Unconstrained License Plate Detection in Hardware

Petr Musil[a], Roman Juránek[b] and Pavel Zemčík[c]
FIT, Brno University of Technology, Brno, Czech Republic

Keywords: ALPR, Soft Cascade, Decision Trees, WaldBoost.

Abstract: In this paper, we propose an FPGA implementation of license plate detection (LPD) in images captured by arbitrarily placed cameras, vehicle-mounted cameras, or even handheld cameras. In such images, the license plates can appear in a wide variety of positions and angles. Thus we cannot rely on a-priori known geometric properties of the license plates as many contemporary applications do. Unlike the existing solutions targeted for DSP, FPGA or similar low power devices, we do not make any assumptions about license plate size and orientation in the image. We use multiple sliding window detectors based on simple image features, each tuned to a specific range of projections. On a dataset captured by a camera mounted on a vehicle, we show that detection rate is 98 % (and 98.7 % when combined with video tracking). We demonstrate that our FPGA implementation can process 1280×1024 pixel image at over 40 FPS with a minimum width of detected license plates approximately 100 pixels. The FPGA block is fully functional and it is intended to be used in a smart camera to parking control in residential zones.

1 INTRODUCTION

License plate detection (LPD) is an essential part of applications, such as detection of vehicles for traffic monitoring and enforcement purposes, or as a basis for automatic license plate recognition. In many industrial use cases, LPD is solved at a sufficiently advanced level in scenarios with restricted size, orientation, and in environment with controlled illumination (e.g. using IR flashes). A vast number of research papers has addressed this controlled scenario, and it is far beyond the scope of this paper to thoroughly review them.

In this paper, we focus on license plate detection implemented in FPGA in an image captured by *arbitrarily placed cameras*, *vehicle-mounted cameras*, or even *handheld cameras*. The target applications include parking control in residential zones, detection of stolen vehicles, and other applications, in which license plates of cars are automatically detected and recognized. Our goal was to improve the performance of license plate detection process, to implement the method in FPGA, and to provide the solution that requires low resources and that is suitable for integration into hardware devices, or even directly into cameras. The license plates captured in the image are distorted from their original rectangular shape by perspective projection, see Figure 1 for a few examples. Due to the character of the data, the detection and localization can not rely on an a-priori information, such as license plate size or orientation in the image. Moreover, lighting conditions, ranging from

Figure 1: We detect license plates in wide variety of deformations in challenging light conditions. The images show the detected license plates with a confidence value.

[a] https://orcid.org/0000-0001-9766-4759
[b] https://orcid.org/0000-0003-0589-0172
[c] https://orcid.org/0000-0001-7969-5877

Musil, P., Juránek, R. and Zemčík, P.
Unconstrained License Plate Detection in Hardware.
DOI: 10.5220/0010174000130021
In *Proceedings of the 7th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2021)*, pages 13-21
ISBN: 978-989-758-513-5

Table 1: Summary of properties of state of the art detection algorithms. The accuracies reported by the works are not directly comparable as the methods were evaluated on different data and by different testing protocols.

	Rotation	Candidate search	Detection	Platform	Accuracy
(Mai et al., 2011)	tilt	Vertical gradient detection	Character recognition	PC	98 %
(Yang et al., 2011)		Chroma and gradient filtering	SVM	FPGA/PC	N/A
(Zhai et al., 2011)		Edge detection, CCA	Filtering by size, orientation and aspect ratio	FPGA	99 %
(Jeffrey and Ramalingam, 2012)		Edge and background detection, CCA	Gradient statistics	PC	97 %
(Khalil and Kurniawan, 2014)		Edge detection	SVM	PC	88 %
(Biyabani et al., 2015)		Edge detection	Segmentation with projections, template matching	FPGA	84 %
(Ha and Shakeri, 2016)		Edge and background detection	Character recognition with template matching	PC	84 %
(Chhabra et al., 2016)		Segmentation with edge detection	Filtering by size and aspect ratio	FPGA	99 %
(Elbamby et al., 2016)		Edge detection	LBP cascade	PC	94 %
(Yuan et al., 2017)		Adaptive threshold and edge detection	SVM	PC	96 %
(Xie et al., 2018)	Free	–	CNN–YOLO	PC/GPU	99.5 %
(Silva and Jung, 2018)	Free	–	CNN	PC/GPU	98.35 %
(Sborz et al., 2019)		Morphologic operation, CCA	Character segmentation	FPGA	–
(Yousefi et al., 2019)		Background subtraction	LBP cascade	PC	98 %
(Chen and Wang, 2020)	Free	–	CNN + AdaBoos cascade	PC/GPU	99.9 %
(Gao et al., 2020)	Free	–	CNN	PC/GPU	99.9 %
(WU et al., 2020)		corner detection and morphological op.	CNN	PC/GPU	97.2 %
This work	**Free**	–	**soft cascade with LBP**	**FPGA**	**98 %**

deep shadows to overexposures, prevents using edge detection, gradient-based methods, and segmentation-based methods.

2 RELATED WORK

An overview of several works on license plate detection with a focus on those implemented in embedded systems or PC is shown in Table 1. Many of the works are intended for stationary cameras and applications where the rough location and size of license plate is known (Biyabani et al., 2015; Ha and Shakeri, 2016; Jeffrey and Ramalingam, 2012; Zhai et al., 2011; Chhabra et al., 2016; Khalil and Kurniawan, 2014; Yang et al., 2011; Yuan et al., 2017; Elbamby et al., 2016; Hsu et al., 2013). In many cases, they rely on the presence of the license plate edges to reduce search space.

Many methods (Anagnostopoulos et al., 2006; Zhai et al., 2011; Jeffrey and Ramalingam, 2012) are based on connected component analysis (CCA) for the search of the candidate components in binarized images. (Zhai et al., 2011) uses CCA with morphological operators and filters the components using knowledge about aspect ratio, size, and orientation. (Anagnostopoulos et al., 2006) segments characters within the CCA components by horizontal and vertical projections. (Mai et al., 2011) uses edge-based approach and rotation-free character recognition to search for tilted license plates. (Wang and Lee, 2003) uses CCA in combination with Radon transform to search for components with license plate properties (size, aspect ratio). (Sborz et al., 2019) proposed license plate localization on FPGA. They use two morphological operations and connected components labelling technique to find license plate candidates.

Template matching-based methods (Ha and Shakeri, 2016; Biyabani et al., 2015) use background subtraction and edge detection for candidate search. Candidates are then scanned for characters using template matching. Other methods (Chhabra et al., 2016; Biyabani et al., 2015) use template matching to find license plate candidates, and searches for characters in horizontal and vertical projections of the license plate images.

Machine learning based methods for licence plate detection typically use either SVM (Khalil and Kurniawan, 2014; Yang et al., 2011; Yuan et al., 2017), cascade classifiers (Elbamby et al., 2016; Arth et al., 2006) or Convolutional neural network(CNN)(Xie et al., 2018; Chen and Wang, 2020; Gao et al., 2020; WU et al., 2020). (Khalil and Kurniawan, 2014) searches for candidate locations by horizontal and vertical edge detection and classifies them by SVM with co-occurrence matrix features. (Yang et al., 2011) presented a hybrid FPGA-PC approach where an image is filtered by chroma filter and gradient filter on FPGA, and then SVM is applied on candidate positions on PC. (Yuan et al., 2017) also proposed adaptive thresholding, density filter and SVM with colour saliency features.

(Arth et al., 2006) proposes a Haar cascade detector on DSP for detection of cars and license plates. (Elbamby et al., 2016) uses cascade classifier with Local Binary Patterns (LBP) features and preprocesses the image by edge detection and morphology filters. (Yousefi et al., 2019) use AdaBoost cascade with LBP features for license plate detection and background subtraction for non-moving areas elimination. They use linear regression for estimation of size of detected plates. (Xie et al., 2018) use YOLO (You only look once) method (Redmon et al., 2015) based on CNN to multi-directional license plate detection. (Chen and Wang, 2020) proposed a method combing the CNN with broad learning system based

Figure 2: **(left)** Image scales and objects detected by the classifiers. **(right)** Final detections after non-maximum suppression. The classifiers are color-coded, the yellow one detects license plates almost aligned, the green one detects slightly tilted license plates.

on AdaBoost for license plate recognition. (Gao et al., 2020) created CNN based encoder-decoder system where encoder detects candidate license plate characters and recognises them without considering the format of the license plate. (WU et al., 2020) uses colour segmentation, corner detection and morphological operations to select the operating area, and finally uses a convolutional neural network to get license plate area.

Generic object detector hardware architectures have also been published, mostly demonstrated on face detection (Cho et al., 2009; Kyrkou and Theocharides, 2011; Zemcik et al., 2013; Said and Atri, 2016; Musil et al., 2020), that could be applied for detection of license plates. In most cases, they implement Haar Cascades (Viola and Jones, 2004) or SVM (Dalal and Triggs, 2005).

In this work, similarly to (Arth et al., 2006; Elbamby et al., 2016), we propose statistical sliding window detectors trained by the machine learning algorithm. Such an approach is more robust than pure image processing, such as in (Anagnostopoulos et al., 2006; Zhai et al., 2011), since it easily adapts to visual variability of data. The difference in our approach from the other ones is that we use multiple detectors tuned to various deformations of license plates in order to boost the quality of detection. We use a large artificially generated dataset of training examples and propose an FPGA implementation of the detection process on Xilinx Zynq platform. Our architecture is based on (Musil et al., 2020) which provides excellent speed/resource tradeoff. We evaluated the method on a large image and video database obtained from industry.

To our knowledge, this work is the first to use boosted classifiers for unconstrained detection of license plates directly in FPGA. Other works that expoit FPGA use mostly segmentation or filtering approaches which are targeted for specific scenarios (and usually fail in unconstraned detection). Recent works employ mostly neural networks which are superior in accuracy of detection but their implementation in FPGA is limited and expensive (Nguyen et al., 2019; Wu et al., 2019) or they require specific hardware (GPU, VPU) (Xie et al., 2018; Chen and Wang, 2020; Gao et al., 2020; WU et al., 2020).

3 LICENSE PLATE LOCALIZATION

Our task is to detect license plates observed by a camera and to localize them; see Figure 1 for example images. The intended applications require high accuracy of the detection result. At the same time, it can tolerate a reasonable amount of false detections (which can be filtered out later, for example, by the license plate recognition process). Our license plate detection algorithm is based on multiple independent boosted classifiers, whose results are merged to get the final results. Each of the classifiers captures license plates with a specific range of observed in-plane rotations. We use constant soft cascade classifiers (Šochman and Matas, 2005; Dollár et al., 2014) with Local Binary Patterns (LBP) features (Zhang et al., 2007; Zemcik et al., 2013). The detection process is illustrated in Figure 2,

3.1 The Classifier

The classifier is a function $H(\mathbf{x})$ which gives the confidence value for an image patch $\mathbf{x}$, formally defined in Equation (1). During the detection process, every location of the input image is analyzed by the classifier. Multi-scale detection is solved by image scaling by a fixed factor. The classifier $H(\mathbf{x})$, is a sequence of T weak classification functions ($T = 512$ in our experiments), and its response on image window $\mathbf{x}$ is a sum of predictions produced by the individual weak classifiers. We use simple weak classifiers based on

Table 2: Ranges of orientation and window sizes for three classifiers.

Range [°]	Window [px]	LP images
$0 < \phi < 15$	18×64	
$10 < \phi < 30$	22×56	
$25 < \phi < 45$	28×43	

LBP features f and lookup tables A with the confidence predictions.

$$H_t(\mathbf{x}) = \sum_{i=1}^{t} A_i(f_i(\mathbf{x})) \quad (1)$$

The detection process on an image $\mathbf{I}$ produces a set of locations and sizes that were not rejected by the classifier, Equation (2). Each candidate comprises of its location x, y in the image, size w, h and confidence score c (the classifier response on the image patch corresponding to the location).

$$H(\mathbf{I}) = \{(x, y, w, h, c)\} \quad (2)$$

An important property of soft cascade classifiers is that the classification function $H(\mathbf{x})$ can be terminated after evaluating k-th weak classifier, when $H_k(\mathbf{x}) < \theta_k$. Thresholds θ are trained, so that majority of background samples is rejected early in the process. The computational complexity of the classifier dramatically decreases compared to the case of evaluation of all T weak classifiers, and it can be evaluated as an average number of weak classifiers evaluated per image position nf (which is usually orders of magnitude lower that T). This value is especially important as it directly influences the speed of the detector in the FPGA implementation.

3.2 Detection with Multiple Classifiers

We propose to use n classifiers $\mathcal{H} = \left\{H^{(1)}, \ldots, H^{(n)}\right\}$ for the localization of license plates, each tuned for LPs with specific range of rotation angle ϕ. Each classifier produces detections independently, and the final set of candidates is obtained as a union of all candidates (3). Final locations are produced by a simple, overlap-based non-maxima suppression algorithm.

$$\mathcal{H}(\mathbf{I}) = \bigcup_i H^{(i)}(\mathbf{I}) \quad (3)$$

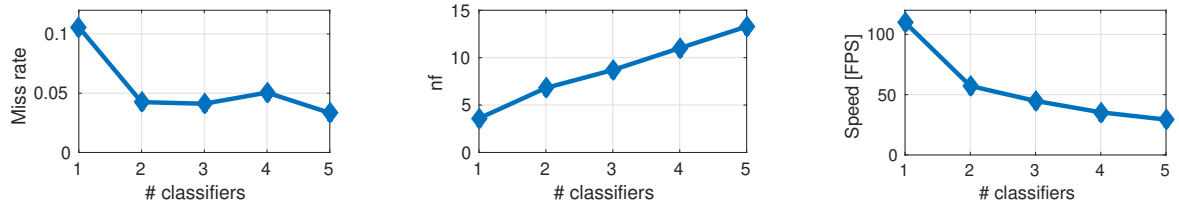
Each detector is trained for a specific range of license plate angles ϕ. The range assigned to the k-th classifier is,

$$\max\left(\frac{45(k-1)}{n} - 5, 0\right) < \phi < \frac{45k}{n},$$

where 45° marks the upper limit of license plate orientations. The classifier window aspect ratio is set as a mean aspect ratio of license plates falling in the range ϕ. The window size is set to constant area of 1024 pixels and 2 pixel margin is added. The ranges and window sizes for $n = 3$ classifiers are summarized in Table 2.

The number of classifiers in the ensemble $\mathcal{H}$ influences the accuracy and speed of the detection process. One classifier can not capture all the variability of the license plates, while more detectors are more accurate but slower. This trend is shown in Figure 3. Two classifiers give already reasonable detection rate and sufficient speed. We observed that using more classifiers results in better localization. Using more classifiers, however, reduces speed, and for this reason, we use three classifiers in our hardware implementation (Section 4).

The training data we used is a broad set of license plate images randomly transformed to match deformations likely to occur in the target application, since the scenario is often known and fixed, e.g. in case of our vehicle-mounted camera. Few samples are shown in Table 2. The advantage of such an approach is that it can be automated and any number of training samples with precise ground truth (including orientation) can be quite effortlessly generated for each application case.



(a) (b) (c)

Figure 3: Effect of the number of the classifiers on the detection. **((a))** Reduced miss rate, **((b))** higher computational complexity and **((c))** lower frame rate. We choose to use 3 classifiers which is a good balance between detection accuracy, speed and hardware requirements (see Section 4).

4 FPGA ARCHITECTURE

We implemented the LP detector on our hardware camera platform based on SoC Xilinx Zynq XC 7Z020. The SoC contains two ARM CortexA9 cores and FPGA interconnected through high-speed AMBA AXI bus. The approach is, however, quite general and can be applied to almost any Xilinx Zynq family member and also to other platforms. In our case, the platform is equipped with a low noise global shutter CMOS image sensor Python1300 from ON Semiconductor, which is attached directly to FPGA, forming a smart camera. The sensor resolution is 1280×1024 pixels and it can capture up to 210 FPS.

Figure 4 shows the block diagram of the camera with the detector integrated into it. As the minimum size of the license plate needed to detect is approximately 100 pixels in width, we downscale the input image to half resolution before the detection phase.

The image is passed to the detector block through balancing FIFO that covers irregularities in the detector speed that depend on the image content and that are hard to predict. The results of detection – locations of detected objects – are transmitted to ARM CPU memory, along with the original image. On the CPU, the detected license plates are tracked using the Kalman filter. The camera outputs cropped license plate images that are transmitted to the server for further processing (OCR, storage, etc.).

The Detector block is based on (Musil et al., 2020) which implements multi-scale soft cascade detector with LBP (Zhang et al., 2007) or LRD features (Hradiš et al., 2008). The engine works as a programmable automaton, using a sequence of feature parameters as instructions. We modified the engine in order to use three classifiers and adapted it to a more recent platform with more resources and memory.

The engine works without external memory, storing only a narrow stripe of the image which is being stored and analyzed directly in BRAM inside the FPGA. The stripe memory size is 4096×32 pixels in 32 BRAMs organized in a way that data for a feature evaluation can be obtained in one clock cycle. The stripe memory is filled from the CMOS, and the scaled versions of the image are created on-the-fly; the process is illustrated in Figure 4.

Table 3: FPGA resource utilization.

	BRAM	LUT	REG
Image acquisition	5	2559	4746
Balancing FIFO	4	115	210
Detector	38	9521	7099
Storage	5	3734	4785
Total	**52**	**15938**	**16840**
7Z020 res.	**37 %**	**30 %**	**16 %**

The classification function is executed overall positions in the stripe memory. The processing is heavily pipelined, we use two pipelines of length 9, so up to 18 positions are processed in parallel. The efficiency of feature extraction is $np = 1.75$ features extracted in clock cycle. Overall performance in frames per second, expressed by equation (4), is, in general, dependent on the clock frequency f and the total number of features that must be calculated on an image for all classifiers (i.e. the number of positions P times the average number of features nf for each individual classifier).

$$F = \frac{f \cdot np}{\sum_{i=1}^{k} P_i \cdot nf_i} \tag{4}$$

In this work we assume $f = 200$MHz; $P_1 = 872487$, $P_2 = 899067$, and $P_3 = 915442$; and $nf_1 = 2.89$, $nf_2 = 3.23$, and $nf_3 = 2.78$. The values of nf were estimated on a testing set of 1522 images and are valid for the experimental classifiers presented in the Section 5. The estimated upper limit of performance of the detection unit is therefore $F = 48$ FPS. Table 3 summarizes resources required by the design. The whole solution takes approximately one third of resources of the 7Z020 FPGA which we use in our solution.

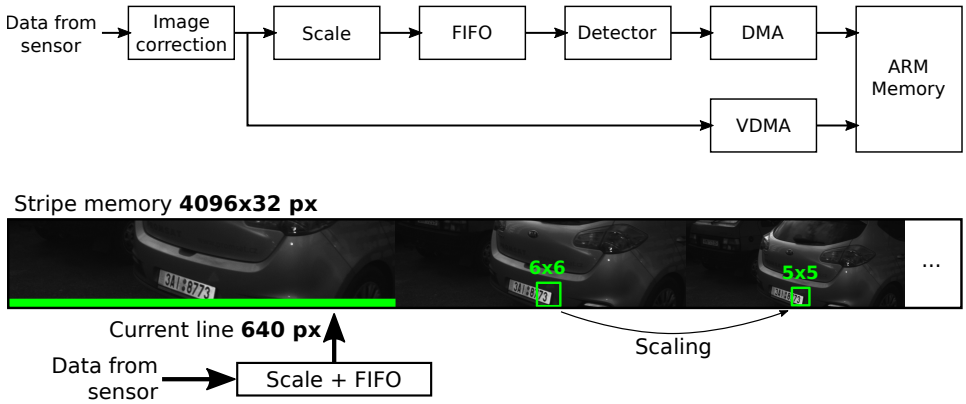


Figure 4: **(Top)** Block scheme of the camera prototype. **(Bottom)** The data from sensor are passed to a narrow image stripe in memory. All scales are created automatically.

5 DETECTOR EVALUATION

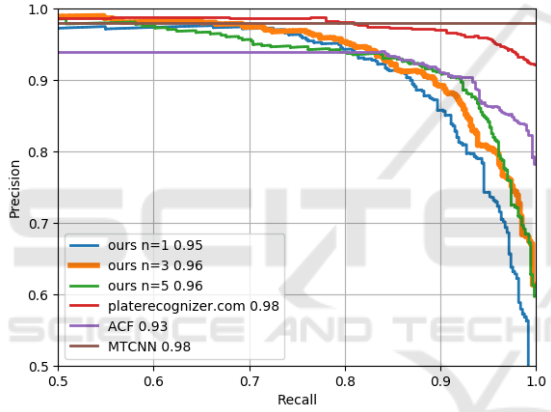


Figure 5: Recall-Precision characteristics for the proposed method (for different n) and comparison with other methods.

We evaluated the proposed method in two modes – *Single image* mode and *Tracking* mode.

In the *Single image* mode we evaluate precision-recall trade-off on a large number of testing images. However, it is impossible to compare to other published license plate detection methods (summarized in Table 1) since they published results on different testing datasets using different testing protocols. Moreover, none of them has available source code. For this reasons, we compare our method to well known general object detection methods – ACF (Dollár et al., 2014) and MTCNN (Zhang et al., 2016), and a commercial solution Plate Recognizer[1]. It must be however noted that none of them is targeted for FPGA with all of its constraints (low memory, integer arithmetic, etc.). MTCNN is a recent method based on neural networks and it require PC/GPU platform. Plate Recognizer internal structure was not published. ACF is a method similar to ours since it shares a common classifier structure and the detection algorithm. For the evaluation, we used our internal dataset of 1 522 images with 811 manually annotated license plate bounding boxes. The results are summarized in Figure 5. Not surprisingly, MTCNN and the commercial solution give almost perfect results. ACF is comparable to our method in terms of recall but with higher precision. The results for our method show that a single classifier already gives reasonable results and adding more classifiers tuned to different transformations further improves them (see also Figure 3). Our method can reach recall over 95% with precision around 0.8, and recall 98% with precision 0.7 which is sufficient for real-world applications. In the *Tracking* mode, we detect and track LPs in video sequences and evaluate the detection rate. We used a simple tracker based on Kalman filter. In the evaluation, we required each license plate to be hit at least once in sequence. From the application point of view, the *Tracking* mode is more important. In the test, we use six sequences, each approximately 10 minutes long taken at ten frames per second. The tracking accuracy is summarized in Table 4. The total detec-

Table 4: Tracking evaluation on testing sequences.

Seq.	#tracked	#missed	#false	Det. rate
1	197	1	59	0.995
2	103	1	43	0.990
3	101	5	15	0.953
4	181	1	26	0.995
5	205	3	23	0.986
6	53	0	2	1.000
Total	**840**	**11**	**168**	**0.987**

[1]www.platerecognizer.com provides free REST API

Figure 6: **(top)** Examples of detected license plates and detector responses. **(bottom)** Missed license plates (annotations marked by red).

tion rate 98.7 %. Missed license plates are in most cases, those with extreme deformations or very dirty, false alarms are license plate-like patterns in the image (various signs, etc.), see Figure 6 for a few examples.

6 CONCLUSIONS

We presented a method for reliable real-time detection and localization of license plates observed from a moving vehicle, and its FPGA implementation suitable for integration into a smart camera. Our solution is unique, since it uses machine learning-based detection method, it does not require external memory, and enables for real-time frame rates on low-end FPGA. In this paper, we focused on the detection of license plates. However, the method is more general and can be adapted for detection of other objects too. The proposed solution is capable of handling complex projections of the license plates, such as deformations caused by perspective projection, scaling, and rotations. The proposed solution uses multiple, in our case three, classifiers to ensure the quality of the results while keeping the performance high. The classifiers are based on simple LBP features which makes it easy to implement them in FPGA. We demonstrated that our hardware solution could process 1280×1024 pixel frames at over 40 FPS while taking only a fraction of resources of low-end FPGA (Zynq Z7020). The accuracy of detection measured on the real-world data is 98 % and 98.7 % when combined with tracking, while producing only a modest amount of false detections.

Future work includes further efficiency improvements, investigation of dependency of quality on freedom of projection of images and sampling of the image frames, and possibly also exploitation of other features and detection mechanisms including hybrid approach with neural networks.

ACKNOWLEDGEMENTS

This work is part of the FitOptiVis project funded by the ECSEL Joint Undertaking under grant number H2020-ECSEL-2017-2-783162.

REFERENCES

Anagnostopoulos, C. N. E., Anagnostopoulos, I. E., Loumos, V., and Kayafas, E. (2006). A license plate-recognition algorithm for intelligent transportation system applications. *IEEE Transactions on Intelligent Transportation Systems*, 7(3):377–392.

Arth, C., Bischof, H., and Leistner, C. (2006). TRICam – an embedded platform for remote traffic surveillance. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pages 125–125.

Biyabani, A. A., Al-Salman, S. A., and Alkhalaf, K. S. (2015). Embedded real-time bilingual alpr. In *Communications, Signal Processing, and their Applications (ICCSPA)*.

Chen, C. L. P. and Wang, B. (2020). Random-positioned license plate recognition using hybrid broad learning system and convolutional networks. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–13.

Chhabra, S., Jain, H., and Saini, S. (2016). Fpga based hardware implementation of automatic vehicle license plate detection system. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1181–1187.

Cho, J., Mirzaei, S., Oberg, J., and Kastner, R. (2009). Fpga-based face detection system using haar classifiers. In *FPGA*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*.

Dollár, P., Appel, R., Belongie, S., and Perona, P. (2014). Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545.

Elbamby, A., Hemayed, E. E., Helal, D., and Rehan, M. (2016). Real-time automatic multi-style license plate detection in videos. In *Computer Engineering Conference (ICENCO)*.

Gao, F., Cai, Y., Ge, Y., and Lu, S. (2020). Edf-lpr: a new encoder decoder framework for license plate recognition. *IET Intelligent Transport Systems*, 14(8):959–969.

Ha, P. S. and Shakeri, M. (2016). License plate automatic recognition based on edge detection. In *2016 Artificial Intelligence and Robotics (IRANOPEN)*, pages 170–174.

Hradiš, M., Herout, A., and Zemcik, P. (2008). Local rank patterns - novel features for rapid object detection. In *Proceedings of International Conference on Computer Vision and Graphics 2008*, Lecture Notes in Computer Science, pages 1–2.

Hsu, G., Chen, J., and Chung, Y. (2013). Application-oriented license plate recognition. *IEEE Transactions on Vehicular Technology*, 62(2):552–561.

Jeffrey, Z. and Ramalingam, S. (2012). High definition licence plate detection algorithm. In *Southeastcon*.

Khalil, M. S. and Kurniawan, F. (2014). License plate detection method for real-time video of low-cost webcam based on hybrid svm-heuristic approach. In *Information Technology: New Generations (ITNG)*.

Kyrkou, C. and Theocharides, T. (2011). A flexible parallel hardware architecture for adaboost-based real-time object detection. In *VLSI Systems*.

Mai, V., Miao, D., Wang, R., and Zhang, H. (2011). An improved method for vietnam license plate location. In *2011 International Conference on Multimedia Technology*, pages 2942–2946.

Musil, P., Juránek, R., Musil, M., and Zemčík, P. (2020). Cascaded stripe memory engines for multi-scale object detection in fpga. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(1):267–280.

Nguyen, D. T., Nguyen, T. N., Kim, H., and Lee, H. (2019). A high-throughput and power-efficient fpga implementation of yolo cnn for object detection. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(8):1861–1873.

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640.

Said, Y. and Atri, M. (2016). Efficient and high-performance pedestrian detector implementation for intelligent vehicles. *IET Intelligent Transport Systems*, 10(6):438–444.

Sborz, G. A. M., Pohl, G. A., Viel, F., and Zeferino, C. A. (2019). A custom processor for an fpga-based platform for automatic license plate recognition. In *2019 32nd Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–6.

Silva, S. M. and Jung, C. R. (2018). License plate detection and recognition in unconstrained scenarios. In *2018 European Conference on Computer Vision (ECCV)*, pages 580–596.

Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *IJCV*.

Šochman, J. and Matas, J. (2005). WaldBoost – learning for time constrained sequential detection. In *CVPR*.

Wang, S.-Z. and Lee, H.-J. (2003). Detection and recognition of license plate characters with different appearances. In *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, volume 2, pages 979–984 vol.2.

Wu, D., Zhang, Y., Jia, X., Tian, L., Li, T., Sui, L., Xie, D., and Shan, Y. (2019). A high-performance cnn processor based on fpga for mobilenets. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pages 136–143.

WU, X., QIU, J., and QIU, A. (2020). An efficient license plate location algorithm based on deep leaming. In *2020 International Conference on Computer Engineering and Application (ICCEA)*, pages 543–546.

Xie, L., Ahmad, T., Jin, L., Liu, Y., and Zhang, S. (2018). A new cnn-based method for multi-directional car license plate detection. *IEEE Transactions on Intelligent Transportation Systems*, 19(2):507–517.

Yang, S. Y., Lu, Y. C., Chen, L. Y., and Cherng, D. C. (2011). Hardware-accelerated vehicle license plate detection at high-definition image. In *2011 First International Conference on Robot, Vision and Signal Processing*, pages 106–109.

Yousefi, E., Nazem Deligani, A. H., Jafari Amirbandi, J., and Karimzadeh Kiskani, M. (2019). Real-time scale-invariant license plate detection using cascade classifiers. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 399–402.

Yuan, Y., Zou, W., Zhao, Y., Wang, X., Hu, X., and Komodakis, N. (2017). A robust and efficient approach to license plate detection. *IEEE Transactions on Image Processing*, 26:1102 – 1114.

Zemcik, P., Juranek, R., Musil, P., Musil, M., and Hradis, M. (2013). High performance architecture for object detection in streamed videos. In *Field Programmable Logic and Applications (FPL)*.

Zhai, X., Bensaali, F., and Ramalingam, S. (2011). Real-time license plate localisation on fpga. In *CVPR 2011 WORKSHOPS*, pages 14–19.

Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503.

Zhang, L., Chu, R., Xiang, S., Liao, S., and Li, S. Z. (2007). Face detection based on multi-block lbp representation. In *ICB*.