A method for real-time implementation of HOG feature extraction

LUO Hai-bo^{1,3,4}, YU Xin-rong^{1,2,3,4,5}, LIU Hong-mei^{5}, DING Qing-hai^{6}
(1. Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, PR China;
2. Graduate School, Chinese Academy of Sciences, Beijing 100039, PR China
3. Key Laboratory of Optical-Electronics Information Processing, Chinese Academy of Science, Shenyang 110016, PR China;
4. Key Laboratory of Image Understanding and Computer Vision, Liaoning Province 110016, PR China;
5. AVIC HONGDU Aviation Industry Group LTD ; Nanchang 330024; PR China;
6. Research Institute of General Development and Demonstration of Equipment, Equipment of Air Force, Bejing 100076, China)

Abstract

Histogram of oriented gradient (HOG) is an efficient feature extraction scheme, and HOG descriptors are feature descriptors which is widely used in computer vision and image processing for the purpose of biometrics, target tracking, automatic target detection(ATD) and automatic target recognition(ATR) etc. However, computation of HOG feature extraction is unsuitable for hardware implementation since it includes complicated operations. In this paper, the optimal design method and theory frame for real-time HOG feature extraction based on FPGA were proposed. The main principle is as follows: firstly, the parallel gradient computing unit circuit based on parallel pipeline structure was designed. Secondly, the calculation of arctangent and square root operation was simplified. Finally, a histogram generator based on parallel pipeline structure was designed to calculate the histogram of each sub-region. Experimental results showed that the HOG extraction can be implemented in a pixel period by these computing units.

Key words: histogram of oriented gradient (HOG), feature extraction, real-time implementation, parallel pipeline

1 INTRODUCTION

HOG descriptors are feature descriptors used in computer vision and image processing to detect object. This technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts [1]. Dalal and Triggs [2] pointed out that HOG descriptors significantly outperform existing feature sets for human detection after studying the question of feature sets for robust visual object recognition. Wang and Guan [3] presented a novel method to implement graph cut for video object segmentation, they combined HOG feature to incorporate a shape prior into graph cut algorithm as a new way to enhance video object segmentation accuracy. In literature [4], HOG descriptors were adopted to perform online classification during object tracking process.

However, computation of HOG feature extraction is too complex for real-time implementation, and due to the increasing of circuit size caused by straightforward hardware implementation of HOG feature extraction, it is unsuitable for hardware implementation, so it is indispensable to simplify and modify the computation scheme in order to achieve efficient implementation suitable for real-time systems. In literature [5], several methods to simplify the computation of

HOG feature extraction was proposed, but so much specific simplification was applied and resulted in a bad influence on calculation accuracy. Furthermore, computation magnitude of gradient by using a look-up-table consumes too much memory resource. In this paper, we proposed a new method for real-time implementation of HOG feature extraction. The method can achieve higher precision in calculation with less logic and memory cost, and is more suitable for real-time HOG feature extraction.

2 HISTOGRAMS OF ORIENTED GRADIENTS[5]

The essential idea behind the Histogram of Oriented Gradient descriptors is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The implementation of these descriptors can be achieved by dividing the image into small connected regions, called cells, and for each cell compiling a histogram of gradient directions or edge orientations for the pixels within the cell. The combination of these histograms then represents the descriptor. For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block.

Based on the above explanation, HOG feature extraction 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.

2.1 Gradient computation

In HOG feature extraction, first of all, 1st order differential coefficients, $G_x(i,j)$ and $G_y(i,j)$, are computed by the following equations.

$$\begin{cases} G_x(i,j) = f(i+1,j) - f(i-1,j) \\ G_y(i,j) = f(i,j+1) - f(i,j-1) \end{cases} \tag{1}$$

where $f(i,j)$ means luminance at (i,j).

Then magnitude m and direction θ of the computed gradients are computed by the following expressions, respectively.

$$m(i,j) = \sqrt{G_x(i,j)^2 + G_y(i,j)^2} \tag{2}$$

$$\theta(i,j) = \arctan\left(\frac{G_x(i,j)}{G_y(i,j)}\right) \tag{3}$$

2.2 Histogram generation

After obtaining the values of m and θ, histograms are generated as follows:

1) Determine the class which $\theta(i,j)$ belongs to

2) Increase the value of the class determined by step 1)

3) Repeat above operations for all gradients belong to the cell.

In order to reduce the effect of aliasing, the values of two neighboring classes are increased. The increment make n indicates a class number which $\theta(i,j)$ belongs to, and n+1 would be the class which is the nearest one to class n. The increased values m_n and m_{n+1} are computed as follows:

$$\begin{cases} n = \left\lfloor \dfrac{b\theta(i,j)}{\pi} \right\rfloor \\ m_n = (1-\alpha)m(i,j) \\ m_{n+1} = \alpha m(i,j) \end{cases} \tag{4}$$

Where b indicates the total number of classes, α is a parameter for proportional distribution of magnitude $m(i,j)$ which is defined as the distance from $\theta(i, j)$ to class n and n+1,

$$\alpha = \frac{b}{\pi}\left(\theta(i,j) \bmod \frac{\pi}{b}\right) \tag{5}$$

2.3 Histogram normalization

Finally, a large histogram is created by combining all generated histograms belonging to a block consists of some cells. In order to reduce the influence of variations in illumination and contrast, L1-norm is adopted in this paper. After obtaining the large combined histogram, it can be normalized as follows:

$$v = \frac{V_k}{\|V_k\| + \varepsilon} \tag{6}$$

where V_k is the vector corresponding to a combined histogram for the block, ε is a small constant, and v is the normalized vector, which is a final HOG feature.

3 REAL-TIME IMPLEMENTATION OF HOG FEATURE EXTRACTION

In this section, how to reduce computational complexity of HOG feature extraction and how to evaluate the calculation accuracy with the proposed simplification will be described. Based on this discussion, a method for real-time implementation of HOG feature extraction is proposed.

3.1 Computation of gradient

Computation of gradient is the second step of HOG extraction, and has square root and arctangent operations described as equation (2) and equation (3), respectively. According to equation (2), straightforward hardware implementation of gradient magnitude computation costs too much logic resources or memory and is hardly realized in hardware-based real-time systems. Dealing with equation (2), it can be reformed into the following equations:

$$\begin{cases} m(i,j) = \sqrt{2}\,|G_x(i,j)| & ,|G_x(i,j)| = |G_y(i,j)| \\ m(i,j) = |G_x(i,j)|\sqrt{1 + G_y(i,j)^2 / G_x(i,j)^2} & ,|G_x(i,j)| > |G_y(i,j)| \\ m(i,j) = |G_y(i,j)|\sqrt{1 + G_x(i,j)^2 / G_y(i,j)^2} & ,|G_x(i,j)| < |G_y(i,j)| \end{cases} \tag{7}$$

Then the square root operation is transformed into $\sqrt{1+x^2}$ with $0 \le x \le 1$. It can be approximated with multi-linear function. Thus, in our implementation, the square root operation is implemented by using a 16 elements reduced look-up-table and linear computation according to equation (8). The curve in Figure 1 shows the error of the proposed algorithm in calculation of $\sqrt{1+x^2}$, and the largest error is 4.8742×10^{-4}.

$$\begin{cases} k = \lfloor 16x \rfloor \\ \sqrt{1+x^2} = \dfrac{16(LUT_{k+1} - LUT_k)}{x - k/16} \end{cases} \tag{8}$$

where LUT_k and LUT_{k+1} indicate the *kth* and *k+1th* elements of look-up-table, respectively.

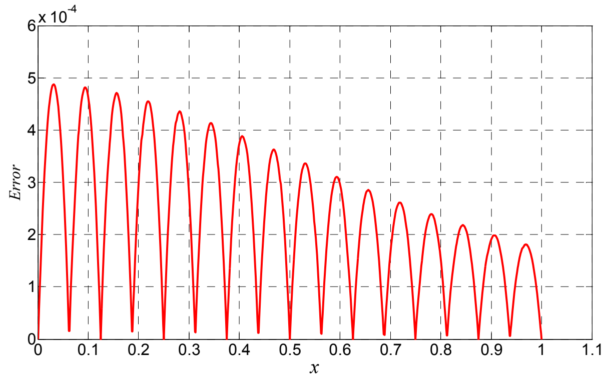

Figure 1. Error curve of the proposed algorithm in calculation of $\sqrt{1+x^2}$

For the arctangent operation, two solutions can be adopted. One is directly using a look-up-table of *arctan*(x), while the other is computing it with a Taylor expansion as equation (9). However, the first one costs plenty of memory and the second one converges only in the interval of [-1, 1]. Moreover, the convergent rate decreases rapidly with the increasing of $|x|$. For example, when x=1, 131 items in equation (9) must be calculated to limit the calculation error to 0.1°. These calculations will consume large amount of multiplying units. Therefore, directly calculating *arctan*(x) with equation (9) in its whole definition domain is not preferable.

$$\arctan(x) = \sum_{k=1}^{\infty} (-1)^{k-1} \frac{x^{2k-1}}{2k-1} \tag{9}$$

Since arctangent is an odd function, *arctan(x)* in [-∞,+∞] can be obtained by its values in [0, +∞], thus, equation (3) can be reformed into the following equations:

$$\begin{cases} \theta(i,j) = \arctan\left(G_x(i,j)/G_y(i,j)\right) & ,G_x(i,j) \le G_y(i,j) \\ \theta(i,j) = 0.5\pi - \arctan\left(G_y(i,j)/G_x(i,j)\right) & ,G_x(i,j) > G_y(i,j) \end{cases} \tag{10}$$

Hence, the computation of *arctan*(x) in [0, +∞] domain was transformed into [0, 1]. Moreover, when x>0.5, suppose $\theta = \arctan(x) = \arctan(0.5) + \alpha$ and $y = \tan(\alpha)$, *arctan*(x) can be deduced by the following equations:

$$\begin{cases} y = \dfrac{2x-1}{2+x} \\ \theta = \arctan(x) = \arctan(0.5) + \arctan(y) \end{cases} \tag{11}$$

Furthermore, when $0.25 < x \le 0.5$,

$$\begin{cases} y = \dfrac{4x-1}{4+x} \\ \theta = \arctan(x) = \arctan(0.25) + \arctan(y) \end{cases} . \tag{12}$$

In addition, *arctan*(x) in [0, 0.25] can be simplified as the linear function in (12), if the required precision is not so strict:

$$\arctan(x) = \frac{\arctan(0.25)}{0.25} x = 56.145x \approx 56.5x \tag{13}$$

Finally, in our implementation, arctangent operation can be obtained by jointly using equation (10), (11), (12), and (13). The curve in Figure 2 shows the error of the proposed algorithm in calculation of *arctan*(x), and the largest error is 0.0862°.

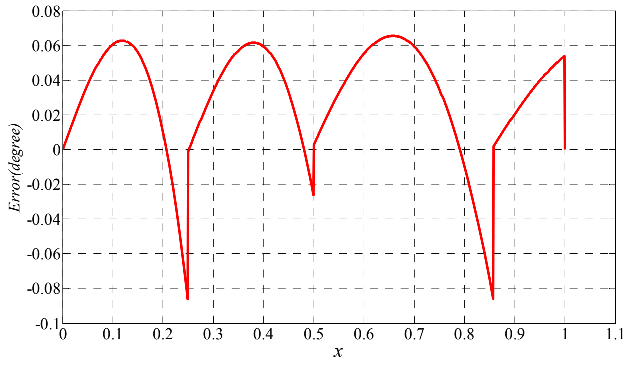

Figure 2. Error curve of the proposed algorithm in calculation of *arctan*(x)

3.2 Histogram generation

In this step, class value is increased using magnitude $m(i, j)$ and $\theta(i, j)$ according to equation (4) and (5). In order to increase throughput efficiently, the histogram generator adopts pipeline architecture as shown in Figure 3. In our implementation, the number of classes is 9, so the histogram storage is composed with 9 blocks embedded dual port RAM. In this paper, m_n is accessed through their right port (named Rp) and m_{n+1} is accessed through their left port (named Lp).

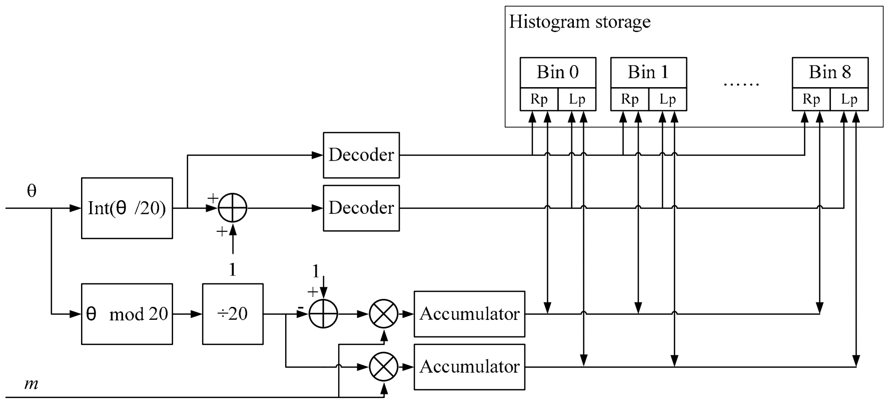

Figure 3. Histogram generator architecture

3.3 Hardware architecture for hog feature extractor

Based on the above discussion, we propose hardware architecture suitable for real-time HOG feature extraction which is described in Figure 4. With this architecture, computation of HOG feature extraction is performed as follows:

1) Images are input to the shift register line by line

2) Gradient calculator receives the input images for computing magnitude and direction of gradients

3) Histograms are generated using computed magnitude and direction of gradients, meanwhile, the sum of gradient magnitude in each cell is calculated(L1-norm)

4) Finally, the histograms are normalized and output

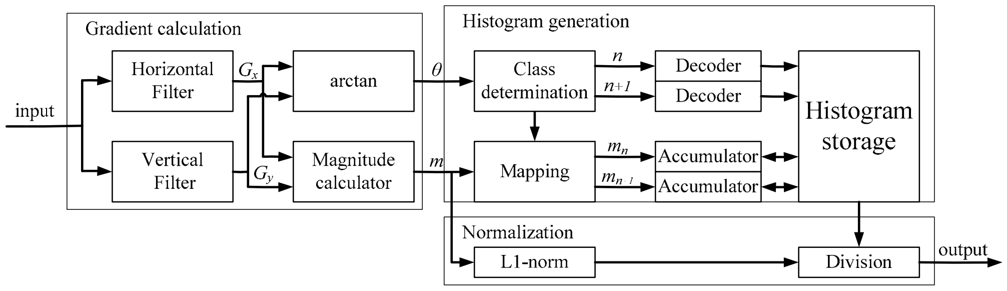

Figure 4. Hardware architecture for proposed HOG feature extractor

4 EVALUATION

In order to evaluate the proposed architecture, it is implemented on an ALTERA Cyclone III FPGA. Details are shown in Table 1, and the implementation results are shown in Table 2. By this implementation, only one cycle is required to compute HOG feature extraction for each pixel with 3 cycles delay.

Table 1: Parameters

Input image	256×256 pixels
Cell	8×8 pixels
Block	4×4 cells
Step stride	8 pixels, vertically and horizontally
The number of classes	9

Table 2: Implementation results

Family	Cyclone III
Device	EP3C5F256C6
Total logic elements	937/5,136(18%)
Total combinational functions	880/5,136(17%)
Dedicated logic registers	175/5,136(3%)
Total registers	175
Total memory bits	146,944/423,936(35%)
Embedded Multiplier 9-bit elements	3/46(7%)
Total PLLs	1/2(50%)

5 CONCLUSION

In this paper, a method for real-time implementation of HOG feature extraction was proposed. To reduce computational complexity and make efficient hardware architecture, this paper describes how to simplify the computation of HOG feature extraction including the computation of arctangent operation, gradient magnitude and histogram generation. To evaluate circuit size and processing performance, the proposed method was implemented on a single ALTERA Cyclone III FPGA chip. As a result, only one cycle is required to compute HOG feature extraction for each pixel with 3 cycles delay. The experimental results show that the proposed method is further suitable for real-time HOG feature extraction. Moreover, the proposed method for arctangent computation and gradient magnitude computation can be used in the system which contains gradient computation such as SIFT, phase congruency feature extraction and so on.

REFERENCES

[1] Histogram of oriented gradients. http://en.wikipedia.org/wiki/Histogram_of_oriented_gradients.

[2] Dalal N., Triggs B.. "Histograms of oriented gradients for human detection" Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), pp. 886-893.

[3] Wang Chun-hao,Ling Guan Ling. "Graph Cut Video Object Segmentation using Histogram of Oriented Gradients" Proceedings of the 2008 IEEE International Symposium on Circuits and Systems (ISCAS 2008), pp. 2590-2593.

[4] Xiao Jiangjian, Cheng Hui, and Feng Han etc. "Object Tracking and Classification in Aerial Videos" Proceedings of the 2009 SPIE Automatic Target Recognition XVIII, Vol. 6967, pp. 696711-1-696711-9.

[5] Kadota Ryoji, Sugano Hiroki, and Hiromoto Masayuki etc. "Hardware Architecture for HOG Feature Extraction" Proceedings of the 2009 IEEE Intelligent Information Hiding and Multimedia Signal Processing, pp. 1330-1333.

LUO Hai-bo E-mail: luohb@sia.cn