ARCHITECTURAL STUDY OF HOG FEATURE EXTRACTION PROCESSOR FOR REAL-TIME OBJECT DETECTION

Kosuke Mizuno[1], Yosuke Terachi[1], Kenta Takagi[1],
Shintaro Izumi[1], Hiroshi Kawaguchi[1] and Masahiko Yoshimoto[1,2]

[1] Department of Information Science, Kobe University,
1-1 Rokkodai-Cho, Nada-Ku, Kobe, Hyogo 657-8501, Japan
[2] JST CREST, Japan

ABSTRACT

This paper describes a Histogram of Oriented Gradients (HOG) feature extraction processor for HDTV resolution video (1920 × 1080 pixels). It features a simplified HOG algorithm with cell-based scanning and simultaneous Support Vector Machine (SVM) calculation, cell-based pipeline architecture, and parallelized modules. To evaluate the effectiveness of our approach, the proposed architecture is implemented onto a FPGA prototyping board. Results show that the proposed architecture can generate HOG features and detect objects with 40 MHz for SVGA resolution video (800 × 600 pixels) at 72 frames per second (fps). The proposed schemes are easily expandable to HDTV resolution video at 30 fps with 76.2 MHz if a high-resolution camera and higher operating frequency are available.

Index Terms— HOG, FPGA, VLSI, HDTV

1. INTRODUCTION

Real-time object detection has been a key technology in various application domains such as surveillance, automotive systems, and robotics. An important algorithm used in object detection systems, Histogram of Oriented Gradients (HOG) [1], has robustness to change of illumination and attains high computational accuracy in detection of variously textured objects.

Recent high-performance general-purpose processors can achieve real-time object detection at a heavy computational cost. However, the processor requires high power consumption and is therefore unsuitable for mobile systems under limited battery conditions. Consequently, a low-power and high-performance HOG feature extraction processor is necessary to widen the range of applications.

Figure 1 presents the image resolution versus frame rate for several published descriptions of HOG hardware. Zhang et al. [2] proposed efficient object detection using GPGPU. Some FPGA implementations [3], [5], [6], [8], [9] and an FPGA-GPU architecture [4] have been proposed for real-time applications. Cao et al. [7] realized an FPGA implementation with the best performance compared with other implementations. However, this study particularly targets stop-sign detection. HOG features are adaptable to widely various applications. Consequently, next-generation HOG feature extraction processors must provide higher expandability and higher performance. Therefore, our goal is to develop design techniques for a real-time HOG feature extraction processor for HDTV resolution video.

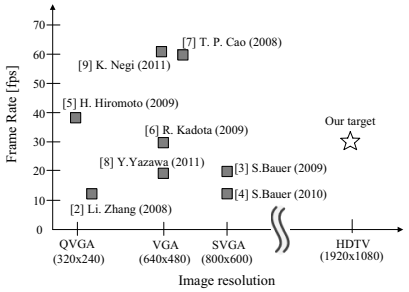

Fig. 1. Previous works of HOG feature extraction processor.

Most conventional processors employ a window-based approach. For the window-based approach, a workload of 447.7 GOPS and memory bandwidth of 55 Gbps are required for HDTV resolution because of repetitive computations. The workload and memory bandwidth are greatly reduced by reusing calculated data or adopting efficient computation. However, data reuse causes an increase of memory capacity and circuit area. Consequently, a cooperative design between algorithm and architecture is necessary.

To achieve real-time and low-power HOG feature extraction for HDTV resolution video, we propose the following three techniques.

- Simplified HOG algorithm with cell-based scanning and simultaneous Support Vector Machine (SVM) calculation for workload reduction.
- Cell-based algorithm and architecture for memory bandwidth reduction.

978-0-7695-4856-2/12 $26.00 © 2012 IEEE
DOI 10.1109/SiPS.2012.57

- Parallelized architectures for cell histogram generation, histogram normalization, and SVM classification to reduce the necessary cycle count.

As described in this paper, details of the simplified HOG algorithm are described in Section 2. The proposed architecture is addressed in Section 3. Then, these are followed by FPGA implementation in Section 4. Section 5 concludes this paper.

2. ALGORITHM

2.1. Original HOG Algorithm

Figure 2 portrays a flow diagram of object detection using the original HOG algorithm [1]. Scanning on the input image is based on detection window. The window is divided into cells, for each cell accumulating a histogram of gradient orientations over the pixels of the cell. For better invariance to illumination, histogram normalization can be done by accumulating a measure of the local histogram energy over blocks and using the results to normalize all cells in the block. The normalized histograms (HOG features) are collected over the detection window. The collected features are fed to a linear SVM for object/non-object classification.

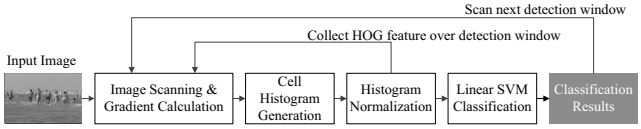

Fig. 2. Original HOG algorithm flow.

2.2. Simplified HOG Algorithm for Hardware Implementation

A simplified HOG algorithm for VLSI implementation is introduced in this subsection. Figure 3 shows a flow diagram of object detection using simplified HOG algorithm. This flow is modified from the original flow using the following five techniques.

1. Cell-based scanning (Section 2.3)
2. Gradient calculation using CORDIC [10]
3. Approximation of weighted voting for spatial and orientation anti-aliasing
4. Newton method with approximated initial value
5. Simultaneous SVM calculation (Section 2.4)

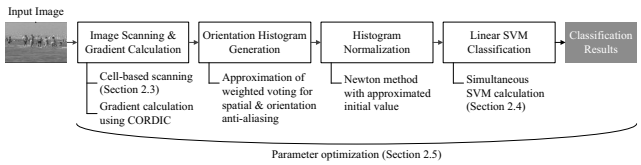

Fig. 3. Simplified HOG algorithm flow.

Figure 4 portrays the workload analysis of HOG-based object detection. Simplified HOG algorithm with cell-based scanning and simultaneous SVM calculation reduces the workload to 10.6 GOPS, as portrayed in Fig. 4. However, the workload of 10.6 GOPS is still heavy for a processor with low operating frequency. To accommodate the workload in real time, our architecture has parallelized modules for cell histogram generation, histogram normalization, and SVM classification.

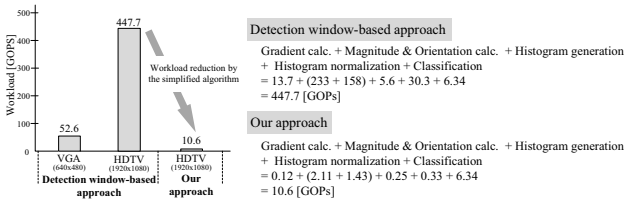

Fig. 4. Workload analysis.

2.3. Cell-based Scanning Method

Object detection with the HOG feature is executed by the scanning detection window on an input image, as presented in Fig. 5 left. The detection window size is compliant with original algorithm [1]. When one window is finished, the next window is scanned using an offset of 1 cell. The memory bandwidth is increased by reloading input pixels for the next window. Consequently, extensive data reuse is desirable for memory bandwidth reduction.

Figure 5 right shows cell-based scanning approach. HOG feature is extracted from cell-based calculations. No cell overlaps with other cells. Consequently, sharing and reuse of a cell have a great impact on memory bandwidth reduction.

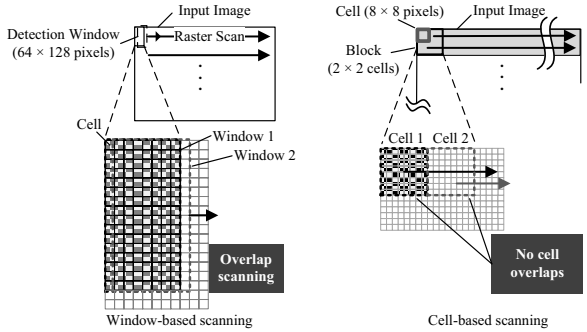

Fig. 5. Image scanning methods.

2.4. Simultaneous SVM Calculation

In the window-based approach, HOG features of 105 blocks are collected. Then the features are multiplied by SVM coefficients corresponding to one window. However, the cell-based approach provides partial HOG features after normalization for one block; then the features are

multiplied by SVM coefficients corresponding to 105 windows.

Figure 6 presents simultaneous SVM calculations for cell-based processing. Partial HOG feature belongs to 105 windows maximally and are located at different positions in each window. Partial HOG features are multiplied and accumulated by the SVM coefficients of each window. The accumulation result is stored and reused in the subsequent SVM calculation. Simultaneous SVM calculation is suitable for parallel computing in hardware.

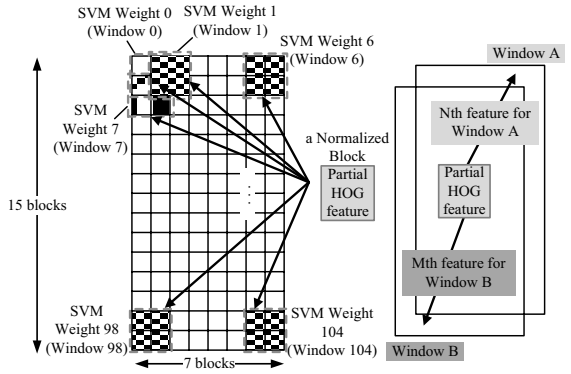

Fig. 6. Simultaneous SVM classification.

2.5. Parameter Optimization

Figure 7 shows parameters of each process in object detection using HOG features. In general, a software implementation employs floating-point calculations to provide high accuracy. However, the floating-point unit uses hardware resources to a great degree. Therefore, fixed-point operation is often used for hardware implementation. The accuracy of fixed-point operation depends on the bit width itself, although the bit width affects the memory capacity and the circuit area. Optimized parameters provide reasonable classification accuracy and minimize hardware costs. Table 1 presents the results of parameter adjustment.

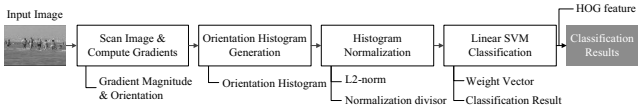

Fig. 7. HOG algorithm parameters.

Table 1. Optimized bit width

Parameter	Bit width [bit]		
	Sign	Integer	Fractional
Gradient magnitude	1	9	0
Gradient orientation	1	3	3
Orientation histogram	0	11	0
1st L2-norm	0	25	0
1st normalization divisor	0	0	11
2nd L2-norm	0	0	14
2nd normalization divisor	0	3	7
HOG feature	0	0	4
SVM coefficient	1	3	7
Classification buffer	1	4	8

2.6. Simulation Results

A simulation was conducted using software for object detection to estimate performance and accuracy degradation by the simplified algorithm. The software was produced using Microsoft Visual C++ 2008 Express Edition (Microsoft Corp.) with the INRIA Person Dataset [11], which includes several people in various backgrounds. Figure 8 presents a graph of false positives per window (FPPW) versus the miss rate. The simulation results with the simplified algorithm and the optimized bit width show that the miss rate degradation is 3% at 0.0001 FPPW. The algorithm that was used provides sufficient performance for general-purpose applications.

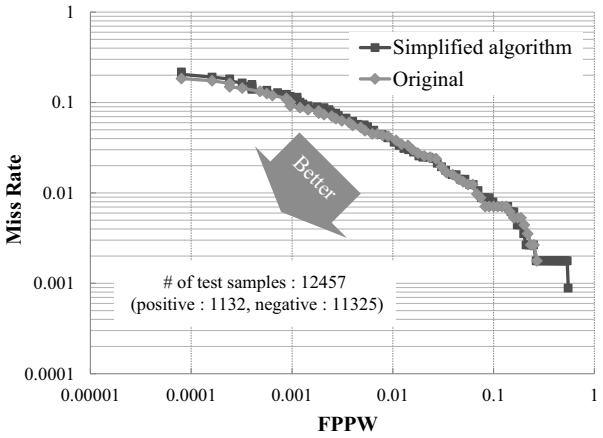

Fig. 8. Accuracy degradation by the simplified algorithm.

3. ARCHITECTURE

3.1. Cell-based Pipeline Architecture

Figure 9 depicts a block diagram of the cell-based pipeline architecture and external peripherals for a demonstration system detailed in Section 4. The proposed architecture comprises a controller, a cell histogram generation module, a histogram normalization module, an SVM classification module, SRAMs for several image data, a CPU interface, and a memory interface. The HOG feature extraction processor is controlled by an external CPU, and the input grayscale image is loaded to a cell-line buffer from an external SRAM via a memory interface. The CPU receives a detection result from HOG feature extraction processor; then it draws the result on an LCD display.

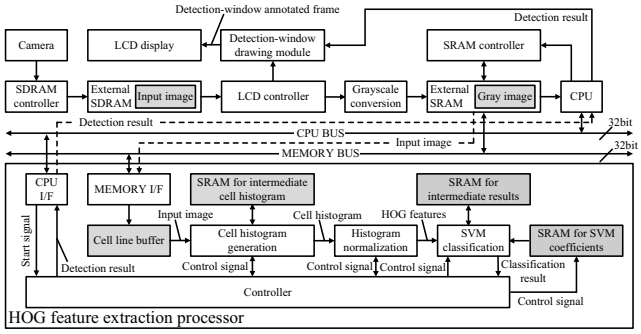

Fig. 9. HOG feature extraction architecture.

Our architecture adopts a cell-based pipeline flow, as presented in Fig. 10. Figure 10 above shows a relation between cells, blocks, windows, and a frame. One cell contains 8 × 8 pixels. One block is composed of 2 × 2 cells. One window is made up of 7 × 15 blocks. Each block overlaps with neighboring blocks. Cell-based pipeline processing is conducted as follows:

1. A cell histogram is generated with cell-based scanning.
2. When the process described above reaches the block level, a block-level cell histogram is normalized; then the block-level HOG feature is extracted.
3. Block-level HOG features and SVM coefficients corresponding to each window are multiplied and accumulated.
4. An accumulation result of window level is compared with the SVM threshold. Then the detection result is obtained.

The cell-based pipeline architecture greatly reduces the memory bandwidth because it prevents reloading of input pixels in different detection windows.

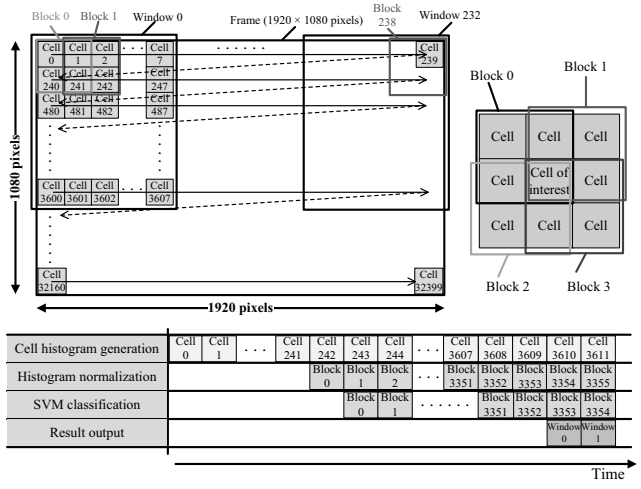

Fig. 10. Cell-based pipeline flow.

Figure 11 portrays the memory bandwidth analysis of HOG-based object detection. The window-based approach for HDTV resolution requires memory bandwidth of 55 Gbps. In general, the mobile system under limited battery conditions adopts a lower operating frequency. Therefore, the memory bandwidth must be reduced as low as possible for low-power and real-time operation. Our approach adopts a cell-based algorithm and architecture to reduce the memory bandwidth to 0.499 Gbps.

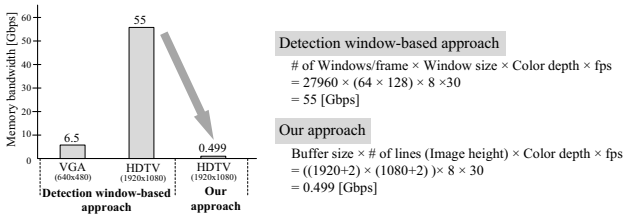

Fig. 11. Memory bandwidth analysis.

3.2. Cell Histogram Generation

In cell histogram generation, a magnitude and an orientation of a pixel gradient are calculated; then a weighted magnitude is voted into a bin corresponding to its orientation. Figure 12 presents the architecture for cell histogram generation. Four-way architecture is adopted because one cell is commonly used for four blocks maximally. One processing element (PE) executes weighted voting and binning to generate a histogram of one cell. Spatial anti-aliasing is conducted in four processing elements corresponding to one block.

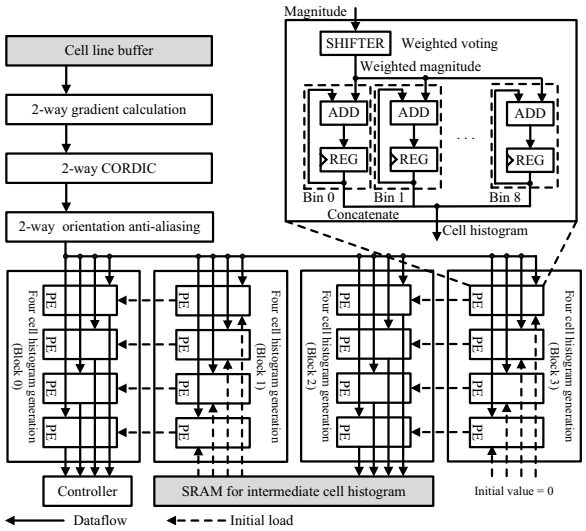

Fig. 12. Block diagram and processing flow of cell histogram generation.

3.3. Histogram Normalization

Figure 13 shows the architecture for histogram normalization. The architecture consists of two stages to

implement L2-Hys normalization [12]. The first stage includes four Cell MAC modules, an approximation module, a Newton method module, and a threshold module. The second stage comprises four Cell MAC modules and a Newton method module.

In the first stage, Cell MAC modules first calculate the sum of squares of input cell histogram. Secondly, an initial value for Newton method is approximated to bit shift operation. Thirdly, Newton method calculates an inverse number of square roots. Furthermore, then Cell MAC modules normalize a cell histogram. Finally, a normalized cell histogram is compared with a threshold and outputted to the second stage.

In the second stage, the sum of squares and an inverse number of square roots is calculated as in the first stage. Furthermore, then Cell MAC modules normalize a cell histogram and extract 36-dimension HOG features.

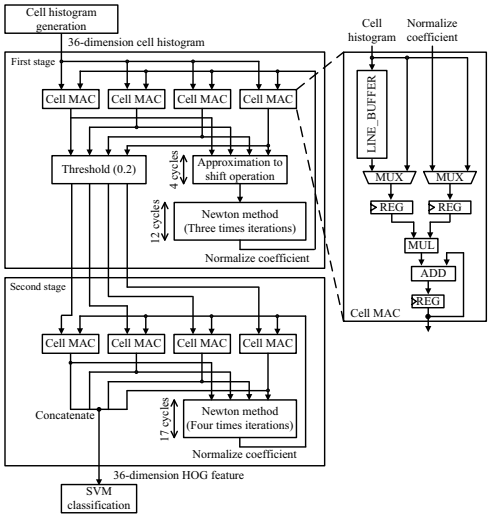

Fig. 13. Block diagram of histogram normalization.

3.4. SVM Classification

In SVM classification, extracted features and SVM coefficients are multiplied and accumulated until the operations reach window level. Then the accumulation result is compared with an SVM threshold to judge whether the window includes a target object. Figure 14 shows a block diagram for simultaneous SVM classification. This architecture includes 15 classification cores. One classification core manages MAC operations of 7 blocks. Consequently, the architecture is able to handle 105 blocks corresponding to one detection window. Sufficient parallelism reduces the required cycle count to manage the workload of 10.6 GOPS.

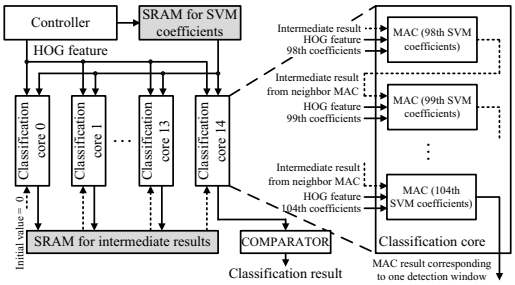

Fig. 14. Block diagram of SVM classification module.

3.5. Performance Evaluation

The number of cycle counts was estimated using a Verilog-HDL simulator. The proposed architecture was compared with architecture without parallelization and without a pipeline. Estimation results are presented in Fig. 15, which demonstrates the superiority of the proposed architecture for HDTV resolution. The parallelization in the cell histogram generation and histogram normalization contributes to reduction of the cycle counts. Introduction of the proposed simultaneous SVM calculation architecture enables the reuse of intermediate results, allowing the cycle count reduction. Results show that the number of cycle counts in cell histogram generation, histogram normalization, and SVM classification are reduced by 85%, 65%, and 99%, respectively, compared with the number of cycle counts of architecture without parallelization and without pipeline. In the proposed architecture, the overall process requires 2.54×10^6 cycles per frame. Therefore, it is inferred that the proposed architecture can accommodate HDTV resolution video at 30 fps with 76.2 MHz.

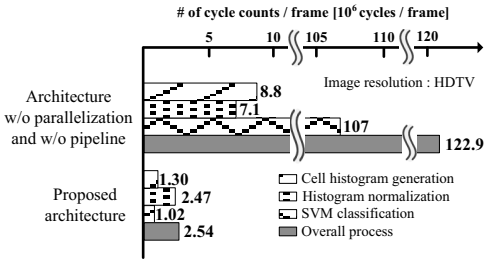

Fig. 15. Reduction of cycle count.

4. FPGA IMPLEMENTATION

To evaluate the effectiveness of our approach, we implemented the proposed architecture onto a prototyping board (tPad Multimedia Development Kit; Terasic Technologies Inc.). The board has DE2-115 with Cyclone IV EP4CE115 (Altera Corp.), a 5-megapixel digital image sensor module, and an 8-inch LCD touch screen module. Figure 16 portrays a demonstration system of real-time object detection to verify the proposed technique.

Resource utilization and comparison to conventional FPGA implementations are presented in Table 2. Our FPGA implementation can generate HOG features and detect objects with 40 MHz for SVGA resolution video at 72 fps. The FPGA resource utilizations are as follows: 34,403 LEs, 68 embedded multipliers, and 0.34 Mbit block RAMs. Our implementation shows the best performance with minimum memory usage and minimum operating frequency. If a high-resolution camera, 0.63 Mbit block RAMs, and the operating frequency of 76.2 MHz are available, then the proposed schemes are readily expandable to HDTV resolution video at 30 fps.

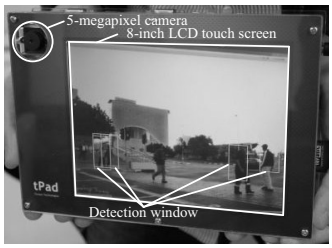

Fig. 16. Architecture verification by FPGA implementation.

Table 2. Resource utilization

	[3]	[5]	[6]	[7]	[8]	[9]	Ours	
FPGA	Spartan 3	Virtex-5	Stratix II	Virtex-4	Cyclone III	Virtex-5	Cyclone IV	
# of LUTs	42,435	28,495	37,940	8,921	34,838	17,383	34,403	
# of registers	N/A	5,980	66,990	4,221	22,612	2,181	23,247	
# of DSP blocks	18	2	120	3	N/A	N/A	68	
Working memory (Mbits)	1.08	2.196	N/A	1.584	2.094	1.296	0.34	0.63
Resolution	800×600	320×240	640×480	752×480	640×480	640×480	800×600	1920×1080
Frame rate (fps)	20	38	30	60	20	62.5	72	30
Operating frequency (MHz)	63	167	127.49	N/A	70	44.85	40	76.2
Image scanning method in HOG feature extraction	Window-based					Cell-based	Cell-based	
Image scanning method in classification	Window-based							

5. CONCLUSION

This paper presents a proposal of a novel architecture of real-time HOG feature extraction for HDTV resolution video. The proposed scheme has a simplified HOG algorithm with cell-based scanning, simultaneous SVM calculation, cell-based pipeline architecture, and parallelized modules. The simplified algorithm contributes to reduction of the workload from 447.7 GOPS to 10.6 GOPS with 3% accuracy degradation. The cell-based algorithm and pipeline architecture provide memory bandwidth of 0.499 Gbps at HDTV resolution. The memory bandwidth of 0.499 Gbps can be handled by a 32-bit memory bus with reasonably low operating frequency. Parallelized modules greatly accelerate HOG feature extraction and object detection. The proposed architecture on FPGA prototyping board shows the best performance with minimum memory usage and minimum operating frequency, compared with the performance of conventional processors. The proposed schemes provide expandability to HDTV resolution video (1920 × 1080 pixels) at 30 fps at 76.2 MHz.

6. ACKNOWLEDGMENTS

This work was supported by the VLSI Design and Education Center (VDEC), The University of Tokyo, in collaboration with Cadence Design Systems Inc. and Synopsys Inc.

7. REFERENCES

[1] N. Dalal, et al., "Histograms of Oriented Gradients for Human Detection," in Proceedings of the 2005 International Conference on Computer Vision and Pattern Recognition, vol. 2. Washington, DC, USA: IEEE Computer Society, 2005, pp. 886–893.

[2] Li Zhang, et al., "Efficient Scan-Window Based Object Detection using GPGPU," IEEE, CVPRW, 2008.

[3] S. Bauer, et al., "FPGA Implementation of a HOG-based Pedestrian Recognition System," MPC-Workshop, July, 2009.

[4] S. Bauer, et al., "FPGA-GPU Architecture for Kernel SVM Pedestrian Detection," IEEE CVPRW 2010.

[5] M. Hiromoto, et al., "Hardware Architecture for High-Accuracy Real-Time Pedestrian Detection with CoHOG Features," IEEE ICCVW 2009.

[6] R. Kadota, et al., "Hardware Architecture for HOG Feature Extraction," in Proceedings of the 2009 International Conference on Intelligent Information Hiding and Multimedia Signal Processing. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1330–1333.

[7] T. P. Cao, et al., "Real-Time Vision-Based Stop Sign Detection System on FPGA," in Proceedings of Digital Image Computing: Techniques and Applications. Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 465–471.

[8] Y. Yazawa, et al., "FPGA Hardware with Target-Reconfigurable Object Detector by Joint-HOG," in Proceeding of SSII. Yokohama, Japan, 2011.

[9] K. Negi, et al., "Deep pipelined one-chip FPGA implementation of a real-time image-based human detection algorithm," IEEE FPT 2011.

[10] J. E. Volder, "The CORDIC Trigonometric Computing Technique," IRE Trans. Electron. Comput. EC-8:330-334, 1959.

[11] INRIA Person Dataset.
http://pascal.inrialpes.fr/data/human/

[12] D. G. Lowe, "Distinctive image features from scale invariant keypoints," International Journal of Computer Vision, Vol.60, No.2, pp.91-110, 2004.