Journal of Computers Vol. 28, No. 2, 2017, pp. 87-103
doi:10.3966/199115592017042802007

FPGA Implementation of a Real-Time Pedestrian Detection Processor Aided by E-HOG IP

Ai-Ying Guo[1,2], Mei-Hua Xu[1], Feng Ran[3], and Ang Li[1]

[1] School of Mechatronics Engineering and Automation, Shanghai University,
Shanghai, China
{gayshh, mhxu, shulivia}@shu.edu.cn

[2] Department of Electrical and Mechanical Engineering, Shan Xi Light Industry and Technical College,
Taiyuan, China

[3] Microelectronics Research and Development Center, Shanghai University,
Shanghai, China
ranfeng@shu.edu.cn

Received 21 December 2015; Revised 21 March 2016; Accepted 11 April 2016

Abstract. This paper describes a real-time pedestrian detection processor of Field Programmable Gate Arrays (FPGA) using a novel structure aided by E-HOG IP. This structure proposes a three stages detection to reduce the amount of calculation and improve the processing speed. Compared to the traditional methods, in the first stage, the Sobel-step can select windows of interest before feature extraction and classification. In the second stage, the uniform-LBP (Local Binary Pattern), which is implemented cell by cell, has approximately half-dimension of HOG and simultaneous SVM (Support Vector Machine) to decrease total computation. In the third stage, improved HOG (Histogram of Oriented Gradient), called E-HOG (Efficient-HOG), is suitable to be realized in hardware. E-HOG and SVM are adopted to improve the detection rate. To evaluate the performance, the proposed structure is implemented into an FPGA while the E-HOG is manufactured as one common IP. The results indicate that this processor can detect pedestrians with 48 fps for VGA resolution under 25Mhz in low FPGA series and satisfy the demand of real-time.

Keywords: E-HOG (Efficient-Histogram of Oriented Gradient), Field Programmable Gate Arrays (FPGA), pedestrian detection, Sobel-Step, uniform-Local Binary Pattern (LBP)

1 Introduction

Pedestrian detection is one of the fundamental problems in object detection that has been widely applied in autonomous robotic navigation and automotive safety. Due to many challenges, such as variable shape and body, illumination, occlusions, and different views, the practical application of pedestrian detection is unlikely to be completely solved.

In the past ten years, many available approaches and structures for pedestrian detection have been proposed. The present detection methods can be roughly divided into two types: monocular-based and binocular-based. Based on the platform, pedestrian detection can be divided into: software-based and hardware-based.

This paper discusses only the monocular detection system. Viola, Jones and Snow first applied Haar wavelets, selected by Ada-Boost, to describe the pedestrian feature [1]. Then, Dalal and Triggs proposed the Histogram of Oriented Gradient (HOG) with a Linear Support Vector Machine (SVM) to detect pedestrians [2]. HOG based on the gradient is the most widely used feature to vividly describe contour lines. HOG performs far better than other approaches and forms the guiding type. Dollar, Wojek, Schiele and Perona computed Haar-like features from the LUV color channel, grayscale channel and gradient magnitude channel and proposed the integral image for fast computation [3]. Wojek and Schiele collected the shapelet features, HOG features and Haar-like features to combine Multi-Ftr and achieve good performance [4]. Felzenszwalb, Girshick and McAllester introduced the mixture of multiscale Deformable Part Models (DPM) along with HOG, even obtaining good results in the PASCAL object detection challenges [5]. In recent years, Convolutional Neural Networks (CNN) with their different layers detect objects more effectively than the other handcraft-based approaches [6-9].The above approaches achieve the state-of-the-art datasets in, for instance, the INRIA dataset and the Caltech dataset [10-12].

The drawback is that these approaches are effective but have heavy computational cost; therefore, they are unsuitable for processors with limited hardware resources, such as FPGA. In order to transfer pedestrian detection to an embedded device, some researchers proposed object detection using FPGA or FPGA-GPU for real-time application. Mizuno et al. designed an FPGA implementation of a HOG-based processor to fast detect objects; however, this system detected pedestrians based on a CPU with FPGA and cannot be transferred to one processor [13].

Most conventional pedestrian detection systems adopt a sliding window-based structure for scanning the whole frames. Computations and memory bandwidth are required for above 20 VGA (640x480 pixel) fps, adopting efficient and simple calculation or reusing intermediate data to satisfy the required computations and memory bandwidth. Conversely, data reuse requires an increase of hardware overload. Consequently, a design over architecture, algorithm and hardware overload is necessary.

In order to achieve real-time pedestrian detection on one embedded device, such as FPGA, for VGA video, this paper proposes a novel algorithm to speed up the detection without sacrificing the detection rate. This paper makes the contributions as following:

- A three-stage pedestrian detection system is proposed to reduce required computations and improve processing speed.
- Sobel-step is proposed to detect windows of interest based on Sobel operator by detecting edges before feature extraction.
- Uniform-LBP (Local Binary Pattern) with SVM (Support Vector Machine) replaces the HOG+SVM to distinguish the pedestrian from non-pedestrian.
- An efficient HOG algorithm, called E-HOG (Efficient-HOG), was designed to improve the detection rate.
- To evaluate the algorithm, a pedestrian detection processor with E-HOG IP is designed for testing.

The results indicate that this processor can detect pedestrians 48 fps for VGA resolution under 25 Mhz. This detection speed can satisfy the demand of real-time.

The remainder of this paper is organized as follows: relative theories are explained in section 2, the hardware system and E-HOG IP, which is based on Global Foundry logic, is addressed in section 3, the performance of E-HOG IP and detection processor are evaluated in section 4, and finally, the study's conclusions are presented in section 5.

2 Relative Algorithm

The success of Dalal's algorithm is owed to its discriminative features, but the drawback is redundant features and lengthy processing time. The sliding window always makes decisions after all pixels are calculated with given approaches. This paper first uses the proposed Sobel-step to select the windows of interest ahead of schedule. In order to decrease calculation, the uniform-LBP is used to replace the HOG. But the uniform-LBP and SBM based on uniform-LBP shows un-satisfy detection rate. Finally, the E-HOG is proposed to distinguish and improve the detection rate.

2.1 Extracting Windows of Interest Based on Sobel-step

Sobel operator is the effective descriptor for edge detection [14]. Based on the Sobel operator, Sobel-step is proposed to extract windows of interest before feature extraction, which can remove a 20%~30% area of the whole image. The following section will explain the algorithm of Sobel-step. Firstly, Sobel image is acquired by convoluted factors with gray image. The convolution factor is shown in the function (1) and (2).

$$C_X = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \textbf{(1)}$$

$$C_Y = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \textbf{(2)}$$

Let us suppose that $f(x,y)$ stands for the gray value at the coordinate (x,y). With the C_X and C_Y, the gradients in direction X and Y can be calculated by function (3) and (4):

$$\begin{aligned} G_x(x,y) = &\{f(x+1,y-1)+2f(x+1,y)+f(x+1,y+1)\}- \\ &\{f(x-1,y-1)+2f(x-1,y)+f(x-1,y+1)\} \end{aligned} \quad \textbf{(3)}$$

$$\begin{aligned} G_y(x,y) = &\{f(x-1,y+1)+2f(x,y+1)+f(x+1,y+1)\}- \\ &\{f(x-1,y-1)+2f(x,y-1)+f(x+1,y-1)\} \end{aligned} \quad \textbf{(4)}$$

According to function (3) and (4), the edge detection image can be generated by function (5):

$$D(x,y) = \sqrt{G_x^2(x,y)+G_y^2(x,y)} \quad \textbf{(5)}$$

In order to decrease the computation, the function (5) can be replaced with the approximate similarity:

$$D(x,y) = \left| G_x(x,y)+G_y(x,y) \right| \quad \textbf{(6)}$$

With the given threshold, the Sobel image can be repressed by:

$$S(x,y) = \begin{cases} 1, D(x,y) \geq Th \\ 0, D(x,y) < Th \end{cases} \quad \textbf{(7)}$$

Fig. 1 shows the image, gray image, edge detection image and Sobel image.

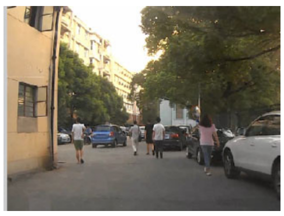


Fig. 1. (a) Image

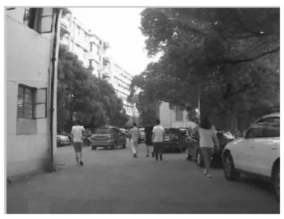


Fig. 1. (b) Gray image

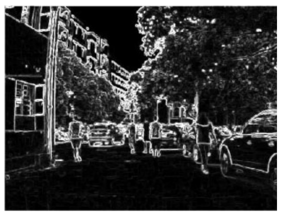


Fig. 1. (c) Edge detection image

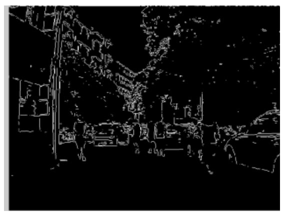


Fig. 1. (d) Sobel image

Analyzing the Fig. 1, the Sobel image can describe the edge with given threshold; however, it is not suited to human vision. So the Sobel image isn't directly used to detect the pedestrian. Considering the difference of contour edge between body and background, Sobel-step based on the Sobel image is proposed to extract the edge contour and remove non-interesting windows.

The Sobel-step operator is constructed with three steps:

Step 1. Dividing the whole image into 8x8 pixels, calling each one cell. The corresponding cells have no overlap.

Step 2. Calculating the value in one cell. Let $St(i,j)$ stand for Sobel integral value over the $i-th$ in vertical and $j-th$ in horizon cell. Thus, the $St(i,j)$ can be calculated by function (8):

$$St(i,j)=\sum_{(x,y)} S(x,y) \tag{8}$$

Step3. Setting the sliding step for the detection window through different thresholds. Additionally, V_1 and V_2 stand for the different threshold's integral value. When $St(i,j)>V_1$ and $St(i,j)<V_2$, it means that this region in the cell has edge contours and may contain a pedestrian. The top left pixel is the start pixel of one detection window.

The Sobel-step operator can be adopted as the first stage in pedestrian detection. If the integral value satisfies the different threshold values, the step of sliding window in second stage is 8. Otherwise, the step of detection window will be $S_{original}(x,y)$ plus 8 as function (9) shows:

$$S_{original}(x,y)=\begin{cases} 8, V_1<St(i,j)<V_2 \\ S_{original}(x-m,y-n)+8, else \end{cases},(m,n)\in\{8\} \tag{9}$$

Through the different values of $S_{original}(x,y)$, the windows of interest will be given ahead of schedule.

2.2 Features Extraction Based on E-HOG

Dalas and Triggers propose HOG features to describe pedestrian features, while the Linear SVM is used to classify pedestrian zone or non-pedestrian zone through stacked 3780 dimension features [2]. However, the main drawback of HOG is highly dimensional features and heavy computational cost. So, uniform-LBP replaces HOG to extract features [15] [16]. To improve the detection rate of uniform-LBP, E-HOG is proposed as the new features extraction and is located after uniform-LBP+SVM. Three main contributions of E-HOG, which is an improved approach based on HOG, are: (1) Effective and non-effective zones are divided in one detection window according to the shape hierarchy; (2) Computation of E-HOG divides each cell in a block into 4 sub-cells, classifies them into 2 types, and treats them differently. Through sub-cells, unimportant interpolation will be removed; (3) Implement different interpolations by Look-Up-Table (LUT).

The size of cell, block and window in E-HOG are still fixed 8x8 pixel, 16×16 pixel and 64×128 pixel, while the corresponding blocks have a half size overlap, as the Fig. 2 (a) demonstrated.

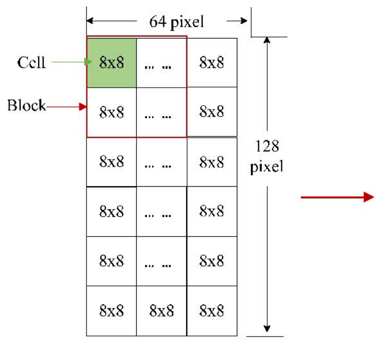


Fig. 2. (a) Cell, block and window

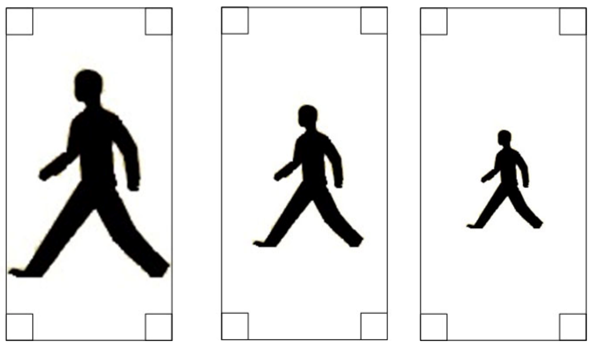


Fig. 2. (b) Pedestrian in detection window

In Dollar's paper, he groups pedestrians into three scales by using different resolution and finds that the distribution of height shows uniform distribution in the real word system detecting pedestrians. We find it interesting that the full body cannot occupy the whole 64x128 pixels because of shape hierarchy as Fig. 2 (b) showed. Considering that the pedestrian is located in the whole image, the pixels of the detection window are grouped into two types: the effective area and the non-effective area. The corner size is one cell. If the pixel is located on the four corner blocks of the detection window, these zones are non-effective zones. These pixels will not make contributions to the feature. Non-effective zones are discarded when calculating features. If the pixel is not located in the corner block, these effective zones will be included in calculating E-HOG.

The E-HOG algorithm is outlined as follows:

Input: the size of the sliding detection window and sliding step $d = 8$.

The first step of E-HOG and HOG extraction is to compute the magnitude and orientation (or angle) of gradient. Let us suppose that $|\nabla f(x,y)|$ and $\theta(x,y)$ stand for the magnitude and orientation.

The second step of E-HOG and HOG extraction is to derive the orientation histogram from the magnitudes and orientations. Each of them has a different process. When we derive the orientation histogram in HOG features, the orientation histogram of each cell has 9 bins. Trilinear interpolation of HOG, which can reduce the aliasing effect, distributes the magnitude to four cells of a block as Fig. 3 showed.

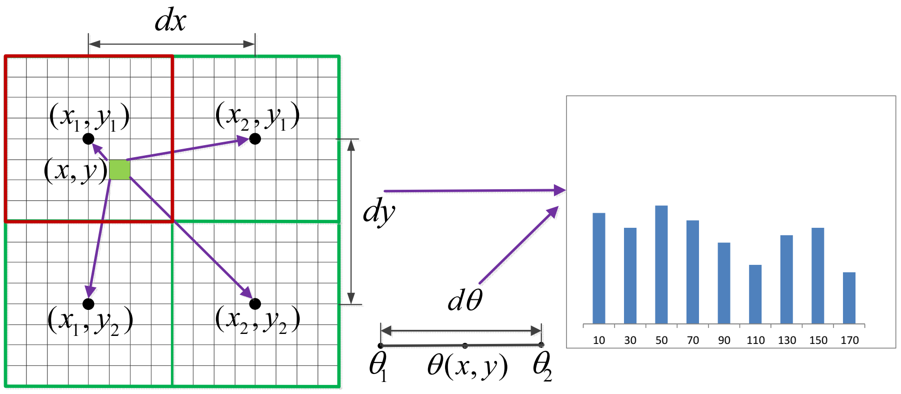


Fig. 3. Trilinear interpolation of HOG

In Fig. 3, the 4 cells of a block are identified by their centers with $((x_i, y_j), i, j = 1,2)$. Every histogram of four cells in 4 cells is represented by $h(x_i, y_j, \theta)$, where θ satisfy function (10):

$$\theta \in \{\{0:\pi/9\},\{\pi/9:2\times\pi/9\},\{2\times\pi/9:3\times\pi/9\},\{3\times\pi/9:4\times\pi/9\},\\ \{4\times\pi/9:5\times\pi/9\},\{5\times\pi/9:6\times\pi/9\},\{6\times\pi/9:7\times\pi/9\},\{7\times\pi/9:8\times\pi/9\}\} \quad \textbf{(10)}$$

Then, trilinear interpolation is given by function (11):

$$h(x_1,y_1,\theta_1) \leftarrow h(x_1,y_1,\theta_1) + |\nabla f(x,y)|(1-\frac{x-x_1}{d_x})(1-\frac{y-y_1}{d_y})(1-\frac{\theta(x,y)-\theta_1}{d_\theta}) \quad \textbf{(11)}$$

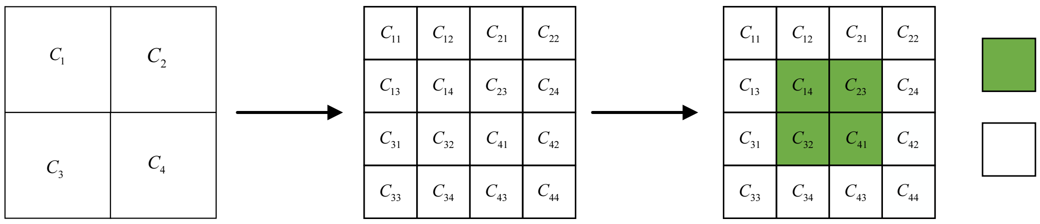
$$h(x_1,y_1,\theta_2) \leftarrow h(x_1,y_1,\theta_2) + |\nabla f(x,y)|(1-\frac{x-x_1}{d_x})(1-\frac{y-y_1}{d_y})(\frac{\theta(x,y)-\theta_1}{d_\theta}) \quad \textbf{(12)}$$

where $d_x = x_2 - x_1, d_y = y_2 - y_1, d_\theta = \theta_2 - \theta_1$.

Equations (11) and (12) are used to calculate trilinear interpolation in every block. The computation of HOG features in each detection window is time-consuming. This is the bottleneck for a real-time pedestrian detection system; therefore, E-HOG based on sub-cell and LUT is applied in trilinear interpolation to decrease the computation complexity. To avoid unimportant interpolation, every cell is divided into four sub-cells (4x4 pixels) according to location. These four sub-cells are classified into 2 types: inner sub-cells and corner sub-cells as Fig. 4 showed.

Let h_1 stand for the HOG features of cell C_1.

$$h_1 = \sum_{i=1}^{4} H(C_{1i}) + hist(C_{21}) + hist(C_{32}) + hist(C_{41}) \quad \textbf{(13)}$$



C_1
C_2
C_3
C_4
C_{11} C_{12} C_{21} C_{22}
C_{13} C_{14} C_{23} C_{24}
C_{31} C_{32} C_{41} C_{42}
C_{33} C_{34} C_{43} C_{44}

Fig. 4. Sub-cells in E-HOG

Next, the question is how to compute the result of Equations (11) and (12). Equations (11) and (12) can be changed to Equation (14):

$$\begin{cases} |\nabla f(x,y)|(1-\dfrac{x-x_1}{d_x})(1-\dfrac{y-y_1}{d_y})(1-\dfrac{\theta(x,y)-\theta_1}{d_\theta}) = |\nabla f(x,y)| P_A P_B P_\theta \\ P_A = 1-\dfrac{x-x_1}{dx} \\ P_B = 1-\dfrac{y-y_1}{dy} \\ P_\theta = \dfrac{\theta(x,y)-\theta_1}{d_\theta} \end{cases} \quad \textbf{(14)}$$

When dealing with P_A and P_B, these two parameters can be calculated off-line and stored into LUT. When dealing with P_θ, every bin is averaged to 4 sub-bins for hardware implementation. Magnitudes, which are given the parameters of {[3/4, 1/4], [1, 0], [1, 0], [3/4, 1/4]} as the factors, are mapped into the corresponding bins. Simple shift and addition can address this calculation to avoid using multipliers.

In the last step, the L2-Uniform is used in E-HOG to normalize all features.

2.3 Three Stages in Pedestrian Detection System

The bottleneck for real-time pedestrian detection is a large amount of data. Local Binary Pattern (LBP) always is adopted to deal with the pedestrian occlusion [15]. In this system, LBP features are used to replace the HOG features and uniform-LBP, which is one kind of LBP and can generates only 59 dimensions at most in one 16x16 pixel, is selected to generate the 1888 dimensions in one detection window [16]. If the uniform-LBP is used to determine the “pedestrian” zones, this operator can decrease the amount of calculation because of the less dimension features. But the error rate of uniform-LBP+SVM is worse than the E-HOG+SVM. So, E-HOG+SVM are located behind the uniform-LBP+SVM to approve the detection rate.

Based on the discussion above, this real-time pedestrian detection system is a combination of three stages: Sobel-step in the first stage; uniform-LBP+SVM in the second stage; and E-HOG+SVN in the third stage, as Fig. 5 showed. This detection system is outlined as follows:

Input: The input image is approximately 640×480 pixel.

Output: The location of different sizes, which are annotation to pedestrians.

Step 1. The scaled input image at a fixed scale.

Step 2. Input image is inserted into the first stage. Sobel-step calculates the sliding step $S_{original}(x,y)$ of the detection window in the second stage.

Step 3. From top to bottom and left to right, scan the whole image with 64×128 based on $S_{original}(x,y)$. Extract the 1888 uniform-LBP features from the sub-image covered by the detection (scanning) window and then apply the learned SVM classifier to the LBP feature vector to give the result $Sc_{LBP}(x,y)$, where $Sc_{LBP}(x,y) \in \{-1,+1\}$. The uniform-LBP features are generated cell by cell and the SVM is stimulated to provide the result, whether is the “pedestrian” zone or “non-pedestrian” zone.

Step 4. From top to bottom and left to right, scan the whole image with 64×128 based on $Sc_{LBP}(x,y) \in \{-1,+1\}$. Extract the 3204 E-HOG features from the sub-image covered by the detection (scanning) window and then apply the learned SVM classifier of the high-dimension E-HOG feature vector to classify the sub-image as pedestrian or non-pedestrian.

Step 5. Non-Maximum Suppression (NMS) is applied to fuse the sub-detection windows into one detection window as one annotated pedestrian-zone.

3 Real-Time Pedestrian Detection Architecture on Board

Fig. 5 shows a flow diagram of pedestrian detection using the E-HOG algorithm. When transplanting this system to FPGA, especially for low series of FPGA, the total of resources is the key restriction. HOG is the basic algorithm for extracting features, so, the first and second stage will be implemented on Cyclone III and E-HOG is designed as hardware IP. This system can be used as the real-time pedestrian detection system on-board. Compared to the others system, this system is on-boarded system can real-time detect human.

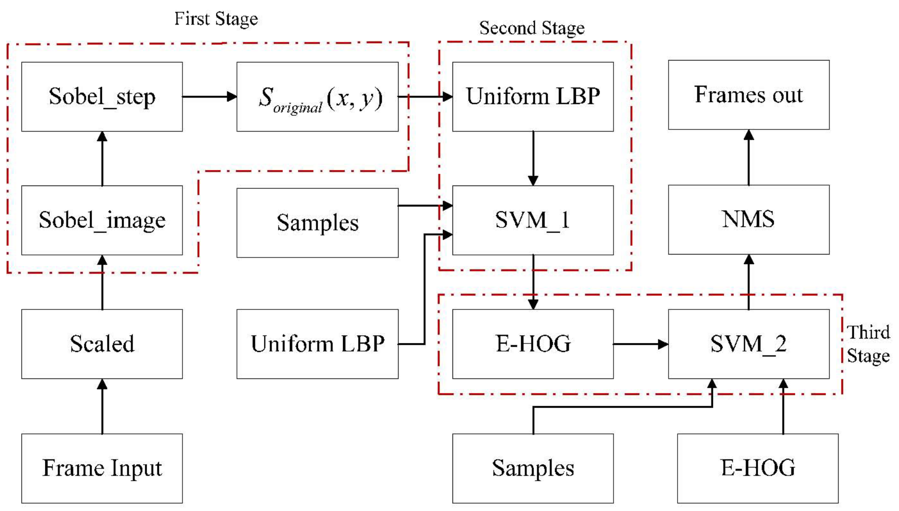


Fig. 5. Structure of pedestrian detection

3.1 Three Stages in Pedestrian Detection System

E-HOG is the key module for the pedestrian detection system. The performance of E-HOG in detection and speed is the primary determinant of performance of the detection system. Fig. 6 shows a flow diagram of detection using the E-HOG algorithm.

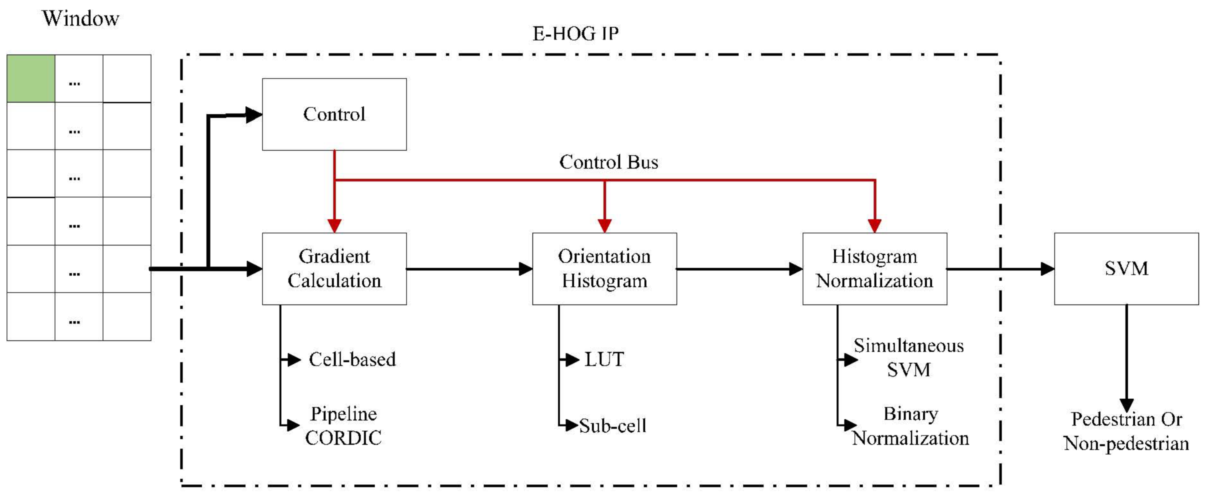


Fig. 6. E-HOG IP

E-HOG is modified based on the HOG using the following approaches:

Cell-based calculation. E-HOG features are extracted using cell-based calculation. This means that one cell at a time is calculated during E-HOG processing. Sharing and reuse of a cell have great impact on calculation and memory.
Pipeline CORDIC for orientation and magnitude. To achieve simplicity of hardware realization when rotating, the key ideas used in Coordinate Rotation Digital Computer (CORDIC) arithmetic are:

(a) Decompose the rotations into a sequence of elementary rotations through predefined angles that can be implemented with minimum hardware cost.

(b) Avoid scaling that might involve square-root or division operations.

The number of iterations and accuracy are directly related but are in inverse proportion to the processing speed. In this IP, the number of iteration is 12 to make sure of the accuracy of results, while 11 stages pipelines are inserted into the CORDIC algorithm to promote processing speed.
LUT and Sub-cell in trilinear interpolation. Calculation of tri-linear interpolation can consume more than 70% of the total feature extraction time. In order to reduce processing time, the tri-linear interpolation is modified using LUT and sub-cell as shown in Fig. 7.

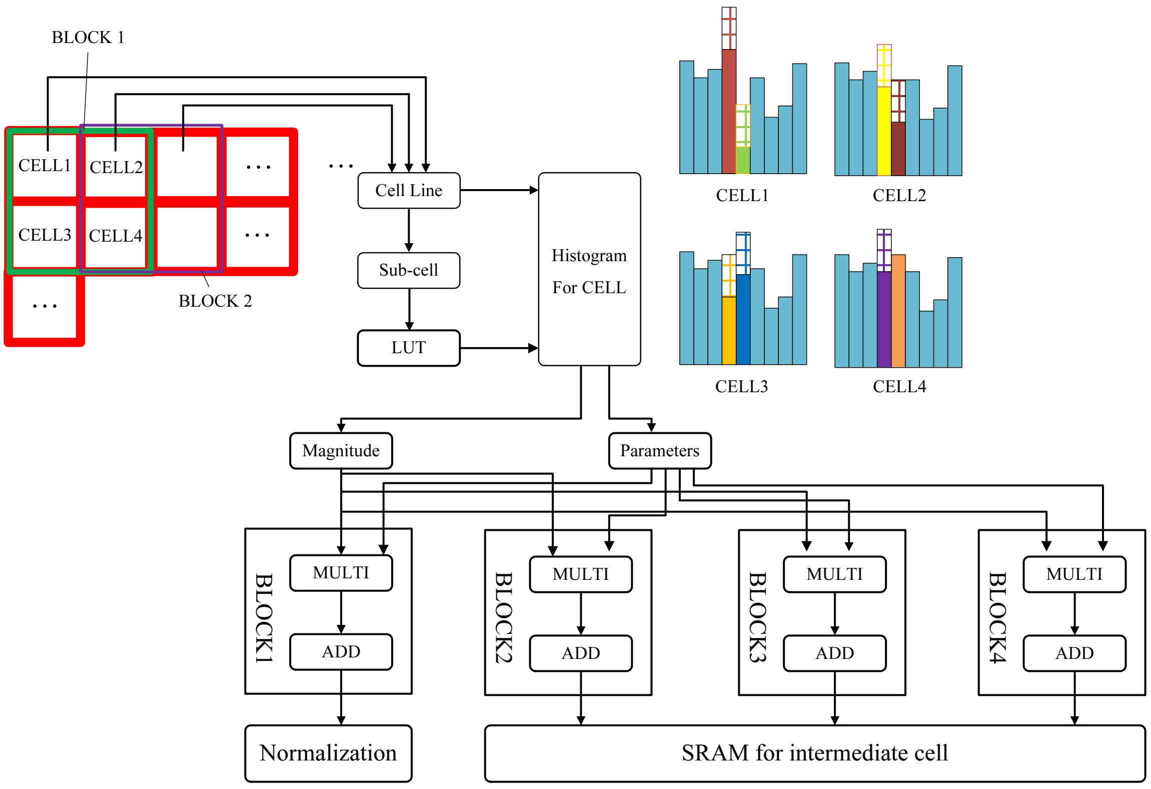


Fig. 7. Structure of E-HOG

Normalization based on binary. The normalization of E-HOG is performed by dividing a fixed histogram by L2 Norm. This operation requires square root calculation and division. To avoid these calculations, the binary normalization is used as:

$$Nh(i,j)=\begin{cases}1, hist(i,j)>d\\0, hist(i,j)<d\end{cases} \tag{15}$$

While $d=\sum_{i=1}^{4}\sum_{j=0}^{8} hist(i,j)$. d stands for the integral values of all histogram in one block and $Nh(i,j)$ for the histogram after normalization.

3.2 Architecture for pedestrian detection processor

Fig. 8 exhibits the whole pedestrian detection processor with E-HOG IP.

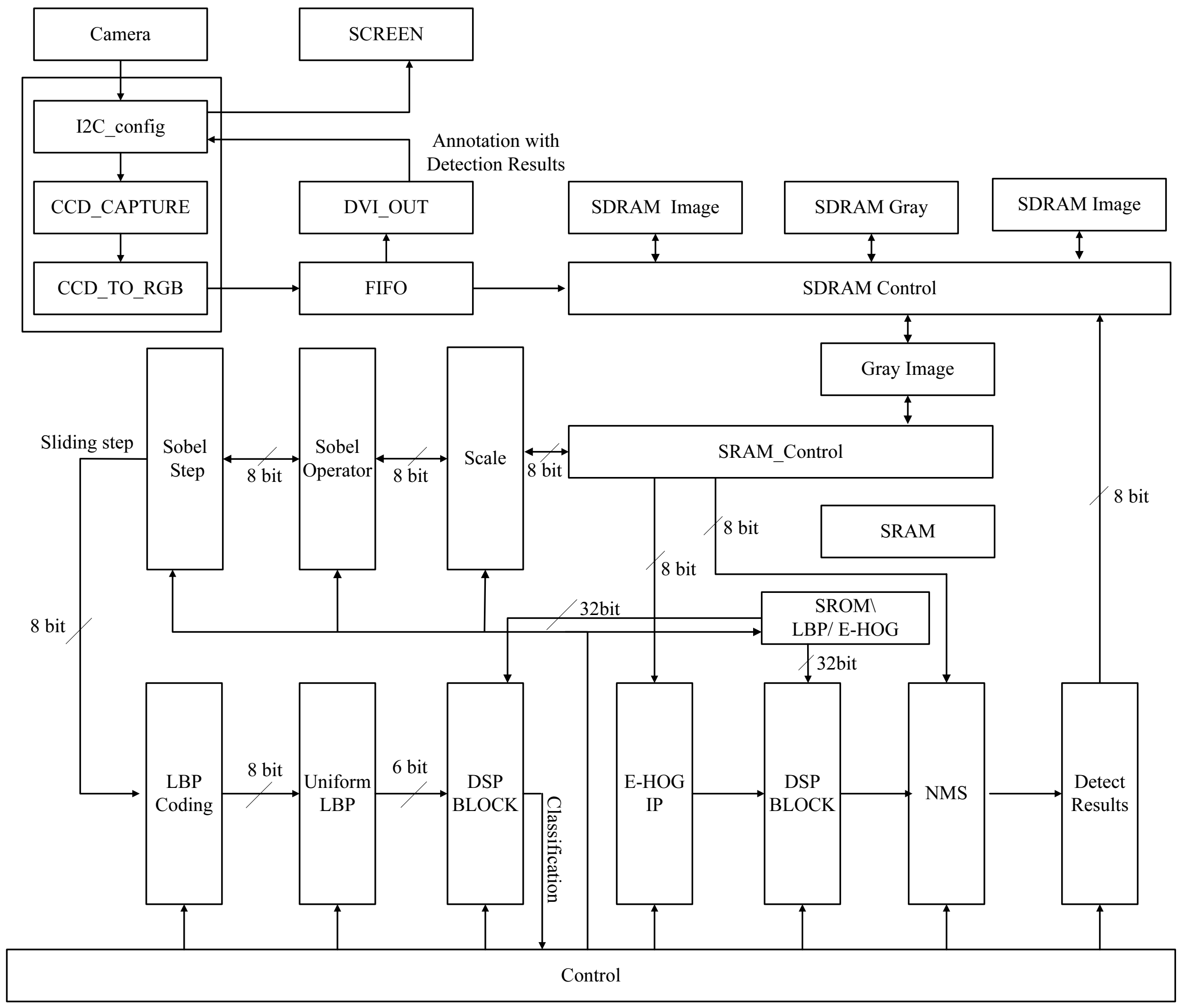


Fig. 8. Architecture of pedestrian detection processor

The proposed architecture contains the main controlling and memory modules: a Control, SDRAM Control, Camera Control (I2C_config module, CCD_CAPTURE module, CCD_TO_ R GB module, one DVI interface, one FIFO, ONE SRAM and some SRAM for intermedia data, SROM for SVM coefficient of Uniform-LBP and E-HOG. The detection system also contains data processing modules: Scale Module, First Stage (Sobel Operator and Sobel Step Module), Second Stage (LBP coding, Uniform-LBP and DSP BLOCK for classifying Uniform-LBP features), Third Stage (E-HOG, DSP_BLOCK, DSP BLOCK for classifying E-HOG features), NMS (Non-Maximum Suppression) and Detection window.

Two SDRAMs are used to store the image from the camera using ping-pong strategy. The left SDRAM is used to store the Gray image. The internal modules process input frames and then output annotated frames with detection results.

The I2C_config, CCD_CAPTURE and CCD_TO_RGB modules receive the frames, which are captured by camera. The DVI_OUT module rejects the frames which are annotated by the detection system into SCREEN, as Fig. 8 shows. Fig. 9 presents pipeline flow based on the three stages.

It vividly shows the relationship between three stages, windows and the frames. Pipeline processing is conducted as follows:

(1) Frames are inputted into the two SDRAMS by the sdram_control module using ping-pong strategy.

(2) GRB frames are stored in the third SDRAM.

(3) The Sobel-step is generated by the first stage with data from SDRAM, and stores the internal image to the SRAMs.

(4) Based on results of the first stage, the step of the uniform-LBP and SVM detection window is determined. In order to decrease calculation, the uniform-LBP is operated cell by cell. As Fig. 9 shows, the LBP is first calculated through every 16x16 pixel, cell by cell. Then, according to the results of sliding steps, different cells and corresponding factors are classified.

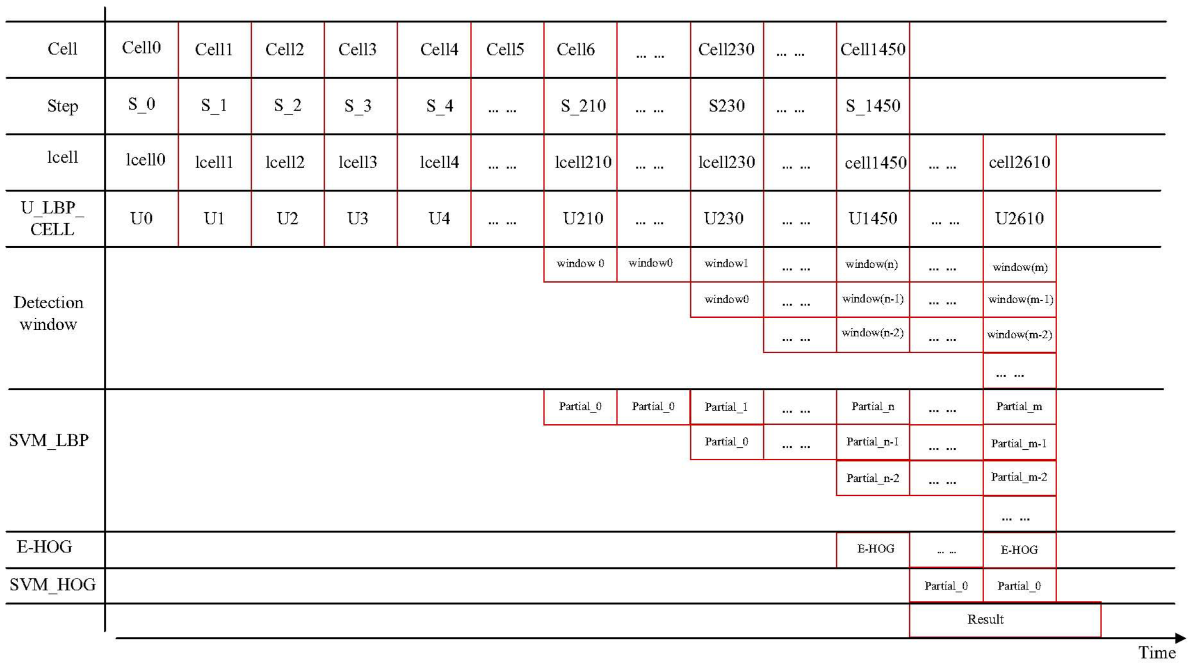


Fig. 9. Three-stages and cell-based pipeline flow

Compared to the cell in HOG, the cell uniform LBP has the same time, but the corresponding cell has no overlap. The step of a cell is 16 pixels both vertical and horizontal. One cell generates 59 dimension features. One detection window will generate 1888 dimensions.

(5) The uniform-LBP with SVM coefficient can classify the “pedestrian” or “non-pedestrian”.

The proposed system has 16 classification cores. One classification core addresses 2 blocks of MAC operation. One detection can manage 32 blocks efficiently. Sufficient parallelism reduces the required cycle count.

(6) In order to improve the detect rate, the E-HOG IP with SVM coefficients finally determines whether “pedestrian zone” or not.

This pedestrian detection system greatly reduces memory bandwidth because it prevents reloading input pixels and accelerates processing speed.

4 Experiments Results

In section 2, Sobel-step (see section 2.1), E-HOG (see section 2.2) and three stages of the system (see section 2.3) are detailed explained. In section 3, the IP and detection processor are described. So, the performance of E-HOG and the pedestrian detection system will be evaluated separately. Next, evaluation of IP and system implementation on board is discussed with the frames from the outdoor scene. Note that HOG plus SVM are the classic structures for pedestrian detection, so the Linear SVM will be used for LBP and HOG as the classifier.

Before evaluating, the pedestrian detection data set must be chosen. It is well-known that INRIA is the first and most frequently used set for evaluating the performance of algorithms [10]. INRIA dataset contains training set and testing set to train linear SVM and test the performance. However, the INRIA data set contain background which sometimes does not agree with the domestic. Other than INRIA data set, several images are still gathered to match domestic background. In this evaluation, 3,000 images were used for offline training, while 14,000 images were used as the evaluation data. The size of training and test images are 64X128 pixels.

The evaluation performance on PC can be divided into two parts: theory evaluation and hardware evaluation. Section 4.1.1 and 4.1.2 shows the theory evaluation. Section 4.1.3 shows the hardware evaluation on PC. Section 4.2 shows the evaluation of E-HOG IP and detection system on board.

4.1 Function performance

The evaluation performance on PC can be divided into two parts: theory evaluation and hardware evaluation. Section 4.1.1 and 4.1.2 shows the theory evaluation. Section 4.1.3 shows the hardware evaluation.

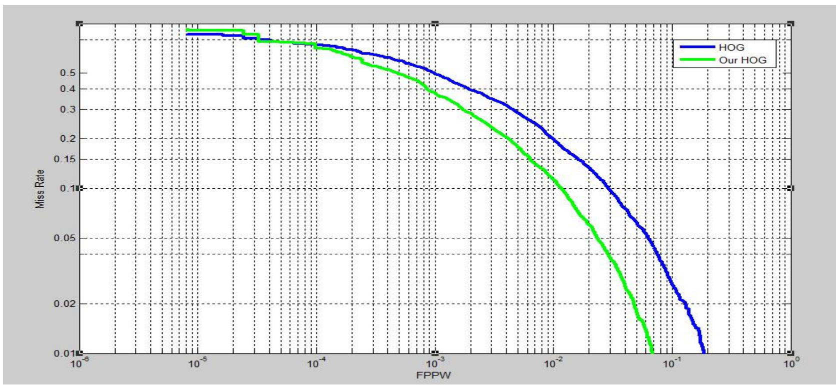
Evaluation of E-HOG. The performance of E-HOG is evaluated using the Matlab 2013a.The first part is performance evaluation of this HOG algorithm. CORDIC based Parameters-fusion LUT HOG is improved based on the HOG. Systems for pedestrian detection, HOG+ Linear SVM and this HOG+ Linear SVM, are compared to each other to distinguish their functions.

The method for quantification is to report miss rates versus false positives per window, calling DET curves of different algorithms. Lower values are better.

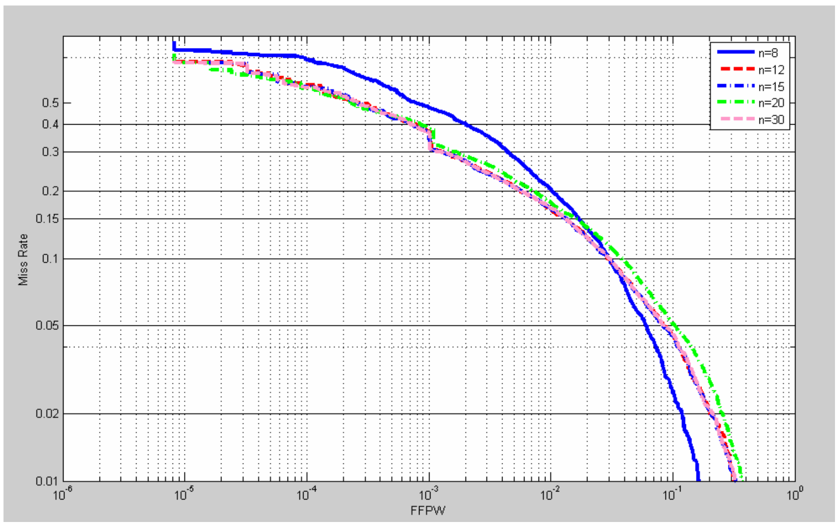
As Fig. 10 shows, the DET curves from HOG+ Linear SVM and Our HOG+ Linear SVM are used to qualify algorithm performance. As Fig. 10 (a) shows, Our HOG is equivalent to the Dalal's HOG at 10^{-4} FPPW. So, the CORDIC based Parameters-fusion LUT HOG can be used to extract an image's feature without performance sacrifice.

The next step is to evaluate the performance of this HOG with different numbers of stages in CORDIC. Fig. 10 (b) shows the impact of different numbers of stages on performance using DET curve.

Fig. 10 (b) shows that if the number of stages in CORDIC is larger than 12, the performance improvement is minor. This minor improvement is not worth the sacrifice of size. So, the CORDIC algorithm will perform 12 iterations to complete calculation of magnitude and orientation. The conclusion from the Fig. 10 is that the E-HOG can be used as the effective feature descriptor.



(a) DET curves of HOG and E-HOG



(b) DET curves with different iteration

Fig. 10. Different DET curves

Theory evaluation of the system. The critical evaluation of a pedestrian detection system is FPPI (Miss Rate vs False Positive Per Image) using Matlab 2013a. There are approximately 16 types of representative features used to detect pedestrians. The HOG and HOG-LBP in this paper are both state of art features for evaluating performance. From Fig. 11, the three-stage structure outperforms the other three algorithms.

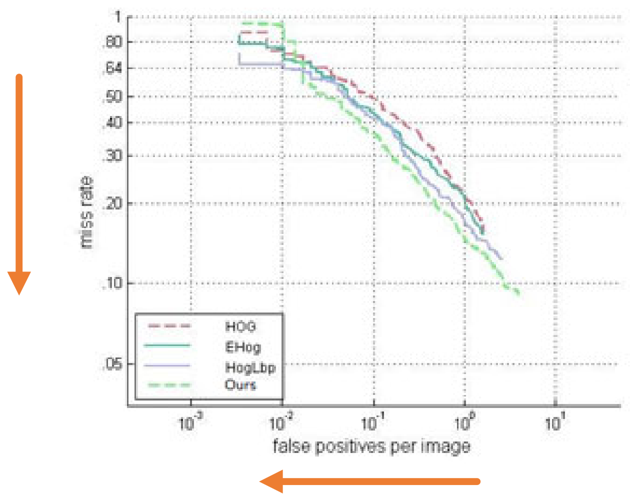


Fig. 11. Performance of different methods

Hardware performance evaluation of proposed system.

(1) E-HOG IP
The evaluation of integrated circuits E-HOG IP is chosen by FPGA. Firstly, the E-HOG is realized on the FPGA-Cyclone III in a Verilog HDL simulation and Altera Quartus II 15. Table 1 lists the hardware overload using the FPGA with the E-HOG hardware IP. Table 2 lists the result report of traditionally HOG from the Dollar's paper.

From the Table 1 and Table 2, the E-HOG can decrease 23.4% hardware cost than the traditionally HOG. The memory size just has the one third memory size of traditionally HOG.

Table 1. FPGA synthesis result report

Slice logic	Used	Utilization (%)
Number of slice registers	3551	22
Total Combination function	6313	41
Total Memory bits	5120	1

Table 2. Traditionally HOG FPGA report

Slice logic	Used	Utilization (%)
Number of slice registers	4657	29
Total Combination function	8156	53
Total Memory bits	16384	3

(2) Evaluation of processor
A Verilog HDL simulation and Altera Quartus II 15.0 is also used to estimate the number of cycle counts.

The results of estimation are presented in Fig. 12. In Fig. 12, HOG+SVM stands for the traditional method; Approved stands for the three stages without cell-based and pipeline; Optimization stands for the three stages and FPGA implementation. The cycle count is standard with 10^6.
From the Fig. 12, the features extraction with simultaneous SVM classification enables the reuse of intermediate results and reduces the processing time. The overall process requires about 0.52×10^6 cycles

per frame. Based on these, this proposed system implementation in FPGA can process VGA resolution video at 48 fps using 25 Mhz.

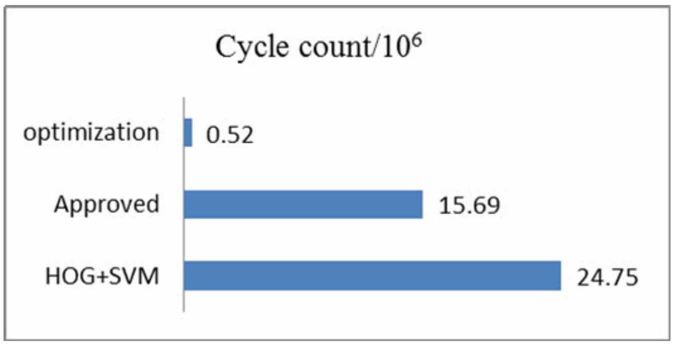


Fig. 12. Cycle count of different methods

Table 3 shows the performance of difference pedestrian detection on board. As Table 3 shows, our system can process 48 fps (VGA) based on 25Mhz. This implementation shows the better performance than [13, 17-18] with minimum operating frequency. Compared to [19], our FPGA processor aided by E-HOG still performs well with less memory usage and hardware overload.

Table 3. Results of FPGA implementation

		[17]	[18]	[13]	[19]	Ours
FPGA	LUT/LEs	28,495	34,403	34,838	16392	14895
	Registers	5,980	23,247	22,612	10900	9800
	DSP Block	2	68	N/A	1	40
	Memory/Mbits	1.08	0.34	2.094	0.7	0.28
Frames rates		38	20	72	50	48
Resolutions		320X240	640X480	800X600	1280X1024	640X480
Frequency(Mhz)		167	70	40	60	25
Serials		Virtex-5	Cyclone III	Cyclone IV	Cyclone III	Cyclone III
Company		Xilink	Altera	Altera	Altera	Altera

4.2 FPGA Implementation Aided by E-HOG IP

E-HOG IP. Except the evaluation using FPGA, another method is adopted to evaluate the performance through ASIC. Fig. 13 shows the E-HOG IP.

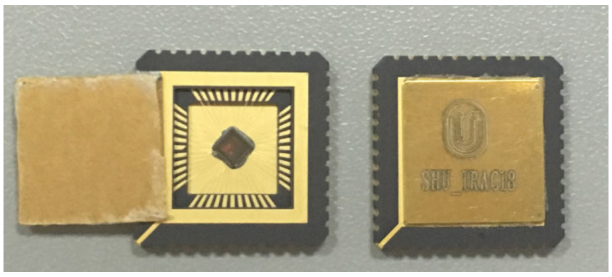


Fig. 13. E-HOG IP

This IP is developed with ceramic leaded chip carrier (CLCC44), type chip of 1cm×1cm, based on the 180nm, 1-poly/4-metal CMOS logic process. This ASIC IP is labeled SHU-IRAC18. SHU-IRAC18 contains one test chain using the technology of design for test (DFT).About the test memory size, simple coding is wrote to test whether SRAM is right or not.
Fig. 14 shows the results waveform of E-HOG IP using Signal Tap II.

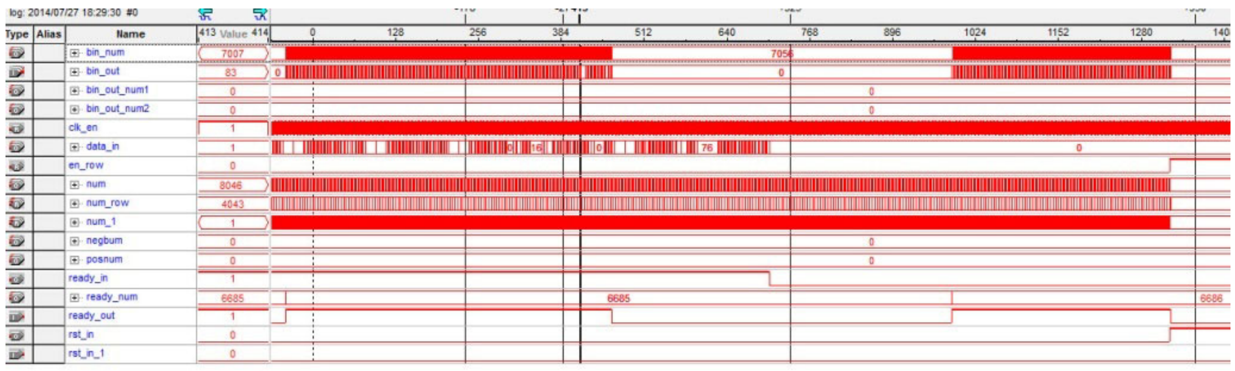


Fig. 14. Signal Tap II waveform

Pedestrian detection processor. To evaluate the effectiveness of this system, the proposed architecture is implemented on our self-developed board as the Fig. 15 and Fig. 16 show.

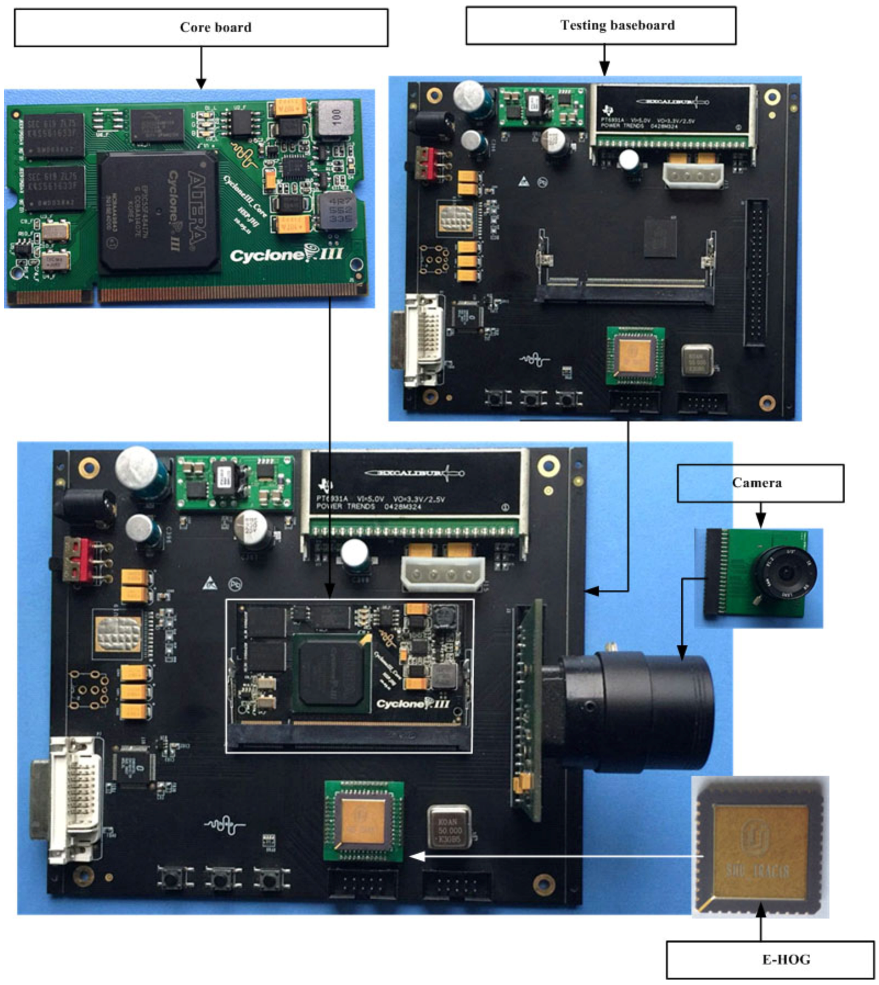


Fig. 15. FPGA board with E-HOG IP

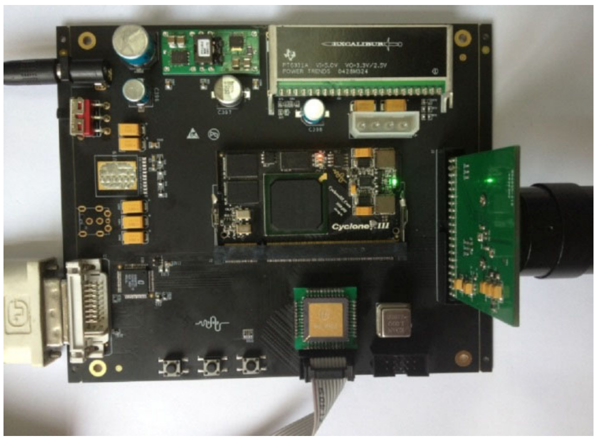


Fig. 16. Architecture verification by FPGA and E-HOG IP

When designing the hardware system, the circuit board is divided into two boards: Cyclone Ⅲ core and backboard. Through two independent boards, the backboard and core board can be extended to other functions. If this platform were to be promoted, just one of them may need to be amended. The core board contains one FPGA, two SDRAM and some other devices. The core device is Cyclone Ⅲ-EP3C55F48417.The backboard main devices are MT9 M111, one SRAM, power and interfaces for debugging and downloading program. E-HOG is embedded into the system on the backboard. The connection between backboard and core board is SODIMM200.

Fig. 17 shows that the detection processor is working when the green LED is burned. Fig. 18 shows the partial results of the pedestrian detection processor.

Resources utilization and comparison on FPGA are presented in Table 3. This proposed implementation on FPGA can detect pedestrians with 25 Mhz at 48 fps and shows the best performance with minimum usage and minimum operating.

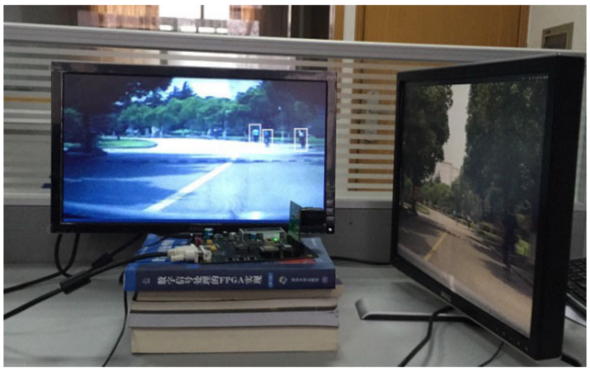
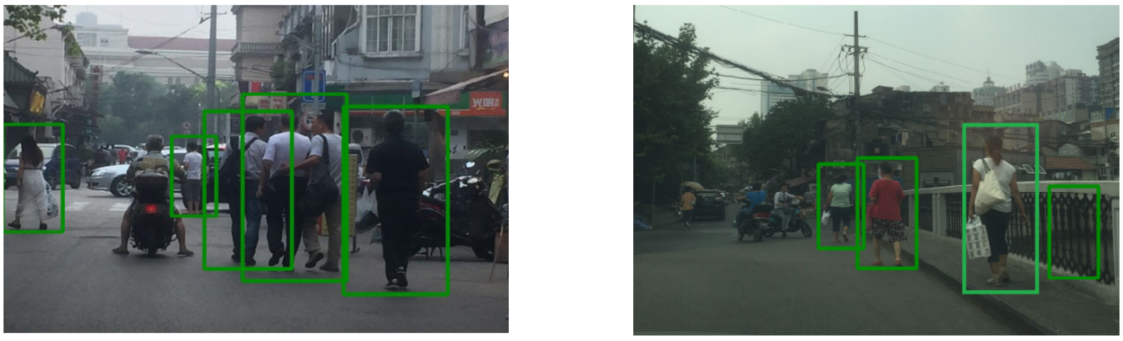


Fig. 17. Working processor



(a) (b)

Fig. 18. Partial results of the pedestrian detection processor

The testing video contains about 988 persons. The results of pedestrian detection processor are that: the number of right detection is 861; the number of miss detection is 93; the number of error detection is 13. The detection rate contains error rate and miss rate.
Analysis from Table 4, the conclusion is that the miss rate is about the 9.41% and the error rate is about 1.3%. The statistical data shows that the performance of this pedestrian detection shows the better than HOG+SVM (miss rate: 28%; error rate: 1.5%). So, this processor can detect the pedestrians with the 9.41 miss rate and performs better than other algorithms.

Table 4. Detection results of pedestrian detection processor

method	Total/Person	Detection/Person	Miss/Person	Error/Person
Three-stages	988	895	93	13

5 Conclusion

This paper has discussed a three-stage pedestrian detection on board that can be used as the real-time pedestrian detection processor. The three-stages contains: Sobel-step is applied as the first stage operator and to extract windows of interest; Uniform-LBP with SVM in the second stage can describe edge detection with less features; In the third stage, E-HOG and Linear SVM will distinguish the pedestrian zone from non-pedestrian. In order to decrease the cycle counts, cell-based is proposed to implement the uniform-LBP. When this proposed system is installed on-board with E-HOG IP, experimental results show that this processor can satisfy real-time pedestrian detection without sacrificing accuracy.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (Grant 61376028) and Shanxi University Science and Technology Innovation Project (Grant 20140063).

References

[1] P. Viola, M. J. Jones, D. Snow, Detecting pedestrians using patterns of motion and appearance, in: Proc. 2005 International Journal of Computer Vision, 2005.

[2] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Proc. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.

[3] P. Dollar, C. Wojek, B. Schiele, P. Perona, Pedestrian detection: An evaluation of the state of the art, IEEE Transactions on Pattern Analysis and Machine Intelligence 34(4)(2012) 743-761.

[4] C. Wojek, B. Schiele, A performance evaluation of single and multi-feature people detection, in: Proc. Joint Pattern Recognition Symposium, 2008.

[5] P. F. Felzenszwalb, R. B. Girshick, D. McAllester. Cascade object detection with deformable part models, in: Proc. Computer Vision and Pattern Recognition, 2010.

[6] J. Schlosser, C.K. Chow, Z. Kira, Fusing LIDAR and images for pedestrian detection using convolutional neural networks, in: Proc. 2016 IEEE International Conference on Robotics and Automation, 2016.

[7] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-c. Loy, X. Tang, Deepid-net: deformable deep convolutional neural networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[8] R. Benenson, M. Omran, J. Hosang, B. Schiele, Ten years of pedestrian detection, what have we learned?, in: Proc. European Conference on Computer Vision, 2014.

[9] W. Ouyang, X. Wang, Joint deep learning for pedestrian detection, in: Proc. the IEEE International Conference on Computer Vision, 2013.

[10] Navneet Dalal, INRIA person dataset. <http://pascal.inrialpes.fr/data/human/>.

[11] Computational Vision at CALTECH, Caltech pedestrian detection benchmark. <http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/>.

[12] Joseph Chet Redmon, Pascal VOC dataset mirror. <http://pjreddie.com/projects/pasccal-voc-dataset-mirror>.

[13] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguchi, M. Yoshimoto, An FPGA implementation of a HOG-based object detection processor, IPSJ Transactions on System LSI Design Methodology 6(2013) 42-51.

[14] W. Gao, X. Zhang, L. Yang, H. Liu, An improved Sobel edge detection, in: Proc. 3rd IEEE International Conference on-Computer Science and Information Technology (ICCSIT), 2010.

[15] X. Wang, T.X. Han, S. Yan, An HOG-LBP human detector with partial occlusion handling, in: Proc. 2009 IEEE 12th International Conference on Computer Vision, 2009.

[16] F. Lu, J. Huang, An improved local binary pattern operator for texture classification, in: Proc. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing, 2016.

[17] M. Hiromoto, R. Miyamoto. Hardware architecture for high-accuracy real-time pedestrian detection with Co-HOG features, in: Proc. 2009 IEEE 12th International Conference on Computer Vision Workshops, 2009.

[18] K. Negi, K. Dohi, Y. Shibata, K. Oguri, Deep pipelined one-chip FPGA implementation of a real-time image-based human detection algorithm, in: Proc. 2011 International Conference on Field-Programmable Technology (FPT), 2011.

[19] Y.Li, K. Gai, M. Qiu, W. Dai, M. Liu, Adaptive human detection approach using FPGA-based parallel architecture in reconfigurable hardware. <http://onlinelibrary.wiley.com/doi/10.1002/cpe.3923/abstract>, 2016.