

Smart Camera System Design

Tiehan Lv, Burak Ozer, Wayne Wolf
Department of Electrical Engineering,
Princeton University, Princeton
NJ 08544, USA
e-mail: {lv,iozer,wolf}@ee.princeton.edu

Abstract

This paper describes a smart camera system developed at Princeton University. The system is used as a real time visual surveillance system that can detect the presence of a person and recognize his/her activities in an indoor environment. The paper outlines the structure of the current prototype system that uses multiple cameras, each with its own video signal processor (VSP), and which is able to recognize several different gestures in a fixed background at a speed of 20 frames/second. Furthermore, this paper discusses possible structures and tradeoffs of the system.

1 Introduction

This paper describes a smart camera project conducted at Princeton University. The purpose of this project is to develop a real time visual surveillance system which is able to detect the presence of human beings in an indoor environment and recognize their activities. A prototype system has been built to detect a single person and recognize his/her gestures in a fixed environment. The system is able to process about 20 frames per second. Our current system has two smart camera nodes hosted by a PC. Skin color extraction, background elimination, region extraction, fitting ellipses to the body parts, and graph matching are used to identify the presence of a person. An abstract model is extracted to describe the body parts where the parameters of the model are then fed into parallel HMM algorithm blocks to track and classify the movement of the parts. Finally, a high-level classifier combined with the HMM algorithm determines specific gestures of the person.

Background elimination simplifies the procedure to find the foreground objects, but this approach requires a-priori knowledge about the background information. L. Davis et al. give an elaborated discussion of using background elimination to identify the foreground objects [4]. A fuzzy pattern matching method for face detection is proposed by H. Wu et al. [8], where Farnsworth color system is employed for skin color recognition. F. Solina and R. Bajcsy use superquadrics for shape representation [9]. This representation is used in our smart camera system to describe the human body parts of the person in the scene. Several different approaches have been developed for human activity recognition such as Hidden Markov Model, Dynamic Time Warping, and Dynamic Bayesian Networks [10],[11],[12],[13]. In our algorithm, we use HMM model combined with Mahalanobis distance classifier to determine the gestures.

Hardware design is important for the development of a real time system. J. A. Watlington and V. M. Bove propose a dataflow model for parallel media processing [1]. L. Davis et al develop a multi-perspective video system in University of Maryland [4]. Jason Fritts et al evaluate the characteristics of multimedia applications for media processors [7]. Multiple cameras are deployed in a FlyCam system to capture panoramic video with the cooperation of the cameras [2]. Real time human tracking systems have also been proposed in literature. A real time person tracking system, Pfinder, developed at MIT media lab [14] uses Maximum A Posteriori Probability (MAP) approach to detect and track people by using 2D models. The Pfinder system works in an arbitrarily complex but single person, fixed camera environment. W4 is another real

time human tracking system developed at University of Maryland. It has the ability to track multiple persons in the scene. It also assumes fixed camera environment. The background information should be collected before the system can track foreground objects. A more detail introduction of work concerning human tracking is given by A. Pentland [3].

In our work, we select and combine several tasks in different image and video processing domain to develop a real time video surveillance system. This system has not only the abilities to detect and tracking people, but also the ability to recognize their activities in real time.

The rest of the paper is organized as follows. Section 2 describes the current smart camera system. Section 3 discusses the structure of a more advanced system being developed. Conclusions are given in Section 4.

2 Design of the Prototype System

The smart cameras are used in a fixed environment where the background and illumination do not change over time and where the cameras are also fixed. While this reduces the complexity of the processing algorithm, it brings in the requirement of using multiple camera nodes to overcome the object occlusion and enlarge the sensing field of the whole system. As a real time system, the processing ability is critical. Considering this, each camera node has its own processing units. The camera nodes can communicate with each other to identify the node having the best view angle. Figure 1 shows the architecture of the smart camera system.

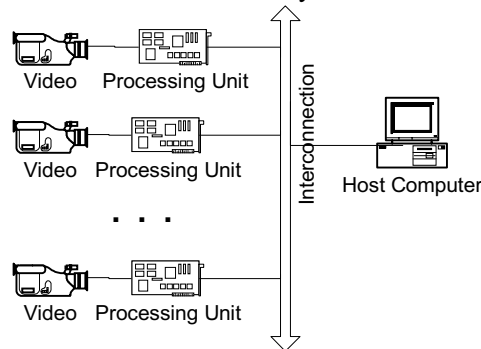


Figure 1: Structure of the whole system

In our prototype system, two cameras, oriented at 90 degrees to each other, are used. The camera nodes, hosted by a PC, can communicate with each other via shared memory.

2.1 Hardware configuration for the prototype system

In our prototype system, a single smart camera node is composed of a standard camera and a video processing board. We use one TriMedia Media processing board [18] in each camera node. A TriMedia board has one TM1300 TriMedia processor, which is specialized for media processing that allows Windows and Macintosh platforms to take advantage of the TriMedia Processor via PCI interface. Multiple TriMedia processing boards can be installed to one host PC to form a more powerful multi-processor system. A 32-bit TM1300 processor has its own dedicated memory and a five issue VLIW (Very Long Instruction Word) CPU together with several coprocessors as shown in Figure 2. The video input and video output units provide a convenient interface for video I/O stream. After initialization, the Video In unit automatically stores the input image into an assigned buffer. When the loading of one frame is finished, an interrupt is arisen to notify the CPU. The CPU can then copy the data in the buffer and supply empty buffers for new frames. Video Out unit puts the frames in the supplied buffers to an output video stream when a frame is processed, the CPU is notified by an interrupt. The frame based

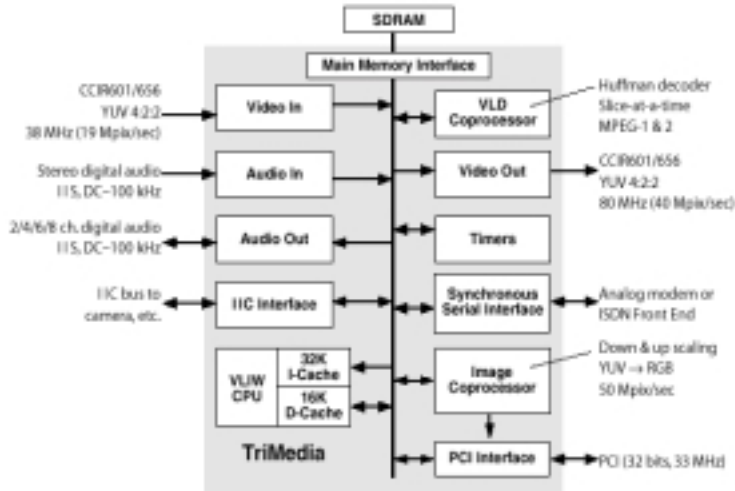


Figure 2: Structure of a TriMedia processor

	Units	#Units	Latency(Cycles)
Functional Unit	Constant	5	1
	Integer ALU	5	1
	Load/Store	2	3
	DSP ALU	2	2
	DSPMUL	2	3
	Shifter	2	1
	Branch	3	3
	Int/Float MUL	2	3
	Float ALU	2	3
	Float Compare	1	1
	Float sqrt/div	1	17
	#Register		
Instruction cache			32KB, 8 way
Data cache			16KB, 8 way
#Operation slots/instruction			5

Table 1: Features of TriMedia processors

interrupt scheme reduces the processing load of the CPU.

The CPU in the processor has multiple functional units and 128 registers. Table 1 displays the major features of a TriMedia CPU. In our system, TM1300 processors run at 100MHz, providing a peak performance of 500 MOPS.

2.2 Algorithm and software architecture

Algorithm used in the system is a major factor of system architecture. In this section, we give a brief introduction to the algorithms used in the system. However, since the main focus of this paper is the design of the system, we do not give in-depth description. More details of the algorithm is presented in [5], [6], and [21]. The algorithm consists of two parts, low and high-level processing. The low-level part performs human detection and extracts parameters for the

abstract graph representation of the image being processed. The high level part uses HMM algorithm to determine the movements of the person's body parts, and uses a distance classifier to detect specific gestures. During these processes, each smart camera node communicates with other nodes, so that the node with the frontal view of the person, yields the final results.

The algorithm blocks of the low-level parts are shown in Figure 3. Figure 4 displays a sample frame processed by the algorithm.

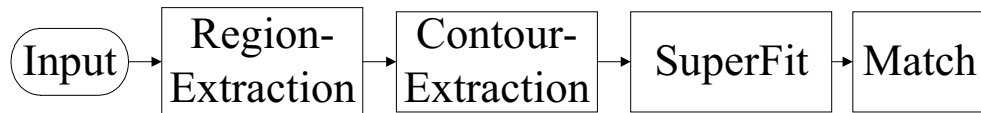


Figure 3: Algorithm blocks for low level processing

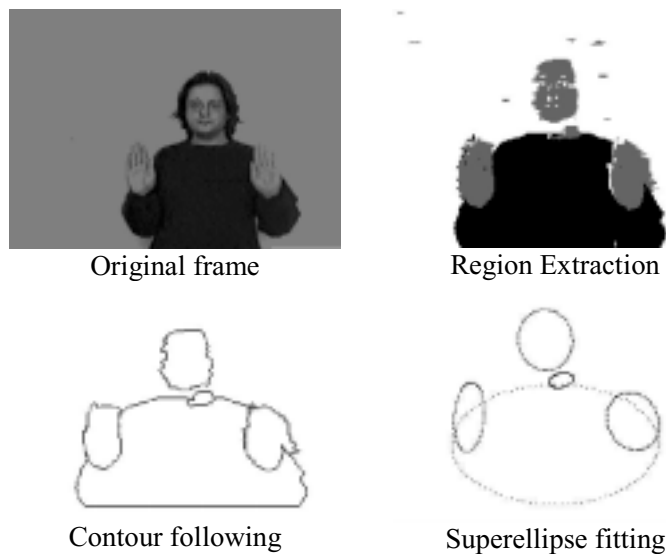


Figure 4: A frame processed by target program

2.2.1 Low-level processing

Region extraction

Region extraction block performs background elimination and skin area detection simultaneously. The output of this block is a frame size buffer and is fed into the next algorithm block.

The background elimination takes 384x240 images as input and extracts foreground objects from the input images. At the same time, skin color detection is performed. Since smart cameras are supposed to operate in a relatively fixed environment, we require that background is known a priori and does not change over time. Hence, the pixel level process simply subtracts background from foreground image. Skin area detection identifies skin regions in the input image by comparing color values of each pixel to a human skin color model. Considering the processing time limit of a real application, we use YUV color model instead of more complex model that employs Farnsworth nonlinear transformation in the algorithm.

Contour extraction:

Contour following uses a 3x3 filter to extract the boundary of each object region obtained by previous block.

Ellipse fitting

In this step, the algorithm finds the ellipse parameters which can optimally describe the boundaries extracted.

Graph matching

The binary features between two regions and the unary features of each region are extracted. These features are fed into the graph matching algorithm to determine the body parts of the person in presence [5].

2.2.2 High-level processing

The high-level algorithm part consists of two subparts. In the first part, the position of each body part, identified by the low-level algorithm, is tracked. HMM algorithm is applied to identify the movement of those parts. In the second part, a distance classifier is employed to combine these movements together to detect the specific gesture made by the person in the scene. While the low-level processing is executed on frame basis, high-level processing takes several consecutive frames to identify the movements.

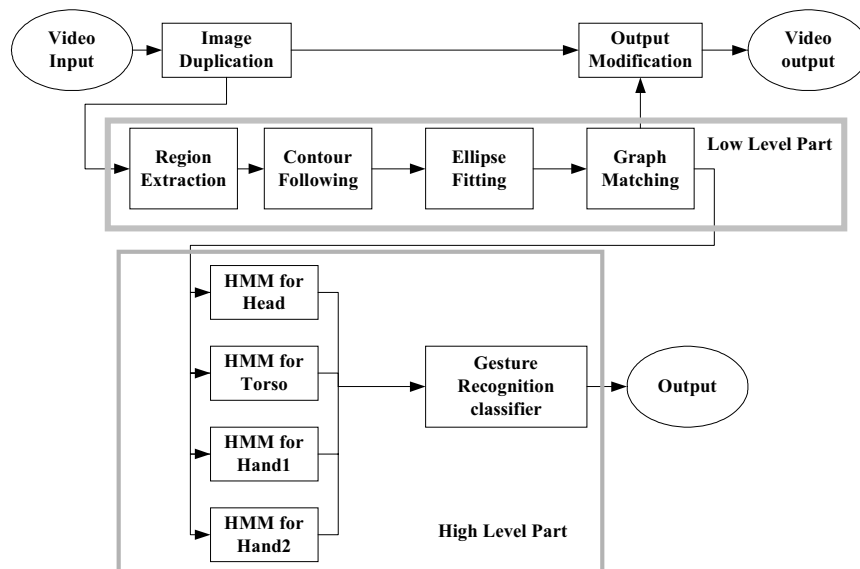


Figure 5: Software architecture

2.2.3 Software Architecture

Software design is a critical part for a real time system. For the prototype system, there are several requirements in addition to the algorithm development.

1. Since the input frame rate is 30 frames/second and the processing capability is varying around 20 frames/sec, there is a need for synchronization.
2. For the demo purpose and for debugging, visualization of the results is needed.
3. The algorithm is still in the developing phase, a flexible software architecture should be used.
4. When the processing capability is concerned, Using of an operation system in a single camera node should be avoided.

Figure 5 shows the software architecture of a smart camera node. In this architecture, we use pipe-filter structure as a basis. By using the interrupt processing ability of the TriMedia processor, we solve the synchronization problem by using the following procedure. A flag variable and a frame buffer are used for video input synchronization. The main processing procedure only accesses the frame buffer when the flag is set to FULL. After the data in the buffer is copied to a safe place, the main processing part sets the flag to EMPTY. The input interrupt-processing procedure only copy data to the frame buffer when the flag is set to EMPTY. If the flag is FULL, the procedure simply discards the current frame. The output synchronization uses a similar scheme. By using this scheme, operation system with multiple thread support is avoided.

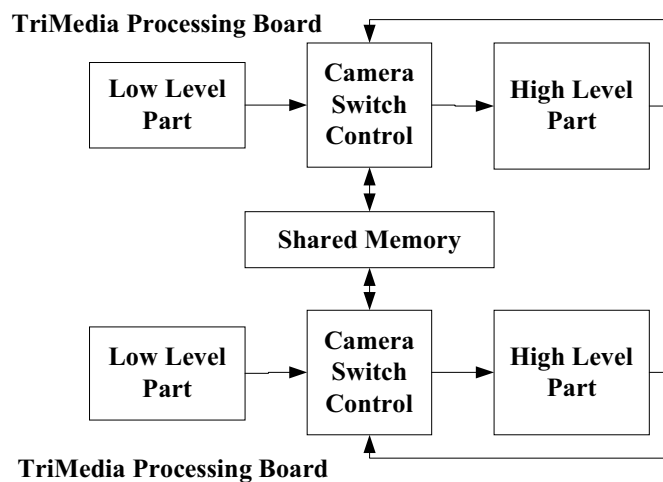


Figure 6: Interconnection between cameras

2.3 Multiple camera coordination

In the prototype system, a simple scheme for multi-camera coordination is implemented (Figure 6). The priority camera switching scheme has two difference levels. The low-level camera switching factor is determined by the comparison between the compactness of the head in two cameras. The high-level switch is triggered by two pre-defined gestures. The high level criteria has priority to low level switch criteria. After the action of camera switch triggered by gestures, the focus is fixed in the active camera for 100 frames before any low-level switch can occur. For each camera node, variables corresponding to the compactness of the head region and the detection of camera switching gesture are calculated and sent into shared memory. Variables from another camera are compared to the local variables to determine if the current camera should be active. In the prototype, only several words are exchanged between two camera nodes for each frame. However, more sophisticated scheme can be adopted for new requirements.

An experiment measuring the communication ability between two camera nodes is conducted. In the experiment, two TriMedia processing boards are configured to exchange frames. It takes 25 seconds to send 10,000 frames from one TriMedia board to another. Each frame contains 92 Kbytes. So the throughput of the shared memory system is 37 Mbytes/second.

Synchronization of the cameras is required in some applications. In the prototype system, the two cameras are not synchronized to each other. But the current system has ability to synchronize the two cameras within one frame, or one thirtieth second.

3 Exploration for the Future System

The goal of the project is to build a smart camera system. We plan to implement the processing unit of a single camera into a single chip. With the advance of the VLIW technology, it is even possible to build the image sensor into the same chip. Therefore, we also conducted experiments to collect information for building a single chip smart camera node.

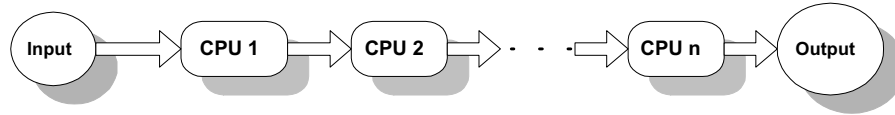


Figure 7: Macro-pipeline architecture

The processing unit can be implemented by using special purpose circuits or by using general CPU-Memory architecture. CPU structure has advantage in its flexibility for the refinement of the system. While in the prototype system, only one processor is deployed in a smart camera node, we also consider using several relative simple processors in a smart camera node. Figure 7 shows a Macro-pipeline architecture model. The pipeline structure is mapped from the pipeline-filter structure of the software. Different algorithm stages can be assigned to different CPUs. A benefit for this approach is that the CPUs can have different capacity. For example, if a CPU is only required to process *region extraction* or *contour following*, the CPU does not need to have floating point unit.

Workload characterization experiments are conducted to determine proper configuration for these CPUs. SimpleScalar tools set is used to conduct the experiment. SimpleScalar tool set is a set of simulators for superscalar microprocessors. Information such as instruction statistics, cache behavior, branch behavior etc can be collected and reported. A detailed instruction for this tool can be find in [19].

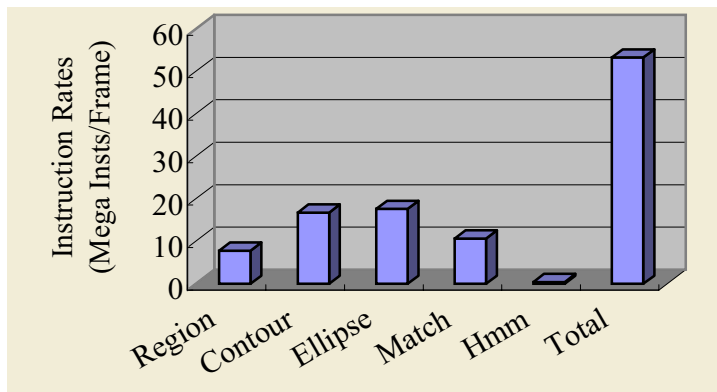


Figure 8: Instruction rates

The first one is the dynamic instruction count of each algorithm block. This is the basis for algorithm block allocation and for determining the object speed of the CPUs. Figure 8 shows the results. The results are counted on frame basis. So the processing capability required by difference blocks are 7.8:16.8:10.7:0.4 or 15:31:33:20:1 approximately, for region:contour:ellipse:match:HMM algorithm blocks, respectively.

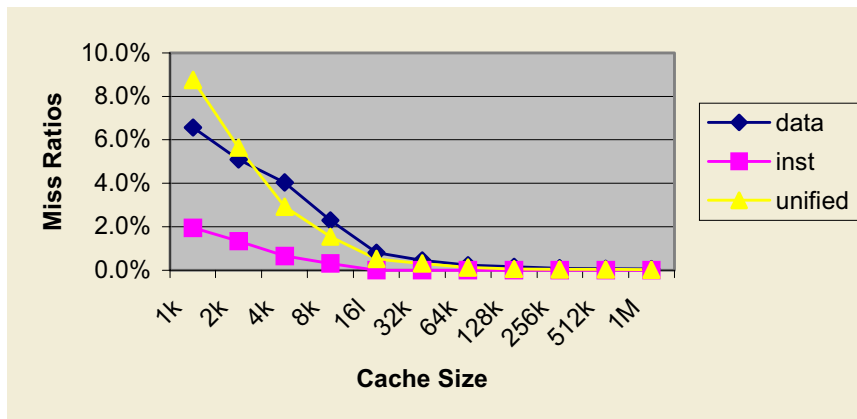


Figure 9: Cache size characteristic of low level algorithm part
(Direct mapped cache, line size 64 bytes)

Another important aspect is the memory system. The cache behavior is also analyzed. Figure 9 shows the effect of different cache sizes for low level algorithm part. The size of a cache is determined by the working set of the program. We define the working set size as the cache size where the miss ratio decreases dramatically with respect to smaller cache size. When such a point does not exist, the smallest cache size resulting the miss ratio below 3% is used. Three classes of cache, data cache, instruction cache and unified cache, are tested. Although our program processes large amount of data, it does not require large data cache. According to the figure, only 8KB cache is required to reduce the data cache miss ratio below 3%. For an instruction cache, a size of 1KB is enough to reduce the miss ratio below 2%. This corresponds to the relatively small size of the program code. Since the instruction cache is small, a unified cache is quite fit to this application. Only 4KB cache can reduce the miss ratio down to 3%. Figure 10 shows the result

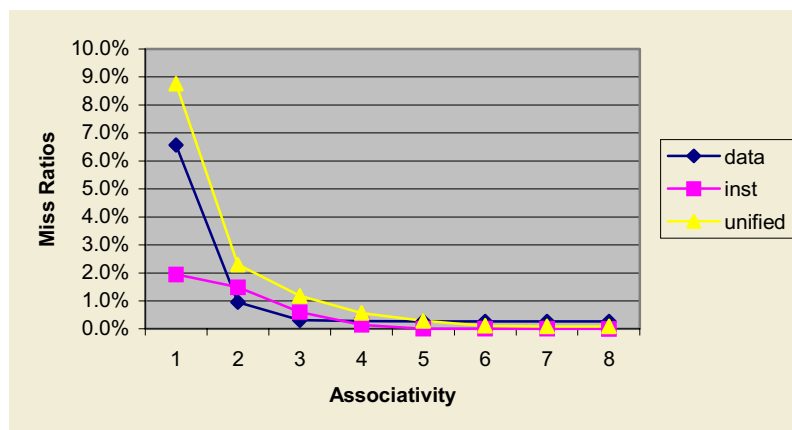


Figure 10: Associativity behavior of low level algorithm part
(Cache size 1kB, line size: 64 bytes)

related to cache associativity. It is clear from the figure that associativity plays an important role in cache behavior. A two-way 1KB data cache outperforms a direct-mapped 8KB data cache. An associativity of 2 or 4 is suitable for the algorithm. In Figure 11, the experimental result concerning cache line size is shown. Although video applications usually process a larger amount of data, this algorithm does not demand larger line size. A line size at 64 bytes is recommended for the algorithm. Figure 12, Figure 13, and Figure 14 display the cache behavior of the high level part. The high level algorithm part has a different data cache behavior from the low level

part. It requires larger data cache size, larger associativity and smaller line size. A possible reason for this is that high level part has a different coding style. The algorithm implementation uses multi-dimensional arrays and uses extensive matrix and vector operations. Since the code size of the high level algorithm part is still kept small, its instruction cache behavior is similar to that of low level algorithm part. Since high level part is not the critical part in terms of running time, the choice of cache parameters largely depends on the cache behavior of low level algorithm. A 1KB four way associative unified cache with 64 bytes cache lines is recommended. We test the effect of such a cache for each algorithm block. The results are shown in Figure 15.

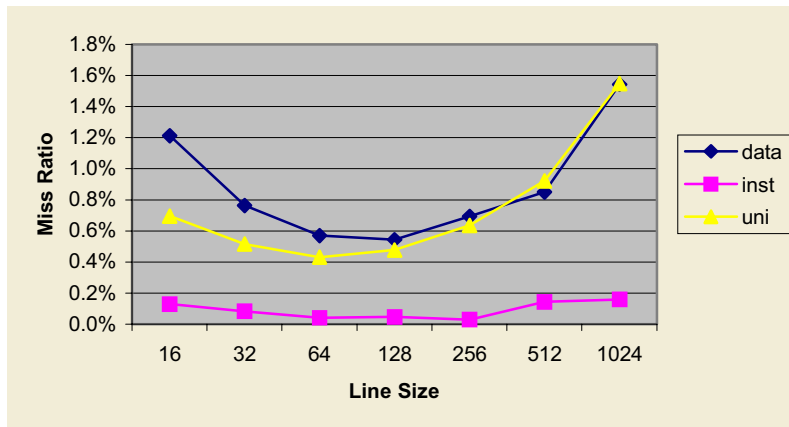


Figure 11: Impact from cache line size for low level algorithm parts
(Cache size: 4kB, associativity:2)

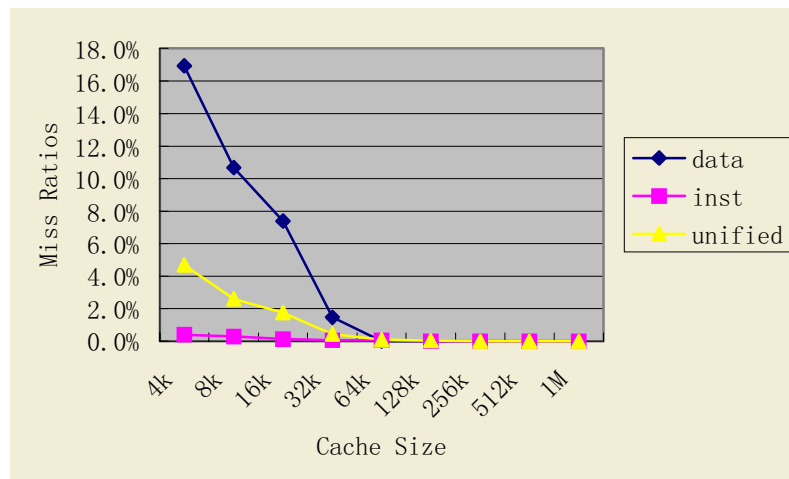


Figure 12: Cache size for high level processing
(Directly mapped cache, line size 64 bytes)

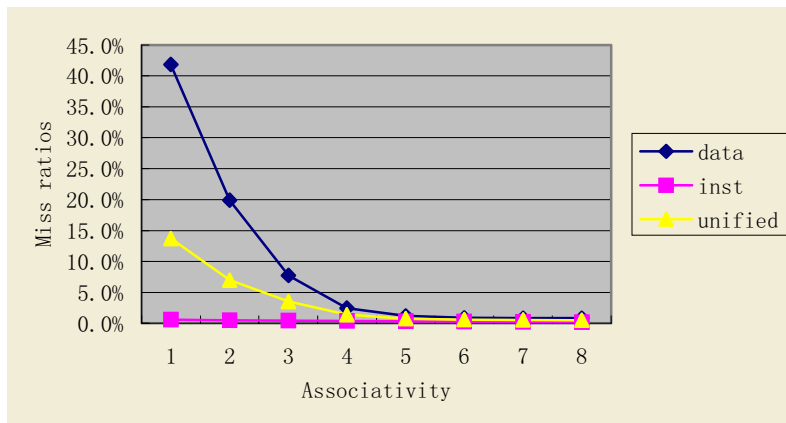


Figure 13: Associativity for high level processing
(1 KB cache, 64 bytes line)

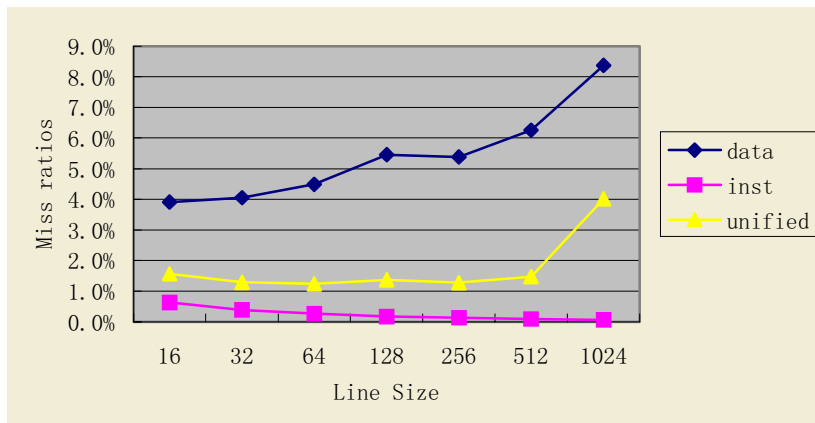


Figure 14: Line size for high level processing

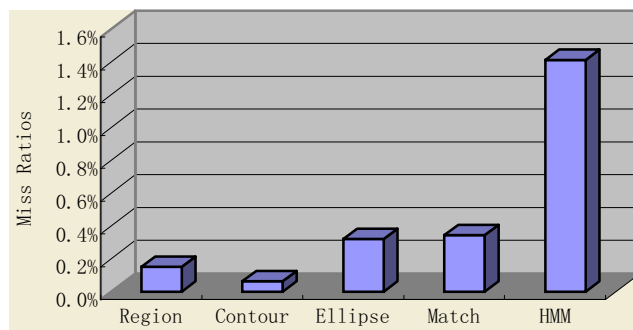


Figure 15: Cache performance for different algorithm blocks

	Execution Time (kcycles)	
	Before optimization	After Optimization
Region Extraction	2737	1168
Contour Following	3145	2070
Whole low level part	7470	4356

Table 2: Effect of special instructions

The impact of media instruction is also tested. A special instruction INONZERO, which performs conditional assignment, is used. Table 2 displays the processing time before and after using this instruction. The results indicate that the instruction is effective to reduce the processing time in low-level pixel processing blocks, namely region extraction and contour following algorithm blocks. Therefore, a similar instruction would be included in the CPU for low level processing in the new system.

4 Conclusions and Future Work

In this paper, we describe a prototype smart camera system developed at Princeton University. Using relatively modest hardware, the system is able to recognize simple gestures of a person in real time. As a step to develop a new hardware system, we also examine the configuration parameters for the new system. Our future work would include developing a new system with improved performance that can recognize more complicated gestures.

References

- [1] J.A. Watlington and V.M. Bove, Jr., "A System for Parallel Media Processing," *Parallel Computing*, Dec 1997
- [2] J. Foote and D. Kimber, "FlyCam: practical panoramic video and automatic camera control", *Proceedings, 2000 International Conference on Multimedia and Expo, IEEE, 2000*
- [3] A. Pentland, "Looking at People: Sensing for Ubiquitous and Wearable Computing", *IEEE PAMI*, Vol 22, No 1, pp. 107-119, Jan. 2000
- [4] L.S. Davis, E. Borovikov, R. Cutler, and T. Horprasert, "Multi-perspective Analysis of Human Action", *Third International Workshop on Cooperative Distributed Vision, 1999*
- [5] B. Ozer, W. Wolf, A.N. Akansu, "Relational Graph Matching for Human Detection and Posture Recognition", *SPIE, Photonic East 2000, Internet Multimedia Management Systems, Boston, MA, November 2000*
- [6] B. Ozer, and W. Wolf, "Video Analysis for Smart Rooms", *SPIE, ITCOM 2001, Denver, CO, Aug 2001*
- [7] J. Fritts, W. Wolf, and B. Liu, "Understanding multimedia application characteristics for designing programmable media processors", *SPIE Photonics West, Media Processors '99, San Jose, CA, pages 2-13, Jan 1999*
- [8] H. Wu, Q. Chen, and M. Yachida, "Face Detection From Color Images Using a Fuzzy Pattern Matching Method", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 21, No. 6, Jun 1999
- [9] F. Solina, and R. Bajcsy, "Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 2, Feb 1990

- [10] T. Darrel, I. Essa, and A. Pentland, "Task-specific Gesture Analysis in Real-Time Using Interpolated Views", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 18, No.12, pp. 1236-1242, Dec 1996
- [11] J. Yamamoto, J. Ohya, and K. Ishii, "Recognizing Human Actions in Time-Sequential Images Using Hidden Markov Models", Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 379-385, 1992
- [12] T. Starner, J. Weaver, and A. Pentland, "Real-Time American Sign Language Recognition using Desk and Wearable Computer Based Video", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 20, No. 12, pp. 1,371-1,375, Dec 1995
- [13] V. Paviovic, B. Frey, and T. Huang, "Classification Using Mixed State Dynamic Bayesian Networks", Proc IEEE Computer Vision and Pattern Recognition, Vol. 2, pp. 609-615, 1999
- [14] C. Wren, A. Azarbayejani, T. Darrell, A. Pentland, "Pfinder: Real-Time Tracking of the Human Body", Proc. of the SPIE Conference on Integration Issues in Large Commercial Media Delivery Systems, Oct 1995
- [15] I. Haritaoglu, D. Harwood, and L.S. Davis, "A Real Time System for Detecting and Tracking People", 3rd International Conference on Face and Gesture Recognition, Nara, Japan, Apr, 1998
- [16] M. Shaw, D. Garlan, "Software Architecture", Prentice Hall, 1996
- [17] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, "Numerical Recipes in C", Cambridge University Press, Second Edition, 1995
- [18] TriMedia documents, <http://www.trimedia.com>
- [19] D. Burger, and T.M. Austin, "The SimpleScalar Tool Set, Version 2.0", Technical Report 1342, University of Wisconsin Madison, CS Department, Jun 1997
- [20] T. Lv, B. Ozer, and W. Wolf, "Workload Characterization for Smart Cameras", Third Workshop on Media and Streaming Processors, Austin, Texas, Dec 2001
- [21] B. Ozer, T. Lv, and W. Wolf, "Real-Time Video Analysis for Smart Rooms", Submitted to IEEE Trans. Pattern Analysis and Machine Intelligence