An Efficient Hardware Implementation of HOG Feature Extraction for Human Detection

Pei-Yin Chen, Chien-Chuan Huang, Chih-Yuan Lien, and Yu-Hsien Tsai

***Abstract*—In intelligent transportation systems, human detection is an important issue and has been widely used in many applications. Histograms of oriented gradients (HOG) are proven to be able to significantly outperform existing feature sets for human detection. In this paper, we present a low-cost high-speed hardware implementation for HOG feature extraction. The simulation shows that the proposed circuit can achieve 167 MHz with 153-K gate counts by using Taiwan Semiconductor Manufacturing Company 0.13-μm technology. Compared with the previous hardware architectures for HOG feature extraction, our circuit requires fewer hardware costs and achieves faster working speed.**

***Index Terms*—Hardware implementation, histograms of oriented gradients (HOG), object detection.**

I. Introduction

RECENTLY, human detection has been an important issue and has been widely used in many applications, such as surveillance, automotive systems, and robotics. A robust human detection system in intelligent transportation systems is desired by people and becomes essential to industries. However, there still are many challenges to attain ideal human detection, such as the diversity of object appearance and the interference of an image due to light changing. These challenges make human detection become more difficult and unreliable.

Many researchers have proposed diverse techniques to deal with object detection [1]–[7]. Generally speaking, object detection includes two significant processes, i.e., feature extraction and classification. There are various feature extractions, such as Harr wavelets [2], [3], scale-invariant feature transform descriptors [4], Gabor filters [5], shape contexts [6], histograms of oriented gradients (HOG) [7], etc. For the present, HOG is proven to offer better feature extraction that could significantly outperform existing feature sets for object detection [7], particularly with regard to pedestrians. However, the calculation of the HOG features is computationally complicated. In this paper, we focus only on the feature extraction method using HOG for real-time applications. For simplicity, a linear support vector machine (SVM) is used as a baseline classifier throughout the study.

For real-time surveillance and safety applications in intelligent transportation systems, high-speed recognition for object detection is necessary and must be considered. So far, some field-programmable gate array (FPGA) implementations [8]–[16] have been proposed for real-time applications. Cao and Deng [8] realized an FPGA implementation with the best performance compared with other implementations. However, this study particularly targets stop-sign detection. In [9], Bauer *et al.* described the implementation of a real-time pedestrian recognition system that combines FPGA-based extraction of image features with a CPU-based object localization and classification framework. Hiromoto and Miyamoto exploited a high degree of fine-grained parallelism and adopted an efficient histogram generator combined with a linear SVM classifier [10]. Kadota *et al.* [11] proposed several methods to simplify the computation, such as conversion of the division and square root, and implemented the proposed architecture on FPGA to achieve real-time requirements. However, in order to reduce the hardware cost, the simplification used in [11] greatly degraded the accuracy rate for human detection. An FPGA hardware with a target-reconfigurable object detector by joint HOG was proposed in [12]. Negi *et al.* [13] proposed a deep pipelined one-chip FPGA implementation by using binary-patterned HOG features, AdaBoost classifiers, and some approximation arithmetic strategies. In [14], Mizuno *et al.* proposed a simplified HOG algorithm with cell-based scanning and simultaneous SVM calculation. A hardware accelerator (HOG-ACC) for multiprocessor system-on-chip (MPSoC) to accelerate HOG descriptor extraction was proposed in [15]. In [16], the authors reuse the features in blocks to construct the HOG features for intersecting detection windows and utilize subcell-based interpolation to efficiently compute the HOG features for each block.

The HOG circuit may be applied in end-user camera equipment or equipped in intelligent transportation systems; hence, how to implement it with a lower hardware cost is an important issue. In previous research, some studies used rough equations to replace the complicated operations, and some focused on reducing the input data to meet the real-time requirements. However, those methods had low accuracy rates. In this paper, we present a pipelined architecture for HOG feature extraction. By using approximation methods to replace the complex

Manuscript received June 14, 2013; revised September 18, 2013; accepted October 2, 2013. Date of publication October 28, 2013; date of current version March 28, 2014. This work was supported in part by the National Science Council of Taiwan under Grant NSC-101-2221-E-006-151-MY3, by the Ministry of Economic Affairs (MOEA) of Taiwan under Grant MOEA 100-EC-17-A-05-S1-192, and by the Headquarters of University Advancement, National Cheng Kung University, which is sponsored by the Ministry of Education, Taiwan. The Associate Editor for this paper was P. Grisleri.

P.-Y. Chen, C.-C. Huang, and Y.-H. Tsai are with the Digital Integrated Circuit Design Laboratory, Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 70101, Taiwan (e-mail: pychen@csie.ncku.edu.tw; chien.chuan.huang@gmail.com; kikicocohaha@hotmail.com).

C.-Y. Lien is with the Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung 80778, Taiwan (e-mail: cylien@cc.kuas.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2013.2284666

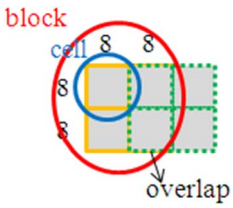

Fig. 1. Cells and blocks used for HOG feature extraction.

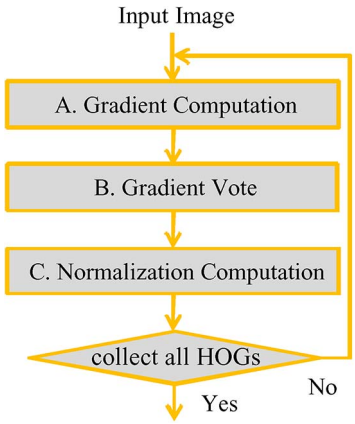

Fig. 2. Dataflow of HOG feature extraction.

operations and adopting the parallel processing design, our proposed design can be implemented with a lower cost while performing high throughput. As compared with [8]–[14], our implementation reduces the logic cells and achieves higher operating frequency for an Altera Stratix II FPGA.

The rest of this paper is organized as follows. The HOG feature extraction is introduced briefly in Section II. Section III describes the approximation methods suitable for hardware implementation and the very large scale integration (VLSI) architecture in detail. Section IV illustrates the experimental results and comparisons. The conclusion is provided in Section V.

II. HOG

This section gives an overview of the HOG feature extraction. There are two computation units in HOG feature extraction. One is the cell and the other is the block. Fig. 1 shows the relationship of cell and block units. The size of the cell is 8×8 pixels and the size of the block is 16×16 pixels. One block consists of four cells. When finishing the computation of the current block, the overlap of the next computation is 8×8 pixels. Fig. 2 summarizes the computation of HOG feature extraction. It can be divided into the following three steps.

A. Gradient Computation

For each pixel located at coordinate (x, y), the magnitude $m(x, y)$ and direction $\theta(x, y)$ need to be calculated. Assume that the pixel to be computed is located at coordinate (x, y) and its luminance value is denoted by $f(x, y)$. The gradients of the x- and y-axes, which are denoted by $f_x(x, y)$ and $f_y(x, y)$, respectively, are computed as

$$f_x(x, y) = f(x + 1, y) - f(x - 1, y) \quad (1)$$
$$f_y(x, y) = f(x, y + 1) - f(x, y - 1). \quad (2)$$

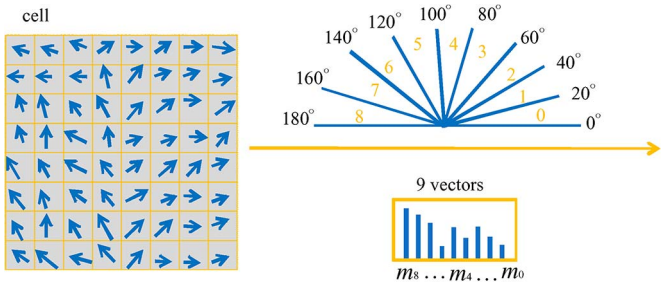

Fig. 3. Nine bins of gradient vote.

Then, the magnitude $m(x, y)$ can be given as

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

The direction $\theta(x, y)$ can be computed by

$$\theta(x, y) = \arctan \frac{f_y(x, y)}{f_x(x, y)}. \quad (4)$$

B. Gradient Vote

After obtaining the magnitude $m(x, y)$ and direction $\theta(x, y)$, each pixel within the cell calculates a gradient vote for an orientation histogram according to the orientation of the gradient element centered on it. The orientation is evenly spaced over 0°–180° and is divided into nine bins, as shown in Fig. 3. The weight of each pixel, which is denoted by α, can be computed as

$$\alpha = (n + 0.5) - \frac{b * \theta(x, y)}{\pi} \quad (5)$$

where n is the bin to which $\theta(x, y)$ belongs and b is 9, which indicates the total number of bins. To reduce aliasing, both values of two neighboring bins are incremented. The incremented values m_n and m_{nearest} can be given by

$$m_n = (1 - \alpha) * m(x, y) \quad (6)$$
$$m_{\text{nearest}} = \alpha * m(x, y) \quad (7)$$

respectively. Votes (m_0 to m_8) are accumulated into orientation bins over local spatial regions.

C. Normalization Computation

Finally, a histogram normalization computation is generated by combining all histograms belonging to one block, which consists of four cells. The normalized result can be realized as

$$v_i^n = \frac{v_i}{\sqrt{\|v\|_2^2 + \varepsilon^2}} \quad (8)$$

where i is a number from 1 to 36 (four cells $\times$ nine bins), v_i is the vector corresponding to a combined histogram for a block region, $\|v\|_2^2 = v_1^2 + v_2^2 + \ldots + v_{36}^2$, and ε is a small constant to avoid dividing by zero.

Obviously, many complex and floating-point operations are needed to determine the parameters if the direct implementation of HOG feature extraction is adopted.

TABLE I
INPUT VALUES AND OUTPUT EQUATIONS OF THE HOG FEATURE EXTRACTION PROCEDURE

	Step 1: Gradient Computation	Step 2: Gradient Vote	Step 3: Normalization Computation
Inputs	$f_x(x,y)$, $f_y(x,y)$	$\theta(x,y)$, n	v_i
Outputs	$m(x,y)=\sqrt{f_x(x,y)^2+f_y(x,y)^2}$ $\theta(x,y)=\arctan\frac{f_y(x,y)}{f_x(x,y)}$	$m_n=(1-\alpha)*m(x,y)$ $m_{nearest}=\alpha*m(x,y)$ $\alpha=(n+0.5)-\frac{b*\theta(x,y)}{\pi}$	$v_i^n=\frac{v_i}{\sqrt{\|v\|_2^2+\varepsilon^2}}$ $\|v\|_2^2=v_1^2+v_2^2+\ldots+v_{36}^2$

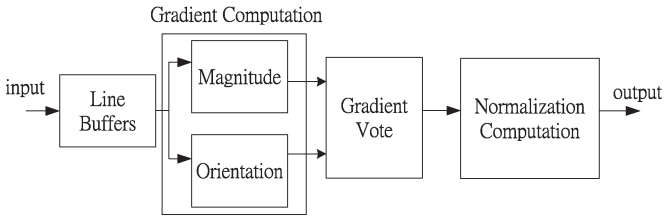

Fig. 4. Hardware architecture of HOG.

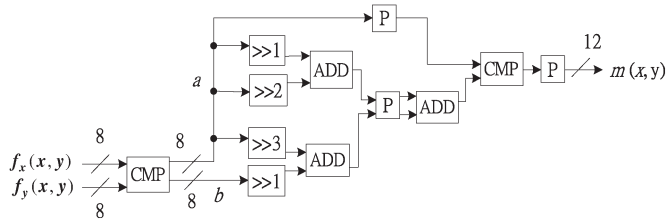

Fig. 5. Architecture of SRA.

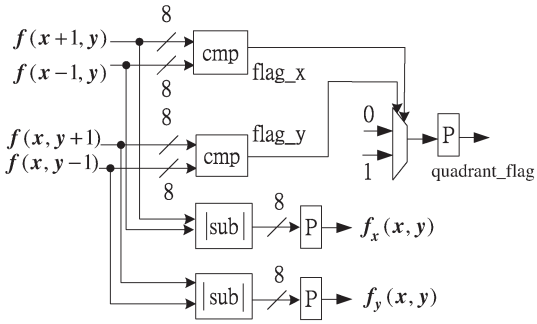

Fig. 6. Architecture of "quadrant_flag."

III. HARDWARE IMPLEMENTATION

The equations for the HOG feature extraction procedure, as shown in Fig. 2, are summarized in Table I, where $f_x(x,y)$ and $f_y(x,y)$ are denoted as the gradients of the x- and y-axes, respectively; n is the bin to which $\theta(x,y)$ belongs; and v_i is the vector corresponding to the histogram of the block. It is obvious that the calculation of HOG feature extraction is computationally complicated and unsuitable for hardware implementation. Hence, we adopt some approximate techniques, which will be mentioned later, to reduce implementation complexity and to improve extraction speed. Fig. 4 shows the hardware architecture of the proposed HOG feature extraction.

A. Gradient Computation

As shown in (3), two-square-and-one-square-root operations are required to calculate the magnitude of one target pixel. In hardware implementation, the cost for the calculation of the square and square root is high. In our design, the square root approximation (SRA) technique suggested in [17] is adopted for low-cost VLSI implementation to calculate $m(x,y)$ properly. It can be given as

$$m(x,y)=\sqrt{f_x(x,y)^2+f_y(x,y)^2}\approx\max\left((0.875a+0.5b),a\right) \quad (9)$$

where $a=\max(f_x(x,y),f_y(x,y))$ and $b=\min(f_x(x,y),f_y(x,y))$. Fig. 5 shows the architecture of the SRA, where P represents the pipelined register and ADD units generate the sum of two inputs. By adopting the SRA technique, we can use shift units easily to avoid using a multiplier operation.

Furthermore, the direction $\theta(x,y)$ in (4) needs to be calculated. In hardware implementation, the widely used method for calculating various trigonometric and transcendental functions by rotating vectors is the coordinate rotation digital computer (CORDIC) module. It performs coordinate transformation by using a series of shift-and-add operations [18], [19]. However, the drawback is that the CORDIC module is usually implemented with many iterations to obtain acceptable calculation precision with high hardware cost. In this case, since the arctangent operation is required for the computation of $\theta(x,y)$, which is used to determine the bin to which $\theta(x,y)$ belongs, we can simplify the arctangent operation by adopting the following two processes of reduction.

First, we eliminate the calculation of $\theta(x,y)$. According to the definition of trigonometric function, $\tan\theta$ can be expressed as

$$\tan\theta=\frac{f_y(x,y)}{f_x(x,y)}. \quad (10)$$

Comparing (10) with (4) and using the characteristic of $\tan\theta$, we can find that

$$\tan\theta_i\le\tan\theta<\tan\theta_{i+1}. \quad (11)$$

Rewriting $\tan\theta$ as $f_y(x,y)/f_x(x,y)$, (11) becomes

$$\tan\theta_i\le\frac{f_y(x,y)}{f_x(x,y)}<\tan\theta_{i+1}. \quad (12)$$

Thus, the bin where $\theta(x,y)$ belongs to can be calculated by satisfying the following condition:

$$f_x(x,y)*\tan\theta_i\le f_y(x,y)<f_x(x,y)*\tan\theta_{i+1}. \quad (13)$$

TABLE II
APPROXIMATE VALUES OF $\tan\theta$

tangent	Approximate value
tan0°	0
tan10°	$2^{-3}+2^{-4}$
tan20°	$2^{-2}+2^{-3}$
tan30°	$2^{-1}+2^{-4}+2^{-6}$
tan40°	$2^{-1}+2^{-2}+2^{-4}$
tan50°	$1+2^{-3}+2^{-4}$
tan60°	$1+2^{-1}+2^{-2}$
tan70°	$2+2^{-1}+2^{-2}$
tan80°	$5+2^{-1}+2^{-3}+2^{-5}$

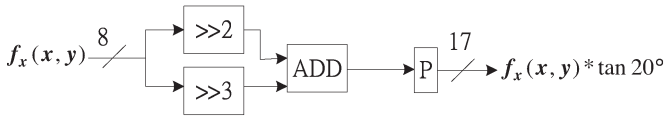

Fig. 7. Architecture of $f_x(x,y)*\tan 20°$.

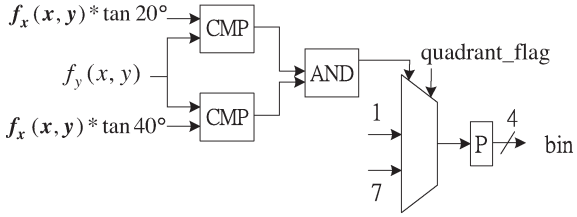

Fig. 8. Architecture of $f_x(x,y)*\tan 20° \le f_y(x,y) < f_x(x,y)*\tan 40°$.

Second, the approximated values of $\tan\theta$ are used for low-complexity hardware implementation. The $f_x(x,y)$ located at the second quadrant can be mapped into the first quadrant by $f_x(x,y) = -f_x(x,y)$. A signal "quadrant_flag" is used to further reduce the size of the lookup table, as shown in Fig. 6. In the previous step, the quadrant_flag will be determined and set as 0 for the first quadrant and 1 for the second quadrant. Hence, the lookup table could be reduced by 50%. Only $\tan 0°$–$\tan 80°$ need to be stored, as listed in Table II. Fig. 7 is the architecture of $f_x(x,y)*\tan 20°$, and Fig. 8 is the architecture used to determine bin 1 or 7, where $\theta(x,y)$ really belongs.

B. Gradient Vote

In the histogram feature extraction procedure, magnitude $m(x,y)$ and direction $\theta(x,y)$ are used to determine α, which is necessary for proportional distribution of $m(x,y)$ to reduce aliasing. However, in the previous simplified scheme, $\theta(x,y)$ is not computed accurately. Hence, we try to set α as a constant 0.5. Thus, (6) and (7) become $m_n = m(x,y) \gg 1$ and $m_{\text{nearest}} = m(x,y) \gg 1$, respectively. Section IV shows the result of pedestrian detection, when α becomes a constant, by using an SVM. The result shows that our simplification does not degrade detection accuracy. Fig. 9 is the architecture of gradient vote, where m_0 to m_8 represent the nine bins, as mentioned in Fig. 3.

C. Normalization Computation

Observing (8), we can find that the inverse square root operation is needed. In hardware implementation, the division and square root are high-complexity operations. Hence, to simplify

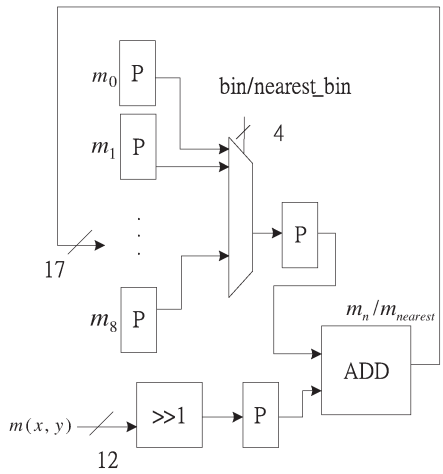

Fig. 9. Architecture of the gradient vote.

these operations, we use the Newton–Raphson method [20] to approximate the inverse square root. It is given as

$$y = \frac{1}{\sqrt{x}} \Rightarrow f(y) = \frac{1}{y^2} - x = 0. \tag{14}$$

Thus, $f(y)$ and $f'(y)$ can be rewritten as $f(y) = 1/y^2 - x$ and $f'(x) = -2/y^3$. We can obtain the approximate value, i.e., $y_{n+1} \cong y$, by repetitively doing the following equation:

$$y_{n+1} = y_n - \frac{f(y_n)}{f'(y_n)} \Rightarrow y_{n+1} = \frac{y_n * (3 - x * y_n^2)}{2}. \tag{15}$$

However, from (15), it is obvious that the initial value is important since it can reduce the iteration time used to obtain the approximate value. Hence, we adopt the magic number, i.e., 0x5f3759df, which is proposed by John Carmack in [21], to calculate the initial value. Thus, the approximate value can be calculated without doing iterations. First, we calculate y_{n_IEEE754} as

$$y_{n\_\text{IEEE754}} = (x_{\text{IEEE754}} \gg 1) - \text{magic\_number} \tag{16}$$

where x_{IEEE754} and y_{n_IEEE754} are the IEEE754 presentation [22] of values x and y_n, respectively. Then, y_{IEEE754} is rewritten as the form of decimal presentation, which is denoted by y_n. Hence, (15) becomes

$$y_{\text{approximate}} = \frac{y_n * (3 - x * y_n^2)}{2} \tag{17}$$

where $y_{\text{approximate}}$ is the approximate value of y. By this approximation, we can reduce the hardware cost and computation time used to do the iterations. For example, $1/\sqrt{25}$ can be calculated as 0.199927, where the exact value of $1/\sqrt{25}$ is 0.2. Fig. 10 shows the architecture of fast inverse square root, where DIC is the unit used to convert decimal representation to IEEE754 representation, IDC is the unit used to convert IEEE754 representation to decimal representation, and MUL units generate the product of two inputs. Table III lists the equations for the HOG feature extraction procedure, which is simplified by the previous approximation method. Compared with Table I, the high-complexity operations, which are not suitable for hardware implementation, are replaced by the approximation method, which is suitable for hardware implementation.

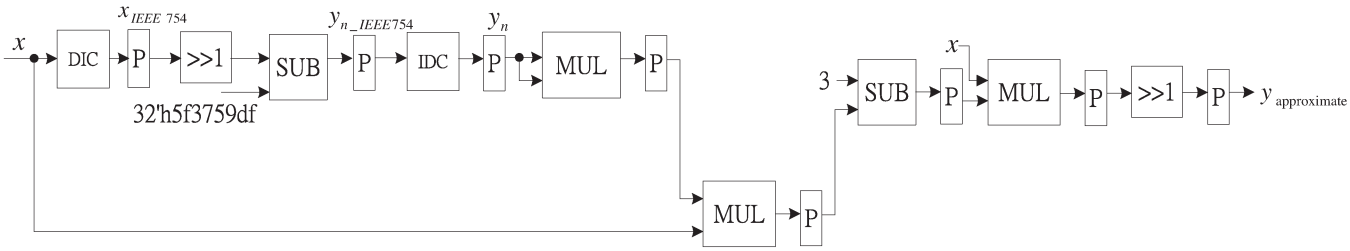

Fig. 10. Architecture of a fast inverse square root.

TABLE III
INPUT VALUES AND OUTPUT EQUATIONS OF THE HOG FEATURE EXTRACTION PROCEDURE

	Step 1: Gradient Computation	Step 2: Gradient Vote	Step 3: Normalization Computation
Inputs	$f_x(x,y)$, $f_y(x,y)$	n	v_i
Outputs	$m(x,y) \approx \max((0.875a + 0.5b), a)$, $a = \max(f_x(x,y), f_y(x,y))$, $b = \min(f_x(x,y), f_y(x,y))$, $f_x(x,y)*\tan\theta_i \le f_y(x,y) < f_x(x,y)*\tan\theta_{i+1}$	$m_n = m(x,y) >> 1$, $m_{nearest} = m(x,y) >> 1$	$v_i^n = v_i \times y_{approximate}$, $y_{approximate} = \frac{y_n*(3 - x*y_n^2)}{2}$

Fig. 11. INRIA nonperson data set [23].

Fig. 12. INRIA person data set [23].

Fig. 13. MIT pedestrian data [24].

IV. EXPERIMENTAL RESULTS AND COMPARISONS

To evaluate the performance of the proposed circuit, we choose human detection for the test application. Two well-known image data sets from the French Institute for Research in Computer and Science Control (INRIA), Paris, France [23] and Massachusetts Institute of Technology (MIT), Cambridge, MA, USA [24], are selected as the input data, as shown in Figs. 11–13. They are widely used as the research work on

TABLE IV
COMPARISON OF THE PROPOSED CIRCUIT AND THE STANDARD HOG ALGORITHM

	Proposed Circuit	HOG Algorithm
Detection Results (INRIA)	TP:538, TN:430, FP:23, FN:25	TP:543, TN:430, FP:23, FN:20
Accuracy (INRIA)	95.27%(968/1016)	95.76%(973/1016)
Detection Results (MIT)	TP:199, TN:450, FP:3, FN:1	TP:199, TN:452, FP:1, FN:1
Accuracy (MIT)	99.38%(649/653)	99.69%(651/653)

TABLE V
SIMULATION RESULTS OF THE PROPOSED CIRCUIT

Design	Proposed circuit
Process	TSMC 0.13μm
Frequency	167 MHz
Gate counts	153 k
Throughputs	6012x10^6 vectors/sec 3200x2048 pixels 1641 images/sec
Accuracy (INRIA)	95.27%(968/1016)
Accuracy (MIT)	99.38%(649/653)

TABLE VI
COMPARISON OF THE PROPOSED CIRCUIT AND PREVIOUS HARDWARE IMPLEMENTATION

	[8]	**[9]**	**[10]**	**[11]**
FPGA	Virtex-4	Spartan 3	Virtex-5	Stratix II
# of LUTs	8921	42425	28495	3794
# of registers	4221	N/A	5980	6699
Frequency (MHz)	N/A	63	167	127
	[12]	**[13]**	**[14]**	**Ours**
FPGA	Cyclone III	Virtex-5	Cyclone IV	Stratix II
# of LUTs	34838	17383	34403	3467
# of registers	22612	2181	23247	172
Frequency (MHz)	70	44	76	241

N/A：Not Available

TABLE VII
COMPARISON OF THE STANDARD HOG ALGORITHM AND THE MAGNITUDE APPROXIMATION

		INRIA	MIT
Standard Hog Algorithm	Detection Results	TP:543, TN:430, FP:23, FN:20	TP:199, TN:452, FP:1, FN:1
	Accuracy	95.76%(973/1016)	99.69%(651/653)
Approximation (magnitude)	Detection Results	TP:538, TN:429, FP:24, FN:25	TP:200, TN:453, FP:0, FN:0
	Accuracy	95.17%(967/1016)	100%(653/653)

TABLE VIII
COMPARISON OF THE STANDARD HOG ALGORITHM AND THE ORIENTATION APPROXIMATION

		INRIA	MIT
Standard Hog Algorithm	Detection Results	TP:543, TN:430, FP:23, FN:20	TP:199, TN:452, FP:1, FN:1
	Accuracy	95.76%(973/1016)	99.69%(651/653)
Approximation (magnitude)	Detection Results	TP:543, TN:430, FP:23, FN:20	TP:199, TN:452, FP:1, FN:1
	Accuracy	95.76%(973/1016)	99.69%(651/653)

TABLE IX
COMPARISON OF THE STANDARD HOG ALGORITHM AND THE GRADIENT VOTE APPROXIMATION

		INRIA	MIT
Standard Hog Algorithm	Detection Results	TP:543, TN:430, FP:23, FN:20	TP:199, TN:452, FP:1, FN:1
	Accuracy	95.76%(973/1016)	99.69%(651/653)
Approximation (magnitude)	Detection Results	TP:546, TN:434, FP:19, FN:17	TP:200, TN:452, FP:1, FN:0
	Accuracy	96.45%(980/1016)	99.84%(652/653)

TABLE X
COMPARISON OF THE STANDARD HOG ALGORITHM AND THE NORMALIZATION COMPUTATION APPROXIMATION

		INRIA	MIT
Standard Hog Algorithm	Detection Results	TP:543, TN:430, FP:23, FN:20	TP:199, TN:452, FP:1, FN:1
	Accuracy	95.76%(973/1016)	99.69%(651/653)
Approximation (magnitude)	Detection Results	TP:543, TN:430, FP:23, FN:20	TP:199, TN:452, FP:1, FN:1
	Accuracy	95.76%(973/1016)	99.69%(651/653)

detection of upright people in images and video. For simplicity, a linear SVM is used as a baseline classifier throughout the study. Table IV lists the comparison of the proposed circuit and the standard HOG algorithm. As compared with the standard HOG algorithm, the performances of the proposed circuit are almost the same. The accuracy rate can be calculated from four types of detection results. It is given as

$$\frac{TP+TN}{TP+TN+FP+FN} \tag{18}$$

where TP means true positive, TN means true negative, FP means false positive, and FN means false negative. A higher accuracy rate reveals better detection results. Table V lists the implementation results of our design. The VLSI architecture of our design was implemented by using Verilog hardware description language. We used SYNOPSYS Design Vision to synthesize the design with Taiwan Semiconductor Manufacturing Company's 0.13-μm cell library. The synthesis result shows that our design contains 153-K gate counts and operates at a clock rate of 167 MHz.

Furthermore, we have implemented and verified the architecture on an FPGA emulation board. Table VI lists the resource utilization of our design and the previous HOG hardware designs [8]–[14]. In Table VI, the proposed circuit can generate HOG features with less cost and higher frequency than in [8]–[14]. Hence, it is a good candidate for low cost, high performance, and high accuracy rate for object detection applications.

Tables VII–X list the comparisons of the standard HOG algorithm and the approximation method at each step mentioned in Section III. The proposed approximation method can acquire almost the same accuracy rate as the standard HOG algorithm. Table X shows that the approximation of fast inverse square root can achieve almost the same result as the standard HOG algorithm.

V. Conclusion

In this paper, we have proposed a low-cost real-time hardware implementation for HOG feature extraction. By using approximation methods to replace the complex operations, the proposed HOG design can be implemented with a lower hardware cost and achieve a faster working speed, as compared with the previous design [8]–[14]. It proves to be a good candidate for low-cost, high-performance, and high-accuracy-rate circuits for human detection. The proposed circuit can be integrated with other image processing components, such as noise removal and image enhancement, on a single chip for more applications in intelligent systems.

References

[1] S. J. Krotosky and M. M. Trivedi, "Person surveillance using visual and infrared imagery," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 8, pp. 1096–1105, Oct. 2008.

[2] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *Int. J. Comput. Vision*, vol. 38, no. 1, pp. 15–33, Jun. 2000.

[3] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio, "Pedestrian detection using wavelet templates," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, Jun. 1997, pp. 193–199.

[4] G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[5] H. Cheng, N. Zheng, and J. Qin, "Pedestrian detection using sparse Gabor filter and support vector machine," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2005, pp. 583–587.

[6] S. Belongie, J. Malik, and J. Puzicha, "Matching shapes," in *Proc. 8th ICCV*, Vancouver, BC, Canada, Jul. 2001, pp. 454–461.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, Jun. 2005, vol. 1, pp. 886–893.

[8] T. P. Cao and G. Deng, "Real-time vision-based stop sign detection system on FPGA," in *Proc. Digital Image Comput., Tech. Appl. Los Alamitos*, Canberra, ACT, Australia, 2008, pp. 465–471, IEEE Computer Society.

[9] S. Bauer, U. Brunsmann, and S. S. -Macht, "FPGA implementation of a HOG-based pedestrian recognition system," in *MPC-Workshop*, Jul. 2009, pp. 49–58.

[10] M. Hiromoto and R. Miyamoto, "Hardware architecture for high-accuracy real-time pedestrian detection with CoHOG features," in *Proc. IEEE ICCVW*, 2009, pp. 894–899.

[11] R. Kadota, H. Sugano, M. Hiromoto, H. Ochi, R. Miyamoto, and Y. Nakamura, "Hardware architecture for HOG feature extraction," in *Proc. IEEE Conf. Intell. Inf. Hiding Multimedia Signal Process.*, Nov. 2009, pp. 1330–1333.

[12] Y. Yazawa, "FPGA hardware with target-reconfigurable object detector by joint-HOG," in *Proc. SSII*, Yokohama, Japan, 2011, pp. 1–7.

[13] 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 *Proc. Int. Conf. FPT*, Dec. 12–14, 2011, pp. 1–8.

[14] K. Mizuno, Y. Terachi, and K. Takagi, "Architectural study of HOG feature extraction processor for real-time object detection," in *Proc. IEEE Workshop Signal Process. Syst.*, Oct. 2012, pp. 197–202.

[15] S. E. Lee, K. Min, and T. Suh, "Accelerating histograms of oriented gradients descriptor extraction for pedestrian recognition," *Comput. Elect. Eng.*, vol. 39, no. 4, pp. 1043–1048, May 2013.

[16] Y. Pang, Y. Yuan, X. Li, and J. Pan, "Efficient HOG human detection," *Signal Process.*, vol. 91, no. 4, pp. 773–781, Apr. 2011.

[17] D. D. Gajski, *Principles of Digital Design*. Upper Saddle River, NJ, USA: Prentice-Hall, 1997.

[18] H. T. Ngo and V. K. Asari, "A pipelined architecture for real-time correction of barrel distortion in wide-angle camera images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 3, pp. 436–444, Mar. 2005.

[19] J. D. Bruguera, N. Guil, T. Lang, J. Villalba, and E. L. Zapata, "CORDIC based parallel/pipelined architecture for the Hough transform," *J. VLSI Signal Process.*, vol. 12, no. 3, pp. 207–221, Jan. 2001.

[20] Newton's method. [Online]. Available: http://en.wikipedia.org/wiki/Newton's_method

[21] Fast inverse square root. [Online]. Available: http://en.wikipedia.org/wiki/Fast_inverse_square_root

[22] IEEE754 single precision binary floating-point format. [Online]. Available: http://en.wikipedia.org/wiki/Single_precision_floating-point_format#IEEE_754_single_precision_binary_floating-point_format:_binary32

[23] INRIA person dataset. [Online]. Available: http://pascal.inrialpes.fr/data/human/

[24] MIT pedestrian data. [Online]. Available: http://cbcl.mit.edu/software-datasets/PedestrianData.html

Pei-Yin Chen received the B.S. and Ph.D. degrees from National Cheng Kung University, Tainan, Taiwan, in 1986 and 1999, respectively, and the M.S. degree from Pennsylvania State University, University Park, PA, USA, in 1990, all in electrical engineering.

He is a Professor with the Department of Computer Science and Information Engineering, National Cheng Kung University. His research interests include very large scale integration chip design, video compression, fuzzy logic control, and gray prediction.

Chien-Chuan Huang received the B.S. and M.S. degrees in computer science and information engineering in 2005 and 2008, respectively, from National Cheng Kung University, Tainan, Taiwan, where he has been working toward the Ph.D. degree in computer science and information engineering since September 2008.

His research interests include image processing, very large scale integration chip design, and data compression.

Chih-Yuan Lien received the B.S. and M.S. degrees from National Taiwan University, Taipei, Taiwan, in 1996 and 1998, respectively, and the Ph.D. degree from National Cheng Kung University, Tainan, Taiwan, in 2009, all in computer science and information engineering.

He is an Assistant Professor with the Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan. His research interests include image processing, very large scale integration chip design, and video coding systems.

Yu-Hsien Tsai received the B.S. degree in computer science from National Chung Cheng University, Chiayi, Taiwan, in 2009 and the M.S. degree in computer science and information engineering from National Cheng Kung University, Tainan, Taiwan, in 2011.

Her research interests include very large scale integration chip design, data compression, and image processing.