A high frame-rate of cell-based histogram-oriented gradients human detector architecture implemented in field programmable gate arrays

Syifa Fuad1, Trio Adiono2, Hans Kasan3

1Hardware Laboratory, Program Studi Sistem Telekomunikasi, Regional Campus of Purwakarta, Universitas Pendidikan Indonesia, Purwakarta, Indonesia
2IC Design Laboratory, Department of Electrical Engineering, School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, Indonesia
3Computer system and Network Laboratory, School of Electrical Engineering and Informatics, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea

ABSTRACT

In respect of the accuracy, one of the well-known techniques for human detection is the histogram-oriented gradients (HOG) method. Unfortunately, the HOG feature calculation is highly complex and computationally intensive. Thus, in this research, we aim to achieve a resource-efficient and low-power HOG hardware architecture while maintaining its high frame-rate performance for real-time processing. A hardware architecture for human detection in 2D images using simplified HOG algorithm was introduced in this paper. To increase the frame-rate, we simplify the HOG computation while maintaining the detection quality. In the hardware architecture, we design a cell-based processing method instead of a window-based method. Moreover, 64 parallel and pipeline architectures were used to increase the processing speed. Our pipeline architecture can significantly reduce memory bandwidth and avoid any external memory utilization. An Altera Field Programmable Gate Arrays (FPGA) E2-115 was employed to evaluate the design. The evaluation results show that our design achieves performance up to 86.51 frame rate per second (Fps) with a relatively low operating frequency (27 MHz). It consumes 48,360 logic elements (LEs) and 4,363 registers. The performance test results reveal that the proposed solution exhibits a trade-off between Fps, clock frequency, the use of registers, and Fps-to-clock ratio.

Keywords:
Field programmable gate arrays
HOG algorithm
Human detection
Parallel architecture
Pipeline architecture

1. INTRODUCTION

Image processing has been widely utilized to detect humans [1], [2]. However, detecting humans in an image is challenging due to their various and wide range of appearance variables [3]. To recognize the existence of humans accurately, extensive computations are required. The system should be highly accurate while still able to maintain low-energy and low resource consumption on real-time processing. In respect of the accuracy, one of the well-known techniques for human detection is the histogram-oriented gradients...
(HOG) method; this method is initially proposed by [3] in 2005. In the publication as mentioned earlier, the HOG technique was proven to outperform other existing human detection techniques significantly. Since then, many researchers have modified the HOG technique to increase the detector performance. Moreover, since HOG has a promising accuracy, the idea to combine the method along with other classification algorithms to distinguish humans under challenging conditions such as illumination, rotation, and even deformation is promising [4]. Besides human detection application, HOG has been widely applied in various cases [5]–[7], such as pedestrian detection, classical dance classification [8], vehicle detection [9], traffic sign detection, crowd density estimation, general object detection, object tracking, feature matching, feature descriptors [10], anomaly detection, digit recognition, and so on.

HOG based human detection has been explored in many aspects. For example, a cascade-of-rejector approach, which is usually utilized for face recognition, was combined with HOG features by [11] to get better accuracy. Jia and Zhang [12] combined the HOG method with Viola’s face detection framework to perform real-time human detection processing. Zhang et al. [13] reported a computational cost reduction using a multi-resolution framework. Wang et al. [14] combined the HOG method with local binary pattern (LBP) as the feature sets in the human detection system. Schwartz et al. [15] utilized a partial least square analysis to provide a richer descriptor. It was similar to edge-based features that utilizes additional color and texture information. There was also research that attempted to combine HOG with human’s body ratio estimation technique to distinguish human from nonhuman category [16]. However, most of these works (on the HOG topics) explored and experimented with improving the accuracy performance using combinations of HOG features and other potential techniques. These accuracy improvements tend to have higher computational costs and complexity than the original HOG algorithm as the result of the technique combinations. Reducing the computation resources is necessary since hardware implementation of HOG is very possible [17].

Hardware implementation offers better speed performance and power efficiency to keep up with real-time processing requirements [7]. Hence it is expected to provide better performance than the software implementation. Many works, as in [18]–[24] utilized a field programmable gate array (FPGA) to implement HOG in hardware as it is able to accommodate parallel architectures and suitable for real-time image processing [18], [25]. Moreover, it can maintain the design configurable and shorten the design time-to-market [26]. Unfortunately, the HOG feature calculation is complicated [4]. Although hardware implementation offers a high-speed computation, it could lavishly consume resource and power if not appropriately designed. On the other hand, resource-efficient and low-power systems are currently in high demand. Trends of electronics and applications are going toward green technology, in which case, resource, and energy consumption are important aspects of being considered (i.e., as low as possible). For this reason, this paper, we designed a simplified HOG algorithm, digital hardware architecture and its FPGA implementation. The proposed design is dedicated to low-power and resource-efficient characteristics.

Section 1 of this paper explains the research background and a glance on the HOG technique. In section 2, the simplified algorithm is presented along with the equations that have been modeled to avoid large division operations. Later, we describe our hardware architecture that realizes the simplified algorithm presented in section 3; it is then followed by the FPGA implementation and its functionality and verification results. Performance evaluation is also presented in section 3, and it is enriched with benchmark comparisons with other techniques. Finally, we draw a concluding remark to highlight the research and its significant contribution.

2. MODIFIED HOG ALGORITHM

To reduce the computation complexity, we simplify the computation of HOG-based human detection algorithm to make it suitable for hardware implementations. Despite the computational reductions, detection quality can still be maintained. The original HOG algorithm has high computational complexity due to its division operations and intensive looping operations in its window-based processing. Thus, it is more suitable to be implemented using the software due to its complex processes. It has to be simplified and modified to suppress its cost and power consumption to be ideal for application-specific integrated circuit (ASIC) implementation, which is commonly referred to as its redundancy and concurrency may be exploited for parallel and pipeline processing (as addressed by [27]). The designed hardware architecture specifications are presented in Table 1.

Based on the basic idea of HOG algorithm, the input image is divided into cells (C), blocks (B), and windows (W) [3], as illustrated in Figure 1. The normalized gradients for these properties, are eventually collected over a Window-based detection for person or non-person classification. The cells consist of 8x8 pixels. Therefore, there are 80×60 cells within a frame. We index the cell in raster scan from 1 to 4,800 (C1 to C4,800). Every 2x2 cells are grouped into a block. There are 50% overlapping Cell data between each Block and its neighbor blocks, both in the horizontal and vertical direction. Therefore, we will have 79x59 Blocks within a frame, indexed as 1 to 4,661 (B1 to B4,661). The window consists of 8x16 cells or equivalent to

_A high frame-rate of cell-based histogram-oriented gradients human detector architecture ..._ (Syifaul Fuada)
64x128 pixels. These numbers of cell size, block size, window size and block overlap give the least miss rate compared to other sizes and overlaps [3]. The window will be moved by one cell column or one cell row after each evaluation. The pixels in the window will be classified by the support vector machine (SVM) for human detection. There are in total 3,285 windows to be analyzed in a 640x480 pixels image. To be fit for hardware implementation, we modify the HOG algorithm by proposing cell-based processing, cell derivatives with neighboring edge anti-aliasing, magnitude calculation using linear approach, fixed-weighted binning, block normalization using newton-raphson algorithm, block-wise SVM classification and fixed-point representation methods. The detail of each technique will be described in the following section.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image size</td>
<td>640x480 pixels</td>
</tr>
<tr>
<td>Window (W) size</td>
<td>64x128 pixels</td>
</tr>
<tr>
<td>Block (B) sizes and Cell (C) size</td>
<td>2x2 cells and 8x8 pixels</td>
</tr>
<tr>
<td>Bin size</td>
<td>9 (0°-180°)</td>
</tr>
<tr>
<td>HOG feature size</td>
<td>3,780</td>
</tr>
<tr>
<td>#Window and #Block row and #Block column</td>
<td>3,285 and 15 and 7</td>
</tr>
<tr>
<td>Gradient binning</td>
<td>Linear approach to avoid Euclidean distance (L2-norm)</td>
</tr>
<tr>
<td>Histogram normalization</td>
<td>Manhattan distance (L1-norm) to avoid Euclidean distance (L2-norm)</td>
</tr>
<tr>
<td>Block overlap</td>
<td>50%</td>
</tr>
<tr>
<td>System clock</td>
<td>27 MHz</td>
</tr>
</tbody>
</table>

![Figure 1. Illustration of HOG image structure, (disclaimer: the figure is not drawn to real scale)](image)

### 2.1. Cell-based processing

Instead of using Window-based, we used Cell-based processing for computing the derivative value in the x-direction (dx) and in y-direction (dy). Figure 2(a) and Figure 2(b) illustrate the how the window-based and cell-based processing in raster scan is executed, respectively. Using cell-based processing, we can extremely reduce derivative computation redundancy by skipping overlapped cell data computations in window-based processing.

As shown in Figure 2(a), cell-based processing eliminates large overlapped cell area, which results in low computational complexity as well as low memory bandwidth requirements [28]. However, the derivative value (dx and dy) results are still identical to the original Window-based HOG algorithm. This method can be applied because we can reuse the computed cell derivative (dx and dy) data for different Windows instead of recalculating the derivative of all the cells inside a window each time a new window is evaluated. This proposed method is different from [22], where the computation of the overlapped data is done using complex pipeline stage. In this method, we store the calculated cell data in temporary random-access memory (RAM). Each time the system analyzes a new window, it will fetch the respective cell calculation results from the RAM, hence avoiding unnecessary recalculation.
2.2. Cell derivates with edge neighboring anti-aliasing

In the HOG algorithm, the derivative values \((dx, dy)\) are computed for every pixel using convolution kernel as (1). Since we utilize cell-based calculation, there will be many edges within a window [29]. The edges may produce invalid \(dx\) and \(dy\) values of pixels located in corner areas as the pixels only possess one adjacent pixel instead of two. In order to combat this problem, we assign the \(dx\) and \(dy\) values of pixels in the edge areas to similar values to its neighbor pixels, as shown in Figure 3. We apply this method to pixels located in both horizontal and vertical edges. As illustrated in Figure 3, grey-colored squares represent pixels with distinct derivatives. meanwhile, blue- and yellow-colored squares represent pixels with identical derivatives as the result of duplicating the derivative values of adjacent pixels.

\[
h_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \quad h_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}
\] (1)

Figure 3. The \(dx\) and \(dy\) values of cell (8x8 pixels) in the edges are similar to the derivative values of their neighbor pixels (color version of this images can be distinguished in the online article version)

2.3. Magnitude calculation using linear approach

The original HOG method used euclidean distance (L2-norm) of each derivative value \((dx, dy)\) to get the magnitude of each pixel. However, the equation of L2-norm consists of a square-root calculation [30], [31], which is very complex to be implemented in the hardware [32]. Therefore, this complicated computation needs to be avoided by other approaches for estimation purposes. In this work, we use a linear method (2) to calculate the magnitude, instead of L2-norm. Figure 4 reveals the comparison between L2-norm \((X: 30, Y: 40)\) and linear methods \((X: 30, Y: 42.43)\).

\[
M(x, y) = \begin{cases} 
\frac{dy}{3} + dx & \text{if } dx \geq dy \\
\frac{dx}{3} + dy & \text{if } dx < dy 
\end{cases}
\] (2)

A high frame-rate of cell-based histogram-oriented gradients human detector architecture ... (Syifaul Fuada)
As formulated in (2), the magnitude is denoted as $M(x, y)$. Our linear method can significantly reduce computational complexity as it merely uses addition and division-by-constant operations. The division by three can be implemented by simple shifting and adding function as described in (3).

$$a_3 = (a \gg 2) + (a \gg 4) + (a \gg 6)$$  (3)

Based on Figure 4, it can be seen that this simplification is able to deliver a satisfactory estimation of the actual $L_2$-norm. Three different approximations – for calculating magnitude, angle and distance – will be used to avoid square root and divisions in the processing of the HOG. In this work, we do not examine the effects of such simplifications as it only to show the proposed linear method compared to $L_2$-norm on a graph, as shown in Figure 4. The overall accuracy for the proposed system will be introduced in the future work as we will provide the error rate for a specific set of benchmarks.

2.4. Fixed weighted binning

For Histogram function, pixel angle can be calculated using complex $\arctangent$ and division operations. However, computing pixel angle with $\arctangent$ function and division will result in a very complex computation and of course, it is not suitable for hardware implementation. Furthermore, since pixel angles are computed for all pixels, there will be a lot of data to be analyzed. This computation demands very large computation cycles and latency. To cope with the requirements, we use a simplified method by setting a fixed region of bin for every 10° (i.e., $\tan(10^\circ)$, $\tan(30^\circ)$, $\tan(50^\circ)$, $\tan(70^\circ)$, $\tan(90^\circ)$, $\tan(110^\circ)$, $\tan(130^\circ)$, $\tan(150^\circ)$, and $\tan(170^\circ)$). Suppose $\tan(10^\circ) = 0.1763269807$, then it will similarly equal to $2^{-3} + 2^{-5} + 2^{-6}$, which is 0.171875. Thus, we will have 9 bins with its approximated $\tan$ values, as shown in Table 2.

<table>
<thead>
<tr>
<th>Tangent</th>
<th>Approximated value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tan(10^\circ)$</td>
<td>$2^{-3} + 2^{-5} + 2^{-6}$</td>
</tr>
<tr>
<td>$\tan(30^\circ)$</td>
<td>$2^{-1} + 2^{-4} + 2^{-6}$</td>
</tr>
<tr>
<td>$\tan(...)$</td>
<td>$...$</td>
</tr>
<tr>
<td>$\tan(170^\circ)$</td>
<td>$-2^{-3} - 2^{-5} - 2^{-6}$</td>
</tr>
</tbody>
</table>

The pixel angle of pixel $A(x, y)$, is computed from derivative $dx$ and $dy$ values using the following pseudocode:

**Algorithm 1** Pixel angle pseudocode

\[
\begin{align*}
\hat{\theta}_i &= \frac{x_i - x_{i-1}}{y_i - y_{i-1}} \\
\theta_i &= 10
\end{align*}
\]
\( \theta_{\text{block#1}} = 20; \)
\( \text{while} \ (d_x \cdot \tan (\theta_1) < d_y \leq d_x \cdot \tan (\theta_{\text{block#1}})) \{
A(x,y) = \theta_{\text{block#1}};
\text{exit();}\}
\text{else}[
\theta_i = \theta_{\text{block#1}};
\theta_{\text{block#2}} = \theta_{\text{block#1}} + 10;\]

The tangent multiplication can be approximated with bit shifting and addition operations in the hardware implementation. By considering computed pixel value \( A(x,y) \), we can calculate the Histogram using the rules as described in Table 3. If the pixel angle \( A(x,y) \) lies on the bin center, then the magnitude value \( M(x,y) \) value is entirely stored into the respective bin. For example, if \( A(4,4) = 20^\circ \) and \( M(4,4) = 1.5 \), then bin\#1 = 1.5. On the other hands, if the pixel angle \( A(x,y) \) lies on the bin boundary, then the magnitude value \( M(x,y) \) is split equally into both neighboring bins. For example, if \( A(5,5) = 30^\circ \) and \( M(5,5) = 4 \), then bin\#1 = 2 and bin\#2 = 2. This scheme has also been used by [33]–[35].

<table>
<thead>
<tr>
<th>Angle</th>
<th>Bin center?</th>
<th>Target Bin</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0°</td>
<td>No</td>
<td>#1 and #9</td>
<td>( M(x,y) / 2 )</td>
</tr>
<tr>
<td>10°</td>
<td>Yes</td>
<td>#1</td>
<td>( M(x,y) )</td>
</tr>
<tr>
<td>20°</td>
<td>No</td>
<td>#1 and #2</td>
<td>( M(x,y) / 2 )</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

2.5. Block normalization using newton-raphson method

There are several normalization methods that can be employed to normalize the Histogram, such as L2-norm and Manhattan distance (L1-norm) [36]–[38]. In this case, L1-norm is more suitable for hardware implementation as it does not use square root operations unlike L2-norm, even though further simplification approaches are still required. Vector normalization is obtained using (4), where L1-sum is Manhattan distance summer.

\[
v_{\text{norm}} = \frac{v}{|v| L1-\text{sum}} \tag{4}\]

Since \( v_{\text{norm}} = v \times d \), the distance \( d \) is stated as (5),

\[
d = \frac{v}{|v| L1-\text{sum}} \tag{5}\]

To calculate \( d \), newton-raphson approximation is used as in (8). It is derived from (6) and (7) that are formula for \( x_0 \) and \( x_1 \) on a newton-raphson digital blocks.

\[
x_0 = (3 \ll n) - (2 \times \text{sum}) \tag{6}\]
\[
x_1 = x_0 [(2 \ll 2n) - (\text{sum} \times x_0)] \tag{7}\]
\[
[\text{representation}] \ d \ll 12 = x_1 \gg (4n - 12) \tag{8}\]

Where \( n \) is defined as \( n = MSB \ (\text{sum}) \). For instance, if \( \text{sum} = 13 \), then \( n = \text{bit 4} \). The result will be delivered in decimal fraction numbers.

2.5.1. Blockwise SVM classification

The idea of this method is to multiply the SVM coefficient blockwise, instead of per-window. However, it is important to note that block\#1 corresponds only to window\#1, but block\#2 corresponds to both window\#1 and window\#2. Thus, block\#2 will be used for SVM classifications of block\#1 and block\#2. This also applies to other blocks that correspond to multiple windows. Section 3 (results and discussion) will further explain the hardware design, which takes advantage of pipelined architecture to handle these complicated calculations. The SVM coefficients are trained with the simplified algorithm. We used the libsvm library to train our SVM with massachusetts institute of technology (MIT) pedestrian dataset. Then, we examined the linear SVM and retrained the false positives. The number of images used for classifier training is amount of 924 and 13,680 for positive and negative images trained, respectively.

* A high frame-rate of cell-based histogram-oriented gradients human detector architecture ... (Syifaul Fuada)
2.5.2. Fixed-point representation

Fixed-point is used to represent the fractional data. The data-width of all the modules is depicted in Table 4; it contains input pixel, derivatives, magnitude, and so on. The bit-width is determined by searching for the shortest bit-width in each module that will not cause any bit-overflow or interfere with the calculation results.

<table>
<thead>
<tr>
<th>Module</th>
<th>Sign</th>
<th>Bit-Width</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input pixel</td>
<td>Unsigned</td>
<td>8</td>
<td>Integer</td>
</tr>
<tr>
<td>Derivatives</td>
<td>Signed</td>
<td>9</td>
<td>Integer</td>
</tr>
<tr>
<td>Magnitude</td>
<td>Unsigned</td>
<td>9</td>
<td>Integer</td>
</tr>
<tr>
<td>Histogram bin</td>
<td>Unsigned</td>
<td>15</td>
<td>Integer</td>
</tr>
<tr>
<td>Normalized</td>
<td>Unsigned</td>
<td>12</td>
<td>Fraction, $\ll 12$</td>
</tr>
<tr>
<td>SVM coefficient</td>
<td>Signed</td>
<td>14</td>
<td>Fraction, $\ll 12$</td>
</tr>
<tr>
<td>Window score</td>
<td>Signed</td>
<td>32</td>
<td>Integer</td>
</tr>
<tr>
<td>Detection</td>
<td>Unsigned</td>
<td>1</td>
<td>Integer</td>
</tr>
</tbody>
</table>

3. RESULTS AND DISCUSSION
3.1. Hardware architecture implementation

Our system block diagram is shown in Figure 5. The input of the system is 640×480 images. The output is a grayscale image on an external display. The output image will be marked in parts of the image that are believed to be human figures. The system is comprised of a control unit, derivative, gradient binning using linear approach, cell grouping, Histogram normalization using $L1$-norm, and sliding window and SVM classification modules.

In order to increase the processing speed and enable the system to work in a real-time, we applied pipeline architecture to our system, as reported in Table 5. The M9K is memory block of altera FPGA DE2-115. The pipeline architecture also enables us to reduce the memory bandwidth as it does not require all cell and block values to be stored, but only some blocks that correspond to the windows being processed at that time. Consequently, we are able to use embedded RAM (internal RAM) as the processing storage instead of external RAM, which translates to a significant reduction of pins utilization and power consumption. The pipeline architecture enables us to reduce the clock cycle latency dramatically. The overall process without pipelining is 13,112; it is obtained from the summation of three digital blocks (i.e., cell grouping, block normalization, and SVM classification processes). However, as a note, pipeline does not allow any latency improvements for the SVM classification, the block normalization and the cell grouping modules.

Control unit module is used to generate read and write addresses for embedded RAM used in each module. This module plays an important role in our pipeline architecture since the RAM access scheduling should be accurate at all times. Embedded RAMs used and its memory size is Table 6.

The derivative module calculates the cell-based derivative ($dx$ and $dy$) of the image; this block contains pixel derivatives as shown in Figure 6 and anti-aliasing filter as shown in Figure 3. The anti-aliasing filter applied to the pixels in the image edges is to overcome the zero-padding convolution problem. We...
designed highly parallel architecture, as shown in Figure 7. Our architecture can simultaneously calculate the derivatives and gradient bins for 64 pixels by calculating them simultaneously. This will significantly reduce clock latency.

Table 5. Reduced cycle count by pipeline architecture

<table>
<thead>
<tr>
<th>Blocks</th>
<th>Without pipelining</th>
<th>With pipelining</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell grouping</td>
<td>4,800</td>
<td>4,800</td>
</tr>
<tr>
<td>Histogram normalization</td>
<td>4,719</td>
<td>4,719</td>
</tr>
<tr>
<td>SVM Classification</td>
<td>3,593</td>
<td>3,593</td>
</tr>
<tr>
<td>Overall process</td>
<td>13,112</td>
<td>4,888</td>
</tr>
</tbody>
</table>

Table 6. Embedded RAMs

<table>
<thead>
<tr>
<th>Embedded RAM</th>
<th>Size</th>
<th>Data-Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame buffer (input image)</td>
<td>(29 M9K block)</td>
<td>8 Bit</td>
</tr>
<tr>
<td>Cell grouping</td>
<td>(4 M9K block)</td>
<td>8 Bit</td>
</tr>
<tr>
<td>Sliding window</td>
<td>(108 M9K block)</td>
<td>8 Bit</td>
</tr>
<tr>
<td>Store detection result</td>
<td>(1 M9K block)</td>
<td>1 Bit</td>
</tr>
</tbody>
</table>

Figure 6. Pixel derivates

Figure 7. Cell-based parallel architecture

Gradient binning module consists of a rotator, magnitude calculator and binning unit as Figure 8. The rotator unit is comprised of four digital blocks (i.e., inverse, bit extender, comparator and multiplexer units), as shown in Figure 9. The inverse unit is used to negate a number based on two’s (2’s) complement notation. Bit extender is used to represent a number with larger bits without altering its value. The $dx$ and $dy$ representation uses 12 bits instead of 9 bits to avoid overflow as the magnitude calculator and binning unit involve shifting and adding the values of $dx$ and $dy$. Comparators and multiplexers are used to determine the quadrant of $dx$ and $dy$. The magnitude calculator conducts approximation in (3) using four digital blocks (i.e., multiplexer, right shifter, comparator, and adder), as shown in Figure 10. Finally, all the magnitudes are grouped and summed to the respective bins based on the rule specified in (5). The output of this module is a cell histogram data consists of 9 bins $\times$ 15 bits that are concatenated into a single line as Figure 11.
This module is used to group cells into a block and performed block normalization processing. To group it with the raster scan sequence, we use line delay, as shown in Figure 12. It is implemented using RAM to delay 80 cells of data, which is the row size of the input image, as shown in Figure 1. Each Cell consists of 9 bins $\times$ 15 bits of data. Using this configuration, we can group four Cells of data that comprise a single Block. For example, when cell #82 data is fed to the module, block#1 which consists of cell #1, #2, #81, and #82, will be constructed. The block data will then be normalized using combinational circuits.

To minimize memory resource usage, we decided to only use line delay RAM with the size of 80 Cells to store all 4800 Cells that will be generated to blocks. The first 80 Cells will be stored initially in the
A high frame-rate of cell-based histogram-oriented gradients human detector architecture ... (Syifaul Fuada)
using the value of the sign bit. The system will continue to analyze a new window after a window has been calculated until the whole image has been inspected. The hardware architecture is Figure 15.

Figure 13. Architecture of Cell grouping hardware design

Figure 14. Histogram normalization architecture

Figure 15. Window sliding and SVM classification
The final module of this design is the display module. In relation with the video graphics array (VGA) controller, this module works by generating read address from frame buffer RAM (input image) and detection RAM (HOG results). As a note, there are no special requirements for the display module because its function is only to draw a detection box over the object to be detected. It uses 640×480 pixels with 25 MHz VGA clock. Both RAMs are fed by the same clock to read the values. By counting the pixels with front and back porch, the VGA counts to 800×600. Therefore, it needs 800×600 per 25,000,000 second to complete one frame, which is around 52 Fps. In summary, the design must operate at minimum 52 Fps in order to fit the VGA configuration. The display module also generates markings on windows that are considered to contain human figures.

3.1.1. Performance implementation

To evaluate our architecture in terms of effectiveness parameter, we implemented our design in FPGA. We used altera DE2-115 board (Cyclone IV EP4CE115 FPGA chip). The board is connected to an external VGA display to show the resulting image. We tested several 640×480 pixels color images to verify the system functionality. For static detection, we chose the images with relatively small-sized pedestrian images with 128×64 pixels instead of full image 640×480 pixels. The image in Figure 16(a) contains the best pedestrian detection. The pedestrians are quite similar to our positive training dataset. In Figure 16(b), the image has various lighting conditions. However, since HOG uses the gradient feature, our detector can still detect the pedestrians and does not interpret the shadows as humans. On the other hand, in Figure 16(c), the HOG detector may not be able to detect various poses reliably as their gradients will vary. To increase the detector performance, images containing various poses should be used as our training dataset.

![Figure 16. Detection results on pedestrians: (a) under uniform lighting; (b) under various lighting conditions and (c) with various poses](image)

Figure 17(a) shows the FPGA displaying the detection result to a monitor. The marks drawn on humans indicate successful detections. Our architecture is coded in Verilog hardware description language (HDL). We use a top-down approach to design the system architecture and a bottom-up approach to code the hardware modules. Each sub-module is designed and tested before being integrated. Figure 17(b) shows the flow summary of the design analysis and synthesis result. The design consumes 48,360 logic elements (LEs), 4,363 registers, and 84 of 9-bit embedded multipliers. It merely consumes 0.141 Mbits of memory. Our architecture requires 4,888 clock cycles to complete one frame detection of image. It also needs 640×480 = 307,200 clock cycles to receive the image data and store them into the frame buffer. Therefore, the overall system needs 312,088 clock cycles (calculated from 307,200 + 4,888) to process one frame of image data. It is important to note that the frame buffer embedded RAM is a huge speed bottleneck as it is only able to deliver one pixel every clock cycle.

The TimeQuest Timing Analyzer shows that the maximum operating frequency allowed for the design is 28.62 MHz. By setting the system clock frequency to 27 MHz, we obtained 86.51 Fps (obtained from 27,000,000/312,088). Since the VGA has 52 Fps refresh rate, our design will be suitable to be used with VGA due to the frame output will always be available every time the VGA refreshes. However, the Fps may be improved significantly by reducing the latency of the frame buffer because most of memory resources are consumed by the input buffer. The actual processing unit (without frame buffer) is able to deliver 5,523.732 Fps (obtained from 27,000,000/4,888). This indicates that the frame buffer latency has severely overshadowed the actual capability of the processing unit. It may possible to reduce resources in particular modules and keep the algorithm consistency at the smaller throughput.
As shown in Figure 15, it is regarded as a first-in-first-out (FIFO) function, a static random-access memory (SRAM) to can be used to replace it. Therefore, gate count and power can be saved. The FIFO always toggles to consume power, but SRAM only activated one cell to access data. Moreover, ping-pong mode can enable a SRAM to perform “read” and “write” concurrently. The detection success of the proposed algorithm will be compared with the software implementation of the original HOG algorithm in the near future work. This becomes an open challenge that should be exploited.

3.1.2. Performance comparison

Table 7 and Table 8 evaluate the performance result with other competitors. The strongest points compare to the others are frame per second (Fps) and Fps-to-clock ratio. All the competitors exploit standard FPGA boards. Compared to the earlier works, our implementation performs significantly well in terms of the ratio of delivered Fps and operating clock frequency. This is made possible with our pipeline architectures and custom-designed hardware modules, instead of using general-purpose processors. Processors may be smaller in size, but dedicated modules are more efficient compared to processors. Moreover, all cells and blocks will not be kept in the RAM concurrently. Unused data will be overwritten to minimize memory usage. Additionally, the low operating clock frequency translates to less power consumption.

Working at a low frequency surely allows power consumption to be reduced. For instance, in Table 7 [23] has a much higher image resolution (1920×1080), higher operational clock (270 MHz), and a lower frame rate (64 Fps). Instead, the proposed work has a lower image resolution (640×480), ten times lower operational clock (27 MHz), and a high frame rate (86.51 Fps). This becomes the architecture design trade-off. In [39] has the highest Fps (162 MHz) but the operating clock is the highest (150 MHz) resulting in lower Fps-to-clock ratio. Instead, the proposed work the highest Fps-to-clock ratio (3.2041), lower frame rate, lower operating clock (27 MHz), more efficient in memory usage (141,872 bits), and the lowest registers (4,363). With the same image resolution usage (640×480), in [40] has the lowest embedded multipliers (40 DSP block) as well as the operating frequency (25 MHz), which is not to close with our...
proposed architecture. But our architecture has a higher Fps and Fps-to-clock ratio, also resource-efficient. In summary, it is safe to say that our architecture is considered the best trade off compared to previous works, in terms of operating clock frequency (MHz), Fps (Hz), Fps-to-clock ratio, and the use of Registers.

Table 7. Performances comparison against other works used Xilinx Virtex FPGA

<table>
<thead>
<tr>
<th>Parameters</th>
<th>[20]</th>
<th>[21]</th>
<th>[23]</th>
<th>[41]</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA board</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUTs**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Registers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Embedded multipliers / DSP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory usage (bits)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating clock frequency (MHz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame per second (Hz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fps-to-clock ratio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image resolution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* N/A: not available.
(↑): The highest value in the comparison table among the equivalent parameters compared
(↓): The lowest value in the comparison table among the equivalent parameters compared
** LUTs in FPGA logic building blocks may not serve as an accurate parameter for design size comparison. LUTs in Cyclone devices have 4 inputs, while Xilinx Virtex-5 has 6 inputs. Moreover, logic building blocks differ among devices, as in Altera’s logic element (LE) and Xilinx’s logic cell (LC), each with their respective hardware design. These factors may cause different synthesis results in Altera and Xilinx FPGAs.
*** Embedded multiplier refers to 9x9 multipliers in altera icycle III and 4. DSP block refers to DSP48E slice in Xilinx Virtex-5, which is equipped with a 25x18 multipliers.

Table 8. Performances comparison against other works used ALTERA Cyclone FPGA

<table>
<thead>
<tr>
<th>Parameters</th>
<th>[43]</th>
<th>[44]</th>
<th>[45]</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA board</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUTs**</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Registers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Embedded multipliers / DSP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory usage (bits)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating clock frequency (MHz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame per second (Hz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fps-to-clock ratio</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image resolution</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(↑): The highest value in the comparison table among the equivalent parameters compared
(↓): The lowest value in the comparison table among the equivalent parameters compared

4. CONCLUSION AND FUTURE WORKS

This paper presents a hardware architecture design to implement a simplified HOG algorithm. We have designed a cell-based raster scanning computation instead of window-based to reduce computation redundancy. The magnitude calculation using a linear approach provides us with a reasonable approximation of magnitude without using exponentiation and square root operations; due to L2-norm approach is too difficult to be implemented in hardware implementation. By using fixed-weighted binning for histogram classification, we can avoid using arc tangent and division operations. Furthermore, by using the newton-raphson algorithm, we can execute block normalization without using any division operations. Finally, the overall parallel and pipeline architecture gives accurate detection with less memory usage and maximum Fps-to-clock frequency ratio. This work used MIT pedestrian dataset for training. The primary feature of this work is to simplify HOG for an efficient hardware implementation. This simplification certainly makes some degradation on the performance of original HOG. It is important to examine in detail the performance of original HOG compared to this work (simplified HOG) further. We will also address several interesting issues, e.g., the impact of computational reduction in term of detection accuracy, measure the throughput achieved of the GPU implementations, and a more objective figure-of-metric (FOM). This will be considered to prove that the proposed hardware architecture is more efficient than the other competitors. Later, the effect of accuracy improvement to computational costs and system complexity will be evaluated further. In the recent years, various other challenging datasets have been introduced by many researchers. Therefore, we will use various dataset provided globally and dataset produced ourselves to train and evaluate our proposed system comprehensively.

A high frame-rate of cell-based histogram-oriented gradients human detector architecture ... (Syifaul Fuada)
AKNOWLEDGMENTS

The publication fee for this work was sponsored by Integrated Circuit Design Laboratory Pusat Unggulan IPTEK Perguruan Tinggi (PU-PT) Mikroelektronika ITB and Program Peningkatan Global Competitiveness Perguruan Tinggi Indonesia Universitas Pendidikan Indonesia 2021 Batch II with No SK 1370/UN40/PT.01/02/2021.

REFERENCES


BIOGRAPHIES OF AUTHORS

Syifaul Fuada is with the the Study Program of Telecommunications, Universitas Pendidikan Indonesia as a young lecturer. Mr. Fuada has several has several achievements, such as the most student outstanding award of Universitas Negeri Malang in 2013, a top of 10-student travel grant to the IEEE Asia Pacific Conference and Systems (APCCAS 2016) that was held in Jeju (South Korea), receiving one of 108 Indonesia Innovations by BIC-LIPI awards (2016) for Smart Home Product, student winner nominee of NOLTA conference (2017), receiving Best Paper Award from IEEE IGBSG 2018 that was held in Yi-Lan, Taiwan, receiving Best Paper Award from IEEE ICTRuDev 2018 that was held in Bali, Indonesia, receiving the Best paper award from a Scopus-indexed journal, i.e., i-JOE in 2019, receiving Best Paper Award from IEEE IGBSG 2019 that was held in Yichang, China, receiving the 111 Indonesia Innovations by BIC awards (2019) for E-Nelayan and LI-FI products, receiving the 112 Indonesia Innovations by BIC awards (2020) for three innovations: Bidirectional DC/DC Converter for Electric ATV, Smart Home, and Contact/Contactless-based Payment Device, the 3rd order human detector architecture … (Syifaul Fuada)
place of UPI’s most productive researcher in the SCOPUS-based publication 2020 (awarded on 2021), the 3rd place of UPI’s inventor awarded on 2021. His study interests include analog circuit design and instrumentation, engineering education, IoT, image processing, System-on-Chip, and Visible Light Communication. He can be contacted at email: syifaufiuada@upi.edu.

**Trio Adiono** received a B.Eng. in electrical engineering and an M.Eng. in microelectronics from Institut Teknologi Bandung (ITB), Indonesia, in 1994 and 1996, respectively. He obtained his Ph.D. in VLSI Design from Tokyo Institute of Technology in 2002, Japan. In 2005, he was a visiting scholar at MESA+, Twente University, Netherlands. He is a full-professor at Sekolah Teknik Elektro dan Informatika ITB. He was also Adjunct Assoc. Prof. NTUST-Taiwan. He has developed several microchips, such as Binary Template Matching Processor, WiMax Baseband Chipset, Smart Card, and IoT. He also has job experience working with Chip Design House in Fukuoka Japan, handling chip design for several multinational companies. He received the “Second Japan Intellectual Property (IP) Award” in 2000 from Nikkei BP. He received Award Karya Lencana Wira Karya from President of Republic Indonesia for his innovation in 4G chip developments. He also holds a Japanese Patent on “High Quality Video Compression System”. He was also chair of IEEE Solid State Circuits Society. His research interests include VLSI design, signal and image processing, VLC, smart cards, and electronics solution design and integration. He can be contacted at email: tadiono@stei.itb.ac.id.

**Hans Kasan** received his B.Eng. degree in Electrical Engineering from Bandung Institute of Technology (ITB), Indonesia in 2017. He joined Computer System and Network Lab (CSNL) in the Department of Electrical Engineering at Korea Advanced Institute of Science and Technology (KAIST) in September 2018 and now he is pursuing Ph.D. degree with the same department under supervision of Professor John Kim. He has several achievements, one of them is the first runnerup (Electronic Device Industry News award) of the 2017 LSI Design Contest in Okinawa. He has published papers about RFID-based warehouse management system and Montgomery multiplier IC design. His current research topic is adaptive routing in interconnection networks. He can be contacted at email: hanskasan@live.com.