# A Real-Time Motion-Feature-Extraction VLSI Employing Digital-Pixel-Sensor-Based Parallel Architecture

Hongbo Zhu, *Member, IEEE*, and Tadashi Shibata, *Life Member, IEEE*

***Abstract*— A very-large-scale integration capable of extracting motion features from moving images in real time has been developed employing row-parallel and pixel-parallel architectures based on the digital pixel sensor technology. Directional edge filtering of input images is carried out in row-parallel processing to minimize the chip real estate. To achieve a real-time response of the system, a fully pixel-parallel architecture has been explored in adaptive binarization of filtered images for essential feature extraction as well as in their temporal integration and derivative operations. As a result, self-speed-adaptive motion feature extraction has been established. The chip was designed and fabricated in a 65-nm CMOS technology and used to build an object detection system. Motion-sensitive target image localization was demonstrated as an illustrative example.**

***Index Terms*— Block-readout scheme, digital pixel sensor (DPS), motion feature extraction (MFE), parallel architecture, vision chip.**

## I. Introduction

REAL-TIME motion recognition is becoming increasingly important in various applications, such as automotive vehicle control, efficient human–computer interaction [1], video surveillance [2], remote gesture control [3], sign language interpretation [4], [5], surgery support, and so forth. In automotive vehicle control, for instance, it can be used for slip detection and in surgery support, monitoring the behavior of a beating heart would play a crucial role. For such applications, high-speed image processing employing high-frame-rate image sensors [about 1000 frames/s] can significantly provide many details of the motion, which is very useful for improving the speed and precision of the recognition. Many applications benefit much from the high-speed image processing (about 1000 frames/s), such as the system reported in [74] for counting objects flowing rapidly on the line, in [75] for microorganisms observation under microscope, and in [76] for object regrasping using a robot hand.

Manuscript received November 5, 2013; accepted March 7, 2014. Date of publication March 26, 2014; date of current version October 1, 2014. This work was supported by the VLSI Design and Education Center, for chip fabrication program, University of Tokyo, in collaboration with STARC, e-Shuttle Inc., Fujitsu Ltd., Synopsys, Inc., Cadence Design Systems, Inc., and Mentor Graphics, Inc. This paper was recommended by Associate Editor S.-Y. Chien.

H. Zhu is with the VLSI Design and Education Center, University of Tokyo, Tokyo, 113-0032, Japan (e-mail: zhu@vdec.u-tokyo.ac.jp).

T. Shibata is with the Japan Society of Applied Physics, Center for Innovative Integrated Systems, Tohoku University, Sendai 980-8577, Japan, and also with the Institute of Space and Astronautical Science, Japan Aerospace Exploration Agency, Tokyo 100-004, Japan (e-mail: shibata@jsap.or.jp).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2014.2313899

Therefore, motion recognition systems with a processing capability of 1000 frames/s have a lot of potential to be implemented. For building such high-speed systems, motion feature extraction (MFE), as an important and computationally heavy process in motion recognition [6], [7], is required to be processed in less than 1 ms.

The traditional software-based MFE algorithms are computationally very expensive, and building high-speed systems is quiet difficult even when state-of-the-art multicore general-purpose processors are utilized [8], [9]. By developing fine-tuned software exploring the single-instruction multiple-data operations on modern processors and/or graphics processing units for particular applications [10], [11], high hardware costs and large power consumptions are severely limiting such approaches to be used in portable devices, as well as in large-scale systems. As a result, high-speed and low-power architectures for MFE are now highly demanded.

A number of very-large-scale integration (VLSI) processors with parallel architectures have been developed for performance enhancement in image processing algorithms. Image data processing usually includes some computationally intensive low-level data processing [12], such as image filtering, which needs be performed repeatedly in every pixel site in the entire image, or at least in the region of interest. Processors in [13]–[16] improved the performance and power efficiency by developing parallel processing circuitries mainly for such low-level image processing tasks. Processors in [17]–[23] implemented fine-grain parallel processing architectures by investigating image processing tasks at all levels (low, middle, and high levels [12]) in some specific applications. Such processors have made real-time image processing systems feasible [14], [20]. Therefore, the parallel architecture is important for improving the performance of image data processing. However, all these processors are primarily targeting still image recognition, and not dealing with motion or action recognition problems. In addition, the delay caused by data transfer from image sensors to processors, which is usually carried out one pixel data per clock cycle, severely limits the efficiency of image data access, making such a system unsuitable for time-critical applications, such as automotive vehicle control. Furthermore, systems composed of multiple chips including image sensors, processors, and in certain cases, memories that buffer frame image data are still very power hungry. In this sense, building system-on-a-chip (SoC) image processing systems is a very promising approach to achieving better power efficiency and small latency.

Computational image sensors, which integrate both image sensors and processing circuits on the same chip, allow us to build image processing SoCs. Smart sensors with analog processing circuits have been developed for image filtering [24]–[27], edge extraction [28], and motion detection [28]–[32]. One of the drawbacks of the signal processing in analog domain is the lack of programmability, which makes such approaches not suitable for performing the middle-level and high-level processing in complex algorithms, such as those used in motion recognition. Furthermore, building analog processing circuits using more advanced process technologies requests much harder design efforts to guarantee the same level of accuracy in processing achieved with more conservative technologies. In [33] and [34], visual sensor SoCs employing parallel digital processing elements (PEs) are developed to process the data at all levels. These works demonstrated the real-time performance of the image processing SoC concept. However, regarding the access to analog image data in active pixel sensors for on-chip digital data processing, it is not very efficient in these works. In [33], since the image sensor and processing circuitries are not directly connected, the image data first need to be buffered in the on-chip memory for the following computation. In [34], although the image sensor and row-parallel PEs are directly connected, the processing circuits need to wait for 256 clock cycles every time they access the digitized image data in a single row because of the analog-to-digital conversion (ADC). Two important issues arise in the direct connection between the image sensor array and digital processing circuitries. One is the ADC of the captured image and the storage of the digitized image data. The other is the scheme for efficient digital image data transfer from the storage to processing circuitries, in which multiple-row data access is essential for kernel processing like image filtering.

The digital pixel sensor (DPS), with in-pixel ADC and digital memories [35], has good compatibility with the digital processing circuitries. As a result, complex algorithms such as on-chip image compression can be implemented [36]–[39] with low-power consumption. In particular, the VLSI implementation of a spatial temporal movie compression algorithm [39] has demonstrated that it is very effective for the DPS-based interframe processing, a fundamental processing employed in most of the motion analysis algorithms. Therefore, developing DPS-based VLSIs with direct connection between the image sensor array and digital processing circuitries provides a promising solution to implementing high-speed image processing systems. Another advantage of using DPS-based VLSIs is the employment of digital circuits not only in each pixel data control, but also in the image processing circuit modules that are built being separated from the image sensor array. Digital circuits are easy to scale down with advanced process technologies, and therefore the functionality of the chip could be easily enhanced.

In the development of application specific VLSIs, employing VLSI implementation-friendly algorithms is quite essential because it allows us to build the system in compact and power-efficient parallel circuitries. However, the most common approaches are to translate already available software tools [40]–[42] into VLSI hardware. This would not be a very efficient way to take. Intelligent image processing algorithms, in particular those for motion recognition applications, are usually very complex, and how to make them VLSI-implementation friendly has not yet been explored very extensively, resulting in a large gap between image processing algorithms and their VLSI implementation. Therefore, it is of paramount importance to consider the VLSI-implementation friendliness from the very early stages of algorithm development.

Recently, many VLSI implementation-friendly algorithms have been proposed [43]–[46]. For example, in [43], a human posture recognition algorithm utilizing specific smart sensors was developed, which can reduce the computation complexity by five times than the algorithms without consideration on the hardware friendliness. In [44], a neural feature extraction algorithm with a particularly developed scalable architecture for implantable neural signal recording was proposed, and a 95% reduction of hardware resource was reported. Such VLSI implementation-friendly algorithms with optimized VLSI architecture provide an efficient approach for developing high-speed and low-power image processing systems.

Mimicking the brain functions provides us with a lot of inspiration in algorithm development because the brain has excellent ability in a variety of image processing tasks. According to the biological discoveries, directional edges are detected in the primary visual cortex as important clues for visual recognition [47], [48]. The output signals from the primary visual cortex are transferred to two different processing pathways: 1) the ventral stream for pattern recognition and 2) the dorsal stream for motion recognition. From the inspiration of such biological principles, VLSI-implementation friendly directional-edge-based image feature representation algorithms were developed in [49] and [50]. Because of such algorithms, VLSI chips for directional edge detection [51]–[53] and those for image recognition [51], [54]–[56] were developed to achieve real-time performance with low power consumption. As a result, high-speed image processing systems have been realized. For example, a still image recognition system with a processing time of only 906 $\mu$s per frame is reported in [57] using the processor in [52]. Besides the still image analysis, a directional-edge-based object tracking algorithm is also proposed in [58] with its FPGA implementation in [59], in which a speed of 150 frames/s was experimentally demonstrated, and a speed of 900 frames/s was expected if the algorithm was directly implemented on VLSI with an on-chip image sensor. The succeeding works [60]–[63] further developed edge-based motion recognition algorithms in which the self-speed-adaptive MFE processing is included as one of the most crucial steps. Although VLSI chips for post-MFE optical flow detection [64], [65] and pattern recognition [55], [56], [66]–[68] have been implemented, a VLSI dedicated to high-speed and low-power MFE has not yet been developed.

In this paper, a VLSI capable of extracting motion features from moving images in real-time has been developed employing fine-grain row-parallel and pixel-parallel architectures based on the DPS technology. The block-readout scheme [69] is combined with row-parallel bit-serial processing circuitries

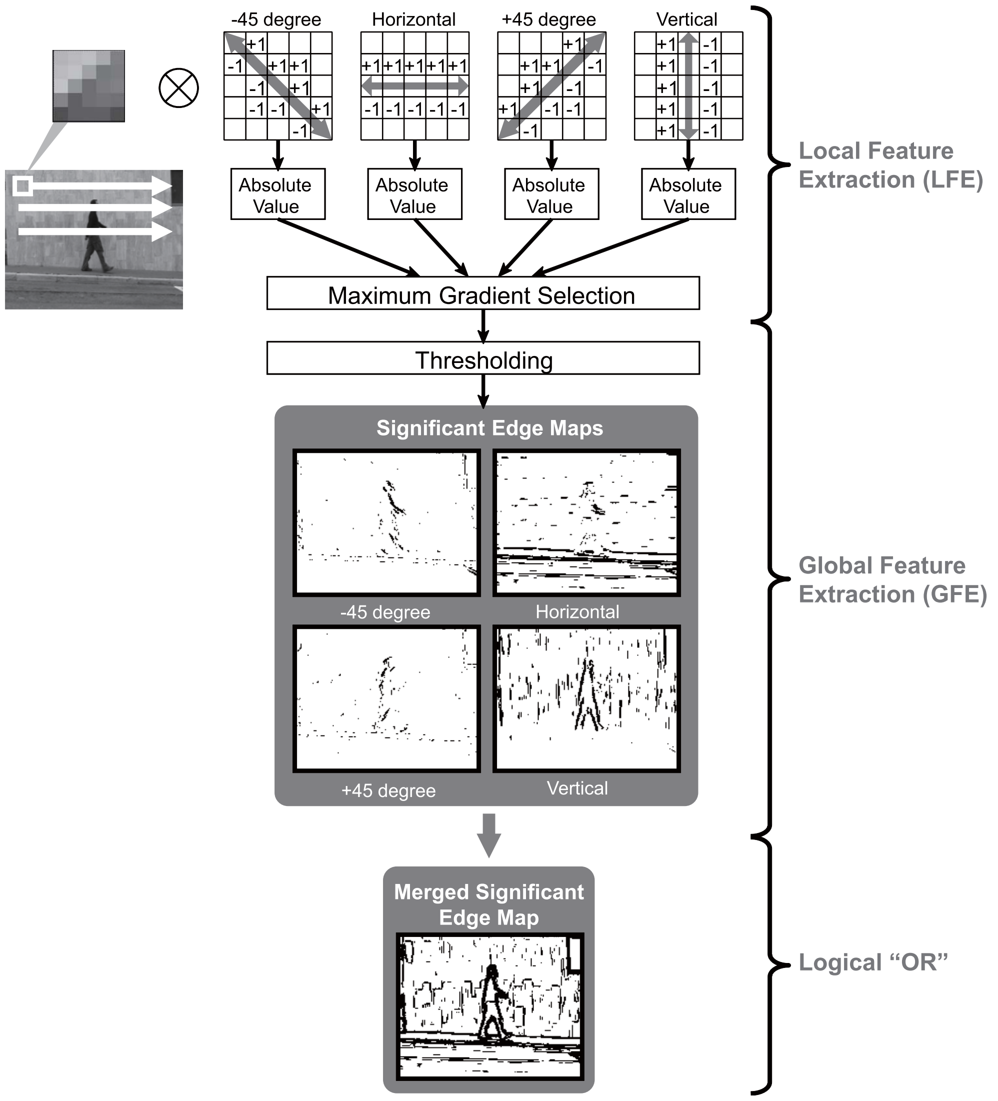


Fig. 1. MSEM generation including LFE, GFE, and logical OR.

to perform 5 × 5-pixel kernel convolution of directional edge filtering immediately following the ADC. To achieve a real-time response of the system for moving image processing, a fully pixel-parallel architecture has been explored in adaptive binarization of filtered images for essential feature extraction as well as in their temporal integration and derivative operations. This allows us to operate the system in a self-speed-adaptive manner. The chip was developed in a 65-nm CMOS process to implement the parallel on-chip digital circuits for DPS with the resolution of 100 × 100. The measurement results show that this chip can perform MFE within 0.9 ms after capturing the image when running at 20 MHz at a power consumption of 9 mW. The directional-edge-based features generated by this chip can be used for motion recognition [60]–[63] as well as for still image recognition [49], [50] and object tracking [58], [59]. An object detecting system is also developed using the fabricated chip. With the motion features, the system can localize the images of target objects only when they are in motion.

The remainder of this paper is organized as follows. Section II explains the MFE algorithm that we developed for VLSI implementation in this paper. Section III describes the VLSI architecture. Section IV presents the experimental results obtained from the prototype chip, and discussions are given in Section V. Finally, Section VI concludes this paper.

## II. MFE Algorithm

This section explains the MFE algorithm that generates self-speed-adaptive motion features from video sequences. This algorithm has two steps: the first step that extracts static features in every frame; the second step that extracts motion features only from the frames where significant amount of motion is detected. In the first step, the merged significant edge map (MSEM) is generated to represent static features, as described in Section II-A. In the second step, the self-speed-adaptive motion feature is generated, as described in Section II-B.

### A. MSEM Generation

Fig. 1 shows the MSEM generation. First, four significant edge maps (SEMs) are generated from an input image based on

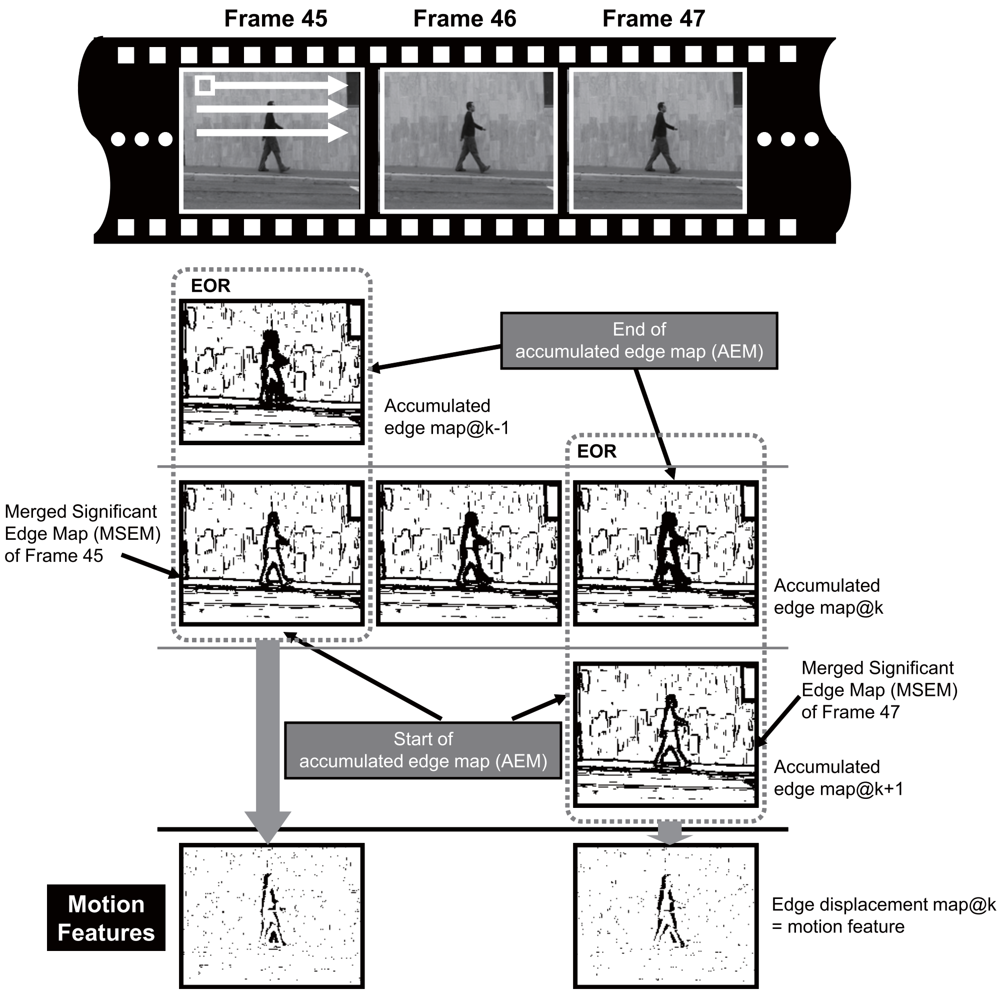

Frame 45
Frame 46
Frame 47
EOR
End of accumulated edge map (AEM)
Accumulated edge map@k-1
EOR
Merged Significant Edge Map (MSEM) of Frame 45
Accumulated edge map@k
Merged Significant Edge Map (MSEM) of Frame 47
Start of accumulated edge map (AEM)
Accumulated edge map@k+1
Motion Features
Edge displacement map@k = motion feature

Fig. 2. MFE employing AEM. Movie clip from Weizmann human action dataset [70].

the four-directional edge filtering (horizontal, $+45°$, vertical, and $-45°$), as in [52]. Then, the four SEMs are merged into a single edge map by taking logical OR, which we call an MSEM. The four-directional edge filtering is composed of two main processes: 1) the local feature extraction (LFE) and 2) the global feature extraction (GFE). Fig. 1 shows these two processes where LFE is shown at the top and GFE is shown in the middle. In LFE, first, convolutions between a $5 \times 5$-pixel local image centered at each pixel site and four $5 \times 5$-pixel filtering kernels (one for each direction) are calculated. Then, based on the four convolution results, the maximum gradient value is selected to determine the edge direction at the pixel site, and this convolution value is preserved. According to the edge direction at each pixel site, an edge flag bit 1 is set at the corresponding location in the respective binary edge map. For example, if the direction of a pixel is horizontal, then this pixel has an edge flag in the horizontal edge map and no edge flag in the $+45°$, vertical, or $-45°$ edge map. Such a process is repeated for every pixel, making the entire image seamlessly scanned by the four filtering kernels. Four-directional kernels produce four binary edge maps, which we call local features. Since each pixel has an edge flag in one of the four binary edge maps after LFE, local features contain a lot of redundant information. Therefore, GFE is performed after LFE to retain only salient features. This is done by selecting only a predetermined percentage (which we call the edge detecting threshold) of significant edge flags out of all pixels that have larger gradient values than the rest. Such a selection is possible because convolution values obtained from every pixel site are all preserved. The four edge maps having only the significant flags after the selection very well represent global features and we call them the SEMs. Fig. 1 shows the SEMs in four directions by setting the edge detecting threshold at 20%. Only the salient features in the original image are highlighted.

The SEMs are used for both static image recognition and motion analysis. For static image recognition, a feature vector representation algorithm called projected principal-edge distribution [49] or averaged principal-edge distribution (APED) [50] is employed to transform the four-directional SEMs into a single 64-D feature vector. For motion analysis, the four-directional SEMs are merged into one MSEM [60].

## *B. MFE*

The motion features are extracted from the sequence of MSEMs. The length of the sequence, defined as the number of frames between the starting frame and the end frame, is very important for extracting good motion features. If it is too short, pixel motion cannot be detected. If it is too long, the accuracy of local motion detection degrades [60]. Therefore, an optimum length of the MSEM sequence must be determined adaptively according to the speed of moving objects. To extract the motion features in a self-speed-adaptive manner using MSEMs, an accumulated edge map (AEM) is first produced, and then need to generate an edge displacement map, which is the motion feature to be used for motion analysis.

Fig. 2 shows the process of MFE in detail. At the beginning, an MSEM is generated from the starting frame (Frame 45) with a predetermined edge detecting threshold and recorded directly as the $k$th AEM. From the next frame (Frame 46), the MSEM is generated with the same edge detecting threshold, and logical OR is taken between this MSEM and the latest AEM to update the AEM for this frame. Then, the edge count in the new AEM is calculated and compared with the motion detection threshold (this threshold is also given as the percentage to the total number of pixels). If the edge count is smaller than the threshold value, it indicates that there has not been significant motion in the scene. Then, the MSEM for the following frame (Frame 47) is extracted and merged with the AEM by taking OR again. Such an accumulation process continues until the edge count of the AEM becomes equal or larger than the motion detection threshold, which indicates that there has been significant motion since the start of the accumulation. In the example of Fig. 2, Frame 47 is the frame in which significant motion is detected. Then, the difference between the AEM at Frame 47 and the MSEM of Frame 47 is calculated by taking exclusive OR between them. The edge map produced in this manner is called the edge displacement map, in which only the edge flags due to the motion having occurred in the MSEM sequence are accumulated and remaining. This edge displacement map is utilized as the motion features for motion analysis.

Then, the same procedure is restarted from the end frame of the previous MSEM sequence, namely, the MSEM of Frame 47 is used as the new AEM at $k + 1$, and accumulation is continued until the next edge displacement map is obtained. Since the number of edge flags remaining in the edge displacement map is set to a constant value, [motion detecting threshold (%)$-$edge detecting threshold (%)] $\times$ total number of pixels, the length of the MSEM sequence for generating an edge displacement map is automatically adjusted as approximately corresponding to the equal amount of motion occurring in the movie.

## III. VLSI Architecture

### A. Chip Configuration

Fig. 3 shows the chip configuration. This chip extracts image features, in particular the motion features from moving images, which is the most computationally expensive processing. A $100 \times 100$ DPS array captures images, and the LFE module carries out $5 \times 5$-pixel kernel convolutions of four-directional edge filtering in row-parallel processing. The SEM generation and the MFE are both conducted by the MFE module in fully pixel-parallel processing.

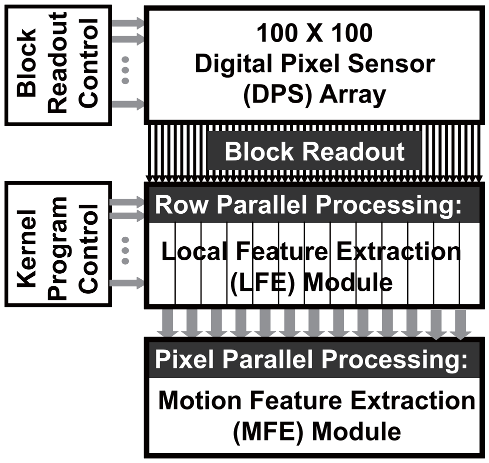

Block Readout Control
100 X 100 Digital Pixel Sensor (DPS) Array
Block Readout
Kernel Program Control
Row Parallel Processing:
Local Feature Extraction (LFE) Module
Pixel Parallel Processing:
Motion Feature Extraction (MFE) Module

Fig. 3. Chip configuration for extracting both static and motion features.

### B. DPS Pixel

Fig. 4(a) shows the basic concept of the DPS [35]. The pixel unit consists of a photodiode, a reset transistor, a shutter transistor, a sampling metal–oxide–semiconductor capacitor, an analog comparator, 8-bit DRAM cells, and a readout buffer. In the global reset process, both the reset transistor and the shutter transistor are turned ON, which forces the voltage $V$ at the MOS capacitor to $V_{\text{reset}}$ in all pixels. After this, the reset transistor is turned OFF while the shutter transistor is ON, then the decrease in the MOS capacitor voltage yields the light intensity at each pixel. After a certain period of time (photo integration time), all shutter transistors are turned OFF simultaneously and the light intensity in each pixel is converted into the analog voltage signal $V$. Then, an analog ramp signal and 8-bit digital ramp codes that are synchronized to each other are provided to all pixel units for ADC. The analog comparator in each pixel compares the voltage $V$ with the analog ramp. At the beginning of ADC, the analog ramp is below $V$, and the comparator output (write enable) is 1. At the timing when the analog ramp voltage exceeds $V$, write enable changes to 0, and the digital code corresponding to $V$ is stored in the memory, thus the ADC of the pixel data is accomplished. Since this operation is conducted simultaneously at all pixels, massively parallel ADC is achieved. Different from the DPS pixel developed in [35], each bit of memory has a separated control signal, as shown in Fig. 4(b). The 8-bit control signal, together with the readout buffer, select which bit in the pixel to read out. Such a design allows us to perform the block-readout scheme [69] of DPS, as described in the following.

### C. Block-Readout Scheme

Fig. 5(a) shows the block-readout scheme [69] for row-parallel processing. To seamlessly scan the entire pixel array with $5 \times 5$-pixel-size filtering kernels, every four rows are grouped together and controlled by the same enable signal generated by block-readout control circuits. In one read cycle, the same bits of image data in pixels from two consecutive groups of rows are read out. For example, in Fig. 5(a), the image data in eight rows (indicated in light gray which contain

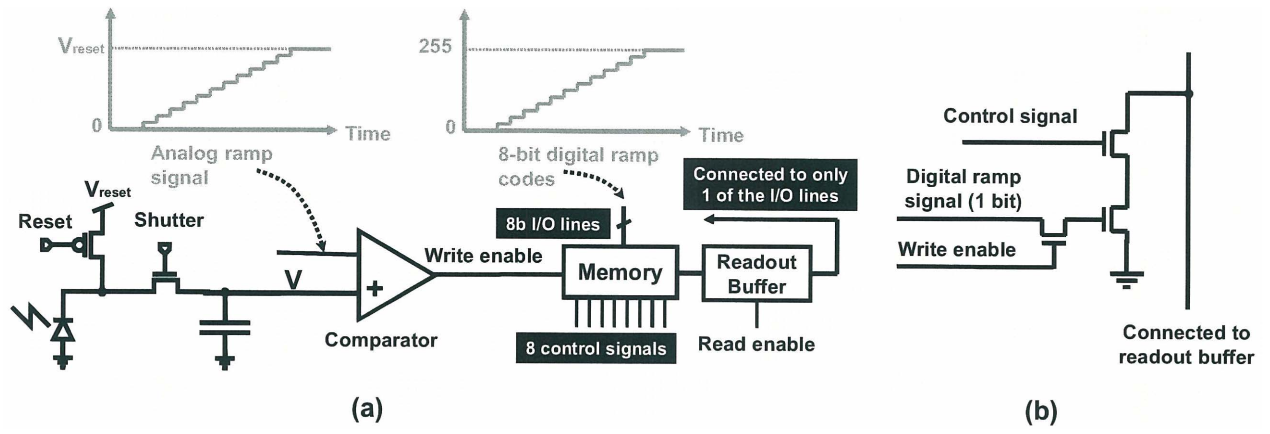


Fig. 4. (a) Circuits in one pixel and (b) 1-bit pixel memory circuit.

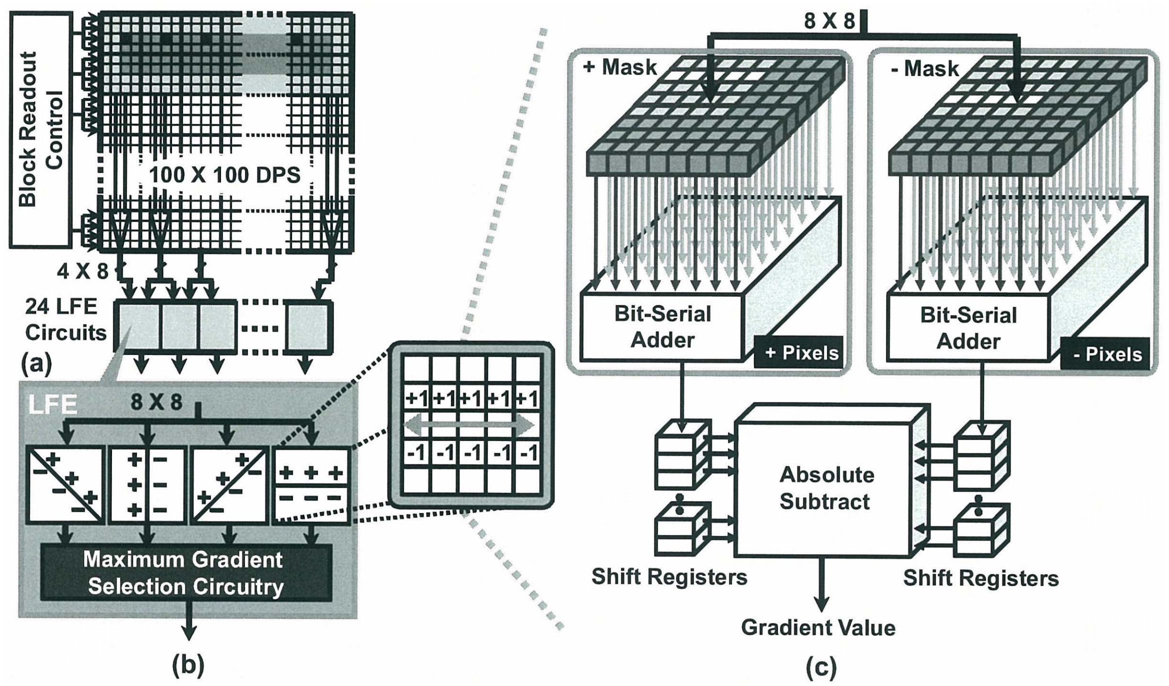


Fig. 5. (a) Block-readout scheme of DPS, (b) LFE, and (c) horizontal edge filter.

100 × 8 pixels) are simultaneously read out to 24 LFE circuits bit serially. Each LFE circuitry receives two 4 × 8 data blocks from the left and right, which form an 8 × 8 data block. The data in light gray are sufficient to perform the 5 × 5 kernel calculations and the maximum gradient selection in the LFE at all dark gray pixel sites (96 × 4 pixel sites) immediately with no extra buffer memories. However, to keep the LFE module rational in the interconnection and chip areas, the kernel calculation should be performed in a row-parallel manner, and 24 groups of LFE circuits are employed in this design. Therefore, for example, the black pixel sites (24 pixel sites in total) are processed by these LFE circuits simultaneously. The edge direction and the maximum gradient value are generated in each LFE, as shown in Fig. 5(b).

Fig. 5(c) shows the horizontal edge filter in detail. In this filter, 8 × 8-pixel data are input to two 64-input adders via two 8 × 8 masking circuitries, allowing only necessary data to go through (indicated in white). The absolute difference of the adder outputs yields the gradient value. By simultaneously shifting the data go-through locations (masking patterns) in all masks for 16 times, kernel convolutions are completed for all dark gray pixel sites in Fig. 5(a). Thus, LFE at 96 × 96 pixel sites is accomplished very efficiently without any extra buffer memories. Here, the masking patterns are generated in the kernel program control circuits (Fig. 3) using shift register arrays and broadcast to all groups of LFE circuits. This LFE architecture was first implemented in a 0.18-$\mu$m technology in [52] for still image recognition with a 68 × 68 DPS array. The following MFE function has been developed for the first time in this paper, and all digital circuits are implemented in a 65-nm technology.

## D. MFE Module

Fig. 6 shows the MFE module. The array of memory columns at the top stores the filtered image data provided

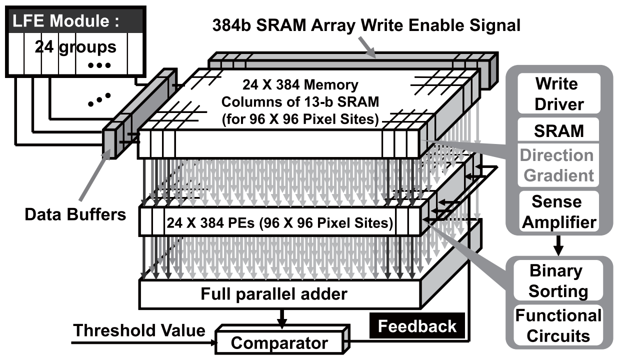


Fig. 6. Pixel-parallel processing architecture of MFE module.

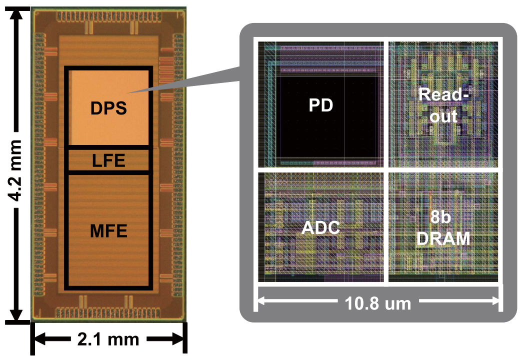


Fig. 8. Chip photomicrograph and circuit layout of one pixel.

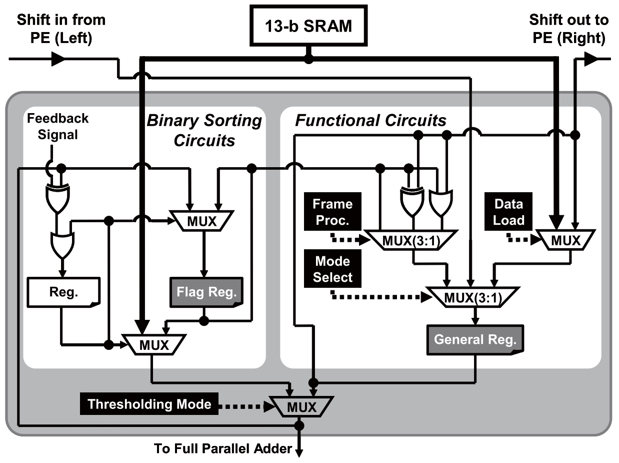


Fig. 7. PE composed of binary sorting circuits and functional circuits.

from the LFE module. Each memory column has a 13-bit static RAM (SRAM): 2 bits for the edge direction (maximum gradient direction) and 11 bits for the gradient value. The array of 24 $\times$ 384 memory columns stores all the filtering results obtained from the 96 $\times$ 96 pixel sites in the 100 $\times$ 100 DPS array. A common signal is provided as the clock for timing control in pixel-parallel SRAM data access. The memory array, the array of PEs, and the full parallel adder are fully connected. The adder's output is fed back to the PE array via the comparator. With this configuration, an efficient binary sorting algorithm, the rank ordering [71] of all 9216 gradient values, is accomplished in only 11 comparison cycles, namely, a predetermined number of edge flags (like 20% of all pixel sites, for instance) having larger gradient values can be selected and retained in the edge map. In this way, the MSEM (Fig. 1) is obtained. When combined with the 2-bit direction data, this MSEM represents the four SEMs.

Fig. 7 illustrates the PE configuration composed of two main parts: 1) the binary sorting circuits for MSEM generation and 2) the functional circuits for MFE and data output. When the LFE for all pixel sites finishes, the results have been already recorded on the memory array. Then, the Thresholding Mode signal selects the output signal from the binary sorting circuits to be transferred to the full parallel adder. According to the rank-order filtering structure, the MSEM can be extracted efficiently [52]. After the binary sorting, the MSEM datum of each pixel site is always preserved in the Flag Reg. of the binary sorting circuits. The General Reg. preserves the AEM datum. Controlled by the Frame Proc. signal and Mode Select signal, the MSEM datum in the Flag Reg. can be either copied directly to, or ORed/EORed with the AEM datum. Controlled by the Thresholding Mode signal, the whole frame of the binary AEM can be directly transmitted to the full parallel adder. The summation result is compared with the motion detecting value by the comparator in Fig. 6, and the comparison result (feedback signal) is used to indicate whether there has been significant motion in the scene. Controlled by the Data Load signal and Mode Select signal, the data in the 13-bit SRAM can be directly loaded to the General Reg. Controlled by the Mode Select signal, all General Regs. can be configured into 24 circulating shift registers to transfer the stored data outside the chip and restore these data after the transfer.

In MFE shown in Fig. 2, the MSEM in each frame is accumulated, i.e., OR is taken among consecutive frames. Thus, an AEM is generated. If there is no significant motion, the total number of edge flag bits in the AEM does not change much, but it does increase when there exists some motion. Such accumulation is continued until the total edge count in the AEM exceeds the motion detecting value. Then, the edge displacement map is generated by taking EOR between the AEM data in the General Regs. and the present MSEM data in the Flag Regs. Therefore, the trajectory of moving parts is highlighted in a self-speed-adaptive manner. At the same time, the edge flags from stationary background are erased.

## IV. Experimental Result

### A. Chip Fabrication and Measurement Results

Fig. 8 shows the photomicrograph of a fabricated chip. The chip was designed in a standard 65-nm CMOS technology with the PD and ADC in each pixel circuitry using 0.18-$\mu$m transistors. Table I shows the specification of this chip. The maximum operation frequency of the present proof-of-concept chip was limited to only 20 MHz due to the clock delay in the interconnects of SRAM module, which was revealed by the postfabrication chip analysis. It is important to eliminate the polimide layer when designing CMOS image sensors using such advanced technologies. In this paper, since this

TABLE I
CHIP SPECIFICATION

| | |
|---|---|
| Technology | 65-nm 1P12M CMOS |
| Pixel Size | 10.8 $\mu$m $\times$ 10.8 $\mu$m |
| Transistors per Pixel | 46 |
| Fill Factor | 13.5% |
| Resolution | 100 $\times$ 100 |
| Total Transistors | 4.3M |
| Die Size | 4.2 mm $\times$ 2.1 mm |
| Core Area | 3.4 mm $\times$ 1.3 mm |
| Supply Voltage | 1.2 V |
| Clock Frequency | 20 MHz |
| Typical Processing Time | 0.9 ms/f @20 MHz |
| Typical Frame Rate | 40 fps(*) |
| Power Consumption (at 40 fps) | 9 mW |

(* The typical shutter time is 24ms because of the polymide layer.)

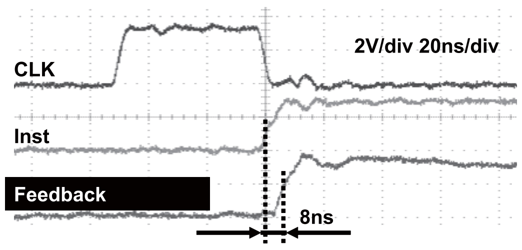


Fig. 9. Measured waveforms showing feedback signal from the chip when thresholding instruction is executed.

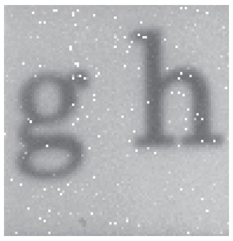


Fig. 10. Original photo taken by this VLSI.

layer was not eliminated, the whole chip except the regions of input/output pads was covered by the polimide, as can be seen in the photomicrograph. As a result, the sensitivity of the image sensor is decreased, and the object capturing range becomes small.

The waveforms in Fig. 9 show the feedback signal (Fig. 6) measured from the chip when the 9216 1-bit inputs addition using the full parallel adder was executed at the rising edge of the control signal Inst. It took about 8 ns for the full parallel adder to calculate the edge count in the AEM. As shown in the figure, a common signal is used as the clock signal for the MFE module.

Fig. 10 shows a photo taken by this chip without any processing. The photo integration time in this experiment was 24 ms. There are some noises in this photo because of the polymide layer. It should be noted that most of these noises can also be eliminated by the significant edge extraction processing circuitry on the chip without affecting the total performance.

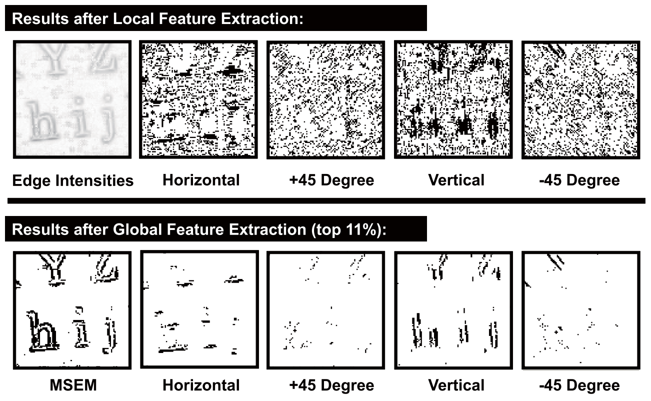


Fig. 11. Measurement results of static feature extraction.

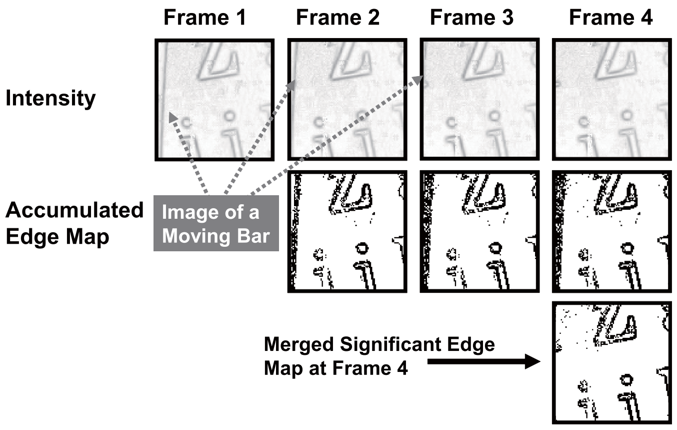


Fig. 12. EOR is taken between AEM and essential edge map at Frame 4 to highlight the moving bar trajectory.

Fig. 11 shows the measurement results when the chip was performing static image feature extraction. The outputs from the LFE module are shown at the top: gradient values (far left) and edge flag maps for four directions. The bottom figures show the MSEM and four-directional SEMs extracted by the MFE module. Top 11% (edge detecting threshold) of edge flags having larger gradient values were left in the MSEM. This is the result of sorting of 9216 gradient values stored in the memory column array.

Fig. 12 shows the measurement results demonstrating MFE from a moving image sequence (four sample frames are shown). The edge detecting threshold and motion detecting threshold were set to be 11% and 16.5%, respectively. A bar was going out from the left-hand side of the scene. In the AEM, the trajectory of the moving bar is accumulated on the left side of the frame. In Frame 4, the edge count in the AEM exceeds the motion detecting value, triggering the generation of a motion feature. If a derivative operation is taken at Frame 4, namely EOR is taken between the AEM and the MSEM at Frame 4, the trajectory of the moving bar can be highlighted as the motion feature.

Table II summarizes the necessary number of clock cycles per pixel for performing different functions employed in the MFE algorithm. The top two functions are performed by the LFE module in row parallel while the bottom two by the MFE module in pixel parallel.

TABLE II
PROCESSING SPEEDS IN CYCLES FOR DIFFERENT FUNCTIONS

| Function | Processing Time |
|---|---|
| 5 × 5 Convolution Filter | 0.28 cycle/pixel |
| Maximum Gradient Selection | 0.083 cycle/pixel |
| Sorting of 9216 Gradient Values | 0.0072 cycle/pixel |
| Inter-Frame Feature Processing | 0.00022 cycle/pixel |

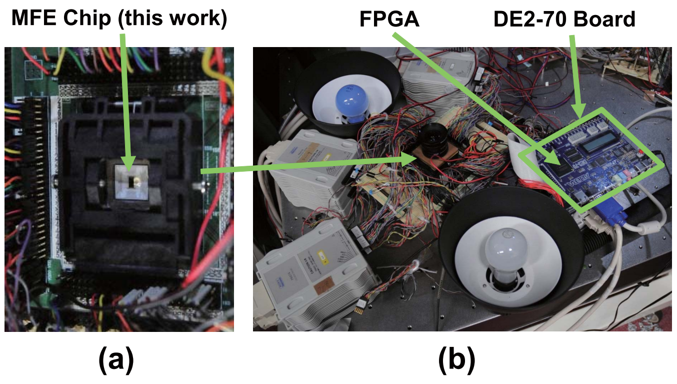


Fig. 13. Experimental setup of object recognition system built with fabricated chip.

TABLE III
SUMMARY OF FPGA PART IN THE SYSTEM

| | |
|---|---|
| Board | DE2-70 FPGA board by Terasic |
| Type of FPGA | Altera Cyclone II EP2C70F896C6 |
| Used logic elements | 16,443 |
| Used memories | 646,736 bits |
| Clock domains | 18 MHz in data buffer |
| | 108 MHz in display controller |
| | 54 MHz in other blocks |

## B. Object Detecting System

An experimental object recognition system was built using a fabricated chip and an FPGA board for demonstration. The chip was controlled by a pattern generator (Tektronix TLA715), and a monitor was used to display the results from the FPGA board. Fig. 13(a) shows a photo of the packaged chip mounted on socket board. Fig. 13(b) shows a photo of the demonstration system without the logic analyzer and the monitor. The socket board is put beneath the lens. Two 100-W lights are used as light sources. Testing objects were put about 30 cm above the lens. The chip extracts both the static features and motion features, and transfers them to the FPGA, in which feature vector generation (FVG) and pattern recognition are performed. The DE2-70 FPGA board from Terasic [72] is used in the system. It contains an Altera Cyclone II 2C70 FPGA as the main device, as well as some auxiliary devices including the video graphics array (VGA) digital-analog convertor (DAC). A summary of the FPGA part in this system is given in Table III.

Fig. 14 shows the system configuration in detail. The MFE chip developed in this paper transfers features (both static and motion features), motion detection results (the feedback

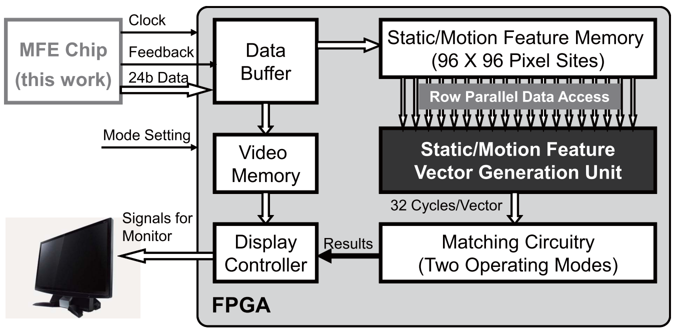


Fig. 14. Configuration of object recognition system built for demonstration.

signal), and clock signal to the FPGA. The clock signal is supplied to an on-chip phase-locked loop inside the FPGA to provide three clocks used for data buffering, signal processing, and display control. Whenever there is a new frame, according to its motion detection result, the data buffer records the extracted features into corresponding SRAM in two memories. One is a video memory for display; the other is a feature memory for data processing. Considering the resolution of the on-chip DPS, the size of the identification window is designed to be 32 × 32. The 64-D APED vector [50] generated from SEMs is used as the static feature vector. For the motion feature vector, a 64-D averaged-displaced-edge-distribution (ADED) vector is proposed in which the 32 × 32 identification window on the edge displacement map (motion features) is equally divided into 64 4 × 4 subregions with the total number of edge flags in each subregion to be one component. A FVG unit is developed to generate both APED and ADED vectors for the matching circuitry. The FVG can access feature data of one row, including both static and motion features, in row parallel. Therefore, it takes 32 clock cycles to generate one pair of APED and ADED vectors. Controlled by the Mode Setting signal, the matching circuitry can run in two operating modes: 1) the multiobject recognition mode and 2) the moving object recognition mode. The on-FPGA display controller generates both the display control/synchronization signals that are directly transmitted to the display and the video signals that are transmitted after being converted to analog signals by the on-board VGA DAC. The operating frequency of the display controller (108 MHz) is determined by the final display format, which is the standard VGA video signal format for 1280 × 1024-pixel display operating at a screen refresh rate of 60 Hz.

Fig. 15 shows the results of two frames when the system was working in multiobject recognition mode. In the experiment, a smiling face was used as the target whose APED vector [50] was extracted as the target vector for pattern recognition. The system identifies the regions that have similar feature vectors with the target vector and marks them with red squares. The successful identification of the target even in case of partial occlusion demonstrates the robustness of the static features in still image analysis.

Fig. 16 shows the results of four frames when the system was working in moving object recognition mode. In the experiment, several copied images of the target were placed

TABLE IV
COMPARISON WITH PREVIOUS DPSs

| | This paper | Ref. [35] | Ref. [36] | Ref. [39] | Ref. [69] | Ref. [52], [57] |
|---|---|---|---|---|---|---|
| **Technology** | 65-nm | 0.18-$\mu$m | 0.35-$\mu$m | 0.18-$\mu$m | 0.35-$\mu$m | 0.18-$\mu$m |
| **Pixel Size** ($\mu$m$^2$) | 10.8 $\times$ 10.8 | 9.4 $\times$ 9.4 | 45 $\times$ 45 | 13 $\times$ 9.5 | 53.65 $\times$ 53.65 | 18 $\times$ 18 |
| **Fill Factor** | 13.5% | 15% | 18% | 15% | 14.9% | 7.4% |
| **Resolution** | 100$\times$100 | 352$\times$288 | 64$\times$64 | 64$\times$64 | 64$\times$48 | 68$\times$68 |
| **Image Processing Function** | Yes | No | Yes | Yes | Yes | Yes |
| **Inter-frame Image Processing** | Yes | No | No | Yes | No | No |

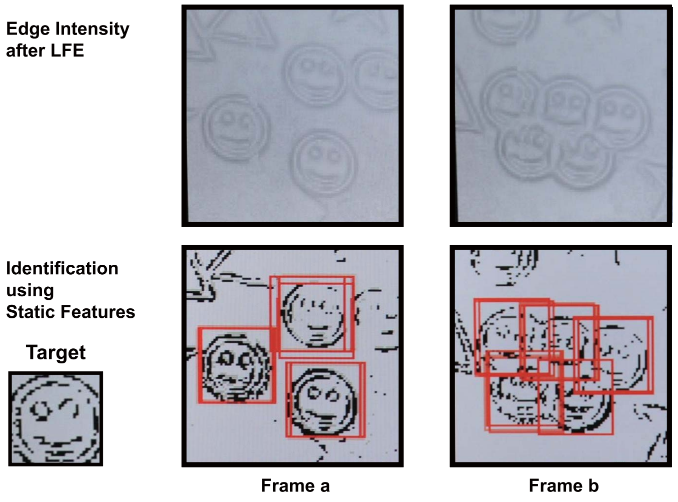


Fig. 15. Experimental results of object recognition system in multiobject recognition mode.

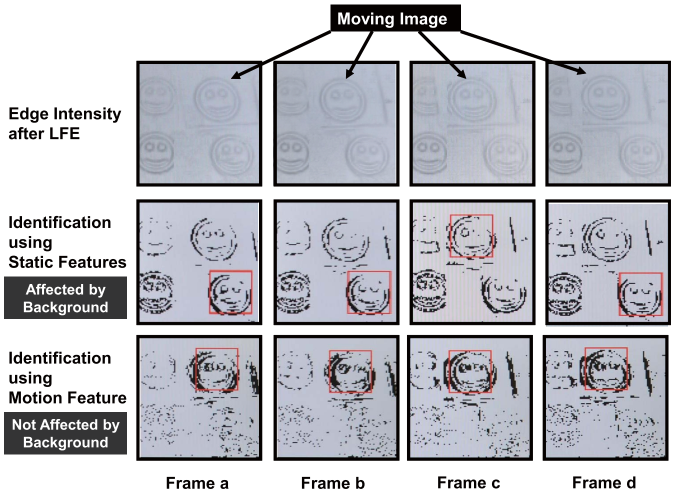

Moving Image
Edge Intensity after LFE
Identification using Static Features
Affected by Background
Identification using Motion Feature
Not Affected by Background
Frame a
Frame b
Frame c
Frame d

Fig. 16. Experimental results of object recognition system in moving object recognition mode.

in the background, and a single image was being moved in front. The system identifies the most similar pattern in the scene using either static features or motion features. When static features are used, the most similar pattern is identified indifferent to its motion. With the motion features, however, only the moving images are identified, as shown in the bottom row. It should be noted that the stationary background images are erased fairly well by temporal derivative operation in MFE, and only the information from moving images is preserved. Such processing is important in robust moving object tracking as well as in action recognition, as demonstrated in [60]. The total processing time of this system in both modes for one frame was 3.5 ms with 2.4 ms consumed in the FPGA (prototype chip operated at 18 MHz; FPGA at 54 MHz for pattern matching and 108 MHz for display control).

### C. Comparison

Table IV shows the comparison between this paper and previous DPSs. Except for the pixel developed in [35], this paper utilizes a smaller pixel size due to the 65-nm process. Furthermore, the advanced process enables the implementation of on-chip large-scale digital processing circuitry to achieve complex functionality. As a result, MFE that employs interframe image processing can be performed in fully pixel parallel.

## V. DISCUSSION

This paper is originally developed to perform the MFE algorithm introduced in Section II. However, both LFE and MFE modules can be improved to implement vision sensor SoCs for other image processing algorithms.

### A. Improvement of LFE Module

In the row-parallel LFE module, eight masks are employed for directional filtering in four directions according to the LFE algorithm in Fig. 1(a). Because there are only $+1$ and $-1$ in the filtering kernels in this algorithm, simple binary masks are used in this paper. These masks are programmable. For example, the original image in Fig. 10 is taken by activating only one mask with only one position allowing the corresponding datum in the 8 $\times$ 8 data block to pass through. For applications that need more complex filtering kernels, such as Gaussian filters or Gabor filters [8], masks with each position having a corresponding weight can be employed. The number of masks, eight in this paper, can also be changed accordingly. The maximum gradient selection circuitry [Fig. 5(b)] can be enhanced to more general purpose processing circuits for other low-level image processing tasks.

### B. Improvement of MFE Module

The MFE module is developed to extract the edge displacement map for the MFE algorithm introduced in Section II.

Based on the extracted motion features, motion fields [64], [65] can be generated in both horizontal and vertical directions for motion analysis. However, the experimental results in [60] show that the motions in vertical direction are more easily detected using the horizontal edges only and vice versa. By activating two masks for one directional kernel and blocking all six other masks, this chip can extract motion features for either horizontal or vertical direction, but not both at the same time. The architecture of the MFE module can be enhanced for performing MFE in two directions (or more) by increasing the number of bits in SRAM columns and designing circuits in PEs to write the data in General Regs. back to the SRAM columns in pixel parallel. In this way, the MFE algorithm introduced in [60], which needs edge displacement maps in both horizontal and vertical directions, can be efficiently implemented.

### C. Analyzing Images With Moving Backgrounds

The MFE algorithm in Section II works well in analyzing images that having static backgrounds, which is quiet important for applications, such as human–computer interface, video surveillance, and remote gesture control. However, the VLSI architecture introduced in this paper can also be used in analyzing images having moving backgrounds. For example, for the ego-motion analyzing algorithms developed in [63] and [73], the computationally expensive GFE procedure can be efficiently carried out using the chip in this paper. In addition, the self-speed-adaptive (or the so-called automatic speed adaptation in [63] and [73]) procedure can be efficiently achieved using the architecture developed in the MFE module. Such ego-motion information is very helpful for analyzing images with moving backgrounds.

## VI. Conclusion

A self-speed-adaptive MFE VLSI has been developed and implemented in a standard 65-nm CMOS process for a VLSI-implementation-friendly algorithm. By employing the DPS, pixel-parallel ADC is achieved. The block-readout scheme provides a direct connection from the image sensor to the LFE module, allowing the row-parallel processing. MFE module has been developed for fully pixel-parallel feature extraction. As a result, a processing time of 0.9 ms for the MFE after capturing an image has been achieved when the chip was running at 20 MHz. The effectiveness of such an architecture was demonstrated by building an object detecting system that can localize images of target objects only when they are in motion.

## References

[1] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 37, no. 3, pp. 311–324, May 2007.

[2] P. Turaga and Y. A. Ivanov, "Diamond sentry: Integrating sensors and cameras for real-time monitoring of indoor spaces," *IEEE Sensors J.*, vol. 11, no. 3, pp. 593–602, Mar. 2011.

[3] S. Berman and H. Stern, "Sensors for gesture recognition systems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 3, pp. 277–290, May 2012.

[4] T. Starner, J. Weaver, and A. Pentland, "Real-time American sign language recognition using desk and wearable computer based video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 12, pp. 1371–1375, Dec. 1998.

[5] T. Shanableh, K. Assaleh, and M. Al-Rousan, "Spatio-temporal feature-extraction techniques for isolated gesture recognition in Arabic sign language," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 3, pp. 641–650, Jun. 2007.

[6] K. Huang, Y. Zhang, and T. Tan, "A discriminative model of motion and cross ratio for view-invariant action recognition," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 2187–2197, Apr. 2012.

[7] M. Ahmad and S.-W. Lee, "Human action recognition using shape and CLG-motion flow from multi-view image sequences," *Pattern Recognit.*, vol. 41, no. 7, pp. 2237–2252, 2008.

[8] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," in *Proc. IEEE 11th ICCV*, Oct. 2007, pp. 1–8.

[9] T. Kobayashi and N. Otsu, "Three-way auto-correlation approach to motion recognition," *Pattern Recognit. Lett.*, vol. 30, pp. 212–221, Feb. 2009.

[10] K. Iwata, Y. Satoh, T. Kobayashi, I. Yoda, and N. Otsu, "Application of the unusual motion detection using CHLAC to the video surveillance," in *Neural Information Processing* (Lecture Notes in Computer Science), vol. 4985. New York, NY, USA: Springer-Verlag, 2008, pp. 628–636.

[11] J. Huang, S. P. Ponce, S. I. Park, Y. Cao, and F. Quek, "GPU-accelerated computation for robust motion tracking using the CUDA framework," in *Proc. 5th Int. Conf. VIE*, 2008, pp. 437–442.

[12] S. Kyo, S. Okazaki, and T. Arai, "An integrated memory array processor for embedded image recognition systems," *IEEE Trans. Comput.*, vol. 56, no. 5, pp. 622–634, May 2007.

[13] W. Raab *et al.*, "A 100-GOPS programmable processor for vehicle vision systems," *IEEE Des. Test Comput.*, vol. 20, no. 1, pp. 8–15, Jan. 2003.

[14] S. Kyo, T. Koga, S. Okazaki, and I. Kuroda, "A 51.2-GOPS scalable video recognition processor for intelligent cruise control based on a linear array of 128 four-way VLIW processing elements," *IEEE J. Solid-State Circuits*, vol. 38, no. 11, pp. 1992–2000, Nov. 2003.

[15] A. A. Abbo *et al.*, "Xetal-II: A 107 GOPS, 600 mW massively parallel processor for video scene analysis," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 192–201, Jan. 2008.

[16] H. Noda *et al.*, "The design and implementation of the massively parallel processor based on the matrix architecture," *IEEE J. Solid-State Circuits*, vol. 42, no. 1, pp. 183–192, Jan. 2007.

[17] J. Tanahe *et al.*, "Visconti: Multi-VLIW image recognition processor based on configurable processor [obstacle detection applications]," in *Proc. IEEE CICC*, Sep. 2003, pp. 185–188.

[18] D. Kim, K. Kim, J. Y. Kim, S. Lee, and H. J. Yoo, "An 81.6 GOPS object recognition processor based on NoC and visual image processing memory," in *Proc. IEEE Int. CICC*, Sep. 2007, pp. 443–446.

[19] D. Kim, K. Kim, J. Y. Kim, S. Lee, S. J. Lee, and H. J. Yoo, "81.6 GOPS object recognition processor based on a memory-centric NoC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 3, pp. 370–383, Mar. 2009.

[20] K. Kim, S. Lee, J. Kim, M. Kim, and H. Yoo, "A 125 GOPS 583 mW network-on-chip based parallel processor with bio-inspired visual attention engine," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 136–147, Jan. 2009.

[21] H. Kondo *et al.*, "Design and implementation of a configurable heterogeneous multicore SoC with nine CPUs and two matrix processors," *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 892–901, Apr. 2008.

[22] J. Oh, G. Kim, J. Park, I. Hong, S. Lee, and H. Yoo, "A 320mW 342GOPS real-time moving object recognition processor for HD 720p video streams," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2012, pp. 220–221.

[23] Y. Tanabe *et al.*, "A 464GOPS 620GOPS/W heterogeneous multi-core SoC for image-recognition applications," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2012, pp. 222–223.

[24] N. Massari and M. Gottardi, "A 100 dB dynamic-range CMOS vision sensor with programmable image processing and global feature extraction," *IEEE J. Solid-State Circuits*, vol. 42, no. 3, pp. 647–657, Mar. 2007.

[25] J. Dubois, D. Ginhac, M. Paindavoine, and B. Heyrman, "A 10000 fps CMOS sensor with massively parallel image processing," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 706–717, Mar. 2008.

[26] Y. Ni and J. Guan, "A 256×256 pixel smart CMOS image sensor for line-based stereo vision applications," *IEEE J. Solid-State Circuits*, vol. 35, no. 7, pp. 1055–1061, Jul. 2000.

[27] A. Graupner, J. Schreiter, S. Getzlaff, and R. Schuffny, "CMOS image sensor with mixed-signal processor array," *IEEE J. Solid-State Circuits*, vol. 38, no. 6, pp. 948–957, Jun. 2003.

[28] Y. Muramatsu, S. Kurosawa, M. Furumiya, H. Ohkubo, and Y. Nakashiba, "A signal-processing CMOS image sensor using a simple analog operation," *IEEE J. Solid-State Circuits*, vol. 38, no. 1, pp. 101–106, Jan. 2003.

[29] V. Gruev and R. Etienne-Cummings, "A pipelined temporal difference imager," *IEEE J. Solid-State Circuits*, vol. 39, no. 3, pp. 538–543, Mar. 2004.

[30] J. Choi, S. W. Han, S. J. Kim, S. I. Chang, and E. Yoon, "A spatial-temporal multiresolution CMOS image sensor with adaptive frame rates for tracking the moving objects in region-of-interest and suppressing motion blur," *IEEE J. Solid-State Circuits*, vol. 42, no. 12, pp. 2978–2989, Dec. 2007.

[31] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 dB 15$\mu$s latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[32] B. Zhao, X. Zhang, S. Chen, K.-S. Low, and H. Zhuang "A 64×64 CMOS image sensor with on-chip moving object detection and localization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 4, pp. 581–588, Apr. 2012.

[33] C. C. Cheng, C. H. Lin, C. T. Li, and L. G. Chen, "iVisual: An intelligent visual sensor SoC with 2790 fps CMOS image sensor and 205 GOPS/W vision processor," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 127–135, Jan. 2009.

[34] W. Zhang, Q. Fu, and N.-J. Wu, "A programmable vision chip based on multiple levels of parallel processors," *IEEE J. Solid-State Circuits*, vol. 46, no. 9, pp. 2132–2147, Sep. 2011.

[35] S. Kleinfelder, S. Lim, X. Liu, and A. E. Gamal, "A 10000 frames/s CMOS digital pixel sensor," *IEEE J. Solid-State Circuits*, vol. 36, no. 12, pp. 2049–2059, Dec. 2001.

[36] S. Chen, A. Bermak, Y. Wang, and D. Martinez, "Adaptive-quantization digital image sensor for low-power image compression," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 1, pp. 13–25, Jan. 2007.

[37] M. Zhang and A. Bermak, "Compressive acquisition CMOS image sensor: From the algorithm to hardware implementation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 3, pp. 490–500, Mar. 2010.

[38] S. Chen, A. Bermak, and Y. Wang, "A CMOS image sensor with on-chip image compression based on predictive boundary adaptation and memoryless QTD algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 4, pp. 538–547, Apr. 2011.

[39] M. Zhang and A. Bermak, "Quadrant-based online spatial and temporal compressive acquisition for CMOS image sensor," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 9, pp. 1525–1534, Sep. 2011.

[40] A. P. Reeves, "Parallel programming for computer vision," *IEEE Softw.*, vol. 8, no. 6, pp. 51–59, Nov. 1991.

[41] L. H. Jamieson, E. J. Delp, C.-C. Wang, J. Li, and F. J. Weil, "A software environment for parallel computer vision," *Computer*, vol. 25, no. 2, pp. 73–77, 1992.

[42] P. Y. Oh, "A template for real-time image processing development: Windows-based PCs and webcams for computer vision research and education," *IEEE Robot. Autom. Mag.*, vol. 12, no. 4, pp. 65–74, Dec. 2005.

[43] S. Chen, P. Akselrod, B. Zhao, J. A. Perez-Carrasco, B. Linares-Barranco, and E. Culurciello, "Efficient feedforward categorization of objects and human postures with address-event image sensors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 302–314, Feb. 2012.

[44] A. M. Kamboh and A. J. Mason, "Computationally efficient neural feature extraction for spike sorting in implantable high-density recording systems," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 21, no. 1, pp. 1–9, Jan. 2013.

[45] J. Kim and T. Chen, "Combining static and dynamic features using neural networks and edge fusion for video object extraction," *IEE Proc.-Vis., Image Signal Process.*, vol. 150, no. 3, pp. 160–167, Jun. 2003.

[46] M. Yagi, H. Ohno, and K. Takada, "Decision-making models compatible with digital associative processor for orthodontic treatment planning," in *Proc. IEEE Biomed. Circuits Syst. Conf.*, Nov. 2009, pp. 149–152.

[47] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *J. Physiol.*, vol. 148, pp. 574–591, Oct. 1959.

[48] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *J. Physiol.*, vol. 195, pp. 215–243, Mar. 1968.

[49] M. Yagi and T. Shibata, "An image representation algorithm compatible with neural-associative-processor-based hardware recognition systems," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1144–1161, Sep. 2003.

[50] Y. Suzuki and T. Shibata, "Multiple-clue face detection algorithm using edge-based feature vectors," in *Proc. IEEE ICASSP*, May 2004, pp. V-737–V-740.

[51] H. Yamasaki and T. Shibata, "A real-time image-feature-extraction and vector-generation VLSI employing arrayed-shift-register architecture," *IEEE J. Solid-State Circuits*, vol. 42, no. 9, pp. 2046–2053, Sep. 2007.

[52] H. Zhu and T. Shibata, "A digital-pixel-sensor-based global feature extraction processor for real-time object recognition," *Jpn. J. Appl. Phys.*, vol. 48, p. 04C080, Apr. 2009.

[53] N. Takahashi, K. Fujita, and T. Shibata, "A pixel-parallel self-similitude processing for multiple-resolution edge-filtering analog image sensors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 11, pp. 2384–2392, Nov. 2009.

[54] M. Ogawa and T. Shibata, "A delay-encoding-logic array processor for dynamic-programming matching of data sequences," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1578–1582, Jul. 2005.

[55] H. Shikano, K. Ito, K. Fujita, and T. Shibata, "A real-time learning processor based on k-means algorithm with automatic seeds generation," in *Proc. IEEE ISSOC*, Nov. 2007, pp. 1–4.

[56] K. Kang and T. Shibata, "An on-chip-trainable Gaussian-kernel analog support vector machine," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 7, pp. 1513–1524, Jul. 2010.

[57] H. Zhu and T. Shibata, "A real-time image recognition system using a global directional-edge-feature extraction VLSI processor," in *Proc. 35th ESSCIRC*, Sep. 2009, pp. 248–251.

[58] H. Zhu, P. Zhao, and T. Shibata, "Directional-edge-based object tracking employing on-line learning and regeneration of multiple candidate locations," in *Proc. IEEE ISCAS*, May 2010, pp. 2630–2633.

[59] P. Zhao, H. Zhu, H. Li, and T. Shibata, "A directional-edge-based real-time object tracking system employing multiple candidate-location generation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 3, pp. 503–517, Mar. 2013.

[60] H. Hayakawa and T. Shibata, "Block-matching-based motion field generation utilizing directional edge displacement," *Comput. Electr. Eng.*, vol. 36, pp. 617–625, Jul. 2010.

[61] R. Bao and T. Shibata, *A Hierarchical Action Recognition System Applying Fisher Discrimination Dictionary Learning via Sparse Representation* (Lecture Notes in Computer Science), vol. 7267. New York, NY, USA: Springer-Verlag, 2012, pp. 468–476.

[62] R. Bao and T. Shibata, "A hardware friendly algorithm for action recognition using spatio-temporal motion-field patches," *Neurocomputing*, vol. 100, pp. 98–106, Jan. 2013.

[63] J. Hao and T. Shibata, "An ego-motion detection system employing directional-edge-based motion field representations," *IEICE Trans. Inf. Syst.*, vol. E93-D, no. 1, pp. 94–106, Jan. 2010.

[64] K. Fujita, K. Ito, and T. Shibata, "A single-motion-vector/cycle-generation optical flow processor employing directional-edge histogram matching," in *Proc. IEEE ISCAS*, May 2009, pp. 3022–3025.

[65] Y. Okano and T. Shibata, "A high-frame-rate dense motion vector field generation processor with simplified best-match searching circuitries," in *Proc. IEEE A-SSCC*, Nov. 2009, pp. 205–208.

[66] Y. Ma and T. Shibata, "A binary-tree hierarchical multiple-chip architecture for real-time large-scale learning processor systems," *Jpn. J. Appl. Phys.*, vol. 49, p. 04DE08, Apr. 2010.

[67] R. Zhang and T. Shibata, "Fully-parallel self-learning analog support vector machine employing compact Gaussian-generation circuits," *Jpn. J. Appl. Phys.*, vol. 51, p. 04DE10, Apr. 2012.

[68] Z. Hou, Y. Ma, H. Zhu, N. Zheng, and T. Shibata, "Real-time very large-scale integration recognition system with an on-chip adaptive k-means learning algorithm," *Jpn. J. Appl. Phys.*, vol. 52, p. 04CE11, Apr. 2013.

[69] K. Ito, B. Tongprasit, and T. Shibata, "A computational digital pixel sensor featuring block-readout architecture for on-chip image processing," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 1, pp. 114–123, Jan. 2009.

[70] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 12, pp. 2247–2253, Dec. 2007.

[71] B. K. Kar and D. K. Pradhan, "A new algorithm for order statistic and sorting," *IEEE Trans. Signal Process.*, vol. 41, no. 8, pp. 2688–2694, Aug. 1993.

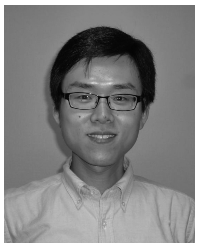
[72] [Online]. Available: http://www.terasic.com.tw/en

[73] J. Hao and T. Shibata, "Digit-writing hand gesture recognition by hand-held camera motion analysis," in *Proc. IEEE ICSPCS*, Omaha, NE, USA, Sep. 2009, pp. 1–5.

[74] Y. Watanabe, T. Komuro, S. Kagami, and M. Ishikawa, "Real-time visual measurements using high-speed vision," *Proc. SPIE*, vol. 5603, pp. 234–242, Oct. 2004.

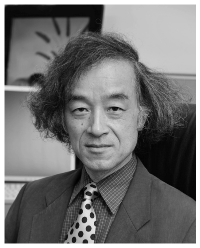
[75] H. Oku, N. Ogawa, M. Ishikawa, and K. Hashimoto, "Two-dimensional tracking of a motile micro-organism allowing high-resolution observation with various imaging techniques," *Rev. Sci. Instrum.*, vol. 76, no. 3, pp. 034301-1–034301-9, 2005.

[76] N. Furukawa, A. Namiki, S. Taku, and M. Ishikawa, "Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system," in *Proc. IEEE ICRA*, May 2006, pp. 181–187.



**Hongbo Zhu** (M'14) received the B.Eng. degree from the Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China; the M.Eng. degree from the Division of Electrical, Electronic and Information Engineering, Osaka University, Osaka, Japan; and the Ph.D. degree from the Department of Electronic Engineering, University of Tokyo, Tokyo, Japan, in 2004, 2007, and 2010, respectively.

He was with the Embedded Systems Research Department, Central Research Laboratory, Hitachi, Ltd., Tokyo, from 2010 to 2011. In 2011 he joined the VLSI Design and Education Center, University of Tokyo, as an Assistant Professor. His research interests include CMOS vision sensors, and intelligent image processing algorithms, circuits, and systems.



**Tadashi Shibata** (LM'14) was born in Japan in 1948. He received the B.S. degree in electrical engineering and the M.S. degree in material science from Osaka University, Osaka, Japan, and the Ph.D. degree from University of Tokyo, Tokyo, Japan, in 1971, 1973, and 1984, respectively.

He was with Toshiba Corporation, Tokyo, from 1974 to 1986, where he was a VLSI Process and Device Engineer and involved in the development of microprocessors, dynamic RAMs, and EEPROMs. From 1978 to 1980 he was a Visiting Research Associate at Stanford Electronics Laboratories, Stanford University, Stanford, CA, USA, where he studied the laser beam processing of electronic materials, including silicide, polysilicon, and superconducting materials. From 1986 to 1997 he was an Associate Professor with Tohoku University, Sendai, Japan, where he was involved in the research of low-temperature processing and ultraclean technologies for VLSI fabrication. Since the invention of a new functional device called Neuron MOS Transistor ($\nu$MOS) in 1989, his research interests shifted from devices and materials to circuits and systems. Since 1997, he has been a Professor with the Department of Electrical Engineering and Information Systems, University of Tokyo. Since his retirement from the University of Tokyo in 2013, he has been serving as an Editor-in-Chief of *Applied Physics Express and the Japanese Journal of Applied Physics*. His research interests include developing human-like intelligent computing systems based on the state-of-the-art silicon technology and the biologically inspired as well as psychologically inspired models of the brain.

Dr. Shibata is a member of the Japan Society of Applied Physics; the Institute of Electronics, Information and Communication Engineers; and the IEEE Electron Devices Society, Circuits and Systems Society, and Computer Society.