Proceedings of the 2002 IEEE
International Conference on Robotics & Automation
Washington, DC • May 2002

A Real-Time Visual Processing System using a General-Purpose Vision Chip

Shingo Kagami†, Takashi Komuro††, Idaku Ishii†††, Masatoshi Ishikawa††

† Department of Mathematical Engineering and Information Physics, University of Tokyo
†† Department of Information Physics and Computing, University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

††† Department of Electrical and Electronic Engineering, Tokyo University of Agriculture and Technology
2-24-16 Naka-cho, Koganei-shi, Tokyo 184-8588, Japan

swk@k2.t.u-tokyo.ac.jp

Abstract

A real-time visual processing system using a general-purpose vision chip, an image sensor in which photo detectors and processing elements are integrated, is described. In order to control the vision chip and process its output at high speed, a novel architecture called SPARSIS, in which control process of the vision chip is pipelined and integrated with a RISC type integer pipeline, has been developed. This architecture can guarantee real-time operation with high temporal resolution, and even makes possible software-controlled A/D conversion. Sample algorithms demonstrating its fine-grained realtimeness, and experimental results with the implemented system, are also described.

1 Introduction

General-purpose real-time vision systems are needed for various applications, such as robot vision, multimedia, human computer interaction, and so on. Many real-time vision systems have been developed, but most of them use video signals as their image input and their frame rates are limited by the video frame rate. The video frame rate is not fast enough, for example, for high-speed visual feedback control of robots, thus preventing the systems from working in a real and unknown environment.

To overcome this limit, digital vision chips, CMOS image sensors that integrate a digital processing element (PE) directly connected to a photo detector (PD) in each pixel, have been developed[1, 2, 3]. Since they have no bottleneck in image input, they can perform real-time visual processing at high frame rates. For example, Komuro et al. designed S^{3}PE architecture, aimed at achieving general-purpose visual processing on a focal plane at a frame rate of over 1000 fps[3]. Many application-oriented works have shown that the high speed vision over 1000 fps is highly effective in robotics applications, by demonstrating high speed target tracking, grasping, gesture recognition, microscopic target controlling, and so on[4, 5, 6, 7, 8].

In this paper, a real-time visual processing system employing a general-purpose vision chip is described. In order to fully exploit its potential, it must be combined with an efficient control structure. From an architectural point of view, a vision chip is a mesh-connected SIMD PE array, that needs to be supplied with instructions at a rate high enough to accomplish image processing tasks within a frame cycle time – typically 1 ms. Furthermore, these instructions must specify not only processing operations but also timing of photo detector control, on which strict real-time restrictions of extremely fine granularity are imposed.

The system we developed consists of a vision chip and its dedicated microcontroller. The vision chip controller is a newly designed RISC type 32-bit processor in which control path and data path for the vision chip are integrated. It is designed to achieve a high instruction rate, and at the same time to guarantee real-time instruction issuing with the resolution of the instruction cycle time. Since it can autonomously perform program control, processing of data extracted from the PE array, or anything needed to utilize the vision chip, application systems equipped with this vision system can devote themselves on their own application tasks.

In the next section we give the whole system overview. Following that, we describe the vision chip controller core architecture, SPARSIS, in detail, followed by section 4 that presents some sample algorithms. In section 5, implementation of the system and some experimental results are shown. The final two sections summarize related works and contain concluding remarks.

2 System Overview

The vision system is an integrated compact module including a digital vision chip and its control unit. An

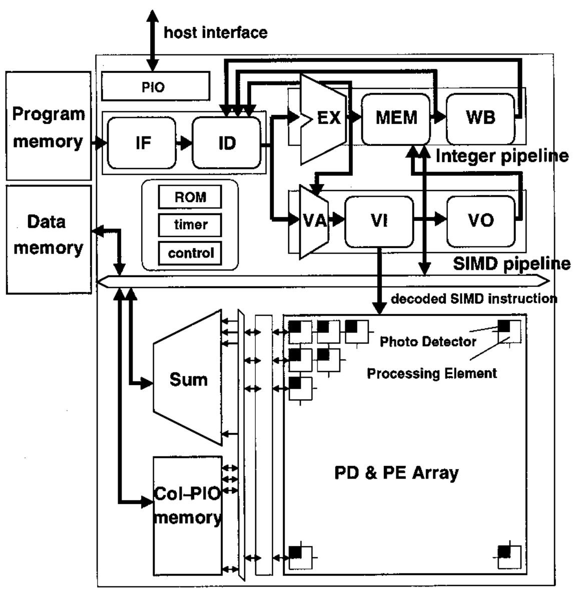

Fig. 1 Block diagram of the system.

overview of the designed vision system is shown in Fig. 1.

2.1 Vision Chip Architecture

To enable general-purpose visual processing on a focal plane, we need very compact pixel circuit design that maintains generality in processing. Our research group has been working on the design of a vision chip architecture to meet this demand, and the development of its VLSI implementation. The vision chip architecture, S^3PE[3], is shown in Fig. 2. It is a SIMD controlled, 4-neighbor connected PE array. Each PE has a bit-serial ALU, and a local memory with 24-bit bitwise random access memory and 8-bit memory-mapped I/O ports. It accepts a 3-operand type instruction word, and every PE executes the specified operation simultaneously.

Since digital processing is carried out in each pixel, pixel-level analog-to-digital (A/D) conversion of image input is needed. This task is performed by software utilizing the functionality of the PE. The block diagram of the PD is shown in Fig. 3. First, the photodiode voltage V_{PD} is reset and charged up to V_{DD}. After disconnecting from V_{DD}, V_{PD} is discharged by photocurrent. The comparator output takes two values, namely, logical 0 and logical 1. It holds logical 0 until the V_{PD} drops under V_{ref}, and then it turns to logical 1. A/D conversion is accomplished by polling the comparator output and measuring the time between the reset and the comparator inversion. This contributes to not only saving the circuit's area needed

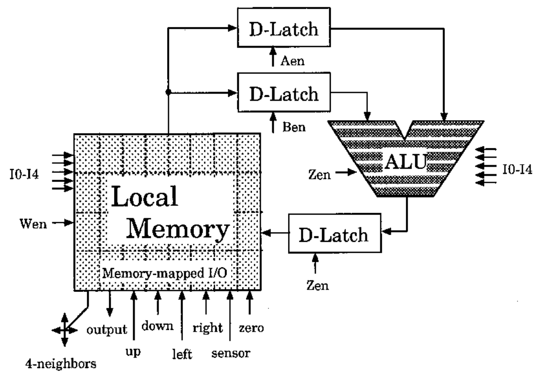

Fig. 2 Processing element in each pixel.

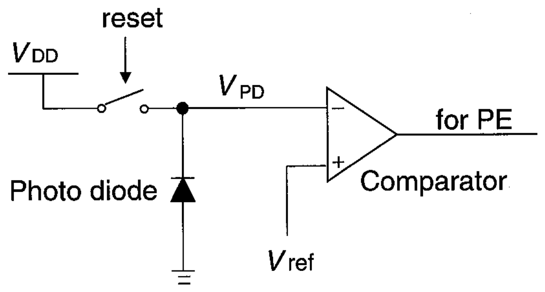

Fig. 3 Block diagram of the photo detector.

by ADC, but also enables programmable and flexible image acquisition.

2.2 Vision Chip Controller

The vision chip PD and PE array itself does not have control structures. A dedicated microcontroller has been designed to manage the whole system and provide instruction words to the vision chip. The vision chip controller is a 32-bit RISC type microprocessor that carries out (I) integer processing, (II) program control operations such as branches and function calls, (III) SIMD instruction to the vision chip, and (IV) parallel data I/O with the vision chip and feature value extraction. Since the system is controlled by itself without the help of external systems, it can be used as an independent vision module. Application systems that use this vision system need only to store programs in the program memory of the system beforehand, and communicate with it at some arbitrary rate – for example, at the rate of a robot control loop. The controller architecture is described in detail in the following section.

A typical system operation cycle can be described as follows:

(1) image acquisition,

(2) pixel-level parallel processing of the image,
(3) feature extraction from the processed image,
(4) sequential processing without parallelism,
(5) feedback to (1) or (2).

Consider an application to robot control using visual feedback for example. In (1) and (2) an image is acquired and A/D converted, and procedures exploiting data parallelism, such as filtering operations and correlating operations, are carried out. In (3) and (4) feature values such as centroids of the targets are extracted.

The processed results, through some sequential operations if needed, can be used directly as command values for actuators, and as feedback information to proces s (1) or (2) in the form of procedural changes or parameter updating and so on.

An important point to be emphasized is that all the above processes, including image acquisition, are fully programmable. This feature brings flexible and adaptive image sensing and visual processing in compliance with environmental conditions or objectives.

3 Control Architecture

3.1 Design Strategy

The major difficulty in introducing an efficient control structure lies in its tight real-time restriction. Since what we need is a real-time vision system that operates at a frame rate over 1000 fps, image-processing cycles must be accomplished within this frame time.

Moreover, the software-controlled A/D conversion, described in the previous section, demands much finer granularity in real-time operation. For example, when obtaining an 8-bit gray scale image with 1 ms photo exposure time, A/D conversion is typically done as follows: An instruction reads out the output of the PD comparator, and the subsequent instructions add it to a counter variable. These operations are repeated 255 times within 1 ms (in general, $2^n - 1$ times to obtain n-bit gray scale). The timing of each PD-reading instruction must be deterministic; otherwise the conversion results are unstable and not dependable. Conversely, if highly fine-grained real-time operations with high temporal resolution are guaranteed and the PD-reading time can be actively controlled, advanced imaging features such as widened dynamic ranges and flexible adaptation to environments can be offered.

To improve the temporal resolution in providing instructions, pipelining is effective since it can increase the number of instructions issued in a unit of time. However, pipeline design of conventional microprocessors comes with run-time dynamic pipeline interlocks, and this cannot satisfy the realtimeness needed by the vision chip.

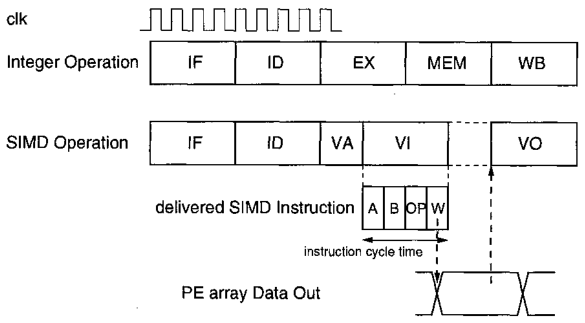

Fig. 4 Pipeline operation timing.

As a solution, we propose an architecture called SPARSIS, standing for Stall-less Pipeline Architecture for Real-time Sensory Information processing System, in which control process of the vision chip is pipelined and integrated with a RISC type integer pipeline; It is designed so that any combinational use of both pipelines never causes dynamic pipeline stalls.

3.2 SPARSIS Architecture

The instruction set architecture of SPARSIS is a RISC type load/store architecture with extra SIMD instructions to be sent to the PE array. As shown in Fig. 1, SPARSIS has two distinct pipelines: an integer pipeline and an SIMD pipeline. The operation timing is depicted in Fig. 4.

After fetching and decoding, integer instructions are issued to the integer pipeline. The integer pipeline is a conventional 5-stage one that performs sequential operations and program controls. It is used for generating data required by applications from results processed by the PE array, as well as handling control variables such as loop counters and branch conditions.

On the other hand, SIMD instructions are issued to the SIMD pipeline after decoding. This is used for vision chip control, that is, generating the instructions in the VA stage, sending them to the PE array in the VI stage, and receiving and postprocessing the processed results in the VO stage if needed.

This architecture adopts 1-slot delayed branch, 1-slot delayed load, and memory access with uniform latency to avoid any dynamic pipeline stalls. Data forwarding paths are introduced not only intra-pipeline but also inter-pipeline to eliminate data hazards caused by the data dependencies between successive integer and SIMD operations.

In this way, temporal resolution of instruction delivery is improved by the pipelining of vision chip con-

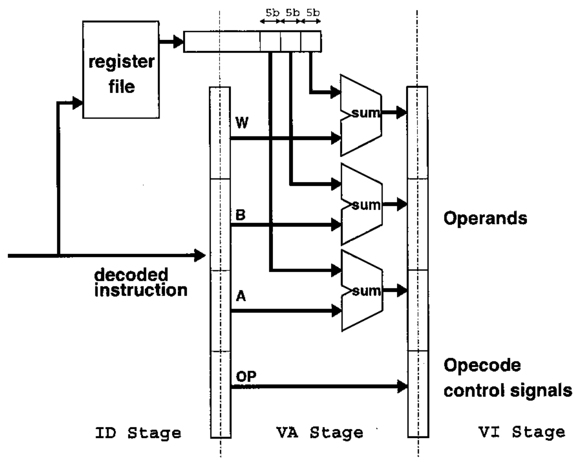

Fig. 5 Displacement addressing for the SIMD instruction.

trol, and at the same time hard real-time instruction issuing is guaranteed by excluding any speculative factors. These efforts can consequently offer fine-grained real-time operations with the resolution of the instruction cycle that can never be achieved by conventional processors with real-time OS.

3.3 SIMD Pipeline Details

In the VA stage of the SIMD pipeline, address calculation for SIMD instructions is carried out. An SIMD instruction stored in the program memory consists of an opecode, three operand displacements, an integer register number, and some miscellaneous control bits. The displacements are, as shown in Fig. 5, added to the corresponding 5-bit fields of a value fetched from the register file, and absolute addresses used in the PE array are generated.

In the VI stage, the generated SIMD instruction is delivered to the PE array, where all the PE operate in parallel. Since one instruction of S^3PE consists of four clock cycles, a cycle time for one pipeline stage is divided into four, and the corresponding instruction fragments are sent.

Half a stage after, the output of the PE array, which can be obtained from the PEs located on the edge line of the array, is latched by dedicated data I/O units if needed. In the current design, data I/O units include a column-parallel I/O unit and a summation output unit shown in Fig. 1 as Col-PIO memory and Sum respectively. The former is a buffer memory with width-converting logic between the integer data path and the PE array output. This can also be used to provide input data for the PE array along with the instruction signals.

```
repeat forever
        parallel integer AD_convert_result = 0;
        integer feature_value;

        // A/D Conversion
        Reset_PD;
        repeat (grayscale_level − 1) times
                Wait_to_adjust_timing;
                AD_convert_result += Read_PD_output;
        endrepeat

        // Image Processing
        Do_some_processing(AD_convert_result);
        feature_value = Feature_extract(AD_convert_result);

        Synchronize_and_write_result(feature_value);
endrepeat
```

Fig. 6 Sample description of an algorithm.

The latter is used to extract feature values from the PE array in the form of weighted summations of all the pixel values. It consists of a tree-adder that sums up the output from the PE array in the VO stage, and a register accumulating its output. This can remove output bottlenecks because feature values required by applications, for example moment values including centroid, area and variance used in visual servoing, can be obtained directly without transferring raw images to applications. In our next design, global summation facility is going to be integrated with the PE array itself. This will produce much faster feature value extraction.

4 Sample Algorithms

A typical example of a visual processing procedure is shown in Fig. 6. In this example, an input image is A/D converted to a digital image in the inner **repeat** loop, and then extracted feature values from the digitized and processed image are passed to external systems. Image processing is mainly carried out by the PE array, where the high operating frequency brought by the pipelined control structure offers large throughput.

By varying the wait intervals in the A/D conversion loop, flexible image acquisition can be realized. The A/D conversion procedure described in Section 3.1 can be formulated as follows: Assuming the photodiode is reset at $t_0 = 0$ and the comparator output is read out at $t_1, t_2, \cdots, t_{n-1}$ to obtain an n-level gray scale image, mapping $\mathcal{D}$ from photocurrent i to a digital value is given by

$$\begin{aligned} \mathcal{D}(i) &= n-k \quad \text{if } i \in [i_k, i_{k-1}), \\ \text{where } i_0 &= \infty \end{aligned}$$

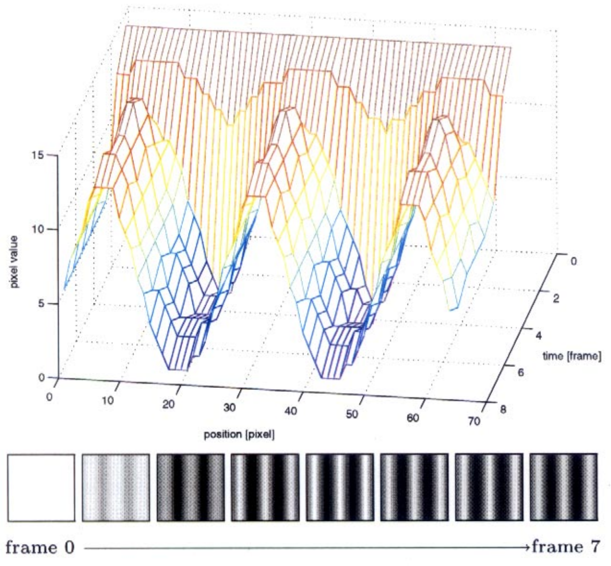

Fig. 7 Simulated result of adaptive image sensing.

$$\begin{aligned} i_k &= \frac{C(V_{\mathrm{DD}} - V_{\mathrm{ref}})}{t_k} \quad (k = 1, \cdots, n-1) \\ i_n &= 0. \end{aligned}$$

C is the photodiode capacitance. When a desired A/D conversion scale $\{i_k\}$, such as linear-to-intensity or logarithmic-to-intensity is given, PD-reading time sequence $\{t_k\}$ can be obtained immediately from the above relation. Temporal resolution of instruction issuing and its realtimeness play an important role here because they actually influence the feasibility of the obtained $\{t_k\}$.

The sequence $\{t_k\}$, thus the scale $\{i_k\}$ can be varied frame by frame. For instance, light-adaptive image sensing that performs histogram equalizing can be realized by recalculating $\{t_k\}$ from the previous cycle's $\{t_k\}$ and histogram information:

$$t^{\mathrm{next}}{}_k = t_{l-1} + \frac{(k/n)S_n - S_{l-1}}{S_l - S_{l-1}}(t_l - t_{l-1}),$$

where S_k is the number of the pixels in which the pixel values are larger than $(n-k-1)$, and l is a minimal number such that $S_l \geq (k/n)S_n$.

A simulated result for an input image that has sine-like intensity with a small amplitude is shown in Fig. 7. We started with the initial $\{t_k\}$ that has uniform intervals. Though the digitized 4-bit image had a low contrast at first, it adapted to the light in a few cycles. The simulation assumes the temporal resolution of PD-control is 1/1000 of the exposure time.

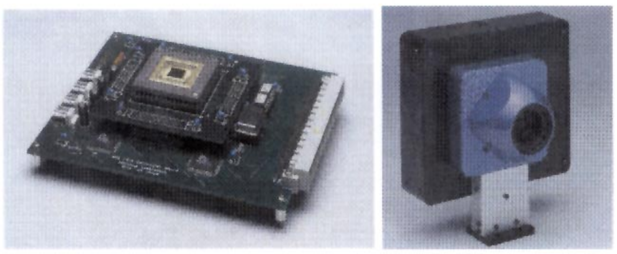

Fig. 8 Photographs of the developed system.

Table 1 Execution times of sample programs.

program	steps	(time)
dilation, erosion(binary)	5	(0.5 μs)
4-neighbor x, y edge (binary)	9	(0.9 μs)
4-neighbor smoothing(4bit)	59	(5.9 μs)
4-neighbor smoothing(8bit)	99	(9.9 μs)
0th moment[*1](binary input)	132	(13.2 μs)
1st moment[*1](binary input)	792	(79.2 μs)
centroid detection[*1](binary input)	1716	(171.6 μs)

[*1] For 64 × 64 pixels.

5 System Implementation

Based on the architecture stated above, we implemented a prototype vision system including a vision chip and a controller. The vision chip is a test chip with 64 × 64 integrated pixels, fabricated using 0.35μm CMOS technology[8]. The controller logic is designed in Verilog-HDL and synthesized for a XILINX FPGA, XC4062XLA. The controller circuit includes approximately 27000 logic gates. This number is less than one tenth of the gate count of the vision chip, and it is not unrealistic for the controller to be integrated with the PE array on a chip. The implemented system is shown in Fig. 8.

The controller operated at 40 MHz, thus the instruction rate was 10 MHz. This allows hard real-time operations with temporal resolution of 100 ns. For example, PD-reading timings can be scheduled with resolution of 1/10000 of the exposure time, assumed to be 1 ms.

The execution times for some sample programs are shown in Table 1. Basic filtering operations can be performed within several μs, and even global feature extraction operations can be carried out within several tens to a few hundreds of a μs. The operating frequency is expected to be much higher when the controller is integrated with the PE array. However, it is high enough to be used for many real applications.

Image sensing and processing results are shown in

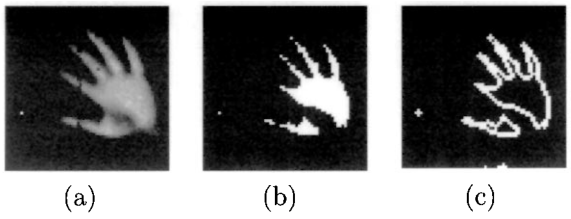
Fig. 9 Experimental results of early visual processing. (a) 4-bit digitized image. (b) Binarized image. (c) Edge-detected image.

Fig. 9. A human hand is captured and digitized as a 4-bit gray scale image with exposure time of 1 ms, and binarization and edge detection are performed. Because the current implementation of the vision chip has a problem with local memory access and is not tolerant of a high clock rate, the experiment was carried out at 10 MHz. We will fix the problem in the next fabrication.

6 Related Works

SIMD-based parallel processing is popular in image processing applications, and much effort has been made to introduce efficient control structures for it. For compact and low-cost image processing systems in particular, recent studies have proposed simple control structures based on a macro-instruction expansion scheme[9, 10]. Their simple architectures are efficient at achieving high PE array utilization, however, they cannot guarantee their real-time operations since their program controls are dependent upon host systems, and are not suitable for vision chip control.

The demand for instruction-level realtimeness is a new viewpoint that is peculiar to vision chips. This demand has motivated us to introduce a new dedicated architecture that is more focused on guaranteeing fine-grained real-time instruction issuing rather than merely achieving large throughput. The architecture proposed in this paper is based on a fully equipped microprocessor type control structure. As shown in the previous section, recent VLSI technology allows us to implement a microcontroller with specialized functions at low cost, and employing such a sophisticated controller is no longer unrealistic.

7 Conclusion

A real-time visual processing system consisting of a general-purpose digital vision chip and its dedicated control structure has been described. The system architecture is described in detail, and sample algorithms and experimental results with an implemented prototype are also presented. We believe the system will have various uses as a compact and low-cost vision sensor module with full programmability – not only in processing but also in sensing itself.

References

[1] Thierry M. Bernard, Y. Zavidovique, and Francis J. Devos, "A programmable artificial retina," *IEEE Journal of Solid-state Circuits*, vol. 28, no. 7, pp. 789–798, 1993.

[2] Robert Forchheimer and Anders Åström, "Near-sensor image processing: A new paradigm," *IEEE Transactions on Image Processing*, vol. 3, no. 6, pp. 736–746, 1994.

[3] Takashi Komuro, Idaku Ishii, and Masatoshi Ishikawa, "Vision chip architecture using general-purpose processing elements for 1ms vision system," in *4th IEEE International Workshop on Computer Architecture for Machine Perception (CAMP'97)*, 1997, pp. 276–279.

[4] Yoshihiro Nakabo, Idaku Ishii, and Masatoshi Ishikawa, "High speed target tracking using 1ms visual feedback system," in *Video Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, 1996, p. 6.

[5] Akio Namiki, Yoshihiro Nakabo, Idaku Ishii, and Masatoshi Ishikawa, "1ms sensory-motor fusion system," *IEEE Transactions on Mechatronics*, vol. 5, no. 3, pp. 244–252, 2000.

[6] Masatoshi Ishikawa, "1ms VLSI vision chip system and its application," in *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 214–219.

[7] Hiromasa Oku, Idaku Ishii, and Masatoshi Ishikawa, "Tracking a protozoon using high-speed visual feedback," in *Proceedings of 1st Annual International IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology*, 2000, pp. 156–159.

[8] Masatoshi Ishikawa and Takashi Komuro, "Digital vision chips and high-speed vision systems," in *Digest of Technical Papers of 2001 Synposium on VLSI Circuits*, 2001, pp. 1–4.

[9] Jeffrey C. Gealow, Frederick P. Herrmann, Lawrence T. Hsu, and Charles G. Sodini, "System design for pixel-parallel image processing," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 4, no. 1, pp. 32–41, 1996.

[10] Martin C. Herbordt, Jade Cravy, Honghai Zhang, Calvin Lin, and Hong Rao, "An array control unit for high performance SIMD arrays," in *5th IEEE International Workshop on Computer Architecture for Machine Perception (CAMP2000)*, 2000, pp. 293–301.