicates that enough frames are captured by the capturing device and are processed by the processing engine. We state that for the tasks of car detection and subsequent license plate recognition *real-time* is a flexible term. As about 20 fps (frames per second) might be enough *real-time* for this tasks when cars are driving at residential speeds, this is insufficient for country roads and highway traffic. In our terminology, *real-time* operation stands for fast enough operation in order to not miss a single object that moves through the scene, irrespective of the object speed.

As we will show, the major advantages of our system over all others are its real-time capabilities in city scenarios and its ability to operate under daytime conditions with sufficient daylight or artificial light from street lamps. The usage of active infrared light is popular because the light is reflected by the license plates only. By using a camera and special filters, the detection of the plates and subsequent character segmentation is relatively easy. However, the usage of alternative light sources comes at additional costs. Thus, one prerequisite during system design was to build a system which operates with conventional off-the-shelf video cameras and without additional lighting, in fa-

vor of the feasibility to deploy and setup the system within a short time. The major reason why we have focused on cars driving at residential speeds is the lack of a digital camera with a high speed shutter. The drawback of using standard TV cameras is that motion blur is a critical issue, so that reasonable character recognition becomes almost impossible when vehicles are moving at high speeds. Note that this is rather a restriction of the image acquisition device than a deficit of the system. Our system runs fully autonomous and embedded on a smart camera, and thus makes integration into an existing system, *e.g*. for access control of parking garages, possible.

In the following, we provide an overview of related work in the area of license plate detection and recognition systems. A description of our embedded platform and an overview of our system is given in section 3. Then, section 4 gives a description of the implementation, the datasets used, and the results achieved with this system, followed by a performance evaluation on the embedded device. Finally, the whole paper is summarized in section 5.

2. Related Work

Generally, license plate recognition consists of two separate parts. First, plates have to be located within the image, termed *license plate detection* followed by *license plate character recognition*, determining the plate number.

Different methods for the detection of license plates can be applied. Shapiro *et al*. [17] use a mixture of edge detection and vertical pixel projection for their detection module. In the work of Jia *et al*. [20] color images were segmented by the MeanShift algorithm into candidate regions and subsequently classified as plate or not. The AdaBoost algorithm was used by Dlagnekov and Belongie [6] for license plate detection on rather low resolution video frames. Similar features like those introduced in [5] were added to the classical set of Haar features, and the located plate was tracked over a sequence of frames. Matas and Zimmermann [13] proposed a different approach for the localization of license plates. Instead of using properties of the plate directly, the algorithm tries to find all character-like regions in the image. This is achieved by using a region-based approach. Regions are enumerated and classified due to a region descriptor using a neural network. If a linear combination of character-like regions is found, the presence of a whole license plate is assumed.

The method described above can be applied to the *segmentation*-task of character recognition as well. Shapiro *et al*. [17] use adaptive iterative thresholding and analysis of connected components for segmentation. The classification task is performed with two sets of templates. Rahman *et al*. [14] used horizontal and vertical intensity projection for segmentation and template matching for classification. Dlagnekov and Belongie [6] use the normalized cross correlation for classification by searching the whole plate, hence skipping segmentation.

Those systems described above have not been specifically designed for embedded systems. A lot of commercial mobile systems exist, most of them equipped with special hardware to improve the quality of the input image. Kamat and Ganesan [10] implemented a license plate detection system on a DSP using the Hough transform. Kang *et al*. [11] implemented a vehicle tracking and license plate recognition system on a PDA. A FPGA was used by Bellas *et al*. [3] to speed-up parts of their license plate recognition system.

A comparison of different LP detection and recognition systems is difficult as each one is subject to differing prerequisites. Furthermore, the lack of a common evaluation database makes evaluations and comparisons unfair. Our focus was on the architecture and implementation of a complete LPR system on our embedded platform. Thus, we have omitted a detailed discussion of algorithm complexity, power efficiency and accuracy. We have only cited those systems which are somehow related to mobile devices or embedded architectures, or which have recently introduced new techniques to the area of LPR.

3. Embedded Platform Description and System Overview

The target platform for our algorithms is similar to the one presented in [2]. Summarizing the most important parameters relevant for the task of implementation, the processing core is a single Texas Instruments™ C64 fixed point DSP with 1MB of cache RAM. Additionally a slower SDRAM memory block of 16MB exists. The embedded system does not incorporate a specific camera but allows for connecting any analog or digital video source over appropriate connectors.

An overview of our software framework is depicted in Figure 1. The system consists of two modules: *License Plate Detection* and *License Plate Character Recognition*. First, license plates are detected within the input-image. After post-processing, a tracker is initialized for each newly detected plate in order to optimize the detection process and to create a relation between subsequent frames. Then, detected plates are handed over to the character recognition module. The segmentation process extracts single characters, to which a class-label is assigned through classification. Finally, post-processing considers history-information acquired by the tracker to improve the classification-result.

3.1. License Plate Detection

The detection module consists of three parts: the detecting, the tracking and the post-processing step. All parts and their interaction are described in the following.

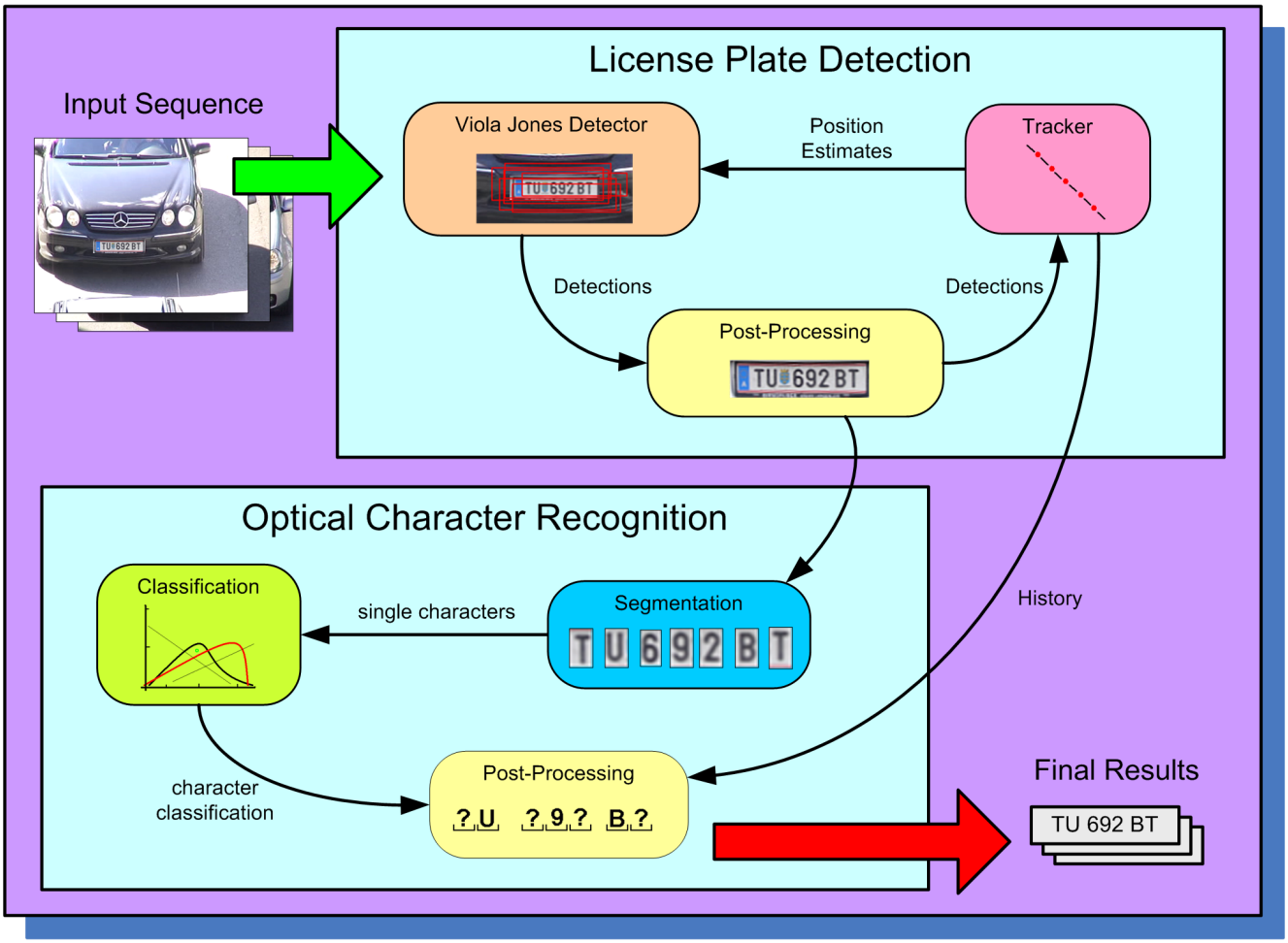

Figure 1. Our system contains two main modules. Given an image sequence as input, the first module, the detection module, applies algorithms for detecting and tracking of license plates. In the second module, segmentation and, subsequently, character recognition is performed to generate the final result.

Detector The license plate detector is based upon the framework proposed by Viola and Jones [19]. Viola and Jones created a fast object detection system by combining the AdaBoost-algorithm [8] with Haar-like features, a classifier cascade and the integral image for fast feature computation. Unlike the original framework, the RealBoost [15] is used for this detector, which provides confidence rated predictions instead of binary class-labels. Furthermore, the classifier cascade is improved by using *inter stage feature propagation* as proposed by Šochman and Matas [18]. The feature-pool is extended with features based on edge orientation histograms as proposed by Levi and Weiss [12].

Post-processing The exhaustive search technique of the Viola and Jones detector leads to multiple detections for a single license plate. Therefore, post-processing methods have to be applied in order to merge those detections. For this purpose the rather simple *non-maximum suppression* is used. The non-maximum suppression merges multiple detections, considering the confidence value of each detection. In our case all overlapping detections are substituted by the single detection with the maximum confidence value. If the false positive rate is not too high, this method achieves sufficient results. Nonetheless, the final detection does not solely contain the license plate but parts of the surroundings as well.

Tracker A Kalman tracker [9] is integrated into the system in order to limit the search of the detector to certain areas of the input image. This requires a rough calibration of the scene. Figure 2 illustrates a possible setting. Objects are assumed to enter from the top. A static region of interest is defined where the Viola and Jones detector search is performed on each frame. For each new detection a Kalman

tracker is initialized, which predicts the new object's position on the next frame. This allows for scanning solely within a small area around the estimated position.

The update and prediction operations of the tracker require less computation time than a detector scan for the whole image. Therefore, the integration is indeed useful. Furthermore, the tracking information will be important for the character classification step as well.

3.2. License Plate Character Recognition

License plate character recognition consists of three separate parts as depicted in Figure 3: First, characters are isolated with a region-based approach, followed by character classification using a *support vector machine* (SVM). Finally, during post-processing, a priori knowledge about the expected license plates and information provided by the tracker is used to improve the classification result.

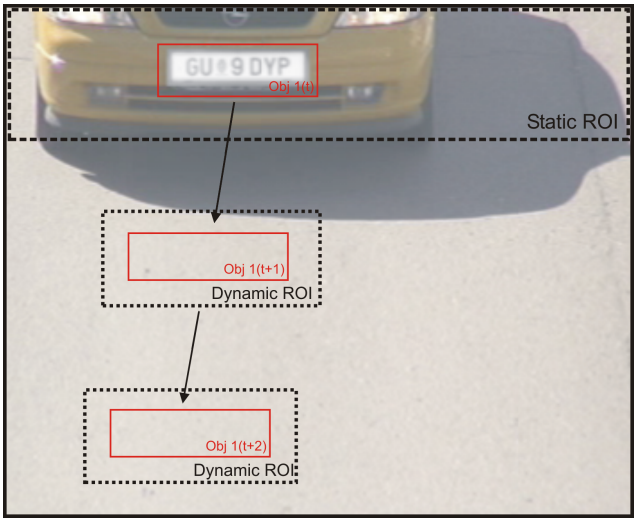

Figure 2. Illustrates a possible setting of static and dynamic search areas. In this case, objects are assumed to enter from the top. A tracker predicts the object's occurrence on the next frame, limiting the Viola and Jones exhaustive search to a small area.

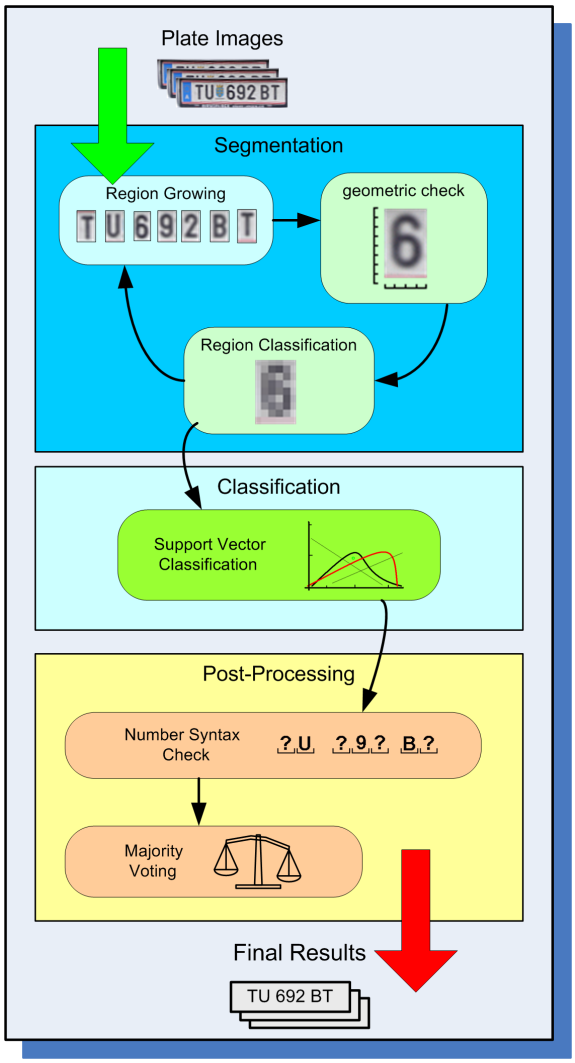

Figure 3. License Plate Character Recognition overview. After segmentation, a geometric check is performed to early discard regions unlikely being a character. The regions are subsequently pre-classified to further eliminate false positives. The Support-Vector classification is subsequently used to determine the true number. All results for a single plate track are involved in a majority-voting step to finally determine the true plate number.

Segmentation A region-based approach is used for character segmentation. Due to dark characters on white background, region-growing seeded with lowest intensity values is applied. In order to determine which regions contain a character two approaches are used. Similar to [13] a region descriptor is computed incrementally, consisting of compactness, entropy of the grayscale histogram, and central invariant statistical moments. Descriptors are classified using support vector classification.

Since the relation between the size of a license plate and its characters is approximately known in advance, classification can be achieved as well with rather simple checks containing size and width/height-ratio of found regions. Furthermore the median-height of all regions is determined and those with a high deviation are discarded. Finally, the correct alignment of subsequent regions is constrained using a Hough transform.

Classification Support vector classification is used for character classification. Segmented regions are scaled to a common patch size. The feature vector consists of direct pixel values. Since character recognition is a multi-class problem a combination of binary classifiers is required.

Two approaches have been used:

OneVsAll. k binary SVMs are trained with one class having a positive label and all other $k-1$ classes having a negative label. k decision functions exist. The target class of a feature is determined by taking the largest value of the decision function.

Tree-like structure. Characters are divided into groups of characters with similar shape. Within the upper tree layers, decisions between those sets have to be made. The determination of the final character class takes place only at the leafs of the tree. The chosen arrangement is shown in Figure 4. It was constructed manually by inspecting a previously determined confusion table.

For the SVMs radial basis functions (RBFs) are applied as kernel functions, since they achieve the best results. A radial basis function has the form

$$K(x_i, x_j) = e^{-\gamma \|x_i - y_j\|^2} \qquad \gamma > 0. \tag{1}$$

The parameters C (the penalty parameter) and γ are determined by searching the parameter space using n-fold cross validation and selecting the parameter pair yielding the highest classification rate. Both SVM structures, one-vs-all and tree shaped, are tested setting all classification-pairs to identical parameters or by determining C and γ for each classification-pair individually. For the OneVsAll approach, the former methods achieve the best results. For the tree-like structure, the best parameters are determined for each node individually. For an introduction to SVMs see [7] or [16].

Post-Processing In this step a priori knowledge about the structure of license plates is exploited *i.e*. spacing between subsequent characters and furthermore the syntax of the final character string. Needless to mention that this applies only in cases where the nationality of the license plate is known in advance or can be derived beforehand.

Additionally, classification results of subsequent frames are combined utilizing the history provided by the tracking module. A simple majority voting individually for each position within the character string is applied. The character achieving the highest count is selected.

4. Experiments

This section starts with details about the implementation on the embedded device. It continues with a description of the datasets used for training and evaluation, followed by a separate discussion of the license plate detector and the character recognition module. Finally, a performance evaluation of the implementation on the embedded device is performed.

4.1. Implementation Details

Due to the interlaced PAL video, a frame resolution of 352×288 was used on the embedded device. The detection and recognition process operates on each individual frame, finally transmitting detection and classification results and a compressed image to a connected client.

Only the classification part of both, the Viola and Jones detector and the support vector machine, was ported to the embedded device. The training of both learning algorithms was conducted off-line on a general purpose computer. The SVM library *LIBSVM* by Chang and Lin [4] was used for the SVM training and was adapted to the use on the DSP. Due to the low amount of memory (*i.e*. 1 MB Cache) available on the embedded device and the necessity for swapping out to the slower SDRAM, the memory layout required a more compact representation. Furthermore, part of the SVM kernel computation was changed to a fixed point representation due to the DSP's lack of a floating point unit.

4.2. Dataset

Two sets of test data mainly containing Austrian license plates were acquired. One set contains 260 images of license plates extracted from parking cars which were taken using a digital photo camera. The resolutions were 90×30 and 120×40. These images were taken for training the Viola-Jones detection algorithm. The second dataset originated from a video sequence, showing slowly moving cars (less than residential speed) acquired from a pedestrian bridge. The extracted video frames showed license plates with a resolution of approximately 90×30. The video was captured on a day with alternating weather conditions (sunny, overcast, highly clouded), so the video contains significant illumination changes. For SVM training 2600 characters were extracted from the first dataset and labeled by hand. In fact, this is the most time consuming step in the whole system setup. The extraction, labeling and training step is necessary for all different plates originating from different countries as long as they include different font types.

4.3. License Plate Detector

Different classifiers are trained for the license plate detector. The quality is evaluated automatically on a labeled dataset using methods as introduced by Agarwal *et al*. [1]. Table 1 lists a set of classifiers and the achieved F-measures.

Edge orientation features achieved the same quality with less than half the number of features compared to a classifier using Haar-like features only. Nonetheless, features based on edge orientation histograms require too much memory in order to be applicable on embedded systems.

Finally, classifier *1* is used on the embedded device due to its low number of features. If the detector is evaluated after the subsequent segmentation, constraining the number

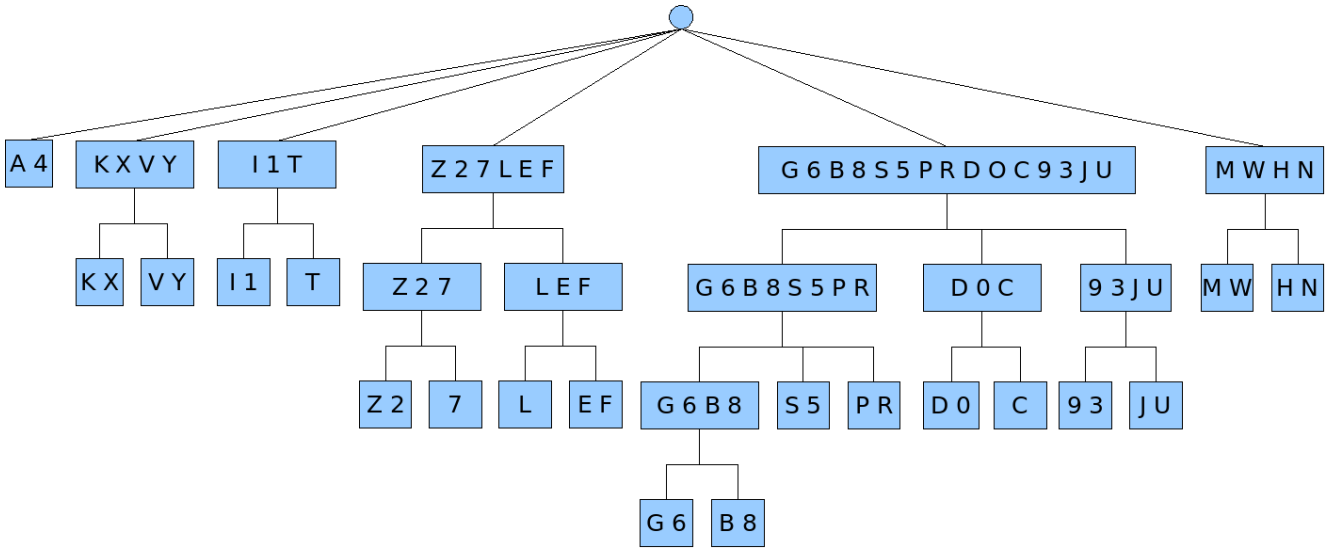

Figure 4. Illustrates the partition of classes as used in the tree-like structure.

ID	#stages	#Haar-ft	#EOH-ft	ISFP	stage dr	stage fp	max F-measure
1	4	14	0	Y	0.99	0.6	0.748718
2	10	37	0	N	0.99	0.5	0.839623
3	8	35	0	Y	0.99	0.5	0.862245
4	8	8	9	N	0.99	0.5	0.864078

Table 1. Overview of the classifiers trained and tested. Columns *Haar-ft* and *EOH-ft*, respectively, list the number of used features of the two categories. *ISFP* indicates whether the classifier was trained with inter stage feature propagation or not. The next two columns list the per stage *detection rate* and *false positive* rate, which are parameters of the training process as well. The maximum *F-measure* that was achieved during evaluation is listed last.

of possible character regions, a high amount of false positives can be discarded, achieving a F-measure of 0.91 for this classifier.

The detector is trained with plates scaled to a resolution of 60×20. Although this resolution hardly allows for robust character recognition, it supports the idea of possibly running the detector at lower resolutions than the subsequent character recognition stage, hence improving the performance of the detector.

4.4. License Plate Character Recognition

For training the support vector machine 2600 characters are used, scaled to a patch size of 6×9, hence resulting in a 54-dimensional feature vector. This resolution achieves the best trade-off between performance and quality. Tests were performed with 8×12 and 10×15 patches as well. A support vector machine trained on those samples resulted only in a slightly better quality (less than 1%). This shows, that with our test-sets an increase of the patch-size does not improve the classification result. The quality limiting factor can be found in the character segmentation step at such low resolutions. The lower the resolution the harder becomes the accurate isolation of individual characters.

Table 2 compares the two different multi-class approaches on the digital camera (1) and the video data (2). The results are determined on a general purpose computer. As expected, the OneVsAll approach achieves higher classification results. The column invalid check indicates, how many plates are correctly detected as being incorrect.

As already described, subsequent classification results are combined on the embedded system. Figure 6 illustrates how the quality improves as more plates are considered for majority voting. Sequences of license plates were chosen having at least 10 license plates each. Note, that a direct comparison to table 2 is not fair, since the input data was converted twice from digital to analog and back (due to the usage of an analog video interface), adding additional noise to the video frames.

4.5. Performance

Table 3 lists performance measurements determined with classifier 1 and support vector classification using the tree

type	test set	post-process level	per char	per plate	invalid check
OneVsAll	1	1	98%	89%	11%
OneVsAll	1	2	98%	96%	60%
OneVsAll	2	1	94%	78%	8%
OneVsAll	2	2	94%	85%	30%
tree	1	1	97%	89%	18%
tree	1	2	97%	94%	61%
tree	2	1	88%	66%	4%
tree	2	2	89%	77%	13%

Table 2. Classification results. Invalid check indicates how many invalid plates are correctly detected as being incorrect.

Figure 5. Testdata. The first set was acquired with a digital camera. For training the detector, those plates were scaled down to a resolution of 60×20. Below, sample frames from the video stream are depicted.

multi-class structure. Note, that Viola and Jones dynamic search, segmentation, and classification are only considered for the measurement, if they are invoked on a license plate effectively. Otherwise, the average time would drop down to zero.

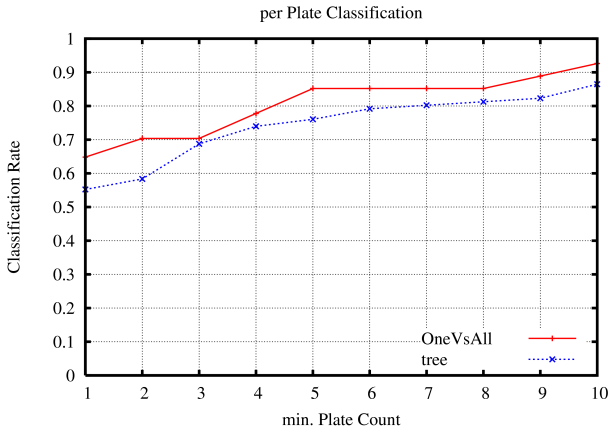

Figure 6. Plate quality after majority voting. As can easily be seen, the recognition performance is improved as classification results from more subsequent frames are incorporated.

On average, the support vector machine takes 7.30 ms for the classification of a full plate. If assuming the number of characters per license plate is seven, the classification process per characters takes approximately 1 ms. Classification using the OneVsAll support vector machine takes on average 20.17 ms, hence requiring approximately 2.88 ms per character.

The time per frame is measured over any frame, whether it contains a license plate or not. Therefore, this value has a high standard deviation. The achieved average frame rate is 19 fps. A closer look at the timing results reveals that the plate detection module takes much more time than the recognition module. Therefore it is advisable to further optimize this part of the system to free resources for the recognition engine. Strengthening the support vector classification algorithm would result in an improvement of the recognition performance.

5. Conclusion

In this paper we have presented a real-time license plate detection and recognition system. The system operates on

Operation	avg. time [ms]	std. dev.
Image acquisition	2.64	0.00
Integral Image computation	3.22	0.53
VJ static	26.72	1.38
VJ dynamic	6.77	1.58
KalmanTracker update	0.18	0.09
Segmentation	1.82	0.65
Classification	7.30	2.64
JPEG compression/sending	10.65	0.26
Message creation/sending	0.11	0.18
per frame	52.11	11.89

Table 3. Performance measurements of *critical functions* on the DSP using the tree-like classifier for character recognition.

image frames acquired with standard video equipment without any additional sensor input. The high performance of the system allows for compensating the low image resolution by considering the classification results of subsequent frames.

Due to the complete integration on an embedded device, the system operates autonomously, reporting only the final classification results to connected operators. Self-evidently, all advantages and capabilities of distributed systems apply here as well.

We are currently working on an optimization to the detection algorithm implementation; our goal is to adapt the algorithm to better fit to the architectural characteristics of our platform. In the future we will also have a look at special region segmentation methods as the one proposed in [13] and examine their applicability on our hardware.

References

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, 2002.

[2] C. Arth, C. Leistner, and H. Bischof. TRIcam: An Embedded Platform for Remote Traffic Surveillance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (Embedded Computer Vision Workshop)*, 2006.

[3] N. Bellas, S. M. Chai, M. Dwyer, and D. Linzmeier. FPGA implementation of a license plate recognition soc using automatically generated streaming accelerators. *In 20th International Parallel and Distributed Processing Symposium (IPDPS)*, page 8 pp., 2006.

[4] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[5] X. Chen and A. L. Yuille. Detecting and reading text in natural scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–366–II–373, 2004.

[6] L. Dlagnekov and S. Belongie. Recognizing cars. Technical Report CS2005-0833, UCSD University of California, San Diego, 2005.

[7] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, second edition, 2000.

[8] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1995.

[9] R. E. Kalman. A new approach to linear filtering and prediction problems. In *Transactions of the ASME - Journal of Basic Engineering*, volume 82, pages 35–45, 1960.

[10] V. Kamat and S. Ganesan. An efficient implementation of the hough transform for detecting vehicle license plates using dsp's. In *Proceedings of the Real-Time Technology and Applications Symposium (RTAS)*, page 58. IEEE Computer Society, 1995.

[11] J. Kang, M. Kang, C. Park, J. Kim, and Y. Choi. Implementation of embedded system for vehicle tracking and license plates recognition using spatial relative distance. *In 26th International Conference on Information Technology Interfaces (ITI)*, 1:167– 172, 2004.

[12] K. Levi and Y. Weiss. Learning object detection from a small number of examples: the importance of good features. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2:53–60, 2004.

[13] J. Matas and K. Zimmermann. Unconstrained licence plate and text localization and recognition. In *Proceedings of the IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 572–577, Vienna, Austria, September 2005.

[14] C. Rahman, W. Badawy, and A. Radmanesh. A real time vehicle's license plate recognition system. In *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 163–166, 2003.

[15] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[16] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[17] V. Shapiro, G. Gluhchev, and D. Dimov. Towards a multinational car license plate recognition system. *Machine Vision and Applications*, 17(3):173–183, August 2006.

[18] J. Sochman and J. Matas. Inter-stage feature propagation in cascade building with adaboost. *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, pages 236–239, 2004.

[19] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004.

[20] X. H. W. Jia, H. Zhang and M. Piccardi. Mean shift for accurate license plate localization. In *Proceedings of the IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 566–571, 2005.