시각매체를 위한 병렬처리 시스템

정회원 이 형*, 박 종 원**

A Parallel Processing System for Visual Media Applications

Hyung Lee*, Jong-won Park** *Regular Members*

요 약

영상과 그래픽 및 비디오와 같은 시각 매체들을 실시간으로 처리하기 위한 구현 기술과 그에 따른 확장성 측면에서 많은 연구들이 진행되고 있는데, 이러한 연구들은 영상처리 전용 프로세서 구현부터 다양한 매체들을 함께 처리할 수 있는 프로세서 구현을 포함하는 범주까지 진행되고 있다. 또한, 다양한 병렬처리 기법들이 실시간 처리를 위한 프로세서의 구현에 적용되고 있다. 본 논문은 이러한 시각매체들을 실시간으로 처리하기 위하여 메모리 시스템과 다수개의 처리기로 구성된 pipelined SIMD 구조를 갖는 병렬처리시스템을 제안한다. 메모리시스템은 m개의 메모리 모듈과 메모리 제어기로 구성되어 있는 다중접근 기억장치로써, m개의 메모리 모듈에서 병렬로 n(=p×q)개의 데이터에 접근하기 위한 다양한 형태, 즉, 행(1×pq)과 열(pq×1) 및 블록(p×q) 접근을 제공한다. 제안한 병렬처리시스템에 얼굴인식과 퐁 음영 및 동영상에서의 자동영상분할을 적용하여 시스템 성능을 분석하였다.

ABSTRACT

Visual media(image, graphic, and video) processing poses challenge from several perspectives, specifically from the point of view of real-time implementation and scalability. There have been several approaches to obtain speedups to meet the computing demands in multimedia processing ranging from media processors to special purpose implementations. A variety of parallel processing strategies are adopted in these implementations in order to achieve the required speedups. We have investigated a parallel processing system for improving the processing speed of visual media related applications. The parallel processing system we proposed is similar to a pipelined single instruction multiple data(SIMD) architecture that consists of processing elements(PEs) and a multi-access memory system(MAMS). The multi-access memory system is made up of m memory modules and a memory controller to perform parallel memory access with a variety of combinations of 1×pq, pq×1, and p×q subarray, which improves both cost and complexity of control. Facial recognition, Phong shading, and automatic segmentation of moving object in image sequences are some that have been applied to the parallel processing system and resulted in faithful processing speed. This paper describes the parallel processing systems for the speedup and its utilization to three time-consuming applications.

I. Introduction

Multimedia processing is becoming increasingly important because of the wide variety of application. Each media in a multimedia environment require different processes, techniques, algorithms and hardware. Hence, it is crucial to design processor architectures that meet the computing requirements of the various media type and the convenient data structures for a given class of applications implemented at the hardware. In the case of visual media in a multimedia system, it contains a significant amount of information, and correspondingly involves a large volume of data

* 대전보건대학 방송제작기술과(hyung@djhealth.ac.kr)
** 충남대학교 정보통신공학과(jwpark@crow.cnu.ac.kr)
논문번호 : 010075-0426, 접수일자 : 2001년 4월 26일

in contrast to the other media. The complexity, variety of techniques and tools, and the high computation, storage and I/O bandwidths associated with visual processing pose several challenges, particularly from the point of view of real-time implementation.

In recent years many researchers have been interested in visual media related to applications, speech analysis and synthesis, character recognition, audio compression, graphic animation, 3-D rendering, image enhancement and restoration, image/video analysis and editing[1,2]. Although tremendous amount of work was done in building application systems, there have been many difficulties in processing on the developed system in real-time. One of them is low processing speed caused by a significant amount of information.

There already exit a variety of machines that are capable of performing high-speed image applications[3,4,5,6,7,8]. In general, these machines can be divided into two classes: the first basically comprises 2-D array processors that operate on an entire image or subimage in a set of parallel processes. Examples of this type of machine are CLIP[3], MPP[4], and PIXIE-5000[5]. In general, all of these machines can also be considered as being the SIMD type. The main drawback of 2-D array processors is their cost. In addition, due to the inherent serial nature of the input-image data, full utilization of the processors may not be attained.

The second class of machines - local-window processors - scan an image and perform operations on small neighborhood window. Examples of such machines include MITE[6], PIPE[7], and Cytocomputer[8]. Note that, with this type of processor, an increase in image size requires a quadratic increase in processor speed in order to maintain a constant processing speed. Most of the above local-window processors are general purpose in nature in that they are programmable. Although theses general-purpose cellular machines are flexible because of their programmability, they do not provide the capability of the simultaneous accesses of image points in a line-segment with an arbitrary degree, which is required in many low-level image processing operations such as edge or line detection in a particular direction, and so on

We have been developing the parallel processing system to improve processing speed during applications using an architecture similar to a pipelined SIMD and an MAMS for satisfying to provide the capability of the simultaneous accesses of image points. In this paper, we propose a parallel processing system involving MAMS[9,10] for accessing the data elements within three access types with a constant interval simultaneously. Also, parallel algorithms for facial recognition, Phong shading, and automatic segmentation of moving object in image sequences to be applied to the system are suggested.

This paper is organized as follows. The parallel processing system and MAMS we proposed are detailed in Section II and Section III, respectively. Section IV presents the simulation of the proposed system with morphological filters. Section V presents three applications and how to apply them to the parallel processing system. Finally, the conclusions are provided in Section VI.

Ⅱ. Parallel Processing System

If an algorithm consists of many identical operations on different image points, an SIMD architecture in which one processor unit operates n processing elements(PEs) synchronously will be adequate to the algorithm. If an algorithm needs to access data in any direction with an interval corresponding to the length of between data, an MAMS , which provides simultaneous access to n data elements with access types, will be adequate to the algorithm. The type may be a block, a horizontal or a vertical with an interval. Their properties are involved in some image processing algorithm or some parts of an algorithm.

The parallel processing system we proposed consists of a processing unit (i960VH Embedded-PCI Processor); a local memory which stores instructions and common data for its programmability; DMA controller which has set of registers

to fetch instructions from the local memory and issue them to n PEs; PE which interprets and executes instructions synchronously; a multi-access memory controller(MAMC) which provides n data elements to n PEs simultaneously; and m external memory modules which store n data elements to be manipulated. The processor unit controls the system and communicates with host computer via PCI bus. DMA controller not only fetches an instruction and stores it in a register pool but also synchronously transfers data or an instruction to n PEs and controls them. PE was designed as ALU with the functionality that is interpreting an issued instruction. And, in order to perform an application, DMA controller steels bus cycles until the end of the application.

PE can perform executions of two kinds of instructions: one is memory-reference instructions for accessing spreaded data in m external memory modules via MAMC, the other includes 16 general instructions including register-reference instructions and I/O instructions. So, An application to be processed on the system is compiled to operation codes within 18 instructions. And, when each of two instructions is in different set of instructions, they are executed at the same time. That is, one of memory-reference instructions and one of general instructions are executed simultaneously. Hence, a memory-reference instruction followed by a general instruction is executed in a memory access cycle. It causes processing time in some modules to be reduced, for example, convolution mask operations frequently used in spatial domain.

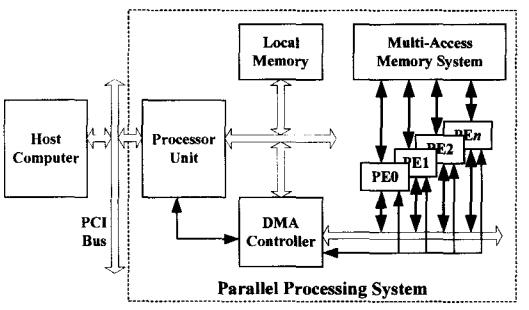


그림 1. 제안한 병렬처리시스템의 블럭도

The system provides logically two-dimensional addressing mode that is used in (r,c)-based image domain. Therefore, most of image processing in spatial domain are done with enough parallel processing power. The block diagram of the system is depicted in Figure 1.

Ⅲ. Multi-Access Memory System

For a parallel processing system with PEs, it is necessary to use an MAMS[9,10] to reduce the memory access time. Also, the memory system has the important goals to provide the efficient utilization for n(=pq) PEs of the parallel processing system we proposed, where p and q are design parameters. The goals are as follows: various access types and constant interval between the data elements, simultaneous access with no restriction on the location, simple and fast address calculation and routing circuitry, and small number of memory modules.

The memory system consists of a memory module selection circuitry, a data routing circuitry for WRITE, an address and a routing circuitry, memory modules, and a data routing circuitry for READ. In order to distribute the data elements of the M×N array I(*,*) among m memory modules, a memory module assignment function must place in distinct memory modules array elements that are to be accessed simultaneously. Also, an address assignment function must allocate different addresses to array elements assigned to the same memory module.

Memory Module Assignment Function

A memory module assignment function, which determines the index of memory module of an element, is μ (i,j)=(iq+j)//m. Here the notation x//y is used to denote the non-negative remainder that results from the integer division of x by y.

Address Assignment Function

The address assignment function which determines the address of an element within a memory module is α (i,j)=(i/p)×s + j/q, where s is any

integer satisfying $s \geq \lceil X \rceil \times N/2q$ and $s \times M/2p \leq c$, where c is the capacity of each memory module.

Address Calculating Circuit

This subsection discusses the address calculating circuit that computes pq addresses of a row with a constant interval r. The differences of pq addresses of the pq data elements within a row with a constant interval r from the base address, $\alpha(i,j)$, are

$$AROW(i,j,r,a)=\alpha(i,j+ar)-\alpha(i,j)=(j//q+ar)/q, 0 \leq a < pq \quad (1)$$

The address calculating circuit computes m addresses by using the m adders provided that the base address $\alpha(i,j)$ and the address differences are supplied to input A and B of the m adders, respectively.

Address and Data Routing

The address routing circuit receives m addresses from the register A1 in the address calculating circuits and moves the m addresses to the m memory modules through the register A2. The index numbers of the memory modules for the m addresses are represented as follows for the different access patterns, where the interval r>0 and $r//m \neq 0$:

$$INROW(i,j,r) = (\mu(i,j)+br)//m, \ 0 \leq b \leq pq-1 \quad (2)$$

The address difference of the row the μ th module ADROW(μ) is obtained from (1) and (2) :

$$ADROW(\mu)=(i//q+((\mu-iq-i) \times r'//m) \times r/q, \ 0 \leq \mu < pq \quad (3)$$

By substituting $(\mu+\mu(0))//m(=((\mu-iq-j) \times r'//m+(iq+j)//m)//m)$ instead of μ into (3), we get the address differences in ascending order of memory modules from $\mu(0)$ as follows:

$$ADROW(\mu+\mu(0)//m)=(j//q+(((\mu-iq-j) \times r'//m+(iq+j)//m)//m) \times r)/q, \ 0 \leq \mu < pq, \text{ where } r \times r'=1//m. \quad (4)$$

For routing the calculated addresses to the memory modules, it is required for the address differences stored in the address difference memory module to be rotated by the index number of the memory module in the first data element. Prearranging and storing the address differences in the address difference memory module make the address routing circuitry inexpensive and simple to control. The role of the data routing circuit is to align data elements read from or to the memory modules. After aligning the pq data elements as follows, right-rotation is perform by $\mu(i,j)$ times in order for the index numbers of the memory modules of those data elements to be in the ascending order: $D2((kr)//m) \leftarrow D1(k)$, $0 \leq k \leq pq-1$, where D1 and D2 registers are the data register and a temporary register, respectively. For the READ operations, the routing patterns are reversed.

Memory Module Selection

The memory module selection circuit enables pq memory modules whose index numbers are represented in (4).

Two ROM are needed in implementation of the proposed MAMS. One ROM is for storing results of the memory modules selection functions because it takes a higher space complexity on the device. The other is for synchronizing three modules. In order to obtain valid addresses to each memory modules, the time complexity of address calculation and routing module is higher than other two modules. One of two is for basic addresses to each memory modules and the other for $\alpha(i,j)$ in the equation (1). The MAMC we redesigned is implemented to the pipelined with three stages because PEs are designed as pipelined. Therefore, in the case of sequential memory operations, the memory access time is reduced in comparing with that of the original. The block diagram of the MAMS is presented in Figure 2.

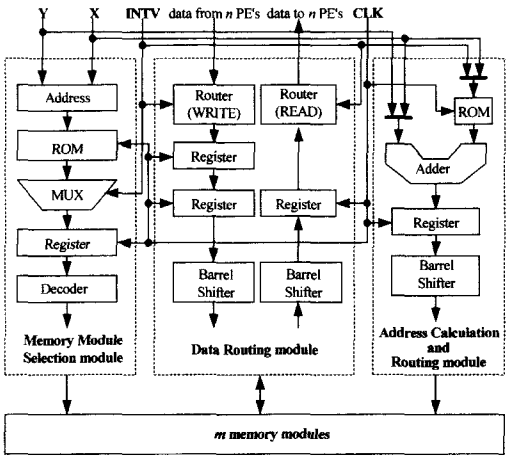


그림 2. 제안한 다중접근 기억장치의 블럭도

Ⅳ. Simulations

To verify the performance of the proposed system, we chose morphological filters based on mathematical morphology that are very effective tool for extracting structural information from image signal. The morphological approach to unsupervised image segmentation is very attractive because it is simple, robust and well adapted to many different types of images. The algorithm[11] is based on four steps: image processing, feature extraction, decision and quality estimation. The preprocessing step is a simplification phase that generates regions with constant graylevel values. It is achieved by the use of grayscale morphological filters by reconstruction[12], which preserves the contour information of the original image. Although its result was very faithful, it is one of time-consuming works.

To achieve the parallel version of the simplification filter consists of erosion and dilation, which are most basic operators, we tried to transform the codes of both operators into available codes to the proposed system. The codes for the erosion and dilation operators are presented in Code 1 and 2, respectively. Notice that two instructions in each row, the left is of memory-reference instructions and the other is of general instructions, are executed simultaneously.

Code 1. Erosion

```
Read(1,0,0)        ValTran AC 256
Read(1,1,0)        Cond1 rd_reg
Read(1,2,0)        Cond1 rd_reg
                   ...
Nop                Cond1 rd_reg
Write(1,1,1)       Nop
```

Code 2. Dilation

```
Read(1,0,0)        ValTran AC 0
Read(1,1,0)        Cond2 rd_reg
Read(1,2,0)        Cond2 rd_reg
                   ...
Nop                Cond2 rd_reg
Write(1,1,1)       Nop
```

The dedicated parallel processing system was described by Verilog-HDL and simulated by CADENCE Verilog-XL hardware simulation package in order to verify its functionality. And the system was compiled and fitted into EFP10K200 EGC599-1 device by MAXPLUSII. The final simulation was performed after getting delay files and doing back-annotation. The waveform generated during the simulation is illustrated in Figure 3. The waveform shows that fetching an instruction, which is one of memory access instructions, and one of arithmetic logic instructions were overlapped. And, we know that execution time of an application applied to the proposed system was depended on the memory access time.

When the size of structuring elements is 3×3 and the system clock is 33MHz, the estimation of both operators processed on the system is as follows.

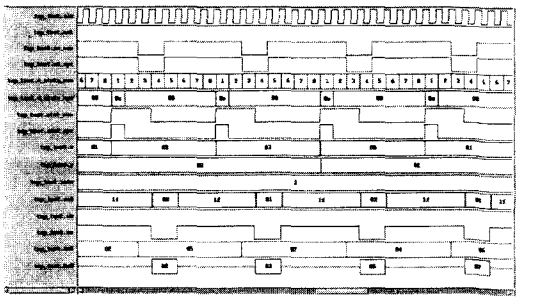


그림 3. post-layout 모의실험으로 얻은 파형

$$T_{total} = 9 \times T_{read} + 1 \times T_{cond} + 1 \times T_{write}$$
$$= 9 \times 8 \times 30(ns) + 2 \times 30(ns) + 6 \times 30(ns)$$
$$= 2400(ns)$$

where T_{read} is the number of clocks to perform read cycle, T_{cond} a conditional operation, T_{write} write cycle. As the same way, when the size of structuring elements is 5×5, the processing time is calculated by 6.24μ s. Table 1 shows estimated processing times on the proposed system.

i960RM SDK board[13], which operates up to 33MHz Bus Clock, was chosen as the target system to be compared with the proposed system because the execution speed and the memory access time are similar to ones of the proposed system. Table 2 shows estimated processing times on i960RM SDK. Experimental results indicate that the proposed parallel processing system is very powerful compared to general purposed computers.

표 1. 제안한 시스템에서 측정한 항목별 처리 시간(sec)

Image Size	SE Size	Erosion	Dilation	Edge Detection
QCIF	3×3	0.0076032	0.0076032	0.017455
QCIF	5×5	0.0197683	0.0197683	0.041786
CIF	3×3	0.0304128	0.0304128	0.069823
CIF	5×5	0.0790732	0.0790732	0.167143

표 2. i960RM SDK로 측정한 항목별 처리 시간(sec)

Image Size	SE Size	Erosion	Dilation	Edge Detection
QCIF	3×3	0.280880	0.263760	0.620080
QCIF	5×5	0.928400	0.912240	1.916080
CIF	3×3	1.136800	1.067760	2.506240
CIF	5×5	3.807440	3.740800	7.849920

V. System Applications

To verify the performance of the proposed parallel processing system, specific three applications were investigated. They were facial recognition, Phong shading, and automatic segmentation. To achieve the parallel version of the applications, we tried to transform serial modules composing the application into parallel ones in step by step. Some modules in an application were transformed with easy and they operated in the same functionality as original modules. But, any modules did not seem to be transformed because any identical memory access type did not detected through all of its portions. In other words, it was possible to transform the modules for the parallel processing system, but the processing speed of transformed ones was insignificant compared to the originals. In this case, new modules should be considered by using different methods.

1. Facial Recognition

The algorithm for facial recognition that we developed consists of a preprocessing and extracting feature region, normalizing feature regions, extracting 4 feature vectors, and a classification[14].

In the preprocessing step, hair on the face scanned was eliminated by histogram method. This method helps to distinguish between hair color and face color. In order to extract the region involved eyes, nose, and mouth whose we interested, Sobel edge operator was used to detect edges of facial components. And based on the length between outsides of eyes, the feature region is extracted finally. After equalizing and binarizing it, the first feature f_1 = h/w, where h is the height and w is the width of it, was yielded. In this processing step, we could not found out any parallel features except for equalization and binarization procedures that were peformed with block access way and horizontal access way with interval 1, respectively.

We normalized the final feature region to compare the same facial images with different sizes. Size normalization is a transformation of image of arbitrary size (x1×x2) into an image of fixed pre-specified size (for example, 48×48) which results in dimensional changes by factors of 48/x1 and 48/x2. Although this process was a crucial preprocess to obscure scale variation of facial images presented to a recognizer, we did not found out any parallel features for our system.

The extraction of feature vectors step was efficiently performed on our system. The normalized region yielded three features and itself became the fourth feature f_4. Each matrix element, which is the second feature f_2, was calculated for each 3×3 mesh of the normalized region. This is processed by horizontal access types with interval 3 and 12 in the MAMS. The third feature f_3 was calculated using 6×6 mesh by horizontal access types with interval 2 and 6.

Four feature vectors are sequentially compared to those of facial image stored in database using equation of which form is $d_i = |f_i-f_i'|$, $1\le i\le4$, where f_i' is the ith feature of some facial image in database. The order of comparison is f_1,f_3,f_2, and f_4 with each limitation value l_i. If d_i is came within l_i percents of total compared images, it needs a horizontal access type with interval 1.

On the result of simulation, we compared serial processing with parallel one. The results showed that the parallel processing was 61.9 times faster than the serial processing, where the number of PEs was 144.

2. Phong shading

Most computer image generation(CIG) system represent curved surfaces as a mesh of planar polygons because polygons can be transformed and rendered quickly with well-known algorithm. Although algorithms for realistic shading of polygons are well known, real-time CIG system did not use them because of the large amount of computation required per pixel. Phong shading, which is one of shading methods, eliminates "flattening out", dull surfaces and reduces Mark bands. However it has not been used in any real-time system because of the large amount of computation required for its usual formulation. Phong's method determines the intensity of each point using an approximate surface normal that is linearly interpolated from the true surface normal specified at the vertices. $N(x,y)=Ax+By+C$, where A, B, and C are chosen to interpolate the normal across the polygon. Interpolation for successive values of x and y and evaluating the illumination model requires 7 additions, 6 multiplications, 1 division, and 1 square root per pixel[15].

In this application, a parallel algorithm for Phong shading was applied to the parallel processing system in order to achieve the 3-D graphic acceleration. A serial Phong shading algorithm was modified for the system and the main key was to divide the polygon into 4×4 squares. And, for processing a square, a PE group logically comprised of 16 PEs performs a square. Since MAMC can support block access type with interval 1, it is possible to do so. In consequence, 16 squares are performed simultaneously.

With the same simulated result as that of the serial algorithm, the speed enhancement by the parallel algorithm to the serial one was 5.68, where the number of PEs was 16. Note that the system modification for this application was more complex than one for other applications because PE had to consist of 2 ALUs fundamentally and 1024×764 pixels of CRT should be mapped to memory modules to be used as the frame buffer.

3. Automatic Segmentation

Mpeg4 video coding standard enables content-base functionalities. In order to support the philosophy of the MPEG-4 visual standard, each frame of video sequences should be represented in terms of video object planes(VOPs). In other words, video objects to be encoded in still pictures or video sequences should prepared before the encoding process starts. Therefore, it requires a prior decomposition of sequences into VOPs so that each VOP represents a moving object.

Unfortunately, the image segmentation is recognized as an ill-posed problem and still remains unsolved. In addition, MPEG-4 assumes that video object plans in video are available prior to encoding process. Therefore, the real-time generation of the VOPs is a very attractive problem and the accurate segmentation of high quality is required for high-level image analysis. As in object recognition, image understanding, and scene interpretation and is necessary for authoring multimedia content, etc. So, a parallel processing

system is needed for the automatic segmentation to be processed in real-time.

We investigated the segmentation methods of moving object in video, using the one developed by ETRI[16], whose performance is comparatively accurate. The method utilizes spatio-temporal information. Although the method showed reasonable performance, one of the most serious drawbacks is the time-consuming processing. We converted processing modules in the method to be applied to the system. The modules were compatible to feature of memory access types. In the transformation procedure, some modules in the application were transformed with easy and transformed modules operated in the same functionality as originals. But, any modules did not seem to be transformed because any identical memory access type did not be detected though all of its portions. In this case, new modules were considered by using different methods. In spite of those procedures to make modules for the system, some of modules were serially processed on the host because of their lower degree of parallelism.

In the simplification step, morphological filters were used for the purpose of simplifying the image to make easier the image segmentation in ETRI's method. But, the module needed longer processing time than other methods because the filters are recognized as a time-consuming work. So, this module should be processed on the proposed system in order to improve the processing speed of this application (see Section IV). A difference image was simply obtained by block access way. In the scene change detection step, a scene changed region was detected by a mathematical statistics, that is, a less than a constant value that resulted from a lot of experiments. A watershed detection to find object boundary consists of flooding section and new minima detection[17]. Flooding section using the block access way was processed in the system. But new minima detection was processed on the host. Finally, foreground was simply detected from the previous step. The processing speed of the segmentation on the system was 1.5 times faster than one of the original method, where the number of PEs was 4.

Ⅵ. Conclusions

The demands for processing multimedia in real-time using unified and scalable architecture are ever increasing with the proliferation of multimedia applications. We presented a parallel processing system to achieve the demands for speedups and applied to three specific applications, which were recognized time-consuming works. It was confirmed that the system had been able to support enough scalability to the applications and improved their processing speed in real-time processing. If large number of PE is equipped with the system, it can be speculated that the speedup performance be pushed up to the limitation predicted by Amdahl's Law.

Although the comparison values previously mentioned were obtained through simulations and speedup performances during each application on the system were achieved, they were just estimated values because the proposed system has not yet been manufactured into a circuit board.

Unfortunately, some problems occurred in transferring a lot of data elements from the host to the system and vise versa during processing the application on the system. That is, the time for transferring data allocated more than the processing time. To solve this, the bus bandwidth needs to be improved on the system side and new specific methods to the system be developed on the method side.

References

[1] Sethuraman Panchanathan, "Architecture Approaches for Multimedia Processing," *4th ACPC'99 Salzburg*, Austria, Feb. 1999.

[2] Edwige Pissaloux, "On Parallel Reconfigurable Architecture for Image Processing," *4th ACPC'99 Salzburg*, Autstia, Feb. 1999

[3] M. Duff, "Parallel Processors for Digital Image Processing," *Advances in Digital Image*

Proce- ssing, pp. 265-279, 1979

[4] K. E. Batcher, "Design a massively parallel processor," *IEEE Trans. Comput.* Vol. C-29, pp. 836-840, Sep. 1980

[5] S. Wilson, "The Pixie-5000 -A Systolic Array Processor," *in Proc, IEEE Comput. Soc. Workshop, APAIDM*, pp.477-483, Nov, 1985

[6] M. J. Kimmel, et al., "MITE:Morphic Image Transform Engine, an architecture for reconfigurable pipelines of neighborhood processors," *in Proc, IEEE Comput. Soc. Workshop, APAIDM*, pp. 493-500, Nov. 1985

[7] E. W. Kent, et al., "Hierarchical Cell Logic and the PIPE processor: structural and functional correspondence," *in Proc, IEEE Comput. Soc. Workshop,APAIDM*, pp.311-319,Nov. 1985

[8] R. M. Laugheed, et al., "Cytocomputer: Architecture for parallel image processing," *in Proc, of the workshop on Picture Data Description Management*, pp. 281-286, Aug. 1980

[9] J. W. Park, "An Efficient Memory System for Image Processing," *IEEE trans. Comput.* Vol. C-35, No. 7, pp. 33-39, 1986

[10] J.W.Park and D.T.Harper III, "An Efficient Memory System for SIMD Construction of a Gaussian Pyramid," *IEEE Trans. on Parallel and Dist.* Vol. 7, No. 7, July 1996

[11] Philippe Salembier, "Morphological multiscale segmentation for image coding," *Signal Processing* Vol. 38, pp. 359-386, 1994

[12] Lic Vincent, "Morphological Grayscale Reconstruction in image analysis: applications and efficient algorithm," *IEEE Trans. on Image Processing*, Vol.2, No.2, pp.176-201, Apr. 1993

[13] "IQ80960RM/RN Evaluation Platform Board Manual," Feb. 1999

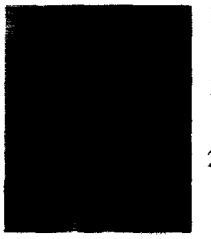
[14] H. Lee, K. A. Moon, J.W.Park, "Design of parallel processing system for facila image retrieval", *4th ACPC'99 Salzburg*, Feb. 1999

[15] http://www.pt.hk-r.se/~pt93mm/thesis/technique-s/fast_phong/siggraph86.html

[16] Munchurl Kim, et al., "A VOP Generation Tool: Automatic Segmentation of Moving Objects in Image Sequences Based on Spatio-Temporal Information," *IEEE Trans. on CSVT*, Vol.9, No.8, pp.1216-1226, Nov. 1999

[17] L. Vincent and P. Soille, "Watershed in digital spaces: An Efficient algorithm based on immersion simulations," *IEEE Trans. on PAMI*, Vol.13, No.6, pp.583-598, Jun. 1991

이 형(Hyung Lee)



1995년 2월 : 충남대학교 컴퓨터과학과 졸업
1997년 2월 : 충남대학교 컴퓨터공학과 석사
2000년 2월 : 충남대학교 컴퓨터공학과 박사수료

2001년~현재 : 대전보건대학 방송제작기술과 전임강사

<주관심 분야> 병렬 처리, 비디오 처리

박 종 원(Jong-won Park)

1979년 2월 : 충남대학교 전자공학과 졸업
1981년 2월 : KAIST 석사
1991년 8월 : KAIST 박사
1995년~현재 : 충남대학교 정보통신공학과 교수

<주관심 분야> 영상처리, 병렬처리, 의공학