Research on HOG-SVM pedestrian detection method based on FPGA

Yanghang Lin

School of Microelectronics; South China University of Technology, Guangzhou, 510000, China

202030323368@mail.scut.edu.cn

Abstract. At present, pedestrian detection technology has become a research hotspot in the field of image processing and computer vision, and has developed to open up a wide range of application prospects. This paper gives a comprehensive introduction to the algorithm and FPGA implementation of object detection algorithm including HOG and SVM. In the field of pedestrian detection, HOG-SVM pedestrian detection method based on FPGA has unique advantages compared with other various object detection methods. In the current research, this implementation method usually has a detection accuracy rate of more than 95% and a detection speed of more than 30 frames/s for the INRIA data sets, which perfectly meets the pedestrian detection requirement. FPGA with its parallel structure and a large amount of programmable logic sources greatly improve the HOG-SVM to perform real-time pedestrian detection. With the development of hardware acceleration and detection algorithms, more and more pedestrian detection methods with better performance will appear in the future.

Keywords: HOG, SVM, FPGA, pedestrian detection, object detection.

1. Introduction

Real-time image-based pedestrian detection is a difficulty and research hotspot in the field of computer vision, which is used in a wide range of applications, such as security, surveillance , entertainment, robot navigation and intelligent vehicle detection [1-3]. In 2005, the classical detection algorithm, histogram of oriented gradients (HOG) was proposed by Dalal and Triggs [4]. It is utilized with support vector machine (SVM) for classification, and can accurately achieve a high detection throughput in difficult conditions [5]. The field programmable gate array (FPGA) with parallel structure is an ideal implementation to perform parallel acceleration on complex algorithm, which has been widely implemented for the HOG and SVM in recent research.

In this paper, research on HOG combined with SVM pedestrian detection method based on FPGA has been conducted. First of all, a thorough review of the principle of object detection algorithm according to its process from image preprocessing, HOG feature extraction to SVM classifier is proposed. Next, an overview of the object detection flow in hardware is summarized accordingly. In addition, the FPGA implementation is introduced in the aspect of image acquisition and display, and object detection algorithm specifically. The next part compares and analyzes different object detection methods in the field of pedestrian detection, thus explaining the advantages of the HOG-SVM method based on FPGA. After summarizing the current research effect in terms of detection accuracy and detection speed, this paper concludes with a summary and outlook. Aimed at few detailed reviews to the process and effect of the HOG-SVM method based on FPGA at present, this paper makes a comprehensive summary based on the results of previous pedestrian detection research, which will help researchers to have an overall grasp of the method.

2. The principle of detection

2.1. Image preprocessing

In order to optimize the detection quality of the image in the subsequent feature extraction and classification process, it is suggested to perform preprocessing on the input image.

(1) Grizzled Processing

Since the color information of the image has little effect and different color spaces are prone to interference, firstly RGB values of pixels are converted into gray values to reduce interference and useless information. By compressing the image data information, the calculation speed has been improved.

(2) Normalization

Next, due to the differences in the weather and environment where the snapshots are taken, it is necessary to normalize the images to reduce the impact of different exposure and shadows. Gamma correction is generally used to adjust the contrast of the image for normalization:

$$I(x,y) = I(x,y)^{gamma} \tag{1}$$

where $\mathrm{I(x,y)}$ represent the value of pixel at $(\mathrm{x,y})$. Therefore, the redundant information of the image is reduced and the features of the target are relatively enhanced.

2.2. HOG feature

HOG feature extraction algorithm is a manually crafted feature descriptor for object detection in image processing. It recognizes the shape and contour information of the target by calculating the oriented gradient histogram of the local area of the image to form a feature set.

2.2.1. Principle. As proposed by proposed by Dalal and Triggs, HOG feature extraction algorithm firstly uses a 128×64 pixels of window to perform sliding detection by eight pixels (the width of a cell) per time on the image. The detection window contains 7×15 sliding blocks, which consists of 2×2 cells respectively. Each cell is composed of 8×8 pixels and generates 9 feature descriptors through the histogram of 9 orientation bins. Therefore, HOG is able to extract 3780 (9×4×7×15) features in one detection window.

2.2.2. HOG feature extraction algorithm. In general, the algorithm consists of three steps as follows:

(1) Gradient Calculation

As shown in equation (2) and (3), the Sobel operator $M_x = [-1, 0, 1]$ and $M_y = [1, 0, -1]^T$, which are used as 2D sharpening filters, are applied in convolution operations with pixel values $\mathrm{P(x,y)}$ to calculate the gradient in horizontal and vertical directions [6].

$$G_x(x,y) = M_x * \mathrm{P(x,y)} = \mathrm{P(x+1,y)} - \mathrm{P(x-1,y)}, \tag{2}$$

$$G_y(x,y) = M_y * \mathrm{P(x,y)} = \mathrm{P(x,y+1)} - \mathrm{P(x,y-1)}. \tag{3}$$

Then the gradient vector can be described in magnitude and orientation according to equation (4) and (5), respectively:

$$\mathrm{M(x,y)} = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}, \tag{4}$$

$$\theta\mathrm{(x,y)} = \arctan\frac{G_y(x,y)}{G_x(x,y)}. \tag{5}$$

Gradient information can be used to describe the contour of the image.

(2) Histogram Generation

In each cell, all pixels perform weighted votes according to the magnitude gradient $M(x, y)$ and the direction gradient $\theta(x, y)$ to generate the histogram, which is divided into 9 orientation bins from 0 to π. As shown in equation (6), (7) and (8), the neighboring bins are increased by M_n and $M_{nearest}$ respectively [7].

$$M_n = (1 - \alpha) \times M(x, y), \tag{6}$$

$$M_{nearest} = \alpha \times M(x, y), \tag{7}$$

$$\alpha = (n + 0.5) - \frac{b \times \theta(x,y)}{\pi}, \tag{8}$$

where n is the order, and b is the amount of orientation bins, which is 9. The 9 descriptor vectors of each cell then are obtained from the histogram.

(3) Block Normalization

The block which consists of 2×2 cells, has a 36-bin (4 cells×9 bins) histogram. Since the gradient is sensitive to light information, the diversification of illumination and background environment will cause a large variation in the gradient. Therefore, the descriptor vector v of the block is normalized again using L2-norm algorithm for better performance [8].

$$v \rightarrow \sqrt{\frac{v}{(\|v\|_2 + \varepsilon)}}, \tag{9}$$

where $\|v\|_2$ is the 2-norm, and ε is a small constant to prevent 0 denominator. Finally, the 3780-dimensional descriptor vector of the detection window is obtained by concatenating all the normalized blocks.

2.3. *SVM classifier*

SVM is a binary classification model of supervised learning. As shown in Figure 1, it maps the feature descriptor vectors of instance to points in the space and searches the classification hyperplane with optimal robustness and generalization ability as the decision function H3.

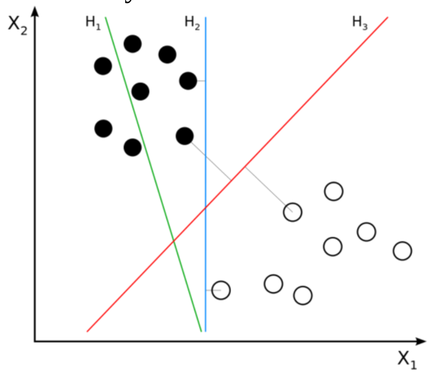

Figure 1. Linear separability optimal hyperplane.

2.3.1. Principle. The core idea of SVM is that support vector samples, which is some sample points with the shortest margin to the classification hyperplane, will play a key role in identifying problems. As shown in equation (10), the commonly used linear SVM decision function can be solved by maximizing the margin between support vectors $\frac{2}{\|\omega\|}$.

$$y(x) = \omega^T x + b, \tag{10}$$

where $\omega = \{\omega_1; \omega_2; \ldots; \omega_d\}$ is the weight vector where d represents the dimension of the feature descriptor vector, x is the descriptor vector and b is the bias to train.

2.3.2. Calculation of decision function. The process of SVM classification can be divided into training and detection. First the linear SVM decision function is trained through the training set to generate a binary classifier. Then the HOG feature descriptor vector is utilized in trained SVM to distinguish whether there are wanted targets in the detection window.

During the training process, previously mentioned x in equation (10) is the training set. In order to train the maximum-margin decision function, it is necessary to solve a convex quadratic programming problem $min\frac{1}{2}\|\omega\|^2$ with the constraints of equation (11).

$$s.t.\, yi(\omega^{\mathrm{T}}\mathrm{x}+\mathrm{b}) \geq 1, i = 1,2,3,\dots,n. \tag{11}$$

Using the Lagrangian optimization method, the problem of solving the maximum margin can be converted into a relatively simple dual optimization problem. First, define the Lagrangian function of the convex quadratic programming:

$$\mathrm{L}(\omega,\mathrm{b},\alpha) = \frac{1}{2}\|\omega\|^2 - \sum_{i=1}^{n}\alpha_i\left[y_i\left(\omega^{\mathrm{T}}\mathrm{x}+\mathrm{b}\right)-1\right], \tag{12}$$

where $\alpha = (\alpha_1,\alpha_2,\dots,\alpha_n)$ is Lagrangian multiplier and $\alpha_i \geq 0$. Let partial derivative of $\mathrm{L}(\omega,\mathrm{b},\alpha)$ with respect to ω and b be 0 to get the results:

$$\omega = \sum_{i=1}^{n}\alpha_i\, y_i x_i, \tag{13}$$

$$\sum_{i=1}^{n}\alpha_i\, y_i = 0. \tag{14}$$

Substitute the equation (13) and (14) into Lagrangian equation for maximum α_i to get:

$$\max_{\alpha}\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\,\alpha_j y_i y_j\, {x_i}^{\mathrm{T}} x_j. \tag{15}$$

Now the margin problem is converted into the dual problem with the constraint of equation (14). It is an extreme value problem of a quadratic function under inequality constraints, which has a unique solution:

$$\omega^* = \sum_{i=1}^{n}{\alpha_i}^* y_i x_i, \tag{16}$$

$$\mathrm{b}^* = y_i - \sum_{i=1}^{n}{\alpha_i}^* y_i {x_i}^{\mathrm{T}} x_i. \tag{17}$$

Finally, the classification hyperplane and decision function are represented as equation (18):

$$y_i = \mathrm{sgn}(\omega^{\mathrm{T}} x_i + \mathrm{b}). \tag{18}$$

3. FPGA hardware implementation

FPGA has rich hardware logic resources and is very suitable for image processing which has lots of pixels. In object detection, it is used for image acquisition and display on the one hand, and used to implement object detection algorithms on the other hand. First, the camera converts the collected image information into digital image signals and stores them in SDRAM. Then, these image signals will be subjected to the above-mentioned object detection process. Finally, the detected rectangular frame information of the target coordinates will also be stored into SDRAM, mixed with the previous input image and output on the display through the VGA interface.

3.1. Detection flow

The process of the detection algorithm is as follows: firstly, the collected image is preprocessed. Next, a sliding detection window traverses the entire image to calculate the HOG feature sets in each window sequentially. Then the feature sets are substituted into the trained SVM decision function to determine whether there are wanted targets included in the window, whose coordinate information is stored. Finally, all the coordinates are fused and displayed on the screen. The flow of the detection algorithm is shown in Figure 2.

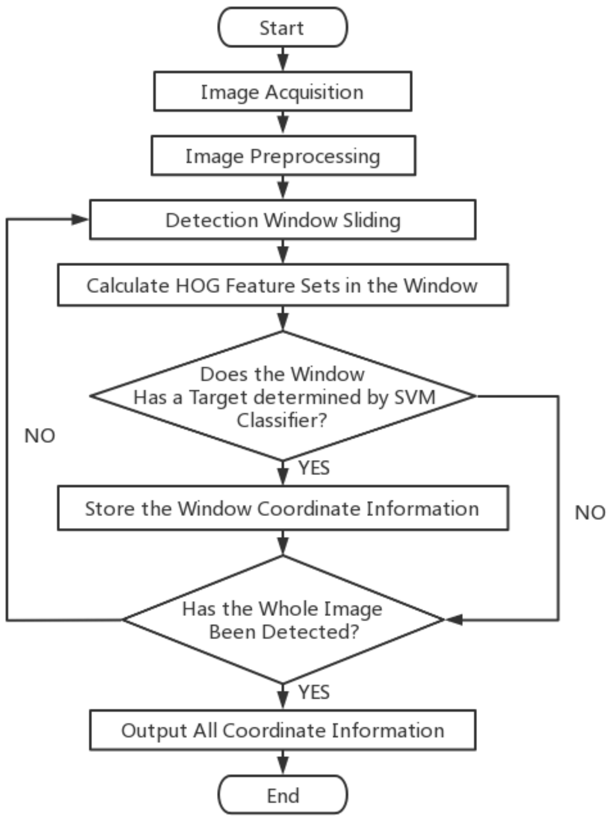
Start
Image Acquisition
Image Preprocessing
Detection Window Sliding
Calculate HOG Feature Sets in the Window
Does the Window Has a Target determined by SVM Classifier?
NO
YES
Store the Window Coordinate Information
NO
Has the Whole Image Been Detected?
YES
Output All Coordinate Information
End

Figure 2. Flow of the detection algorithm.

3.2. Detection algorithm on FPGA

3.2.1. Image preprocessing. The preprocessing process includes converting the image to grayscale and normalization. In hardware implementation, normalization has almost no impact on the detection effect. However, it consumes some hardware resources, which reduces the real-time detection performance. Therefore, normalization is not applied in the implementation process.

As shown in equation (19), (20) and (21), RGB888 digital image format is firstly converted to YCbCr444 in grayscale conversion [9].

$$\mathrm{Y} = 0.299\mathrm{R} + 0.587\mathrm{G} + 0.114\mathrm{B}, \tag{19}$$

$$\mathrm{Cb} = -0.172\mathrm{R} - 0.339\mathrm{G} + 0.511\mathrm{B} + 128, \tag{20}$$

$$\mathrm{Cr} = 0.511\mathrm{R} - 0.428\mathrm{G} + 0.083\mathrm{B} + 128. \tag{21}$$

Since floating-point calculations cannot be performed on FPGA devices, the entire right-hand side of the above equations can be multiplied by 256 and then right-shifted by 8 bits to remain the same value approximately.

The process is designed in a pipeline according to the different types of arithmetic operations involved and requires a total of three clock cycles. The first stage of the pipeline performs multiplication, the second stage performs addition, and the last stage computes the sum of the equations.

3.2.2. HOG feature. The computational cost of complex arithmetic for FPGA like square and square roots is quite high. Therefore, approximation, pipeline structure and parallel structure are often utilized to optimize algorithm without much impact on accuracy.

(1) Gradient Calculation

As mentioned before, gradient vector can be described in magnitude and orientation according to equation (4) and (5), respectively. The magnitude gradient $\mathrm{M}(\mathrm{x},\mathrm{y})$ in equation (4) can be approximated by using Square Root Approximation (SRA) method.

$$\mathrm{M}(\mathrm{x},\mathrm{y}) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \approx \max((0.875\mathrm{a} + 0.5\mathrm{b}), \mathrm{a}), \tag{22}$$

where $\mathrm{a} = \max(G_x(x,y), G_y(x,y))$, and $\mathrm{b} = \min(G_x(x,y), G_y(x,y))$. Figure 3 shows the structure of SRA, where P is the pipeline register.

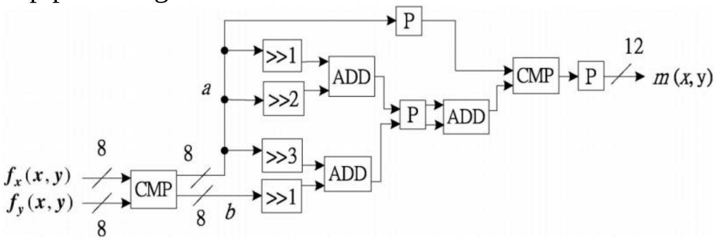

Figure 3. Diagram of SRA

The direction gradient $\theta(\mathrm{x},\mathrm{y})$ in equation (5) can be approximated by using the monotonicity of $\tan\theta$ and the relationship of $\tan\theta = \frac{G_y(x,y)}{G_x(x,y)}$. Hence, the orientation bin of $\theta(\mathrm{x},\mathrm{y})$ is determined according to the range of boundary $\tan\theta$ value of the bin.

(2) Histogram Generation

Figure 4 shows the structure of histogram generation, where m_0 to m_8 are previously mentioned as M_n and $M_{nearest}$ in equation (6) and (7), representing the values of each orientation bins. Since the previously simplified $\theta(\mathrm{x},\mathrm{y})$ is not accurate enough, α in equation (8) is set as a constant 0.5, which is proved to be no harm to accuracy in the implementation [7].

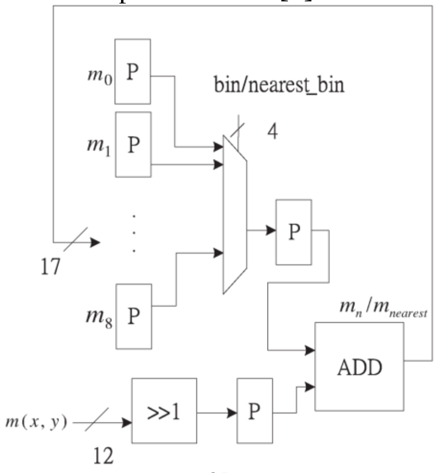

Figure 4. Diagram of histogram generation.

(3) Block Normalization

The algorithm of block normalization shown in equation (9), needs to perform the inverse square root operation $\mathrm{y} = \frac{1}{\sqrt{\mathrm{x}}}$, which is very complex for FPGA implementation. It can be simplified by using Newton-Raphson method and initial value of the magic number 0x5f3759df to get:

$$y_{approximate} = y_n - \frac{f(y_n)}{f'(y_n)} = \frac{y_n \times (3 - x \times y_n{}^2)}{2}. \tag{23}$$

Figure 5 shows the structure of the inverse square root operation, where DIC and IDC are utilized for the conversion between different number system.

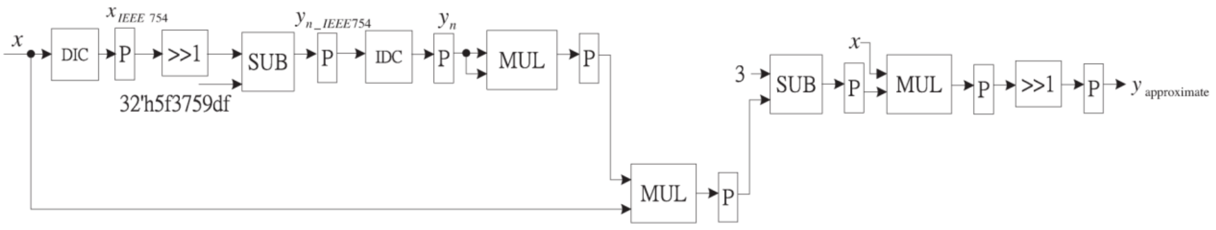

Figure 5. Diagram of the inverse square root operation.

Therefore, the descriptor vector v of the block is normalized as shown in equation (24):

$$\mathrm{v} \rightarrow \mathrm{v} \times y_{approximate}. \tag{24}$$

3.2.3. SVM classifier. The linear SVM decision function is firstly trained and next utilized to perform classification on the detection window. In hardware implementation, the weight vector ω and bias b are trained from some open-source SVM library and stored in the ROM of FPGA.

Since a detection window containing 3780 features will consume lots of memory before SVM classification, it is helpful to reduce memory utilization by using a cell-based scan structure. Then equation (10) is modified as shown in equation (25):

$$\mathrm{y(x)} = \sum_{i=0}^{105} \omega_i{}^{\mathrm{T}} \mathrm{x}_{Bi} + \mathrm{b}, \tag{25}$$

where x_{Bi} is the descriptor vector of a block in each window.

Figure 6 shows the structure of the SVM classifier, where each block RAM is consists of 15 rows with respect to the 15 blocks of one window [10]. Each row of RAM is processed by 2 multiply-accumulators (MAC) and a total of 30 MACs are used to calculate 30 descriptors vector each period. After a window has been detected, the block RAM is emptied and another detection period will start.

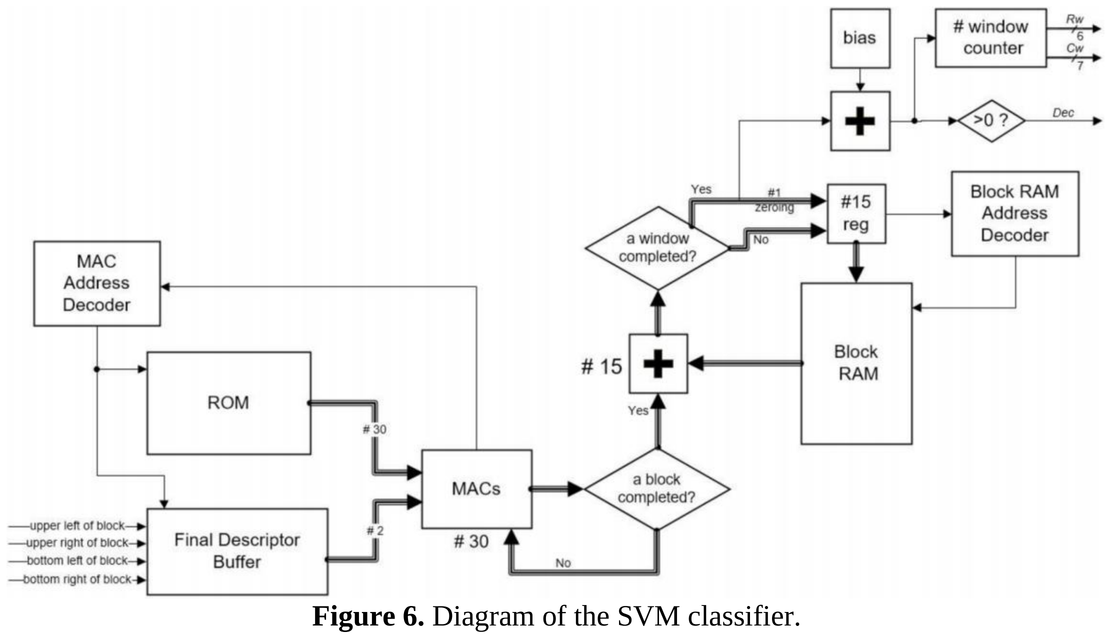

Figure 6. Diagram of the SVM classifier.

4. Pedestrian detection

In the field of computer vision, pedestrian detection with many different characteristics such as appearance, contours, heights, movements and clothing, is a special category in the aspect of object detection. The current mainstream algorithm is by adopting the method of machine learning, to extract features from a large number of training samples and establish a specific mathematical model, so as to transform the problem of pedestrian detection into the problem of pattern classification.

4.1. Detection method

In pedestrian detection, feature extraction algorithms and classifier algorithms are widely studied by scholars. Commonly used feature descriptors are Haar-like, Local Binary Pattern (LBP) and HOG:

(1) Haar-like feature descriptor

Haar-like calculates the pixel difference between white and black rectangular areas in different rectangular templates to describe local features such as edges and straight lines with local gray-scale mutations [11]. It has the advantages of simple calculation and convenient implementation. Therefore, it is effective for feature detection with relatively single contours such as faces and vehicles, but not suitable for pedestrian detection with complex and changeable detection scenes.

(2) LBP feature descriptor

LBP compares the pixel value of the peripheral point with the center point in a 3×3 window, the larger ones are set to 1 and the smaller are set to 0 [12]. It has the advantages of simple calculation and insensitivity to illumination and rotation. Therefore, it has a good detection effect for applications with simple background and intuitive texture, but the detection effect is poor when pedestrians are in an environment with complex texture features.

(3) HOG feature descriptor

HOG is able to accurately detect pedestrians under the influence of unfavorable factors such as deformation, rotation, or illumination changes. However, the calculation of HOG features is complex, the feature dimension is high, and the amount of data is large. Therefore, the real-time performance is relatively poor.

Based on the extracted features, objects can be classified. Common classifiers are AdaBoost (Adaptive Boosting), Artificial Neural Network (ANN) and SVM:

(1) AdaBoost classifier

AdaBoost trains multiple weak classifiers and finally combines them to obtain a strong classifier [13]. It has high precision, but it is sensitive to abnormal samples in the data set, leading to lower classification accuracy with complex and changeable sample characteristics.

(2) ANN classifier

ANN forms different structures through different kinds of connection layers, activation functions and weights. It has a high detection throughput, but it takes a long time to train due to the complexity of the model and the large number of samples required.

(3) SVM classifier

SVM is able to accurately process high-dimensional feature sets without spending lots of efforts on sample training. However, since it need to continuously perform complex calculations, the computational cost for hardware is high.

There are three types of mainstream embedded platforms: FPGA, DSP, and ARM, each with its own advantages. In view of abundant logic resources and powerful parallel computing capability of FPGA, it is helpful to improve the real-time performance of HOG. Besides, considering the advantages of low power consumption and cost, and low latency and high throughput, FPGA is very suitable for the computationally intensive SVM. Therefore, current research on pedestrian detection is usually realized by FPGA based on HOG-SVM method.

4.2. Detection performance

In previous studies, the pedestrian detection method used in this paper usually has two analysis indicators: detection accuracy and detection speed. The detection accuracy can be directly calculated as shown in equation (26), or represented indirectly by FPPW (False Positives Per Window)-MissRate, where the lower the false detection rate, the higher the accuracy rate [14].

$$\mathrm{R} = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}}, \tag{26}$$

where T, F, P, N means true, false, positive, and negative, respectively. The detection speed is usually represented in the form of frame(s)/s. According to the standard algorithm of 640×480 resolution image and 16×16 block used by Dalal and Triggs, summarize the research at present for the French Institute for Research in Computer Science and Control (INRIA) data sets as follows:

The detection accuracy expressed by equation (26) usually reaches 95%, which basically meets the pedestrian detection standard. The detection throughput is usually greater than 30 frames/s, which can meet the speed requirements of pedestrian detection.

5. Conclusion

In the scientific discipline field of computer vision, image-based pedestrian detection technology is a hot topic at present and has been developed for more than two decades. During the research history, the HOG algorithm proposed in 2005 is the most classic pedestrian detection algorithm, but has a high calculation cost due to the repetition and complexity. Therefore, many improved methods have been proposed by later generations, which has achieved better detection effect from describing human characteristics, better robustness in complex environment and faster detection throughput due to the development of hardware accelerators. This paper summarizes the effects that the FPGA-based HOG-SVM pedestrian detection method has achieved at this stage. In particular, the algorithm and FPGA implementation of HOG and SVM are comprehensively introduced. Through comparison, the advantages and disadvantages of this method in pedestrian detection are analyzed in detail. In conclusion, FPGA with its parallel structure and large logic resources is very suitable for the pedestrian detection method based on HOG-SVM. Moreover, the detection accuracy of this implementation method usually reaches 95%, and the speed reaches more than 30 frames/s for the INRIA data sets, which can well complete the work of pedestrian detection.

The advantage of FPGA implementation is that area can be traded for processing speed. Although the resource usage of the HOG-SVM method is less than that of CNN, it is still necessary to consider simplification and optimization of the algorithm in conjunction with a specific FPGA chip to avoid insufficient resources. Current methods are conducted by using static HOG descriptors and fixed-size sliding windows. To improve detection performance, dynamic detection can be achieved by combining dynamic HOG descriptors, and multi-scale sliding windows can be used to detect targets of different sizes. In addition, the new generation of FPGA chip supports partial dynamic reconfigurable technology, which can reconfigure the FPGA hardware at runtime according to different requirements, thereby improving the flexibility of the system. In the future, CNN with highly scalable in the learning capability might be utilized with the development of hardware implementation.

References

[1] S. Bauer, S. Kohler, K. Doll, and U. Brunsmann, “FPGA-GPU architecture for kernel SVM pedestrian detection,” *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, Jun. 2010.

[2] C.-S. Fahn, C.-P. Lee, and Y.-S. Yeh, “A real-time pedestrian legs detection and tracking system used for autonomous mobile robots,” *2017 International Conference on Applied System Innovation (ICASI)*, May 2017.

[3] A. Broggi, R. I. Fedriga, A. Tagliati, T. Graf, and M. Meinecke, “Pedestrian Detection on a Moving Vehicle: an Investigation about Near Infra-Red Images,” *2006 IEEE Intelligent Vehicles Symposium*.

[4] N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, 2005.

[5] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[6] H. Madadum and Y. Becerikli, "The implementation of Support Vector Machine (SVM) using FPGA for human detection," *2017 10th International Conference on Electrical and Electronics Engineering (ELECO)*, Bursa, Turkey, 2017, pp. 1286-1290.

[7] P. -Y. Chen, C. -C. Huang, C. -Y. Lien and Y. -H. Tsai, "An Efficient Hardware Implementation of HOG Feature Extraction for Human Detection," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 656-662, April 2014.

[8] P. Dai, J. Tang, J. Yuan, and Y. Yu, “A Hardware-Efficient HOG-SVM Algorithm and its FPGA Implementation,” *2021 2nd International Symposium on Computer Engineering and Intelligent Communications (ISCEIC)*, Aug. 2021.

[9] J. Wang, “Research on Pedestrian Detection System Based on FPGA,” Mar. 2021.

[10] J. Luo and C. Lin, “Pure FPGA Implementation of an HOG Based Real-Time Pedestrian Detection System,” *Sensors*, vol. 18, no. 4, p. 1174, Apr. 2018.

[11] X. Wen, L. Shao, Y. Xue, and W. Fang, “A rapid learning algorithm for vehicle classification,” *Information Sciences*, vol. 295, pp. 395–406, Feb. 2015.

[12] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[13] G. Rätsch, T. Onoda, and K.-R. Müller, “Soft Margins for AdaBoost,” *Machine Learning*, vol. 42, no. 3, pp. 287–320, 2001.

[14] M.-S. Wang and Z.-R. Zhang, “FPGA implementation of HOG based multi-scale pedestrian detection,” *2018 IEEE International Conference on Applied System Invention (ICASI)*, Apr. 2018.