Low-power coprocessor for Haar-like feature extraction with pixel-based pipelined architecture

To cite this article: Aiwen Luo et al 2017 Jpn. J. Appl. Phys. 56 04CF06

View the article online for updates and enhancements.
Low-power coprocessor for Haar-like feature extraction with pixel-based pipelined architecture

Aiwen Luo1,*, Fengwei An2,*, Yuki Fujita1, Xiangyu Zhang2, Lei Chen3, and Hans Jürgen Mattausch1,3,4*

1Graduate School of Advanced Sciences of Matter, Hiroshima University, Higashihiroshima, Hiroshima 739-8530, Japan
2Graduate School of Engineering, Hiroshima University, Higashihiroshima, Hiroshima 739-8530, Japan
3HiSIM Research Center, Hiroshima University, Higashihiroshima, Hiroshima 739-8530, Japan
4Research Institute for Nanodevice and Bio Systems, Hiroshima University, Higashihiroshima, Hiroshima 739-8530, Japan

*E-mail: aiwen-luo@hiroshima-u.ac.jp; anfengwei@hiroshima-u.ac.jp; hjm@hiroshima-u.ac.jp

Received October 18, 2016; accepted December 30, 2016; published online March 7, 2017

Intelligent analysis of image and video data requires image-feature extraction as an important processing capability for machine-vision realization. A coprocessor with pixel-based pipeline (CFEPP) architecture is developed for real-time Haar-like cell-based feature extraction. Synchronization with the image sensor’s pixel frequency and immediate usage of each input pixel for the feature-construction process avoids the dependence on memory-intensive conventional strategies like integral-image construction or frame buffers. One 180 nm CMOS prototype can extract the 1680-dimensional Haar-like feature vectors, applied in the speeded up robust features (SURF) scheme, using an on-chip memory of only 96 kb (kilobit). Additionally, a low power dissipation of only 43.45 mW at 1.8 V supply voltage is achieved during VGA video procession at 120 MHz frequency with more than 325 fps. The Haar-like feature-extraction coprocessor is further evaluated by the practical application of vehicle recognition, achieving the expected high accuracy which is comparable to previous work. © 2017 The Japan Society of Applied Physics

1. Introduction

Machine vision is a notable and attractive research issue due to its great potentials in many innovative applications, such as smart home, robotics, wearable vision or interactive games.1,2 Feature-vector-based object detection and recognition play an important role in these fields. Commonly, feature vectors are extracted from raw input pixels for the parts of interest in an image.3 Many feature extraction algorithms were proposed to provide descriptors of significant image features based on a variety of principles.4–9 Specifically, the original scale-invariant feature-transform (SIFT) algorithm is considered as one of the most robust approaches because of its invariance in scale, rotation, and illumination, which are essential in many applications. The speeded up robust features (SURF)3,6 algorithm was proposed to solve the high computational cost and storage requirements of SIFT. The original SURF consists of two distinct stages: detection and description. The main work of the detection stage is to search for essential interest points in scaled image-frame pyramids. The description stage of SURF aims at collecting Haar-filter responses around each interest point in chosen orientations, which constitutes the components and determines the dimension of the feature vector. The Haar sequence was proposed by Alfréd Haar9 in 1910, and has been widely used and highly developed until now. The advantage of the Haar-wavelet responses, which are calculated for the SURF descriptor, is that they enable higher processing speed and better repeatability than other descriptors, thus decreasing object-matching complexity and enhancing computational performance.

Both hardware and software technologies were introduced to support and accelerate the improvement of feature descriptors.10–12 However, these improved implementations for feature extraction are computationally demanding and therefore represent a bottleneck for the processing speed. As hardware platforms, field programmable gate arrays (FPGAs) and graphics processing units (GPUs) are adopted in many previous designs for realizing parallel architectures in embedded vision applications.13–18 The resulting practical designs have enhanced computational capability significantly and enable the handling of enormous data volumes. Specifically, the design cycle using an FPGA is shorter and costs less than an application-specific integrated circuit (ASIC) for small amounts of produced systems. However, the power consumption becomes higher and the design options are limited by the resources in an FPGA platform. By comparison, a GPU solution is often a faster, cheaper and more efficient solution during the hardware design by applying the same instruction to multiple data points, i.e., applying the single-instruction multiple-data (SIMD) concept. However, there is hardly any mature standard model for GPUs, and not all algorithms can be speeded up with a GPU.

Compared to central processing unit (CPU)-based or GPU-based systems, Ref. 16 presented a standalone FPGA-based embedded module for the SURF implementation which can process up to 10 XGA (1024 × 768 pixels) frames per second while the FPGA-based solution in Ref. 17 can support up to 56 VGA (640 × 480 pixels) frames per second. Both works implemented the complete calculation of SURF including detection and description stages and are based on the integral-image concept. Reference 18 took advantage of an FPGA implementation to accelerate multi-object tracking at high frame rate using higher-order local autocorrelation (HLAC) as a shift-invariant feature for 512 × 512-pixel images. However, the HLAC feature cannot reflect the overall information of an image.

Furthermore, four directional edges (vertical; horizontal; ±45°) were detected in Ref. 19 from the data of each 5 × 5-pixel filtering kernel in a fixed 64 × 64-pixel partial image achieving 5 VGA frames per second at 100 MHz operation frequency. Reference 20 presented an improved edge-filtering solution for real-time motion-feature extraction by preserving the maximum gradient value only among the four-directional edge filters so that the processing speed in Ref. 20 is faster than Ref. 19. The solution of an all-hardware SIFT accelerator21 could reach real-time operation when the number of features was no more than 890 in one image.

On the other hand, software libraries like OpenCV as well as OpenCL can support rapid prototyping and shorter
development periods, but the dependence on high-performance hardware devices or platforms makes these software-library solutions unsuitable for largely outdoor mobile-application scenarios.

The concept of integral-image calculation has been employed for remarkable improvements in feature calculation speed throughout most of the current algorithms. Consequently, many prior works on the FPGA implementation of the SURF descriptor have therefore been based on the integral-image concept. These integral images allow fast computation of rectangular Haar-like features at a high constant speed, independent of filter size. Although the calculation of integral images only consists of a few simple addition operations per pixel, in total a massive number of operations is necessary, owing to the generally large amount of necessary pre-stored image data. Therefore large storage capacity for raw image buffering and integral image storage is required. In other words, the previous software- and hardware-based implementations are restricted in their capabilities by the limited resources, in particular memory resources, which are available in embedded applications.

For efficiency-critical and resource-constrained embedded system, computation and storage of the integral image present therefore great challenges, due to the strict timing and hardware limitations. Specific problems are that most state-of-the-art image recognition strategies consume significant resources and power, when involving the sliding-window paradigm. Consequently, data flow organizations and computational resources, as well as processing speed are crucial aspects to be taken into account. Particularly, improvements in the memory management may enable a substantial reduction of the access time.

In original SURF algorithm, the Haar-like features are extracted only around each detected interest point for object matching. Whereas, in this work, instead of only the interest points, an approach with overlapping scan windows is performed for covering the entire image. In particular, we extract Haar-like feature vectors (FVs) for recognizing the target objects in every scan window. Therefore, we can construct a pixel-based hardware-friendly algorithm of the cell-based Haar-like feature extraction, which is used in original SURF algorithm for multi-size images, without buffer memories for storing raw or integral images. The designed coprocessor implementation synchronizes with the raster scan manner of the image sensor so that the FV extraction of scan windows can start immediately at the time of serial pixel input. Moreover, the intermediate storage space can be repeatedly used for different rows of scan windows. As a result, the hardware resources, especially for memory, and the power consumption are significantly reduced, due to the pixel-based architecture and cell-based extraction concept. Furthermore, a cell-based scan-window construction algorithm enables parallel recognition for multiple scan windows of the entire image with high processing speed.

This paper is organized as follows. In Sect. 2, we propose a computationally efficient hardware implementation for extraction of the cell-based Haar-like descriptor and the corresponding feature vector of a scan window. The experimental results for the fabricated coprocessor and a discussion of the vehicle-detection application are summarized and evaluated in Sect. 3. Finally, Sect. 4 gives the conclusions.

2. Parallelized feature extraction for cell-based Haar-like descriptor

2.1 Pixel-based parallel Haar-like feature descriptor

A set of Haar-like feature types such as edge features, line features, and rotated features, have been introduced with different restrictions in previous works. The designated sizes of the scan window (SW) and its sub-regions are depended on the location and size of different target objects in the image. For example, in Ref. 35, a window with 64 (width) × 128 (height) pixels is selected for human detection. Then, different components such as head, legs and arms are identified by various sub-regions with 42 × 42 pixels, 69 × 46 pixels, 47 × 31 pixels, etc.

In this paper, we apply the Haar-wavelet features used in the SURF descriptor, which are robust in decreasing object-matching complexity as well as enhancing computational performance. For SURF-descriptor extraction in this work (see Fig. 1), a 4 × 4-pixel array is defined as a sub-cell and further used for constructing cells consisting of 2 × 2 sub-cells as well as a complete scan window (SW) with 16 (columns) × 8 (rows) cells. The SW is used for scanning the image to identify locations containing target objects (i.e., cars) and is represented by a 1680-dimensional Haar-like FV.

For FV construction, blocks of 2 × 2 cells are shifted and overlapped by one cell within each SW. Consequently, each cell has to be reused a number of times during construction of the SW’s Haar-like FV. Furthermore, for capturing all target objects within the complete image during the recognition process, the SW is also moved in units of blocks (i.e., 2 × 2 cells) in horizontal and vertical direction within the image, as illustrated in Fig. 1.

Traditional object recognition strategies generally start by sliding SWs in the pre-stored input images and decompose the SWs into overlapping blocks. As each block is subdivided into cells that are constructed by sub-cells and further by pixels, this processing procedure from image layer to pixel layer causes complex and lengthy searches of pixel positions. The integral image method can help to reduce the computational cost for the SW’s FV. However, the location of overlapping blocks and their inner non-overlapping cells has to be confirmed for every SW separately. Furthermore, most cells are included in many different overlapping SWs, since the SW is shifted in steps of one non-overlapping block across the image, resulting in a massive number of similar computations among the same cells.

Considering the transformation characteristics of each layer in Fig. 1 during above Haar-like FV extraction procedure for the SWs, we propose an alternative solution of FV construction with a pixel-based pipelined architecture, which avoids both image buffers and integral-image calculations. According to the raster scan manner of the general image sensors, we immediately process the line-by-line raw pixel input to progressively determine the Haar-like features of the FV descriptor. For each sub-cell, horizontal $D_1$ and vertical $D_1$ are calculated according to Eqs. (1) and (2) in the sub-cell layer, respectively.
Here, \( p(x) \) and \( p(y) \) represent sub-cell pixel values in horizontal and vertical direction, respectively, as related to image and scan-window orientation of each sub-cell. Absolute values \( |D_x| \) and \( |D_y| \) are also determined to capture the polarity of intensity changes.\(^6\)

Similarly, the sequentially constructed and non-overlapped sub-cells belong to specific cells depending on their location in the image. For the local Haar-like wavelet response of one cell, the response values of four sub-cells that belong to the same cell are added up and form the local four-dimensional cell-response vector \( v_{cell} \) of Eq. (3) at the cell layer.

\[
v_{cell} = \left\{ \sum D_x, \sum D_y, \sum |D_x|, \sum |D_y| \right\}
\]  

As the block is shifted by one cell within one SW in raster scan manner, it causes reutilization of cells by different overlapped blocks for up to 4 times, depending on their respective positions in the SW. The reutilization of cells by different overlapped blocks in one SW can be summarized at the SW layer, as illustrated in Fig. 2. The concept of “regular rule of reusing times” (RRRT) represents the reusing time \( R \) (i.e., 1, 2, or 4) of each cell in one SW and is used for the window-based Haar-like feature construction according to the cell position. Consequently, all 128 cells of one SW are scanned only once and then used to construct the window-level FV of the Haar-like descriptor with \( 4 \cdot \sum_{128} R(\text{cell}) = 1680 \) dimensions according to their reusing time \( R \).

Furthermore, as shown in the window layer of Fig. 1, owing to the sequential pixel input from the image sensor in raster scan manner and the window sliding algorithm, most of the local cell-descriptor vectors belong to much more than one SW. In other words, the same cell is in most cases simultaneously located at different positions in multiple SWs. Both, the number of cell-overlapping SWs and corresponding positions of the current cell in different SWs are also collected at the window-layer. A large portion of the complete window-level FVs can be constructed by simple operations with a look-up table.

Once the local feature vector \( v_{cell} \) of the last cell in a SW is finished, the complete 1680-dimensional Haar-like descriptor FV of this SW is obtained. All related SWs that overlap the same cell are concurrently processed in parallel at the image layer. Each pixel is scanned only once when transferred from the image sensor, and then the corresponding local cell FV is progressively calculated, so that all related window-level FVs can be constructed in real time. The proposed pixel-based Haar-like feature-extraction method thus prevents the occurrence of massively redundant calculations, leading to high computational performance and resource efficiency.

### 2.2 Local cell-based feature vector extraction

Rather than buffering entire images for integral image computing, we developed a cell-based feature-extraction circuitry in which the calculation starts from the serial input pixels of the image sensor. According to the above analysis of Haar-like FVs, applied in the SURF descriptor extraction scheme, a concept which we call “cell feature extraction by a pixel-based pipeline” (CFEPP) is applied to calculate the local four-dimensional cell-descriptor vector component \( v_{cell} \) as explained in Sect. 2.1.

As illustrated in Fig. 1, the concept of the reported Haar-like feature extracting hardware implementation eliminates the pre-storage unit of image frames by transforming the raw image pixels into real-time calculated \( D_x \) and \( D_y \) values of the

---

\( D_x = \sum_{p(x)\in\text{left cell}} p(x) - \sum_{p(x)\in\text{right cell}} p(x) \)  
\( D_y = \sum_{p(y)\in\text{up cell}} p(y) - \sum_{p(y)\in\text{down cell}} p(y) \)
sub-cells. In this research, the raw pixels from the image sensor are transferred to the input unit and then further to the corresponding calculators for $D_x$ and $D_y$ of each sub-cell, as shown in the upper-right parts of the Fig. 3.

Specifically, the pixel values of left- and right-half rectangles in a sub-cell, respectively, are added up by the $D_x$ calculator. On the other hand, the $D_y$ calculator uses an adder/subtractor circuit to sum-up the pixels in the upper-half rectangle of each sub-cell and then to subtract the lower-half sum for obtaining the $D_y$ values. Given an input image with $w \times h$ pixels, there are $w/e$ cells with $e \times e$ pixels and $w/s$ sub-cells with $s \times s$ pixels in the horizontal direction. In this work, each cell has four sub-cells, i.e., $e = 2s$. Intermediate sub-cell calculation results are temporarily saved in the memories of the input unit.

For the case of $D_x$, only $2w/s$ intermediate partial-addition $D_x$ values for one row of left and right sub-cells are stored in the first dual-port memory of the input unit. The final accumulation results for left and right part of each sub-cell are then loaded into the corresponding registers (i.e., left part sum and right part sum in Fig. 4) for subtraction to obtain $D_x$ according to Eq. (1). While processing the last pixel line of the sub-cell row, the $D_x$ responses of this sub-cell row can be sent to the output unit for constructing the cell responses according to Eq. (3) after every $s$ clock cycles. When the processing of the current sub-cell row is completed, the storage space of the first dual-port memory can be reused for next sub-cell row. Therefore, the first dual-port memory needs $2w/s$ storage locations for intermediate storage of $D_x$ in the case of sub-cells with $s \times s$ pixels and a $w \times h$-pixel image. Consequently, all pixels can be scanned successively by the image sensor and can then be used immediately for constructing the corresponding sub-cell response $D_x$, as illustrated in Fig. 4, according to the pixel location in the input image. In other words, no duplicate input of the same pixel is needed so that the computational efficiency is substantially improved.

In the case of $D_y$, calculation, the operating principle to create the sub-cell response $D_y$ is similar to the $D_x$ calculator. By comparison, the $D_y$ calculator uses an adder/subtractor circuit for obtaining the $D_y$ values. At first, the adder/subtractor sums up the pixels in the upper-half rectangle of each sub-cell and then subtracts the pixels in the lower-half rectangle of this sub-cell, as illustrated in Fig. 3. It is not necessary to store the upper- and lower-half of each sub-cell separately in the second dual-port memory. Therefore, the structure of the $D_y$ calculator is simpler than that of the $D_x$ calculator. Only $w/s$ corresponding storage locations in the second dual-port memory for sub-cells have to be allocated for intermediate sub-cell summation results of $D_y$ according to Eq. (2), due to the orientation of $D_y$ and the line-wise input of pixels from the image sensor.

The two multiplexers in the input unit form an iterative structure to initialize the calculation for each sub-cell and to execute a recursive data loop for the two dual-port memories. After the completion of calculations in one sub-cell row, the storage space of the two dual-port memories in the input unit can be initialized and reused for the $D_x$ and $D_y$ calculation of the next sub-cell row. Consequently, the processing of input images with principle unlimited height $h$ is enabled on the basis of this relatively small amount of dual-port storage space, which significantly enhances the utilization ratio of on-chip hardware resources and increases the area efficiency.

Once the last relevant pixel of a sub-cell has been processed, the $D_x$ or $D_y$ result will be transferred instantaneously to the output unit for the accumulation of the local feature vector $v_{cell}$ of the current cell with $e \times e$ pixels. The third and fourth dual-port memories have to store $w/e$
intermediate cell-summation results of $\sum D_x$, $\sum |D_x|$ and $\sum D_y$, $\sum |D_y|$, respectively. The function of the other two multiplexers in the output unit is similar to the ones in the input unit, but the main usage is for cell accumulations. The storage space of the two dual-port memories in the output unit can also be overwritten afterwards during processing of the next cell row, once the processing for the previous cell row is completed. According to the analysis above, image width, sub-cell size, and cell size, jointly determine the sizes of the four dual-port memories, the processing cost for extraction of the cell-based FVs, and the recognition delay for a SW.

In fact, the height $h$ of the input image is unlimited since the storage space of these four memories is reused for each sub-cell or cell row of the image. The cyclic utilization of the storage space in the four dual-port memories makes the proposed circuit sufficient for achieving image-size flexibility and memory-usage efficiency.

The counter group in the control unit is used to create corresponding enable signals and in cooperation with the logic gates for instant stepwise execution of the CFEPPP architecture. Particularly, with respect to the input parameter of image width $w$, three bit-alterable counters including one $[\log_2(w/2)]$-bit counter, one $[\log_2(w/4)]$-bit counter and one $[\log_2(w/8)]$-bit counter, are introduced for enabling flexibility of image resolution. As explained before, the height $h$ of the input image is not a constraint for the developed architecture. Finally, as illustrated in Fig. 1, the local cell-feature vector $v_{cell}$ is constructed in a serial manner on the cell layer and then outputted to the window layer for forming the complete 1680-dimensional Haar-like FVs of related scan windows (SWs).

### 2.3 Cell-based parallelized construction of multiple FVs for related SWs

In order to enhance the computational capability of window-feature extraction and object recognition, all SWs overlapping the same cell are partially processed in parallel on the window layer. Each cell-based local feature vector $v_{cell}$ is scanned and extracted only once at the cell layer without complex re-computations, and then reutilized for all corresponding SWs. This cell-based parallelized SW scheme leads to great improvement in computational-cost reduction in comparison to previous designs.

As an alternative hardware solution to traditional window-scanning strategies, a cell-based perspective for window search and construction is used for confirming the number of overlapping SWs and the RRRT value of the current cell in each of these SWs (see Sect. 2.1). For this purpose, the sequence of overlapping SWs and the corresponding inner position of the current cell in each SW have to be determined.

In case of a SW with $16$ (width) $\times 8$ (height) cells, a window-level Haar-like FV with $k = 4 \cdot \sum_{i=128}^{1680} R(\text{cell}) = 1680$ dimensions is constructed from 128 reused four-dimensional cell-based feature vectors $v_{cell} (\sum D_x, \sum |D_x|, \sum D_y, \sum |D_y|)$, which are generated by the circuit shown as Fig. 3. Similar to the effect of sub-cell size and cell size, larger window size also leads to a slower processing speed for extracting the window-level FV. Since only $w/e$ cell-based FVs have to be temporarily stored before constructing the FV of the first window, the memory requirement is again much lower than in the conventional integral-image-based implementations.

For the parallelized FV extraction of all SWs related to the current cell $C[c, r]$, indices of column ($c$) and row ($r$) are monitored in real time by a location analyzer. The initial index $i$ of all SWs related to current cell $C[c, r]$, which represents the first window $W_i$ containing the current cell $C[c, r]$, can be derived as illustrated in the lower left part of Fig. 5.

The maximal SW number $N$ in horizontal direction confines the initial index $i$ of each row of SWs to a multiple of $N$. Furthermore, the multiplication parameters in horizontal (hor) and vertical (ver) directions, which illustrate the number of SWs containing $C[c, r]$ in these directions, can also be confirmed through look-up tables (Tables I and II) based on the cell’s column ($c$) and row ($r$) indices. Therefore, the initial index $i$, the maximal SW number $N$ in the horizontal direction and the multiplication parameters hor and ver,
define the indices of all overlapping SWs containing the current cell $C[c, r]$ as specified in Eq. (4), where \( w_w \) represents the window width.

\[
\{W_{\text{index}}\} = \{i + j + l \cdot N, j \in [0, \text{hor} - 1], l \in [0, \text{ver} - 1], N = (w - w_w)/16 + 1\} \tag{4}
\]

The window number calculator in Fig. 5 operates according to Eq. (4). Each SW, declared by \( W_{\text{index}} \), is transferred to the window search unit. The window search unit is applied for sorting all SWs that contain the current cell \( C[c, r] \). Consequently, the local four-dimensional feature vector of the current cell can be invoked correctly according to the RRRT value, which depends on the cell position in each SW. Once the \( C[c, r] \) position within the current window is confirmed, the RRRT value of the current cell’s local four-dimensional feature vector in the declared current windows \( W_{\text{index}} \) is determined. The hardware implementation for confirming the RRRT value is realized by a decoder and a separate cache space (storage space) for updating the cell locations within the SW. The RRRT value of a cell in a SW varies with block size. There are only three cases (i.e., 1, 2, or 4) of RRRT values for the current cell when a 2 × 2-cell block is sliding by one cell with raster manner in a 16 × 8-cell SW (see Fig. 2).

The reusing time of each individual cell in a SW is summarized by the RRRT values in a table for simpler hardware operation. In this way, according to the positional parameters of the current cell \( C[c, r] \), the order of all related SWs and the corresponding RRRT values are confirmed and outputted for the construction of 1680-dimensional Haar-like SURF descriptor FVs.

Given an input image with \( w \times h \) pixels, the maximal SW number \( N \) in the horizontal direction is equal to \((w - w_w)/16 + 1\), and there are \( M = (h - h_w)/16 + 1 \) SWs in the vertical direction. All images with a width \( w \) larger than the SW width \( w_w \) (e.g., 128 pixels) and a height \( h \) larger than the SW height \( h_w \) (e.g., 64 pixels) can be processed. Consequently, the total number of SWs for images with \( w \times h \) pixels is calculated as \( N \times M \). The feature dimensionality of a Haar-like descriptor depends on the SW size, which quantifies the similarity to references for detection of an object in the image.

The cell-based recognition process for each SW is executed by the nearest-neighbor-search (NNS) classifier, which has been implemented by the pipeline with parallel p-word input (PPPI) architecture\(^{36-39}\) with an improved cache structure. The cache mechanism completes shortly after the construction of the \( k \)-dimensional FV \( \{d_1, d_2, \ldots, d_k\} \) from all required local cell-based feature vectors. The dimensionality \( k \) for the paradigm of Haar-like feature types is 1680 in this work, which is applied in the SURF descriptor scheme for one SW. Consequently, the global minimum squared Euclidean distances (SEDs) between the newly constructed FV of each declared SW and the reference FVs are calculated by the NNS classifier, and then the nearest reference is identified as the classification result (i.e., the NNS winner) for this SW.

### 3. Results and application discussion

#### 3.1 Experimental implementation results

A VLSI prototype of the Haar-like feature-extraction coprocessor was implemented in 180 nm CMOS technology with 1.76 mm\(^2\) core area, as shown in the photomicrograph of Fig. 6. The power consumption of this coprocessor is 4.78 mW at 1.8 V supply voltage and 12.5 MHz frequency, which is sufficient for processing 30 fps VGA (640 × 480 pixels) images from general image sensors. The lower working frequency leads to lower power dissipation according to different speed requirements of various applications. Figure 7 demonstrates the power consumptions over a range of supply voltages and clock frequencies. Chip operation at up to 120 MHz is verified, indicating 325 fps processing capability for VGA-size videos.
Due to the flexibility in image resolution, the prototype chip can deal with a maximum width $w$ of 1024 pixels and an unlimited height $h$. Specifically, a sub-cell with $4 \times 4$ pixels and a cell with $2 \times 2$ sub-cells are defined to capture sufficient details of the target object in this work. Therefore, the first dual-port memory only consumes $w/2 = 512$ words, and the storage capacity of the second dual-port memory only requires $w/4 = 256$ words. The word precision for both the first and the second dual-port memory is 16-bit, which is sufficient for providing accurate accumulation of eight 8-bit gray pixels in a half rectangle of each sub-cell. Consequently, only $w/8 = 128$ words for temporary summations of the two 16-bit horizontal cell-FV dimensions (i.e., $\sum D_x$ and $\sum |D_x|$) and the two 16-bit vertical cell-FV dimensions (i.e., $\sum D_y$ and $\sum |D_y|$) have to be stored in the third or fourth dual-port memory, respectively.

### 3.2 Performance comparison

The detailed comparison to above previous works$^{22,23,40-42}$ is shown in Table III, verifying that the architecture proposed in this work can extract all 851 1680-dimensional Haar-like feature vectors in a VGA frame with a shorter processing time of 3.072 ms (i.e., 325 fps frame rate), low power dissipation of 43.45 mW at 120 MHz with 1.8 V supply voltage and smaller memory size of only 96 kbyte (i.e., 12 kB), even though the used CMOS technology is much less advanced.

A simplified SURF ASIC implementation based on an improved integral image generation was proposed in Ref. 22. The large required memory for storing integral images of frames still occupied the largest area of the chip leading to more power consumption. The hardware-oriented SURF accelerator in Ref. 23 consumed less power, because of its lower operation frequency and the much more advanced CMOS technology. In addition to the 56 kbyte FIFO for delaying the image in the descriptor, multiple FIFOs with different sizes for Gaussian box filters, a two-dimensional image integrator, and a three-dimensional local maxima detector, etc., are necessary for temporary data storage in Ref. 23, resulting in a lower frame rate. In Ref. 40, a semantic analysis SoC (SASoC) was developed to accelerate Haar-like feature extraction and machine learning, giving retrieval results with a frame rate of 156 fps in 160 × 120 pixel resolution and different computational times, depended on the applications for video processing and the machine-learning algorithms. Reference 41 reported a detect-and-track hardware-observation system to recognize objects based on Haar-like features.

AdaBoost classifiers composed of cascaded stages are applied to analyze the windows for object detection. Comparison between weak-classifier thresholds of the AdaBoost stages and the Haar-like-feature values must be taken into account in each stage to accumulate the weighting values. Therefore, the bandwidth for memory access and communication becomes a bottleneck of Ref. 41 since all classifiers are accessed frequently. A register-array-based architecture was implemented in Ref. 42. Although it employed the power-optimization techniques of signal-domain gating for integral-image extraction and update, the scheme applied in Ref. 42 needs additional storage capacity for the integral image. The better performance in our proposed work mainly results from lower computational cost and storage requirements.

### 3.3 Application discussion

The reported hardware architecture is capable of handling different applications for object detection and recognition. For the case of vehicle detection, we have trained a classifier employing 1680-dimensional Haar-like features implemented with the proposed CFEEPP architecture.

We employed the car recognition as an example to evaluate the efficiency of our proposed algorithm by software-based simulation as illustrated in Fig. 8. A set of positive samples (i.e., cars) from the car detection dataset UIUC$^{53,44}$ and negative samples (i.e., non-cars) from INRIA

| Table III. Performance comparison to previous works. |
|---------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Technology                      | Ref. 22         | Ref. 23         | Ref. 40         | Ref. 41         | Ref. 42         | This work       |
| Die area (mm$^2$)               | 65 nm           | 28 nm           | 90 nm           | 40 nm           | 45 nm           | 180 nm          |
| Design target                   | Feature extraction | Feature extraction | Object recognition | Object recognition | Object recognition | Feature extraction |
| Extraction scope                | Entire frame    | Entire frame    | Entire frame    | ROI only        | Entire frame    | Entire frame    |
| Image resolution                | 1920 × 1080     | 640 × 480       | 160 × 120       | 1280 × 960      | 640 × 480       | 1024 × ∞        |
| Feature type                    | SURF hardware   | SURF hardware   | Haar-like hardware | Haar-like hardware | Haar-like hardware | Haar-like hardware |
| Frequency (MHz)                 | 200             | 27              | 200             | 220             | 200             | 400             |
| Core/IO voltage (V)             | 1.2/2.5         | 0.72/—          | 0.72/—          | 1.8/3.3         | 1.8/3.3         |
| Power (mW)                      | 220             | 2.8             | 1214 (peak)     | 69              | 45.2            | 43.45           |
| Maximum frame rate (fps)        | 57              | 30              | Depend on applications | 300             | 70              | 325 (VGA)       |
| Memory usage (kbyte)            | 400             | >7              | 149             | 25.3            | 45              | 12              |
dataset45) are selected for verification in this work. A reasonable quantity of samples plays a significant role in establishing sufficient diversity to correctly detect the input data from the testing dataset.46) Therefore, 550 positive samples from the UIUC dataset and 12180 negative samples obtained by cropping non-car images from the INRIA dataset are resized to 128 × 64 pixels (16 × 8 cells) for training. Furthermore, 199 different positive samples from the UIUC dataset and 1812 different negative samples cropped from the INRIA dataset are used for the image set for testing.

For evaluating the recognition performance of our proposed algorithm, SEDs between the extracted FVs of the SWs and the reference vectors are calculated for nearest-neighbor-search (NNS) classification, as illustrated in Fig. 8. The receiver operating characteristic (ROC) curve in Fig. 9 is evaluated with the image set for testing. The ROC curve indicates that the proposed classifier’s detection accuracy increases with an increased quantity of testing samples.

The one-sided increase of true positive rate (TPR) or false positive rate (FPR) can also impact the validity of the vehicle-detection accuracy (ACC) with different quantities of testing samples according to the relationship (5). Here P is the number of all positive samples, N represents the number of non-car samples (negative samples), TP is the number of vehicle images detected correctly among all positive samples, TN is the number of correctly detected negative samples among all negative samples, and FP is the number of negative samples that are detected as positive samples. The specificity (SPC) is an associated variable of FPR (SPC = 1 − FPR).

\[
\text{TPR} = \frac{TP}{P} \times 100\%; \quad \text{FPR} = \frac{FP}{N} \times 100\%;
\]
\[
\text{SPC} = \frac{TN}{N} \times 100\%; \quad \text{ACC} = \frac{TP + TN}{P + N} \times 100\% \quad (5)
\]

Rather than using the entire training dataset as reference data for NNS, the k-means algorithm is further used to cluster the dataset into k groups. The centroids of these k groups are finally taken as the reference data for classification. In the case of vehicle detection, the integral-image strategy of previous software solutions47–49) attained an accuracy range from 85.7 to 100%. In our verification, we have achieved a maximal TPR of 97.49%, a SPC of 99.67% (i.e., FPR at 0.33%), and a comparably good accuracy (ACC) of 99.45%.

The Haar-like feature extraction coprocessor is particularly suitable for mobile applications, due to the low power dissipation and the low hardware resource requirements. In addition, the image-size flexibility of the proposed architecture can be controlled by the implemented bit-alterable counters for real-time processing of higher-resolution images than VGA, e.g., XGA (1024 × 768 pixels), at 120MHz operating frequency.

4. Conclusions

In this paper, a VLSI coprocessor for Haar-like feature extraction was implemented in 180nm CMOS technology, which is suitable for a large variety of different mobile applications, such as vehicle detection. A hardware architecture of “cell feature extraction by a pixel-based pipeline” (CFEPP) is proposed for online feature construction of the local four-dimensional cell-based feature-vector parts. The coprocessor immediately processes every input pixel without pre-storage of raw images and reuses on-chip memory space for local cell-based feature-vector calculation of different cell rows in the processed image, making it highly flexible for multi-size image processing with low memory requirements. The applied coprocessor architecture also shortens the critical path and the searching time of pixel positions, which reduces the redundant computing cost, resulting in the achievement of very fast real-time processing speed and high energy efficiency. Finally, comparably good accuracy is verified for the mobile-application case of vehicle detection.

Acknowledgments

The reported research work is supported by the Program for Leading Graduated Schools, Ministry of Education, Culture, Sports, Science and Technology. The VLSI-chip was fabricated through the chip fabrication program of VDEC, the University of Tokyo in collaboration with, Renesas Electronics, Synopsys, and Cadence.

9) A. Haar, Math. Ann. 69, 331 (1910) [in German].


45) N. Dalal, INRIA Human Dataset (2005) [http://pascal.inrialpes.fr/data/human/].


