2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing

# Hardware Architecture for HOG Feature Extraction

Ryoji Kadota, Hiroki Sugano, Masayuki Hiromoto, Hiroyuki Ochi
Department of Communications and Computer Engineering
Kyoto University
Yoshida-hon-machi, Sakyo, Kyoto, Japan
{ryo-z, hiroki, hiromoto}@easter.kuee.kyoto-u.ac.jp
ochi@kuee.kyoto-u.ac.jp

Ryusuke Miyamoto
Department of Information Systems
Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma, Nara Japan
miya@is.naist.jp

Yukihiro Nakamura
Research Organization of Science and Engineering
Ritsumeikan University
1-1-1 Nojihigashi, Kusatsu, Shiga, Japan
y-nakamura@fs.ritsumei.ac.jp

***Abstract*— Pedestrian recognition on embedded systems is a challenging problem since accurate recognition requires extensive computation. To achieve real-time pedestrian recognition on embedded systems, we propose hardware architecture suitable for HOG feature extraction, which is a popular method for high-accuracy pedestrian recognition. To reduce computational complexity toward efficient hardware architecture, this paper proposes several methods to simplify the computation of HOG feature extraction, such as conversion of the division, square root, arctangent to more simple operations. To show that such simplifications do not spoil the recognition accuracy, the detection performance is also evaluated using a support vector machine. Moreover, we implement the proposed architecture on an ALTERA Stratix II FPGA using Verilog HDL to evaluate the circuit size and the processing performance of the proposed architecture. Implementation results show that real-time processing for 30 fps VGA video can be achieved if 10 instances of the proposed hardware are used in parallel.**

## I. Introduction

Nowadays, pedestrian recognition is one of the most challenging problems in the field of computer vision. There have been many recognition algorithms proposed for purposes such as prevention of traffic accidents by using vehicle cameras and crime deterrence by using security cameras. For embedded systems, however, a recognition algorithm that achieves not only high accuracy but also real-time processing in an environment with limited resources is required.

Pedestrian recognition includes two significant processes, detection and tracking. Both processes use feature descriptors as probability indicator of pedestrians. Various feature descriptors for pedestrian recognition have been proposed, such as the method with Haar wavelets [1], with Haar-like features [2], and with Gabor filters [3]. However, there is no system that completely satisfies the demands for pedestrian recognition.

In this paper, we focused on the feature extraction method using Histograms of Oriented Gradients (HOG) [4] for real-time pedestrian recognition. HOG is an efficient feature extraction scheme, and in combination with various classification algorithms, it is possible to discriminate a pedestrian in difficult conditions like deformation, rotation or illumination change. However, the calculation of the HOG features is computationally complicated.

Therefore, in order to achieve highly accurate and real-time pedestrian recognition on an embedded system, this paper proposes a hardware architecture to accelerate calculation of HOG features.

The rest of this paper is organized as follows. Section 2 explains HOG feature extraction. In section 3, we discuss how to simplify the computation of HOG feature extraction and propose a novel hardware architecture suitable for HOG feature extraction. Section 4 evaluates the proposed architecture and Section 5 concludes this paper.

## II. Histograms of Oriented Gradients

This section explains how to extract a HOG feature, which consists of many histograms of orientated gradients in localized areas of an image. In this explanation, computation of HOG feature extraction is divided into the following three steps.

### A. Gradient Computation

Figure 1 shows an idea of cells and blocks used for HOG feature extraction. In the HOG feature extraction, first of all, 1st order differential coefficients, $f_x(x, y)$ and $f_y(x, y)$, are computed by the following equations.

$$\begin{cases} f_x(x, y) = f(x + 1, y) - f(x - 1, y), \\ f_y(x, y) = f(x, y + 1) - f(x, y - 1), \end{cases} \tag{1}$$

where $f(x, y)$ means luminance at $(x, y)$. Then magnitude $m$ and direction $\theta$ of the computed gradients are computed by

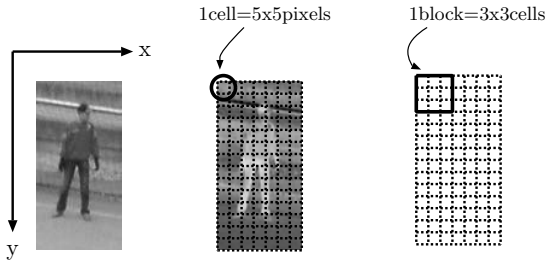
$$m(x, y) = \sqrt{f_x(x, y)^2 + f_y(x, y)^2}, \tag{2}$$

and

$$\theta(x, y) = \arctan \frac{f_x(x, y)}{f_y(x, y)}, \tag{3}$$

respectively.

978-0-7695-3762-7/09 $26.00 © 2009 IEEE
DOI 10.1109/IIH-MSP.2009.216




(a)input image (b)cell and gradient (c)block and normalization

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

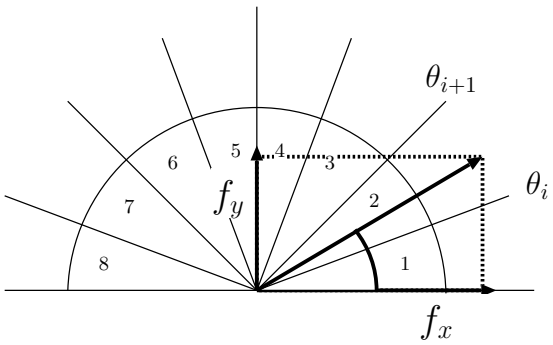



Fig. 2. Class determination at the histogram creation step.

### B. Histogram Generation

After computation of magnitude and direction, histograms are generated as follows:

1) determine which class $\theta(x, y)$ belongs to,
2) increment the value of the class determined by the previous step, and
3) repeat above operations for all gradients belong to the cell.

In order to reduce the effect of aliasing, both values of the neighboring two classes are incremented. while the increment process. Let $n$ indicate a class number where $\theta(x, y)$ belongs, and $nearest$ be a class that is nearest to the class $n$. The incremented values $m_n$ and $m_{nearest}$ are computed as follows:

$$m_n = (1 - \alpha)m(x, y), \tag{4}$$

and

$$m_{nearest} = \alpha m(x, y), \tag{5}$$

where $\alpha$ is a parameter for proportional distribution of magnitude $m(x, y)$. $\alpha$ is defined by using the distance from $\theta(x, y)$ to class $n$ and $nearest$,

$$\alpha = \frac{b\theta(x, y)}{\pi} - (n + 0.5), \tag{6}$$

where $b$ indicates a total number of classes.

### C. Histogram Normalization

Finally, a large histogram is created by combining all generated histograms belonging to a block that consists of some cells. After the large combined histogram is obtained, it is normalized as follows:

$$\boldsymbol{v} = \frac{\boldsymbol{V_k}}{\sqrt{||\boldsymbol{V_k}||^2 + 1}}, \tag{7}$$

where $\boldsymbol{V_k}$ is a vector corresponding to a combined histogram for the block, and $\boldsymbol{v}$ is a normalized vector, which is a final HOG feature.

## III. Hardware Architecture

In this section, we describe how to reduce computational complexity of HOG feature extraction, and evaluate the detection accuracy when we use the proposed simplification in combination with a support vector machine as a classifier. Based on this discussion, hardware architecture suitable for HOG feature extraction is proposed.

### A. How to reduce computational complexity

Computation of HOG feature extraction includes complicated operations unsuitable for hardware implementation. Since straightforward hardware implementation of these operations causes an increase in circuit size, simplification and modification of computation is indispensable to achieve efficient implementation suitable for embedded applications. This subsection describes how to reduce computational complexity of such operations in each step of the HOG extraction without degradation of accuracy.

*1) Computation of Gradient:* This is the first step of HOG extraction, and has square root and arctangent operations described as Equation (2) and Equation (3), respectively.

In our implementation, the square root operation that is necessary to compute magnitude of gradient is implemented by using a look-up-table. This kind of simplification is often adopted for hardware design of embedded systems. For the arctangent operation, more specific simplification is applied. The arctangent operation is required for computation of $\theta(x, y)$ that is used only to determine class of a histogram. Hence, we can simplify the arctangent operation to comparing operations that satisfy the following conditions:

$$f_x(x, y) \tan\theta_i \leq f_y(x, y) < f_x(x, y) \tan\theta_{i+1}. \tag{8}$$

By this equation, $\theta_i$ and $\theta_{i+1}$ corresponding to two classes incremented at histogram generation step are obtained as shown in Figure 2.

*2) Histogram generation:* In the histogram generation step, class value is incremented using magnitude $m(x, y)$ and $\theta(x, y)$. However, $\alpha$, a parameter necessary for proportional distribution of $m(x, y)$ to reduce aliasing, cannot be obtained because $\theta$ is not computed accurately by the previous simplified scheme. To obtain this parameter for proportional distribution, we try to let $\alpha$ be a constant. Figure 3 shows the result of pedestrian detection when $\alpha$ becomes a constant by using a support vector machine trained with INRIA data set [4]. This result shows that our simplification does not degrade detection accuracy. Based on the result, we have chosen to let $\alpha$ be a constant.

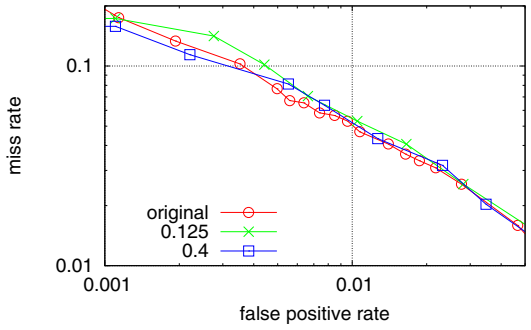



Fig. 3. Detection accuracy with constant $\alpha$.

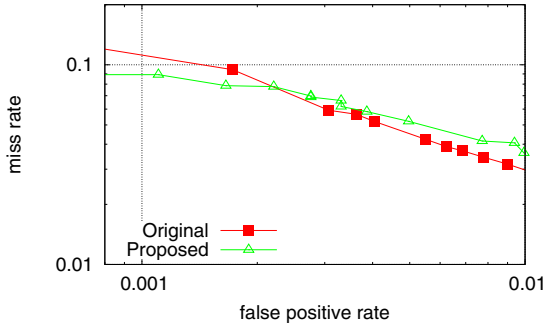



Fig. 4. Detection accuracy of fixed-point and original floating-point implementations.

*3) Histogram Normalization:* The histogram normalization step has division and square root operations. To simplify these operations, we approximate $||\boldsymbol{V_k}||^2 + 1$ to $2^n$, where $n$ is a natural number. By this approximation, Equation (7) is transformed as follows:

$$\sqrt{||\boldsymbol{V_k}||^2 + 1} = 2^{\frac{n+1}{2}}. \quad (9)$$

If $n$ is a odd number, $\sqrt{2}$ is treated as 1.375. The effect to accuracy of feature extraction is evaluated with the fixed point computation described in the next part.

*4) Fixed Point Computation:* After above simplification, we try to adopt fixed point representation instead of floating point representation for numeric computation to reduce computational complexity much more. Figure 4 shows the result of pedestrian detection with a support vector machine using INRIA data set. This result shows the proposed simplified scheme with fixed point computation achieves the comparable accuracy with of the original scheme.

### B. Hardware Architecture

Based on the above discussion, we propose hardware architecture suitable for HOG feature extraction. The proposed hardware architecture is described in Figure 5 and the functions of modules used are shown in Table I.

TABLE I
FUNCTIONS OF MODULES

| module | function |
|---|---|
| grad | calculation of gradient |
| mag | calculation of magnitude |
| arctan | calculation of direction |
| sqrt | look-up-table for square root |
| hist | histogram processing |
| make_hist | histogram generation |
| sq_hist | calculation of the sum of square in each cell |
| div | normalization |

TABLE II
PARAMETERS

| | |
|---|---|
| input image | $16 \times 32$ pixels |
| cell | $4 \times 4$ pixels |
| block | $3 \times 3$ cells |
| step stride | 4 pixels, vertically and horizontally |
| the number of classes | 8 |

Using this architecture, computation of HOG feature extraction is performed as follows:

1) images are inputted to the shift register line by line,
2) modules receive the input data for computing magnitude and direction of gradients,
3) histograms are generated using computed magnitude and direction of gradients.

For the histogram generation, to increase throughput efficiently, this architecture adopts pipelined processing as shown in Figure 6.

## IV. EVALUATION

To evaluate the proposed architecture, we have implemented it on an ALTERA Stratix II FPGA using Verilog HDL. Details are shown in Table II.

By this implementation, 672 cycles are required to compute HOG feature extraction for a window, which means a $16 \times 32$ input image. If the operating frequency of the proposed hardware is 127.49 MHz, it takes 5.2 $\mu$s to extract HOG feature of a window.

For practical use, detection is performed in raster-scan-order for an input image. In addition, to detect several sizes of pedestrian images, the detection should be performed for several numbers of scaled source images. We evaluate the computation time required for detection with the parameters defined by Table IV.

Under these conditions, the number of windows to be computed is 56466. Therefore, to achieve real-time processing

TABLE III
IMPLEMENTATION RESULTS

| | |
|---|---|
| Combinational ALUTs | 3794 /143520 |
| Dedicated logic registers | 6699 /143520 |
| DSP block 9-bit elements | 12/768 |
| Operating Frequency | 127.49MHz |

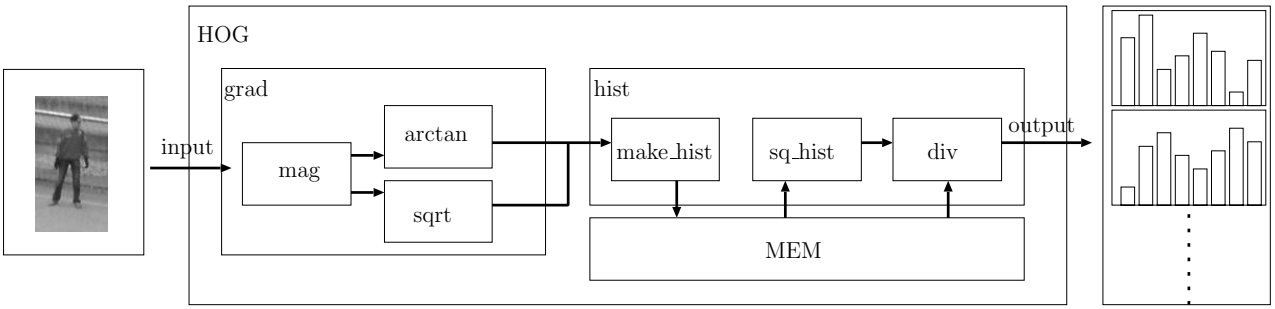



Fig. 5. Proposed hardware architecture.

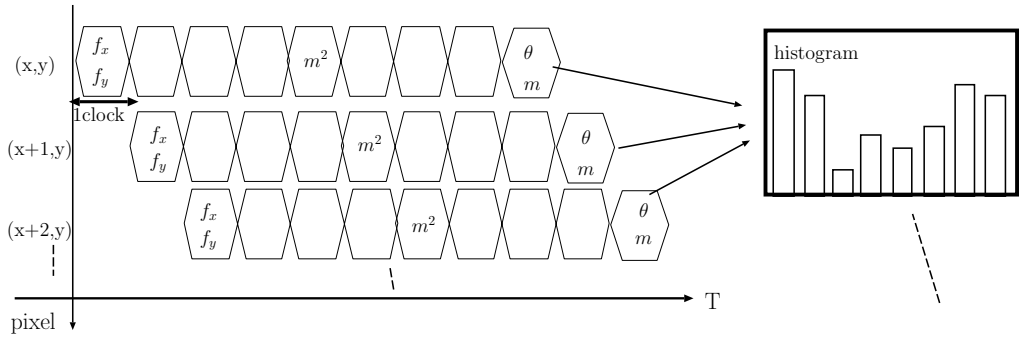



Fig. 6. Processing flow of histogram generation.

TABLE IV
PARAMETERS USED FOR EVALUATION

| | |
|---|---|
| image size | $640 \times 480$ pixels |
| detection window | $16 \times 32$ pixels |
| scan stride | 4 pixel |
| HOG feature $M$ | 864 dimensions |
| scale parameter | 1.2 |

of HOG feature extraction for 30fps video at this resolution, 10 instances of the proposed architecture must be used in parallel. Since the circuit size of the implementation is not so large and the feature extraction can be easily divided into portions, such parallel implementation can be realized on a single FPGA chip.

## V. Conclusion

In this paper, we proposed hardware architecture suitable for HOG feature extraction aiming real-time pedestrian recognition on embedded systems. To reduce computational complexity toward efficient hardware architecture, this paper describes how to simplify the computation of HOG feature extraction. The experimental results of pedestrian detection with a trained support vector machine show that application specific simplification, which includes the replacement of arctangent by comparison at the computation of gradient step and the fixation of the parameter $\alpha$ used for the histogram generation step, does not degrade the accuracy of detection. To evaluate circuit size and processing performance, we implemented the proposed architecture on an ALTERA Stratix II FPGA using Verilog HDL. As a result, the maximum frequency is 127.49 MHz and this achieves real-time processing for $640 \times 480$ size and 30 fps video when 10 instances of proposed hardware are used in parallel.

## Acknowledgements

This work was partly supported by a Global Center of Excellence (G-COE) program of the Ministry of Education, Culture, Sports, Science and Technology of Japan and by Grant-in-Aid for Young Scientists (B) 20700048, JSPS Fellows 20·2451, and JSPS Fellows 21·2488. This work was also supported by the VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Mentor Graphics, Inc.

## References

[1] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Proggio, "Pedestrian detection using wavelet templates," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1997, pp. 193–199.

[2] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision*, vol. 63, pp. 153–161, 2005.

[3] H. Cheng, N. Zheng, and J. Qin, "Pedestrian detection using sparse gabor filter and support vector machine," *Proc. of IEEE Intelligent Vehicles Symposium*, pp. 583–587, 2005.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.