IMPLEMENTATION OF HOG BASED FEATURE EXTRACTION METHOD AND ITS VLSI ARCHITECTURE

J.Thilagavathy
Electronics and communication Engineering,
Dr.SivanthiAditanar College of Engineering

ABSTRACT

Human detection on emerging intelligent transportation systems is a challenging task in hardware implementation. The histogram of oriented gradients (HOG) based human detection is the most successful algorithm due to its superior performance. Unfortunately, more intensive computations and poor performance at a multi-scale and low-contrast makes human detection more difficult and unreliable. To address the above-stated problems, an efficient histogram of edge oriented gradients based human detection is proposed for preserving the edge gradients at low-contrast and to support the multi-scale detection. The proposed algorithm uses approximation methods and adopts pipelined structure that utilizes low-cost and high-speed respectively. Experiments conducted on various challenging human datasets, shows that the outcome of the proposed method provides efficient detection. This algorithm has been synthesized on Xilinx Spartan 3 FPGA software and board achieves better hardware utilization compared to other state-of-the-art approaches.

KEYWORDS: Real-time pedestrian detection, HOG, Co-HOG, Neural Network Classifier.

1.INTRODUCTION

Detecting humans in images is a challenging task owing to the invariable appear ance and the wide range of poses that they can adopt. The first need is a robust feature set that allows the human form to be discriminated cleanly, even in cluttered backgrounds under difficult illumination. We study the issue of feature sets for human detection, showing that locally normalized Histogram of Oriented Gradient (HOG) descriptors provide excellent performance relative to other existing feature sets including wavelets. The proposed descriptors are reminiscent of edge orientation histograms, SIFT descriptors and shape contexts, but they are computed on a dense grid of uniformly spaced cells and they use overlapping local contrast normalizations for improved performance. We make a detailed study of the effects of various implementation choices on detect or performance, taking “pedestrian detection” (the detection of mostly visible people in more or less upright poses) as a test case. For simplicity and speed, we use linear SVM as a baseline classifier throughout the study. Then we detect or give essentially perfect results on the MIT pedestrian test set, so we have created a more challenging set with a large range of poses and backgrounds.

Recently, human detection has been an important issue and has been widely used in many applications, such as surveillance, automotive systems, and robotics. A robust human detection system in intelligent transportation systems is desiredby people and becomes essential to industries.

2.EXPERIMENTAL SECTION

2.1.HISTOGRAM OF GRADIENT

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

computation is 8 × 8 pixels. Fig. 2 summarizes the computation of HOG feature extraction. It can be divided into the following three steps

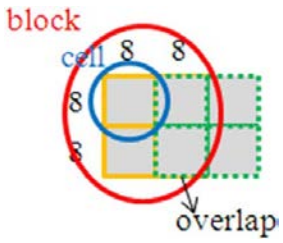


Fig.1.Cells and blocks for HOG feature extraction

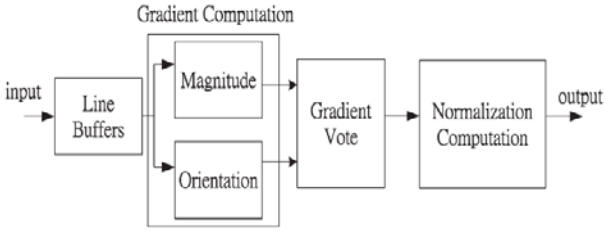


Fig.2. Hardware Architecture of HOG

2.1.1.Gradient computation

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

$$(x, y) = f(x + 1, y) - f(x - 1, y) \quad (1)$$

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

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

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

The direction *θ*(*x*, *y*) can be computed by

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

2.1.2.Gradient vote

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

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

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

$$m_n = (1 - \alpha) * m(x, y) \quad (6)$$

$$m_{nearest} = \alpha * m(x, y) \quad (7)$$

2.1.3.Normalization computation

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

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

where *i*is a number from 1 to 36 (four cells × nine bins), v_i is the vector corresponding to a combined histogram for a block region, $\| v \|_2^2 = v_1^2 + v_2^2 + \ldots + v_{36}^2$, and ε is a small constant to avoid dividing by zero. Obviously, many complex and floating-point operations are needed to determine the parameters if the direct implementation of HOG feature extraction is adopted.

2.2.HARDWARE IMPLEMENTATION OF HOG ALGORITHM

2.2.1.Sobel edge detection algorithm

The **Sobel operator**, sometimes called the Sobel–Feldman operator or Sobel filter, is used in image processing and computer vision, particularly within edge detection algorithms where it creates an image emphasising edges. At each point in the image, the result of the Sobel–Feldman operator is either the corresponding gradient vector or the norm of this vector.

The Sobel–Feldman operator is based on convolving the image with a small, separable, and integer-valued filter in the horizontal and vertical directions and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation that it produces is relatively crude, in particular for high-frequency variations in the image.

The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives – one for horizontal changes, and one for vertical. If we define **A** as the source image, and **G***x*and **G***y*are two images which at each point contain the vertical and horizontal derivative approximations respectively, the computations are as follows:

$$VM = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

$$HM = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

2.2.2.Sobel Filter for Gradient Calculation

The computation of gradient value and gradient direction must have low complexity and fit hardware-based implementations. The Sobel operator is used for calculating the gradient-values in directions by convolution with a separable 3*3 Sobel mask and yields the spatial derivatives for each pixel of the raw image. The gradient magnitude for each pixel is simplified to avoid the root operation.

$$M(i, j) = |G_x(i, j)| + |G_y(i, j)| \qquad (9)$$

The incoming pixel is at the centre of the mask and the line buffers produce the neighboring pixels in adjacent rows and columns. The two filters for G_x and G_y works in parallel and from their output data $M(i, j)$ can be obtained in the calculation unit.

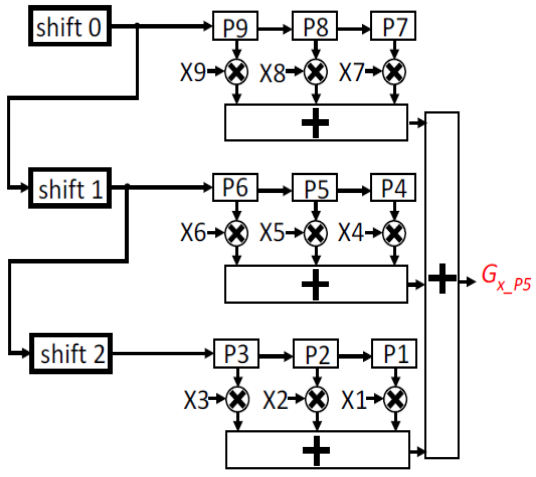


Fig.3. Gradient Calculation in Sobel Filter

2.2.3.Sobel Filter for Orientation Calculation

The calculationof the arc-tangent function is computationally expensive,especially for hardware implementation. Most of the hardware friendly approximation algorithms adopt aniterative architecture resulting in deceleration of the system,or a lookup-table (LUT) method requiring large amounts ofmemory.

$$\tan\theta_1 < \frac{G_y}{G_x} < \tan\theta_2 \qquad (10)$$

$$G_x.\tan\theta_1 < G_y < G_x.\tan\theta_2 \qquad (11)$$

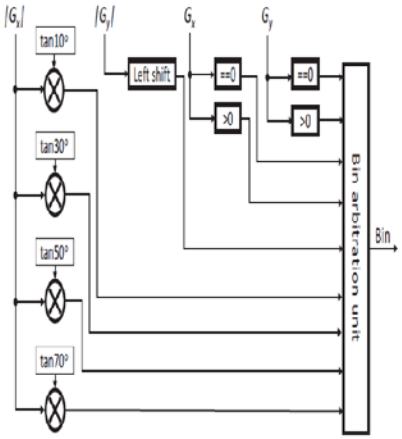


Fig.4. Orientation Calculation in Sobel Filter

2.2.4.Hardware Architecture for Magnitude Computation

Gradient magnitude computation of each pixel has two-square and one-square root operation, requires high hardware cost. To reduced error and hardware cost, square root approximation (SRA) technique used. The magnitude $G(x, y)$ is computed using

$G(x, y) = \max((0.875\,a + 0.5\,b), a)$

where, $a = \max(Gx(x, y), Gy(x, y))$ and $b = \min(Gx(x, y), Gy(x, y))$.

CMP1 block provides the maximum and minimum value of the gradients. CMP2 block gives the maximum value as gradient magnitude.

The computation of gradient value and gradient direction must have low complexity and fit hardware-based implementations. The incoming pixel is at the center of the mask and the line buffers produce the neighboring pixels in adjacent row and columns.

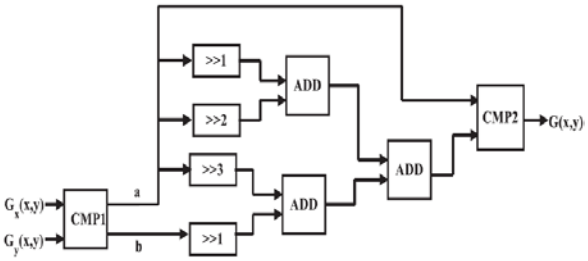


Fig.5. Magnitude computation architecture

2.2.5.Hardware Architecture for Orientation Computation

The shift based orientation method is employed to realize with reduced hardware cost, initially the orientation is uniformly spaced over 0 °−180 °and is divided into nine bins. Each bin lies between two different angles. The values of bin boundaries are given in below table and the value are discerned using a shift-based technique.

The angles between 0°and 90°lie in first quadrant and remaining angles till 180°lie in the second quadrant. The magnitude values of both quadrants are same but differ in polarity. Polarity decides the quadrant value.

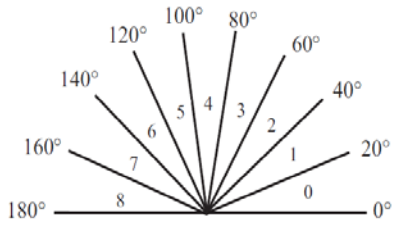


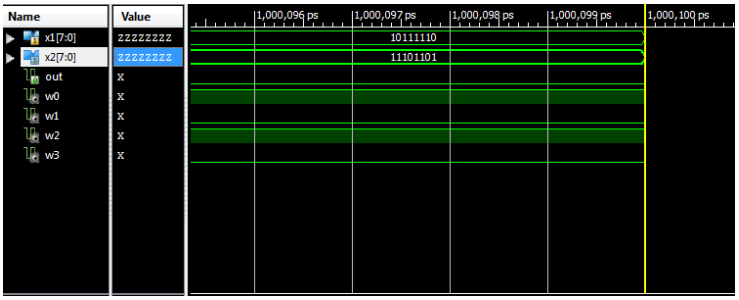
Fig.6. Gradient Orientation bins

3.EXPERIMENTAL RESULTS AND DISCUSSION

The architectural overview of the HOG algorithm. The proposed FPGA architecture supports multiple image sizes. Hardware architecture for HOG feature extraction. The input image is divided into blocks and is stored in the memory. The 3 ×3 mask each pixel of weight one is moved over the block, the corresponding pixels are buffered and processed. Gradient magnitude and orientation of each pixel is calculated in pipelined fashion. Current pixel and eight neighborsgradient magnitude and orientation is stored in the memory. Histogram is generated for each block based on gradient vote. Histogram of each block is normalized using block histogram normalization in pipelined fashion and then the features are stored in the memory. The processing unit is synthesis on Xilinx Spartan 3 FPGA software. The processing unit computes the HOG features of each block..The processing unit extracts HOG features in five stages. The modules in each stage are running in parallel as shown and each one is processing different block at a time.

3.1.Simulation for magnitude computation

The FPGA based simulation for magnitude computation is given in Fig 3.1



3.2.RTL schematic for magnitude computation

The FPGA based RTL Schematic for processing a blockmagnitude computation is given in Fig 3.2

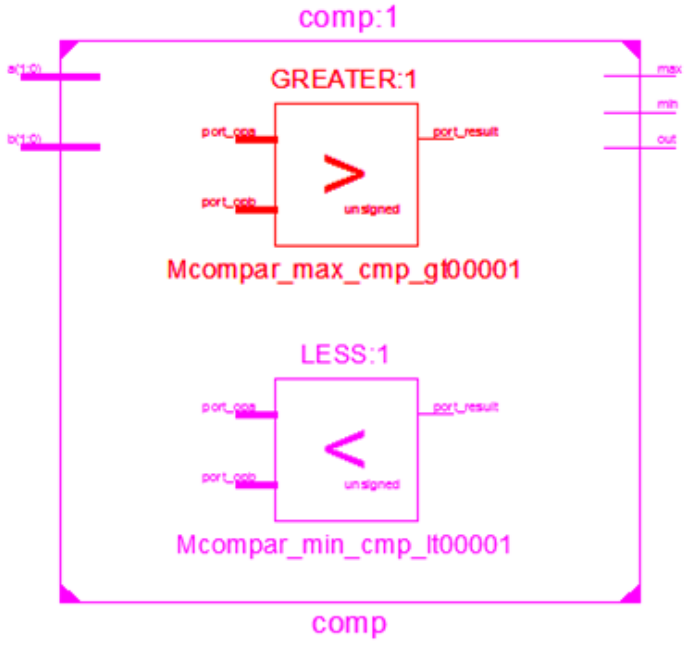


Fig 3.2RTL Schematic of Magnitude Computation

3.3.Resource utilization for magnitude computation

The FPGA resource utilization for processing a blockmagnitude computation is given in Table 3.1 .

Table 3.1.Resource Utilization

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	1	768	0%
Number of 4 input LUTs	2	1536	0%
Number of bonded IOBs	7	124	5%

Block size Slice registers Slice LUTs Total used memory for gradient and magnitude computation , Gradient vote computation 16 ×16 block, and memory used 1010000 kilobytes.

4.SUMMARY

In HOG, the implementation for gradient computation the memory used is 251716 kilobytes with clock rate of 5.01 nano sec, for magnitude computation the memory used is 251972 kilobytes with clock rate of 4.74 nano sec, for normalized computation the memory used is 253956 kilobytes with clock rate of 7.07 nano sec, for gradient vote the memory used is 252676 kilobytes with clock rate of 4.65 nano sec, for bin the memory used is 252356 kilobytes with clock rate of 4.76 nano sec. The synthesis result shows that our contain total memory used for HOG is 1010000 kilobytes and operates at a clock rate of 38.124MHZ.

5.CONCLUSION

The conventional HOG algorithm relies on the detection of human with fixed window size and has many complex operations in direct implementation. In order to increase the robustness and to meet real-time requirements, an efficient HOG algorithm is proposed that has an ability to detect human atlow-contrast. To support this multi-scale generation of an input image and gradient preserving techniques are introduced. Further, the low cost is achieved by replacing the complex operations with approximation methods.Thereby implementation of architecture for each block is analyzed on Xilinx Spartan 3. And the memory utilization is computed and their by enabling the design of ASIC is possible for real time application.

6.REFERENCES

[1] NazmaNausheen, Ayan Seal, Pritee Khanna, SantanuHalder,"A FPGA based implementation of Sobel edge detection", 0141-9331/@2017 Elsevier B.V.

[2] P. Chen , C. Huang , C. Lien , Y. Tsai , An efficient hardware implementation of HOG feature extraction for human detection, IEEE Trans. Intell. Transport. Syst. 15 (2) (2014) 656–662 .

[3] Sujith B Jyothiprakash ,"Pedestrain Detection-A Comparative Study Using HOG and COHOG" vol.2,special issue 5,October 2014.

[4] Xiangyu Zhang, Fengwei An, Ikki Nakashima, AiwenLuo,LeiChen,Idaku Ishii and Hans Jurgen Mattausch.,"A hardware-oriented histogram of oriented gradients algorithm and its VLSI implementation," in japanese journal of applied physics.

[5] Yanwei Pang, Senior Member, IEEE ,He Yan , Yuan , Senior Member , IEEE and Kongqiao Wang , "Robust Co-HOG feature extraction in human-centered image/video management system" IEEE transactions,vol-42,no 2,April 2012.

[6] Y. Yuan , X. Lu , X. Chen , Multi-spectral pedestrian detection, Sig. Process 110 (2015) 94–100 .

[7] J. Echanobe , I. del Campo , K. Basterretxea , M.V. Martinez , Faiyaz Doctor , An FP- GA-based multiprocessor-architecture for intelligent environments, Micropro- cess. Microsyst 38 (7) (2014) 730–740 .

[8] G.K. Gultekin , A. Saranli , An FPGA based high performance optical flow hard- ware design for computer vision applications, Microprocess. Microsyst 37 (3) (2013) 270–286.

[9] L. Araneda , M. Figueroa , A compact hardware architecture for digital image stabilization using integral projections, Microprocess. Microsyst. 39 (8) (2015) 987–997 .

[10] C. Papageorgiou , T. Poggio , A trainable system for object detection, Int. J. Com- put. Vis. 38 (1) (20 0 0) 15–33 .