Journal of Software Engineering and Applications, 2019, 12, 1-19
http://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

An FPGA-Based HOG Accelerator with HW/SW Co-Design for Human Detection and Its Application to Crowd Density Estimation

Shih-Shinh Huang[1*], Shih-Yu Lin[2], Pei-Yung Hsiao[2]

[1]Department of Computer and Communication Engineering, National Kaohsiung University of Science and Technology, Taiwan
[2]Department of Electrical Engineering, National University of Kaohsiung, Taiwan
Email: *powwhuang@gmail.com, pyhsiao@nuk.edu.tw

How to cite this paper: Huang, S.-S., Lin, S.-Y. and Hsiao, P.-Y. (2019) An FPGA-Based HOG Accelerator with HW/SW Co-Design for Human Detection and Its Application to Crowd Density Estimation. *Journal of Software Engineering and Applications*, **12**, 1-19.
https://doi.org/10.4236/jsea.2019.121001

Received: December 12, 2018
Accepted: January 26, 2019
Published: January 29, 2019

Abstract

Human detection is important in many applications and has attracted significant attention over the last decade. The Histograms of Oriented Gradients (HOG) as effective local descriptors are used with binary sliding window mechanism to achieve good detection performance. However, the computation of HOG under such framework is about billion times and the pure software implementation for HOG computation is hard to meet the real-time requirement. This study proposes a hardware architecture called One-HOG accelerator operated on FPGA of Xilinx Spartan-6 LX-150T that provides an efficient way to compute HOG such that an embedded real-time platform of HW/SW co-design for application to crowd estimation and analysis is achieved. The One-HOG accelerator mainly consists of gradient module and histogram module. The gradient module is for computing gradient magnitude and orientation; histogram module is for generating a 36-D HOG feature vector. In addition to hardware realization, a new method called Histograms-of-Oriented-Gradients AdaBoost Long-Feature-Vector (HOG-AdaBoost-LFV) human classifier is proposed to significantly decrease the number of times to compute the HOG without sacrificing detection performance. The experiment results from three static image and four video datasets demonstrate that the proposed SW/HW (software/hardware) co-design system is 13.14 times faster than the pure software computation of Dalal algorithm.

Keywords

Human Detection, HOG, HW/SW Co-Design

1. Introduction

Over the last decade, academic research on human detection systems has attracted significant attention in many applications, such as intelligent surveillance system (ISS) and intelligent transportation systems (ITS). Among various local features, Histograms of Oriented Gradients (HOG) proposed by Dalal and Triggs [1] have been proven its robustness in the literature of human detection. The most common way to use HOG for human detection is with the binary scanning window framework. A region of interest (ROI) window with various sizes is scanned over the entire image. The appearance inside the ROI window is thus described in terms of HOG that are further used for determining the presence of the pedestrian. This kind of approaches has high detection accuracy, but it has high computation burden. To alleviate this, some works have devoted themselves for reducing computation complexity, such as [2] [3] [4]. However, it is still hard to reach the real-time processing of 30 frames per second under pure-software realization. In recent years, there have been researchers engaging in the development of HOG hardware accelerators for human detection. Besides human detection, HOG has been widely used in many other topics, such as object detection [5] and vehicle detection [6] [7] [8]. Driven by this, the work on accelerating HOG computation has attracted significant attention recently. The studies on accelerating the HOG computing speed with hardware can be generally divided into two categories including the FPGA emulations and the application-specific integrated circuit (ASIC) implementations. Both of them rely on sophisticated platform of HW/SW co-design [9] [10] [11] to fulfil an embedded real-time application to crown density estimation and analysis presented in this paper.

In the area of FPGA emulations, Kadota *et al.* [12] proposed a simplified HOG calculation and designed the HOG hardware with Altera Stratix II FPGA. Hemmati *et al.* [13] applied Xilinx Zynq-7000 FPGA to design the HOG calculation speed up to 60 FPS under 1920 × 1080 HDTV resolutions. Although the speed in above research could reach real-time, it was merely the calculation result of a single HOG, but ignored the connection and the computing time of the rest functional modules required for data input/output and applied them to the entire set of human detection system. Hatto *et al.* [14] made an improvement on this part, but there was a storage requirement for keeping the huge data of all HOG features in all detecting windows of an image in their work. Therefore, they decided to decrease the accuracy of HOG specifications to reduce the demand for large memory space, but the experimental result therefore revealed the detection accuracy was decreased about 2.68%. Furthermore, Bauer *et al.* [15] [16] combined FPGA with Graphics Processing Unit (GPU) on the display card in a personal computer for the computing of the human detection system. A PCI-e interfaced Camera Link image capture card had to be equipped on a PC for the system structure so that the image data entered the FPGA on the capture card, through the Camera Link interface, for the HOG computing. The short-

coming was that the input source of the image had to be restricted to the use of the Camera Link interface, and the detecting window size and the scan shift step in that study were fixed. That system therefore could not solve the problem of different human sizes, showing inadequate flexibility.

For ASIC implementation, Chen *et al.* [17] used a simple calculation approximating the original complicated calculation in HOG to reduce the computational complexity and hardware cost in their HOG ASIC implementation. Suleiman and Sze [18] also concerned and dealt with the low-power problem, such as the application of a human detection system to Unmanned Aerial Vehicle (UAV) being restricted to batteries. Multi-Scale Support was used in the study to solve the problem of various human sizes by scaling up and down the original image to generate a multi-layer pyramid image. Mizuno *et al.* [19] [20] proposed in 2013 to achieve the real-time object detection with high-performance and high-cost processors, but the required high-power consumption was not suitable for portable batteries. That study therefore proposed the design of a low-power HOG accelerator and utilizes FPGA as well as ASIC for the full hardware human detection system design. Nonetheless, most of the complicated circuits in that work were applied to implement the control unit instead of the data-path units. Therefore, such circuits do not reduce too much computing time but increase a lot of unnecessary hardware costs.

The main contribution of this study is two-fold. Firstly, a SW/HW co-design system is proposed for human detection. The functionalities with less computation burden, such as tuning parameters is implemented using software; the computation of HOG, the most intricate part, is through the FPGA-based hardware. The incorporation of the software for flexible consideration with the hardware for HOG computation accelerates the processing of the human detection. Secondly, to further speed up the proposed system, only a set of B blocks instead of all ones is selected for HOG computation. The selection of B blocks is through the AdaBoost algorithm to guarantee the effectiveness of the human descriptor and only sacrifice a little detection performance. The resulting system achieves real-time processing and applicable to different applications.

The remaining of this study is organized as follows. HOG feature descriptors and the HOG-AdaBoost-LVF based human classifiers proposed in this study are explained in Section 2. The use of FPGA for the hardware circuit design of HOG hardware accelerators is described in Section 3. The experiments on popular datasets and crowd density estimation are done in Section 4 and the conclusion is made in Section 5.

2. HOG-Based Human Detector

In this section, we will briefly describe the steps of computing HOG descriptor followed by the time percentage of each step in computing HOG implemented with pure software in PC. After that, we introduce the Dalal's approach for detection human based on HOG as well as the proposed HOG Adaboost

Long-Feature Vector (HOG-Adaboost-LFV) which significantly decreases the number of times to compute HOG feature vector for describing a detection window.

2.1. HOG Computation

In Dalal's detection approach, a rectangular block in a detection window is divided into four cells and each of which is described by a 9-D HOG feature vector. The concatenation of four 9-D HOG feature vectors forms a 36-D HOG vector for describing a rectangular block in a detection window. Figure 1 illustrates the procedure of how a 36-D feature vector is established. The computation of a 9-D HOG vector is consisted of calculating gradient component, gradient magnitude, gradient orientation, accumulated histogram, and histogram L2-normalization.

The first step is to calculate the horizontal gradient component, $G_x(x,y)$, and the vertical gradient component, $G_y(x,y)$, of each pixel at (x,y) by using [−1, 0, 1] and $[-1, 0, 1]^{\mathrm{T}}$ as the masks, respectively, for the convolution operation. Accordingly, the gradient magnitude, $mag(x,y)$, and orientation, $\theta(x,y)$, can be derived as follows:

$$mag(x,y)\sqrt{G_x(x,y)^2 + G_y(x,y)^2} \tag{1}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{G_x(x,y)}{G_y(x,y)}\right) \tag{2}$$

In order to insensitive to the relative change between background and human regions, HOG adopts the unsigned gradient, that is, $\theta(x,y)$ in the third and the forth quadrants are fold to first and the second quadrants, respectively. Therefore, the range of $\theta(x,y)$ is in the interval [0, 180].

In the step of accumulated histogram, we equal-partition the range of the $\theta(x,y)$ into nine uniform sub-intervals $[20(k-1), 20(k)]$, where $k = 1,2,\cdots 9$. Each sub-interval is called a bin. For every pixel (x,y) in a cell, the corresponding bin $\left\lfloor \frac{\theta(x,y)}{20} \right\rfloor + 1$ is accumulated by a value $mag(x,y)$. The accumulation of all pixels in a cell results in a 9-D histogram which is referred to as 9-D HOG. After that, the four 9-D histograms are concatenated to form a 36-D HOG. Finally, the resulting 36-D HOG histogram is further normalized to uniform range [0.0, 1.0] in order to be invariant to block size. In this work, L2-norm is chosen in the normalization process.

From implementation consideration, the five steps to compute a HOG for describing a block are divided two modules, gradient module and histogram module. The computation time percentage of these five steps with pure software implementation in PC is shown in Figure 2. From Figure 2, gradient orientation submodule requires 55.6% of the software computation time, more than a half of total computation time because of the square, square-root, and trigonometric function calculation. Histogram normalization submodule requires the least

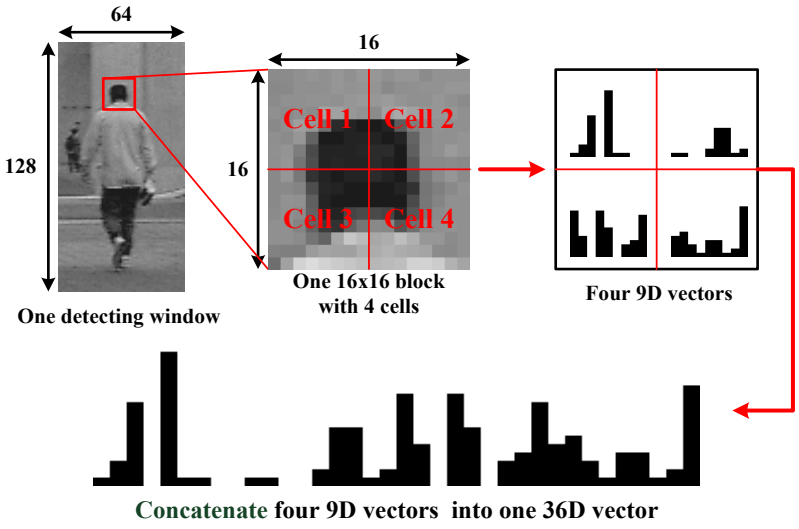


Figure 1. A 36-DHOG feature descriptor for one block.

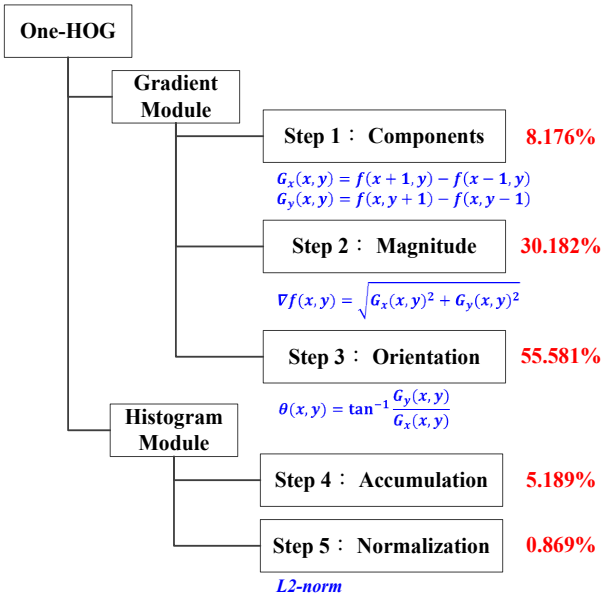


Figure 2. Software submodules of computing one HOG and their computation-time.

time, about 0.9% of total time; therefore, in this study, it is not realized in hardware.

2.2. Human Classifier

Besides hardware realization for HOG computation, this study proposes a method to enhance the computation speed by decreasing the number of times to compute HOGs in describing the detection window. In Dalal's approach, a detection window is divided into 105 overlapped rectangular blocks and each of which is represented by the aforementioned 36-D HOG feature vector. Such 105 36-D vectors could be concatenated into a 3780-D feature vector and then classi-

fied with the classifier named linear support vector machine (SVM) learned from training stage to determine the presence of the human in a detection window. To speed up the detection process, Zhu [2] applied the AdaBoost algorithm to select a set of B effective blocks to form a human classifier called HOG-AdaBoost human classifier. Each selected block is associated with a weight and a linear SVM for classifying the 36-D block HOG to human or non-human and is generally referred to as the weak classifier in the boosting literature. The classifier that combines these weak classifiers by weighted summation of the prediction results from all LSVMs. Apparently, the number of HOG feature vectors calculated with Zhu's algorithm [2] is about 4.56 times less than that it with Dalal's algorithm [1]. The efficiency (speed) is therefore enhanced. Nevertheless, all the linear SVMs of weak classifiers have to be performed; this significantly increases the computation complexity.

To alleviate this, a new method called Histograms-of-Oriented-Gradients AdaBoost Long-Feature-Vector (HOG-AdaBoost-LFV) Human Classifier is proposed. The idea behind is to use one LSVM for all HOGs descriptors from the selected B blocks but the weighting values. The feature vectors of the acquired B blocks are concatenated into a $36\times B$-Dlong vector. Then, a linear SVM is trained for further human/non-human discrimination. The flowchart and how to process one detecting window of the presented HOG-AdaBoost-LFV human classifier are shown in **Figure 3**.

3. FPGA-Based HOG Accelerator

The designed One-HOG hardware accelerator is operated on FPGA of Xilinx Spartan-6 LX-150T, and the hardware architecture is shown in **Figure 4**. The module of One-HOG operation is called One-HOG Vector Generator Module. The calculation of gradient module corresponds to Steps 1 to 3 of HOG in **Figure 2**, and histogram module corresponds to Step 4. ARM_Mem_Interface (AM_IF) module is responsible for the data transmission between ARM processor and FPGA. Data are transmitted to mem 1 from ARM processor block by block through the module, then, the start signal is sent to inform One-HOG Vector Generator Module starting the operation. After completing the operation and storing 36-D vector, to mem 2, the operation completion signal would trigger AM_IF module returning the data to ARM processor. In the following sections, we will detail the design of the gradient module as well as histogram module.

3.1. Gradient Module

Gradient module contains three submodules, gradient component, component-to-magnitude, and component-to-orientation, with the functions of calculating the components of a gradient, transferring the square root of gradient components into a magnitude, and introducing the orientation of a gradient.

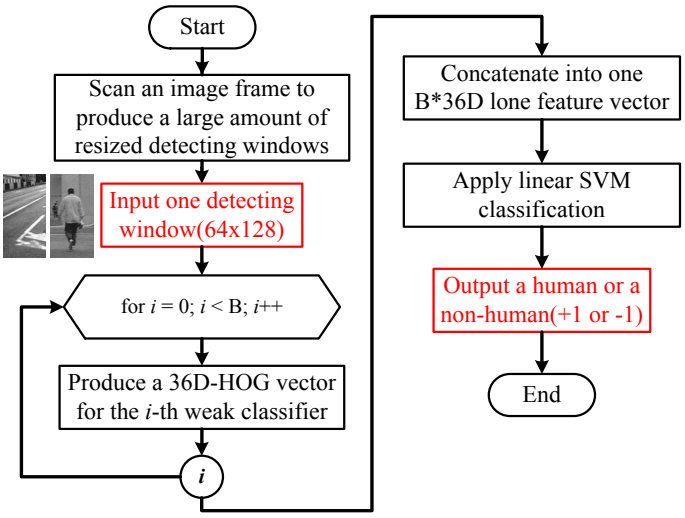


Figure 3. Flowchart of the proposed HOG-AdaBoost-LFV classifier.

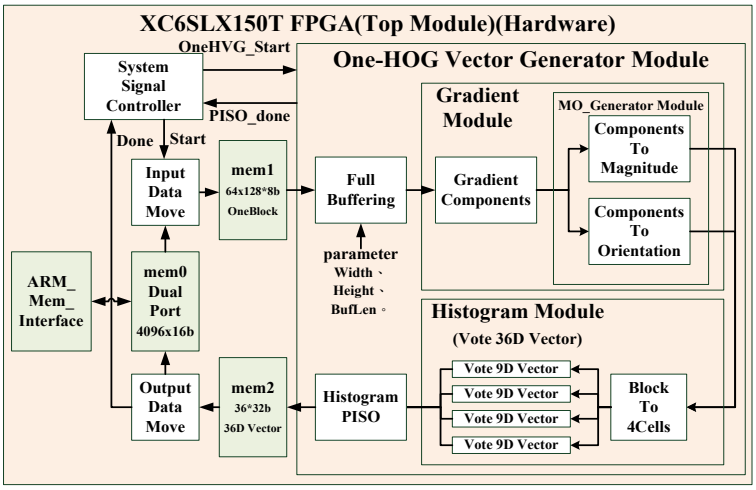


Figure 4. The hardware architecture for One-HOG accelerator.

3.1.1. Full Buffering and Gradient Components

Gradient components submodule functions to calculate horizontal and vertical gradient components, G_x and G_y. From the selected convolution masks [−1, 0, 1] and $[-1, 0, 1]^{\mathrm{T}}$, the intensities of four neighbours need to be operated. Moreover, full buffering scheme [20-21, 3] is utilized, in which shift register operates serial in or parallel out, and block pixels pre-stored in mem 1 are input the buffering circuit with one byte data at one time reading (input). After the latency, when shift register is fully stored temporary data, four bytes data are parallel output to components submodule for calculation. The function of the gradient components submodule is to deduct four bytes input data with two substractors and then output two bytes data, dataOut_X and dataOut_Y, namely, G_x and G_y. Finally, the results are directly input the successive two submodules, component-to-magnitude submodule and component-to-orientation submodule. The circuits of the full buffering scheme and the components submodule are combined, as shown in **Figure 5**.

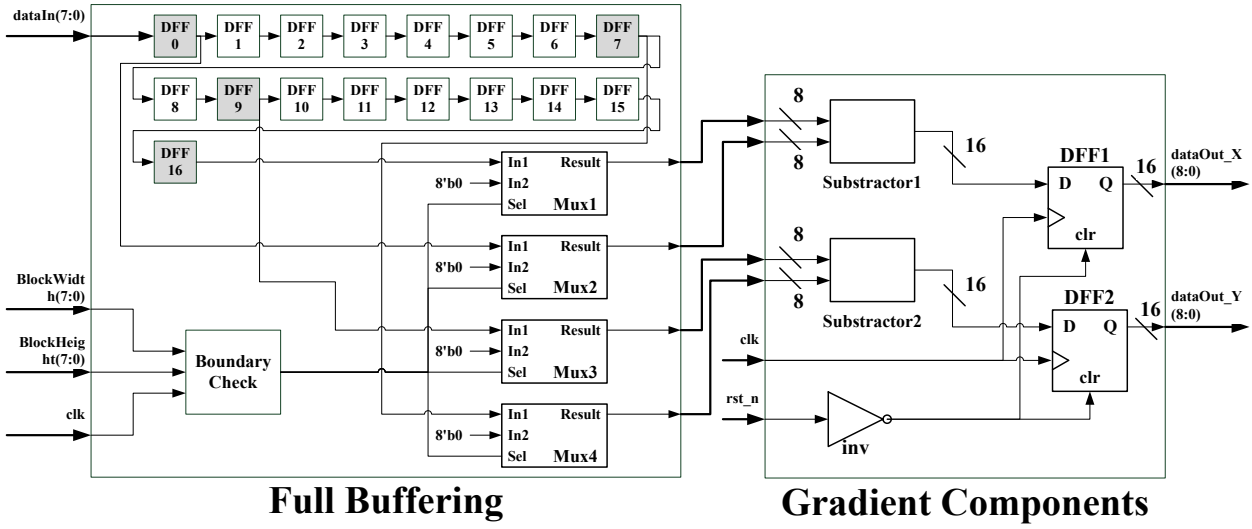


Figure 5. The combined circuit of the full buffering scheme and the gradient-to-component design.

Furthermore, convolution would not be able to operate on the boundary pixels surrounding an image. The shrinking method used in this study solves such a problem by adding boundary checking circuit. When pixels locate on boundary pixels, the four data output 0. The number of shift registers and the latency in clock cycle of such small circuits are twice the block width plus 1, e.g. 33 shift registers and 33 clock-cycle latency are required for block width 16.

3.1.2. Gradient Magnitude

The function of component-to-magnitude submodule is to take the square-root of the sum of the square of G_x and G_y and its circuit design is shown in **Figure 6(a)**. Because the gradient magnitude computation only regards the magnitudes of G_x and G_y no matter when they are positive or negative, only the magnitudes of G_x and G_y are sent to the inputs of two multipliers for operating the square of G_x and the square of G_y The results are further sent to the inputs of an adder. Then, the sum acquired from the adder is entered the square-root module generated with Xilinx ISE Core Generator. Moreover, two DFFs are used for controlling the signal flow of the data path. The gradient-to-magnitude hardware is merely calculated to the integer, which is the major factor in errors between output HOG vector and pure software calculated result.

3.1.3. Gradient Orientation

The function of component-to-orientation submodule is to calculate the gradient orientation at each pixel and classify the result to one of the 9 orientation bins. To have an identical angle accuracy calculated with hardware module to that calculated with software, a large amount of hardware costs would be required. Fortunately, the calculation of classifying the results to the corresponding orientation bins does not require so accurate angles. The successive circuit processing is to accumulate all values in the same bin. Since unsigned gradient is

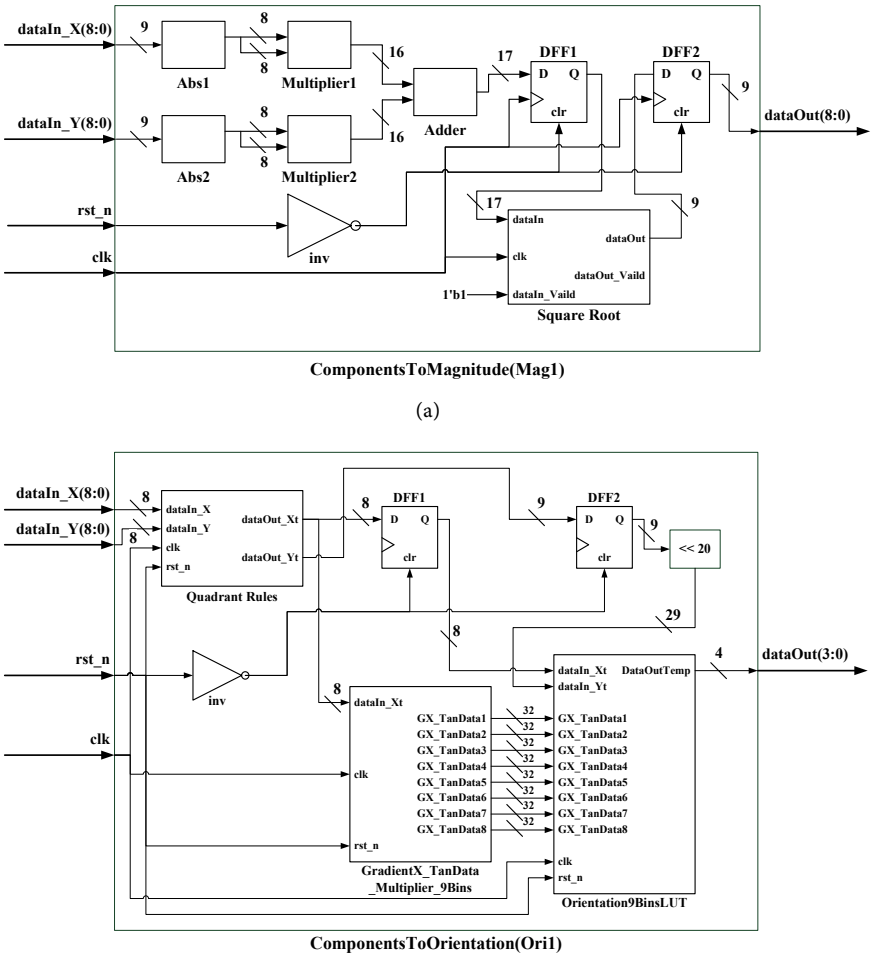


Figure 6. Circuit diagram of the component-to-magnitude (a) and component-to-orientation (b) submodules.

used, the orientation angle would appear in [0, 180] in degree. Using 20 degrees as an interval to divide bins, total 9 bins are divided.

As the calculation of gradient orientation requires complicated trigonometric function operation, table-lookup is selected for the implementation and its equation is expressed as:

$$\tan\left(\theta_i\right) \le \frac{G_y\left(x,y\right)}{G_x\left(x,y\right)} < \tan\left(\theta_{i+1}\right) \tag{3}$$

However, (3) is directly used for the implementation, dividers are required for circuits that a large amount of hardware resources would be spent for the circuit design. Instead of (3), (4) is used for the calculation so that merely multipliers are required for the implementation. Nevertheless, G_x in (4) needs to be positive, otherwise, the judgment in the equation would be changed. To keep positive G_x without affecting the table-lookup result, (5) is further derived, where the operation would be the same no matter the minus sign appears on denominator or numerator.

$$G_x(x,y) * \tan(\theta_i) \le G_y(x,y) < G_x(x,y) * \tan(\theta_{i+1}) \tag{4}$$

$$\begin{cases} \tan^{-1}\dfrac{+G_y(x,y)}{-G_x(x,y)} = \tan^{-1}\dfrac{-G_y(x,y)}{+G_x(x,y)} \\ \tan^{-1}\dfrac{-G_y(x,y)}{-G_x(x,y)} = \tan^{-1}\dfrac{+G_y(x,y)}{+G_x(x,y)} \end{cases} \tag{5}$$

Nonetheless, tan(θ) in (5) would appear decimal when establishing the table that it is adverse to the circuit design of FPGA. In this case, all numerical values have to be magnified to integers for the actual circuit design. Moreover, register shifting is used for magnifying all numerical values 2^n times in order to reduce multipliers. Here, shift registers are used for magnifying the numerical value with 2^{20} = 048576 magnification so that all circuit operations become integer operations. The value of tan(θ) is shown in Table 1. As the ratio of accumulation deflection designed with above mentioned method is 0%, the results operated with the completed gradient orientation hardware module are identical to those operated with pure software that there is no error. After integrating (4) with Table 1, the decision rules in Table 2 could be acquired.

Because the range of bin 5 spans across the first and the second quadrants, we need two different rules for specifying the classifying in bin 5 for the gradient orientations are in the first and the second quadrants, respectively. In the table, $2^{20} \times G_y$ is designed with shifting, and the achievement of decision rules is implemented by designing the hardware circuit submodule of a look-up table.

Figure 6(b) shows the circuit diagram of orientation submodule, where Quadrant Rules submodule is used for solving aforementioned problem of G_x being positive. The treated G_x is input to the GradientX_TanData_Multiplier_9Bins submodule, through dataOut_Xt pin, for multiplication with tan(θ). Meanwhile, the treated G_y is displaced 20 bits (<<20) to the left through data Out_Yt pin and input to the Orientation 9Bins LUT submodule for classifying the output to the bin interval after the table-lookup judgment.

3.2. Histogram Module

After acquiring gradient magnitude and the bins from gradient orientation, histogram module is used for generating a 36-D feature vector. Figure 7 presents the circuit scheme of the histogram accumulation module, in which BlockTo4Cells circuit submodule functions to judge the input gradient magnitude and gradient orientation bin being in one of the 4 cells. The combined signals of OrientationBin (4 bits) and Magnitude (16 bits) are delivered, according to the location of cell, to the successive 4 vector circuits, Vote9Dvector1 to Vote9Dvector4, through 1-to-4 de-multiplexer for the accumulation.

The circuit design of Cell Localizer is based on a counter and two hard-wired parameters, *CL_P*1 and *CL_P*2. The values of *CL_P*1 and *CL_P*2 are calculated according to (6) and (7), respectively.

$$CL\_P1 = \text{Block\_Width} \times (\text{Block\_Height} \gg 1) \tag{6}$$

Table 1. The original floating values and the resized integers of tan(θ).

tan(θ)	Value	2^{20}
tan(0˚)	0	0
tan(20˚)	0.36397	381,650
tan(40˚)	0.839099	879,859
tan(60˚)	1.73205	1,816,186
tan(80˚)	5.671281	5,946,770
tan(100˚)	−5.67128	−5,946,770
tan(120˚)	−1.73205	−1,816,186
tan(140˚)	−0.8391	−879,859
tan(160˚)	−0.36397	−381,650
tan(180˚)	0	0

Table 2. Lookup table of the decision rules and the corresponding bins.

Decision rule	Bin
$0 \le 2^{20}*Gy < 381{,}650*Gx$	1
$381{,}650*Gx \le 2^{20}*Gy < 879{,}859*Gx$	2
$879{,}859*Gx \le 2^{20}*Gy < 1{,}816{,}186*Gx$	3
$1{,}816{,}186*Gx \le 2^{20}*Gy < 5{,}946{,}770*Gx$	4
$5{,}946{,}770*Gx \le 2^{20}*Gy$ or $2^{20}*Gy < -5{,}946{,}770*Gx$	5
$-5{,}946{,}770*Gx \le 2^{20}*Gy < -1{,}816{,}186*Gx$	6
$-1{,}816{,}186*Gx \le 2^{20}*Gy < -879{,}859*Gx$	7
$-879{,}859*Gx \le 2^{20}*Gy < -381{,}650*Gx$	8
$-381{,}650*Gx \le 2^{20}*Gy < 0$	9

$$CL\_P2 = \text{Block\_Width} \gg 1 \tag{7}$$

When the first set of gradient magnitude and gradient orientation bin is input, the counter starts counting from 0. When the counter value is smaller than *CL_P*1, the set of data is either Cell_1 or Cell_2; otherwise, the set of data is Cell_3 or Cell_4. Furthermore, *Block_Width* is taken the remainder with the counter value; when it is smaller than *CL_P*2, the set of data is either Cell_1 or Cell_3; otherwise, it is Cell_2 or Cell_4. At the below part of **Figure 8**, the sub-module consisting of four Vote9DVector assigns Magnitude to correct bin slot for accumulation according to the input OriBin (Orientation Bin). The function of such hardware design is accomplished with de-multiplexer. That is, inputting OriBin as the Select signal of de-multiplexer, and Magnitude is regarded as the data input; the design of a Vote9Dvector circuit is completed by 9 adders and registers.

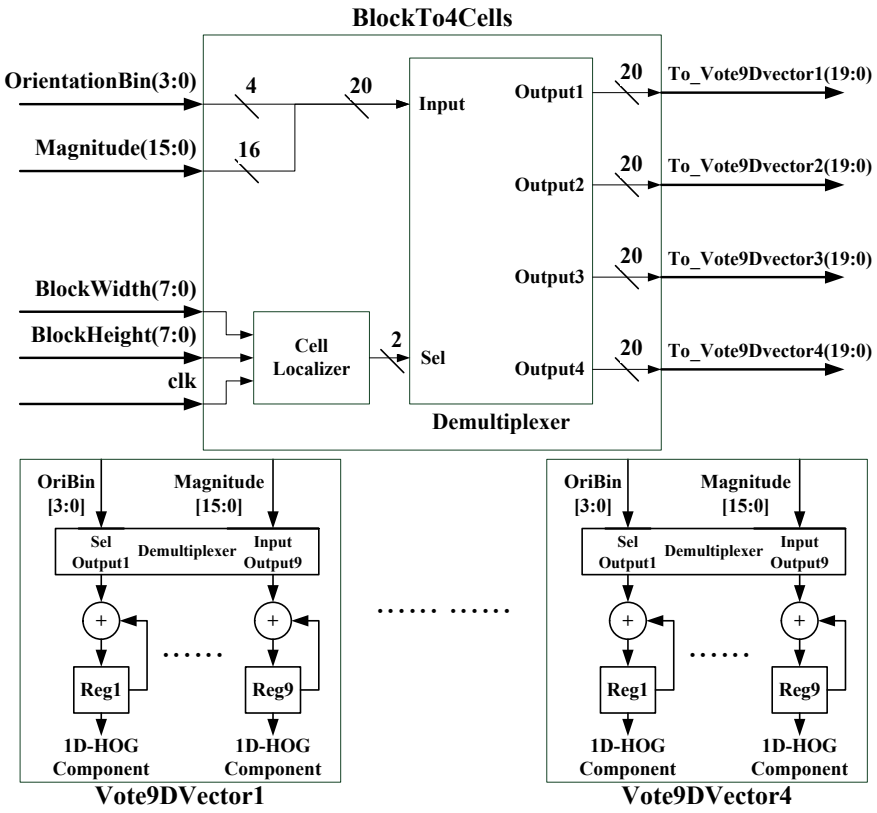
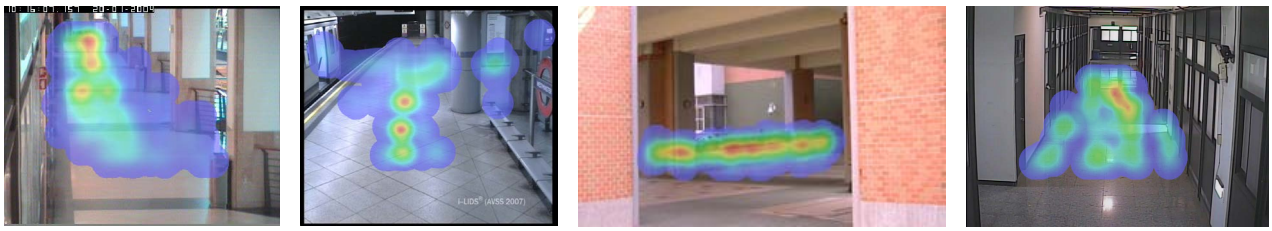


Figure 7. The presented circuit of the histogram accumulation module.



(a) Caviar (b) AVSS (c) Campus (d) Surveillance Camera

Figure 8. Heat maps for the 4 video datasets.

4. Experiment Results

For validating the proposed system for human detection, the well-known and popular datasets including three static image datasets and four video datasets are utilized for experiment. Besides, the Dalal and the proposed HOG-AdaBoost-LFV human classifiers are respectively implemented on PC and embedded platforms for performance comparison. In the following sections, we discuss the detection results for static image and video datasets, respectively.

4.1. Static Image Dataset Analysis

The three static image datasets used for validation are CBCL Pedestrian Dataset [21], CVC Virtual Pedestrian Dataset [22], and INRIA Person Dataset [23]. These three datasets are widely used in the human detection literature for per-

formance validation. The statistics of images from the three datasets for training and testing is listed in Table 3. Since CBCL does not provide non-human samples (negative samples), non-human images offered by INRIA Person Dataset are chosen as the negative samples. The experimental design concerns the results of the detecting performance of classifiers and the computation efficiency of classifiers on PC and embedded platforms. The detecting performance is measured with Positive Predictive Value (PPV), True Positive Rate (TPR), False Positive Rate (FPR), False Negative Rate (FNR), and Accuracy. The definitions of these metrics are expressed as:

$$PPV = \frac{TP}{TP + FP} \tag{8}$$

$$TPR = \frac{TP}{TP + FN} \tag{9}$$

$$FPR\left(FPPW\right) = \frac{FP}{FP + TN} \tag{10}$$

$$FNR\left(\text{Miss Rate}\right) = \frac{FN}{TP + FN} \tag{11}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{12}$$

TP (True Positive) denotes real humans being labelled humans; *TN* (True Negative) presents real non-humans being labelled non-humans; *FP* (False Positive) reveals real non-human being labelled humans; and, *FN* (False Negative) refers to real humans being labelled non-humans.

For computation efficiency analysis, two human classifiers are experimented on CBCL dataset under two different platforms, with the detection window as the unit. The experimental results are shown in Table 4. With the aid of a HOG hardware accelerator on the MaCube platform, 1051 detection windows could be computed per second for the proposed HOG-AdaBoost-LFV method that is significantly efficient than Dalal one with only 267 detection windows per second.

The performance results of all metrics for the three datasets with two human classifiers are shown in Table 5. From the experimental results in Table 5, the pure software computation with two human classifiers appears slightly different accuracy ±0.05% ~ 1.09% from the software/hardware co-design result with an HOG hardware accelerator. Dalal method presents the best performance, followed by HOG-AdaBoost-LFV method, but the difference is small.

Observing with the accuracy, the testing results between Dalal and the INRIA dataset used in this study reveal higher differences because of high inconsistency of the human posture of positive samples in INRIA sample sets resulting in merely 11 weak classifiers being selected with AdaBoost. The testing results merely show 0.04% ~ 0.66% difference between CBCL and CVC datasets. Based on the requirement and practicability to establish an embedded real-time human detection system, the proposed SW/HW Co-Design presents 3.94 times faster algorithms than Dalal and 13.14 times faster than the pure software computation of Dalal algorithms.

Table 3. Statistics for training and testing of 3 public image static datasets.

DataSets	Training/Testing	Positive Samples	Negative Samples	Total
CBCL	Training	462	1038 (INRIA)	1500
	Testing	462	1359 (INRIA)	1821
CVC	Training	571	1326	1897
	Testing	571	2048	2619
INRIA	Training	2416	4872	7288
	Testing	1132	1812	2944

Table 4. Computation efficiency of Dalal and the proposed methods based on PC and Macube embedded platforms.

Platforms	Weak Classifiers (SW/HW)	Detecting Window	PC (Desktop)	Macube (SW)	Macube (SW/HW)
Dalal	105/105	HOGs (ms)	5.314	12.283	3.591
		LSVM (ms)	0.031	0.144	0.144
		Total (ms)	5.345	12.427	3.735
		DW/Sec	187	80	267
HOG-Adaboost-LFV	33/23	HOGs (ms)	1.231	4.12	0.925
		LSVM (ms)	0.008	0.037	0.026
		Total (ms)	1.239	4.157	0.951
		DW/Sec	807	240	1051

Table 5. Detecting performance for 3 static image datasets.

Method			CBCL_V1	CBCL_V2	CBCL_V3	CBCL_V4	CVC	INRIA
Dalal	Pure SW	PPV	99.56	99.77	99.78	99.56	100	96.62
		TPR	99.78	96.96	99.13	99.78	100	83.39
		FPR (%)	0.14	0.07	0.07	0.14	0	1.82
		FNR	0.21	3.03	0.86	0.21	0	16.6
		Accuracy	99.83	99.17	99.72	99.83	100	92.49
	SW/HW Co-Design	PPV	99.56	100	99.56	99.56	100	96.80
		TPR	99.78	96.75	98.91	99.56	100	80.38
		FPR (%)	0.14	0	0.14	0.14	0	1.65
		FNR	0.21	3.24	1.08	0.43	0	19.61
		Accuracy	99.83	99.17	99.61	99.78	100	91.44
HOG-AdaBoost-LFV	Pure SW	PPV	99.35	98.68	98.91	99.13	99.47	93.88
		TPR	99.35	97.18	98.48	99.13	100	59.71
		FPR (%)	0.22	0.44	0.36	0.29	0.14	2.42
		FNR	0.64	2.81	1.51	0.86	0	40.28
		Accuracy	99.67	98.95	99.34	99.56	99.88	83.01
	SW/HW Co-Design	PPV	98.7	98.87	98.28	99.13	99.82	94.26
		TPR	99.13	95.23	99.35	99.56	100	62.45
		FPR (%)	0.44	0.36	0.58	0.29	0.04	2.37
		FNR	0.86	4.76	0.64	0.43	0	37.54
		Accuracy	99.45	98.51	99.39	99.67	99.96	84.10

4.2. Static Image Dataset Analysis

Four video datasets of Caviar video dataset [24], underground surveillance video dataset [25] form AVSS, the campus video self-shot with a camera, and the video acquired from the surveillance camera self-installed on the laboratory hall are utilized for the experiment. Since there is no existing training sample in the video datasets, positive and negative samples are picked and segmented from such videos. The statistics of the 11 videos from 4 video datasets are summarized as Table 6. In the video dataset experiment, two human classifiers, which independently contain pure software computation and software/hardware co-design, are applied so that there are 4 experimental results for each video.

Two metrics used for evaluating the performance of video datasets are Detection Rate (DR) and False Rate (FR) that are defined as:

$$\text{Detection Rate} = \frac{\text{Detected Humans}}{\text{Total Humans}} \tag{13}$$

$$\text{Flase Rate} = \frac{\text{False Labeled DWs}}{\text{Total Labeled DWs}} \tag{14}$$

In comparison with the experimental results as shown in Table 7, the performance of video datasets and static image datasets are consistent. The proposed modified algorithm and HOG-AdaBoost-LFV do not appear large differences in the performances from the original Dalal algorithm. That is, by averaging Detection Rate (DR) and False Rate (FR) of 11 videos, the average DR and the average FR present 66.73% and 10.39%, respectively, with Dalal method, while the average DR 68.55% and the average FR 11.51% are shown with the proposed methods.

4.3. Crowd Density Estimation

In addition to human detection, the crowd densities of the scenes in the video datasets are also estimated by using the proposed HW/SW co-design system. The estimation of the crowd density is from the frame-by-frame statistics of the detection results and is illustrated in the form of heat map. The 11 videos of the previously mentioned 4 video datasets are computed with the human detection system and the crowd densities of all videos are estimated as well. Figure 8 shows four heat maps of the video datasets. The red area in the figure stands for the area which longer crowd stays in or more individual pedestrian passing by. By analyzing the pedestrian flow distribution with such a way, the labor cost could be largely reduced. The corresponded platform of HW/SW co-design system is illustrated in Figure 9.

5. Conclusion

To promote the human detection system from the PC-based platform to a real-time embedded one, this paper designs and realizes One-HOG hardware accelerator operated on FPGA to accelerate the speed of computing HOG feature. The designed One-HOG accelerator mainly consists of gradient module

Table 6. Statistics of 11 videos from 4 video datasets.

DataSets	Resolution	FPS	Video	No. of Frames	No. of Pedestrians
Caviar	384 × 288	25	Caviar Test Video 1	111	3 ~ 6
2007 AVSS	720 × 576	25	AVSS Test Video 1	298	0 ~ 2
			AVSS Test Video 2	214	0 ~ 4
			AVSS Test Video 3	295	1 ~ 3
			AVSS Test Video 4	266	0 ~ 2
Campus	1280 × 720	5	Campus Test Video 1	120	0 ~ 6
			Campus Test Video 2	120	0 ~ 8
			Campus Test Video 3	120	1 ~ 6
Surveillance Camera	704 × 480	30	Ours Test Video 1	360	4
			Ours Test Video 2	360	2 ~ 4
			Ours Test Video 3	360	3 ~ 5

Table 7. Performance results for video datasets.

DataSets	Video	Method	Dalal		HOG-AdaBoost-LFV	
			DR (%)	FR (%)	DR (%)	FR (%)
Caviar	Caviar Test Video 1	Pure SW	51.52	10.2	52.28	6.62
		SW/HW Co-Design	53.99	9.92	50.76	11.47
2007 AVSS	AVSS Test Video 1	Pure SW	90.62	11.76	90.62	17.24
		SW/HW Co-Design	93.75	14.54	87.5	19.35
	AVSS Test Video 2	Pure SW	88.88	15.07	97.53	20.14
		SW/HW Co-Design	88.88	11.53	93.82	14.28
	AVSS Test Video 3	Pure SW	82.24	28.37	95.32	31.3
		SW/HW Co-Design	85.04	28.75	96.26	21.56
	AVSS Test Video 4	Pure SW	89.04	15.87	90.41	18.04
		SW/HW Co-Design	90.41	16.66	93.15	22.54
Campus	Campus Test Video 1	Pure SW	68.48	18.12	63.33	18.81
		SW/HW Co-Design	68.78	18.53	66.66	21.23
	Campus Test Video 2	Pure SW	53.96	2.89	54.71	2.22
		SW/HW Co-Design	55.28	3.66	53.77	5.27
	Campus Test Video 3	Pure SW	49.5	3.8	46.5	4.4
		SW/HW Co-Design	51.09	2.63	49.1	6.61
Surveillance Camera	Ours Test Video 1	Pure SW	55.2	3.88	55.2	2.79
		SW/HW Co-Design	54.86	4.39	57.63	3.24
	Ours Test Video 2	Pure SW	59.37	0.7	60.26	3.31
		SW/HW Co-Design	58.48	0.7	60.26	0
	Ours Test Video 3	Pure SW	45	2.98	47.3	1.45
		SW/HW Co-Design	43.84	3.81	45.76	1.48

Figure 9. The developed system of HW/SW co-design for application to crowd density estimation and analysis.

and histogram module. Gradient module which aims at computing gradient magnitude and orientation contains three submodules: gradient component, component-to-magnitude, and component-to-orientation. In the orientation computation, a table-lookup is selected for the implementation so that the complicated trigonometric function operation can be avoided. After acquiring gradient magnitude and the bins from gradient orientation, histogram module is used for generating a 36-D feature vector through 1-to-4 de-multiplexer for the accumulation. To further reduce the times to compute HOG, only a set of discriminative blocks is selected by AdaBoost algorithm for describing the human appearance. Our experiments show that the effectiveness and efficiency of the proposed SW/HW co-design system are validated through the use of three static image and four video datasets. The computation speed is 13.14 faster than pure software computation of Dalal algorithm but only with less than 1% accuracy deterioration. Furthermore, the proposed system is applied to estimate the crowd density and achieves the real-time performance.

Acknowledgements

This work was supported in part by the Ministry of Science and Technology (MOST) under Project MOST 103-2221-E-390-028-MY2, MOST 105-2221-E-390-024-MY3, and MOST 106-2221-E-327-028.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Dalal, N. and Triggs, B. (2005) Histograms of Oriented Gradients for Human Detection. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, **1**, 886-893.

[2] Zhu, Q., Avidan, S., Yeh, M.C. and Cheng, K.T. (2006) Fast Human Detection Using a Cascade of Histograms of Oriented Gradients. *IEEE Computer Society Confe-*

rence on Computer Vision and Pattern Recognition, **2**, 1491-1498.

[3] Rabia Rauf, A.R., Shahid, S.Z. and Asad, A.S. (2016) Pedestrian Detection Using HOG, LUV and Optical Flow as Features with AdaBoost as Classifier. 6*th International Conference on Image Processing Theory Tools and Applications* (*IPTA*), Oulu, 12-15 December 2016, 1-4.

[4] Gajjar, V., Gurnani, A. and Khandhediya, Y. (2017) Human Detection and Tracking for Video Surveillance: A Cognitive Science Approach. *IEEE International Conference on Computer Vision Workshop* (*ICCVW*), Venice, 22-29 October 2017, 2805-2809. https://doi.org/10.1109/ICCVW.2017.330

[5] Cao, T.P., Elton, D. and Deng, G. (2012) Fast Buffering for FPGA Implementation of Vision-Based Object Recognition Systems. Journal of Real-*Time Image Processing*, **7**, 173-183. https://doi.org/10.1007/s11554-011-0201-1

[6] Moranduzzo, T. and Melgani, F. (2014) Detecting Cars in UAV Images with a Catalog-Based Approach. *IEEE Transactions on Geoscience and Remote Sensing*, **52**, 6356-6367. https://doi.org/10.1109/TGRS.2013.2296351

[7] Madhogaria, S., Baggenstoss, P.M., Schikora, M., Koch, W. and Cremers, D. (2015) Car Detection by Fusion of HOG and Causal MRF. *IEEE Transactions on Aerospace and Electronic Systems*, **51**, 575-590. https://doi.org/10.1109/TAES.2014.120141

[8] Ilas, M.-E. (2017) New Histogram Computation Adapted for FPGA Implementation of HOG Algorithm: For Car Detection Applications. 9*th Computer Science and Electronic Engineering* (*CEEC*), Colchester, 27-29 September 2017, 77-82. https://doi.org/10.1109/CEEC.2017.8101603

[9] Anirudh, B.K., Vivek, V., Abhishek, R.K. and Sumam, D.S. (2017) Accelerating Real-Time Computer Vision Applications Using HW/SW Co-Design. *International Conference on Computer, Communications and Electronics* (*Comptelix*), Jaipur, 1-2 July 2017, 458-463. https://doi.org/10.1109/COMPTELIX.2017.8004013

[10] Bartosz, M., Tomasz, K. and Marek, G. (2017) Embedded Vision System for Pedestrian Detection Based on HOG+SVM and Use of Motion Information Implemented in Zynq Heterogeneous Device. *Signal Processing: Algorithms, Architectures, Arrangements, and Applications* (*SPA*), Poznan, 20-22 September 2017, 406-411.

[11] Yu, Z., Yang, S., Sillitoe, I. and Buckley, K. (2017) Towards a Scalable Hardware/Software Co-Design Platform for Real-Time Pedestrian Tracking Based on a ZYNQ-7000 Device. *IEEE International Conference on Consumer Electronics-Asia* (*ICCE-Asia*), Bengaluru, 5-7 October 2017, 127-132. https://doi.org/10.1109/ICCE-ASIA.2017.8307853

[12] Kadota, R., Sugano, H., Hiromoto, M., Ochi, H., Miyamoto, R. and Nakamura, Y. (2009) Hardware Architecture for HOG Feature Extraction. *Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Kyoto, 12-14 September 2009, 1330-1333. https://doi.org/10.1109/IIH-MSP.2009.216

[13] Hemmati, M., Biglari-Abhari, M., Berber, S. and Niar, S. (2014) HOG Feature Extractor Hardware Accelerator for Real-Time Pedestrian Detection. *Euromicro Conference on Digital System Design* (*DSD*), Verona, 27-29 August 2014, 543-550. https://doi.org/10.1109/DSD.2014.60

[14] Niar, M., Miyajima, T. and Amano, H. (2015) Data Reduction and Parallelization for Human Detection System. *Workshop on Synthesis and System Integration of Mixed Information Technologies*, Yilan, 16-17 March 2015, 134-139.

[15] Bauer, S., Brunsmann, U. and Schlotterbeck-Macht, S. (2009) FPGA Implementation of a HOG-Based Pedestrian Recognition System. *MPC Workshop*, Karlsruhe, July 2009, 49-58.

[16] Bauer, S., Kohler, S., Doll, K. and Brunsmann, U. (2010) FPGA-GPU Architecture for Kernel SVM Pedestrian Detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, San Francisco, CA, 13-18 June 2010, 61-68. https://doi.org/10.1109/CVPRW.2010.5543772

[17] Chen, P.Y., Huang, C.C., Lien, C.Y. and Tsai, Y.H. (2014) An Efficient Hardware Implementation of HOG Feature Extraction for Human Detection. *IEEE Transactions on Intelligent Transportation Systems*, **15**, 656-662. https://doi.org/10.1109/TITS.2013.2284666

[18] Suleiman, A. and Sze, V. (2014) Energy-Efficient HOG-Based Object Detection at 1080HD 60 fps with Multi-Scale Support. *IEEE Workshop on Signal Processing Systems* (*SiPS*), Belfast, UK, 20-22 October 2014, 1-6. https://doi.org/10.1109/SiPS.2014.6986096

[19] Mizuno, K., Terachi, Y., Takagi, K., Izumi, S., Kawaguchi, H. and Yoshimoto, M. (2012) Architectural Study of HOG Feature Extraction Processor for Real-Time Object Detection. *IEEE Workshop on Signal Processing Systems* (*SiPS*), Quebec City, 17-19 October 2012, 197-202. https://doi.org/10.1109/SiPS.2012.57

[20] Takagi, K., Mizuno, K., Izumi, S., Kawaguchi, H. and Yoshimoto, M. (2013) A Sub-100-Milliwatt Dual-Core HOG Accelerator VLSI for Real-Time Multiple Object Detection. *IEEE International Conference on Acoustics, Speech and Signal Processing* (*ICASSP*), Vancouver, BC, 26-31 May 2013, 2533-2537. https://doi.org/10.1109/ICASSP.2013.6638112

[21] CBCL Pedestrian Database. http://cbcl.mit.edu/software-datasets

[22] Marín, J., Vázquez, D., Gerónimo, D. and López, A.M. (2010) Learning Appearance in Virtual Scenarios for Pedestrian Detection. *IEEE International Conference on Computer Vision and Pattern Recognition*, San Francisco, 13-18 June 2010, 137-144.

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

[24] ECFunded CAVIAR Project/IST 2001 37540. http://homepages.inf.ed.ac.uk/rbf/CAVIAR

[25] I-LIDS Datasets for 2007 Advanced Video and Signal Based Surveillance (AVSS). http://www.eecs.qmul.ac.uk/~andrea/avss2007_d.html