Adaptive Vehicle Detection for Real-time Autonomous Driving System

Maryam Hemmati
Dept. of Electrical &
Computer Engineering
University of Auckland
Auckland, New Zealand
m.hemmati@auckland.ac.nz

Morteza Biglari-Abhari
Dept. of Electrical &
Computer Engineering
University of Auckland
Auckland, New Zealand
m.abhari@auckland.ac.nz

Smail Niar
LAMIH/CNRS
Polytechnic University
Hauts-de-France
Valenciennes, France
smail.niar@uphf.fr

***Abstract*—Modern cars are being equipped with powerful computational resources for autonomous driving systems (ADS) as one of their major parts to provide safer travels on roads. High accuracy and real-time requirements of ADS are addressed by HW/SW co-design methodology which helps in offloading the computationally intensive tasks to the hardware part. However, the limited hardware resources could be a limiting factor in complicated systems. This paper presents a dynamically reconfigurable system for ADS which is capable of real-time vehicle and pedestrian detection. Our approach employs different methods of vehicle detection in different lighting conditions to achieve better results. A novel deep learning method is presented for detection of vehicles in the dark condition where the road light is very limited or unavailable. We present a partial reconfiguration (PR) controller which accelerates the reconfiguration process on Zynq SoC for seamless detection in real-time applications. By partially reconfiguring the vehicle detection block on Zynq SoC, resource requirements is maintained low enough to allow for the existence of other functionalities of ADS on hardware which could complete their tasks without any interruption. Our presented system is capable of detecting pedestrian and vehicles in different lighting conditions at the rate of 50fps (frames per second) for HDTV (1080x1920) frame.**

I. Introduction

Autonomous driving systems (ADS) are getting more popular as they provide more accurate detection and consequently, become more reliable. Robust and reliable detection requires several different circumstances to be taken into account within the detection algorithm, which results in more intensive computations. Considering the stringent real-time requirement of such systems, dedicated hardware accelerators with parallel and pipelined architecture seem to be an inevitable option. However, resource limitations in the implementation of parallel architectures could become an additional bottleneck during the system level design of ADS. On the other hand, many of the features available on these sophisticated systems are not employed all the time and in all different driving environments. As an instance, animal detection on the road could be a useful feature for ADS since, in some countryside roads, animals might appear and cross the road. However, this feature might not be used in most of the times when the driving area is limited to urban roads. Moreover, even driving in the same area does not guarantee the effectiveness of a unique algorithm, as the lighting condition is dynamically changing and is affecting the quality of the captured image through vision cameras. In these situations, a system with dynamic capabilities could be an advantage to maintain a rich set of features and overcome the resource constraints of the system.

We present a real-time adaptive detection system which partially reconfigures the vehicle detection module and employs the most suitable detection algorithm for various environmental conditions. Our system uses a machine learning approach and consists of a static part as well as a dynamically reconfigurable part. The static part which includes data capture and pedestrian detection continues its operation during the reconfiguration intervals and guarantees the real-time and safe behavior of the system which is very important in safety-critical systems such as ADS. The reconfigurable part is designed to detect the vehicle with the choice of three different algorithms, each suitable for a particular environmental lighting condition. An external signal which indicates the light intensity changes is considered to trigger the reconfiguration of the hardware accelerator on programmable logic (PL) which is accomplished by the processing system (PS) on Zynq SoC. Our contribution has two folds; first, we present a novel method for detection of vehicles in very dark environments which uses deep belief networks (DBN) to detect the presence of taillights in thresholded image. A selection of detected taillights based on their obtained size features and their distance are fed to a support vector machine (SVM) classifier to localize the vehicle. Second, we present a PR reconfiguration controller on the Zynq PL that transfers the partial bit frames from the PL DDR through the DMA engine to avoid any delay which is usually generated by the PS interconnect system during the reconfiguration process. It results in the speed up of more than 2.6 times for the reconfiguration throughput [1].

The rest of the paper is organized as follows. A review of state of the art driver assistance system and vehicle detection algorithms is provided in Section II. Section III presents the explanation of our method for vehicle detection in different conditions followed by the implementation of hardware accelerators. System-level implementation and reconfiguration process, as well as the static partition, and reconfigurable partition on FPGA are explained in Section IV. Concluding remarks are discussed in Section V.

II. Related Works

Several works have been done in the area of vehicle detection either for driver assistance system and autonomous vehicles [2]–[5] or surveillance cameras on the road [6]. Vehicle detection methods are mostly categorized into two types: detection during the day [7], [8] and detection during the night [5], [9]–[11]. Different detection algorithms have been developed for each category as the visually dominant features of the vehicle vary significantly for each of these circumstances. While most of the detection during night time relies upon the information of taillights, the methods used for daytime detection, focus on the visual appearance of the vehicle as well as features such as symmetry and shadow under the car. Several works have incorporated the tracking information for efficient detection [3]–[5].

Many research works have used the histogram of oriented gradients (HOG) features [12] followed by a classifier such as support vector machine (SVM) [13] for vehicle detection in normal light conditions. Kim et al. [8] presented a new feature named the position and intensity included histogram of oriented gradients (PIHOG) which compensates the information loss involved in the construction of a histogram with position information to improve the discriminative power in detecting on-road vehicles.

Several works have addressed the challenge of vehicle detection during the night. Chien et al. [14] used a combination of morphological and logical operations to extract the overexposed central region of taillights for proper segmentation and range estimation. Satzoda et al. [11] presented a nighttime vehicle detection method which involved both learned classifiers and explicit rules, to address the challenge of correct detection for complex road and ambient lighting conditions. Their method called vehicle detection using active-learning during night time (VeDANt) employs a modified form of active learning for training AdaBoost classifiers with Haar-like features using gray-level input images. Schamm et al. [10] applied perspective blob filters and paired the results to detect the rear or front lights and consequently locate the vehicles in front. O' Malley et al. [3] present a taillights pair detection and tracking system by introducing a hardware system comprised of standard vision camera and a complementary CMOS sensor and a color filter. Detection and tracking method by image segmentation of headlights and taillights based on automatic multilevel histogram thresholding, followed by a spatial clustering and tracking procedure was presented in [6] for traffic surveillance system during the night. Their proposed system was able to classify moving cars as well as motorbikes.

III. Vehicle Detection

Many vehicle detection approaches use the features that are extracted in a manner to minimize the lighting and luminance variance, however, they are all affected by the change of environmental conditions to some extent. This is because the vehicle itself is not a static object with regards to its appearance in different lighting conditions. Consequently, the detection method requires to be updated adaptively to maintain reasonable detection accuracy in all different situations.

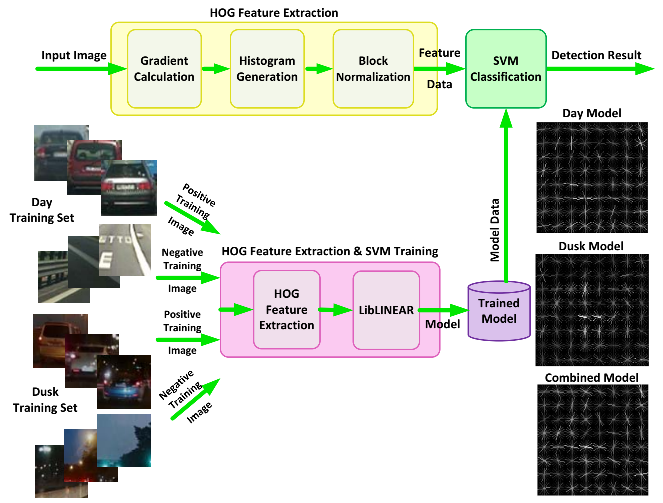

Fig. 1. Detection method by HOG+SVM during daylight and dusk conditions. Trained models are used as an input to the classifier during the detection.

We propose an adaptive vehicle detection approach where the algorithm is changed dynamically based on the object variations. Mostly, the vehicle detection is categorized into two different conditions of light and dark. We consider the third situation of low light where the visual appearance of the object changes extensively, but the vehicle shape features could yet be considered as an identifying feature and hence could be considered in the classification stage. We name these three different conditions as day, dusk, and dark. We present a novel algorithm based on machine learning to detect and match the taillights of vehicles in the dark. We employ the well known HOG+SVM detection method for the other two circumstances; however, we use two different trained classifiers for each of these scenarios.

A. Detection During Day and Dusk

During the day and in normal lighting conditions the edge and shape boundaries of vehicles are good candidates to be considered as the identifying feature. In moderate lighting conditions, when the environmental light decreases to some extent, the visual features of vehicles are slightly changed as the boundaries are not as sharp as they are in light environment. Moreover, as the drivers turn on their headlights, a new visual feature is added to the vehicle appearance which could be a good identifying feature in detection. Due to the presence of road light, the objects both in far and near are reasonably visible, and the edge and shape boundaries of vehicles are yet informative enough to be considered as a feature. In this scenario, it is more promising to consider both the vehicle boundaries and its taillights features to achieve more robust detection.

Figure 1 shows the training and test procedures where HOG features of an input image is extracted and classified

TABLE I
DETECTION ACCURACY AND NUMBER OF TRUE POSITIVES (TP), TRUE NEGATIVES (TN), FALSE POSITIVES (FP), AND FALSE NEGATIVES (FN) FOR TWO TEST SCENARIOS BY THREE DIFFERENT MODELS. TEST IMAGES OF UPM DATASET [15] AND SYSU DATASET [4] ARE USED FOR TEST OF DAY AND DUSK CONDITION RESPECTIVELY.

	Day test with UPM dataset					Dusk test with SYSU dataset					Dusk test with subset of SYSU dataset				
SVM Model	**Accuracy**	**TP**	**TN**	**FP**	**FN**	**Accuracy**	**TP**	**TN**	**FP**	**FN**	**Accuracy**	**TP**	**TN**	**FP**	**FN**
Day	96.00%	195	21	4	5	73.78%	659	680	72	404	77.55%	650	680	72	313
Dusk	20.89%	23	24	1	177	82.37%	744	751	1	319	86.88%	739	751	1	224
Combined	91.56%	185	21	4	15	85.34%	809	740	12	254	90.09%	805	740	12	158

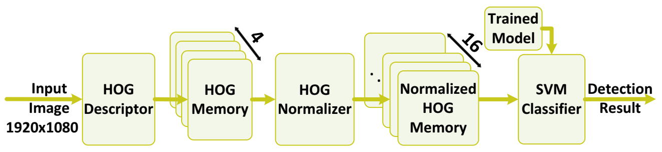

Fig. 2. Pipeline of implemented vehicle detection in day and dusk.

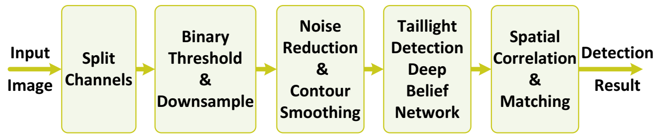
Input Image
Split Channels
Binary Threshold & Downsample
Noise Reduction & Contour Smoothing
Taillight Detection Deep Belief Network
Spatial Correlation & Matching
Detection Result

Fig. 3. Block Diagram of Vehicle Detection in Dark.

against a pre-trained model. SVM model is created during the training phase, where HOG features are extracted for all the training images of a dataset. Training images are divided into two sets of positive and negative, where positive images are those including the vehicles and negative images are those without it. Extracted features with their corresponding positive or negative label are then fed to the SVM training module, i.e. LibLINEAR [16], to generate an SVM model. It is noticeable that the trained model in these three cases look very different.

Since we are looking at two different scenarios of day and dusk, two different datasets have been chosen to train separate models. UPM vehicle dataset [15] and SYSU nighttime vehicle dataset [4] are used for day and dusk conditions respectively. SYSU nighttime vehicle dataset [4] is aimed at night vehicles, however, since the images are taken from near cars and in the urban area with reasonable lighting, their visual appearance is quite visible, and we categorize them into dusk condition.

Having two different datasets, we trained three models named as day, dusk, and combined. As the naming implies, the first two models are trained by only considering the training data from one dataset. The final model is trained by using both datasets to check if it could improve the accuracy. To evaluate this hypothesis, positive and negative images from both UPM [15] and SYSU [4] dataset are tested against both of day and dusk models as well as the combined model.

Detection accuracy is defined in Equation (1). *TP* stands for true positives which are the correctly detected vehicles, *TN* stands for true negatives which are those images that are correctly classified as non-vehicle, *FP* is the number of false positives which are the images incorrectly classified as vehicle, and *FN* is the number of false negatives which are the images that include the vehicle but are not classified correctly.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Table I shows the result of tests with three different SVM models. Detection accuracy as well as *TP*, *TN*, *FP*, and *FN* are included for each test case. It is clearly realized that the accuracy in the day is higher than in the dusk, as expected. Also, the best classifier model for detection in day is the day model which is trained solely by the day dataset. Combined SVM model outperforms the other two models in dusk condition with moderate environment light. Since some of the test images in SYSU nighttime vehicle dataset [4] are taken in very dark environment, they could be excluded from the dusk test. These images could be used for the evaluation of dark condition instead. The last set of columns in Table I includes the result for the subset of SYSU dataset [4] and shows considerable improvement in the accuracy of detection. The results confirm the need of two separate SVM models for better accuracy.

Implemented pipeline for these two scenarios are similar to [17] and is presented in Figure 2. These two configurations are implemented in the same way but with different versions of the trained model which are stored in two block RAM. The implemented pipeline consists of three main stages where first the gradients and histogram are generated. The extracted features are then normalized to suppress the effects of different local brightness. In the final stage, images are classified by use of trained SVM model. In the middle of each two stages, intermediate temporary storage is considered to address the requirements of memory access patterns within the algorithm and maintain the high throughput.

B. Detection During Dark

In very dark conditions during the night, when the car headlights are the main source of light on the road, only the objects within near distance of the car could be recognized by their shape and visual appearance. The only other detectable objects on the road, are the light sources which are either the road lights or the other vehicle taillights. In this situation, using the appearance features such as HOG, which are very promising in normal light condition, are not helpful in detecting the cars. Under very dark conditions, even the human drivers understanding of vehicle presence mostly relies on the existence of taillights with expected size, color, and position.

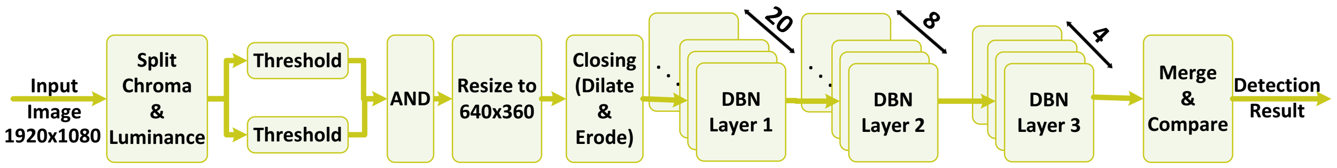

Fig. 4. Pipeline of implemented vehicle detection in dark.

We propose a method in which a two stage detection is accomplished by training a DBN to extract size and shape features of taillights in the first stage. In the second stage, we use the generated features of previous stage, which are considered in a spatial manner to match the detected taillights and detect the existence of vehicles. Two preprocessing stages are considered as well to subtract the background and eliminate the noise. Figure 3 shows the block diagram of our proposed method for detection of vehicles in dark.

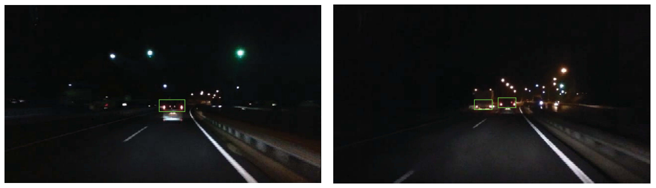

Fig. 5. Sample results of vehicle detection in dark on images of iROADS [18] dataset.

Region of interests in the captured image are defined as those areas with presence of any light source which could be considered as a potential taillight. Since presence of the light in dark environment results in a distinctive contrast, thresholding the luminance channel defines the area in which light sources are available. However, for detection of front cars, these areas should be only limited to the taillights therefore, headlights of the vehicles coming from the other direction as well as on road lights should be eliminated in the detection process. The color information of the light is one of the main determining factors in this case. Similar to most of the other works which aim at the detection of vehicles in dark environments, the initial stage of the detection subtracts the background by thresholding. Instead of relying only on the luminance information, we consider both the chrominance and luminance channels during the threshold stage which creates the binary output for further processing. Merging these two selections provides a binary image which is followed by downsampling, dilation and erosion. These operations remove the noise that might have been generated during the threshold stage and improve the smoothness of contours by closing small holes in the blobs.

The next stage is a sliding window DBN which detects and localizes the taillights and extracts their shape and size features. DBNs are categorized as the generative class of deep learning architectures. They are probabilistic models composed of multiple layers of stochastic, hidden variables. These layers are separately trained restricted Boltzmann machines (RBM) which are stacked on top of each other to extract the hidden features. We train a DBN with 81 visible inputs corresponding to the binary values of a 9x9 window of the image. Our DBN consists of two hidden layers with 20 and 8 hidden nodes, respectively. We use the cropped images of taillights from the training images of SYSU [4] dataset. The final Output layer consists of 4 nodes which determine the size and shape class of taillights. The trained network is then applied to the pre-processed image by sliding it over a window of 9x9 with the stride of 2. The final stage is the spatial correlation which is achieved by using a trained SVM classifier over a selection of detected taillights. Since the distance between the two taillights is expected to be within a specific range, only a particular region around each detected taillight is processed for matching. This will reduce the processing time and increase the reliability of our detection. Implemented pipeline is shown in Figure 4.

For the purpose of evaluations, a subset of SYSU dataset [4] was tested with our detection method and accuracy of 95% is obtained. We also evaluate our method on a subset of iROADS [18] dataset in very dark environments. Figure 5 shows a sample result with our detection method.

IV. System-level Implementation

Our detection unit for ADS is implemented by employing both the processing system and programmable logic on Zynq SoC. While the computationally intensive tasks of image processing and object detection are offloaded on hardware accelerators, the software applications on ARM cores, manage the data transfer between PS and PL and control the reconfiguration process.

Figure 6 shows the implemented system and the interfaces between different hardware modules and processing system. Input image and detection results are transferred through high performance ports to minimize the delay. Since the input and output of the pedestrian and vehicle detection modules are stream data, AXI DMA cores are required to manage the conversion between the memory mapped and stream data. All AXI DMA cores and detection modules are controlled by the PS through their AXI-Lite interfaces which is connected to PS general-purpose port of AXI-GP-0. Processing system initiates the DMA data transfer by writing to its registers and defining the size of data. Parameters of detection modules are also accessible by PS and could be updated through AXI-Lite interface. DMA cores and detection modules generate interrupt

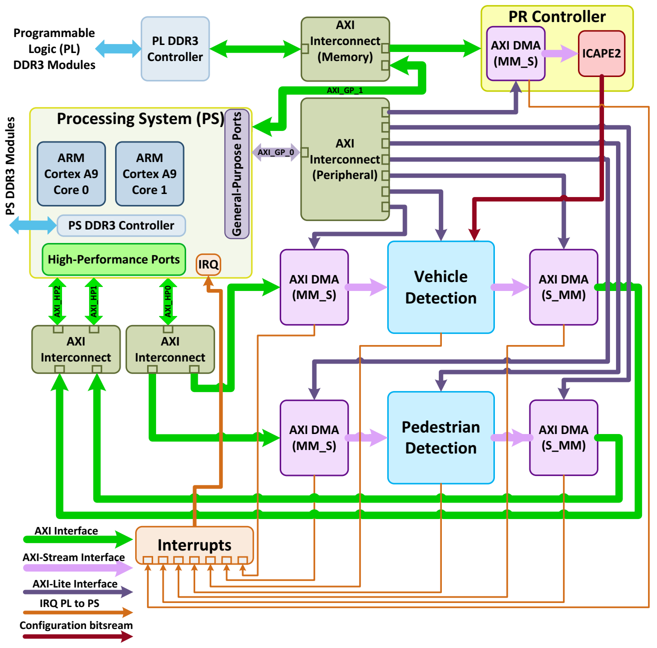

Fig. 6. Block diagram of implemented system and interfaces between the PS and PL components.

requests and inform PS of their completed assigned task as part of the communication between PL and PS.

We divide the implementation into two parts of static and dynamic. The data capturing part, PS to PL connection, PR controller, and pedestrian detection are included in static partition and the vehicle detection unit is in the reconfigurable part. Two different partial configurations are generated separately: one for the day and dusk, and the other one for the dark condition. The two partial configurations have the same interface to the other parts of the design. This guarantees the proper and isolated functionality of the other parts of the design for different configurations and during the reconfiguration process.

A. Static Design

The static part of the implementation includes PS to PL connection and functional modules which are not changed during the reconfiguration. The main functional block in the static part of the system detects pedestrians on the road. Similar to the method that is used for detection of vehicles in day and dark, it extracts HOG features of input image and use linear SVM classifier to detect pedestrians on the road. Implemented pipeline is based on the work presented by Hemmati et al. [17] to achieve real-time throughput and minimize the memory bottlenecks. This part is available in the implemented system to showcase the seamless operation of other detection modules during the partial reconfiguration.

Our PR controller is also implemented on the PL in the static part. The easiest method for downloading the bitstream from PS to the PL in Zynq SoC is through the processor configuration access port (PCAP). Using PCAP at 100MHz, should ideally result in 400 MB/sec throughput [1], however,

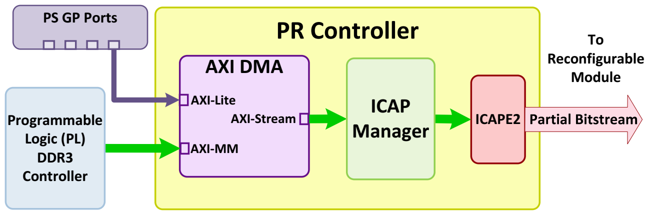

Fig. 7. Block diagram of partial reconfiguration (PR) controller.

due to the delays generated by the central interconnect in the PS, the reconfiguration rate is only around 145 MB/sec. Since most of the required time for reconfiguration process is spent on transferring the data from the DDR memory to PCAP bridge on the hardware, using configurations similar to ZyCAP [19] could further improve the process and decrease the reconfiguration time. Even though there is a dedicated DMA controller to transfer bitstreams from memory to PCAP, since it is passed through the central interconnect of PS, unwanted delays could affect the reconfiguration time. ZyCAP [19] instantiates a DMA core on the PL and transfers the configuration data to the PL through an AXI HP port. Then the bitstream data is loaded to the reconfigurable module through internal configuration access port (ICAP). This method could improve the throughput to 382 MB/sec which is 95.5% of the theoretical maximum [1], [19].

We use a new approach for the reconfiguration, where we eliminate any delay that could be imposed by the PS and leave the AXI HP port of PS for other high speed data transfers. Figure 7 shows the block diagram of our implemented PR controller. In our method, we use ICAP instead of PCAP which has the same theoretical throughput of 400 MB/sec. Although Xilinx provides its own AXI_HWICAP core [1], its throughput is only 19 MB/sec which is considerably lower than the performance which could be achieved by use of PCAP. The reason behind that low throughput is the overhead which is caused by the transfer of bitstreams through general-purpose ports of PS. In our method, we initially transfer partial bitstreams to the DDR module which is dedicated to PL. During the reconfiguration process, these bitstreams are transferred through the DMA controller to the ICAP manager and ICAP primitive which is instantiated on the PL. Once the reconfiguration process is initiated by the PS, the DMA core connected to the ICAP manager is triggered through its AXI interface. The bitstream is transferred to the ICAP primitive and it reconfigures the reconfigurable partition. Then the DMA core generates an interrupt to the PS to signal the end of reconfiguration process. Our measurement by use of ARM performance event counter as well as Vivado integrated logic analyzer (ILA) shows the throughput of 390MB/sec is achieved through this method.

B. Reconfigurable Design

Two different hardware accelerators are implemented as explained in Section III. The final design is then generated

TABLE II
RESOURCE UTILIZATION OF STATIC MODULE AND DIFFERENT CONFIGURATIONS OF RECONFIGURABLE MODULE.

	LUT	FF	BRAM	DSP48
Available Resources	277400	554800	755	2020
Static Design	21%	10%	12%	1%
Reconfigurable Partition	45%	45%	40%	40%
Day and Dusk Design	19%	9%	11%	1%
Dark Design	40%	23%	19%	29%
Total Usage	66%	55%	52%	41%

including the reconfigurable vehicle detection module as well as the static module. The reconfigurable part of the design is initially considered as a black box which defines its interface and boundaries and separates it from the static part. Considering the required resources for the largest configuration of vehicle detection, the reconfigurable module is floor-planned on the device. The partial bitstreams are generated for each configuration and are used during the dynamic reconfiguration process to update the functionality of vehicle detection.

Table II shows a summary of resource utilization for each configuration as well as the static part. Available resources on the reconfigurable partition is also presented. The second configuration for dynamic partition, corresponding to vehicle detection during dark has the most resource utilization. Total resource utilization is the summation of resources used for static design and the resources considered for the reconfigurable partition. The area of reconfigurable partition is considered big enough to fulfill the resource requirement of the largest configuration. Since the dark configuration consumes more resources on the FPGA fabric, about 1.2 times of its required resources is considered for the reconfigurable module during the floor-planning.

With our partial bit files of 8MB, the reconfiguration time is measured as 20ms which is equivalent to missing one frame in a sequence of 50fps. However, during this reconfiguration time, the pedestrian detection module continues its work. Moreover, the transition between dusk and dark which requires the reconfiguration does not happen frequently. For example, scenarios such as entering the tunnel is simply handled by the transition between day and dusk as the tunnel environment is well lighted and is categorized as dusk.

V. CONCLUSION

An adaptive system capable of detecting vehicles at different environmental lighting condition was presented by exploiting the partial reconfiguration feature of Zynq SoC. The system implemented on Mini ITX development board includes the pedestrian detection functionality as well, which is unchanged during the partial reconfiguration of the vehicle detection. Three different categories of day, dusk, and dark were defined based on the ambient lighting condition. We showed that by using three different partial configurations for vehicle detection, the adaptive detection could be done at no extra cost of resource utilization, resulting in more free resources available on the hardware for the other complex features of ADS to be added to the system. Our design running at 125MHz is capable of real-time detection at the rate of 50 fps.

REFERENCES

[1] *Partial Reconfiguration User Guide*, Xilinx Inc., 4 2017.

[2] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[3] R. O'Malley, E. Jones, and M. Glavin, "Rear-lamp vehicle detection and tracking in low-exposure color video for night conditions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 453–462, June 2010.

[4] L. Chen, X. Hu, T. Xu, H. Kuang, and Q. Li, "Turn signal detection during nighttime by cnn detector and perceptual hashing tracking," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–12, 2017.

[5] J. M. Guo, C. H. Hsia, K. Wong, J. Y. Wu, Y. T. Wu, and N. J. Wang, "Nighttime vehicle lamp detection and tracking with adaptive mask training," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 4023–4032, June 2016.

[6] Y. L. Chen, B. F. Wu, H. Y. Huang, and C. J. Fan, "A real-time vision system for nighttime vehicle detection and traffic surveillance," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 5, pp. 2030–2044, May 2011.

[7] H. T. Chen, Y. C. Wu, and C. C. Hsu, "Daytime preceding vehicle brake light detection using monocular vision," *IEEE Sensors Journal*, vol. 16, no. 1, pp. 120–131, Jan 2016.

[8] J. Kim, J. Baek, and E. Kim, "A novel on-road vehicle detection method using πhog," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3414–3429, Dec 2015.

[9] N. Kosaka and G. Ohashi, "Vision-based nighttime vehicle detection using censure and svm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2599–2608, Oct 2015.

[10] T. Schamm, C. von Carlowitz, and J. M. Zllner, "On-road vehicle detection during dusk and at night," in *2010 IEEE Intelligent Vehicles Symposium*, June 2010, pp. 418–423.

[11] R. K. Satzoda and M. M. Trivedi, "Looking at vehicles in the night: Detection and dynamics of rear lights," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–11, 2016.

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 886–893 vol. 1.

[13] V. Vapnik, S. E. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Advances in Neural Information Processing Systems 9*. MIT Press, 1996, pp. 281–287.

[14] C. L. Chien, H. M. Hang, D. C. Tseng, and Y. S. Chen, "An image based overexposed taillight detection method for frontal vehicle detection in night vision," in *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, Dec 2016, pp. 1–9.

[15] J. Arróspide, L. Salgado, and M. Nieto, "Video analysis-based vehicle detection and tracking using an mcmc sampling framework," *EURASIP Journal on Advances in Signal Processing"*, vol. 2012, no. 1, p. 2, Jan 2012. [Online]. Available: https://doi.org/10.1186/1687-6180-2012-2

[16] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.

[17] M. Hemmati, M. Biglari-Abhari, S. Niar, and S. Berber, "Real-time multi-scale pedestrian detection for driver assistance systems," in *Proceedings of the 54th Annual Design Automation Conference 2017*, ser. DAC '17. Austin, TX, USA: ACM, 2017, pp. 49:1–49:6.

[18] M. Rezaei and M. Terauchi, "Vehicle detection based on multi-feature clues and dempster-shafer fusion theory," in *Image and Video Technology*, R. Klette, M. Rivera, and S. Satoh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 60–72.

[19] K. Vipin and S. A. Fahmy, "Zycap: Efficient partial reconfiguration management on the xilinx zynq," *IEEE Embedded Systems Letters*, vol. 6, no. 3, pp. 41–44, 2014.