Accurate and Low Complex Cell Histogram Generation By Bypass The Gradient of Pixel Computation

Huy-Hung Ho, Ngoc-Sinh Nguyen, Duy-Hieu Bui, Xuan-Tu Tran
SIS Laboratory, VNU University of Engineering and Technology (VNU-UET),
144 Xuan Thuy Road, Cau Giay, Hanoi, Vietnam. Email: {sinhnn_55,hieubd,tutx}@vnu.edu.vn

***Abstract*—Histogram of Oriented Gradient (HOG) is a popular feature description for the purpose of object detection. However, HOG algorithm requires a high performance system because of its complex operation set. ~~Especially~~In HOG algorithm, the cell histogram generation is one of the most complex part, it uses inverse tangent, square, square root, floating point multiplication. In this paper, we propose an accurate and low complex cell histogram generation by bypass the gradient of pixel computation. It employs the bin's boundary angle method to determine the two quantized angles. However, instead of choosing an approximate value of tan, the nearest greater and the nearest smaller of each *tan* values from ratio between pixel's derivative in *y* and *x* direction are used. ~~Magnitude~~ The magnitudes of two bins are solutions of a system of two equations, which ~~represent~~ represents the equality of the gradient of a pixel and its two bins in both vertical and horizontal direction. The proposed method spends only 30 addition and 40 shift operations to identify two bins of a pixel. Simulation results show that the percentage error when reconstructing the difference in *x* and *y* direction are always less than 2% with 8-bit length of the fractional part. Additionally, manipulating the precision of gradient magnitude is very simple by pre-defined *sine* and *cosine* values of quantized angles. ~~Synthesizing the hardware implementation presents that its area cost is~~ The synthesis results of a hardware implementation of the proposed method consume 3.57 ~~KGates with~~ KGEs in 45nm NanGate standard cell library. The hardware module ~~runs~~ operates at the maximum frequency of 400 MHz, and ~~the throughput is~~ its corresponding throughput is up to 0.4 (pixel/ns) for a single module. It is able to support ~~about~~ 48 fps with 4K UHD resolution.**

I. Introduction

The Histogram of Oriented Gradients (HOG) [1] is a feature descriptor used in computer vision and image processing for object detection. It gets the quantity of strong ~~characteristic~~ characteristics from the shape change of the object by dividing a local domain into plural blocks and making the incline of each the histogram. Practically, HOG feature achieves very high accuracy level, it is up to 96.6% of the detection rate with 20.7% of the false positive rate [2]. Hence, it is the key role of wide range application domains including robotic, security surveillance, automotive. However, the complexity level of HOG is the most difficult problem when using in embedded systems or implementing into hardware.

The HOG algorithm can be separated into 3 phases: cell histogram generation, block normalization, and the SVM classification. The first phase plays as the most complex one and consumes the most energy. Because ~~,~~ extracting the feature of each pixel requires series of complex operations: inverse tangent, square, square root, floating point multiplication. Additionally, in the age of cloud computing, the two last phases can be done in the server for IoT devices. Hence, simplifying cell histogram generation part is very necessary, especially for IoT devices.

Many researchers have proposed diverse approximate methods such as LUT or CORDIC or bin's boundary angles in cell histogram generation to reduce complexity ~~level~~levels. Even though approximate models reduce the accuracy of extracting pixel feature, they still require too ~~much~~ many memory areas or ~~use~~ many iterations.

In this work, we would like to propose a new ~~methodology~~method, which extracts features of a pixel accurately and is suitable for low resource systems and hardware implementation. The difference in this ~~methodology~~ method is that it is not an approximate model, and it computes the quantized orientations and ~~gradient magnitude of quantized organized~~ corresponding magnitude of each quantized orientation directly. The idea comes from the properties of decomposing the gradient of a pixel into two components (bins). The proposed ~~methodology~~ method includes only addition and shift operation. ~~As our simulation results, it requires~~ It spends about 30 additions and 40 shifts for 8-bit of fraction of magnitude. In ~~the accurate side~~terms of accuracy, error when reconstructing the pixel's derivatives in both direction from voted bins is always less than 2%. A hardware implementation consumes about 3.57 ~~KGates with 45nm NanGate~~ KGEs in NanGate 45nm standard cell library. The maximum operating frequency of the hardware is 400MHz, and its throughput is up to 0.4 pixel/ns.

The rests of the paper are organized as follows. Section II introduces the cell histogram generation and some previous optimization works. Section III shows the detail of proposed ~~methodology~~method. Section IV presents details of the hardware implementation of proposed ~~methodology~~ method and simulation results. Finally, section V gives a summary and our expected future.

II. Cell histogram generation and its approximations

A. Conventional cell histogram generation

Histogram of Oriented Gradient (HOG), pioneered by DALAL and TRIGGS [1], ~~become~~ becomes one of the most popular methods for feature extraction. In the conventional HOG, the cell histogram generation part is the most dominant power consumption, up to 58% HOG power in Takagi's work [3]. Authors in [4] analyzed the cell histogram generation part account for 91% workload in detection window-based approach.

Equation 1 - 7 present all equations using in the conventional HOG. As shown in these equations, the high complexity of the cell histogram generation comes from its complex operations: two squares, one square root, one inverse tangent, four floating point multiplications.

Pixel derivatives with respect to *x*,*y*:

$$d_x = I(x, y+1) - I(x, y-1) \tag{1}$$

$$d_y = I(x+1, y) - I(x-1, y) \tag{2}$$

Gradient magnitude calculation:

$$||\overrightarrow{M(x,y)}|| = \sqrt{d_x^2 + d_y^2} \tag{3}$$

Gradient orientation/angle calculation:

$$\alpha = \arctan\left(\frac{d_y}{d_x}\right) \tag{4}$$

Identifying two voted bins:

$$\theta_i \quad < \alpha < \theta_{i+1} \tag{5}$$

$$||\overrightarrow{B_{\theta_i}}|| = ||\overrightarrow{M(x,y)}|| \times \frac{\alpha - \theta_i}{w} \tag{6}$$

$$||\overrightarrow{B_{\theta_{i+1}}}|| = ||\overrightarrow{M(x,y)}|| \times \frac{\theta_{j+1} - \alpha}{w} \tag{7}$$

where

- $||\overrightarrow{M(x,y)}||$ is the gradient magnitude of pixel $I(x,y)$;
- α is the gradient orientation of pixel $I(x,y)$;
- θ_j, θ_{j+1} are two quantized orientations;
- $||\overrightarrow{B_{\theta_i}}||, ||\overrightarrow{B_{\theta_{i+1}}}||$ are the magnitude of two quantized orientations: θ_i, θ_{j+1};
- w is the angle distance between two continuous orientations, $w = 20$.

In order to reduce the complexity of cell histogram generation and to be able for hardware implementation, several approximate algorithms have been applied. All of approximate models have been employed to avoid ~~all~~ non-linear operations in gradient magnitude and gradient orientation computation.

B. Gradient magnitude approximations

Computing gradient magnitude by L2-norm (Equation 3) spends two squares and one square root, which are ~~very complex to implement into hardware robustly.~~ inefficient to be implemented into hardware To solve this problem, the ~~most simplest methodology~~ simplest method is using LUT as used in Kadota's work[5]. Because the LUT method requires very high areas cost, so approximate computations have been applied. ~~Papers [6] and [7] have~~ Iandola [6] and Suleiman [7] employed L1-norm as Equation 8 instead of L2-norm. ~~Other papers [8] [9] have~~ Hsiao and Chen [8] [9] used an approximate model, called square root approximation technique (SRA) as Equation 9. ~~A patent [10] have~~ Munteanu [10] implemented Equation 10. However, approximate models reduce the accuracy level.

$$||\overrightarrow{M(x,y)}|| \approx |d_x| + |d_y| \tag{8}$$

$$||\overrightarrow{M(x,y)}|| \approx max((0.875 \times g_{max} + 0.5 g_{min}), g_{max}) \tag{9}$$

$$||\overrightarrow{M(x,y)}|| \approx |d_x| + |d_y| - \frac{g_{min}}{2} \tag{10}$$

where g_{max} and g_{min} is the maximum and minimum of $|d_x|, |d_y|$, respectively.

C. Gradient orientation approximations

Inverse tangent operation is the most ~~challenge~~ challenging operation for hardware implementation. There are several approaches in practice to implement this non-linear operation. ~~Paper [6] has~~ Iandola [6] used full LUT for inverse tangent, and consumed 256KB. ~~Paper [11] has~~ Peker [11] combined the LUT and piece-wise approximations to obtain higher accuracy with less demand on the resources. ~~Paper [3] has~~ Takaghi [3] applied by CORDIC to avoid high memory resources. However, CORDIC module is usually implemented with many iterations to obtain acceptable calculation precision.

D. Determining two quantized orientation without gradient orientation

A more effective approach is bin's boundary angles. In this method, the actual angle of the gradient does not need to be calculated. It uses Equation 11 to ~~determines~~ determine the two quantized gradient angles. In order to simplify the implementation, Equation 11 has been ~~convert to some~~ converted to other formulas. ~~Paper [9] has used~~ Chen [9] proposed Equation 12 with fixed $\tan\theta$. In contrast, publications [2] [3] [7] [8] [12] [13] ~~have~~ multiplied $\tan\theta$ with a constant A, then rounding $A\tan\theta$ value as Equation 13.

$$\tan\theta_1 < \frac{d_y}{d_x} < \tan\theta_2 \tag{11}$$

$$\begin{cases} \tan\theta_1 \times d_x < d_y < \tan\theta_2 \times d_x \\ \tan\theta = \sum_{i=-m}^{n} 2^i, m, n \in N \end{cases} \tag{12}$$

$$\lfloor A \times \tan\theta_1 \rceil \times d_x < A \times d_y < \lfloor A \times \tan\theta_2 \rceil \times d_x \tag{13}$$

Converting Equation 11 into Equation 12 and 13 has caution of choosing fixed $\tan\theta$ or the A values. It affects to determine the two quantized orientation directly. As shown, Equation 12 cannot present a rational number, whose divisor is a prime number. Let take an example~~to see limitation of Equation 13~~, a $\frac{d_y}{d_x} \equiv \frac{56}{97} = 0.57732 < \tan 30$ expects to contribute its gradient to two angles: 10 and 30 degree, but the $\frac{97}{168} = 0.577381$ is

in the angle range (30:50) degree. In this case, the A value have to be equal or greater than 10^5 to distinct the those two values. However, increasing the A values means increasing the complexity of implementation.

Another disadvantage of this method is that it does not have enough information for voting magnitude as Equation 6 and 7. In this moment, those mentioned papers using this ~~methodology~~ method have employed fixed weight for voting magnitude. Consequently, ~~the accurate level of cell histogram generation is reduced~~all gradients in an orientation range which have the same magnitude, fixed weight method produces homogeneous couple of bins.

III. Proposed Cell Histogram Generation

In this work, we proposed a robust and low complex methodology to determine the two angle and calculate the magnitude of the two voted bins. This methodology has no calculation of the gradient of pixel. To determine the two quantized orientations exactly, a trusted form of Equation 11 was discovered. It is based on the limitation of the $\frac{d_y}{d_x}$ and the $\tan\theta$ values. Gradient magnitude of two bins are the solutions of a system of two equations instead of voting gradient.

A. Robust quantized angles determination

Equation 14 shows our basic ideas. In stead of choosing fixed bit length of fractional part of $\tan\theta$ or representing $\tan\theta$ by structure $\sum 2^i$, our proposed method bounds each $\tan\theta$ by two rational numbers: its nearest smaller $\frac{d_y}{d_x}$, called $\frac{A_s}{B_s}$ and its nearest greater $\frac{d_y}{d_x}$, called $\frac{A_g}{B_g}$. Those fractions did not choose from the *tan* only, it was from examining all cases of $\frac{d_y}{d_x}$ and *tan* values to reach the expected results.

$$\frac{A_s}{B_s} < \tan\theta < \frac{A_g}{B_g} \tag{14}$$

Table I shows all $\frac{A_s}{B_s}$ and $\frac{A_g}{B_g}$ in a case of 8-bit length pixel. ~~As shown, with~~ With those rational numbers ~~, we do not need to multiply~~ in the table, multiplying with very large number such as 10^5 is not necessary to determine quantized orientation exactly.

TABLE I
$\tan\theta$ AND ITS NEAREST VALUES $\frac{d_y}{d_x}$

Smaller and Nearest $\lvert\frac{d_y}{d_x}\rvert$	$\tan\theta$	**Greater and Nearest** $\lvert\frac{d_y}{d_x}\rvert$
$\frac{43}{244}$	tan(10)	$\frac{3}{17}$
$\frac{56}{57}$	tan(30)	$\frac{97}{168}$
$\frac{230}{193}$	tan(50)	$\frac{87}{73}$
$\frac{250}{91}$	tan(70)	$\frac{11}{4}$

Equation 14 leads to equivalent ~~equations~~ Equation Equations 15 - 16 for determining two quantized orientations: θ_i and θ_{i+1}, respectively. As shown the two equations, there are only multiplication and comparison operations at this moment, and it is much easier than inverse tangent for hardware implementation.

$$\alpha > \theta \Leftrightarrow \begin{cases} |\frac{d_y}{d_x}| > \frac{A_s}{B_s} \\ |\frac{d_y}{d_x}| \geq \frac{A_g}{B_g} \end{cases} \Leftrightarrow \begin{cases} B_s \times |d_y| > A_s \times |d_x| \\ B_g \times |d_y| \geq A_g \times |d_x| \end{cases} \tag{15}$$

$$\alpha < \theta \Leftrightarrow \begin{cases} |\frac{d_y}{d_x}| \leq \frac{A_s}{B_s} \\ |\frac{d_y}{d_x}| < \frac{A_g}{B_g} \end{cases} \Leftrightarrow \begin{cases} B_s \times |d_y| \leq A_s \times |d_x| \\ B_g \times |d_y| < A_g \times |d_x| \end{cases} \tag{16}$$

where

- $\frac{A_s}{B_s}$ is a $\frac{d_y}{d_x}$ number, and it is the nearest smaller or equal number of $\tan\theta$ value;
- $\frac{A_g}{B_g}$ is a $\frac{d_y}{d_x}$ number, and it is the nearest greater or equal number of $\tan\theta$ value;
- A_s, B_s, A_g, B_g is integer number in $|d_x|, |d_y|$ range.
- θ is quantized orientation;
- α is the gradient orientation of the current pixel.

B. Magnitude of two bins calculation

As discussion in section II-D results of bin's boundary angles do not provide enough arguments for voting. Our work proposed a system of two equations from information: $\theta_i, \theta_{i+1}, d_x$ and d_y to calculate gradient magnitude of two bins exactly.

Back to the ideas of HOG, it decomposes gradient of a pixel $I(x,y)$ into a combination of two bins as described in Figure 1 and Equation 17. Hence, in each direction, the sum of two bins ~~has to equal~~ equals to the gradient of the pixel. Equation 18 and 19 present those equalities in the *Ox* and *Oy* axises, respectively. Finally, the two magnitudes are the solutions of the two equations, and have the form as Equation 20 and Equation 21, respectively.

$$\overrightarrow{M(x,y)} = \overrightarrow{B_{\theta_i}} + \overrightarrow{B_{\theta_{i+1}}} \tag{17}$$

$$||\overrightarrow{M(x,y)}|| \times \cos\alpha = d_x = ||\overrightarrow{B_{\theta_i}}|| \cos\theta_i + ||\overrightarrow{B_{\theta_{i+1}}}|| \cos\theta_{i+1} \tag{18}$$

$$||\overrightarrow{M(x,y)}|| \times \sin\alpha = d_y = ||\overrightarrow{B_{\theta_i}}|| \sin\theta_i + ||\overrightarrow{B_{\theta_{i+1}}}|| \sin\theta_{i+1} \tag{19}$$

$$||\overrightarrow{B_{\theta_i}}|| = \frac{sin(\theta_{i+1})d_x - cos(\theta_{i+1})d_y}{sin(20)} \tag{20}$$

$$||\overrightarrow{B_{\theta_{i+1}}}|| = \frac{cos(\theta_i)d_y - sin(\theta_i)d_x}{sin(20)} \tag{21}$$

The ~~competitive~~ advantages of this ~~methodology, it allows~~ method is computing gradient magnitude with only d_x, d_y information and the two quantized angles. The most complex part are $\sin, \cos$ values, but they are pre-computed. By defining those numbers, the accurate level is manipulated ~~completely~~by defining those *sin* and *cos* values. To achieve these ~~good~~advantages, this method requires determining the

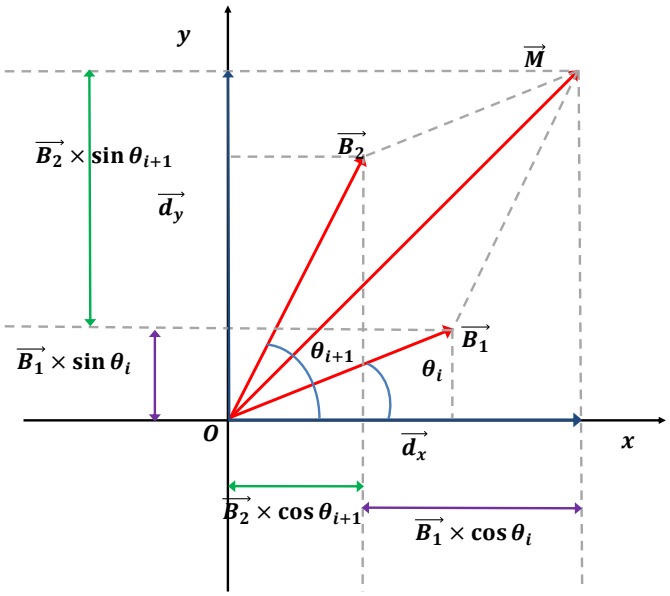

Fig. 1. Decomposition a vector into form of two vectors

quantized orientations exactly. If there is any errors in determining direction of two bins, it produces negative magnitudes.

C. Proposed cell histogram generation scheme

Figure 2 shows the data flow of proposed cell histogram generation, which ~~employed our methodology. As shown, the~~ was employed our method. The input contains 4 neighbor pixels $I(x+1)(y), I(x-1,y), I(x,y+1), I(x,y-1)$ of pixel $I(x,y)$. ~~The~~ In the first step, the gradient in x and y directions d_x and d_y are computed. ~~At~~ In the second step, we have absolute values of d_x and d_y, and the quadrant position of the gradient of pixel I. If the sign of d_x is opposite sign of d_y, the gradient $\overrightarrow{M(x,y)}$ is in the second quadrant. Otherwise, it is in the first quadrant. Because the left side of Oy axis is a reflection of the right side of Oy, so computing the magnitude of two voted bins in the first quadrant is enough. The third step is to determine the two quantized angles θ_i, θ_{i+1} via comparison between $A \times d_x$ and $B \times d_y$, where A is the dividend, and B is the divisor in Table I. In this scheme, we use the nearest greater values $\frac{d_y}{d_x}$ of $\tan\theta$. ~~Depend~~ Depending on situation, ~~you are able to choose~~ it is able to use the left side of the Table I. After ~~having two quantized angles~~the orientation of two bin are determined, the two magnitudes $||\overrightarrow{B_{\theta_i}}||, ||\overrightarrow{B_{\theta_{i+1}}}||$ of two bins are from Equations 20 and 21. Finally, we have to re-correct the angles from quadrant position information. If those angles are in the second quadrant, converting ~~the~~ θ in the first quadrant into the second quadrant is done by a subtraction $\theta = 180 - \theta$.

As shown in Figure 2, ~~all of the multiplication operations,~~ we have already known one of two numbers in multiplication operations. It allows converting all multiplications into form of shifts and additions. Table II shows all of the transformations with 8-bit length of fractional part in *sine* and *cosine*. It is equivalent to multiply by 256. The size of fractional part

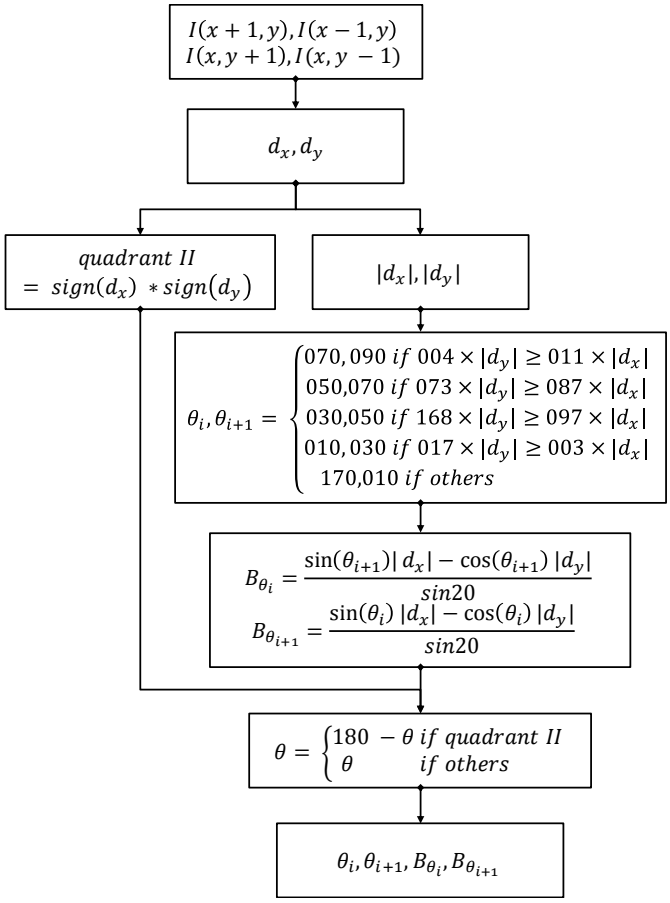

Fig. 2. Data flow of proposed methodology

of magnitude is controlled by the *sine*, *cosine* values in the Table II. For example, if an application needs 10-bit length of fractional part, the *cosine*, *sine* values have to multiply 1024 instead of 256. From Table II and Figure 2, the total number of operations to identify two voted bins are about 30 additions and nearly 40 shifts.

IV. Hardware Implementation and Experimental Results

We implemented the proposed ~~methodology~~ method into hardware and examined its results. Figure 3 shows the hardware module of the proposed ~~methodology~~method. It includes 4 input pixels $I(x+1,y), I(x-1,y), I(x,y+1), I(x,y-1)$ with 8-bit pixel length. Output contains the magnitude and direction of two bins. Angle is 4-bit length data, and magnitude is 17-bit length data with 8-bit of fractional part.

As shown in Figure 3, the module includes 4 ~~states~~steps. Firstly, the difference d_x and d_y are computed from the 4 input pixels. In the second ~~state,~~ step the absolute of d_x, d_y are computed, and the quadrant position of the gradient of pixel $I(x,y)$ is detected. After having those absolute values, the pre-calculating ~~state calculates~~ step alculates all the multiplications in Table II. The fourth ~~state~~ step produces two voted bins, include orientation and magnitude.

TABLE II
CONVERTING ALL MULTIPLY TO SHIFT AND ADDITION OPERATIONS WITH 8-BIT OF FRACTIONAL PART

$B \times d_y$	**Shift and addition**	$A \times d_x$	**Shift and addition**
4	2^2	11	$2^3 + 2^1 + 2^0$
73	$2^6 + 2^3 + 2^0$	87	$2^6 + 2^4 + 2^3 - 2^0$
168	$2^7 + 2^5 + 2^3$	97	$2^6 + 2^5 + 2^0$
17	$2^4 + 2^0$	3	$2^1 + 2^0$
$cos(10) \times 256$	$2^8 - 2^2$	$sin(10) \times 256$	$2^5 + 2^3 + 2^2$
$cos(30) \times 256$	$2^8 - 2^5 - 2^1$	$sin(30) \times 256$	2^7
$cos(50) \times 256$	$2^7 + 2^5 + 2^2 + 2^0$	$sin(50) \times 256$	$2^7 + 2^6 + 2^2$
$cos(70) \times 256$	$2^6 + 2^4 + 2^3$	$sin(70) \times 256$	$2^8 - 2^4 + 2^0$
$cos(90) \times 256$	0	$sin(90) \times 256$	2^8
$\frac{1}{sin(20)}$	$2 + \frac{15}{16}$		

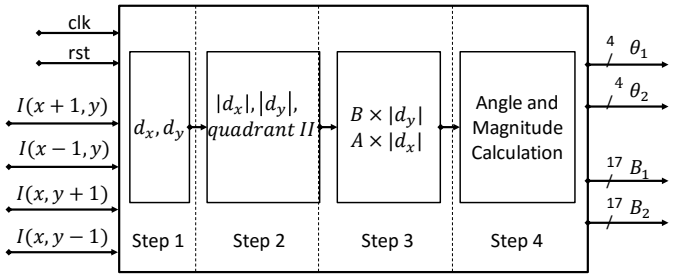

Fig. 3. Hardware implementation of proposed methodology

Figure 4 presents the detail of the fourth ~~state~~steps. It has two multiplexers. The first multiplexer has 4 input from 4 angle comparisons. This multiplexer chooses the highest angle θ in the first quadrant. For each quantized angle, it also produces corresponding gradient magnitude from Equation 20 and 21. The second multiplexer converts the θ in first quadrant into the second quadrant if the $sign(d_x)$ is opposite $sign(d_y)$.

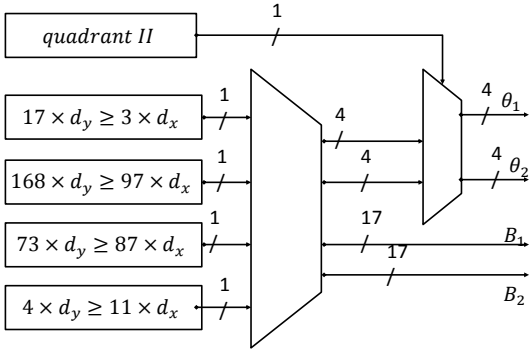

Fig. 4. Submodule angle and magnitude calculation

Table III shows the hardware implementation results of some state-of-the-art publications and our proposed method. Pei-Yin and Hsiao's works calculated the gradient of current pixels with approximate model via the SRA approximation for magnitude and CORDIC for angle. Munteanu combined Equation 10 and bin's boundary angles method. As shown, the proposed methodology and Munteanu's works can run at ~~highest frequency~~ the maximum operating frequency of 400Mhz. They are also the two smallest areas cost of implementation. However, the area of Munteanu's module still more than fifth times as much as our proposed method.

TABLE III
HARDWARE COMPARISON OF CELL HISTOGRAM GENERATION

Publications	**Technology**	**Areas (KGates)**	**Frequency**
Pei-Yin [9]	130nm	85.5	167Mhz (all HOG)
Hsiao [8]	90nm	29.1	N/A
Munteanu [10]	N/A	20	Max. 400Mhz
Our Proposed	45nm NanGate	3.57	Max. 400Mhz

Figure 5 shows our error of calculation model for the proposed ~~methodology~~method. From the voted bin, we calculate a model of d_x and d_y by Equation 18 and 19, called d'_x and d'_y. With the d'_x, d'_y, d_x, d_y, the percentage error of calculation at both Ox and Oy axises are computed. Tested with all cases, the results prove that our proposed methodology provides the percentage error of calculations e_x e_y, which are always less than 2% with 8-bit length of fractional part.

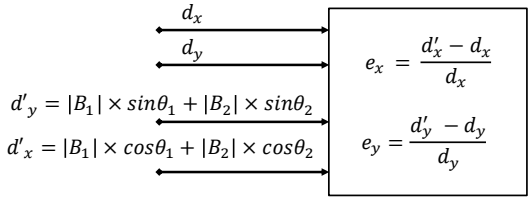

Fig. 5. Error of calculation model

V. CONCLUSION

In this paper, we have proposed a new ~~methodology~~ method to do cell histogram generation part without calculating the actual gradient of pixel. It identifies the two bins by the properties of decomposing ~~gradient of~~ the gradient of a pixel into two standard vectors. This ~~methodology extracts HOG feature of~~ method extracts HOG features of a pixel with 30 additions and 40 shifts. The error of the reconstructed values of d_x and d_y from the two voted bins are always less than 2%. Additionally, it is very flexible, we only need to define new constants for *sine* and *cosine* values of quantized angles for changing the precision. Implementing the proposed methodology into hardware takes about 3.57 ~~KGates with 45nm NanGate~~ KGEs in NanGate 45nm standard cell library The hardware module ~~runs at maximum frequency~~ operates

at the maximum frequency of 400 MHz, and ~~the throughput reaches~~ it is equivalent to the throughput of 0.4 ~~(~~pixel/ns ~~)~~ for a single module.

REFERENCES

[1] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893. [Online]. Available: http://ieeexplore.ieee.org/document/1467360/

[2] K. Negi, K. Dohi, Y. Shibata, and K. Oguri, "Deep pipelined one-chip FPGA implementation of a real-time image-based human detection algorithm," in *2011 International Conference on Field-Programmable Technology*, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/document/6132679/

[3] K. Takagi, K. Mizuno, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A sub-100-milliwatt dual-core HOG accelerator VLSI for real-time multiple object detection," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2533–2537. [Online]. Available: http://ieeexplore.ieee.org/document/6638112/

[4] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "Architectural Study of HOG Feature Extraction Processor for Real-Time Object Detection," in *2012 IEEE Workshop on Signal Processing Systems*. [Online]. Available: https://pdfs.semanticscholar.org/f2c3/9ff8ee36aca20915f6ce7195317b4e1a34cd.pdf

[5] R. Kadota, H. Sugano, M. Hiromoto, H. Ochi, R. Miyamoto, and Y. Nakamura, "Hardware Architecture for HOG Feature Extraction." IEEE, pp. 1330–1333. [Online]. Available: http://ieeexplore.ieee.org/document/5337209/

[6] F. N. Iandola, M. W. Moskewicz, and K. Keutzer, "libHOG: Energy-Efficient Histogram of Oriented Gradient Computation," in *IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 1248–1254. [Online]. Available: http://ieeexplore.ieee.org/document/7313297/

[7] A. Suleiman and V. Sze, "An Energy-Efficient Hardware Implementation of HOG-Based Object Detection at 1080HD 60 fps with Multi-Scale Support," vol. 84, no. 3, pp. 325–337. [Online]. Available: http://link.springer.com/10.1007/s11265-015-1080-7

[8] S.-F. Hsiao, J.-M. Chan, and C.-H. Wang, "Hardware design of histograms of oriented gradients based on local binary pattern and binarization," in *2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. IEEE, pp. 433–435. [Online]. Available: http://ieeexplore.ieee.org/document/7803995/

[9] P.-Y. Chen, Chien-Chuan Huang, Chih-Yuan Lien, and Yu-Hsien Tsai, "An Efficient Hardware Implementation of HOG Feature Extraction for Human Detection," vol. 15, no. 2, pp. 656–662. [Online]. Available: http://ieeexplore.ieee.org/document/6648678/

[10] "A Method for producing a histogram of oriented gradients," Patent Patent No. WO2 016 083 002. [Online]. Available: https://patentscope.wipo.int/search/docservicepdf_pct/id00000033608412/PAMPH/WO2016083002.pdf

[11] M. Peker, H. Altun, and F. Karakaya, "Hardware emulation of HOG and AMDF based scale and rotation invariant robust shape detection," in *2012 International Conference on Engineering and Technology (ICET)*, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/6396145/

[12] S. Bauer, S. Kohler, K. Doll, and U. Brunsmann, "FPGA-GPU architecture for kernel SVM pedestrian detection," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*. IEEE, pp. 61–68. [Online]. Available: http://ieeexplore.ieee.org/document/5543772/

[13] T. P. Cao and G. Deng, "Real-Time Vision-Based Stop Sign Detection System on FPGA," in *2008 Digital Image Computing: Techniques and Applications*. IEEE, pp. 465–471. [Online]. Available: http://ieeexplore.ieee.org/document/4700058/