VLSI Circuits and Systems for Directional-Edge-Based Intelligent Image Processing

（方向性エッジ情報に基づく知的画像処理回路・システムに関する研究）

Hongbo ZHU

Supervisor: Professor Tadashi SHIBATA

Department of Electronic Engineering,

The University of Tokyo

December 2009

For the memory of my father Jianmin Zhu ...

Abstract

The continuous progress in semiconductor VLSI technologies during the past several decades has provided the opportunity of realizing *real-time intelligent image processing systems* such as image recognition, object tracking, motion recognition, etc. However, the traditional approach of running image processing algorithms on general purpose processors is not practical for building efficient systems at rational costs with low power-consumption. Therefore, a number of VLSI chips having parallel processing architectures such as graphics processing units (GPUs) have been developed to enhance the performance. Although the processing time can be reduced greatly, such approaches are not still efficient enough due to the complex and expensive image processing algorithms which usually include a number of floating point operations. In order to resolve the problem of such a large gap between the algorithms and their VLSI implementation and to maximally utilize the power of semiconductor technologies, we try to develop algorithms which are compatible with the physical characteristics of VLSI circuits.

The robust nature of the human brain in visual information processing has been attracting a lot of researchers to discover better ways of image processing. Physiology research has revealed that the *directional edge information* in images is utilized as the most important clue in visual object recognition. Being inspired by such a biological principle, a series of *direction-edge-based* VLSI-implementation-adapted intelligent image processing algorithms as well as the corresponding VLSI circuits and systems have been proposed and developed in our laboratory.

This work succeeds the research in such bio-inspired algorithms, circuits, and systems. In order to minimize the latency caused by the image data transfer between the image sensor and the processing circuits, the most serious bottleneck in such systems, digital-pixel-sensor-embedded (DPS-embedded) processors were proposed and designed. The performance of such processor has been verified by building a real-time image recognition system with a very low latency. In

addition, a directional-edge-based object tracking algorithm was also proposed and partially implemented in an object tracking system by building processing circuits on FPGAs. In the followings, the work is described in more detail.

Firstly, a DPS-embedded global feature extraction VLSI processor for real-time image recognition has been developed. By combining the block-readout architecture of DSP and parallel processing elements, the latency of local feature extraction has been markedly reduced. By adapting the rank-order filter algorithm to hardware implementation, global feature extraction is accomplished in only 11 cycles. A prototype chip was designed in a 0.18-μm five-metal CMOS technology. The measurement results show that the VLSI processor can extract features more than 400 times faster than software processing running on a 2-GHz general-purpose processor when operating at 60 MHz.

Then, a DPS-embedded early-visual-processing VLSI processor for real-time intelligent image processing has been developed. Compared with the first chip, the enhancement in the functionality of processing elements in the global image processing block further improves the programmability of the processor. As a result, such a chip can handle multiple algorithms efficiently. A prototype chip was designed in a 65-nm 12-metal CMOS technology. The simulation results show that this VLSI processor can achieve all expected functions.

In order to demonstrate the power of such chips, a real time image recognition system has been developed. The system is based on a VLSI-implementation friendly image recognition algorithm. By using the global directional-edge-feature extraction VLSI processor, the latency between the image capture and the final recognition as small as 906 μs has been demonstrated. The merit of the global feature extraction algorithm that it can focus on more significant features automatically has also been experimentally verified.

In the research on algorithms, a directional-edge-based object tracking algorithm was developed. By using directional-edge-based feature vectors, the system has been made robust against illumination variation. The on-line learning technique and the statistical multiple-candidate-location generation have further improved the performance, making the system robust against object size variation, partial occlusion, and object deformation. The performance was verified by experiments under varying disturbing conditions.

Finally, a simple real time object tracking system based on a restrained version of the prior algorithm has been implemented successfully. By experimental results, this system shows satisfying performance in simple tracking tasks by employing only eight candidate locations. Thanks to the fine-grained VLSI-implementation of the object tracking algorithm implemented in an FPGA, the total processing time for the tracking task has been reduced to about 0.1 ms when the system is running at a frequency of 60 MHz.

In this work, *circuits and systems* for brain-mimicking algorithms have been developed based on a very naïve model of the brain. In the algorithms of image processing, *directional edge information* plays an essential role for perception of still images as well as moving images. In these systems, the vast amount of subconscious processing in the mind has been implemented by VLSI chips or FPGAs. In order to build "*real-time responding human-like intelligent systems*" with small hardware volume and low powers, such development of hardware-friendly algorithms and their VLSI implementation in fine-grain parallel architectures are most essential.

Acknowledgements

With the utmost gratitude I would like to thank my advisor Professor Tadashi Shibata for his loyal support and enduring guidance during the three years. His enthusiasm for teaching and research, his constant encouragement and guidance on my research led me to become a full-fledged person. I feel very fortunate to have taken him as my supervisor, and the precious experiences in this laboratory will be irreplaceable assets in my life.

I would like to acknowledge my dissertation committee: Prof. Kunihiro Asada, Prof. Takayasu Sakurai, Prof. Makoto Ikeda, and Prof. Toshihiko Yamasaki for their preview of the thesis and their extremely valuable comments, discussions to my research. Their constructive suggestions were indispensable for making my dissertation study successful.

I would like to thank Prof. Yoshio Mita for his prompt support of my research as well as his insightful lectures which have greatly expanded my interest in and enriched my knowledge of MEMS.

I express my appreciation to Ms. Kimiko Mori for her help on so many documentaries; Ms. Motoko Inagaki, Ms. Yoko Inoue, and Ms. Kimiko Shigihara for their supports.

I am very thankful to my colleagues and friends: Dr. Kiyoto Ito, Dr. Hitoshi Hayakawa, Dr. Bui Trong Tu, and Dr. Jia Hao for their invaluable support and advice; Mr. Norihiro Takahashi, Mr. Robert Grou-Szabo, Mr. Shigetaka Morikawa, Mr. Yitao Ma, Ms. Mio Nishiyama, Mr. Pushe Zhao, Mr. Ruihan Bao, and Ms. Dandan Han for being dependable and supportive; Mr. Satoshi Morishita, Mr. Takashi Miyoshi, Mr. Hyun-soo Kim, Mr. Seungho Shin, Mr. Zhuoli Sun, and Mr. T. M. P. S. Weerawardhana, for their input and enthusiasm; and all the members in Shibata & Mita lab for the time we shared together.

I am immensely grateful to my parents Jianmin Zhu and Baofen Liu for their very prudent unconditional support. Finally, my deepest gratitude I must reserve

for my wife Xuhua Zhang, whose patience and understanding with both myself and my son I am very thankful for.

Many thanks for the Global Center of Excellence (GCOE) program for financial supporting of my life. The VLSI chips in this study have been fabricated in the chip fabrication program of VDEC, the University of Tokyo with the collaboration with Rohm Corp. and Toppan Printing Corp. and with STARC, e-Shuttle, Inc., and Fujitsu Ltd., respectively.

Hongbo Zhu
The University of Tokyo

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Following the well-known Moore's law (1), semiconductor very large scale integration (VLSI) technologies have made remarkable progress in the past several decades and will continue to advance. Such a rapid development has provided the opportunity of realizing *real-time intelligent image processing systems* such as image recognition, object tracking, and association. However, the traditional approach of running image processing tasks on general purpose processors is not practical for building efficient systems at rational costs with low power-consumption. In order to solve this problem, a number of VLSI chips, including the popular used graphics processing units (GPUs), are developed to enhance the performance. These chips can significantly increase the performance of such systems; for example, in a particular application, a speed up of 263 has been reported by using a GPU (2). Although the processing time can be reduced greatly, such approaches are not efficient enough regarding to the complex and expensive image processing algorithms that usually include a number of floating point operations. The large gap between the algorithm and the VLSI implementation severely restricts the full utilization of the power of semiconductor technologies. Therefore,

intelligent image processing algorithms that can be implementation directly into VLSIs are very desirable. In addition, in the era of nanoscale integration, devices have presented difficult problems such as crosstalk, electromagnetic interference (EMI), radiations, process variations, simultaneously switching outputs (SSO), leakage current, temperature fluctuations, etc. The variation problem (3; 4; 5; 6), in particular, is a very serious issue because it is rooted in the fundamentals of nature. As a result, for building circuits and systems in such advanced technologies, the algorithms should be robust enough to tolerate the unexpected errors.

To meet such requirements in developing intelligent image processing systems, instead of using or improving the existed algorithms such as scale-invariant feature transform (SIFT) (7), gradient location and orientation histogram (GLOH) (8; 9), or Speeded-up robust features (SURF) (10) which are very difficult to be implemented into VLSI directly, we search ideas from the human brains. This is because of the brains' robust functions, especially its strong visual information processing capability. Physiology research (11) has revealed that the *directional edge information* in images is utilized as the most important clue in visual object recognition. Being inspired by such a biological principle, a series of *direction-edge-based intelligent image processing algorithms* as well as the corresponding *VLSI circuits and systems* have been proposed and developed.

In §1.2, related works in circuit implementation for intelligent image processing are described. §1.3 explains our bio-inspired image processing algorithms and their corresponding circuits. Then §1.4 shows the scope of the thesis, and finally the organization of the thesis is described in §1.5.

1.2 Related Works

1.2.1 VLSI processors for real-time performance

In this section, some VLSI implementations for intelligent image processing and their problems are introduced.

For performance enhancement, many configurable processors are developed by employing single instruction/multiple data (SIMD) architecture (12; 13; 14; 15). In reference (12), a one-dimensional SIMD linear array including 128 processing elements (PEs) is employed for achieving real-time performance in intelligent vehicle video recognition applications. Such kind of SIMD architectures are also employed in reference (14), in which the 320 PEs are grouped into 40 compute tiles for the simple and power-efficient interconnection complexity; and in reference (15), in which the 2048 fine-grained PEs are used for energy and cost improvements. One drawback of such processors is: these processors focus only on the low-level image data processing operations like image filtering, and they are not suitable for object-level parallelism, which is essential for higher level vision applications such as object recognition.

On the other hand, architectures for not only the data level parallelism, but also the task level parallelism are also developed (16; 17; 18; 19). For example, the chip in (18) integrates 10 SIMD PEs for data/task level parallelism. A more advanced work from the same group is also reported as a SIMD/MIMD dual-mode parallel processor that contains 8 SIMD linear array PE clusters which have 8 PEs each in (16). Heterogeneous architecture (20) has also been developed for performance improvement.

Another very influential trend for performance enhancement of image processing is the Graphics Processing Units (GPUs) (2). In (21), a GPU-based Cellular Neural Network (CNN) simulator that can run 8-17 times faster than a CPU-

based CNN simulator is reported; and in some particular applications such as the MRI reconstruction in (2), a maximum speed up of 263 has been reported.

Therefore, by employing such kind of VLSI chips, the image data processing time can be dramatically reduced. However, since such chips are designed to be compatible with many existing algorithms, for each particular application, some redundant energy consumption is inevitable, which limits the power efficiency of the whole system. In addition, the delay caused by the data transfer from the image sensor to the processor, which is usually in a one pixel per clock cycle way, severely limits the efficiency of the image data access; make such a system not suitable for time critical applications. Also, the employment of image sensor, processors, and sometimes memories also makes such systems to be complex and burden, thus not applicable for size-critical applications.

1.2.2 Smart sensors for low-latency

To develop real time image processing systems with low latency and compact size, it is very preferable to integrate the image sensor and the processing circuits on the same chip. The compatibility between the CMOS image sensor and the processing circuits together with the requirement of efficient image data access method leads to many researches on smart sensors in which the image processing functions are implemented on the image sensor.

Several works design the processing circuit mainly inside each pixel to achieve functions such as: image filtering (22; 23), gradient extraction (24), contrast processing (25; 26), selective region output (27; 28), inter-frame processing (22; 26; 29; 30; 31), pixel-parallel ADC (32), and bright dot tracking (33). While embedding the processing circuits in each pixel can achieve very fast processing speed, too complex tasks or unsuitable processing algorithms can cause the pixel to be very complex or even impossible. On the other hand, implement the

column/row-parallel or serial processing circuits on the image sensor but out of the pixel-array can handle more flexible tasks such as: complex image filtering (34; 35), histogram-based processing (34), color processing (36), 3D image sensor (37; 38), multi-sensing (38), and motion detection (39). When combine the high processing speed of in-pixel circuits and the flexible function of the off pixel-array circuits, many works also report further image sensor performance improvement (40; 41; 42), edge-extraction (41), motion detection (41; 43), 3D applications (44; 45), matrix transform (46; 47), and image compression (48). With the power of smart sensors, systems with small size and low latency can be achieved. Some image sensors are expected to be used for bio-application (49; 50; 51).

Therefore, the development of smart image sensors opens the possibility of designing even high level functional sensors that can handle more complex tasks such as intelligent image processing. However, compared with the smart sensors that can handle normal image processing tasks, the smart sensors for intelligent image processing are less reported. In reference (52), a tracking function is developed on the image sensor but the application is limited to a particular condition: eye-tracking. A really intelligent visual sensor is reported in (53), which implement the image sensor and the vision processor on the same chip. But since the image data of one frame are firstly buffered in a group of memory, and then processed by the vision processor, the merit of on chip image sensor is not fully utilized which limit the power and processing efficiency. In addition, because the embedded vision processor is developed to be compatible with many existing algorithms which are probably not VLSI-implementation friendly oriented developed, such an incompatibility between algorithm and VLSI also leads to system inefficiency.

1.3 Our Approach

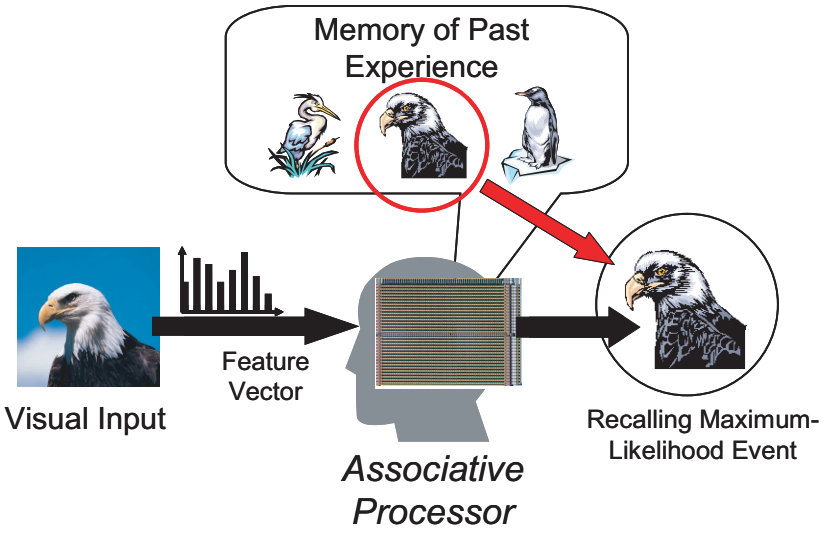


Figure 1.1: Psychologically-inspired VLSI brain model based on the associative principle.

As introduced in §1.1, researches have been advanced in our laboratory aiming at realization of flexible information processing like a human brain. Our laboratory has originally proposed the "psychologically-inspired VLSI brain model" as the foundation (54). Fig. 1.1 shows the psychologically-inspired VLSI brain model based on the *associative principle*. Since the biological details about the higher cognitive functions of the brain are not yet known, we call it psychological inspiration, by which we mean the model has been built by observing our mental activities by ourselves. The model is based on the assumption that human intelligence is produced by the vast amount of past memories accumulated through our life long experience, and that the most similar event to the current event is automatically retrieved from the past memories and is utilized to understand the situation and make a decision (55). The core of the system is an associative processor, the maximum-likelihood search engine, which has been developed in

digital (56; 57; 58) as well as analog (59; 60; 61; 62) CMOS technologies.

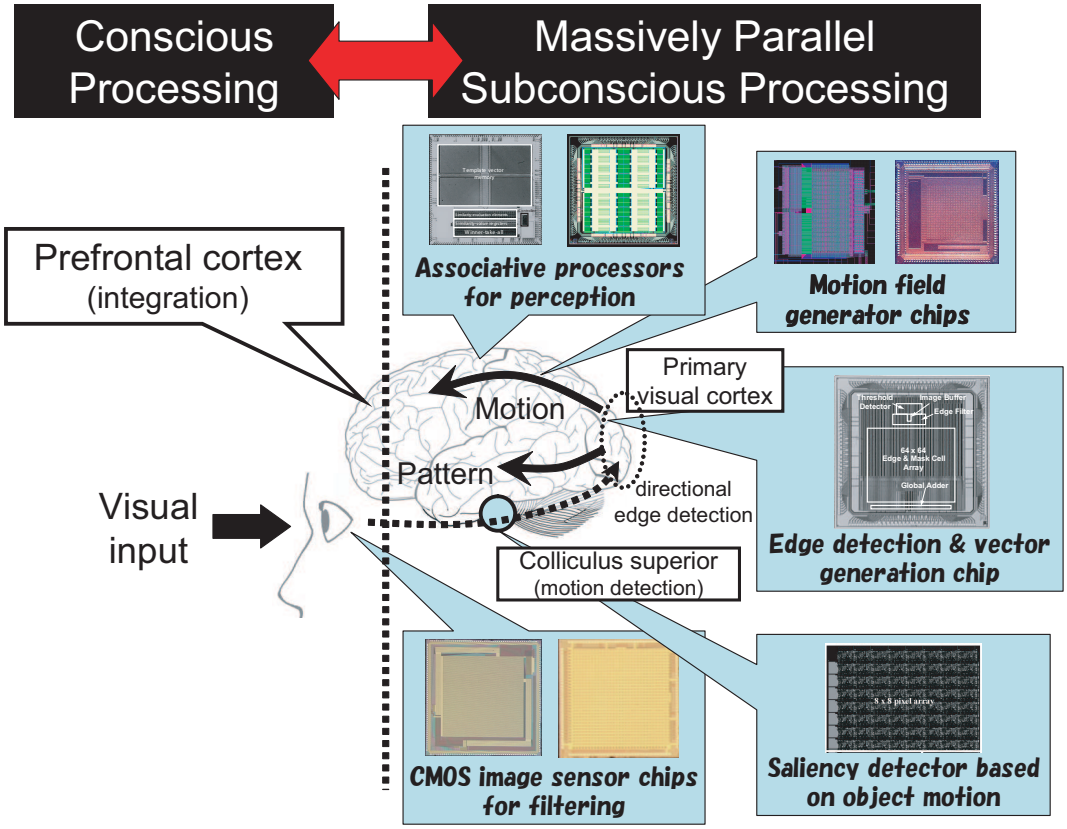


Figure 1.2: Brain-mimicking VLSI system composed of dedicated VLSI chips having fine-grain massively-parallel architectures for subconscious processing.

Fig. 1.2 (54) illustrates a *brain-mimicking VLSI system*, a very naïve (or maybe over-simplified) model of the brain aiming at VLSI implementation. Massively parallel *subconscious processing* is carried out by VLSI chips dedicated to each processing according to the intelligent image processing algorithms for image recognition (63; 64; 65), object tracking (66), ego-motion detection (67), and motion recognition (68; 69) we developed originally. Such VLSI chips included achieve real-time performance of such algorithms in moving object detection (70; 71), directional edge detection (72; 73) and image feature representation

(74; 75), motion field generation and their vector representation (76; 77) as well as their perception by associative retrieval of past experiences (56; 57; 58; 59; 60; 61; 62; 78; 79). Each chip is implemented in analog, digital or mixed signal technology in a *fine-grain massively-parallel architecture*.

1.4 Scope of This Thesis

Although the parallel computation architectures of these VLSI chips introduced in §1.3 can accelerate the processing speeds of the intelligent image processing algorithms considerably. Most of these processors must read image data one pixel at a time from an off-chip image sensor (74). As a result, a large latency occurs during this data transfer, making these approaches not suitable for time-critical applications. On the other hand, for those chips with embedded image sensor introduced in §1.3, the functionalities are also limited to very particular processing tasks, thus can not be used directly to build a system (70; 72; 73).

Therefore, a scope of this study is to develop a VLSI architecture to attack the problem of data transfer delay between the image sensor and the processors in such systems. The digital pixel sensor (DPS (32; 80; 81)) featuring the block-readout technique (82; 83) has been used to break the bottleneck of the low data transfer bandwidth between image sensors and processing circuits in this architecture. As a result, it has become possible to carry out the local image processing such as local feature extraction at each pixel in a line-parallel manner without extra buffer memory to store the image data. In addition, an efficient global image processing unit consists of SIMD processing elements for each pixel site is developed to run the intelligent image processing algorithms (65; 66; 68) efficiently.

A proof-of-concept chip for the proposed architecture was designed in a 0.18

μm five-metal complementary metal-oxide-semiconductor (CMOS) technology. The effectiveness of this architecture has been demonstrated by building a real-time image recognition system. Thanks to the fast response of this processor, the system achieves a latency of only 906 μs which is very suitable for time-critical applications. Another chip with more flexible functions was also designed in a 65 nm 12-metal CMOS technology and the measurement results shows that it can work correctly.

In addition, a VLSI-implementation-friendly object tracking algorithm that is robust in various circumstances was proposed. To enhance the performance of a tracking system, an on-line learning technique together with a statistical approach were developed. This statistical approach is based on regeneration of multi-candidate locations which is similar to the particle filter but much easier to be implemented into VLSI. By simulation experiments under various conditions, this algorithm has been shown to be robust against illumination variation, object size variation, partial occlusion, and object deformation. Furthermore, a restrained version of this algorithm has been implemented into a real-time object tracking system using FPGAs successfully. By experimental results, this system shows satisfying performance in simple tracking tasks. Thanks to the fine-grained VLSI-implementation of the object tracking algorithm inside an FPGA, the total time consumption of the image processing task has been reduced to about 0.1 ms when the system is running at a frequency of 60 MHz.

1.5 Thesis Organization

In chapter 2, a digital-pixel-sensor-based global feature extraction VLSI processor for real-time object recognition is introduced with algorithm, architecture, implementation, and measurement results. An enhanced version of this processor is

introduced in chapter 3, in which more flexible functions are included. Then in chapter 4, a real-time image recognition system using the VLSI processor developed in chapter 2 is described. In chapter 5, a directional-edge-based object tracking algorithm employing on-line learning and regeneration of multiple candidate locations is introduced with software simulation results. Chapter 6 gives a system implementation of a restrained version of the object tracking algorithm introduced in chapter 5. Finally, chapter 7 summarizes major accomplishments of this study.

Chapter 2

A Digital-Pixel-Sensor-Based Global Feature Extraction VLSI Processor for Real-Time Object Recognition

2.1 Introduction

There has been considerable interest in real-time object recognition since they play essential roles in various applications such as automotive car control, video surveillance, robot control, human-computer interfaces, and so forth. A number of algorithms have been proposed in this field to achieve this task, aiming at higher precision, faster response time, and lower computational cost. Generally, these algorithms were developed as software programs running on general-purpose processors(84; 85; 86; 87; 88). Because these algorithms are computationally expensive, to achieve real time performances in object recognition tasks, very large scale integration (VLSI) chips such as digital signal processors (DSPs), field programmable gate arrays (FPGAs), and application-specific integrated circuits (ASICs) were developed to accelerate the computation(89; 90; 91; 92; 93; 94).

In ref. (89), a highly parallel DSP architecture that can process many complex functions such as 5×5 spatial filtering was designed for real-time image recognition. In ref. (91), the reported image processing applications employing FPGA are between 8 and 800 times faster than the equivalent programs running on a Pentium III processor. In ref. (94), a resonant adiabatic mixed-signal ASIC was designed and a real-time template-based face detection function was realized as one application of this work. Therefore, with the power of such programmable VLSI chips, the processing speed of existing recognition algorithms can be clearly enhanced.

However, in many cases, algorithms are not tuned to be efficiently implemented in VLSI hardware. Since VLSI hardware implementation is essential for building real-time image recognition systems, some researchers began to conceive of algorithms that can be easily integrated on a chip for more compact implementation with lower power consumption(95; 96). Reference (95) presents an algorithm for iris recognition using phase-based image matching. To reduce the size of iris data, they introduce the idea of two-dimensional Fourier phase code for representing iris information, which can be implemented using state-of-the-art DSP technology. In ref. (96), an object detection/recognition algorithm based on principal component analysis was proposed and implemented by introducing three-dimensional integration of multiple chips.

Human like flexible recognition algorithms specifically adapted to VLSI implementation have been developed(97). Physiology research results reveal that the directional edge information in images is utilized as the most important clue in visual object recognition(11). Being inspired by such a biological principle, a directional edge-based feature vector representation algorithm called projected principal-edge distribution (PPED) was proposed(98) and has been successfully applied to medical radiograph analysis as well as to handwritten pattern

recognition(63). This image recognition algorithm is composed of four steps as illustrated in Fig. 2.1: image capture, feature extraction, vector generation and template matching. Some VLSI chips were designed for this architecture(56; 74; 78; 79). In ref. (74), a VLSI processor for feature extraction and vector generation was developed. In refs. (56; 78; 79), different VLSI chips were designed to accelerate template matching. Thanks to the parallel computation architectures of these VLSI chips, the processing speed of the recognition algorithm has been considerably accelerated, making real-time object recognition feasible. However, since such processors must read image data one pixel at a time from an off-chip image sensor, a large latency occurs, making these approaches not suitable for time-critical applications.

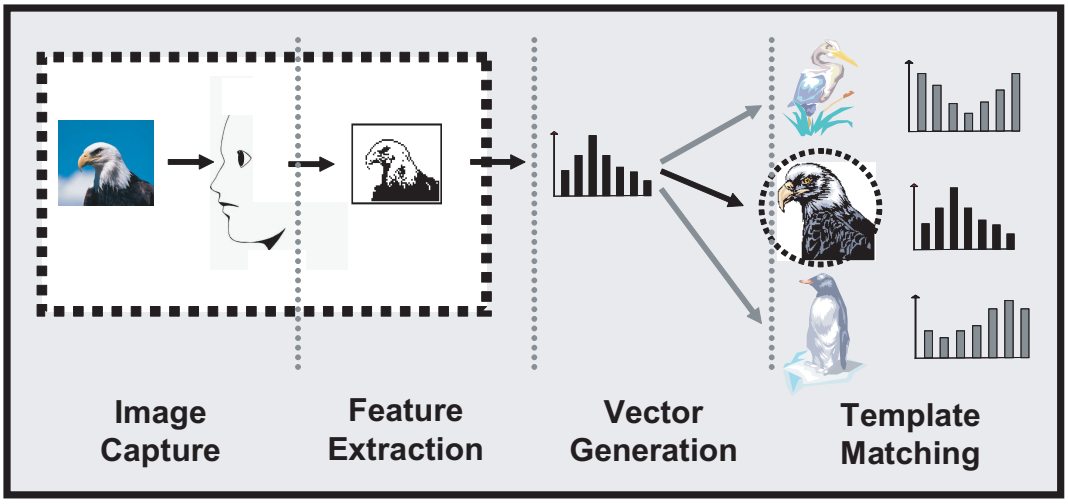


Figure 2.1: Four steps of image recognition algorithm.

Thus, the purpose of this study is to develop a global feature extraction VLSI processor that is more compatible with time-critical image recognition tasks. The digital pixel sensor (DPS(32; 80; 81)) featuring the block-readout architecture(82) has been used to break the bottleneck of the low data transfer bandwidth between image sensors and processing circuits. As a result, it has become possible to carry

out the local feature extraction processing at each pixel in a line-parallel manner. To eliminate trivial local features, an efficient global feature extraction circuitry that can retain any given number of relatively more significant features has been designed and integrated on the same chip. By measurement results, it has been shown that this VLSI processor is capable of extracting features 400 times faster than software running on a 2-GHz general-purpose processor when operating at 60 MHz.

The organization of this chapter is as follows. In §2.2, the feature extraction algorithm proposed in this study is explained. In §2.3, the VLSI implementation of the algorithm is described. In §2.4, the layout of the test chip and the measurement results are presented. Finally, the conclusions are given in §2.5.

2.2 Feature Extraction Algorithm

The feature extraction algorithm developed in the present study is based on four types of directional edge filtering (horizontal, $+45°$, vertical, and $-45°$) as in the original PPED algorithm (98). However, the present algorithm differs from the original PPED in terms of how the significant features are selected to generate the final directional edge maps. It is composed of two main processes: the local feature extraction and the global feature extraction. Figure 2.2(a) shows the local feature extraction processing at each pixel site. Firstly, convolutions between the 5×5-pixel region centered at the pixel site and four 5×5-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 most significant edge direction at the pixel site and its convolution value is preserved. This process is repeated for every pixel, and the entire image is scanned with the filtering kernels. Figure 2.2(b) shows the edge flags obtained in the local feature extraction

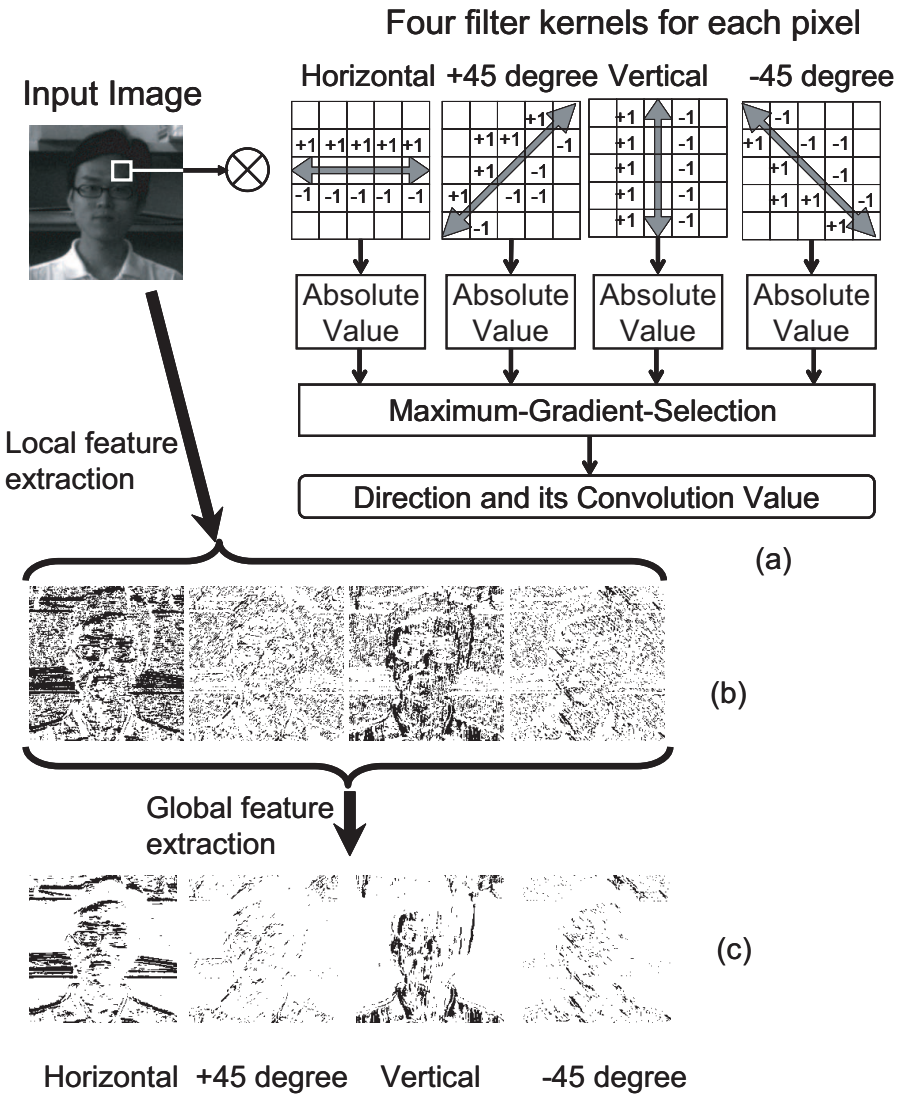


Figure 2.2: Feature extraction algorithm. (a) Local feature extraction at each pixel site by four-directional edge filtering, (b) extracted local features, and (c) feature representing edge flags in four directions.

stage. Since these edge flag maps are very noisy and contain much redundant information, only salient features are selected and retained in the global feature extraction stage. This is carried out by selecting only a predetermined number of significant edge flags out of all the pixels in the image, which have larger gradient values than the others. Such a selection is possible because all the convolution values obtained from every pixel site are preserved. Figure 2.2(c) shows four global feature maps when only 40 % of the more significant edge flags are retained. Such a feature detection scheme is different from the original PPED in which the local variance of luminance distribution was utilized to select essential features. It was shown that the global features obtained in the present algorithm show more robust performances in certain recognition tasks including, in particular, motion perception (66; 69).

2.3 VLSI Implementation

2.3.1 System organization

A VLSI processor was accurately designed following the feature extraction algorithm described in §2.2. Figure 2.3 shows the overall architecture. It is composed of three main blocks: 68 × 68 DPS, 16 groups of local feature extraction (LFE) circuits, and a global feature extraction (GFE) unit. DPS is used to capture the image. Sixteen groups of processing elements are used for parallel local feature extraction. The complex interconnection between DPS and 16 LFE circuits is markedly simplified by the block-readout architecture, which will be explained in detail later. Because the size of each filter kernel is 5 × 5, a 68 × 68 original image is converted to four 64 × 64 directional edge maps with the maximum convolution value at every pixel site. Then, the directional edge flags with their convolution values are stored in static random access memories (SRAMs) in the

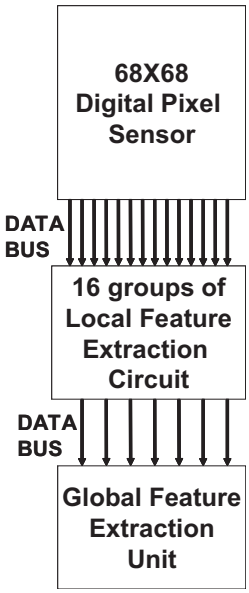


Figure 2.3: Architecture of global feature extraction VLSI processor.

GFE unit. Rank-order-filter algorithm adapted for hardware implementation is employed in the GFE unit so that the global thresholding process for all 4096 11-bit data can be accomplished in only 11 cycles. Such a fast processing capability is contrasted with those complex algorithms used in software (99).

2.3.2 DPS featuring block-readout architecture

After the DPS is introduced in ref. (32), which integrated both single-slope bit-parallel analog/digital converter (ADC) and 8-bit dynamic random access memory (DRAM) cells into each pixel, the research on DPS has been advancing to achieve more flexible performances (80; 81; 82). In ref. (82), a new architecture for pixel data readout was developed, making massive and parallel access to image data possible. This architecture enables us to read out image data from eight rows in a bit-serial manner at one read cycle and to seamlessly scan the entire pixel array with filtering kernels up to 5 $\times$ 5-pixel sizes. As a result, the DPS has

become compatible with various image processing algorithms. Therefore, the single-slope bit-parallel ADC introduced in ref. (32) and the enhanced 8 bit DRAM cells with eight control signals for block-readout architecture introduced in ref. (82) are both employed in the pixel design in the present chip.

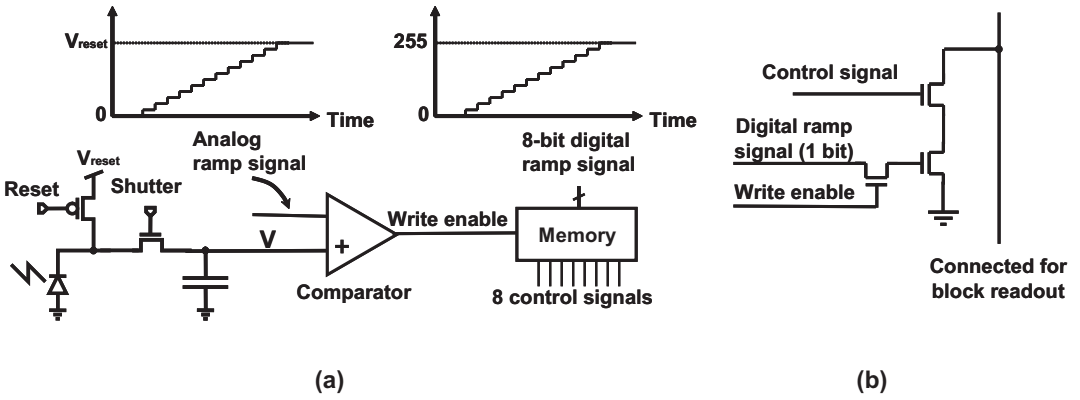


Figure 2.4: Circuits in one pixel (a) and 1-bit pixel memory circuit (b).

Figure 2.4(a) shows the basic concept of the DPS (32). The pixel unit consists of a photodiode, a reset transistor, a shutter transistor, a sampling metal-oxide-semiconductor (MOS) capacitor, an analog comparator, and 8-bit DRAM cells. Figure 2.4(b) shows 1-bit pixel memory circuit. A global reset process in which both *reset* transistor and *shutter* transistor are on forces the voltage V at the MOS capacitor to V_{reset} for all pixels. After that, the *reset* transistor turns off while the *shutter* transistor remains on, then the decrease in voltage V at the MOS capacitor corresponds to the light intensity at each pixel. After some period, all the *shutter* transistors are turned off simultaneously and the light intensity in each pixel is converted into the analog signal V. Then, an analog ramp signal and eight-bit digital ramp codes, which are synchronized to each other, are provided to all pixel units for ADC. The analog comparator in each pixel compares 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 time 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. This operation is conducted simultaneously at all pixels, thus achieving massively parallel ADC. Eight control signals are used here to select which bit to read out during a read process.

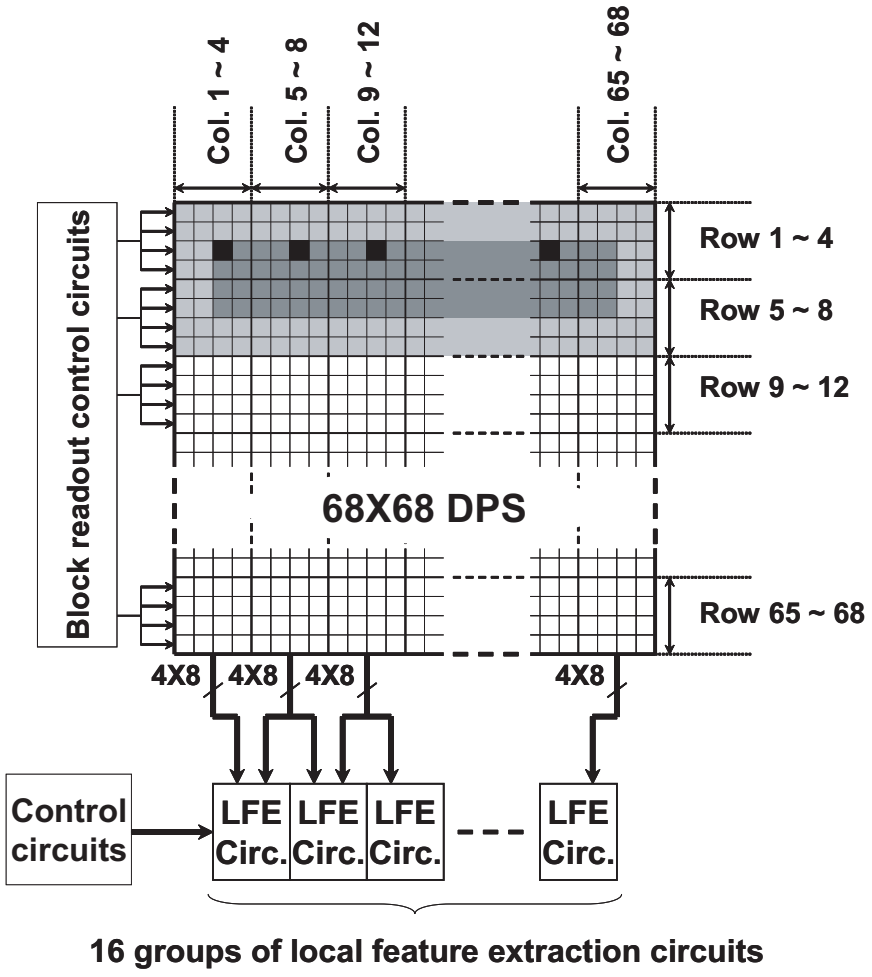


Figure 2.5: Block readout architecture.

Figure 2.5 shows the principle of block-readout architecture (82). To seam-

lessly scan the entire pixel array with 5 × 5-pixel-size filtering kernels, four rows are grouped together and controlled by the same enable signal generated by "block-readout control circuits". In one read cycle, the same bit in pixels from two consecutive groups of rows are read out. For example, in Fig. 2.5, the same bit in "light gray" pixels from rows 1 to 8 can be read out simultaneously. These data are sufficient to perform the 5 × 5 kernel calculations at all "dark gray" pixel sites in a bit-serial manner immediately with no extra buffer memories. Therefore, simultaneous filtering at all "dark gray" pixel sites is possible in principle. However, to keep the circuits rational in the interconnection and chip areas, the kernel calculation should be performed in a line-parallel manner.

Sixteen groups of LFE circuits are employed as shown in Fig. 2.5. The leftmost LFE circuits process the data from columns 1 to 8, the second LFE circuits from the left process the data from columns 5 to 12, and the rightmost LFE circuits process the data from columns 61 to 68. Figure 2.5 shows an example in which 5 × 5 kernel operations centered at all "black" pixel sites are conducted by these 16 groups of LFE circuits at the same time. Thus, the kernel calculations of one line are completed in four iterations. The details of the signal processing in the LFE circuit is explained in section §2.3.3 using Figure 2.6. Thanks to this architecture, the one-pixel by one-pixel image data transfer and extra image data storage are unnecessary. As a result, the latency is reduced considerably.

2.3.3 Local feature extraction circuits

Fig. 2.6 shows the implementation of a single LFE circuit shown in Fig. 2.5 in more detail. By using the block-readout architecture, 8 × 8 1-bit data from a selected block in the DPS are transferred to each LFE circuit at each clock cycle. An LFE circuit possesses eight masks, two masks for each kernel: one for the plus component, the other for the minus component. In each mask, there are three

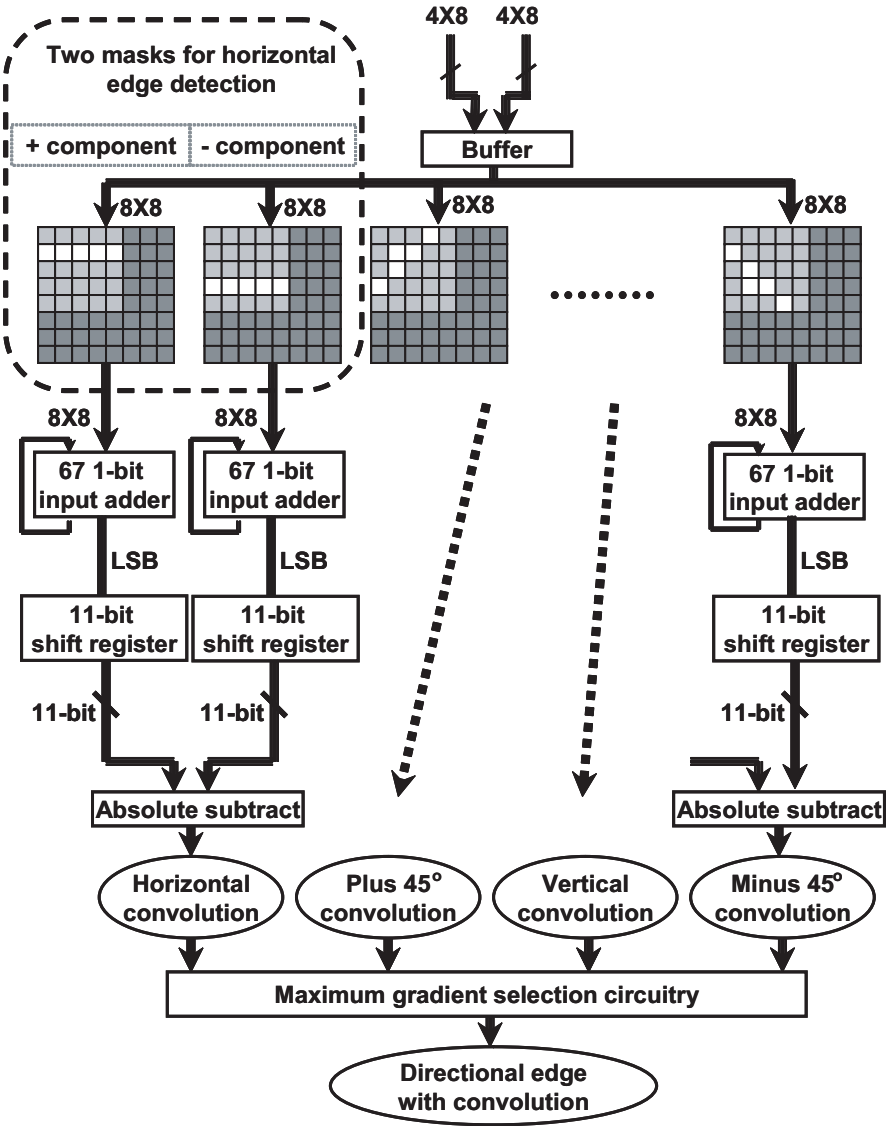


Figure 2.6: Local feature extraction circuit.

types of pixels, which are indicated as "dark gray", "light gray" and "white" in Fig. 2.6. In the "dark gray" pixels, the bit data are blocked and converted to zero, thus transforming the 8 × 8 1-bit data block into a 5 × 5 1-bit data block. The "light gray" pixels are also blocked to zero to generate kernel filter patterns, where plus and minus coefficients are generated by two separate masks. Then the data bits only in the "white" pixels are transferred to the 67 1-bit input adder. This adder outputs the result in four bits. Here, 64 inputs are for the image data and the extra three bits are for the upper three bits of the previous summation results, which are fed back to inputs in each cycle.

For example, in Fig. 2.6, the leftmost mask forces the 59 signals in both "light gray" and "dark gray" pixels to be zero before being transferred to the adder and only allows the five plus components for the horizontal direction to pass. These mask patterns are controlled using two groups of shift registers, one group for vertical scanning, the other for horizontal scanning. Controlled by these signals, the mask can process 5 × 5 filtering kernel at all 4 × 4 pixel sites in the middle of the 8 × 8 data block. Therefore, with 16 groups of LFE circuits, the "dark gray" area in Fig. 2.5 can be scanned by 5 × 5 filtering kernels seamlessly.

Because the block-readout architecture transfers one bit in each read cycle from the least significant bit (LSB) to the most significant bit (MSB), the adder counts the number of 1's in its 64 (8 × 8) inputs. The maximum of this number is five because only five pixel data can pass through the mask. The LSB of the adder's output is sent to the "11-bit shift registers", while the upper three bits are fed back to the input for accumulation. Since only five 8-bit image data can pass through each mask and be added in a bit-serial manner, the maximum summation result is 255 × 5, which is an 11-bit number. As shown in Fig. 2.6, after these shift registers collect all 11 bits of the summation results, the "absolute subtract" circuit calculates the absolute difference between the "+" component

and the "-" component, which is the 11-bit convolution value for this direction. Then, the maximum gradient selection circuitry that is designed employing the two-dimensional bit-propagating scheme (56) determines the edge direction at the pixel site. In this manner, the edge direction at this pixel site is determined with its corresponding convolution value. The results are stored in SRAM arrays in the GFE unit for future processing of global feature extraction.

2.3.4 Global feature extraction unit

Traditionally, the rank order filter is used to determine the nth order in a set of N data (99). For example, $n = 1$ yields the maximum value, $n = N$ the minimum value, and $n = (N + 1)/2$ the median value. In this paper, the algorithm is modified for marking the values that are larger than the nth order and implemented in a simplified circuitry for global feature extraction.

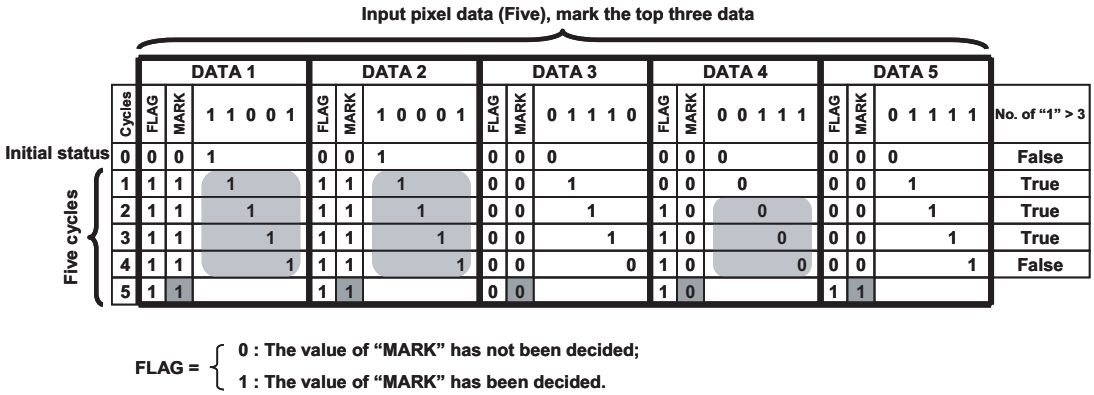


Figure 2.7: Global feature extraction algorithm.

The algorithm employed in the implementation is explained using an example given in Fig. 2.7. The figure shows the operation of finding the larger three data in a five-data set (DATA1 $\sim$ DATA5). The operation begins from MSB and propagates to the following bit in the next clock cycle. Each datum is accompanied by

a “FLAG” and a “MARK”. The “FLAG” shows whether the value of “MARK” has been decided or not; while the “MARK” shows whether this datum is larger than or equal to the third rank order datum. Both “FLAG” and “MARK” are set to “0” at the beginning.

In the first clock cycle, all MSBs of the five data are summed up. If the sum is greater than the assigned order, the result of this operation is “True (1)”; otherwise, it is “False (0)”. In the example, the sum is equal to 2; thus, the summation result in this bit is “False (0)”. Then, “FLAG” for the data whose values are different from the result, i.e., DATA1 and DATA2, is converted to “1”, and “MARK” is altered to the same value as its bit. In this manner, all the following bits in DATA1 and DATA2 are masked by their “MARK”, i.e., “1” in this example.

In the second clock cycle, the processing is carried out on second MSBs from the five data. The same procedure as in the first clock cycle is repeated. According to the example, the sum is four, and it is larger than the assigned order number. Thus, the result is “True (1)”. Here, the less significant bits of DATA4 are all altered to “0”. The same set of operations is repeated down to LSB. The result is available as soon as the calculation of LSB is finished. Therefore, DATA1, DATA2, and DATA5 are marked “1” as the three larger values out of five.

Figure 2.8(a) shows the configuration of the GFE unit, which is composed of 4096 processing elements, a 4096 1-bit input adder, and a comparator. In each cycle, the outputs from the 4096 processing elements are added by the 4096-input adder, from which the rank order number (TH) is subtracted. The MSB of the subtraction result, which is “0” when the summation is larger than the rank order number (TH) and “1” otherwise, is inverted and fed back to each processing element for determining “FLAG” and “MARK”. Figure 2.8(b) shows one processing element, which is composed of two parts: a 13-bit SRAM and a

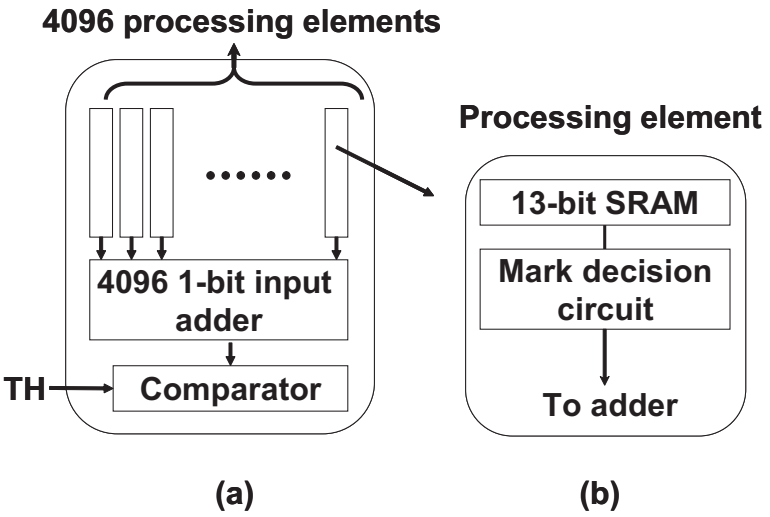


Figure 2.8: Global feature extraction circuit.

"mark decision" circuit. The 13-bit SRAM stores 11-bit convolution value and 2-bit value for four directions ("00": horizontal, "01": +45°, "10": vertical, and "11": −45°).

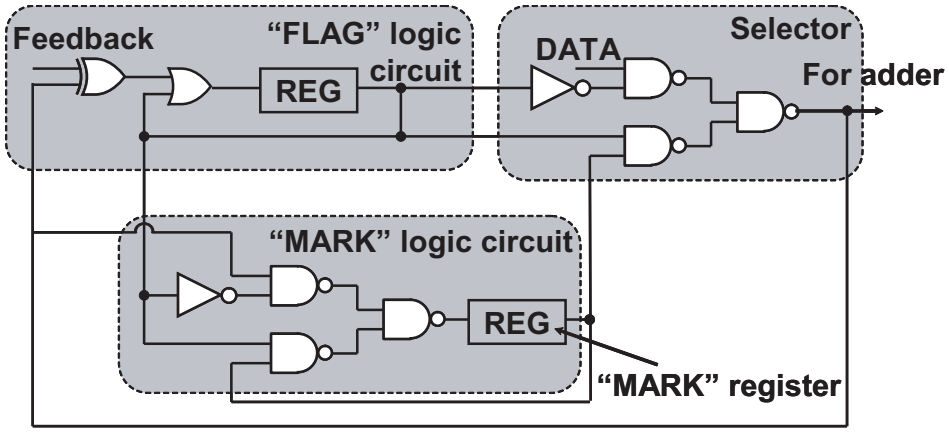


Figure 2.9: Schematic of the "mark decision" circuit.

Figure 2.9 shows the simplified schematic of the "mark decision" circuit. The circuit is comprised of three parts: a "FLAG" logic circuit, a "MARK" logic

circuit, and a selector. It processes the pixel data in a bit-serial manner according to the algorithm illustrated in Fig. 2.7. After 11 cycles, 4096 "MARK" values are decided and preserved in "MARK" register in each "mark decision" circuit. The data in these "MARK" registers can be read out; therefore, both the "MARK" values and the data in the SRAM array can be read out.

2.4 Chip Design and Measurement Results

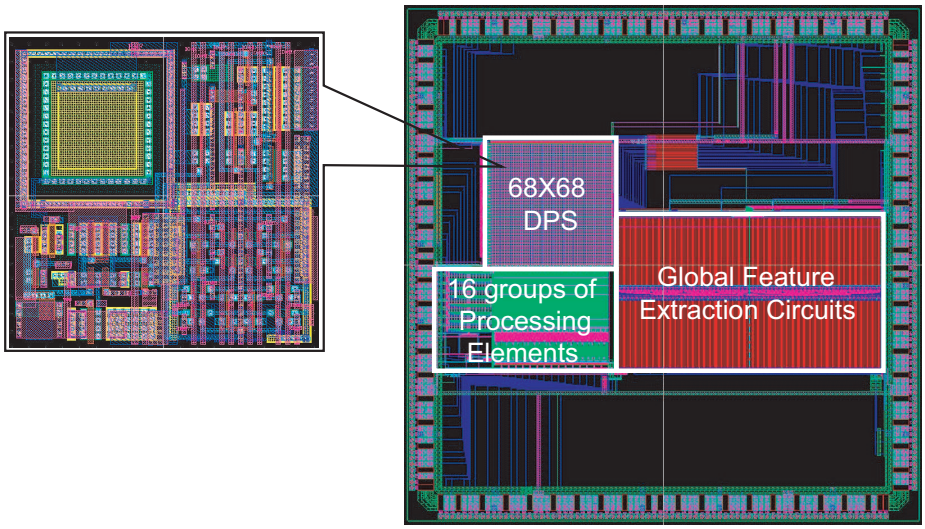


Figure 2.10: Layout of global feature extraction VLSI processor without the top metal, which is used for light shielding.

A VLSI chip was designed in a 0.18-μm five-metal complementary metal-oxide-semiconductor (CMOS) technology. Figure 2.10 shows the entire layout without the top metal that is used for light shielding. This chip includes all three blocks described in §2.3. The specification of this chip is summarized in Table 2.1.

Fig. 2.11 shows the environment of the seasurement, in which the chips is mounted on a socket and placed beneath the lens. The object is put above the

Table 2.1: Specifications of test chip. ($\star$ the light integration time for photodiodes is not considered)

Technology	0.18-μm CMOS, 1-poly, 5-metal
Core size (mm^2)	4.5 $\times$ 2.5
Number of pixels	68 $\times$ 68
Pixel pitch (μm^2)	18 $\times$ 18
Fill factor	7.4%
Transistor count	1.6 million
Supply voltage	1.8 V
Clock freq.	60 MHz
Power	67 mW($\star$)
Frame rate	5000 f/s($\star$)

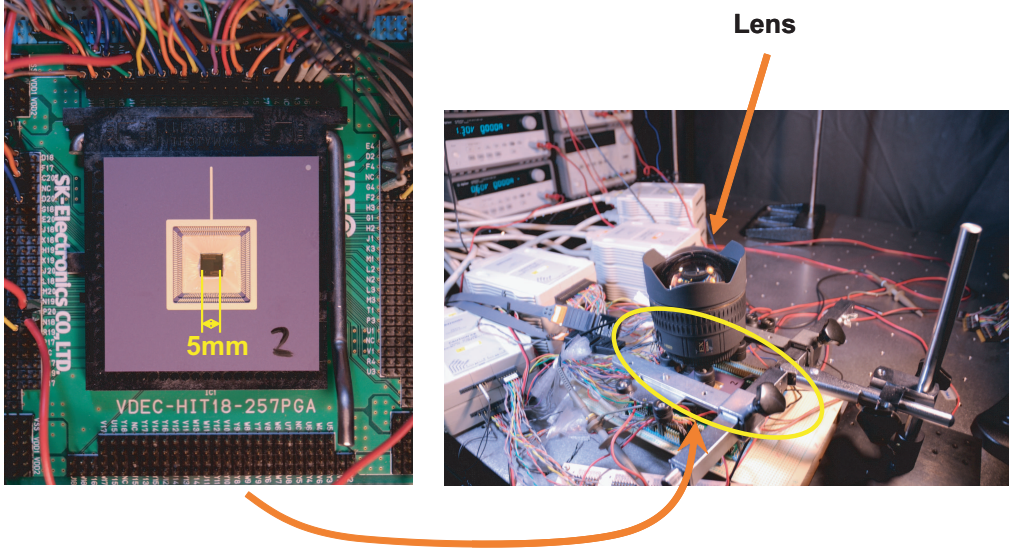


Figure 2.11: A photo of the measurement environment.

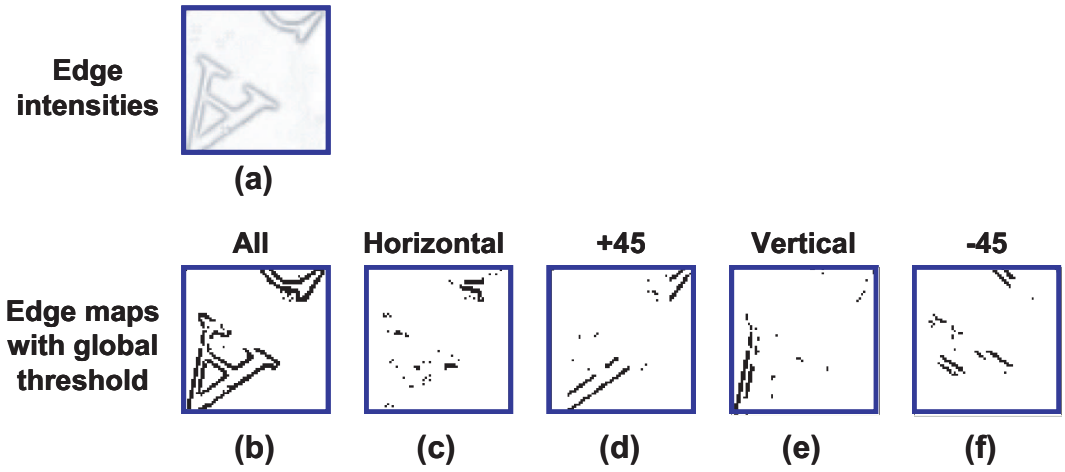


Figure 2.12: Measurement results of global feature extraction VLSI processor. Global Threshold: 512; (a) edge intensities, (b) all edge map (485 out of 4096), (c) edge map in horizontal, (d) edge map in +45°, (e) edge map in vertical, and (f) edge map in −45°.

lens. By measurement results, the function of the system has been verified. Fig. 2.12 shows the measurement results. The global threshold was set at 512, which denotes that it should retain more significant 512 edges out of 4096 pixels. The simulation results give 485 edges as shown in Fig. 2.12(b). The missed edges are due to the pixels that have the same convolution value. The remaining edges for each direction are shown in Figs. 2.12(c) - (f). The processing time is 0.19 ms at 60 MHz, which is more than 400 times faster than software processing running on a 2-GHz general-purpose processor. If distribution histograms are produced from these edge maps, they play essential roles in intelligent image recognition applications as shown in refs. (63) and (64). Such a high-speed performance with the minimal latency is in particular important in applications to motion recognition (68; 69).

2.5 Summary

a DPS-embedded global feature extraction VLSI processor for real-time image recognition has been developed. By combining the block-readout architecture of DSP and parallel processing elements, the latency of local feature extraction has been markedly reduced. By adapting the rank-order filter algorithm to hardware implementation, global feature extraction is accomplished in only 11 cycles. A prototype chip was designed in a 0.18-μm five-metal CMOS technology. The measurement results show that the VLSI processor can extract features more than 400 times faster than software processing running on a 2-GHz general-purpose processor when operating at 60 MHz.

Chapter 3

Design of Advanced Early-Visual-Processing VLSI Processor Using 65-nm Technology

3.1 Introduction

Nowadays, in our daily lives, image sensors can be found in many commodities such as cameras, mobile phones, computers, game machines, cars, and so forth. Thanks to these little "eyes", recording and transferring pictures and movies become very convenient, which greatly broaden our view as well as memories. With an image sensor at the back of the car, we don't need to turn back during parking, with an ultra-high speed camera, we can see what exactly happens when a needle is penetrating a balloon, and with a camera on the mobile phone, we don't need to memorize the bus schedule. Besides these ordinary task, in which we just need raw images, many new applications are also under researching and developing, aiming for more sophisticated systems which can provide the desired information from the images: such as natural human-computer interface (100), autonomous

robot/vehicle (101; 102), visual surveillance (103), and the ubiquitous environment (104; 105). Since such kind of real-time intelligent image processing tasks are always computationally very expensive, how efficiently we can process the image data acquired by image sensors is an essential issue in developing these systems. There are two important factors that affect the system performance, one is how to access the image data efficiently, and the other is how to process the image data efficiently.

The vast availability of image sensors is due to the continuously improvement of the silicon process technology. Although charge-coupled devices (CCDs) have traditionally been the dominant image-sensor technology, recent advances in the design of complementary metal oxide semiconductor (CMOS) technologies have made them more preferable for their lower power, lower price, and wider dynamic range. In addition to these benefits, the compatibility of CMOS image sensor with processing circuits is very important for performance enhancement. By developing specific circuits for different purposes, Ref. (106; 107; 108; 109) achieve noise-reduction, Ref. (110; 111; 112) get wider dynamic-range, Ref. (113) also reports seamlessly binning mode change. Furthermore, since the design of functional circuits on CMOS image sensor has much flexibility, it is also possible to achieve even more complex image processing tasks on the same chip.

The appearance of CMOS image sensors with low price and sufficient performance brings the opportunity of realizing the algorithms that are verified by software processing of image data stored on some media (e.g. a hard disc). As a result, the real-time intelligent image processing systems become possible. For example, the "open source computer vision (OpenCV) (114)", a very popular library of programs, collects many basic functions for real time applications such as segmentation, recognition, classification, motion tracking, etc. However, since the computation of such algorithms are usually very heavy, in many cases the

performance of software running on general purpose CPUs is far from sufficient even with the multi-core technique. To attack this problem, many VLSI chips are developed for accelerating these algorithms by employing different parallel processing circuits (12; 16; 20). A very influential trend toward this problem is the graphics processing units (GPUs) (2). In (21), a GPU-based cellular neural network (CNN) simulator that can run 8-17 times faster than a CPU-based CNN simulator is reported; and in some particular applications such as the MRI reconstruction in (2), a maximum speed up of 263 has been reported. Therefore, by employing such kind of hardware accelerators, the image data processing time can be dramatically reduced. However, since such chips are designed to be compatible with many existing algorithms, for each particular application, some redundant energy consumption is inevitable, which limits the power efficiency of the whole system. In addition, the delay caused by the data transfer from the image sensor to the accelerator, which is usually in a one pixel per clock cycle way, severely limits the efficiency of the image data access; makes such a system not suitable for time critical applications. Also, the employment of image sensor also makes the system to be complex and burden, thus not applicable for size-critical applications.

The compatibility between the CMOS image sensor and the processing circuits together with the requirement of efficient image data access method leads to many researches on smart sensors in which the image processing functions are implemented on the image sensor. Several works design the processing circuit mainly inside each pixel to achieve functions such as: image filtering (22; 23), gradient extraction (24), contrast processing (25; 26), selective region output (27; 28), inter-frame processing (22; 26; 29; 30; 31), pixel-parallel ADC (32), and bright dot tracking (33). While embedding the processing circuits in each pixel can achieve very fast processing speed, too complex tasks or unsuitable process-

ing algorithms can cause the pixel to be very complex or even impossible. On the other hand, implement the column/row-parallel or serial processing circuits on the image sensor but out of the pixel-array can handle more flexible tasks such as: complex image filtering (34; 35), histogram-based processing (34), color processing (36), 3D image sensor (37; 38), multi-sensing (38), and motion detection (39). When combine the high processing speed of in-pixel circuits and the flexible function of the off pixel-array circuits, many works also report further image sensor performance improvement (40; 41; 42), edge-extraction (41), motion detection (41; 43), 3D applications (44; 45), matrix transform (46; 47), and image compression (48). With the power of smart sensors, systems with small size and low latency can be achieved. Some image sensors are expected to be used for bio-application (49; 50; 51).

Therefore, the development of smart image sensors opens the possibility of designing even high level functional sensors that can handle more complex tasks which we named intelligent image processing such as: image recognition, tracking and even motion recognition. However, compared with the smart sensors that can handle normal image processing tasks, the smart sensors for intelligent image processing are less reported. In reference (52), a tracking function is developed on the image sensor but the application is limited to a particular condition: eye-tracking. A really intelligent visual sensor is reported in (53), which implement the image sensor and the vision processor on the same chip. But since the image data of one frame are firstly buffered in a group of memory, and then processed by the vision processor, the merit of on chip image sensor is not fully utilized which limit the power and processing efficiency. In addition, because the embedded vision processor, as well as many other visual processors (12; 16; 20), is developed to be compatible with many existing algorithms which are probably not VLSI-implementation friendly oriented developed, such an incompatibility

between algorithm and VLSI also leads to system inefficiency. As a result, how to develop the intelligent image processing algorithms to make them easy to be implemented directly at the device and circuit levels is very important. Being inspired by the biological principle (11), a series of VLSI-implementation friendly algorithms were proposed and have been successfully applied to pattern recognition (63; 65; 98), object tracking (66) and motion recognition (68; 69). Some VLSI chips have been designed for accelerating these algorithms such as the VLSI processor for feature extraction and vector generation (74) and template matching VLSI chips in both analog and digital domains (56; 78; 79). To further reduce the latency of such systems, a Digital-Pixel-Sensor (DPS (32; 80; 81)) embedded global feature extraction VLSI processor has been developed for real-time image recognition (115; 116). By employing a block read-out architecture (83) and an efficient sorting algorithm, the feature extraction processing: the most computational intensive step in the image recognition algorithm, can be achieved in an efficient method with low latency. By employing this smart sensor, a real-time image recognition system was developed in reference (117). Thanks to the efficient image data access and processing methods, the latency between the image capture and the final recognition has been reduce to only 906 μs, which makes such a sensor very suitable for time critical applications. One drawback of this work is, since this smart sensor is developed especially for static image recognition, it is difficult to handle other algorithms such as object tracking (66) or motion recognition (68) directly, requiring much processing out of this chip.

Thus, in this paper, we succeed the design in reference (115; 116), to develop a DPS embedded early-visual-processing VLSI processor which can be used for more applications. By properly balancing the processing tasks into: pixel-parallel ADC, line-parallel local image processing and pixel-parallel global image processing, a compact and programmable design is achieved. The DPS technique

achieves fast pixel-parallel ADC, providing the possibility of accessing the digitized image data directly from the sensor array, which also gives the opportunity of saving the specific memories for image data buffering. The block read-out architecture (83) realizes the efficiency image data access; enable the bit-serial kernel calculation without any data buffering. For local processing, a programmable line-parallel local image processing circuits which can handle filtering and maximum gradient selection functions which need to be performed repetitively for each pixel site. The local processed data are stored in the memories inside the processing element (PE) of global processing circuits, which perform frame level calculations such as salient feature selection, differential feature map generation, integrated feature map generation, and self-adapted frame selection. This architecture is designed using a 65-nm 12-Metal standard CMOS process. The function of this chip was verified by measurement results.

The organization of this paper is as follows. In §3.2, the targeted intelligent image processing algorithm in this study is explained. In §3.3, the VLSI implementation is described. In §3.4, the layout of the test chip and the chip measurement results are presented. Finally, the conclusions are given in §3.5.

3.2 Intelligent Image Processing Algorithms

3.2.1 Global feature extraction for static image recognition algorithm

As introduced in chapter 2 and also references (63; 65; 98), the feature extraction algorithm for the static image recognition is based on four types of directional edge filtering (horizontal, $+45°$, vertical, and $-45°$). It is composed of two main processes: the local feature extraction (LFE) and the global feature extraction (GFE). In the LFE, convolutions between the 5×5-pixel region centered at

the pixel site and four 5 × 5-pixel filtering kernels (one for each direction) are calculated. Then, based on the four convolution results, the direction with the maximum gradient value is selected as the most significant feature, which we call the local feature, at the pixel site and its convolution value is preserved. The direction in all pixel sites are expressed by four binary edge maps (horizontal, +45°, vertical, and −45°), for example, a black point (binary number 1) in the horizontal edge map means the direction of this pixel site is horizontal. In the GFE, based on the convolution values for all pixel sites, only a determined number of pixel sites with larger convolution values are selected and retained. The determined number is emphasized here because it is closely related to the other two intelligent image processing algorithms. Since such a global feature extraction algorithm is well explained in §2.2, the details are not shown here.

3.2.2 Differential directional-edge image generation for object tracking algorithm

A robust object tracking algorithm is proposed in reference (66). By introducing the concept of the differential directional-edge image (DDEI), a map of edge flags produced from the difference of two consecutive edge images, the information from background has been effectively removed. As a result, this algorithm is robust against the influence of cluttered background, as well as illumination change and speed variation. Since the smart sensor in this paper mainly focuses on how to generate the DDEI efficiently for such kind of tracking algorithms, only the process of DDEI generation is introduced in detail.

In order to generate the DDEI, firstly edge filtering is carried out at every pixel site in the tracking window using a 5 × 5-pixel directional filtering kernel, the vertical kernel is usually used for tracking in the horizontal direction. After filtering each pixel site has a convolution value. Then perform the process of

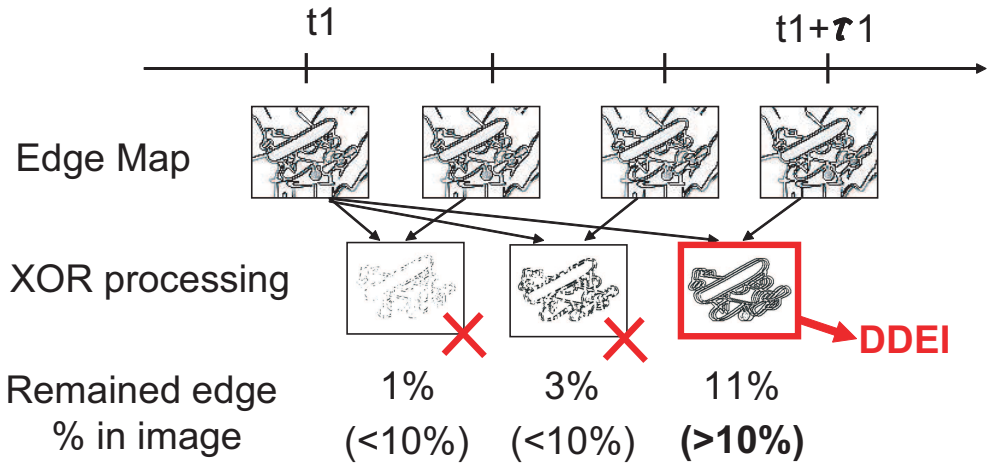


Figure 3.1: DDEI generation for the object tracking algorithm.

GFE to retain only a determined number of the more significant edges as the binary edge map. Generation of a DDEI is illustrated in Fig. 3.1. From the two directional-edge images at $t1$ and $t1 + \tau 1$, DDEI is generated by taking XOR. In generating DDEI, the number of edge flags after XOR is counted in every frame. When the edge count reaches a predetermined value, i.e., Th_{DDEI}, the edge flag map at the moment is accepted as the DDEI at $t1$. Th_{DDEI} is adopted as a percentage of the total number of pixels in the tracking window. Initially, Th_{DDEI} is set by hand, and then this value is controlled automatically by some other mechanism.

3.2.3 Directional edge displacement (DED) map generation for motion recognition algorithm

In reference (68), a motion field generation algorithm using block matching of edge flag histograms has been developed aiming at its application to motion recognition systems. The usage of edge flags instead of pixel intensities has made the algorithm robust against illumination changes. In order to detect local motions

of interest effectively, a new adaptive frame interval adjustment scheme has been introduced in which only the edge flags due to local motions present in the frame are accumulated and utilized in block matching. As a result, the computational cost for best match search has been substantially reduced. Since the smart sensor in this paper mainly focuses on how to generate the DED map efficiently, only the process of DED map generation is introduced in detail.

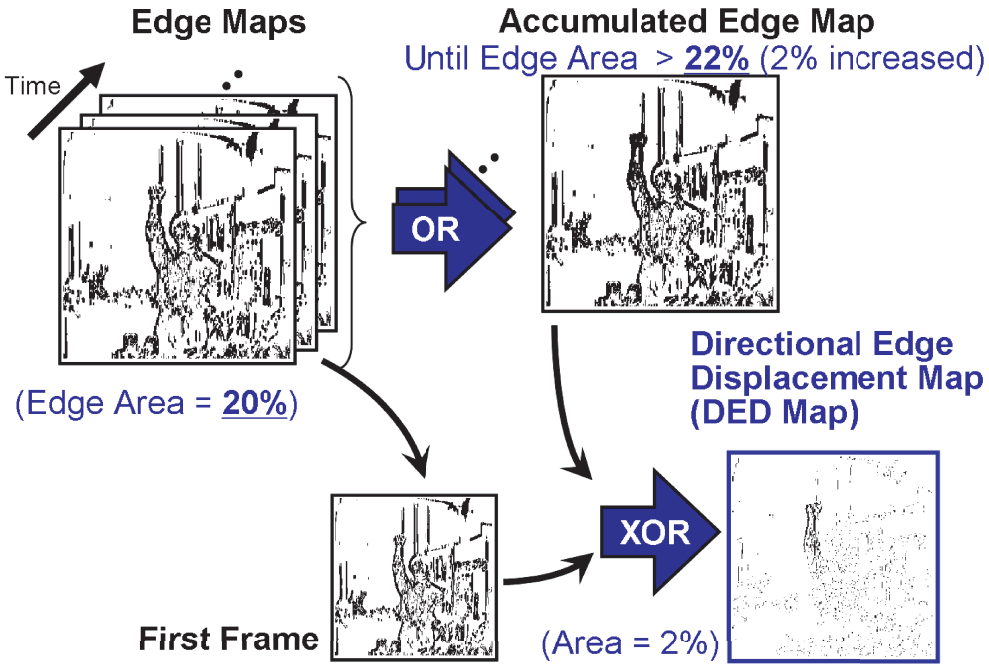


Figure 3.2: Process of DED edge map generation for the motion recognition algorithm.

Fig. 3.2 shows the process of DED edge map generation, which is similar with the DDEI generation process. Firstly, an edge map is generated from an image frame by setting the determined number so that the number of edge flags remaining becomes a certain percentage of the total number of pixels in the frame. This edge map is called the source edge map. Then, the edge map is generated from the next-frame image with the same determined number and logic OR is

taken between the edge map and the source edge map. The combined edge map thus produced is called the accumulated edge map. From the third frame image, edges are detected with the same edge percentage, which is then merged with the accumulated edge map by taking OR again. In this manner, the accumulated edge map is updated at every incoming image frame. Such a procedure is repeated until the edge count in the accumulated edge map increases to a certain value. Then the resultant accumulated edge map is compared with the source edge map and the difference is detected. Namely, Exclusive OR is taken between the accumulated map and the source map. The edge map produced in this manner is called the directional edge displacement map (DED map), in which only the edge flags due to the motion occurring during the image sequence are accumulated and remaining.

3.3 VLSI Implementation

3.3.1 System organization

A VLSI processor was accurately designed for the intelligent image processing algorithms described in §3.2. Figure 3.3 shows the overall architecture. It is composed of three main blocks: 100 × 100 DPS, 24 groups of local image processing (LIP) circuits, and a global image processing (GIP) unit. DPS is used to capture the image. 24 groups of processing elements are used for parallel filtering or local feature extraction. The complex interconnection between DPS and 24 LIP circuits is markedly simplified by the block-readout architecture, which has been explained in detail in §2.3. Because the size of each filter kernel is 5 × 5, a 100 × 100 original image is converted to 96 × 96 directional edge map(s) with the maximum convolution value at every pixel site. Then, the directional edge flags with their convolution values are stored in static random access mem-

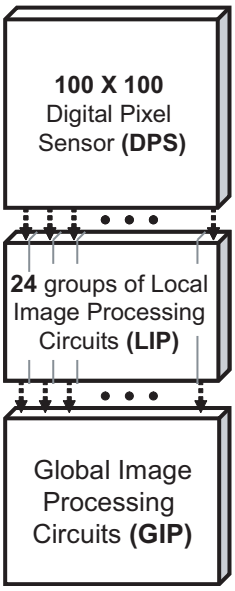


Figure 3.3: Architecture of this early-visual-processing VLSI processor

ories (SRAMs) in the GIP unit. An efficient binary sorting algorithm adapted for hardware implementation is employed to perform the GFE processing so that the global thresholding process for all 9216 11-bit data can be accomplished in only 11 cycles. Such a fast processing capability is contrasted with those complex algorithms used in software (99).

3.3.2 Digital-pixel-sensor and local image processing (LIP) circuit

The architecture of the digital-pixel-sensor and the block-readout method have been introduced in detail in §2.3. This section only gives a brief description of these two parts. Figure 3.4 shows the block diagram of both DPS and LIP. To keep the circuits rational in the interconnection and chip areas, the kernel calculation should be performed in a line-parallel manner. Thus, 24 LIP circuits are employed which can perform the filtering or LFE for one row (96 pixel sites)

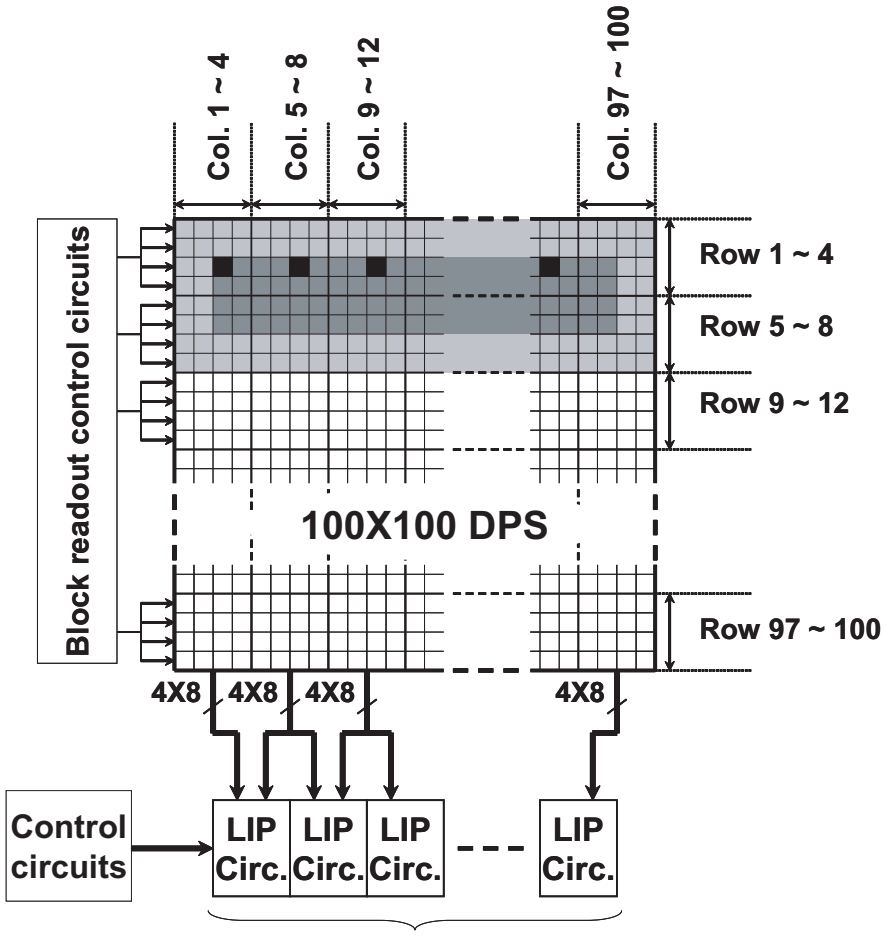


Figure 3.4: Block-readout method for DPS

in 4 iterations.

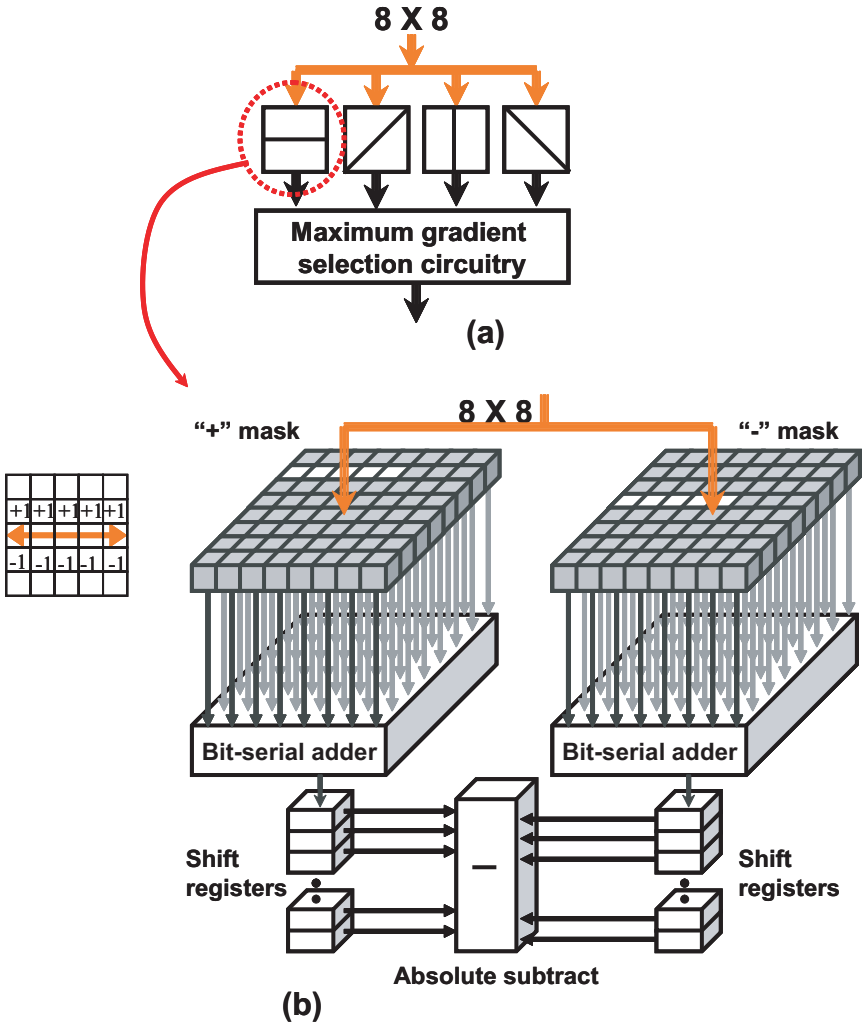


Figure 3.5: Local image processing circuits

Fig. 3.5(a) shows the implementation of a single LIP circuit shown in Fig. 3.4 in more detail. By using the block-readout architecture, 8×8 1-bit data from a selected block in the DPS are transferred to each LIP circuit at each clock cycle. An LIP circuit has four kernel processing circuits, which contains 2 masks: one for the plus component, the other for the minus component. The detail structure

of the horizontal kernel processing circuits is shown in Fig. 3.5(b). In each mask, there are two types of pixels, which are indicated as "gray" and "white". In the "gray" pixels, the bit data are blocked and converted to zero, while the data bits in the "white" pixels are allowed to pass through to generate kernel filter patterns for a particular pixel site. The plus and minus coefficients are generated by two separate masks. The data bits in the "white" pixels are transferred to the bit-serial adder. The LSB of the adder's output is sent to the "shift registers" for accumulation. After these shift registers collect all bits of the summation results, the "absolute subtract" circuit calculates the absolute difference between the "+" component and the "-" component, which is an 11-bit convolution value for this direction. Then, the maximum gradient selection circuitry that is designed employing the two-dimensional bit-propagating scheme (56) determines the edge direction at the pixel site. In this manner, the edge direction at this pixel site is determined with its corresponding convolution value. The results are stored in SRAM arrays in the GIP unit for future processing of global feature extraction.

3.3.3 Global image processing unit

3.3.3.1 Circuits design

The efficient binary sorting algorithm developed from the rank-order-filter algorithm for the global feature extraction was explained in detail in §2.3, this section starts from the introduction of the overall architecture of the GIP unit.

Figure 3.6(a) shows the configuration of the GIP unit, which is composed of 9216 (96 pixels × 96 pixels, one PE for processing one pixel site) processing elements, a full parallel 9216 1-bit input adder, and a comparator. This GIP unit has two system modes, which is explained in detail later. One is GFE mode; the other is frame processing mode. In each cycle, the outputs from the 9216 processing elements are added by the 9216-input adder, from which the rank

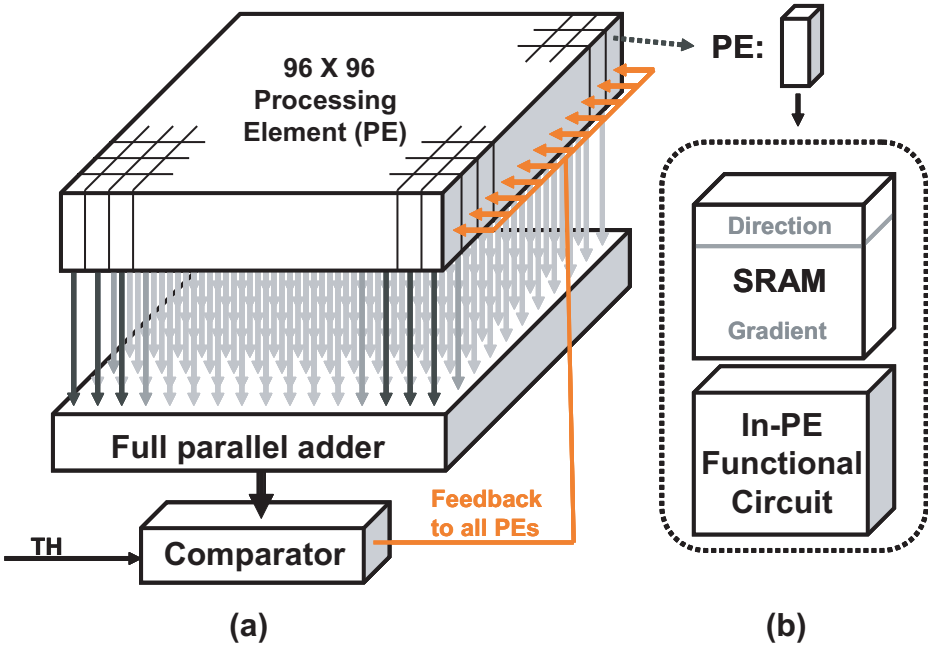


Figure 3.6: Global image processing circuits

order number (TH) is subtracted in the comparator. In GFE mode, the MSB of the subtraction result, which is "0" when the summation is larger than the rank order number (TH) and "1" otherwise, is inverted and fed back to each processing element for determining "FLAG" and "MARK" as described in the binary sorting algorithm in §2.3. While in the frame processing mode, two edge maps from two frames are combined by "OR" or "XOR" calculation in a pixel-parallel way, and the total number of the combination result is compared with the other threshold to determine whether it should be recorded or discarded. Figure 3.6(b) shows one processing element, which is composed of two parts: a 13-bit SRAM and a "In-PE functional circuit". The 13-bit SRAM stores 11-bit convolution value and 2-bit value for four directions ("00": horizontal, "01": +45°, "10": vertical, and "11": −45°).

Figure 3.7 shows the schematic of the "In-PE functional circuit", which con-

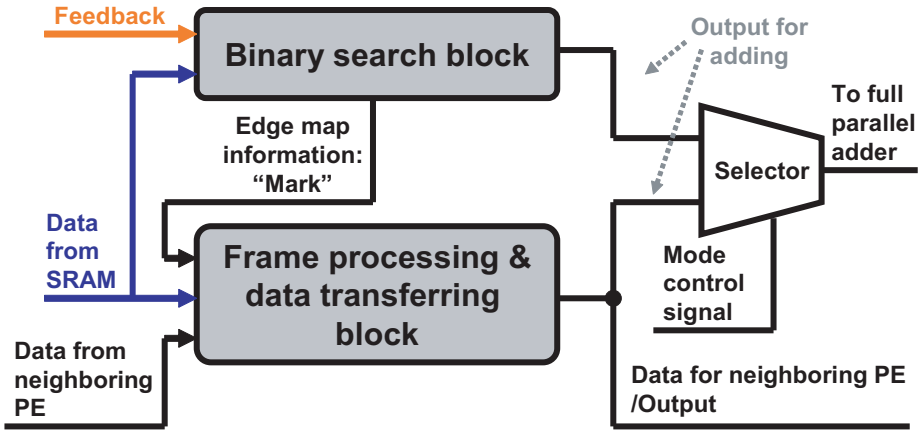


Figure 3.7: "In-PE" functional circuit

tains two main blocks: one is the "binary search block"; the other is "frame processing & data transferring block". The system mode is controlled by the signal "mode control signal" as shown in this figure. This signal decides the output of which block should be transferred to the full parallel adder for summation. The data from the SRAM in the same PE is transferred to both of the two blocks. Besides the image data that preserved in the SRAM, the "binary search block" also receives the "feedback" signal which is generated from the "comparator". In the GFE mode, it generates the proper signal for adding in order to perform the GFE as well as the final result, which appears as the edge map information: "mark" as shown in this figure. With many control signals, the "frame processing & data transferring block" has three main functions: data loading, pixel-parallel edge map processing, and shift the results out. It receives the data both from the SRAM, the "binary search block", and the result data provided by the neighboring PE, while providing proper data for adding or as result data to the next neighboring PE.

Figure 3.8 shows the schematic of the "binary search block" circuit in detail. The circuit is comprised of three parts: a "FLAG" logic circuit, a "MARK" logic

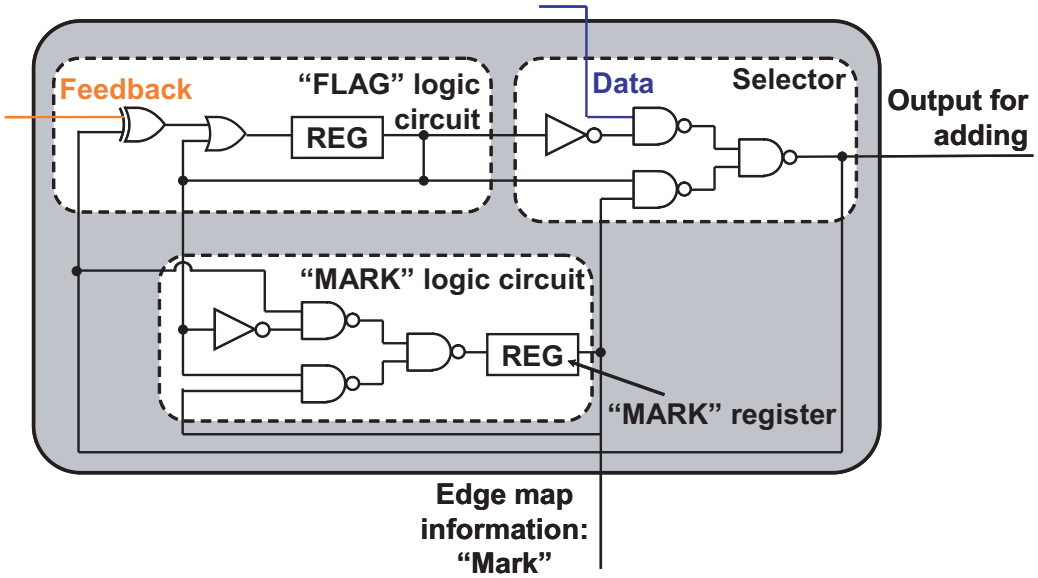


Figure 3.8: Binary search block in the "In-PE" functional circuit

circuit, and a selector. It processes the pixel data in a bit-serial manner according to the algorithm illustrated in §2.3. After 11 cycles, 4096 "MARK" values are decided and preserved in "MARK" register in each "mark decision" circuit. The data in these "MARK" registers are transferred out as the edge map information: "mark" and read out though the "frame processing & data transferring block".

Figure 3.9 shows the schematic of the "frame processing & data transferring block". It is composed of a multi-purpose register for recording the result and three combinational sub-blocks which are named as: "working mode select", "data select", and "function select" as shown in the figure. There are three control signals for the "working mode select" sub-block. When "LOAD_Data" is enabled, the function are further decided by the "data select" sub-block. If the "Load_Data_SRAM" is active, the datum from the SRAM is preserved in the register; while if the "Load_Data_SELF" is active, the datum in this register itself is reloaded. Such a design is employed because this register shares some control

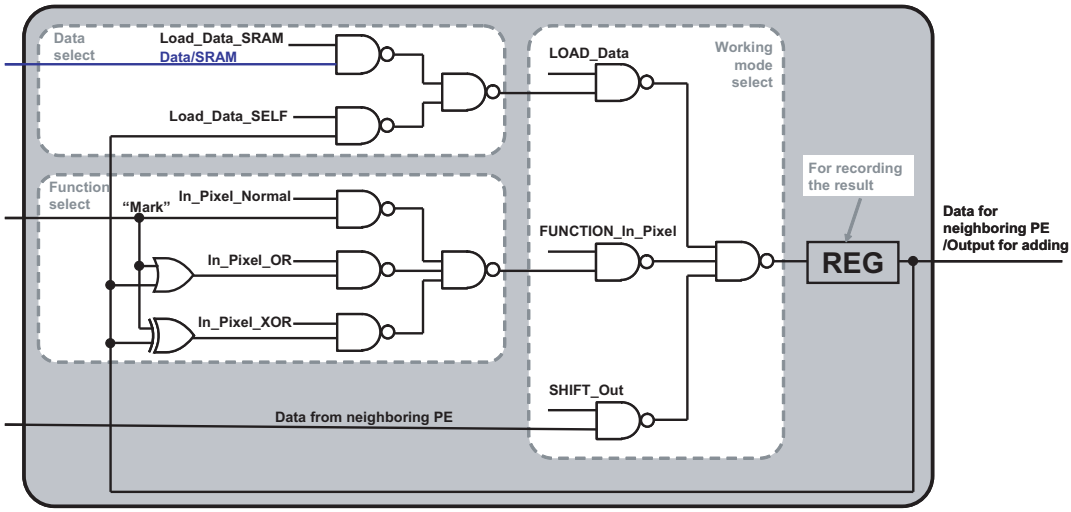


Figure 3.9: Frame processing & data transferring block in the "In-PE" functional circuit

signal with the two registers in the "binary search block" and it should keep the result during the GFE of a new frame. When the "FUNCTION_In_Pixel" is enabled, the function are further decided by the "function select" sub-block. If "In_Pixel_Normal" is active, the edge information: "mark" is directly recorded into the register. While if the "In_Pixel_OR" or "In_Pixel_XOR" is active, the "mark" preserved in the register and the "mark" from the "binary search block" are processed with the "OR" or "XOR" computation correspondingly. Finally, when the "SHIFT_Out" is enabled, all registers in this "frame processing & data transferring block" of the PEs are connected into 24 shift register chains, which can shift circularly to output the results. The output of the last register is feedback to the first register in each chain so that the results are not affected in the output process. The data preserved in the register is output for adding when "FUNCTION_In_Pixel" is enabled or as the data for neighboring PE of the "SHIFT_Out" is enabled. A summary of the instruction codes is shown in Table 3.1. These eight-bit codes are created by connecting the logic values of the eight

control signals in this order: LOAD_Data, FUNCTION_In_Pixel, SHIFT_Out, Load_Data_SRAM, Load_Data_SELF, In_Pixel_Normal, In_Pixel_OR, and In_Pixel_XOR.

Table 3.1: Summary of the instruction codes.

Instruction code	Function
100 10 XXX	Load data from SRAM
100 01 XXX	Keep the prior value
010 XX 100	Load the edge map information: "Mark"
010 XX 010	Perform the OR calculation
010 XX 001	Perform the XOR calculation

3.3.3.2 Operation

For the global feature extraction used in static image recognition, the "system mode" is constantly kept to be GFE mode to extract the global feature for each frame and records the results to the multi-purpose registers; then the results are transferred out of the circuits. For generating a DDEI, the first frame is processed in system mode GFE and the results are recorded into the multi-purpose registers. From next frame, after the GFE of this new frame, instead of recording the result for directly the multi-purpose registers, the system mode is changed to frame processing mode and a pixel-parallel "XOR" calculation is performed. The number of edge marks in the DDEI results are summed by the full-parallel adder and compared with the corresponding threshold. If it is less than the threshold, which means the object has not moved enough for generating a DDEI, the system will restore the original result by applying another "XOR" for the calculated results and the edge map of the new frame, waiting for evaluating the next frame. If it is equal or larger than the threshold, which means the object has moved enough for creating a DDEI, then the DDEI results are shifted out and the edge map generated for this frame is recorded into the multi-purpose registers for

generating next DDEI. For generating a DED map for motion recognition, again, the first frame is processed in system mode GFE and the results are recorded into the multi-purpose registers. From next frame, after the GFE of this new frame, instead of recording the result directly to the multi-purpose registers, the system mode is changed to frame processing mode and a pixel-parallel "OR" calculation is performed to calculate the accumulated edge map. The number of edge marks in the accumulated edge map are summed by the full-parallel adder and compared with the corresponding threshold. If it is less than the threshold, which means the movement is not enough for generating a DED map, the system will just keep the accumulated edge map, waiting for the next frame. If it is equal or larger than the threshold, which means the movement in the scene is large enough for creating a DED map; then a XOR is applied to the accumulated edge map and the edge map generated for this frame. The resultant DED map results are shifted out and the edge map generated for this frame is recorded into the multi-purpose registers for generating the next DED map.

3.4 Chip Design and Measurement Results

A VLSI chip was designed in a 65-nm 12-metal CMOS technology. Figure 3.10(a) shows the entire layout and Figure 3.10(b) shows a photomicrograph of the fabricated chips. The top 3 levels of metal are used for light shielding. To achieve good performance and high integrity, the photodiode as well as analog circuits inside each pixel are designed using the 180nm process, while all digital circuits are designed using the normal 65-nm process. This chip includes all three blocks described in §3.3. The specification of this chip is summarized in Table 3.2.

Figure 3.11 shows the environment of the measurement. This chip is mounted on a socket and put beneath the lens. Two 100W lights are used as the light

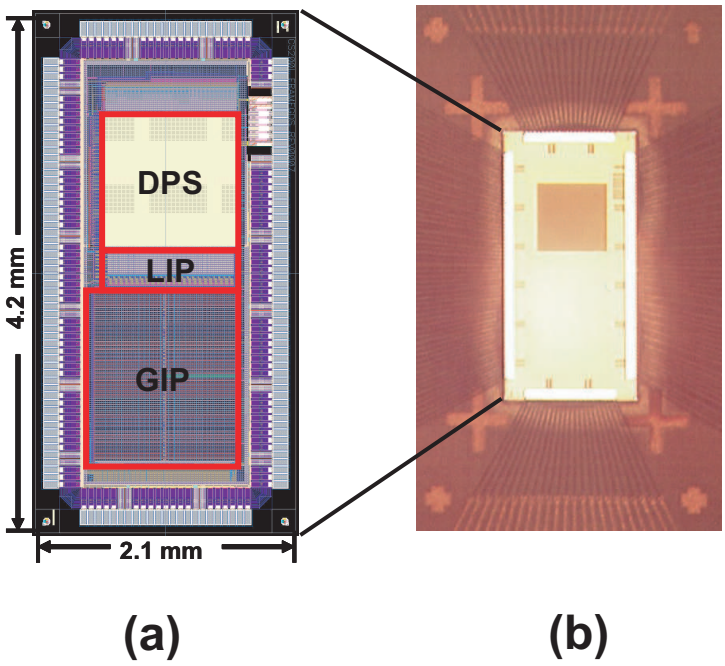


Figure 3.10: Layout and photomicrograph of this chip

Table 3.2: Specifications of this chip. ($\star$ a typical operating condition)

Technology	65-nm CMOS, 1-poly, 12-metal
Core size (mm^2)	1.3 × 3.4
Number of pixels	100 × 100
Pixel pitch (μm^2)	10.6 × 10.6
Fill factor	14%
Transistor count	4.3 million
Supply voltage	1.3 V($\star$)
Clock freq.	20 MHz($\star$)
Power	13 mW($\star$)
Frame rate	40 f/s($\star$)

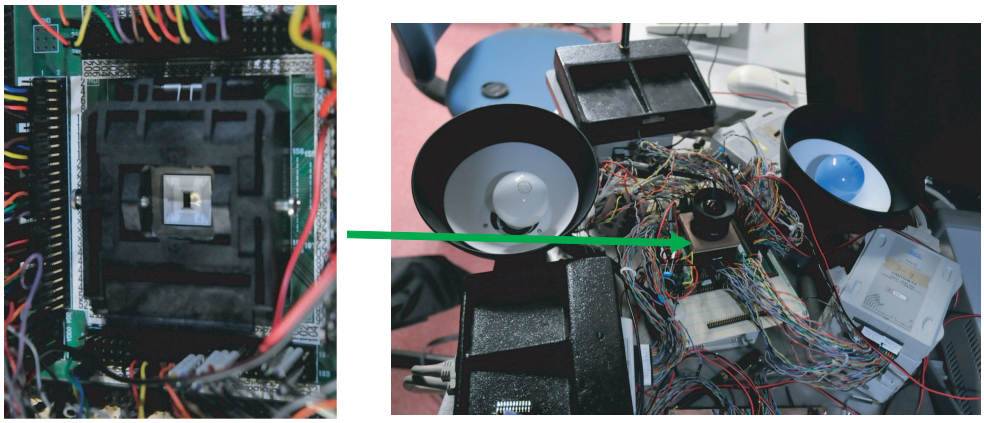


Figure 3.11: Measurement environment.

source, and image is put about 20 cm from the lens.

Figure 3.12: An image taken by the DPS

Figure 3.12 shows an image taken by this sensor with a shutter speed of 24ms. Since this foundry process which was designed for regular CMOS logic circuits and analog circuits is not tuned for developing photo-diode and DRAM used in each pixel of this image sensor, there are some noises in this image. However, the purpose of this work is to develop the fine-grained parallel image processing circuits for the image sensor to achieve efficient implementation. Furthermore, such noises can be greatly eliminated in our image processing algorithms by employing

the global feature extraction.

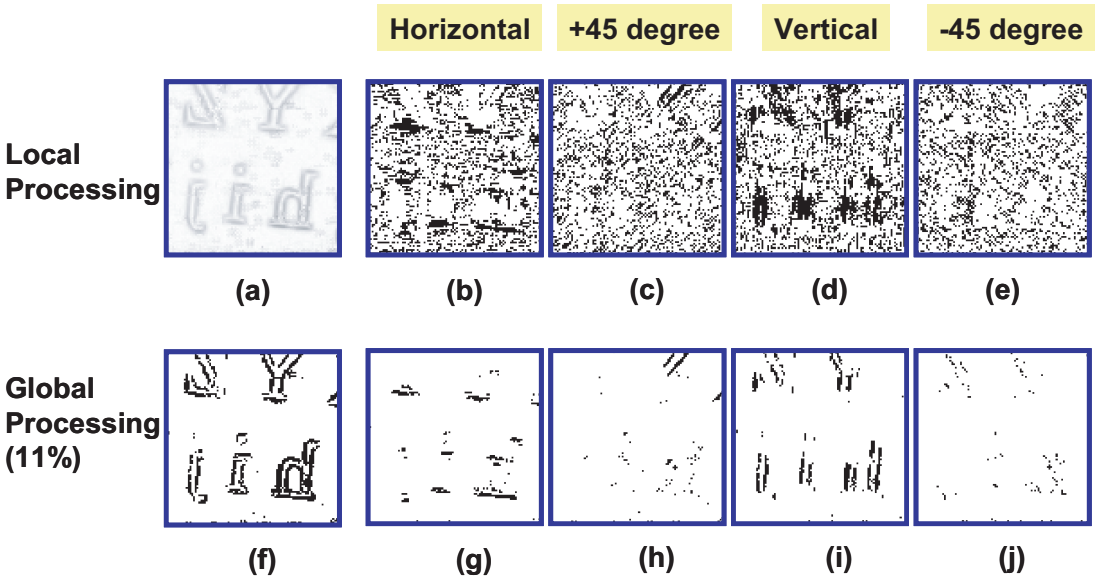


Figure 3.13: Measurement results of the global feature extraction function. (a) edge intensity, local feature edge map in (b) horizontal, (c) +45°, (d) vertical, and (e) −45°, (f) merged edge map after global feature extraction, and global feature edge map in (g) horizontal, (h) +45°, (i) vertical, and (j) −45°.

Figure 3.13 shows the measurement results for both the local image processing and global feature extraction. The data after the local image processing are shown in Figure 3.13 (a-e), in which (a) showing the intensity of each pixel site and (b-e) showing the direction of each pixel site. The data after the global feature extraction are shown in Figure 3.13 (f-j), in which (f) showing the merged edge map of four direction and (g-h) showing the edge map for each direction. The threshold was set at 11%. Most of the noises in the original image have been eliminated in the global feature extraction. If distribution histograms are produced from these edge maps, they play essential roles in intelligent image recognition applications as shown in refs. (63) and (64).

Figure 3.14 shows the measurement results for the self-adapted differential

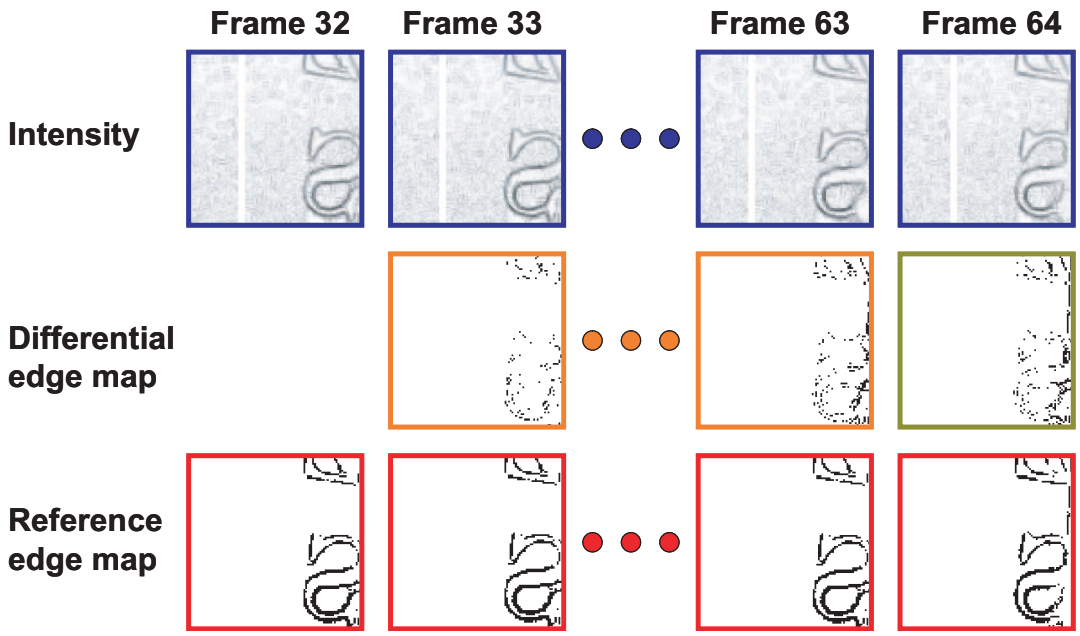


Figure 3.14: Measurement results of self-adapted differential edge map generation function.

edge map generation. In frame 32, a new reference edge map is generated. From frame 33 to frame 63, since the movement is not significant in the scene, each time after generating the differential edge map, the edge map is recovered to the reference edge map of frame 32. In frame 64, since the number of edges in the differential edge map exceeded the differential threshold, which means there are significant movement in the scene, after generating the differential edge map, instead of recover the original reference edge map generated in frame 32, the edge map of frame 64 is generated as a new reference edge for the following differential edge map. In the differential edge map generated in frame 64, more edges are from the moving part and the background has been reduced greatly. In this experiment, the global threshold is set to be 5.5% and the differential threshold is set to be 2.8%. Here only the right half of scene has objects as image since some problems in layout affected the stability of the "XOR" function in the left

part of the scene.

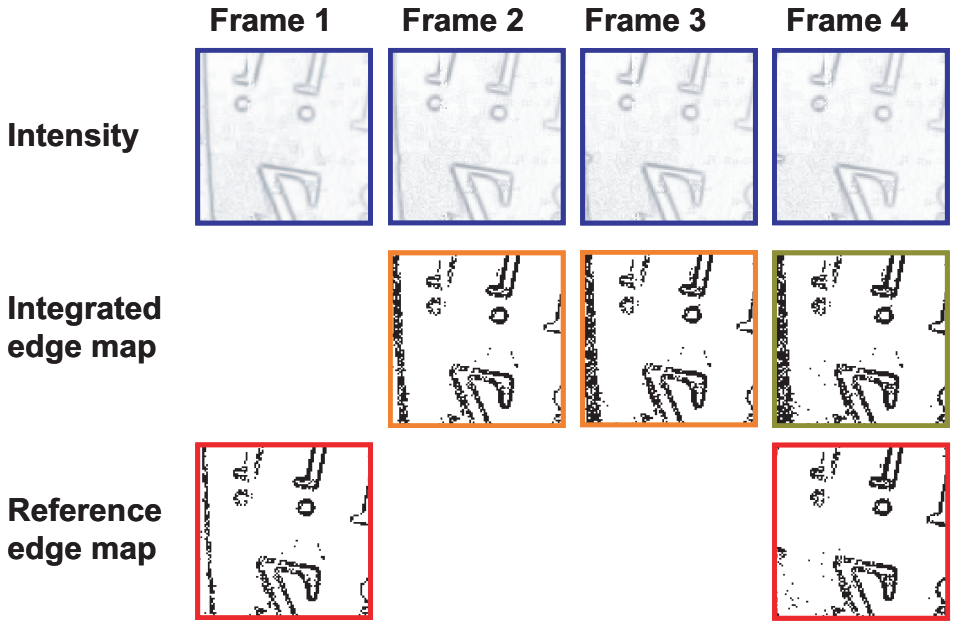


Figure 3.15: Measurement results of self-adapted integrated edge map generation function.

Figure 3.15 shows the measurement results for the self-adapted integrated edge map generation. Four frames are shown in this image and the global threshold is set to be 11% and the integrated threshold is set to be 16.5%. In the first frame, it will output an edge map as the first reference edge map. In the second frame, the second edge map is integrated with the first one to make an integrated edge map, since the total number of edges in this integrated edge map is smaller than the integrated threshold, which means the movement in the scene is not enough, the image sensor will keep integration. In the third frame, the third edge map is also added to the integrated edge map. In the fourth frame, after integrating the fourth edge map to the integrated edge map, the total number become larger than the integrated threshold, which means the movement in the scene is significant enough for the following processing. Then the integrated edge map is output and

the fourth edge map is also output as the second reference edge map for the next integration. In this experiment, the global threshold is set to be 11% and the integrated threshold is set to be 16.5%.

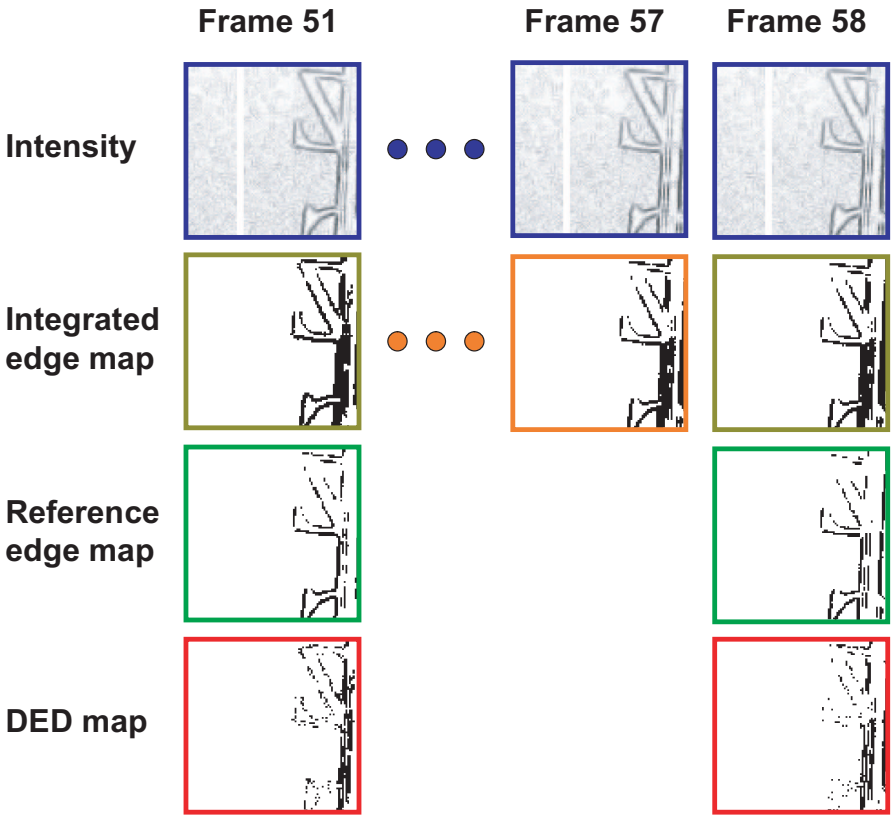


Figure 3.16: Measurement results of DED map generation function.

Figure 3.16 shows the measurement results for the DED map generation. However, to achieve a compact design, this algorithm has been modified that the DED map is generated by processing XOR between the integrated edge map and the newly generated reference edge map, instead of using the prior reference edge map. In frame 51, the total number of the edges in the integrated edge map exceeded the integration threshold; then the DED map is generated by performing XOR between this integrated edge map and the newly generated

reference edge map in this frame. From frame 52 to frame 58, the movement is not significant enough to generate DED map, so the edges are just collected in the integrated edge map. In frame 58, again, the total number exceeded the integration threshold; the same processing in frame 51 is performed. In this experiment, the global threshold is set to be 5.5% and the integrated threshold is set to be 11%.

3.5 Summary

A digital-pixel-sensor-based early-visual-processing VLSI processor for real-time intelligent image processing has been developed. By combining the block-readout architecture for DPS and parallel processing elements, the latency of local image processing has been markedly reduced. By adapting the rank-order filter algorithm to hardware implementation, global feature extraction is performed in a very fast manner. The enhancement in the functionality of processing element improves the programmability of the processor greatly. As a result, such a chip can handle multiple algorithms efficiently. A prototype chip was designed in a 65-nm 12-metal CMOS technology. The measurement results show that this VLSI processor can achieve all expected functions.

Chapter 4

A Real-Time Image Recognition System Using a Global Directional-Edge-Feature Extraction VLSI Processor

4.1 Introduction

Real time image recognition has become an emerging research area for various applications such as intelligent robot control, automotive vehicle control, video surveillance, natural human-computer interface, smart game controller, and so forth. The traditional software-based recognition algorithms running on general purpose processors are computationally very expensive, and building real-time response systems at rational costs and power-consumption is not practical. To enhance the performance, a number of VLSI chips have been designed for specific image processing operations such as feature extraction and template matching (93; 94). Instead of designing chips for already available algorithms, a more effective way is to make algorithms themselves more adaptive to VLSI hardware implementation. The face detection core in (118) is designed with a particular

face detection algorithm. The network-on-chip in (16) is equipped with a special visual-attention-based object recognition algorithm.

Being inspired by the biological principle (11), a VLSI-implementation friendly image recognition algorithm was proposed and has been successfully applied to pattern recognition (63). There are three steps in this algorithm as illustrated in Fig. 4.1: feature extraction (step I), vector generation (step II) and template matching (step III). Some VLSI chips have been designed for this architecture such as the VLSI processor for feature extraction and vector generation (74) and template matching VLSI chips in both analog and digital domains (59). Thanks to the parallel computation in these chips, the processing speed of the recognition algorithm has been considerably accelerated, making real-time image recognition feasible. However, because most of these chips were designed for only one or at most two steps for this algorithm, it is difficult to build a fast response recognition system since the latency caused by the large quantity of data transfer between every consecutive two chips. The most serious bottleneck occurs between the image sensor chip and the feature extraction chip.

In this paper, real-time image recognition is demonstrated by building a simple test system using specialized VLSI chips. This system has two main components: one is a digital-pixel-sensor-embedded global directional-edge-feature extraction VLSI processor which was introduced to solve the most serious bottleneck in step I; the other is a vector-generation and template-matching processor (step II & III) which was implemented on an FPGA development board. The monitor control for displaying the recognition results is also implemented on the FPGA. The architecture of the feature extraction VLSI processor has been presented in (115; 116). However, its function was only verified by NANOSIM simulation. In this work, a real working chip was used for the first time in conjunction with the auxiliary functions implemented on an FPGA to build a very-low latency

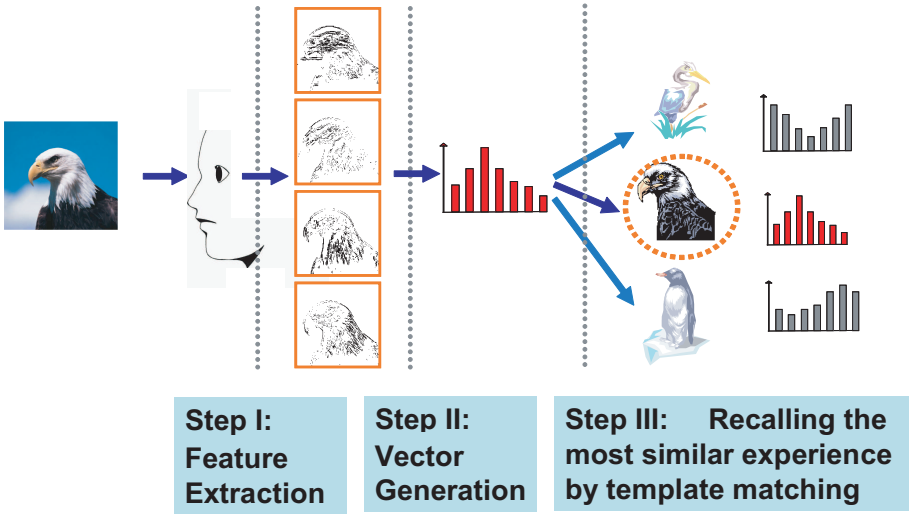


Figure 4.1: VLSI-implementation friendly image recognition algorithm.

real-time image recognition system. Furthermore, the capability of the system for automatic adaptation to selecting more significant features has been experimentally demonstrated.

4.2 VLSI-Implementation Friendly Recognition Algorithm

The algorithm has two modes of operation: the learning mode and the recognition mode. Step I and step II are common for both modes, while step III depends on which mode is working.

4.2.1 Global directional-edge-feature Extraction

The feature extraction algorithm is based on four types of directional edge filtering (horizontal, +45°, vertical, and −45°). It is composed of both local feature extraction and global feature extraction. During local feature extraction, the convolutions between the 5×5 partial image centered at each pixel site and four

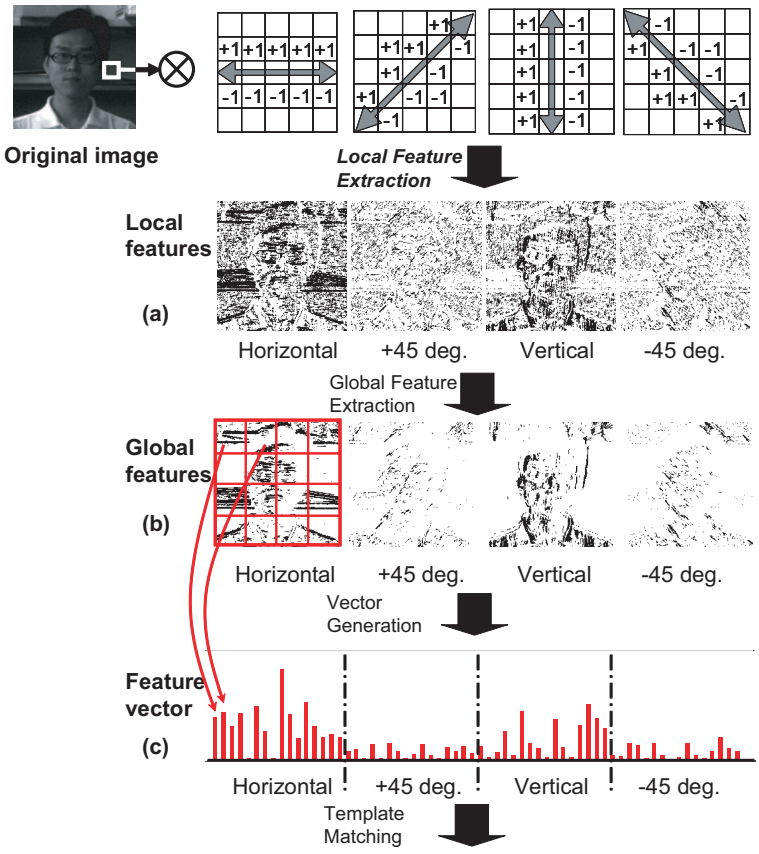


Figure 4.2: Feature extraction and vector generation.

5×5-pixel filtering kernels (one for each direction) are calculated. For each pixel site, only the maximum convolution value is preserved as the gradient value with its direction as the directional edge flag. Fig. 4.2(a) shows the edge flags obtained in the local feature extraction stage. In order to develop a self-adaptive algorithm that can select only salient features automatically, a global feature extraction process was developed. In this process, only a predetermined number of significant edge flags which has larger gradient values than the others are retained. Fig. 4.2(b) shows four global feature maps when only 40% of more significant edge flags are retained.

4.2.2 Feature vectors

Although feature maps very well represent image features, the amount of data is still massive and dimensionality reduction is essential for efficient processing. 64-dimension feature vectors are generated from feature maps by taking the spatial distribution histograms of edge flags. Fig. 4.2(c) illustrates the generation procedure for a feature vector called averaged principle-edge distribution (APED) vector. In this process, each feature map is equally divided into 4×4 cells and the edge flags in each cell are counted and each element in the feature vector indicates the number of edge flags within the corresponding cell.

4.2.3 Learning and recognition

In the learning mode, the generated APED is recorded as a new template. Since a same object is displayed in different conditions for better learning, the same object can have several different template vectors. In the recognition mode, the generated feature vector is compared with all template vectors that the system memorized during the learning mode. The template vector yielding the minimum distance (the Manhattan distance is utilized in this work) is detected as the maximum-likelihood case to the input.

4.3 System Implementation

4.3.1 Architecture of the system

Fig. 4.3(a) shows the whole architecture of the real-time image recognition system. Global directional-edge-feature extraction VLSI processor is used for image capture and feature extraction. Then the extracted features are transferred to the FPGA development board, in which the vector generation and learning/recognition are performed, and the results are displayed on a monitor. Since

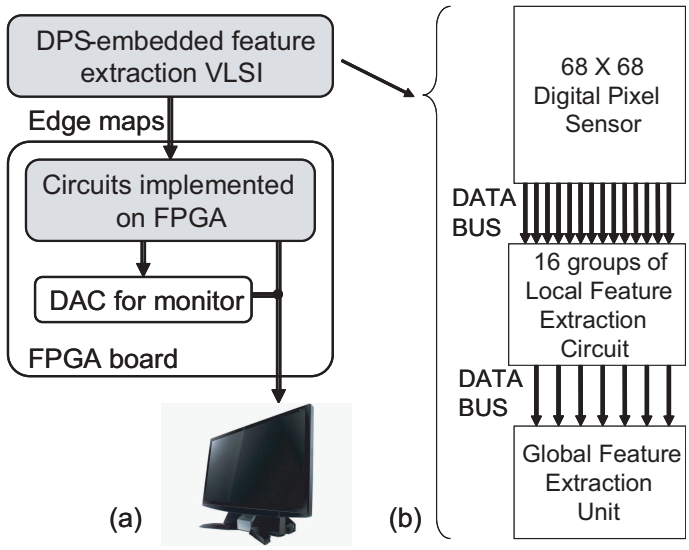


Figure 4.3: Architecture of the recognition system.

the extracted features are edge maps and the data size is much smaller than the original image size, the latency due to the data transfer has been decreased dramatically.

4.3.2 Global directional-edge-feature extraction VLSI

Fig. 4.3(b) shows the architecture of the global directional-edge-feature extraction VLSI. It is composed of three main blocks: a 68×68 digital-pixel-sensor (DPS), 16 groups of local feature extraction (LFE) circuits, and a global feature extraction (GFE) unit. DPS is used to capture the image. Sixteen groups of processing elements are used for parallel local feature extraction. The complex interconnection between DPS and 16 LFE circuits is markedly simplified by the block-readout architecture introduced in (83). Because the size of each filter kernel is 5×5, a 68×68 original image is converted to four 64×64 directional edge maps with the maximum convolution value preserved at every pixel site. Then the directional edge flags with their convolution values are stored in SRAMs in the GFE unit.

Rank-order-filter algorithm adapted to hardware implementation is employed in the GFE unit so that the sorting process for all 4096 11-bit data in the global feature extraction can be accomplished in only 11 bit-comparison cycles. Such a fast processing capability is contrasted with those complex algorithms used in software. Through the GFE unit, the four edge maps are compressed into a full edge map, which is a superposition of all four edge maps along with the directional information at each pixel site. More details about this VLSI architecture are available in (116).

4.3.3 Circuits implemented on FPGA

Fig. 4.4 shows the circuits implemented on FPGA. The compressed full edge map information from the feature extraction VLSI is stored into an SRAM on the FPGA. After the data of one frame is transferred completely, the data are decomposed into four edge maps and stored in four edge-map SRAMs separately. An extra copy of the four edge maps and the full edge map are also stored in a special purpose SRAM for display. In the APED vector generation circuitry, the four edge flags on the same pixel location are read out simultaneously and accumulated in parallel by individual accumulators, thus generating APED vector components. Then the results are sent to present APED vector recording circuitry.

The function of the system after APED vector generation changes depending on the mode of operation, which is controlled by a toggle switch. In learning mode, a push button is used to trigger a learning process. Each time the button is pushed, the APED vector generated at the moment is stored in the memory as a template. In the recognition mode, every APED vector generated is matched with all template vectors stored in the memory for recognition. Fig. 4.5 shows the architecture of the template matching circuitry. The elements of 64 APED

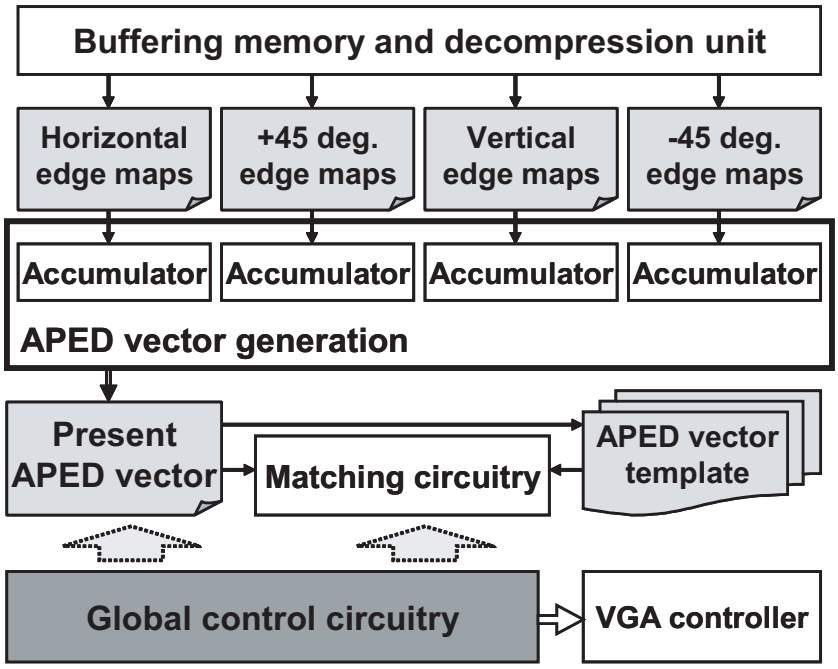


Figure 4.4: Circuits implemented on FPGA.

template vectors are read out in parallel and sent to the sum-of-absolute-difference (SAD) circuits. At the same time, the corresponding component of the present APED vector is broadcasted to all SADs. The output of each SAD is accumulated and compared to find the minimum to yield the recognition result.

4.4 Experimental Results

Fig. 4.6 shows a photomicrograph and specifications of the global directional-edge-feature extraction VLSI processor fabricated in a 0.18-μm 5-metal CMOS technology. In order to achieve a compact layout and high-performance operation, the chip was entirely designed by hand-layout. This chip can achieve 5000 frame/s at 60 MHz if the light integration time for photodiodes is not considered, with a power consumption of 67 mW at 1.8 V.

Fig. 4.7 shows three edge maps that demonstrate the automatic critical fea-

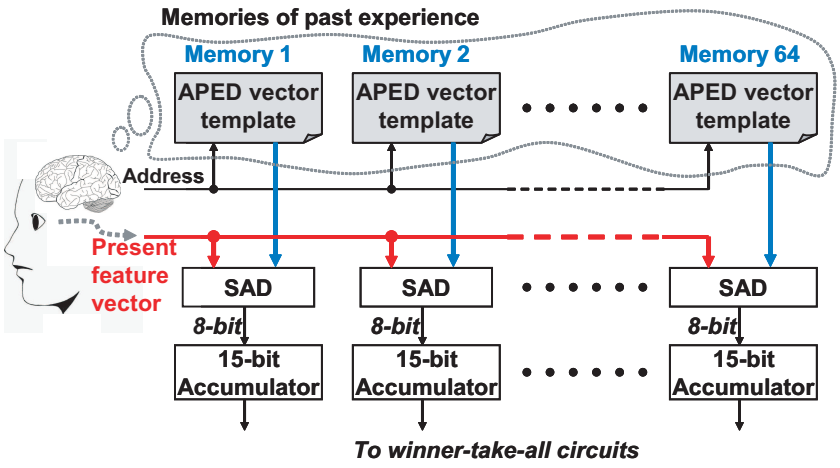
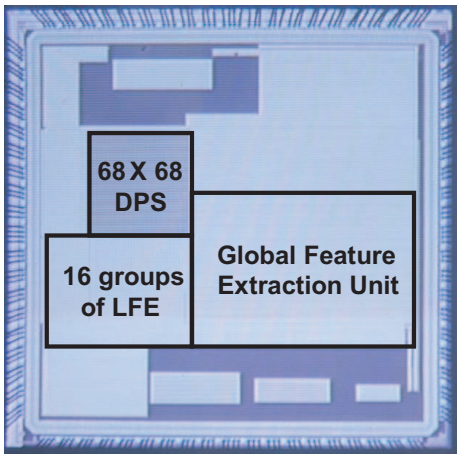


Figure 4.5: Template matching circuitry.



Chip specifications

Technology	0.18µm CMOS, 5-layer AL
Transistors	1.6M
Core size	11 mm²
Supply voltage	1.8 V
Clock freq.	60 MHz
Power	67 mW (*)
Frame rate	5000f/s (*)

Figure 4.6: Photomicrograph and specifications of the fabricated chip (*the light integration time for photodiodes is not considered).

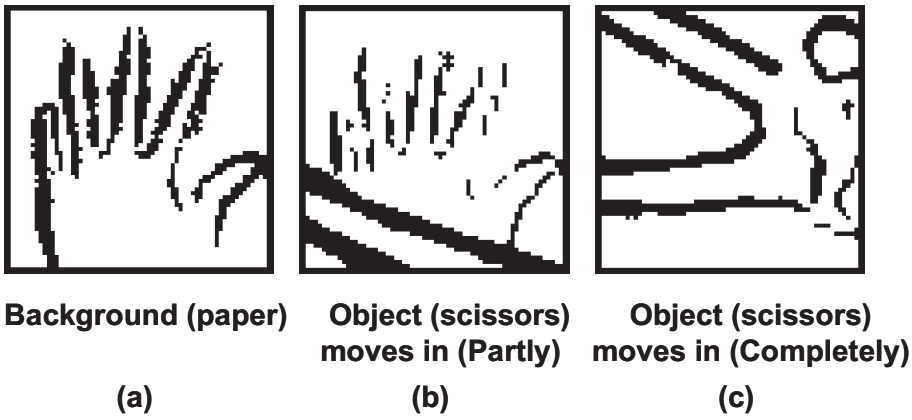


Figure 4.7: Automatic critical feature adjustment.

ture attention capability of the chip. Here, four separate directional edge maps are merged to one. In this experiment, an image of a hand is used as the background. Fig. 4.7(a) is the merged edge map when only the background is shown in the scene. When another object begins to appear in the scene (Fig. 4.7(b)), the number of edge flags from the background image decreases, because it is in a little out of focus condition. When the object comes into the scene completely (Fig. 4.7(c)), the whole background disappears and only the edge features from the object remain because it is more sharply focused than the background image.

Fig. 4.8 shows measured waveforms from the system. The top signal goes from 0 to 1 when the extracted features of one frame are completely transmitted to the FPGA. Then the decompression circuits generate four edge maps from the full edge map. The time for decompression is 38 μs. The signal in the middle goes from 0 to 1 when the four edge maps are generated and goes from 1 to 0 when an APED vector is generated. The time for APED vector generation is 38 μs. The bottom signal goes to 0 when template matching is being performed. Thanks to the parallel architecture in template matching, the processing time is less than 2 μs. Restricted by the data transfer between the VLSI chip and the FPGA board, the feature extraction chip was operated at 20 MHz, resulting in

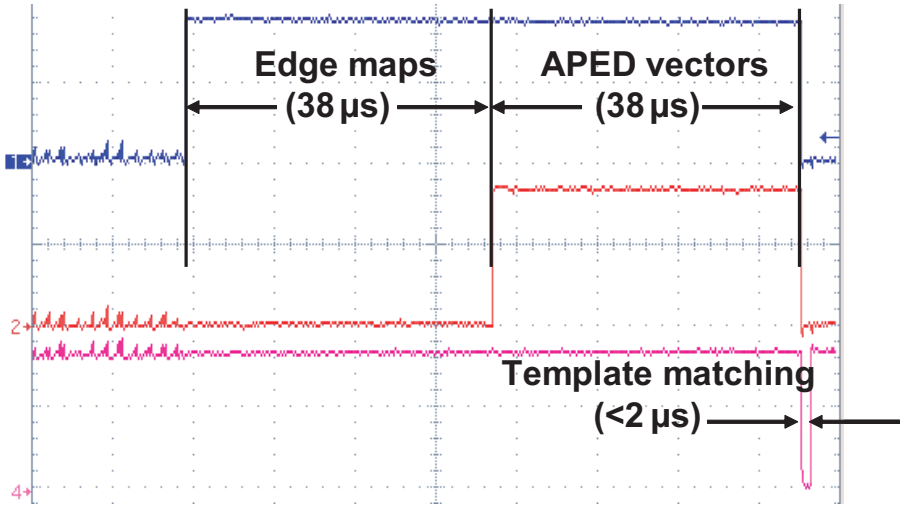


Figure 4.8: Measured waveforms showing the processing time.

the feature extraction time of 482 μs. The data transfer time is 346 μs. Therefore, the latency between the image capture and the final recognition is only 906 μs, which is fully compatible to time critical applications.

Fig. 4.9 shows the demonstration system. It can learn 64 template vectors. Most memories on FPGA were used for displaying the results. The global directional-edge-feature extraction VLSI processor is mounted under the lens. In the monitor, the top five square images are the merged edge map and edge maps in horizontal, $+45°$, vertical, and $-45°$ from left to right. Below these edge maps are two APED vectors, the present APED vector on the left, and the most similar APED vector in the memory on the right. Finally, all edge maps in the memory are displayed at the bottom with a color mask on the most similar. Using this system, a series of simple hand gesture recognition experiments were performed, in which hand gestures of eight different persons were matched with 40 templates for five gestures generated from one person. The recognition rate was almost 100% when the hand was posed about the same position.

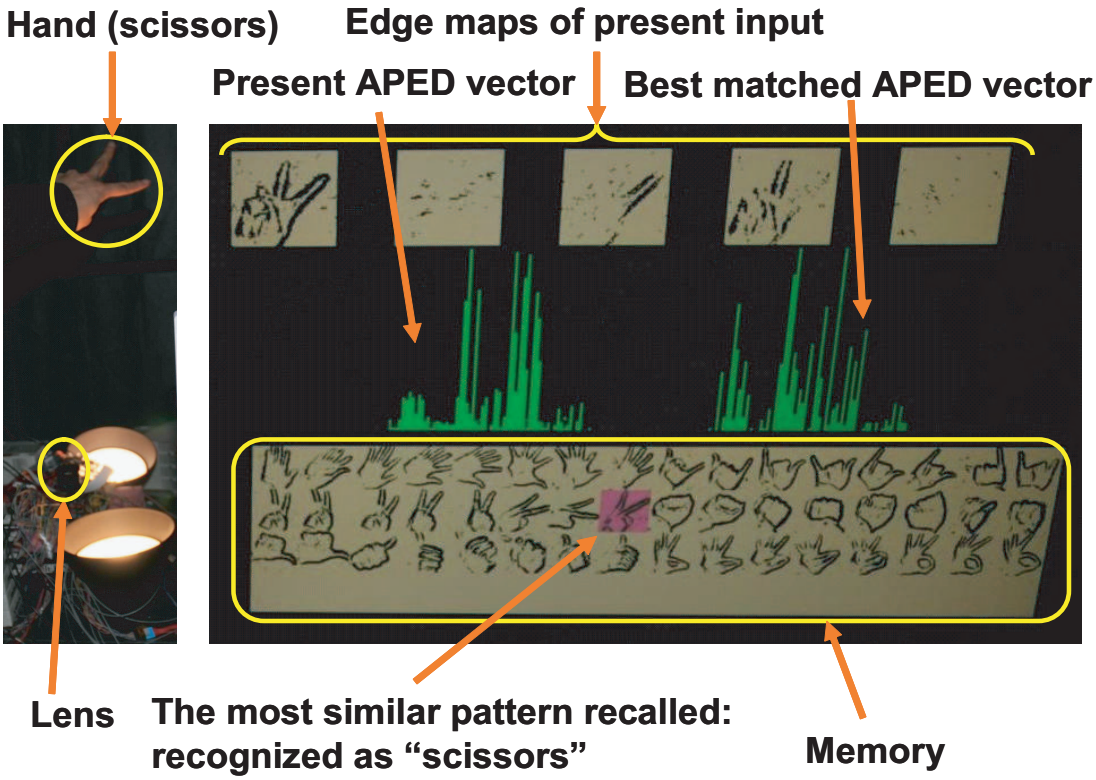


Figure 4.9: Operation of the demonstration system.

4.5 Summary

A real time image recognition system has been developed. This system is based on a VLSI-implementation friendly image recognition algorithm. By using the global directional-edge-feature extraction VLSI processor, the latency between the image capture and the final recognition as small as 906 μs has been achieved. The merit of the global feature extraction algorithm that it can focus on more significant features automatically has also been experimentally verified.

Chapter 5

Directional-Edge-Based Object Tracking Employing On-Line Learning and Regeneration of Multiple Candidate Locations

5.1 Introduction

Object tracking is the critical task in many practical applications such as video surveillance, human-computer interface, vehicle navigation, and robot control. For a robust object tracking system, there are many challenges: illumination variation, object size variation, partial occlusion, and object deformation. A number of tracking algorithms have been proposed (119) including template matching (120), "mean shift" tracking (121), active contour, and statistical approaches such as Kalman filters and particle filters (122).

However, while each algorithm demonstrating good performance in its particular conditions, a robust solution with reasonable calculation cost is still far from being completed. Each approach has some weaknesses. Template matching can not deal with the object deformations of the moving objects very well. Mean

shift is vulnerable to the variation in illumination. Active contour needs off-line learning, i.e., which is not always available in practice. Statistical approaches such as Kalman filters and particle filters need a lot of computation to achieve good performance.

In contrast to the limited performances of existing tracking algorithms, the object tracking abilities in most animals are excellent. Although the mechanism of the visual perception system has not thoroughly been understood, many researches on this topic give a lot of inspiring results (11). The work in (11) reveals that the visual perception in animals relies heavily on directional-edges for both static object and moving object. Based on such a biological principle, many directional-edge-based object recognition algorithms have been proposed and give good performance (123). However, such a concept has not been widely used in object tracking.

Regarding the calculation time, since real-time performance is usually demanded, sometimes hardware is developed specifically to accelerate the processing speed. For example, since the computation cost of the particle filter is very heavy, FPGA-based VLSI architecture has been designed to dramatically enhance the processing speed (124). But in many cases, tracking algorithms are not tuned to be efficiently implemented in VLSI hardware.

Therefore, the purpose of this study is to develop a VLSI-implementation-friendly object tracking algorithm that is robust in various circumstances. Inspired by the biological research in (11), a directional-edge-based feature vector is introduced to represent the features of the object in an image. With such feature vectors, the very basic template matching algorithm can work well in condition of illumination variation. To further enhance the performance of a tracking system, an on-line learning technique as well as a statistical approach were developed. This statistical approach is based on regeneration of multi-candidate

locations which is similar to the particle filter but much easier to be implemented into VLSI. By simulation experiments under various conditions, this algorithm has been shown to be robust against illumination variation, object size variation, partial occlusion, and object deformation.

5.2 Object Tracking Algorithm

5.2.1 Directional-edge-based feature vector generation

The first step of this feature vector generation algorithm is the feature map generation which extracts edge information from an input image. Fig. 5.1(a-e) shows an input image and its four-directional edge maps. Each edge map represents the distribution of edge flags corresponding to each direction, i.e. horizontal, +45 degree, vertical, or -45 degree in the original image. These edge maps are the most fundamental features extracted from the original image. Then a 64 dimension vector is generated as spatial distribution histograms of edge flags in these edge maps as shown in Fig. 5.1(f). The percentage of the total edge flags in an input image is fixed, thus for images of the same size, the summation of all components of such a vector is a constant (N). This average principle-edge distribution (APED (65)) feature vector is used throughout the present tracking algorithm.

When the feature vector is combined with template matching, a tracker robust to illumination variation can be achieved. However, in more complex conditions such as object size variation, partial occlusion, and object deformation, such a straightforward approach is prone to failure. Since statistical algorithms usually work well in tracking with ambiguities, being inspired by the basic concept of particle filter, a statistical approach employing regeneration of multiple candidate locations has been developed to improve the system performance.

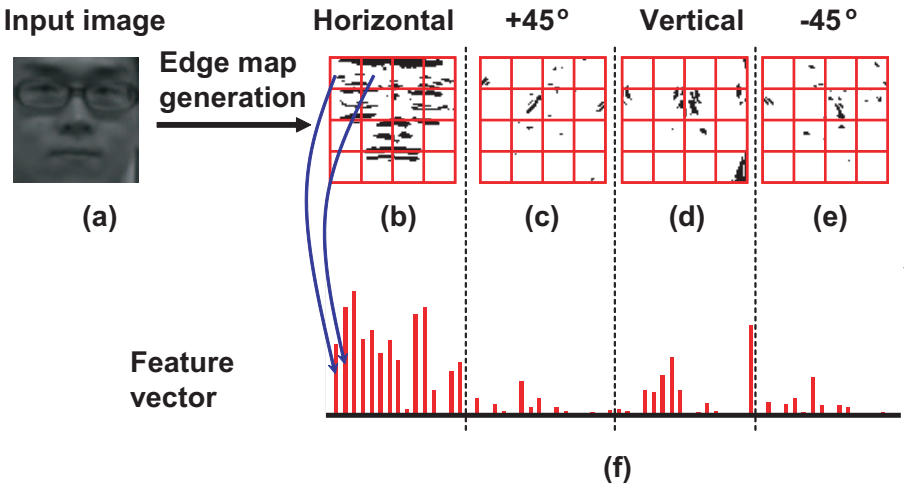


Figure 5.1: Feature vector generation.

5.2.2 Overall flow of the tracking algorithm

Fig. 5.2(a) shows the overall flow of this tracking algorithm. The first frame of the movie is processed by the "initialization block". In the initialization, the target should be chosen as a square area as shown in the inside part of the red rectangular in Fig. 5.2(b). Then the "feature vector generation" function generates the first feature vector and records it in the "feature vector templates container". The "candidate locations initialization" function initializes all candidate locations based on the position of the selected target as the blue points shown in Fig. 5.2(c) and record them in the "candidate locations container". After the initialization, whenever a new frame is available, the tracking block is processed once. In tracking block, candidate locations are regenerated and new feature vector may be created and added to the "feature vector templates container". With more feature vector templates, the system learns more about the target. Since such an on-line learning system usually add new feature vector templates when something occurs on the target, the description below suppose there are more than one feature vector templates in the "feature vector templates

container". All candidate locations' center of gravity is calculated as the tracking result as shown is Fig. 5.2(d,e), in which the blue points are the regenerated candidate locations.

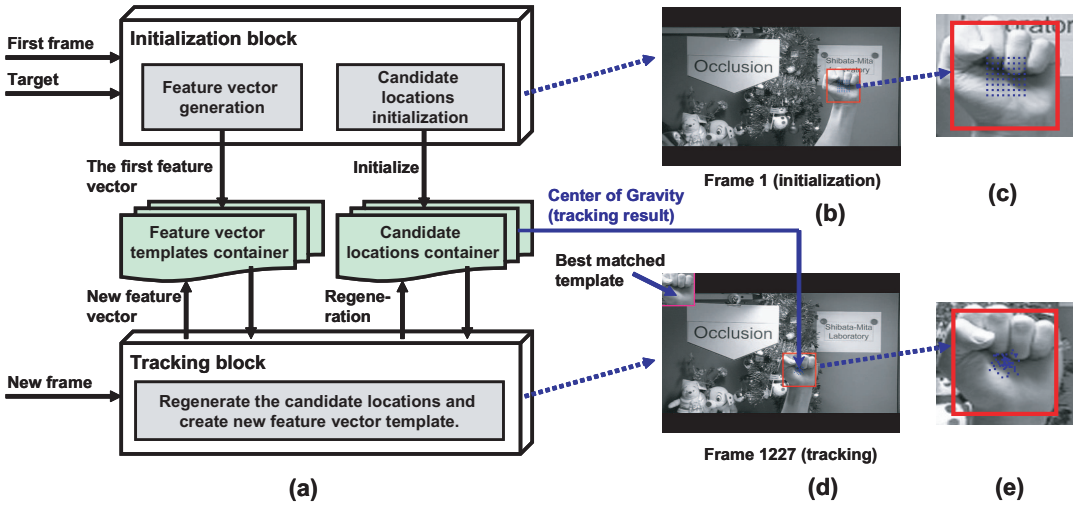


Figure 5.2: Basic flow of this tracking algorithm

5.2.3 On-line learning and regeneration of multiple candidate locations

The tracking process is shown as three steps in Fig. 5.3. Suppose the total number of candidate locations is C. Whenever there is a new frame, the tracking process starts at step 1: for each candidate location in the "candidate locations container", the feature vector of the square region centered by this location in the new frame is extracted. The size of the square region is the same as used during initialization. Then the Manhattan distances between the corresponding feature vector and all feature vector templates are calculated and the minimum distance value is recorded. Based on the minimum Manhattan distance, each candidate location is assigned a weight. Since small Manhattan distance means

the candidate location is similar to one of the feature vector templates, smaller Manhattan distance get a larger weight.

For example, the weighting strategy in the present work is described as follow. Define the summation of all components of a feature vector is N. Then if the Manhattan distance is larger than N, a weight of 0 is assigned. For Manhattan distances $M_{distance}$ smaller than N, integer weights are assigned nearly proportional to the evaluation formula: $10 - INT(10 \times M_{distance}/N)$. Here $INT(x)$ means to take the integer component of x. When $M_{distance}$ is zero or smaller than $N/10$, the formula gets the maximum value 10, then the corresponding weight is W_{max}, which equals to $C/4$ in the experiments of this work. After step 1, each candidate locations is assigned with a weight.

In step 2, the system firstly checks candidate locations one by one to find whether there are some candidate locations having the weight W_{max}. For each of such candidate location, a new candidate location is generated randomly in its vicinity. After the processing for all candidate locations is finished, if the number of newly generated candidate locations is smaller than the total number of candidate locations C, the system continue to check candidate locations one by one to find whether there are candidate locations have the weight $\geq (W_{max} - 1)$. For such candidate locations, a new candidate location is generated in the vicinity of each location. This process is repeated until C new candidate locations are regenerated. Then replace the present candidate locations with these newly generated candidate locations into the "candidate locations container" for the next frame as shown in Fig. 5.3. In this way the candidate locations that have larger weights are more probable to generate more new candidate locations nearby. These two steps are similar to the weighting and resampling in particle filters, while much easier to be implemented into VLSI.

To further enhance the performance of the system, an on-line feature vector

template generation technique is developed as step 3 in Fig. 5.3. After the former two steps, new candidate locations are generated. Then the feature vector of the square region centered by the center of gravity of all newly generated candidate locations in the present frame is calculated and compared with all the feature vector templates in the "feature vector templates container". When the minimum Manhattan distance between the feature vector and all feature vector templates is larger than a threshold (for example: $0.06 \times N$ in the present work), this feature vector is added as a new feature vector template. This method gives the system an on-line learning capability, which further improves the performance of the system.

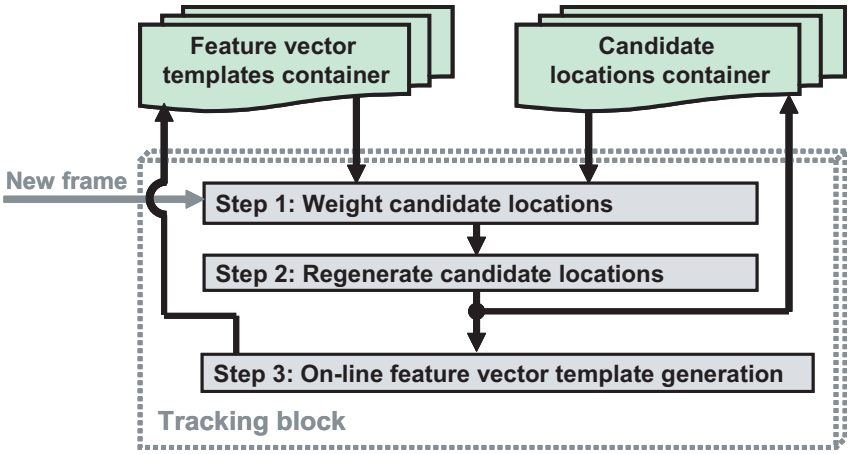


Figure 5.3: Tracking process by regenerate multiple candidate locations.

5.3 Experimental Results

The video sequence in ref. (66) is used to evaluate the performance of this tracking algorithm. Eight frames are selected from the results as examples and shown in Fig. 5.4(a). Frame 1 is used for initialization in which the position and size of the hand are selected; then the first APED vector in template is created. Near

frame 267, the hand transforms a little. In frame 646, the system generates an ADED vector automatically, which is added as a new template. Therefore, in frame 647, a new template appears at the upper-left corner. The illumination changes in frame 954 and 1197. In the rest three frames, partial occlusion occurs. The tracking system works well in all such complex conditions. The templates are shown in Fig. 5.4(b). The number below each template is the frame number from which the template was generated. Except the leftmost template, which is decided by human, all the other templates are generated automatically by the tracking algorithm when there are some things occur on the target. Based on these results, the on-line learning ability of the tracking system is verified.

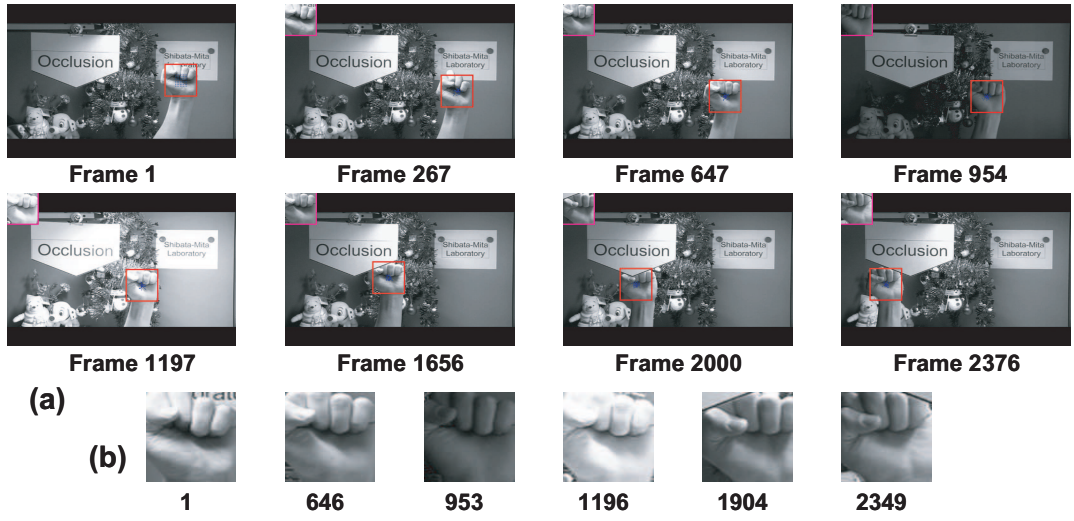


Figure 5.4: Tracking hand sequence in complex condition. (64 candidate locations)

Fig. 5.5(a) shows 10 frames from the results of another video sequence, which contains more complex conditions. After initialization in frame 1, the system tracks the target by an on-line generated template nearing frame 714. Near frame 1254, the person walks to the camera so the face becomes bigger, while

near 1902 the person walks farther to the camera. A new template is generated in each case. For consecutive frames 1770 and 1771, the change in position is about 13 pixels, which is about 15 percents of the target size. In the last three frames shown in Fig. 5.5(a), more severe partial occlusion occurs. The person removes his glasses and moves for a while then wear it again. This system works well in all such conditions. Fig. 5.5(b) shows all nine templates in the "feature vector templates container" in which eight templates on the right were on-line generated automatically.

Figure 5.5: Tracking face sequence in complex condition. (64 candidate locations)

5.4 Summary

A directional-edge-based object tracking algorithm was developed. By using directional-edge-based feature vectors, the system has been made robust against illumination variation. The on-line learning technique and the statistical multiple-candidate-location generation have further improved the performance, making the system robust against object size variation, partial occlusion, and object defor-

mation. The performance was verified by experiments under varying disturbing conditions.

Chapter 6

FPGA Implementation of a Directional-Edge-Based Real-Time Object Tracking System

6.1 Introduction

Real-time object tracking is highly demanded in many practical applications such as video surveillance, human-computer interface, vehicle navigation, and robot control. For developing a robust object tracking system, there are many challenges: abrupt object motion, changing appearance patterns of both the object and the scene, non-rigid object structures, object-to-object and object-to-scene occlusions, and camera motion. Regarding these complex environments, in order to achieve good performance in object tracking, many algorithms has been proposed (119). Particle filter is one of the very popular and powerful algorithms for solving this problem in various conditions (122; 125; 126; 127; 128). However, the real-time performance, which is very critical for many applications, is difficult to achieve especially for high precision algorithms such as (122) by using

software running on the general purpose serial CPU, because of the heavy computation cost. Although by employing the high technique of multi-core, digital signal processors (DSPs) and graphics processing units (GPUs) (2; 21) real-time performance is feasible, it also results a complex power-hungry system, which is not desirable.

Instead of using general purpose processing circuits, developing specific VLSIs for performance enhancement is one solution to achieve high speed, compact implementation, and low power. In reference (124), a particle filter design methodology is proposed and implemented on a field programmable gate array (FPGA). The simulation results shows such a processor outperforms conventional DSPs in both speed and power consumption. But since the computation of particle filter is generally very complex and expensive, including a number of floating point operations, it is not very desirable in terms of direct VLSI implementation. Such a problem dues to the fact that the particle filter algorithm is not tuned to be efficiently implemented in VLSIs.

Therefore, in chapter 5 a VLSI-implementation-friendly object tracking algorithm that is robust in various circumstances is proposed. Inspired by the biological research in (11), a directional-edge-based feature vector is adopted to represent the features of the object in an image. With such feature vectors, the very basic template matching algorithm can work well in condition of illumination variation. To further enhance the performance of a tracking system, a particle filter inspired statistical approach was developed which we named the regeneration of multi-candidate locations. Since this algorithm simplified the particle filter algorithm into a non-floating point processing only manner, it is much easier to be implemented into VLSI. Also an on-line learning technique is developed for dealing with the complex environment. By simulation experiments under various conditions, this algorithm has been shown to be robust against illumination

variation, object size variation, partial occlusion, and object deformation.

In this chapter, a simple real-time object tracking system based on the idea of multi-candidate locations is developed. By implementing the VLSI-implementation friendly functions on an FPGA, the processing speed of this system is very fast. According to the experimental result, this system can finish the calculation task in about 0.1 ms after finishing the image data transfer from the image sensor to the FPGA. Since in this work, only eight candidate locations are employed for tracking and the on-line learning is not included, the performance is not enough for real applications. Nevertheless, such a fast image processing speed gives the opportunity of designing a better tracking system with high performance when developing control circuits for pipelining the processing circuits and other functionalities such as the on-line learning.

The organization of this chapter is as follows. In §6.2, the targeted intelligent image processing algorithm in this study is explained. In §6.3, the VLSI implementation is described. In §6.4, the layout of the test chip and the circuit simulation results are presented. Finally, the conclusions are given in §6.5.

6.2 Restrained Object Tracking Algorithm

The first step of this feature vector generation algorithm is the feature map generation which extracts edge information from an input image. Fig. 5.1(a-e) in chapter 5 shows an input image and its four-directional edge maps. Then a 64 dimension vector is generated as spatial distribution histograms of edge flags in these edge maps as shown in Fig. 5.1(f). The percentage of the total edge flags in an input image is fixed, thus for images of the same size, the summation of all components of such a vector is a constant (N). This average principle-edge distribution (APED (65)) feature vector is used throughout this tracking

algorithm.

6.2.1 Overall flow of the tracking algorithm

Fig. 6.1(a) shows the overall flow of this tracking algorithm. The first frame of the movie is processed by the "initialization block". In the initialization, the target should be chosen as a square area as shown in the inside part of the red rectangular in Fig. 6.1(b). Then the "feature vector generation" function generates the "target feature vector". The "candidate locations initialization" function initializes all candidate locations based on the position of the selected target as the blue points shown in Fig. 6.1(c) and record them in the "candidate locations container". After the initialization, whenever a new frame is available, the tracking block is processed once, in which every candidate locations are regenerated to move with the target. All candidate locations' center of gravity is calculated as the tracking result as shown is Fig. 6.1(d,e), in which the blue points are the regenerated candidate locations.

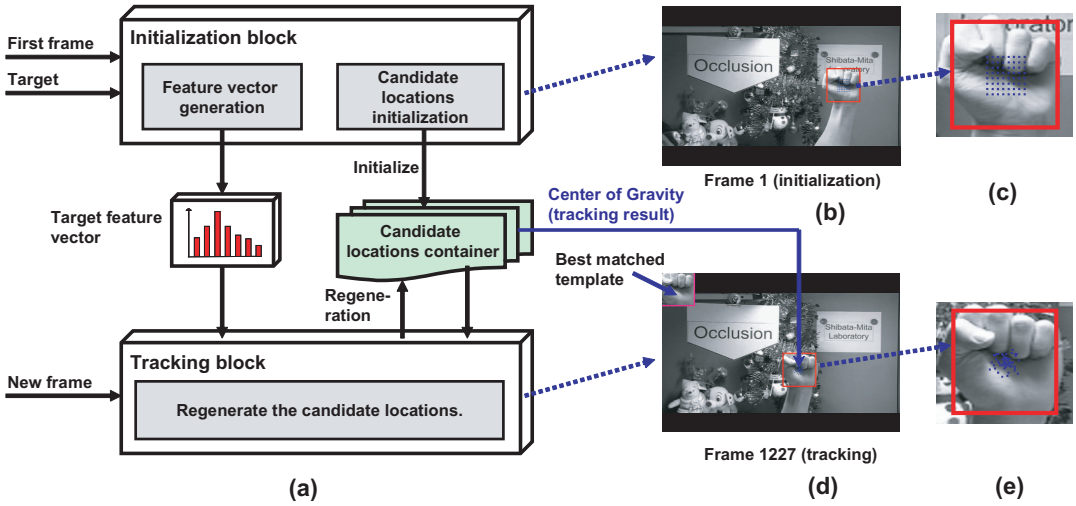


Figure 6.1: Basic flow of this tracking algorithm.

6.2.2 Regeneration of multiple candidate locations

The tracking process is shown as two steps in Fig. 6.2. Whenever there is a new frame, the tracking process starts at step 1: calculate a "weight" for each present candidate location in the "candidate locations container" based on the new frame and the "target feature vector", Then in step 2, the candidate locations are regenerated based on the present candidate locations and their weights which are generated at step 1.

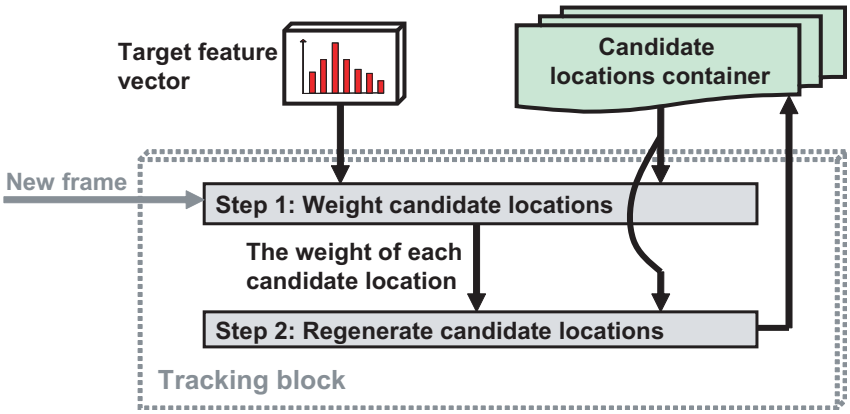


Figure 6.2: Tracking process by regenerate multiple candidate locations.

Fig. 6.3 shows "step 1" in more detail. For each candidate location in the "candidate locations container", the square region centered by this location in the new frame, which is named "the image provided by a candidate location" in this figure, is collected. The size of the square region is the same as used during initialization. Then the feature vector of this square region is extracted by the "feature vector generation" function. Next, the Manhattan distances between this feature vector and the "target feature vector" is calculated and recorded. Based on this distance, each candidate location is assigned a weight. Since small Manhattan distance means the candidate location is similar to target feature vector, smaller Manhattan distance get a larger weight.

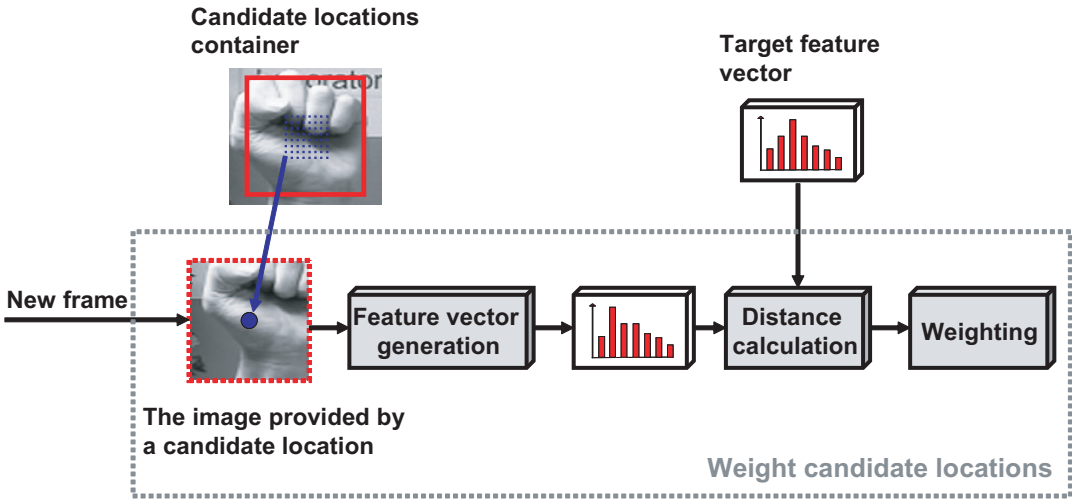


Figure 6.3: Weight candidate locations.

For example, the weighting strategy in the present work is described as follow. Suppose the total number of candidate locations is C. Define the summation of all components of a feature vector is N. Then if the Manhattan distance is larger than N, a weight of 0 is assigned. For Manhattan distances $M_{distance}$ smaller than N, integer weights are assigned nearly proportional to the evaluation formula: $10 - INT(10 \times M_{distance}/N)$. Here $INT(x)$ means to take the integer component of x. When $M_{distance}$ is zero or smaller than $N/10$, the formula gets the maximum value 10, then the corresponding weight is W_{max}, which equals to 15 in the experiments of this work. After "step 1", each candidate locations is assigned with a weight.

Fig. 6.4 shows "step 2" in more detail. Since each candidate location has a weight, the location is expressed by (X_i, Y_i, W_i) as shown in Fig. 6.4(a), while i is the number of the candidate location; X_i and Y_i stands for the horizontal and vertical coordination of this position; W_i stands for the weight. As illustrated in Fig. 6.4(b), the system firstly checks candidate locations one by one to find whether there are some candidate locations having the weight W_{max} (7 in this

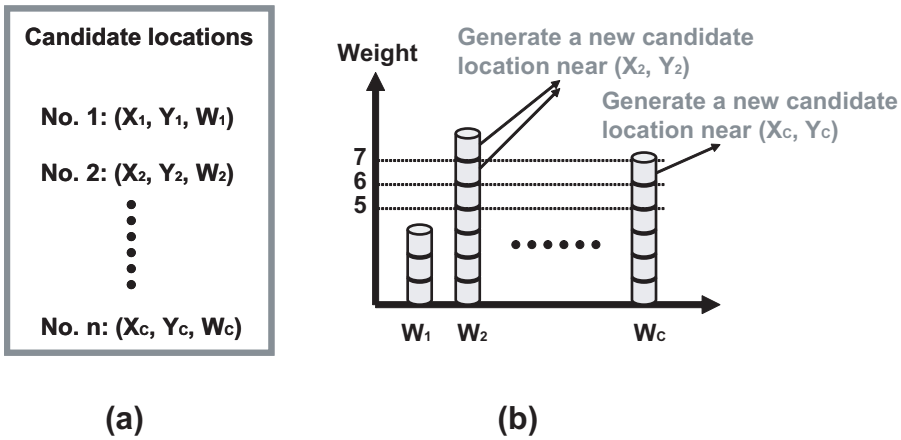


Figure 6.4: Regenerate candidate locations.

example). For each of such candidate location, a new candidate location is generated randomly in its vicinity (one new candidate location near (X_2, Y_2)). After the processing for all candidate locations is finished, if the number of newly generated candidate locations is smaller than the total number of candidate locations C, the system continue to check candidate locations one by one to find whether there are candidate locations have the weight $\geq (W_{max} - 1)$ (6 in this example). For such candidate locations, a new candidate location is generated in the vicinity of each location (one new near (X_2, Y_2) and one new near (X_C, Y_C)). This process is repeated until C new candidate locations are regenerated. Then replace the present candidate locations with these newly generated candidate locations into the "candidate locations container" for the next frame as shown in Fig. 6.2. In this way the candidate locations that have larger weights are more probable to generate more new candidate locations nearby.

These two steps are similar to the weighting and resampling in particle filters, while much easier to be implemented directly into hardware.

6.3 FPGA-Implementation of Object Tracking Algorithm

6.3.1 System organization and architecture of tracking processor

Based on the algorithm introduced in §6.2, a real-time object tracking system which achieves satisfying performance in simple environment by employing only eight candidate locations are developed. The organization of this system is shown in Fig. 6.5. It contains four main blocks including: an image sensor, a tracking processor which is implemented on an FPGA board, display control circuits on a separated FPGA board, and a display for showing the tracking result. The main processing tasks are executed on the tracking processor, in which a powerful FPGA chip of Altera: the Stratix III EP3SL340, is used. This FPGA contains 338,000 logic elements (LEs) with 20497 Kb RAM, which is enough to handle this system. Since there is not display connection on this board, another FPGA board is used especially for showing the results in real time. This board embeds many convenient chips including Cyclone II C270 FPGA device, VGA DAC with VGA-out connector, two 32-Mbyte SDRAM, switches, etc.

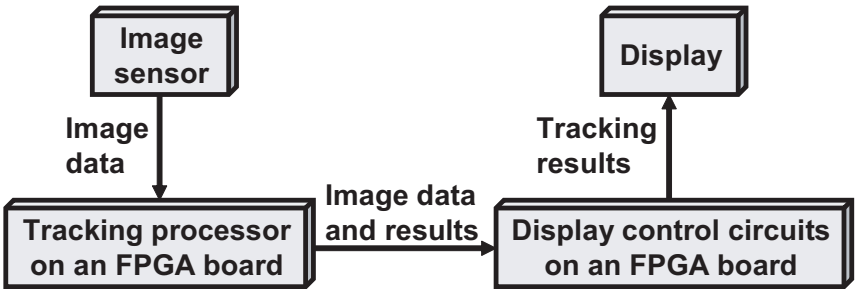


Figure 6.5: System organization.

The over-all data flow of is system is also shown in Fig. 6.5. The image sensor

captures the image and transfers the image to the tracking processor, on which the images are processed. The tracking results, including the position of the eight candidate locations and the position of the object, together with the image data, are transferred to the display control circuits. The display control circuits output the result through the VGA-out connector to a display. The most important part is the tracking processor.

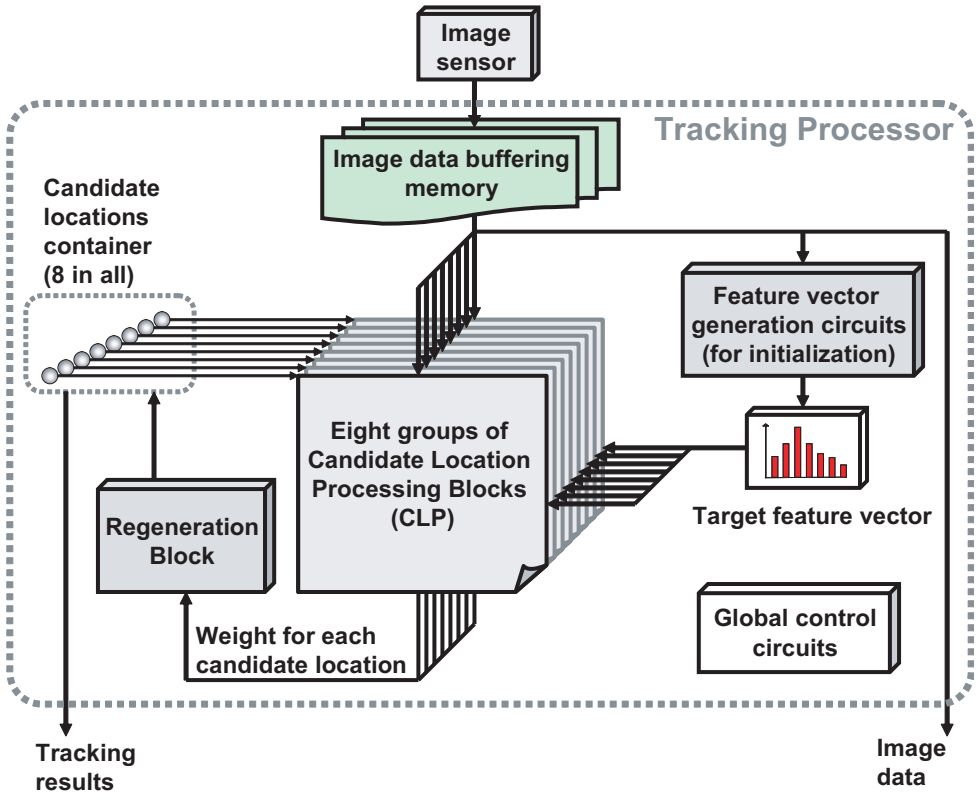


Figure 6.6: Architecture of the tracking processor.

The architecture of this tracking processor in illustrated in Fig. 6.6. This system has eight candidate locations, the positions of which are recorded in the "candidate locations container". Each candidate location has a "candidate location processing (CLP)" block for itself. So there are eight CLP blocks in all. The

data from the image sensor are firstly buffered into a memory, and then broadcasted to eight groups of CLP blocks, the "feature vector generation circuits" especially for initialization, and the "display control circuits" outside this FPGA. During initialization, which is triggered by a push button, the "feature vector generation circuits" in this figure generates the "target feature vector" based on the position of the target object which is manually set. This "target feature vector" is supplied to each CLP for weighting a candidate location. But processing the information of image data and the position, a CLP generates a weight for the candidate location. The "regeneration block" renew the information of all candidate locations based on the weight of each candidate location. The signals for controlling the flow of the system is generated by a "global control circuits". After reset, the initialization is triggered by a push button. During initialization, the "target feature vector" is generated, and the position of each candidate location is set based on the position of the target. Then, when ever a new frame is transferred to the FPGA, a tracking calculation is performed. The size of the target is set to be 64×64 in this system.

6.3.2 Candidate location processing block

As shown in Fig. 6.7(a), the CLP contains both the "feature vector generation circuits", which is the same with the one especially for initialization in Fig. 6.6 and the "weighting circuits". By efficiently processing image data from the buffering memory, the "feature vector generation circuits" generates the feature vector for the candidate location in the new frame. Then the "weighting circuits" calculates a weight by comparing the generated feature vector and the target feature vector. A more detailed structure of the "feature vector generation circuits" is shown in Fig. 6.7(b). It consists of three function components. The "small image selection circuits" receives the image data and the position of the candidate location,

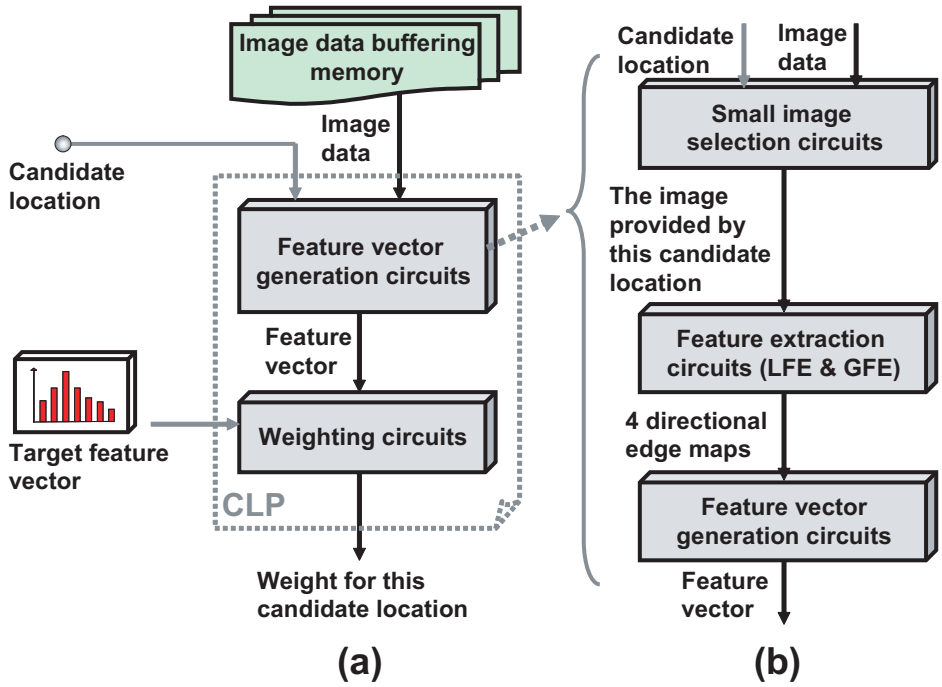


Figure 6.7: Structure of candidate location processing (CLP).

and record the small 68 × 68 image centered by the candidate location for the following "feature extraction circuits", in which the global features are extracted as four 64 × 64 directional edge maps as illustrated in §6.2. Then the "feature vector generation circuits" convert the edge maps into an APED feature vector.

Fig. 6.8 shows the "small image selection circuits" in more detail. By controlling the write enable signal and the write address, a 68 × 68 image centered by the candidate location is recorded into the SRAM. After the whole frame has been transferred from the image sensor to the SRAMs in the "small image selection circuits" of each candidate location, the "feature extraction circuits" starts to read small image data and performing feature extraction, which consists of both the local feature extraction (LFE) and global feature extraction (GFE).

Fig. 6.9 shows the LFE in detail. The image data preserved in the SRAM in

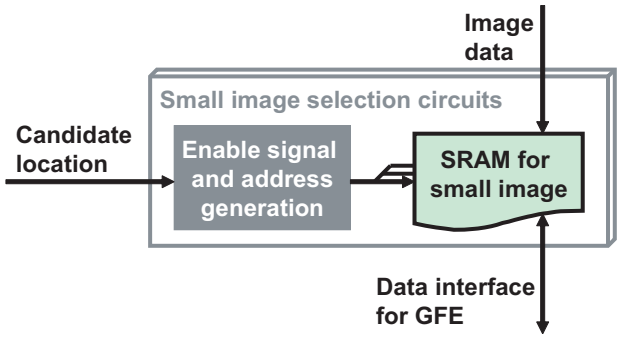


Figure 6.8: Small image selection circuits.

"small image selection circuits" (as the eagle in this Fig. 6.9(a)) are transferred one pixel by one pixel to four groups of 68 pixels shift registers for recording four rows of pixels as shown in Fig. 6.9(b). Each pixel records a 12-bit intensity datum. When the first pixel, as the upper left pixel in Fig. 6.9(a), reaches the upper right position of the four rows of registers, the 1×5 array composed of the pixel from the SRAM (as the red pixel in Fig. 6.9(b)) and the right most pixel of each row of the register array (as the four blue pixels in Fig. 6.9(b)) just records the pixels in the 1 by 5 region at the upper left of the image. The same concept is illustrated in Fig. 6.9(b) by using the same image of eagle, in which the blue region is the pixels recorded in the shift registers and the pixel from the SRAM is in red. This 1×5 array is transferred to a 5×5 scanning shift registers as shown Fig. 6.9(c), which is used for seamlessly scanning every 5×5 region within the 68×68 image. For each 5×5 region, the data are processed by the "pixel site processing" block which generates the gradient and the direction of this pixel site. The gradient data which are further used in GFE, are recorded in a 64×64 16-bit shift register array as shown in Fig. 6.9(d), while the direction data which are transferred to the "feature vector generation circuits" are recorded in a 64×64 2-bit shift register array as shown in Fig. 6.9(e).

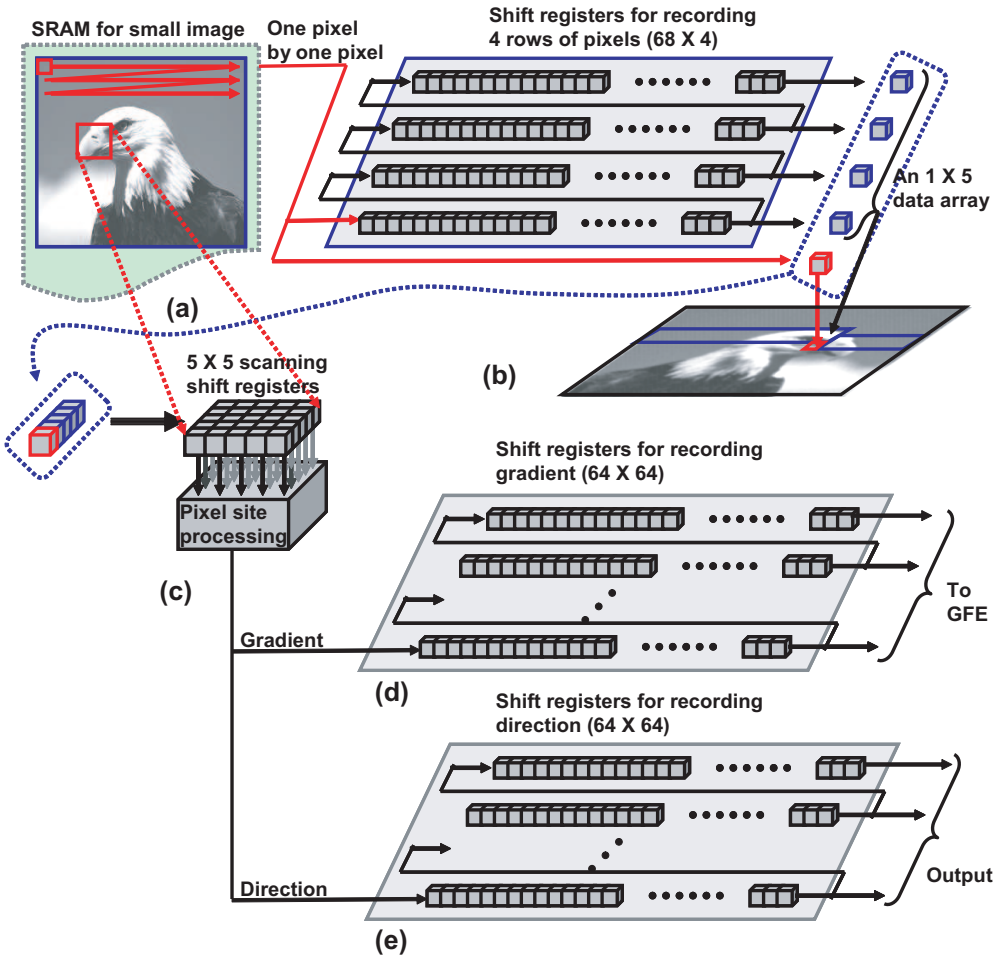


Figure 6.9: Local feature extraction circuits (LFE).

The structure of "pixel site processing" block is designed exactly according to the LFE algorithm introduced in §2.2. The kernel calculation is achieved in fully parallel by four adders and four absolute value calculation circuits. Then combinational maximum-gradient-selection circuits are used to select the largest value. This value together with the direction is buffered as the outputs.

Fig. 6.10 shows the architecture of the GFE circuits. The same sorting algorithm is employed as introduced in §2.3 which contains both a "mark" and a "flag" for each pixel site. However, since such a parallel processing array as used

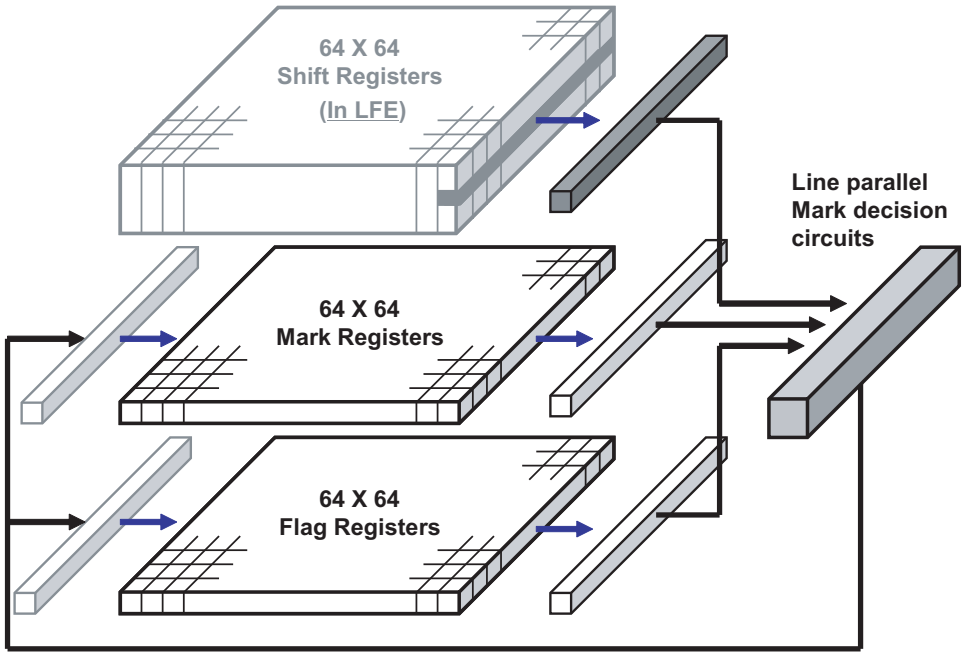


Figure 6.10: Global feature extraction circuits (GFE).

in the chip of chapter 2 is not feasible in FPGA, for a better trade of between the complexity of the circuits and the speed, we developed this algorithm in a line-parallel way, in which the processing of a particular bit for each gradient value is divided into 64 cycles. In each cycle, as shown in Fig. 6.10, the gradient values, marks, and flags of 64 pixels are processed and the results are feedback to both the "mark" and "flag" shift register arrays.

Fig. 6.11 shows the architecture of the "feature vector generation circuits". After GFE, the features are expressed by four edge maps. For fast feature vector generation, the data of one column in each edge map are transferred to the "feature vector generation circuits" simultaneously. Each edge map has its own "vector component generation block". Each of them contains four "accumulators" for generate four components at the same time in 16 clock cycles. Such architecture enables the generation of a 64-dimensional APED vector in only 64

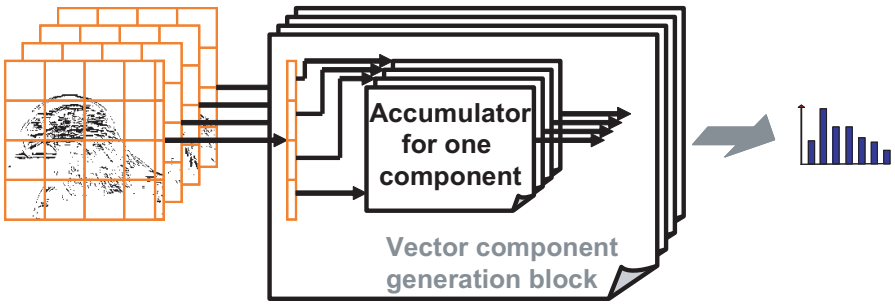


Figure 6.11: Feature vector generation circuits.

clock cycles.

The APED vector generated by the circuits in Fig. 6.11 are compare with the "target feature vector" in the "weighting circuits". The "Manhattan distance", which is very easy to be implemented into circuits, is calculated for evaluating the corresponding weight (integer). In this system, the weight is just calculated by divide the "Manhattan distance" by a constant value. However, more flexible relationship between the "Manhattan distance" and the weight can be easily implemented by a look-up table.

6.3.3 Regeneration block

Fig. 6.12 shows the concept of candidate locations regeneration circuits in this system. The "down counter" starts from the maximum value of all possible weights values. The comparator compare the value from the "down counter" and the weight for each location in a serial manner. When a weight (W_i) larger than or equal to the "down counter" value, a new candidate location is generated near (X_i, Y_i). This process finishes when there are enough (eight) new candidate locations generated.

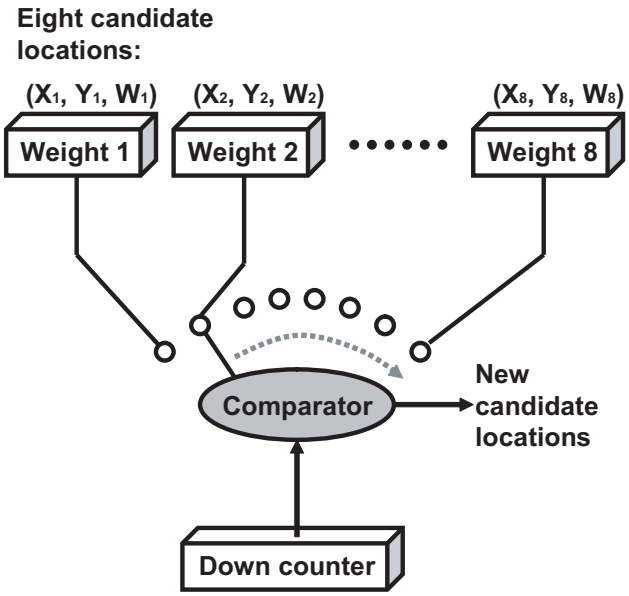


Figure 6.12: Regeneration block

6.4 Experimental Results

Fig. 6.13 shows the result in which the knob of a door is tracked when the camera is moved. In frame 163 and 230, the scene was shifted to the left; while in frame 410 and 522, the scene was shifted to the lower right direction. Finally, the scene was shifted to right in frame 579. In such a simple test, the tracking system can focus on the object faithfully without missing. This system is running at a frequency of 60 MHz. Thanks to the VLSI-implementation-friendly object tracking algorithm, the total processing time for one frame after the complete transferring of this frame is reduced to about 6000 clock cycles, which means only 0.1 ms at 60 MHz. About 94.5% of these clock cycles are used for feature extraction, in which local feature extraction takes more than 80%, since it needs the repetitive calculation for each pixel site in a serial manner. The calculation time for the global feature extraction, which is very time consuming in software-based algorithms,

has been reduced greatly due to a very efficient VLSI-implementation adapted algorithm.

Figure 6.13: Experimental results

Considering building a system that faithfully runs the algorithm introduced in chapter 5, if developing the control circuits for using the processing circuits in this system in a pipeline manner with on-line learning, a robust tracking system with a latency of less than 1 ms second is also feasible. Compared with the processing time for one frame on a 1.6 GHz general purpose CPU, which is about 5 seconds, a speed up of 5000 times can be expected. Such an order of performance enhancement is much faster than the available GPUs (2). At the same time, it cost less power since the frequency is not high.

6.5 Summary

In this chapter, a simple real time object tracking system based on a restrained version of the prior algorithm has been implemented successfully. By experi-

mental results, this system shows satisfying performance in simple tracking tasks by employing only eight candidate locations. Thanks to the fine-grained VLSI-implementation of the object tracking algorithm implemented in an FPGA, the total processing time for the tracking task has been reduced to about 0.1 ms when the system is running at a frequency of 60 MHz. This performance gives the opportunity of designing the robust object tracking system on the same FPGA with a performance speed up of 5000 times compared with the serial CPUs. Such an approach is also much efficient that employing the parallel processing circuits for general purpose such as GPUs (2).

Chapter 7

Conclusions

7.1 Summary of This Thesis

In this work, *VLSI circuits and systems* for *directional-edge-base intelligent image processing* algorithms have been researched and developed. In order to minimize the latency caused by the image data transfer between the image sensor and the processing circuits, DPS-embedded processors were proposed and designed. The performance of such processor has been verified by building a real-time image recognition system with a very low latency. In addition, a directional-edge-based object tracking algorithm was also proposed and partially implemented in an object tracking system by building processing circuits on FPGAs. The followings are summaries through this thesis.

In chapter 2, a DPS-embedded global feature extraction VLSI processor for real-time image recognition has been developed. By combining the block-readout architecture of DSP and parallel processing elements, the latency of local feature extraction has been markedly reduced. By adapting the rank-order filter algorithm to hardware implementation, global feature extraction is accomplished in only 11 cycles. A prototype chip was designed in a 0.18-μm five-metal CMOS technology. The measurement results show that the VLSI processor can extract features more than 400 times faster than software processing running on a 2-GHz

general-purpose processor when operating at 60 MHz.

In chapter 3, a digital-pixel-sensor-based early-visual-processing VLSI processor for real-time intelligent image processing has been developed. By combining the block-readout architecture for DPS and parallel processing elements, the latency of local image processing has been markedly reduced. By adapting the rank-order filter algorithm to hardware implementation, global feature extraction is performed in a very fast manner. The enhancement in the functionality of processing element improves the programmability of the processor greatly. As a result, such a chip can handle multiple algorithms efficiently. A prototype chip was designed in a 65-nm 12-metal CMOS technology. The measurement results show that this VLSI processor can achieve all expected functions.

In chapter 4, a real time image recognition system has been developed. The system is based on a VLSI-implementation friendly image recognition algorithm. By using the global directional-edge-feature extraction VLSI processor, the latency between the image capture and the final recognition as small as 906 μs has been demonstrated. The merit of the global feature extraction algorithm that it can focus on more significant features automatically has also been experimentally verified.

In chapter 5, a directional-edge-based object tracking algorithm was developed. By using directional-edge-based feature vectors, the system has been made robust against illumination variation. The on-line learning technique and the statistical multiple-candidate-location generation have further improved the performance, making the system robust against object size variation, partial occlusion, and object deformation. The performance was verified by experiments under varying disturbing conditions.

In chapter 6, a simple real time object tracking system based on a restrained version of the prior algorithm has been implemented successfully. By experi-

mental results, this system shows satisfying performance in simple tracking tasks by employing only eight candidate locations. Thanks to the fine-grained VLSI-implementation of the object tracking algorithm implemented in an FPGA, the total processing time for the tracking task has been reduced to about 0.1 ms when the system is running at a frequency of 60 MHz. This performance gives the opportunity of designing the robust object tracking system on the same FPGA with a performance speed up of 5000 times compared with the serial CPUs. Such an approach is also much efficient that employing the parallel processing circuits for general purpose such as GPUs.

7.2 Future Perspective

In this work, the image sensor technique, parallel digital circuits, and directional-edge-based intelligent image processing algorithms are combined to develop systems that can achieve, although rather naïve, some human-like functions. As all these targeted algorithms are developed to be easily implemented into VLSI directly at the very beginning, an efficient trade off between processing speed, response time, and the corresponding system complexity has been achieved. Regarding to the power consumption, since such systems are still in the demonstration level, including functions on FPGA, display control circuits, and many auxiliary chips, the total energy is not low. However, once such systems were well established, the total power consumption could be reduced dramatically by developing more functions into ASICs.

Accompanied with the further development of today's semiconductor process technology and the deeper understanding on the visual processing mechanism of the human brain, I believe the bio-inspired VLSI/ULSI devices will become more and more feasible, with matured functionality. Such devices don't need to exactly reproduce the whole brain as it is on the semiconductor. Instead, it takes

more hint from this nature, implement the existed functions by fully utilizing the available characteristic of materials to maximally achieve an optimized system. Just considering how much time it takes for the formation of the human brain, the time for the "formation" of human-made semiconductor-based brain may take much, much less time.

7.3 Conclusions

VLSI Circuits and systems for brain-mimicking algorithms have been developed based on a very naïve model of the brain. In the algorithms of image processing, *directional edge information* plays an essential role for perception of still images as well as moving images. In these systems, the vast amount of subconscious processing in the mind has been implemented by VLSI chips or FPGAs. In order to build "real-time responding human-like intelligent systems" with small hardware volume and low powers, such development of hardware-friendly algorithms and their VLSI implementation in fine-grain parallel architectures are most essential.

References

[1] G. Moore, "Progress in digital integrated electronics," in *International Electron Devices Meeting (IEDM)*, 1975, vol. 21, pp. 11–13. 1

[2] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with *cuda*," *Queue*, vol. 6, no. 2, pp. 40–53, March/April 2008. 1, 3, 4, 32, 81, 96, 97

[3] T. Mizuno, J. Okamura, and A. Toriumi, "Experimental study of threshold voltage fluctuation due to statistical variation of channel dopant number in mosfetfs," *IEEE Trans. Electron Devices*, vol. 41, no. 11, pp. 2216–2221, 1994. 2

[4] P. A. Stolk, F. P. Widdershoven, and D. B. M. Klaassen, "Modeling statistical dopant fluctuations in mos transistors," *IEEE Trans. Electron Devices*, vol. 45, no. 9, pp. 1960–1971, 1998. 2

[5] A. Asenov, "Suppression of random dopant-induced threshold voltage fluctuations in sub-0.1-μm mosfetfs with epitaxial and δ-doped channels," *IEEE Trans. Electron Devices*, vol. 46, no. 8, pp. 1718–1724, 1999. 2

[6] K. Takeuchi, T. Fukai, T. Tsunomura, A. T. Putra, A. Nishida, S. Kamohara, and T. Hiramoto, "Understanding random threshold voltage fluctuation by comparing multiple fabs and technologies," in *International Electron Devices Meeting (IEDM)*, 2007, pp. 467–470. 2

[7] D. G. Lowe, "Distinctive image feature from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, November 2004. 2

[8] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, "A comparison of affine region detectors," *Int. J. Comput. Vision*, vol. 65, no. 1-2, pp. 43–72, 2005. 2

[9] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, October 2005. 2

[10] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer Vision - ECCV 2006*, pp. 404–417, 2006. 2

[11] 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, October 1959. 2, 12, 34, 58, 71, 81

[12] 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," *J. Solid-State Circuits*, vol. 38, no. 11, pp. 1992–2000, November 2003. 3, 32, 33

[13] W. Raab, N. Bruels, U. Hachmann, J. Harnisch, U. Ramacher, C. Sauer, and A. Techmer, "A 100-gops programmable processor for vehicle vision systems," *IEEE Des. Test Comput.*, vol. 20, no. 3, pp. 8–15, January-February 2003. 3

[14] A. A. Abbo, R. P. Kleihorst, V. Choudhary, L. Sevat, P. Wielage, S. Mouy, B. Vermeulen, and M. Heijligers, "Xetal-ii: a 107 gops, 600 mw massively parallel processor for video scene analysis," *J. Solid-State Circuits*, vol. 43, no. 1, pp. 192–201, January 2008. 3

[15] H. Noda, M. Nakajima, K. Dosaka, K. Nakata, M. Higashida, O. Yamamoto, K. Mizumoto, T. Tanizaki, T. Gyohten, Y. Okuno, H. Kondo, Y. Shimazu, K. Arimoto, K. Saito, and T. Shimizu, "The design and implementation of the massively parallel processor based on the matrix architecture," *J. Solid-State Circuits*, vol. 42, no. 1, pp. 183–192, January 2007. 3

[16] 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," *J. Solid-State Circuits*, vol. 44, no. 1, pp. 136–147, January 2009. 3, 32, 33, 58

[17] J. Tanahe, Y. Taniguchi, T. Miyamori, Y. Miyamoto, H. Takeda, M. Tarui, H. Nakayama, N. Takeda, K. Maeda, and M. Matsui, "Visconti: Multi-vliw image recognition processor based on configurable processor," in *Custom Integr. Circuits Conf. (CICC)*, 2003, pp. 185–188. 3

[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 *Custom Integr. Circuits Conf. (CICC)*, 2007, pp. 443–446. 3

[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, March 2009. 3

[20] H. Kondo, M. Nakajima, N. Masui, S. Otani, N. Okumura, Y. Takata, T. Nasu, H. Takata, T. Higuchi, M. Sakugawa, H. Fujiwara, K. Ishida, K. Ishimi, S. Kaneko, T. Itoh, M. Sato, O. Yamamoto, and K. Arimoto, "Design and implementation of a configurable heterogeneous multicore *soc* with nine *cpus* and two matrix processors," *J. Solid-State Circuits*, vol. 43, no. 4, pp. 892–901, April 2008. 3, 32, 33

[21] T. Y. Ho, P. M. Lam, and C. S. Leung, "Parallelization of cellular neural networks on *gpu*," *Pattern Recognition*, vol. 41, no. 8, pp. 2684–2692, August 2008. 3, 32, 81

[22] N. Massari and M. Gottardi, "A 100 *db* dynamic-range *cmos* vision sensor with programmable image processing and global feature extraction," *J. Solid-State Circuits*, vol. 42, no. 3, pp. 647–657, March 2007. 4, 32

[23] J. Dubois, D. Ginhac, M. Paindavoine, and B. Heyrman, "A 10 000 fps *cmos* sensor with massively parallel image processing," *J. Solid-State Circuits*, vol. 43, no. 3, pp. 706–717, March 2008. 4, 32

[24] M. Barbaro, P. Y. Burgi, A. Mortara, P. Nussbaum, and F. Heitger, "A 100×100 pixel silicon retina for gradient extraction with steering filter capabilities and temporal output coding," *J. Solid-State Circuits*, vol. 37, no. 2, pp. 160–172, February 2002. 4, 32

[25] P. F. Ruedi, P. Heim, F. Kaess, E. Grenet, F. Heitger, P. Y. Burgi, S. Gyger, and P. Nussbaum, "A 128×128 pixel 120-*db* dynamic-range vision-sensor chip for image contrast and orientation extraction," *J. Solid-State Circuits*, vol. 38, no. 12, pp. 2325–2333, December 2003. 4, 32

[26] M. Gottardi, N. Massari, and S. A. Jawed, "A 100 μw 128×64 pixels contrast-based asynchronous binary vision sensor for sensor networks applications," *J. Solid-State Circuits*, vol. 44, no. 5, pp. 1582–1592, May 2009. 4, 32

[27] O. Schrey, J. Huppertz, G. Filimonovic, A. Bubmann, W. Brockherde, and B. J. Hosticka, "A $1k$×$1k$ high dynamic range *cmos* image sensor with on-chip programmable region-of-interest readout," *J. Solid-State Circuits*, vol. 37, no. 7, pp. 911–915, July 2002. 4, 32

[28] Y. Sugiyama, M. Takumi, H. Toyoda, N. Mukozaka, A. Ihori, T. Kurashina, Y. Nakamura, T. Tonbe, and S. Mizuno, "A high-speed *cmos* image sensor with profile data acquiring function," *J. Solid-State Circuits*, vol. 40, no. 12, pp. 2816–2823, December 2005. 4, 32

[29] S. Mizuno, K. Fujita, H. Yamamoto, N. Mukozaka, and H. Toyoda, "A 256×256 compact *cmos* image sensor with on-chip motion detection function," *J. Solid-State Circuits*, vol. 38, no. 6, pp. 1072–1075, June 2003. 4, 32

[30] Y. M. Chi, U. Mallik, M. A. Clapp, E. Choi, G. Cauwenberghs, and R. Etienne-Cummings, "*cmos* camera with in-pixel temporal change detection and *adc*," *J. Solid-State Circuits*, vol. 42, no. 10, pp. 2187–2196, October 2007. 4, 32

[31] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 *db* 15 us latency asynchronous temporal contrast vision sensor," *J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, February 2008. 4, 32

[32] S. Kleinfelder, S. Lim, X. Liu, and A. E. Gamal, "A 10000 frames/s *cmos* digital pixel sensor," *J. Solid-State Circuits*, vol. 36, no. 12, pp. 2049–2059, December 2001. 4, 8, 13, 17, 18, 32, 34

[33] V. Brajovic and T. Kanade, "Computational sensor for visual tracking with attention," *J. Solid-State Circuits*, vol. 33, no. 8, pp. 1199–1207, August 1998. 4, 32

[34] Y. Ni and J. Guan, "A 256×256 pixel smart *cmos* image sensor for line-based stereo vision applications," *J. Solid-State Circuits*, vol. 35, no. 7, pp. 1055–1061, July 2000. 5, 33

[35] A. Graupner, J. Schreiter, S. Getzlaff, and R. Schuffny, "*cmos* image sensor with mixed-signal processor array," *J. Solid-State Circuits*, vol. 38, no. 6, pp. 948–957, June 2003. 5, 33

[36] K. Yoon, C. Kim, B. Lee, and D. Lee, "Single-chip *cmos* image sensor for mobile applications," *J. Solid-State Circuits*, vol. 37, no. 12, pp. 1839–1845, December 2002. 5, 33

[37] Y. Oike, M. Ikeda, and K. Asada, "Design and implementation of real-time $3-d$ image sensor with 640×480 pixel resolution," *J. Solid-State Circuits*, vol. 39, no. 4, pp. 622–628, April 2004. 5, 33

[38] L. Lindgren, J. Melander, R. Johansson, and B. Moller, "A multiresolution $100-gops$ $4-gpixels/s$ programmable smart vision sensor for multisense imaging," *J. Solid-State Circuits*, vol. 39, no. 4, pp. 622–628, April 2004. 5, 33

[39] 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," *J. Solid-State Circuits*, vol. 42, no. 12, pp. 2978–2989, December 2007. 5, 33

[40] D. Stoppa, A. Simoni, L. Gonzo, M. Gottardi, and G. F. D. Betta, "Novel *cmos* image sensor with a 132-*db* dynamic range," *J. Solid-State Circuits*, vol. 37, no. 12, pp. 1846–1852, December 2002. 5, 33

[41] Y. Muramatsu, S. Kurosawa, M. Furumiya, H. Ohkubo, and Y. Nakashiba, "A signal-processing *cmos* image sensor using a simple analog operation," *J. Solid-State Circuits*, vol. 38, no. 1, pp. 101–106, January 2003. 5, 33

[42] J. Yuan, H. Y. Chan, S. W. Fung, and B. Liu, "An activity-triggered 95.3 *dbdr* -75.6 *dbthdcmos* imaging sensor with digital calibration," *J. Solid-State Circuits*, vol. 44, no. 10, pp. 2834–2843, October 2009. 5, 33

[43] V. Gruev and R. Etienne-Cummings, "A pipelined temporal difference imager," *J. Solid-State Circuits*, vol. 39, no. 3, pp. 538–543, March 2004. 5, 33

[44] Y. Oike, M. Ikeda, and K. Asada, "A 120×110 position sensor with the capability of sensitive and selective light detection in wide dynamic range for robust active range finding," *J. Solid-State Circuits*, vol. 39, no. 1, pp. 246–251, January 2004. 5, 33

[45] Y. Oike, M. Ikeda, and K. Asada, "A 375×365 high-speed $3-d$ range-finding image sensor using row-parallel search architecture and multisampling technique," *J. Solid-State Circuits*, vol. 40, no. 2, pp. 444–453, February 2005. 5, 33

[46] A. Bandyopadhyay, J. Lee, R. W. Robucci, and P. Hasler, "*matia*: a programmable 80 μw/frame *cmos* block matrix transform imager architecture," *J. Solid-State Circuits*, vol. 41, no. 3, pp. 663–672, March 2006. 5, 33

[47] A. Nilchi, J. Aziz, and R. Genov, "Focal-plane algorithmically-multiplying *cmos* computational image sensor," *J. Solid-State Circuits*, vol. 44, no. 6, pp. 1829–1839, June 2009. 5, 33

[48] W. D. Leon-Salas, S. Balkir, K. Sayood, N. Schemm, and M. W. Hoffman, "A *cmos* imager with focal plane compression using predictive coding," *J. Solid-State Circuits*, vol. 42, no. 11, pp. 2555–2572, November 2007. 5, 33

[49] A. Rothermel, L. Liu, N. P. Aryan, M. Fischer, J. Wuenschmann, S. Kibbel, and A. Harscher, "A *cmos* chip with active pixel array and specific test features for subretinal implantation," *J. Solid-State Circuits*, vol. 44, no. 1, pp. 290–300, January 2009. 5, 33

[50] H. G. Graf, C. Harendt, T. Engelhardt, C. Scherjon, K. Warkentin, H. Richter, and J. N. Burghartz, "High dynamic range *cmos* imager technologies for biomedical applications," *J. Solid-State Circuits*, vol. 44, no. 1, pp. 281–289, January 2009. 5, 33

[51] T. D. Huang, S. Sorgenfrei, P. Gong, R. Levicky, and K. L. Shepard, "A 0.18-μm *cmos* array sensor for integrated time-resolved fluorescence detection," *J. Solid-State Circuits*, vol. 44, no. 5, pp. 1644–1654, May 2009. 5, 33

[52] D. Kim and G. Han, "A 200μs processing time smart image sensor for an eye tracker using pixel-level analog image processing," *J. Solid-State Circuits*, vol. 44, no. 9, pp. 2581–2590, September 2009. 5, 33

[53] 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," *J. Solid-State Circuits*, vol. 44, no. 1, pp. 127–135, January 2009. 5, 33

[54] T. Shibata, "Bio-inspired devices, circuits and systems," in *Proc. 35th European Solid-State Circuits Conference (ESSCIRC 2009)*, 2009, pp. 8–15. 6, 7

[55] T. Shibata and T. Ohmi, "Neural microelectronics," in *International Electron Devices Meeting (IEDM)*, 1997, pp. 337–342. 6

[56] M. Ogawa, K. Ito, and T. Shibata, "A general-purpose vector-quantization processor employing two-dimensional bit-propagating winner-take-all," in *Proc. Symp. VLSI Circuits*, 2002, pp. 244–247. 7, 8, 13, 23, 34, 43

[57] K. Ito, M. Ogawa, and T. Shibata, "A high-performance ramp-voltage-scan winner-take-all circuit in an open loop architecture," *Jpn. J. Appl. Phys.*, vol. 41, no. 4B, pp. 2301–2305, April 2002. 7, 8

[58] H. Hayakawa, M. Ogawa, and T. Shibata, "Right-brainleft-brain integrated associative processor employing convertible multiple-instruction-stream multiple-data-stream elements," *Jpn. J. Appl. Phys.*, vol. 44, no. 4B, pp. 2109–2118, April 2005. 7, 8

[59] M. Ogawa and T. Shibata, "A delay-encoding-logic array processor for dynamic-programming matching of data sequences," *J. Solid-State Circuits*, vol. 40, no. 7, pp. 1578–1582, July 2005. 7, 8, 58

[60] M. Ogawa and T. Shibata, "Nmos-based gaussian-element-matching analog associative memory," in *Proc. 27th European Solid-State Circuits Conference (ESSCIRC 2001)*, 2001, pp. 272–275. 7, 8

[61] T. Yamasaki and T. Shibata, "Analog soft-matching classifier using floating-gate mos technology," *IEEE Trans. Neural Networks*, vol. 14, no. 5, pp. 1257–1265, September 2003. 7, 8

[62] D. Kobayashi, T. Shibata, Y. Fujimori, T. Nakamura, and H. Takasu, "A ferroelectric associative memory technology employing heterogate fgmos structure," *IEEE Trans. Electron Devices*, vol. 52, no. 10, pp. 2188–2197, October 2005. 7, 8

[63] M. Yagi and T. Shibata, "An image representation algorithm compatible with neural-associative-processor-based hardware recognition systems," *IEEE Trans. Neural Networks*, vol. 14, no. 5, pp. 1144–1161, September 2003. 7, 13, 28, 34, 35, 52, 58

[64] Y. Suzuki and T. Shibata, "Validating directional edge-based image feature representations in face recognition by spatial correlation-based clustering,"

in *Proc. the 15th European Signal Processing Conference (EUSIPCO 2007)*, 2007, pp. 1940–1944. 7, 28, 52

[65] Y. Suzuki and T. Shibata, "Multiple-clue face detection algorithm using edge-based feature vectors," in *in Proceedings of ICASSP 2004*, 2004, pp. V737–V740. 7, 8, 34, 35, 72, 82

[66] S. Kim and T. Shibata, "Feature-based object tracking using spatial matching of differential directional-edge images," in *Proc. The International Conference on Signal Processing and Communication Systems 2007 (ICSPCS 2007)*, 2007, pp. 193–197. 7, 8, 16, 34, 36, 76

[67] J. Hao and T. Shibata, "A vlsi-implementation-friendly ego-motion detection algorithm based on edge-histogram matching," in *in Proceedings of ICASSP 2006*, 2006, pp. II–245–248. 7

[68] H. Hayakawa and T. Shibata, "Spatiotemporal projection of motion field sequence for generating feature vectors in gesture perception," in *Proc. The 2008 International Symposium on Circuits and Systems (ISCAS'08)*, 2008, pp. 3526–3529. 7, 8, 28, 34, 37

[69] H. Hayakawa and T. Shibata, "Block-matching-based motion field generation utilizing directional edge displacement," in *Proc. The International Conference on Signal Processing and Communication Systems 2007 (ICSPCS 2007)*, 2007, pp. 90–95. 7, 16, 28, 34

[70] K. Ito and T. Shibata, "A time-domain gradient-detection architecture for vlsi analog motion sensors," in *Proc. 2006 International Symposium on Circuits and Systems (ISCAS'06)*, 2006, pp. 201–204. 7, 8

[71] Y. Niki, Y. Manzawa, S. Kametani, and T. Shibata, "Moving-object-localization hardware algorithm employing or-amplification of pixel activities," *Jpn. J. Appl. Phys.*, vol. 47, no. 4, pp. 2767–2773, April 2008. 7

[72] Y. Nakashita, Y. Mita, and T. Shibata, "Analog edge-filtering processor employing only-nearest-neighbor interconnects," *Jpn. J. Appl. Phys.*, vol. 44, no. 4B, pp. 2119–2124, April 2005. 7, 8

[73] 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*, vol. 56, no. 11, pp. 2384–2392, November 2009. 7, 8

[74] H. Yamasaki and T. Shibata, "A real-time image-feature-extraction and vector-generation *vlsi* employing arrayed-shift-register architecture," *J. Solid-State Circuits*, vol. 42, no. 9, pp. 2046–2053, September 2007. 8, 13, 34, 58

[75] T. Nakagawa and T. Shibata, "A real-time image feature vector generator employing functional cache memory for edge flags," in *Proc. 2009 International Symposium on Circuits and Systems (ISCAS'09)*, 2009, pp. 3026–3029. 8

[76] K. Fujita, K. Ito, and T. Shibata, "A single-motion-vectorcycle-generation optical flow processor employing directional-edge histogram matching," in *Proc. 2009 International Symposium on Circuits and Systems (ISCAS'09)*, 2009, pp. 3022–3025. 8

[77] Y. Okano and T. Shibata, "High-frame-rate dense motion vector field generation processor with simplified best-match searching circuitries," in *Proc. IEEE Asian Solid-State Circuits Conference (A-SSCC)*, 2009. 8

[78] T. Yamasaki, M. Yagi, and T. Shibata, "A fully-parallel analog vector matching *lsi* for robust image recognition," in *Proc. IEEJ Int. Analog VLSI Workshop*, 2002, pp. 128–133. 8, 13, 34

[79] T. T. BUI and T. Shibata, "Compact bell-shaped analog matching-cell module for digital-memory-based associative processors," *Jpn. J. Appl. Phys.*, vol. 47, no. 4, pp. 2788–2796, April 2008. 8, 13, 34

[80] A. Kitchen, A. Bermak, and A. Bouzerdoum, "A digital pixel sensor array with programmable dynamic range," *IEEE Trans. Electron Devices*, vol. 52, no. 12, pp. 2591–2601, December 2005. 8, 13, 17, 34

[81] C. Shoushun, A. Bermak, W. Yan, and D. Martinez, "Adaptivequantization digital image sensor for low-power image compression," *IEEE Trans. Circuits Syst. I*, vol. 54, no. 1, pp. 13–25, January 2007. 8, 13, 17, 34

[82] B. Tongprasit, K. Ito, and T. Shibata, "A computational digital-pixelsensor *vlsi* featuring block-readout architecture for pixel-parallel rankorder filtering," in *Proc. The 2005 International Symposium on Circuits and Systems (ISCAS'05)*, 2005, vol. 3, pp. 2389–2392. 8, 13, 17, 18, 19

[83] 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: Regul. Pap.*, vol. 56, no. 1, pp. 114–123, January 2009. 8, 34, 35, 62

[84] J. Shotton, A. Blake, and R. Cipolla, "Multi-scale categorical object recognition using contour fragments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1270–1281, July 2008. 11

[85] J. Yang, D. Zhang, A. F. Frangi, and J. Yang, "Two-dimensional *pca*: a new approach to appearance-based face representation and recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 1, pp. 131–137, January 2004. 11

[86] T. Zhang, B. Fang, Y. Y. Tang, G. He, and J. Wen, "Topology preserving non-negative matrix factorization for face recognition," *IEEE Trans. Image Process.*, vol. 17, no. 4, pp. 574–584, April 2008. 11

[87] A. Diplaros, T. Gevers, and I. Patras, "Combining color and shape information for illumination-viewpoint invariant object recognition," *IEEE Trans. Image Process.*, vol. 15, no. 1, pp. 1–11, January 2006. 11

[88] A. Laika and W. Stechele, "A review of different object recognition methods for the application in driver assistance systems," in *Proc. of WIAMIS'07*, 2007, pp. 10–10. 11

[89] H. Kawai, Y. Inoue, R. Streitenberger, and M. Yoshimoto, "A highly parallel *dsp* architecture for image recognition," *IEICE Trans. Fundam.*, vol. E78-A, no. 8, pp. 963–970, August 1995. 11, 12

[90] N. K. Ratha, K. Karu, S. Chen, and A. K. Jain, "A real-time matching system for large fingerprint databases," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 8, pp. 799–813, August 1996. 11

[91] B. A. Draper, J. R. Beveridge, A. P. W. Bohm, C. Ross, and M. Chawathe, "Accelerated image processing on $fpgas$," *IEEE Trans. Image Process.*, vol. 12, no. 12, pp. 1543–1551, December 2003. 11, 12

[92] P. A. Ruetz and R. W. Brodersen, "Architectures and design techniques for real-time image-processing $ic's$," *J. Solid-State Circuits*, vol. SC-22, no. 2, pp. 233–250, April 1987. 11

[93] G. M. Bo, D. D. Caviglia, and M. Valle, "An analog *vlsi* implementation of a fature extractor for real time optical character recognition," *J. Solid-State Circuits*, vol. 33, no. 4, pp. 556–564, April 1998. 11, 57

[94] R. Karakiewicz, R. Genov, and G. Cauwenberghs, "480-$gmacs/mw$ resonant adiabatic mixed-signal processor array for charge-based pattern recognition," *J. Solid-State Circuits*, vol. 42, no. 11, pp. 2573–2584, November 2007. 11, 12, 57

[95] K. Miyazawa, K. Ito, T. Aoki, K. Kobayashi, and H. Nakajima, "An effective approach for iris recognition using phase-based image matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 10, pp. 1741–1756, October 2008. 12

[96] H. Ando, S. Kameda, D. Arizono, N. Fuchigami, K. Kaya, M. Sasaki, and A. Iwata, "Principal component analysis-based object detection/recognition chip for wireless interconnected three-dimensional integration," *Jpn. J. Appl. Phys.*, vol. 47, no. 4, pp. 2746–2748, April 2008. 12

[97] T. Shibata, "Intelligent signal processing based on a psychologically-inspired *vlsi* brain model," *IEICE Trans. Fundam.*, vol. E85-A, no. 3, pp. 600–609, March 2002. 12

[98] T. Shibata, M. Yagi, and M. Adachi, "Soft-computing integrated circuits for intelligent information processing," in *Proc. Int. Conf. Information Fusion*, 1999, vol. 1, pp. 648–656. 12, 14, 34, 35

[99] B. K. Kar and D. K. Pradhan, "A new algorithm for order statistic and sorting," *IEEE Trans. Signal Processing*, vol. 41, no. 8, pp. 2688–2694, August 1993. 17, 23, 40

[100] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: a review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 677–695, July 1997. 30

[101] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: a survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 237–267, February 2002. 31

[102] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "*monoslam*: real-time single camera *slam*," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, June 2007. 31

[103] I. Saleemi, K. Shafique, and M. Shah, "Probabilistic modeling of scene dynamics for applications in visual surveillance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 8, pp. 1472–1485, August 2009. 31

[104] A. Pentland, "Looking at people: sensing for ubiquitous and wearable computing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 107–119, January 2000. 31

[105] G. C. de Silva, T. Yamasaki, and K. Aizawa, "An interactive multimedia diary for the home," *Computer*, vol. 40, no. 5, pp. 52–59, May 2007. 31

[106] S. Kavadias, B. Dierickx, D. Scheffer, A. Alaerts, D. Uwaerts, and J. Bogaerts, "A logarithmic response *cmos* image sensor with on-chip calibra-

tion," *J. Solid-State Circuits*, vol. 35, no. 8, pp. 1146–1152, August 2000. 31

[107] M. Sakakibara, S. Kawahito, D. Handoko, N. Nakamura, H. Satoh, M. Higashi, K. Mabuchi, and H. Sumi, "A high-sensitivity *cmos* image sensor with gain-adaptive column amplifiers," *J. Solid-State Circuits*, vol. 40, no. 5, pp. 1147–1156, May 2005. 31

[108] M. F. Snoeij, A. J. P. Theuwissen, K. A. A. Makinwa, and J. H. Huijsing, "A *cmos* imager with column-level *adc* using dynamic column fixed-pattern noise reduction," *J. Solid-State Circuits*, vol. 41, no. 12, pp. 3007–3015, December 2006. 31

[109] H. Takahashi, T. Noda, T. Matsuda, T. Watanabe, M. Shinohara, T. Endo, S. Takimoto, R. Mishima, S. Nishimura, K. Sakurai, H. Yuzurihara, and S. Inoue, "A 1/2.7-in 2.96 *mpixelcmos* image sensor with double *cds* architecture for full high-definition camcorders," *J. Solid-State Circuits*, vol. 42, no. 12, pp. 2960–2967, December 2007. 31

[110] M. Mase, S. Kawahito, M. Sasaki, Y. Wakamori, and M. Furuta, "A wide dynamic range *cmos* image sensor with multiple exposure-time signal outputs and 12-bit column-parallel cyclic a/d converters," *J. Solid-State Circuits*, vol. 40, no. 12, pp. 2787–2795, December 2005. 31

[111] N. Akahane, S. Sugawa, S. Adachi, K. Mori, T. Ishiuchi, and K. Mizobuchi, "A sensitivity and linearity improvement of a $100-db$ dynamic range *cmos* image sensor using a lateral overflow integration capacitor," *J. Solid-State Circuits*, vol. 41, no. 4, pp. 851–858, April 2006. 31

[112] G. Storm, R. Henderson, J. E. D. Hurwitz, D. Renshaw, K. Findlater, and M. Purcell, "Extended dynamic range from a combined linear-logarithmic *cmos* image sensor," *J. Solid-State Circuits*, vol. 41, no. 9, pp. 2095–2106, September 2006. 31

[113] S. Yoshihara, Y. Nitta, M. Kikuchi, K. Koseki, Y. Ito, Y. Inada, S. Kuramochi, H. Wakabayashi, M. Okano, H. Kuriyama, J. Inutsuka, A. Tajima,

T. Nakajima, Y. Kudoh, F. Koga, Y. Kasagi, S. Watanabe, and T. Nomoto, "A 1/1.8-inch 6.4 *mpixel* 60 frames/s *cmos* image sensor with seamless mode change," *J. Solid-State Circuits*, vol. 41, no. 12, pp. 2998–3006, December 2006. 31

[114] "http://opencv.willowgarage.com/wiki/," . 31

[115] H. Zhu and T. Shibata, "A digital-pixel-sensor-based global feature extraction *vlsi* for real-time image recognition," in *Proc. International Conference on Solid State Devices and Materials (SSDM)*, 2008, pp. 476–477. 34, 58

[116] 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, no. 4, pp. 04C080–1–7, April 2009. 34, 58, 63

[117] H. Zhu and T. Shibata, "A real-time image recognition system using a global directional-edge-feature extraction *vlsi* processor," in *Proc. 35th European Solid-State Circuits Conference (ESSCIRC 2009)*, 2009, pp. 248–251. 34

[118] Y. Hori and T. Kuroda, "A 0.79-mm^2 20-mw real-time face detection core," *J. Solid-State Circuits*, vol. 42, no. 4, pp. 790–797, April 2007. 57

[119] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: a survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 1–45, 2006. 70, 80

[120] F. Jurie and M. Dhome, "Hyperplane approximation for template matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 996–1000, July 2002. 70

[121] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, May 2003. 70

[122] Y. Rathi, N. Vaswani, and A. Tannenbaum, "A generic framework for tracking using particle filter with dynamic shape prior," *IEEE Trans. Image Process.*, vol. 16, pp. 1370–1382, 2007. 70, 80

[123] J. Mutch and D. G. Lowe, "Multiclass object recognition with sparse, localized features," in *CVPR*, 2006, vol. 1, pp. 11–18. 71

[124] S. Hong, J. Lee, A. Athalye, P. M. Djuric, and W. Cho, "Design methodology for domain specific parameterizable particle filter realizations," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 54, no. 9, pp. 1987–2000, September 2007. 71, 81

[125] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, February 2002. 80

[126] X. Xu and B. Li, "Adaptive rao-blackwellized particle filter and its evaluation for tracking in surveillance," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 838–849, March 2007. 80

[127] O. Lanz, "Approximate bayesian multi-body tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1436–1449, September 2006. 80

[128] Y. Lao, J. Zhu, and Y. F. Zheng, "Sequential particle generation for visual tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 9, pp. 1365–1378, September 2009. 80

List of Publications

List of Publications Related to This Research

Journal Papers:

1. **Hongbo Zhu** and Tadashi Shibata, "A Digital-Pixel-Sensor-Based Global Feature Extraction VLSI for Real-Time Object Recognition," *Japanese Journal of Applied Physics.* vol. 48 no. 4, pp. 04C080-1∼7, 2009.

Refereed Papers at International Conferences:

1. **Hongbo Zhu** and Tadashi Shibata, "A Digital-Pixel-Sensor-Based Global Feature Extraction VLSI for Real-Time Image Recognition," in Extended Abstracts of the 2008 International Conference on Solid State Devices and Materials (SSDM 2008), pp. 476∼477, Tsukuba, Japan, Sep. 24-26, 2008.

2. **Hongbo Zhu** and Tadashi Shibata, "A Real-Time Image Recognition System Using a Global Directional-Edge-Feature Extraction VLSI Processor," in Proceedings of the 35th European Solid-State Circuits Conference (ESSCIRC 2009), pp. 248∼251, Athens, Greece, Sep. 14-18, 2009.

3. **Hongbo Zhu**, Pushe Zhao and Tadashi Shibata, "Directional-Edge-Based Object Tracking Employing On-Line Learning and Regeneration of Multiple Candidate Locations," Accepted for publication in the 2010 International Symposium on Circuits and Systems (ISCAS'10), Paris, France, May 30-June 2, 2010.

Other Publications

Journal Papers:

1. Hiroki Inubushi, Noriyuki Takahashi, **Hongbo Zhu**, and Kenji Taniguchi, "Ultrasonic 3D Image Sensor Employing PN code and Beam-Forming Technologies," *電子情報通信学会論文誌 A*. vol. J90-A no. 6, pp. 517∼523, 2007. (*in Japanese)*

Refereed Papers at International Conferences:

1. **Hongbo Zhu**, Hiroki Inubushi, Noriyuki Takahashi, and Kenji Taniguchi, "An ultrasonic 3D image sensor employing PN code", The 5th IEEE conference on sensors. pp. 319∼322, Daegu, Korea, Oct. 22-25, 2006.

Patent:

1. 谷口 研二、犬伏 裕基、高橋 紀行、朱 弘博、古橋 壮之, 「超音波センサ、超音波立体視装置及び距離方向特定方法」 特開2007－278805