Proc. of the 2nd International Conference on Electrical, Communication and Computer Engineering (ICECCE)
12-13 June 2020, Istanbul, Turkey

An Optimized Hardware/Software Co-Design Framework for Real-Time Pedestrian Detection

Mirza Athar Baig
Department of Electrical Engineering,
School of Science and Engineering,
LUMS, Lahore, Pakistan
Email:15060027@lums.edu.pk

Muhammad Adeel Pasha
Department of Electrical Engineering,
School of Science and Engineering,
LUMS, Lahore, Pakistan
Email:adeel.pasha@lums.edu.pk

Shahid Masud
Department of Electrical Engineering,
School of Science and Engineering,
LUMS, Lahore, Pakistan
Email:smasud@lums.edu.pk

***Abstract*—Real-time Pedestrian Detection is an important feature of on-board intelligent Advanced Driver Assistance System (ADAS). The system employs feature detection and classification to identify human objects for pedestrian detection and their collision avoidance. However, these processes are computationally expensive and too complex for real-time implementation on a general-purpose processor. To address the issue of real-time deployment of such systems, this paper presents an approach using hardware/software co-design based on Xilinx Zynq FPGA and optimized OpenCV implementation. It uses Histogram of Oriented Gradients (HOG) features for human object detection and multi-stage Support Vector Machine (SVM) for object classification. The results show that our proposed co-design approach is at least 7× lower in latency than the reported software-based implementations and 6.6× faster than the previously reported HW/SW co-design approaches. In addition, our solution achieves 94% recall and 92% precision for 640×480 resolution images. The results have been verified using INRIA and MIT datasets.**

***Index Terms*—Co-design Methodology; FPGAs; Object Classification; HOG; SVM**

I. Introduction

The global status report on road safety 2015 indicates that 1.35 million people per year die in road accidents including 26% are pedestrians and cyclists [1]. In Pakistan, every year more than 27,000 people die in traffic accidents [2]. Most of these deaths are caused due to human error or negligence. According to World Health Organization (WHO) in an accident, pedestrians are 1.5× more likely to be killed than the driver of the vehicle [3]. The reasons behind these accidents include a limited view of the driver and human perception limitations. Most of the pedestrian-vehicle collisions occur in urban environment where complex scenes and continuous motion of the scene-elements make it more challenging for the drivers to act promptly and avoid collisions.

To avoid above-mentioned issues, Advanced Driver Assistance Systems (ADAS) are being developed where the main objective is to enhance the human visual information using one or more cameras mounted outside the driver's cabin [4]. The major processing workload performed by pedestrian detection systems can be partitioned into two phases: (i) feature extraction from the input image and (ii) its classification into human objects. As far as feature extraction is concerned, different features can be used for the object detection in images that include HAAR-like, Discriminatory Haar Features (DHFs), HAAR-cascade, etc.

Similarly, the authors in [5] suggested Histogram of Oriented Gradients (HOG) features for pedestrian detection that have proven very effective for human detection. The HOG features are robust in terms of color and shape of the person, and distinguish them from the background. However, these features require huge amount of computing resources that becomes a bottleneck for real-time embedded implementation to be used in ADAS like embedded applications.

Subsequently, for the classification phase, several methodologies can be used that include Artificial Neural Network (ANN), Decision Tree (DT), Support Vector Machine (SVM) and Fuzzy Classification. Since the base work done in [5] uses SVM-based classification, we have also incorporated a multi-stage SVM for object classification in our design-flow. However, multi-stage SVM classification is also too complex for real-time implementation on a general-purpose processor.

To facilitate embedded implementation, we are proposing a Hardware/Software (HW/SW) co-design methodology that can be used for such object detection and classification system. The proposed approach starts with system profiling to gauge the computational intensities of different tasks involved in pedestrian detection system. The compute-intensive tasks are then off-loaded to the hardware while the control-oriented tasks are run on the programmable logic or the software. We have used the Xilinx Zedboard as co-design platform that includes a Xilinx Zynq FPGA for the hardware part and an ARM-Core A9 processor for the software part. Xillybus IP-CORE [6] is used for data communication between the hardware and ARM-Core eliminating the complexities and latencies of AXI ports. The merits of both processor (software) and FPGA (hardware) are combined to obtain better results and real-time performance. For demonstration purposes, we

978-1-7281-7116-6/20/$31.00 ©2020 IEEE

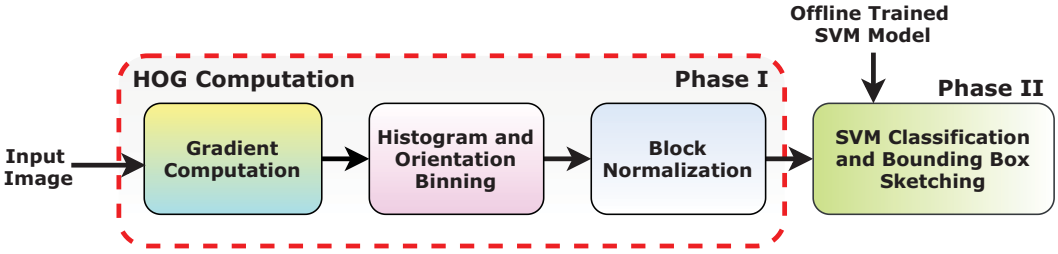


Fig. 1. System-Level Design-Flow of HOG-based Pedestrian Detection System

have used HOG features for object detection and multi-stage SVM for the classification.

The results show that our proposed implementation has a speedup of 6.6× with a 4–6% better precision than the existing HW/SW co-design approaches reported in the literature [7]. Similarly, it is at least 7× faster than the software-only approaches. Moreover, our proposed system can detect more than seven different-sized human objects using multi-scale object detection. The proposed system uses an offline trained model for classification that is implemented in software on ARM-Core A9 processor. This feature makes the overall system very flexible and can be adapted for applications other than the pedestrian detection by just adding or changing the offline trained classification models.

The remainder of the paper is organized as follows. In Section II, related work is presented. The algorithmic overview of feature-detection system and a system-level load profiling is discussed in Section III. Section IV describes in detail our proposed HW/SW co-design approach along with the contribution in improving the efficiency of OpenCV-based classification system. Section V presents detailed results of our implementation and its comparison against existing designs. The paper is finally concluded in Section VI.

II. Related Work

As we are using HOG features for human object detection and then SVM-based object classification as a test-case of our co-design methodology, we restrict ourselves to the related work in these two domains.

The authors in [5] proposed HOG features to detect human objects in images and video frames. Since then, many researchers have attempted to efficiently implement these features on different processing architectures such as graphical processing units (GPUs), digital signal processing (DSP) processors and field programmable gate arrays (FPGAs). GPU based solutions in [8] and [9] are not feasible for real-time embedded applications due to their high power and energy consumption. In [10], HOG+SVM-based pedestrian detection system is implemented on a DSP processor that achieved 8 frames per seconds (fps) on 640×480 resolution images.

In [11], 60 fps is achieved for images of same resolution through an FPGA-based implementation. However, these can detect only single-sized human objects and the implementations would fail if the size of human object varies in the image.

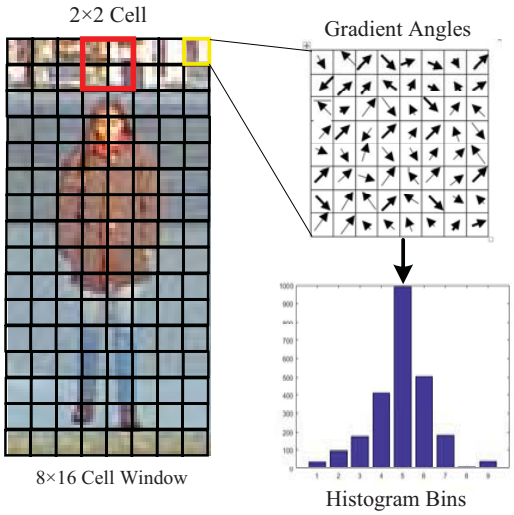


Fig. 2. Cell, Block, and Histogram in Feature Extraction

Such approaches cannot be used in ADAS application as the human object size will vary as a pedestrian will move relative to a moving vehicle. A better approach would be a multi-scale detection system that can detect human objects of different sizes.

The work reported in [12] is an FPGA-based hardware-only approach that achieves 90.2% detection rate on high resolution 1080P video streams but lacks in precision which is reported to be 86.6%. A safety critical application like pedestrian detection would require higher values for recall and accuracy. Similarly, work done in [7] is an extension of [12] towards HW/SW co-design but it achieved only 0.45 fps on a 350×175 resolution image. One major factor in this significant latency degradation is the data transfer bottleneck between the hardware and software components of their co-design.

III. Algorithmic Overview and System Profiling

Fig. 1 shows the generic system-level design-flow of HOG-based pedestrian detection system. The overall system can be partitioned into two major phases: (i) HOG feature extraction from the input image and (ii) the classification of HOG features into human objects using SVM.

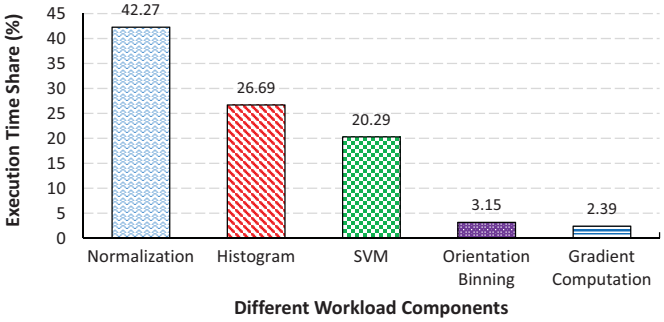


Fig. 3. System Profiling to identify the Compute Intensive Tasks

A. Feature Extraction

In the first phase (Phase-I), a sliding window approach is used to compute HOG features. For each pixel of the image, two-dimensional (2D) gradients are calculated using Eq. (1) and (2).

$$f_x(x,y) = \partial I(x,y)/\partial x = I(x+1,y) - I(x-1,y) \quad (1)$$

$$f_y(x,y) = \partial I(x,y)/\partial y = I(x,y+1) - I(x,y-1) \quad (2)$$

The magnitude of the final vector can be calculated by the square root of the sum of the squares of each gradient as given in Eq. (3).

$$m(x,y) = \sqrt{f_x(x,y)^2 + f_y(x,y)^2} \quad (3)$$

Similarly, the angle of the final vector is calculated by calculating the tangent of the horizontal and vertical gradient as given by Eq. (4).

$$\theta(x,y) = \tan^{-1}(f_x/f_y) \quad (4)$$

The whole image window is divided into spatial locations of predefined size, called a "cell" and a combination of cells is called a "block". For instance, Fig. 2 shows a cell with a size of 8×8 pixels while a block is made of 2×2 cells. Each pixel (in a cell) votes for a histogram bin. The angle of the pixel vector (θ) decides the bin where its vote will be counted for and the vote is simply the magnitude of the final vector (m) as shown in Fig. 2. Further details on HOG extraction and binning can be found in [5].

B. Feature Classification using Support Vector Machine (SVM)

In second phase (Phase-II) of the design-flow, the extracted features are fed to a multi-stage support vector machine (SVM) for classification between pedestrian and non-pedestrian. For classification of the HOG feature vector $\mathbf{h}$, we have used a linear SVM that computes a binary classification function $z(\mathbf{h})$ using Eq. (5).

$$z(\mathbf{h}) = w^t.\mathbf{h} + b \quad (5)$$

Where the weight vector, w and the bias, b are determined during the training phase of the SVM. Small changes in these parameters have a strong impact on classification accuracy as described in [13]. Consequently, both parameters require floating point (FP) implementation. If $z(\mathbf{h}) > 0$, a pedestrian is considered as detected, and vice versa.

C. System Profiling

Since different tasks involved in the design-flow are computationally disparate, we started with a system-level profiling to find out the most compute-intensive tasks that can then be moved to hardware in a co-design approach. Fig. 3 presents the system profiling results of the overall pedestrian detection design-flow. It is evident that more than 75% of the time is consumed in the execution of subroutines of HOG feature extraction, i.e. gradient computation, histogram for orientation binning and block normalization, which makes it a strong candidate for hardware acceleration on FPGA. Besides this, HOG algorithm has a lot potential for parallel processing due to its sliding window nature lending the possibility to compute many windows in parallel. In addition, as mentioned earlier, the variables involved in SVM classification require a floating point (FP) precision for accurate computation which is quite costly and not favored for FPGA implementation. Hence, to get the best of both the hardware and software resources, we decided to place Phase-I and Phase-II of our design-flow to FPGA fabric and ARM-Core A9 processor, respectively.

IV. Proposed Hardware/Software Co-Design Framework

Xilinx Zedboard is used as a target platform for our HW/SW co-design framework. This board has two parts: (i) Programmable Logic (PL) and (ii) Processing System (PS). PL consists of Xilinx Zynq 7 series fabric with 85 k programmable logic cells while the PS consists of an ARM dual-core Cortex-A9 MPCore which can be used as a Linux-based system [14]. The overall architecture is shown in Fig. 4.

A. Software Platform and Implementation

The PS is used as the software platform. It uses Xilinx proprietary operating system called Xilinux that is based on Ubuntu 12.4 LTS distribution.

OpenCV-2.3.1v is used on the PS side to perform all the operations. The ARM processor reads the input image, converts it into greyscale and sends it to the FPGA fabric (PL) to extract the HOG descriptors as shown in Fig. 4. The PL then sends back the HOG descriptors to the PS through Xillybus and the PS subsequently implements linear SVM for the classification phase of the pedestrian detection.

In our co-design, we used Xillybus IP core for data communication between the PL and the PS that removes the complexity of AXI ports for data synchronization. Xilinux contains the drivers for the Xillybus IP-core which is based on DMA and can be used for data transfer between a processor running Linux and a Xilinx FPGA core. It resulted in a significant reduction in the development time for the hardware

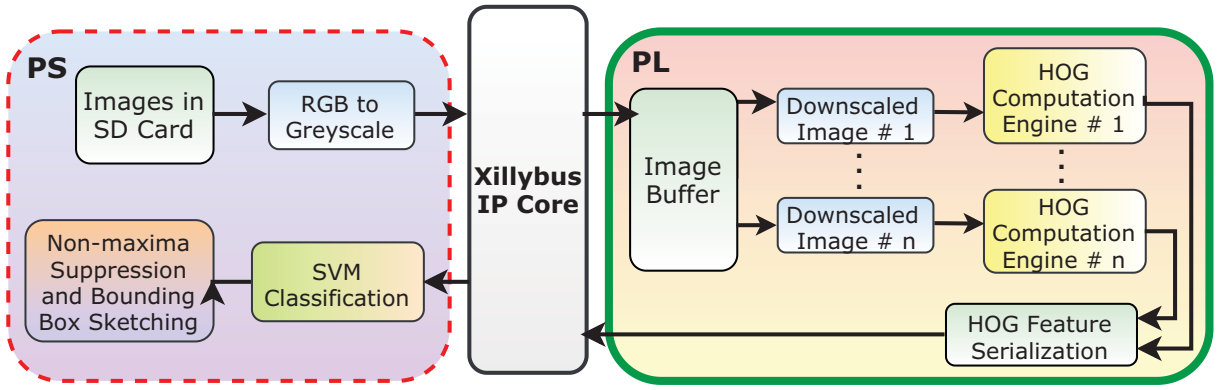


Fig. 4. Generic Overview of HW/SW Co-Design Framework.

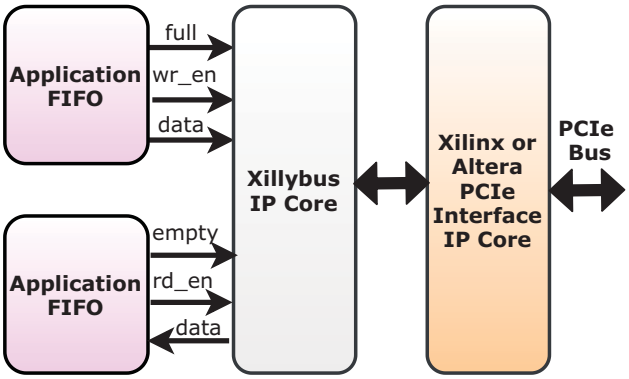


Fig. 5. Simplified Block Diagram of Xillybus IP Core.

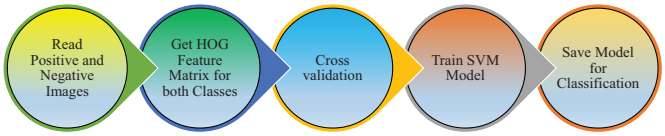


Fig. 6. Single-Stage of SVM Model Training

Fig. 7. Hard Examples of False Positives

part of the framework as we avoided the coding for complex AXI or PCIe interfaces. In addition, due to cleaner interface, the processing time of overall design is reduced which partially improved the throughput when compared to the existing co-design approaches such as [7]. Fig 5 presents a simplified block diagram of the Xillybus IP Core interfaces [6].

The SVM was trained on INRIA dataset. Training is a one-time offline process and the trained model is saved in the PS for the classification. One of the major contributions of our co-design approach is to reduce the false positive detection rate of default OpenCV implementation that uses a linear SVM. Experiments show that the default OpenCV implementation of HOG + SVM based pedestrian detection results in very low precision of around 80.86% due to a high false positive rate. We filled this performance gap by a multi-stage training process. In multi-stage training, the training process is divided into several distinct training steps where each step employs the results of the previous ones to further refine the training process. The process is repeated until no further improvement is detected. Fig. 6 shows the overall training phase of a single-stage linear SVM for classification.

To elaborate further, the HOG features extracted from the training samples of INRIA Pedestrian dataset [15] are fed to the SVM training class to train the initial classifier. The trained SVM is applied to the original negative training set and a positive detection in this set means a false positive as shown by an example in Fig. 7.

The windows marked red in Fig. 7 are false positive that

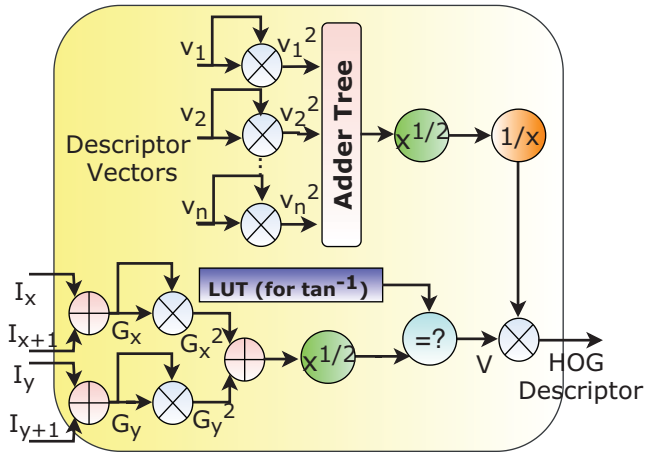


Fig. 8. A Basic PE designed for HOG Feature Extraction.

occurred while testing the first stage linear SVM trained model and these are called "hard examples". In multi-stage classification, these windows are cropped, re-sized to 64×128 pixels and added to the negative training set. The SVM model then is retrained with this enhanced negative set and this process is reiterated until no further hard examples are found. This method helps in significant reduction in false positives. The SVM classification is followed by a non-maxima suppression to avoid multiple detections of the same window due to multi-scale detection. Finally, a bounding box is drawn around the detected window.

B. Hardware Platform and Implementation

For co-design framework, the PL part of ZedBoard is used as a target hardware platform to implement the HOG feature extraction algorithm. We developed the HOG feature extraction IP in C++ and used the Xilinx Vivado High Level Synthesis (HLS) 2015.2 to generate and optimize that IP into synthesizable Verilog hardware description language (HDL). Vivado is also used for the connection of PL and PS through Xillybus as described earlier. Finally, a bit-stream for FPGA implementation is generated.

For multi-scale detection, the original image is down-sampled using bi-linear interpolation to obtain several images of different scales. For this purpose, an intermediate image buffer is created that stores one image at a time. This buffer is updated as soon as a new image is available. Hence, instead of sliding one window at a time, we can parallelize the operation and calculate all windows of a particular section simultaneously.

All the low-scaled images with the scale factor of 1.2 are computed. The default configurations of HOG, as proposed in [5], were used in hardware design and it is implemented as shown in Phase-I of the design-flow (Fig. 1). The gradient of the image is then calculated by simply subtracting the neighboring pixel using Eq. (1) and (2). Magnitude and angle of the gradient are calculated using the Eq. (3) and (4). Fig. 8 shows one basic processing element (PE) that was manually designed and implemented to perform HOG descriptor extraction. Given the logic resources available in PL for Zedboard, we were able to instantiate fifteen (15) parallel PEs to increase the throughput of our design.

These equations are implemented in single precision FP to gain the maximum of accuracy through features. These gradients are then fed to the histogram and binning blocks to compute a histogram. Finally, block normalization is done and the feature matrix is returned to the PS part for classification, non-maxima suppression and drawing rectangle around the detected pedestrian as shown in Fig. 4.

V. Experimental Results

We tested our proposed co-design approach using the MIT pedestrian dataset [16] and it gives more than 98.9% detection rate as shown in Table I. We then selected another challenging dataset, INRIA dataset [15], to evaluate our proposed design-flow. Table II summarizes the detection performance results for images of different resolutions. INRIA dataset contains 288 images of different sizes and conditions as test images. Moreover, the optimized multi-stage training of SVM helped in reducing the false positives and improved precision by 5% as shown in Fig. 9.

TABLE I
Experimental Results for MIT Dataset [16]

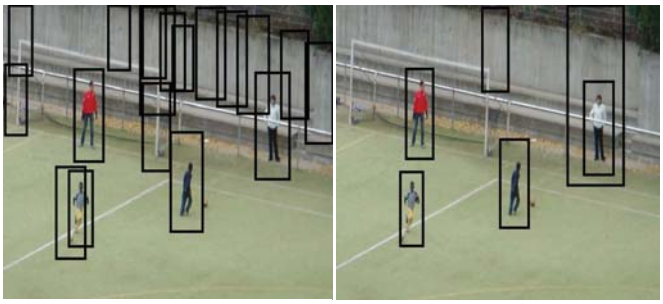
Image Type	RGB	Gray
Miss Rate	0.0109	0.0219
Precision	100.00	100.00
Recall	98.9	97.8
F Score	99.44	98.89

TABLE II
Experimental Results for INRIA Dataset [15]

Image Resolution	640×480	576×432	544×384
Miss Rate	5.97	8.57	11.06
Precision	92.4	89.54	87.29
Recall	94.02	91.4	88.93
Time (sec)	2.42	1.57	1.41

Fig. 10 summarizes performance results to show that the proposed co-design approach achieves classification performance equal to that of software-only approach which uses full precision FP implementation.

Similarly, Table III shows the latency and performance comparison of our approach against both HW/SW co-design and optimized OpenCV-based SW-only approaches for different image resolutions. The test-case design, generated through our methodology, achieves a speedup of 7.3× against the software-only approach while maintaining the same performance numbers (recall and precision). Similarly, our approach achieves a speedup of 6.6× against the existing HW/SW co-design approach with a 4–6% improvement in performance. Fig. 11 shows two examples of pedestrian detection using the proposed co-design framework.



Pre-Model Performance
Trained with INRIA dataset

Final Model Performance
Trained with INRIA dataset

Fig. 9. Pre-Model and Final Model Comparison after Multi-Stage SVM Training

TABLE III
OVERALL PERFORMANCE COMPARISON OF OUR FRAMEWORK AGAINST EXISTING APPROACHES

	Processing time per frame (640×480) (s)	Recall (%)	Precision (%)
Optimized OpenCV-based SW-Only Approach	2.49	94.02	92.4
HW/SW Co-design Approach [7]	2.27	-	-
Proposed Approach	0.34	94.0	92.4

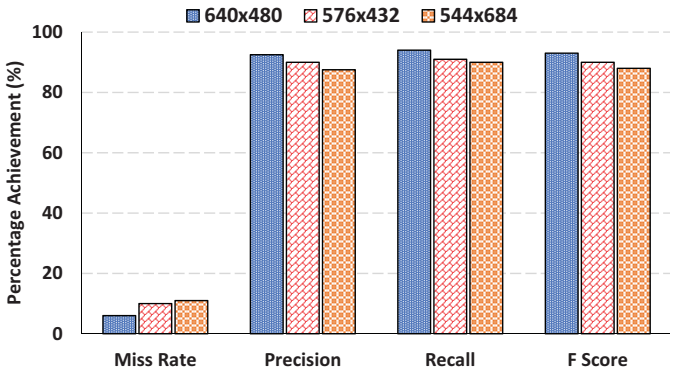


Fig. 10. Detection Performance of HW-SW Co-design on Different Resolution images

Fig. 11. Sample Results for our Proposed HW/SW Co-Design Approach

VI. CONCLUSIONS

Real-time pedestrian detection is an important feature of intelligent Advanced Driver Assistance System (ADAS). However, being computationally expensive, a general-purpose processor-based implementation is not suitable for real-time applications. To tackle this problem, we have proposed a HW/SW co-design approach that starts with system-level load profiling to offload more compute-intensive onto FPGA while control-oriented or precision-conscious tasks onto a soft-core processor. The main contribution is to combine the speedup of hardware (FPGA) and, precision and flexibility of software (processor) to get the best of both worlds. A test-case design of HOG+SVM-based pedestrian detection system, generated through our approach, shows a recall and precision numbers of 94.02% and 92.4%, respectively while giving at least 7× lower latency against the reported software implementations. Moreover, it is 6.6× faster than the previously proposed HW/SW co-design approaches with a 4–6% improvement in performance numbers. Moreover, the proposed co-design methodology is generic and can be adapted for other object detection systems.

REFERENCES

[1] WHO. (2018) Global Status Report on Road Safety. [Online]. Available: http://www.who.int/violence_injury_prevention/publications/road_traffic/

[2] W. H. Rankings, "Health Profile: Pakistan," Report, 2019. [Online]. Available: https://www.worldlifeexpectancy.com/pakistan-road-traffic-accidents

[3] U.S. Department of Health & Human Services, "Pedestrian Safety," Report, 2019. [Online]. Available: https://www.cdc.gov/motorvehiclesafety/pedestrian_safety/

[4] K. V. Sakhare, T. Tewari, and V. Vyas, "Review of Vehicle Detection Systems in Advanced Driver Assistant Systems," *Archives of Computational Methods in Engineering*, pp. 1–20, Mar 2019. [Online]. Available: https://doi.org/10.1007/s11831-019-09321-3

[5] N. Dalal, B. Triggs, and C. Schmid, "Human Detection Using Oriented Histograms of Flow and Appearance," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 428–441.

[6] Xillybus Ltd., "IP Core Product Brief," Techical Project, Xilinx, 2019. [Online]. Available: http://xillybus.com

[7] J. Rettkowski, A. Boutros, and D. Gohringer, "HW/SW Co-Design of the HOG algorithm on a Xilinx Zynq SoC," *Journal of Parallel and Distributed Computing*, vol. 109, pp. 50–62, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731517301569

[8] K. Lillywhite, D. J. Lee, and D. Zhang, "Real-time human detection using histograms of oriented gradients on a GPU," in *2009 Workshop on Applications of Computer Vision (WACV)*, Dec 2009, pp. 1–6.

[9] V. Campmany, S. Silva, A. Espinosa, J. Moure, D. Vazquez, and A. Lopez, "GPU-based Pedestrian Detection for Autonomous Driving," *Procedia Computer Science*, vol. 80, pp. 2377–2381, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050916309395

[10] C.-Y. Chiang, Y.-L. Chen, K.-C. Ke, and S.-M. Yuan, "Real-Time Pedestrian Detection Technique for Embedded Driver Assistance Systems," in *2015 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2015, pp. 206–207.

[11] K. Negi, K. Dohi, Y. Shibata, and K. Oguri, "Deep Pipelined One-Chip FPGA Implementation of a Real-time Image-based Human Detection Algorithm," in *2011 International Conference on Field-Programmable Technology (FPT'11)*, Dec 2011, pp. 1–8.

[12] J. Rettkowski, A. Boutros, and D. Gohringer, "Real-Time Pedestrian Detection on a Xilinx Zynq using the HOG Algorithm," in *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig'15)*, Dec 2015, pp. 1–8.

[13] K. P. Bennett and C. Campbell, "Support Vector Machines: Hype or Hallelujah?" *SIGKDD Explor. Newsl.*, vol. 2, no. 2, pp. 1–13, Dec. 2000. [Online]. Available: http://doi.acm.org/10.1145/380995.380999

[14] Xilinx, "Zedboard Documentation," Technical Report, 2019. [Online]. Available: http://zedboard.org/product/zedboard

[15] INRIA Grenoble, "INRIA Person Dataset," Technical Project, INRIA, 2019. [Online]. Available: http://pascal.inrialpes.fr/data/human/

[16] CBCL MIT, "MIT pedestrian dataset," Technical Project, MIT, 2019. [Online]. Available: http://poggio-lab.mit.edu/codedatasets