Cell histogram generation with only shift and addition operators

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: {H3email?,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 performance system because of the complex operations set. Especially, the cell histogram generation in HOG (feature extraction) is one the most complex part. IT requires one inverse tangent In this paper, we propose a methodology to improve the cell histogram generation part by using only shift and addition operations. For each pixel, it spends about 30 addition and 40 shift operations. Simulation results show that the percentage errors of calculations are always less than 2% with 8-bits length of fractional part. Synthesize the hardware implementation presents that the area cost is 3.5 KGates with 45nm NanGate standard cell library. The hardware module runs at maximum frequency 400 MHz, and the throughput is about 0.4 (pixel/ns) for single module. It is able to support about 48 frames/s 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 objection detection. It gets quantity of strong characteristic 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 key role of wide range application domains including robotic, security surveillance, etc. However, the complexity level of HOG is the most difficult problem when implementing it into embedded systems and battery systems.

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 feature of each pixel requires series of high complex operation: $\arctan$, square root, float multiply. Moreover, in the age of cloud computing, the two last phases can be done in the server for IoT devices. It means that the local devices only need to do cell histogram generation part, then transferring all voted bins to their servers. It allows extending the life of battery systems,

In this work, we study about the cell histogram generation and would like to optimize this phase to be able to run in low resource embedded systems. Our methodology is transforming all of the complex operations $\arctan$, square root and float multiply into series of shift and addition operations. As our experiments, voting gradient of a pixel into the two standard bins requires about 30 addition operations and 40 shift operations. About the accuracy of computing, the percentage errors are only about 1% of d_x d_y. Hardware module of proposed methodology consumes about 3.57 KGates with 45nm NanGate standard cell library. Maximum 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. Section IV presents details of hardware implementation of proposed methodology and simulation results. Finally, section V gives summary and our expected future.

II. Conventional non-normalize feature extraction in HOG

Histogram of Oriented Gradient (HOG), pioneered by DALAL and TRIGGS [1], become 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 Figure 1 [3]. Authors in [4] analyzed the cell histogram generation part account for 91% workload in detection window-based approach.

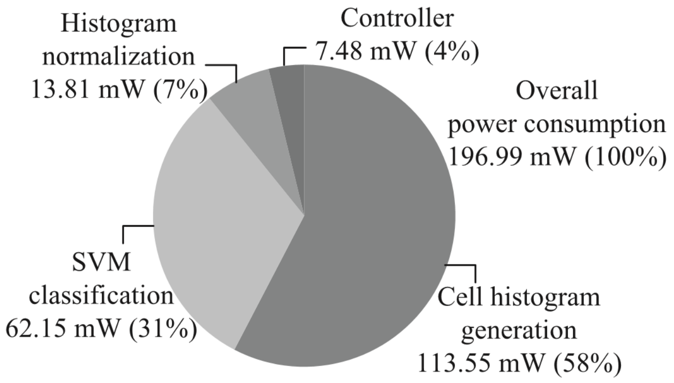


Fig. 1: Power consumption in FPGA [3]

$$d_x = I(x, y+1) - I(x, y-1) \quad (1)$$
$$d_y = I(x+1, y) - I(x-1, y) \quad (2)$$
$$M(x,y) = \sqrt{d_x^2 + d_y^2} \quad (3)$$
$$\theta = \arctan\left(\frac{d_y}{d_x}\right) \quad (4)$$
$$M_j = M(x,y)\frac{\theta - c_j}{w} \quad (5)$$
$$M_{j+1} = M(x,y)\frac{c_{j+1} - \theta}{w} \quad (6)$$

Where:

- $M(x,y)$ is the magnitude of the gradient of pixel $I(x,y)$
- θ is the angle of the gradient of pixel $I(x,y)$
- c_j, c_{j+1} are two voted standard bins.
- M_j, M_{j+1} is is the magnitude of two voted standard bins.
- w is the distance between two continuous bins, $w = 20$

The high complexity and power consumption of the cell histogram generation come from the complex operations set: $\arctan$, square root, float multiply, and float division. As shown, the Equation 3 - 4 use $\arctan$ and square root for computing the real gradient(angle and magnitude) of pixel $I(x,y)$. Additionally, voting the gradient of pixel $I(x,y)$ into the two continuous standard angle (bin) requires series of float multiplies and float divisions as Equation 5 and 6

In order to reduce the complexity of cell histogram generation, some publications try to avoid the nonlinear arctangent operation in the Equation 4. Several approximation methods are proposed such as piece-wise [5], look up tables (LUT) [6] [7], and using CORDIC [8] [3] [4] . They save up the power consumption and reduce the complexity of via pre-computing $\arctan$ values. However, pre-computing $\arctan$ values requires large memory area, and only solves the $\arctan$ problem. The other operations: square root in computing magnitude of pixel $I(x,y)$, and float division and float multiply in voting magnitude are still bottleneck at both performance and power consumption sides.

A more effective approach in [2] [9] has been proposed, which try to detect the two voted angles directly instead of computing the angle of pixel $I(x,y)$ firstly. This methodology uses the fixed-point arithmetic with a 10-bit fractional part for angle calculation. This arithmetic system is equivalent to integer arithmetic in which all the values are left-shifted by 10-bits as the Equation 8. However, multiplying 1024 is not cover all cases of comparison between $\frac{d_y}{d_x}$ and $\tan\theta$. Because, both the $\frac{d_y}{d_x}$ and $\tan\theta$ are able to be infinite fraction. Let take an example of popular bins [10..30..170](which are not same as the case in [2] [9]), $\frac{56}{97} = 0.57732$ expects to contribute its gradient to two bins: 10 and 30 degree, but the $\frac{97}{168} = 0.577381$ should be two bins at angle 30 and 50 degree. In another hand, multiplying 1024 is still larger than the FIXME, the expected values. The same as the LUT solution, calculating magnitude to two voted bins in this methodology is still complex.

TABLE I: $\tan\theta$ and its nearest values $\frac{d_y}{d_x}$

Smaller and Nearest $\frac{d_y}{d_x}$	$\tan\theta$	Greater and Nearest $\frac{d_y}{d_x}$
$\frac{43}{244}$	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)	$\frac{11}{4}$

$$\tan\theta_1 < \frac{d_y}{d_x} < \tan\theta_2 \quad (7)$$
$$1024\tan\theta_1 \times d_x < 1024 d_y < 1024\tan\theta_2 \times d_x \quad (8)$$

In the paper [10] [11] , calculating magnitude of the two voted angles is simpled by computing an approximate model of magnitude of the pixel $I(x,y)$ as Equation 9 , and used fixed weight for voting magnitude as Equation 10 - 11. However, following those approximate calculations can be causes of wrong bin. Additionally, computing the magnitude of pixel $I(x,y)$ is may not necessary.

$$\begin{aligned} M(x,y) &= \sqrt{d_x^2 + d_y^2} \\ &\approx max((0.875a + 0.5b), a) \quad (9) \\ B_1 &= M(x,y) >> 1 \quad (10) \\ B_2 &= M(x_y) >> 1 \quad (11) \end{aligned}$$

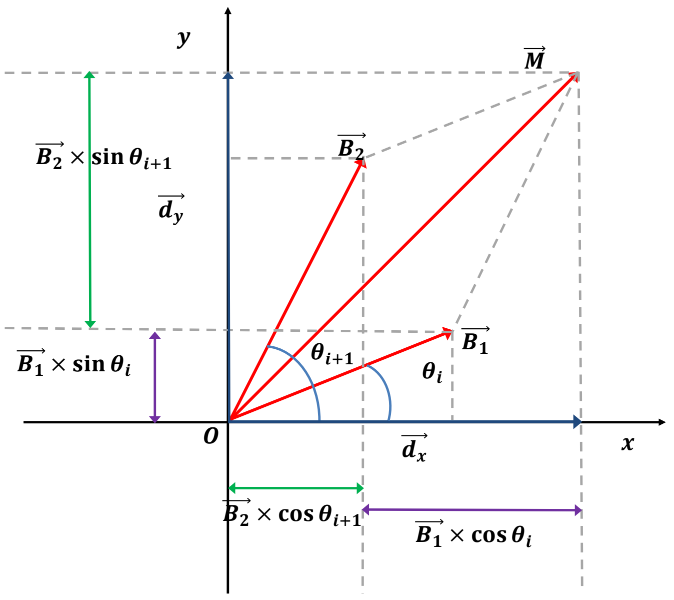
where $a = max(d_x, d_y)$ and $b = min(d_x, d_y)$

III. Proposed cell histogram generation

In this work, we propose a robust and efficient methodology of finding the two voted angles (bins) and calculating the magnitude of the two voted bins, which only uses a set of very simple operations: shift and addition.

As the Equation 1, 2 in the Section II, the d_x, d_y range are finite and depend on the range of pixel value. With the 8-bit length pixel, the d_x, d_y are integers in the range [-255, 255]. Consequently, the $\frac{d_y}{d_x}$ are float numbers in the set [-255, 255]. By examining all the cases of $\frac{d_y}{d_x}$ and the $\tan\theta$ values with θ in the first quadrant, we archive Table I. This table shows the $\tan\theta$ values, its smaller and nearest $\frac{d_y}{d_x}$, and its greater and nearest $\frac{d_y}{d_x}$. Those numbers will play as the critical thresholds to determine the two voted bins of a pixel $I(x,y)$. In fact, all the divisor in $\frac{d_y}{d_x}$ are smaller than the 1024 in the Equation 8.

Back to the ideas of HOG, it tries to represent the real gradient of pixel $I(x,y)$ into form of two standard bins as Figure 2 and Equation 12. The two Equations 13 and 14 are the consequent equations of the Equation 12 in the Ox and Oy axis, respectively. If we know about the d_x, d_y, the θ_i and the θ_{i+1}, the magnitude of two voted bins will be the roots of system of two Equations 13-14.



y
$\overrightarrow{M}$
$\overrightarrow{B_2} \times \sin\theta_{i+1}$
$\overrightarrow{d_y}$
$\overrightarrow{B_2}$
$\overrightarrow{B_1}$
θ_{i+1}
θ_i
$\overrightarrow{B_1} \times \sin\theta_i$
O
$\overrightarrow{d_x}$
x
$\overrightarrow{B_2} \times \cos\theta_{i+1}$
$\overrightarrow{B_1} \times \cos\theta_i$

Fig. 2: Convert a vector into form of two vectors

$$\overrightarrow{M} = \overrightarrow{B_{\theta_i}} + \overrightarrow{B_{\theta_{i+1}}} \tag{12}$$

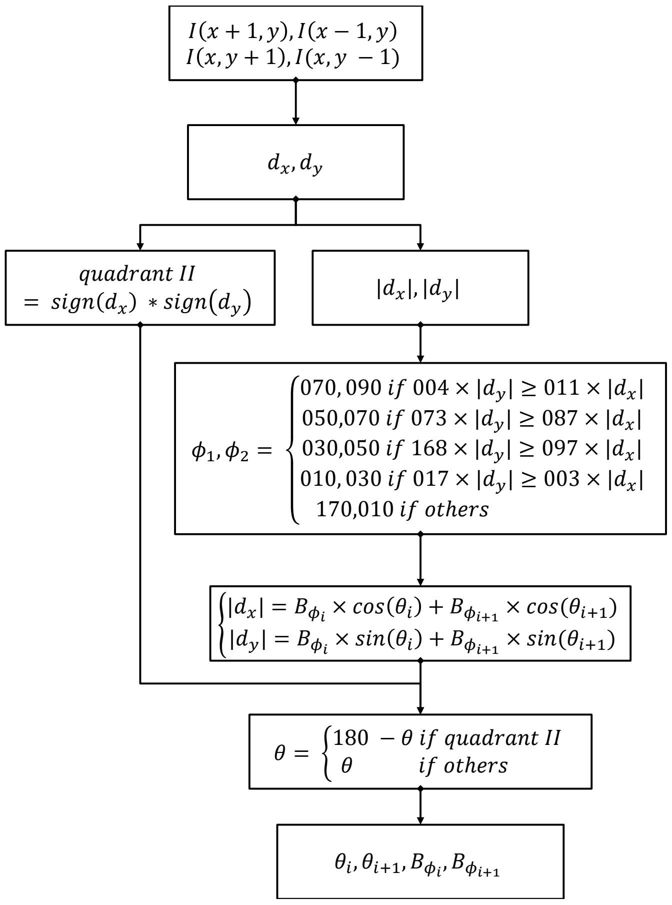
$$\begin{aligned}\|\overrightarrow{M}\|\cos(\alpha) &= d_x \\ &= \|\overrightarrow{B_{\theta_i}}\|\cos(\theta_i) + \|\overrightarrow{B_{\theta_{i+1}}}\|\cos(\theta_{i+1})\end{aligned} \tag{13}$$

$$\begin{aligned}\|\overrightarrow{M}\|\sin(\alpha)| &= d_y \\ &= \|\overrightarrow{B_{\theta_i}}\|\sin(\theta_i) + \|\overrightarrow{B_{\theta_{i+1}}}\|\sin(\theta_{i+1})\end{aligned} \tag{14}$$

$$\|\overrightarrow{B_{\theta_i}}\| = \frac{sin(\theta_{i+1})d_x - cos(\theta_{i+1})d_y}{sin(20)} \tag{15}$$

$$\|\overrightarrow{B_{\theta_{i+1}}}\| = \frac{cos(\theta)d_y - sin(\theta)d_x}{sin(20)} \tag{16}$$

From the experiments, which are shown in the two previous paragraphs, we propose a methodology of calculating cell histogram generation with only shift and addition operations. Figure 3 shows the details of the data flow of the proposed methodology. As shown, 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 first step, the d_x and d_y are computed. At the second step, we have absolute values of d_x and d_y, and the position of the gradient of pixel I. If the sign of d_x is opposite sign of d_y, the gradient of $I(x,y)$ is in the second quadrant. Otherwise, the gradient is in the first quadrant. Because the left side of Oy axis is a mirror of the right side of Oy, we only need to compute the magnitude of twos voted bin in the first quadrant. Next, finding the two voted angles of the pixel $I(x,y)$ is via comparison between $A \times d_x$ and $B \times d_y$, where A is the dividend , and B is the divisor in the in the Table I. In this data flow, we uses only the greater and nearest value $\frac{d_y}{d_x}$ of $\tan\theta$. Depend on application, you are able to chose the left side of the Table I. After have two voted angles, the magnitude of both angle are the roots of system of two Equations 13 - 14. Finally, the $sign(d_x)$ and $sign(d_y)$ are used to determine the angle of the current pixel and its twos voted angles, which are in the first quadrant or the second quadrant? If those angles are in the second quadrant, converting the θ in the first quadrant into the second quadrant is done by a subtraction $\theta = 180 - \theta$.



$I(x+1,y), I(x-1,y)$
$I(x,y+1), I(x,y-1)$
d_x, d_y
$quadrant\ II = sign(d_x) * sign(d_y)$
$|d_x|, |d_y|$
$\phi_1,\phi_2 = \begin{cases} 070,090\ if\ 004 \times |d_y| \geq 011 \times |d_x| \\ 050,070\ if\ 073 \times |d_y| \geq 087 \times |d_x| \\ 030,050\ if\ 168 \times |d_y| \geq 097 \times |d_x| \\ 010,030\ if\ 017 \times |d_y| \geq 003 \times |d_x| \\ 170,010\ if\ others \end{cases}$
$\begin{cases} |d_x| = B_{\phi_i} \times cos(\theta_i) + B_{\phi_{i+1}} \times cos(\theta_{i+1}) \\ |d_y| = B_{\phi_i} \times sin(\theta_i) + B_{\phi_{i+1}} \times sin(\theta_{i+1}) \end{cases}$
$\theta = \begin{cases} 180 - \theta\ if\ quadrant\ II \\ \theta\ \ if\ others \end{cases}$
$\theta_i, \theta_{i+1}, B_{\phi_i}, B_{\phi_{i+1}}$

Fig. 3: Data flow of proposed methodology

As shown in the Figure 3 and Equations 15 16, all of multiply operations, we have already known one of two numbers. It allows us converting all multiples into series of shift and addition. Table II shows all of multiply cases in cell histogram generation in form of shift and addition. With sin and cos functions, 8-bits length of fractional part is used, it is equivalent to multiply 256. To change the size of fractional part, we only need to change the cos, sin parts in the Table II. For example, if an application needs 10-bits length of fraction, the cos, sin functions need to multiply 1024 instead of 256. From the Table II and the data flow in Figure 3, the total number of operations to calculate two voted bins spends above 30 additions and nearly 40 shifts.

IV. Implementing Proposed Methodology into Hardware and Experimental Results

A. Hardware implementation

In this section, we implement the proposed methodology into hardware and examine the results. Figure 4 shows the hardware module of the proposed methodology. It includes 4

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

input pixels $I(x+1, y), I(x-1, y), I(x, y+1), I(x, y-1)$ with 8-bits pixel length. Output contains the 2 couples of angle and magnitude, which represents the voted bins of current pixel $I(x, y)$. The angle output is 4-bits length, and the magnitude is 17-bits length with 8-bits of fractional part.

As shown in Figure 4, the module includes 4 states. Firstly, the difference d_x and d_y are computed from the 4 input pixels. In the second state, the absolute of d_x, d_y are computed, and the quadrant position of the gradient of the pixel $I(x, y)$ is detected. After having those absolute values, the pre-calculating state calculates parallel all the multiples in the Table II. At the fourth state, the two voted angles are detected by only comparison operations and one subtraction, and the magnitude is from one subtraction as and only 1 multiply with $\frac{1}{\sin(20)}$ as the Equation 15 or 16.

B. Experimental results

Figure 5 shows a error of calculation model of the proposed methodology. From the voted bin, we calculate a model of d_x and d_y by Equation 13 14: d'_x and d'_y. With the d'_x, d'_y, d_x, d_y, the percentage error of calculation at both Ox e_x and Oy e_y axises are computed. Tested with all cases, the results prove that our proposed methodology provides The percentage error of calculations e_x e_y are always less than 2% with 8-bits length of fractional part.

table of hw results comparison

V. Conclusion

In this paper, we have proposed the methodology to do cell histogram generation part with only set of addition and shift operation. With this methodology, we spend about 30 additions and 40 shifts to compute the two voted bin of a pixels. Another important characteristic of this methodology is that it provides very low percentage errors, it is always less than 2%. And to change the percentage errors, we only need to change the size of fraction part via multiplying cos sin function with another number. Implementing the proposed methodology into hardware takes about 3.5 KGates of area costs with 45nm NanGate standard cell library The hardware module runs at maximum frequency 400 MHz, and the throughput is about 0.4 (pixel/ns) for single module. It is able to support about 48 frames/s with 4K UHD resolution.

Our future work includes implementing full HOG algorithm into hardware with our optimization in cell histogram generation and the other optimizations in the normalization and SVM phases.

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] F. Karakaya, H. Altun, and M. A. Cavuslu, "Implementation of HOG algorithm for real time object recognition applications on FPGA based embedded system," in *IEEE 17th Signal Processing and Communications Applications Conference*, pp. 508–511. [Online]. Available: http://ieeexplore.ieee.org/document/5136444/

[6] Q. Gu, T. Takaki, and I. Ishii, "Fast FPGA-Based Multiobject Feature Extraction," vol. 23, no. 1, pp. 30–45. [Online]. Available: http://ieeexplore.ieee.org/document/6210371/

[7] 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/

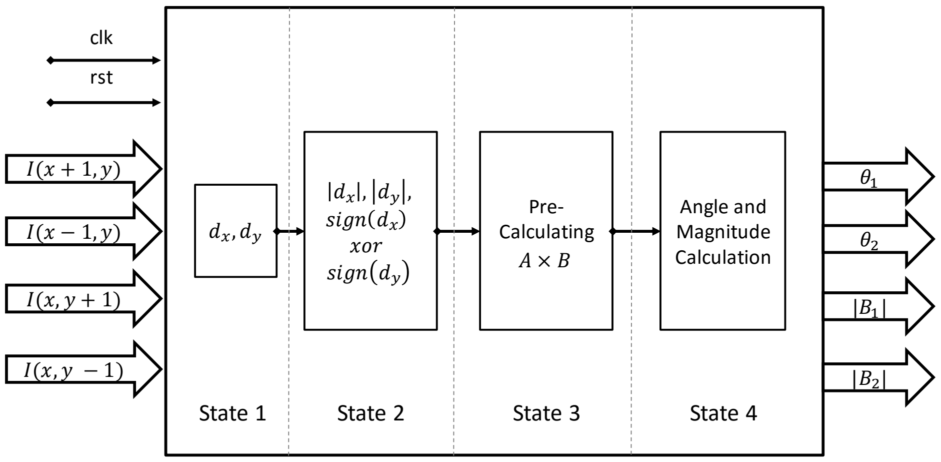


Fig. 4: Hardware implementation of proposed methodology

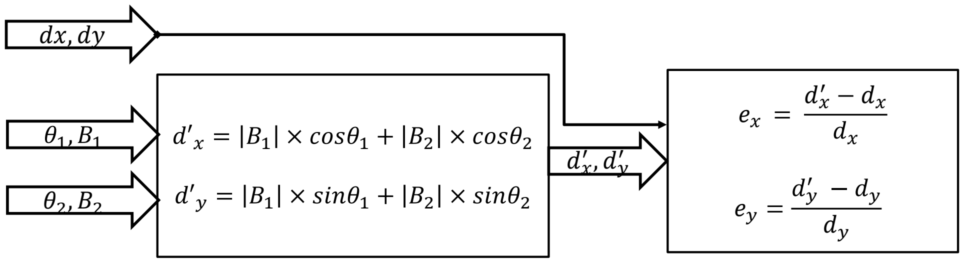

dx, dy
θ_1, B_1
θ_2, B_2
$d'_x = |B_1| \times cos\theta_1 + |B_2| \times cos\theta_2$
$d'_y = |B_1| \times sin\theta_1 + |B_2| \times sin\theta_2$
d'_x, d'_y
$e_x = \frac{d'_x - d_x}{d_x}$
$e_y = \frac{d'_y - d_y}{d_y}$

Fig. 5: Error of calculation model

[8] 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/

[9] A. Suleiman and V. Sze, “Energy-efficient HOG-based object detection at 1080HD 60 fps with multi-scale support,” in *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/6986096/

[10] Pei-Yin 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/

[11] S.-F. Hsiao, J.-M. Chan, and C.-H. Wang, “Hardware design of histograms of oriented gradients based on local binary pattern and binarization.” IEEE, pp. 433–435. [Online]. Available: http://ieeexplore.ieee.org/document/7803995/