Designing Video Processing Surveillance System

EE382 Embedded Software Systems

Final Report

Koichi Sato

Abstract

The objective of this project is to design a hybrid intelligent surveillance system. That is, the system consists of embedded systems and a PC-based system. I propose an embedded system that performs some of image processing tasks and sends the processed data to a PC. Under the supervision of Dr. Aggarwal, I have created a PC-based intelligent surveillance system that tracks persons and recognizes two-person interactions using a grayscale monochrome side-view image sequence captured by a single static camera [7]. Based on my previous PC-based research, I explored a division of tasks between an embedded machine and a PC, simulated the embedded machine on Ptolemy, and designed an actual embedded machine using a 16bit CISC microprocessor. This embedded machine performs within three frame-cycle periods with a small code and a small internal RAM usage.

Introduction

Tracking and recognizing objects using video sequence is an important topic in computer vision field and has a variety of potential applications. Many researchers have proposed actual applications with this research problem. In [1], A. Pentland proposed a “wearable device” that sees people using an image sensor and understands the environment so that

computer can act or respond appropriately without detailed instructions. In [2], Mahonen proposed a wireless intelligent surveillance camera system that consists of a digital camera, a CPU or DSP for image processing, and a wireless radio modem.

Unlike a multimedia system, the output of a surveillance system is not always an image sequence, but could also be the positions of humans, image content features and so on. Therefore, the architecture of video surveillance system should differ greatly depending on what its objective and outputs are. Several other researchers [3-6] proposed varieties of embedded image processing systems. Even in systems combined with a PC, many of the applications embed the image processing part in their camera modules. Some of the reasons for this decision are: i) difficulty or high cost concerning transforming a huge amount of video data; ii) restriction with respect to the size or weight of the component; and iii) requirement for a real-time solution.

In [10], Shirai *at el*. proposed a real time surveillance system. They used a linear array processor consisting of DSP boards and I/O boards, which perform TV images in real-time. The I/O board digitizes an NTSC analog signal as an array of 8 bit unsigned integers and transfers them to the DSP boards. The I/O board also functions as the video signal output for display. The I/O board converts the stored image data processed in the DSP board into an analog video signal. Each DSP board has two DSP chips (TI TMS320C40) that is a 32 bit floating point processor, runs on 50MHz clock speed, and transfers one background memory per clock cycle. These DSP processors are connected in tandem on the DSP boards, forming a linear array, and are capable of performing independently. They used four board connected serially to compute the pixel velocity using the optical flow method. The optical flow method is adopted in order to compute the image motion using the spatial gradation in the image and the temporal intensity

difference of image sequence. Each four board performs spatial filtering, temporal filtering, velocity calculation and display processing respectively in this order and the last board outputs the velocities.

In previous research [7-8] Sato and Aggarwal have proposed a real-time surveillance system that recognizes human interactive activities using a side-view image sequence. The system recognizes humans based on the pixel velocities extracted using "temporal spatio-velocity transform". The system consists of a single monochrome video camera, a video capturing device and a PC. When it turns to an actual application, the system makes use of several cameras. This is due to the small observation range of the single system. This causes several problems: (i) several PCs are required because one PC can manage up to two cameras; and (ii) synchronization between small system sets is difficult.

To overcome problem (i), I propose an embedded system that performs the fundamental image processing part (that is human segmentation processing) and sends the resulting data to a PC. In this paper, I begin with reviewing the partitioning problem between the embedded system and the PC, which has been discussed in the previous paper. Then, I describe the algorithm details, followed by the simulation on Ptolemy, hardware design and a conclusion.

A division of tasks between the embedded system and the PC-based system

As was addressed in the previous report, we considered a best division of tasks between the embedded system and the PC-based system. Figure 1 shows the overall system structure. Table 1 is the advantages and disadvantages of an embedded system and a PC-based image system. The congeniality for the embedded system is evaluated in Table

2. Figure 2 shows the transfer rate and computation amount of each step. Based on these considerations, I assigned the process assignment with bold lined framed shown in Figure 1.

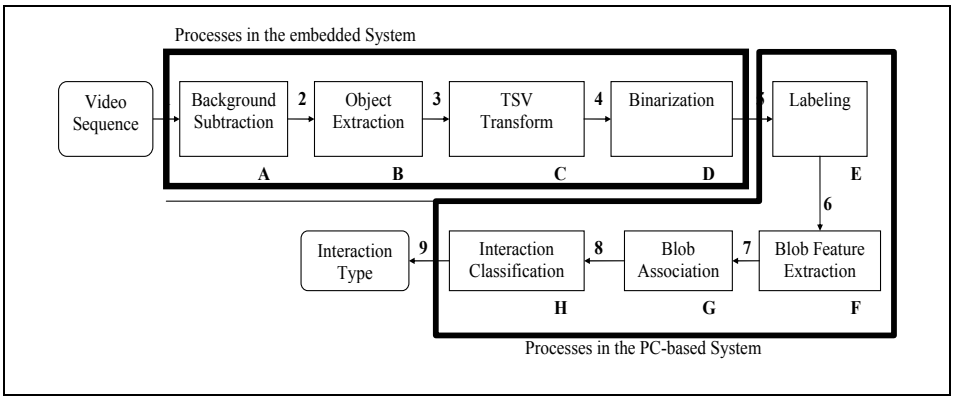

Figure 1 Overall System Structure and Process Assignment

	Embedded image processing	PC based image processing
Merit	- Cost effective - Effective for pipeline-type process	- Good for complex processing
Demerit	- Expensive for a process using large amount of data (like database search)	- Camera-PC transmission data is huge -> The number of connectable cameras to a PC is restricted

Table 1. Embedded image processing vs. PC-based image processing

Step	Description	Congeniality
A	Subtraction and comparison for entire image	◎
B	Addition for specific region and comparison one line	◎
C	Parallelogram shift and addition for entire TSV image	◎
D	Comparison for entire TSV image	◎
E	Labeling process for Binary TSV image	○
F	Calculation of blob features	○
G	Associating blob using features	○
H	Activity recognition	△

Table 2. Congeniality of each process to embedded system

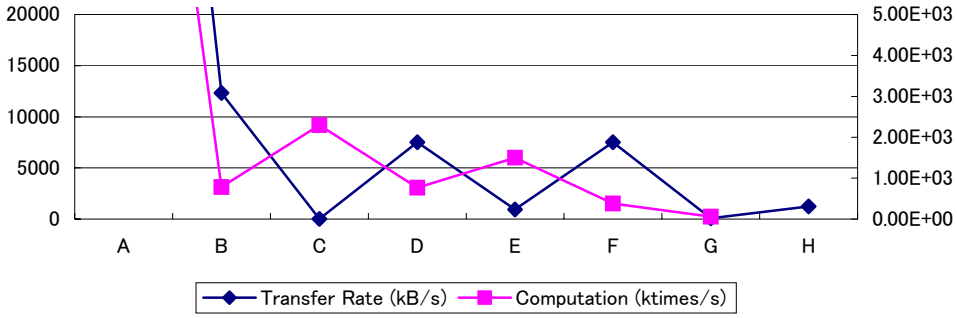

Figure 2 The transfer rate and computation amount of each step

Algorithm Details

The overall system is a hybrid intelligent surveillance system consisting of an embedded system and a PC system. Figure 3 is the block diagram of the embedded system. The embedded system performs cropping, background subtraction, vertical projection, TSV transform and binarization. The transformed data are transmitted to the PC system. In the following sections, I describe the detail of each process.

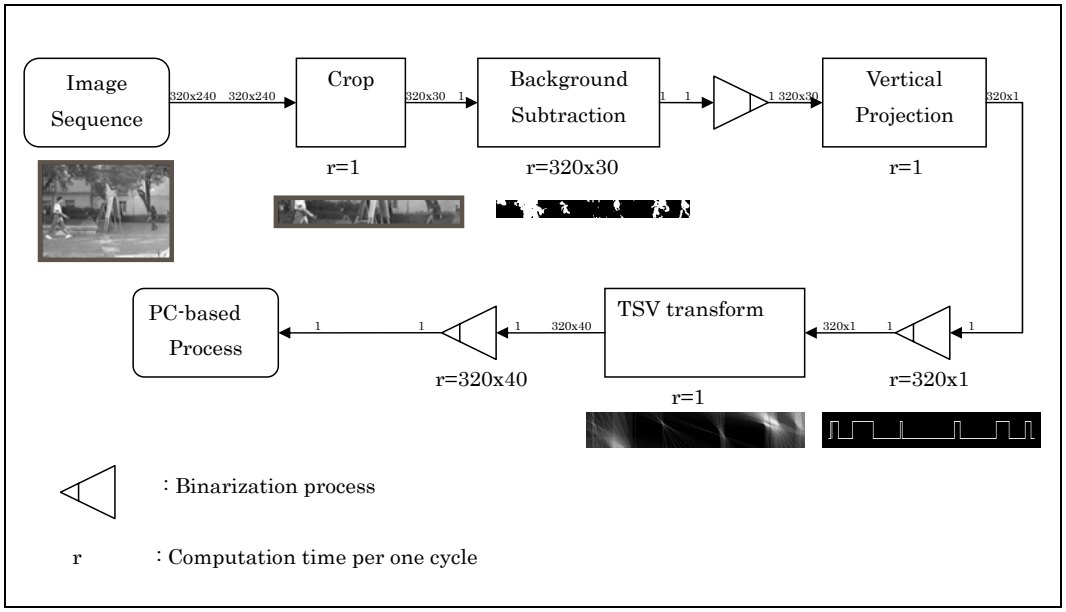

Figure 3 Block diagram of the embedded system

Cropping

In order to extract a standing object, I crop the image into a specific region that only the standing object blobs cross. The region is a torso level in a human image. In this operation, the image size is reduced to 320x40 pixel^2.

Background Subtraction

We use simple background subtraction and binarization to segment foreground region using threshold *Th*,

$$S(x,y) = \begin{cases} 1 & |I(x,y) - B(x,y)| > Th \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Object Extraction

The Object extracting process is performed by projecting blobs vertically over entire image (320x40 size) and re-binarizing the projection value by threshold T_H.

$$H(x) = \begin{cases} 1 & \sum_{y=a}^{b} S(x,y) > T_H \\ 0 & otherwise \end{cases} \quad (2)$$

where T_H is a threshold, which is constant in all situations and $H(x)$ is the object extracted binary image.

Temporal Spatio-velocity transform (TSV transform)

We proposed TSV transform in the previous paper [7,8]. TSV transform extracts the pixel velocity from binary image sequences. Here, we use one-dimensional binary image sequence coming from object extraction.

$$V_n(x,v) = e^{-\lambda} V_{n-1}(x-v,v) + (1-e^{-\lambda}) H_n(x) \quad (3)$$

To group the pixels with similar velocity, we binarized the temporal spatio-velocity image by a fixed threshold Th_v.

$$\widetilde{V}_n(x,v) = \begin{cases} 1 & \text{if } V_n(x,v) \geq Th_V \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Experiment

I designed and simulated the system on Ptolemy, as well as designing and simulating a hardware system using 16bit CISC microprocessor.

Ptolemy Simulation

The Figure 4 is the system design on Ptolemy. The diagram follows same steps as the block diagram discussed in Figure 3. The task of each icon is described in Table 3.

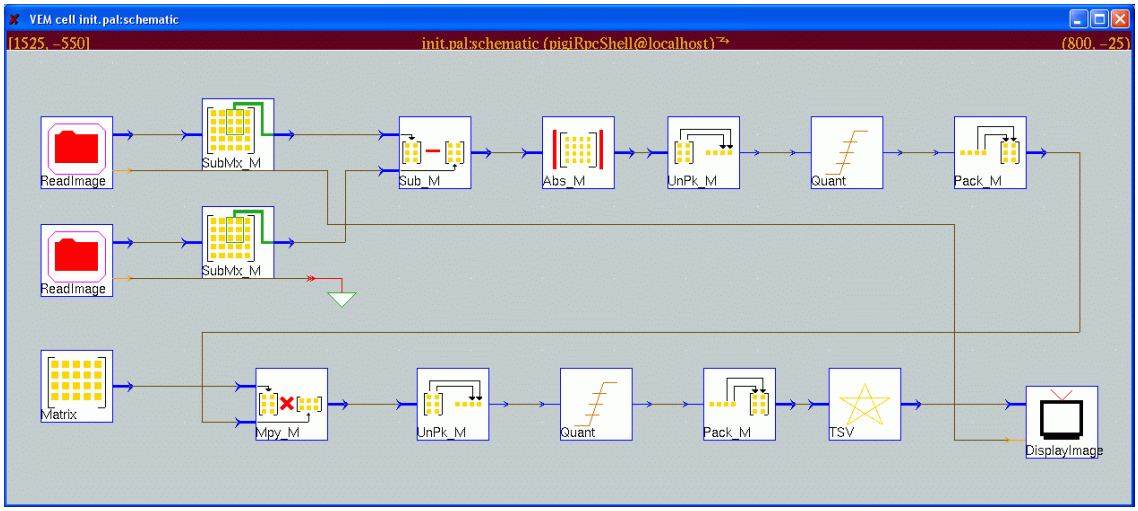

Figure 4 System design on Ptolemy

Icon	Explanation
Read Image	Reading image, the upper icon is for obtaining the sequence of images, the lower icon is for obtaining the background image
SubMx_M	Obtaining a part of the image. Used for crop
Sub_M Abs_M	Used for Background subtraction.
UnPk_M Quant Pack_M	Used for Binarization

Maxtrix Mpy_M	Used for Object Extraction. Summing up the matrix vertically
TSV	TSV transform

Table 3. Explanation of each icon

Figure 5 Result of the Ptolemy simulation

In Figure 5, the input images at 20th, 40th, 60th, 80th and 100th and the TSV result images at each correspondence frame are shown. The horizontal axis of the original images and the TSV images are compatible and the vertical axes of the TSV images are velocity axes. The bright intensity represents the measure of existence in the position and velocity. Thus, a bright blob in the TSV image represents a human blob consisting of pixels with similar velocities and locations. Actually, we can see the bright blobs located in the same horizontal coordinate as the persons. We obtained same result in the PC-based system.

Hardware Design

Under the specification described in Table 4 and I designed a hardware that performs the tasks that I simulated in Ptolemy. Figure 6 is the diagram of the embedded system. Figure 7 is a top view picture of the hardware.

CPU	Hitachi H8/3048F Microprocessor 16bit CISC, 16MHz
Memory	Mitsubishi Multi-port DRAM M5M442256 (1Mbit x 4)
Implementation	Hitachi C Compiler/Assembler

Table 4. Hardware Specification

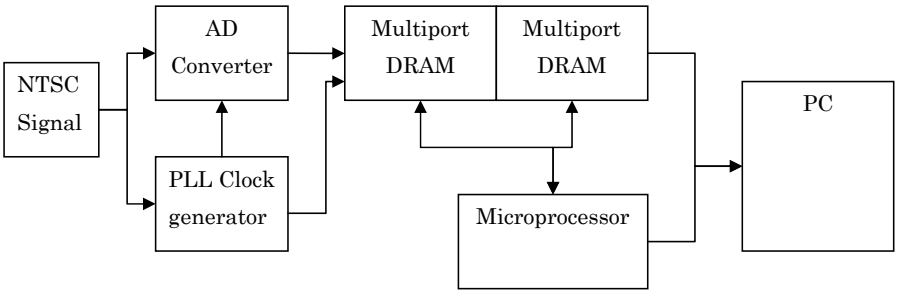

Figure 6 Diagram of the hardware

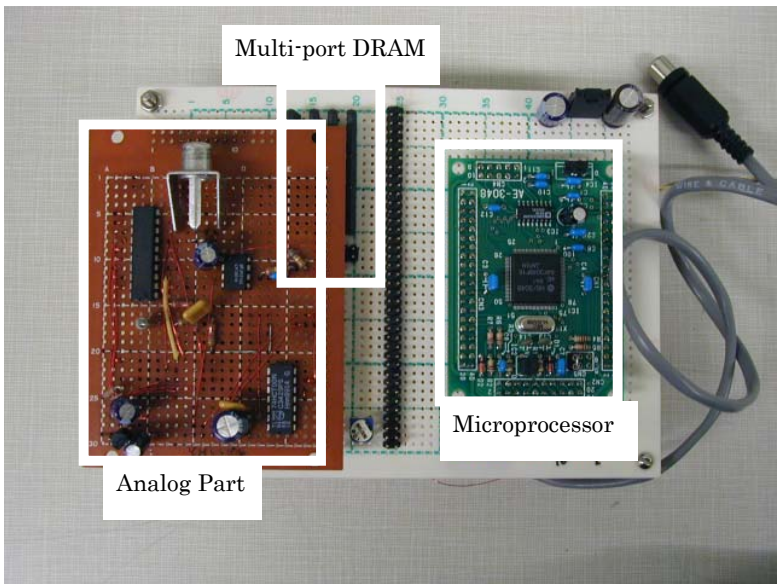

Figure 7 Picture of hardware design

Table 5 shows the result of the hardware simulation. This system performs one frame process within 89 msec, which means that this system computes a frame data within three image frame cycles and performs 10 frames/second. Also, we can see that the code size, internal RAM usage, and the data transfer rate to a PC are all very small.

Computation time per one frame	89 [msec] < 100 [msec] = 3 frame-cycle
Code size	5732 bytes
Internal RAM usage	854 bytes
Data rate to PC	128kbps

Table 5. Result of Hardware simulation

Conclusion

In conclusion, I designed an embedded system that is a part of a hybrid system consisting of an embedded system and a PC-based system. I used a low-power CISC microprocessor and four multi-port DRAM in this application. The system performs 89 msec execution time per one frame, that is, 10 frame/second. The bottleneck of the performance of the system is the DRAM accessing time, because a huge amount of image data has to be transferred between the CPU and the DRAMs. In order to reduce the accessing time, I used some techniques, such as (i) an efficient code design that minimize the DRAM access time, (ii) scheduling of the DRAM access order so that the CPU can use the DRAM block transfer mode.

As described above, I discussed a surveillance system using a CPU. However, in general, a DSP is considered to be most appropriate for a system dealing with a huge amount of data. Simulation and system design using a DSP, therefore, might better serve this purpose. I leave it as a future research project.

References

[1] Pentland, A. “Looking at people: sensing for ubiquitous and wearable computing”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume: 22 Issue: 1, Jan. 2000, Page(s): 107 -119

[2] Mahonen, P., “Wireless video surveillance: system concepts”, *Proceedings International Conference on Image Analysis and Processing*, 1999., 1999 , pp 1090 -1095

[3] Hougen, D.F.; Benjaafar, S.; Bonney, J.C.; Budenske, J.R.; Dvorak, M.; Gini, M.; French, H.; Krantz, D.G.; Li, P.Y.; Malver, F.; Nelson, B.; Papanikolopoulos, N.;

Rybski, P.E.; Stoeter, S.A.; Voyles, R.; Yesin, K.B., "A miniature robotic system for reconnaissance and surveillance**",** *IEEE International Conference on Robotics and Automation,* Vol. 1, pp: 501 -507, 2000.

[4] Adler, E.; Clark, J.; Conn, M.; Phuong Phu; Scheiner, B. "Low-cost technology for multimode radar", *IEEE Aerospace and Electronics Systems Magazine*, Vol. 14, No. 6 , June 1999, pp 23 -27

[5] Marcenaro, L.; Oberti, F.; Foresti, G.L.; Regazzoni, C.S., "Distributed architectures and logical-task decomposition in multimedia surveillance systems", *Proceedings of the IEEE*, Vol. 89, No. 10, Oct. 2001, pp 1419 -1440

[6] Soatto, S.; Frezza, R.; Perona, P., "Motion estimation via dynamic vision**"**, *IEEE Transactions on Automatic Control*, Vol. 41, No. 3 , March 1996, pp 393 -413

[7] K.Sato and J.K.Aggarwal, "Tracking and Recognizing Two-person Interaction in Outdoor Image Sequences", in *IEEE Workshop on Multi-Object Tracking*, pp.87-94, Vancouver, CA, July, 2001.

[8] K.Sato and J. K. Aggarwal, "Tracking objects using temporal spatio-velocity transform", *2001 IEEE Workshop on Performance Evaluation of Tracking and Surveillance*, Kauai, Hawaii, December, 2001.

[9] Y. Nara and S. Nagasaka, "Basic transistor TV textbook", *Ohm Publication*, Japan.

[10] Shirai, Y.; Miura, J.; Mae, Y.; Shiohara, M.; Egawa, H.; Sasaki, S, "Moving object perception and tracking by use of DSP", *Proceedings of Computer Architectures for Machine Perception*, Nov. 1993, pp. 251 -256