Journal of VLSI Signal Processing 42, 57–68, 2006
© 2006 Springer Science + Business Media, Inc. Manufactured in The Netherlands.
DOI: 10.1007/s11265-005-4163-z

Designing an Embedded Video Processing Camera Using a 16-bit Microprocessor for a Surveillance System

KOICHI SATO
Department of Electrical and Computer Engineering and Computer and Vision Research Center, The University of Texas at Austin, Austin, TX 78712, USA

BRIAN L. EVANS
Department of Electrical and Computer Engineering and Embedded Signal Processing Laboratory, The University of Texas at Austin, Austin, TX 78712, USA

J.K. AGGARWAL
Department of Electrical and Computer Engineering and Computer and Vision Research Center, The University of Texas at Austin, Austin, TX 78712, USA

Received October 7, 2003; Revised January 29, 2004; Accepted March 21, 2004

Abstract. This paper describes the design and implementation of a hybrid intelligent surveillance system that consists of an embedded system and a personal computer (PC)-based system. The embedded system performs some of the image processing tasks and sends the processed data to the PC. The PC tracks persons and recognizes two-person interactions by using a grayscale side view image sequence captured by a stationary camera. Based on our previous research, we explored the optimum division of tasks between the embedded system and the PC, simulated the embedded system using dataflow models in Ptolemy, and prototyped the embedded system in real-time hardware and software using a 16-bit CISC microprocessor. This embedded system processes one 320 × 240 frame in 89 ms, which yields one-third of the rate of 30 Hz video system. In addition, the real-time embedded system prototype uses 5.7 K bytes of program memory, 854 K bytes of internal data memory and 2 M bytes external DRAM.

Keywords: interaction, recognition, tracking, surveillance, video, temporal spatio-velocity transform, velocity, microprocessor, DRAM, embedded system, Ptolemy

1. Introduction

Tracking, recognizing and detecting objects using a video sequence are topics of significant interest in computer vision. Cai and Aggarwal and [1] reported a variety of methods for analyzing human motion. In [2,3], Haritaoglu *et al.*, tracked both single and multiple humans in outdoor image sequences using a PC. In [4], Oliver *et al.*, recognized two-person interactions from perspective-view image sequences. They tracked persons and recognized their interaction using trajectory patterns. Meanwhile, some researchers developed applications that included one or more embedded systems. In [5], Pentland proposed a "wearable device" that sees people using an image sensor and understands the environment. In such equipment, a computer can act or respond appropriately without detailed instructions from humans. In [6], Mahonen proposed a wireless intelligent surveillance camera system that consists of a digital camera, a general-purpose processor or DSP for image processing and a wireless radio modem. In [7], Shirai *et al.* introduced a real-time surveillance system with parallel DSP processors (TI TMS320C40). Their system consists of several boards connected in series. It can compute optical flow in floating point faster than 30 frames per second. In this system, a DSP is located between two memories. The DSP computes and transfers the image data from the video memory to the other memory connected to the next processing stage.

In [8], Sato and Aggarwal proposed a surveillance system that recognizes interactive human activities using a side view image sequence. The system consisted of a single monochrome video camera, a video-capturing device and a PC. The system is located along a sidewalk and captures human movements from a lateral perspective. Persons in the sequences move horizontally and the pattern of their motion is used to determine the interaction.

Our surveillance system made use of outdoor, side view image sequences. There are some potential difficulties associated with the external conditions of our system. First, outdoor image sequences tend to contain shadows and moving trees and leaves. These features can make segmentation difficult. Second, side view image sequences are more prone to occlusion than are top-view or perspective images. Our strategy to overcome these problems was to use the temporal spatio-velocity (TSV) transform [8, 9]. The TSV transform estimates pixel velocities from a binary image sequence. The segmentation based on the TSV-transformed images can separate occluded blobs moving at different velocities.

In the practical implementation of the system, we use several cameras to avoid the problem of the limited field of view inherent in a single-camera system. This in turn causes a problem: more PCs are required because one PC can only achieve real-time performance when processing video data from one camera with 320×240 pixels at a 30 Hz frame rate.

To overcome this problem, we propose an embedded system for each camera that performs the fundamental image processing tasks (that is, the human segmentation processing) and sends the output data to a PC. This system is potentially able to connect up to 12 embedded cameras simultaneously at 10 frames per second.

The structure of the paper is as follows. Section 2 describes the TSV transform. Section 3 introduces its application in the human tracking system. Section 4 illustrates the algorithms used in our system and addresses the issue of optimizing the division of tasks between the embedded system and PC-based system. Section 5 models and simulates the embedded system using dataflow models in [10]. Section 6 discusses the hardware design. Section 7 is a conclusion. This paper is an expanded version of [11].

2. Temporal Spatio-Velocity Transform

The temporal spatio-velocity (TSV) transform, first proposed in [8, 9] is a pixel extraction method dealing with binary image sequences. Its interesting feature is that it generates velocity spectrums. The generated spectrum consists of spatial and velocity axes. Each value in the spectrum represents the weight of its coordinates. We use this feature in segmenting moving blobs. By thresholding the velocity spectrum, we obtain stable segmentation of the blobs based on position and velocity similarities.

The TSV transform is a combination of the Hough transform and a windowing process. The dimension of the resulting TSV image sequence equals to the sum of the positional and velocity dimensions. For example, a TSV transform over a one-dimensional binary image sequence yields a two-dimensional spectrum consisting of positional and velocity axes.

Let the input binary image sequence in the nth frame be $H_n(x)$ and the windowing function be $F(n)$. Then, the TSV transform generates a two-dimensional spectrum $V_n(p, v)$,

$$V_n(p, v) = \underset{x=vn+p}{\text{Hough}}\{F(n) \cdot H_n(x)\}, \tag{1}$$

where $x = vn + p$ is the line used in Hough transform.

We use an exponential window so that newer frames affect the TSV transform more. So,

$$F(n) = \begin{cases} (1 - e^{-\lambda})e^{-\lambda n} & (n \geq 0) \\ 0 & (n < 0). \end{cases} \tag{2}$$

We can simplify (2) as

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

where $e^{-\lambda}$ is the coefficient in the windowing function. It controls the accepted acceleration range. Figure 1 illustrates the computation process of the TSV transform.

One of the advantages of TSV transform is its computational simplicity. As seen in (3), the TSV trans-

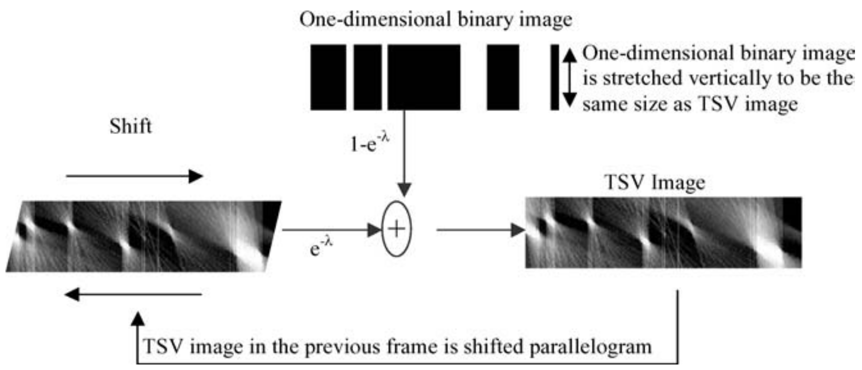
One-dimensional binary image
One-dimensional binary image is stretched vertically to be the same size as TSV image
Shift
1-e^-λ
e^-λ
TSV Image
TSV image in the previous frame is shifted parallelogram

Figure 1. Computing the TSV transform.

form consists of simple iterative numerical operations. Therefore, this method is easily applicable to embedded systems. Another advantage is its noise suppressive feature. In general, velocity extraction is a certain form of a differential operation, which tends to be a noisy process. However, our TSV transform is not a differential operation; it is an instance of a low-pass filtering method, which smoothes noise.

3. Tracking Humans in Lateral-View Image Sequences

We apply the TSV transform to human tracking applications. Specifically, we consider tracking persons that walk along a sidewalk and recognize their interactions. The system uses lateral view grayscale images, in which persons walk on a flat plane. Each image may contain many blobs that are distinct in size depending on their distance from the camera. There are frequent occlusions among the blobs. We adopt the TSV transform in order to segment and track the human blobs without failure due to occlusion. The system tracks the persons and recognizes their interactions using the trajectory of the persons.

In [8], we implemented our interaction recognition system using a PC and a camera. Figure 2 shows an overview of our system. In the figure, letters A-I represent process stages and numbers 1–9 denote data transfer stages. The camera takes video images and generates the video signal. The PC captures the video signal and completes all the processing stages A-I faster than 30 frames/s. However, the system has a limitation that the observation range depends on the field of view of the camera. In order to cover a wider observation range, we use multiple cameras. Since the PC in the system [8] only achieves real-time performance when processing 320 × 240 pixels video data from one camera at a 30 Hz frame rate, more than one PC is needed.

In order to reduce the total cost, our system consists of several embedded systems and one PC. The embedded systems with a camera and a small microprocessor perform some of the tasks and the PC performs the rest.

4. Assignment of Tasks to the Embedded System

In designing the surveillance system, we considered an optimal division of tasks between the embedded system and the PC-based system that takes full advan-

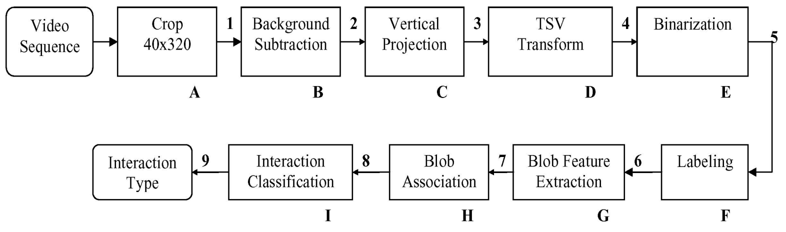

Figure 2. Overview of single camera system with process tasks A-I and process connection 1–9.

tage of unique characteristics of each. For example, a statically schedulable and fine granularity process such as an image differencing operation is more effectively performed on an embedded processor, whereas a database operation is more effectively performed with a PC-based system. In order to optimize the division of tasks between the embedded and PC-based systems, the computational complexity of each operation and the transfer rate and overall suitability of each processor were evaluated.

In the discussion below, we first describe the algorithms, and then discuss and evaluate the way we divide the tasks. As a measure of the computational complexity, we use the total number of the operations such as additions, subtractions, multiplications and comparisons. We use computational complexity and data transfer rate to determine the division of tasks.

4.1. *Crop*

In order to extract persons, we crop an $S_H \times S_V$ image into an $S_H \times R_V$ region, where S_H, S_V are the horizontal, vertical size of the image and R_V is the height of the region. This region is set to slightly above and below the horizon level so that all the human blobs intersect with this region. As the result, the noise outside the region such as tree and shadow is eliminated without missing any human blob. Since the output size of the image is $S_H \times R_V$ pixels and each pixel consists of 8-bit data, the data transfer rate to the next process is estimated as $8S_H \times R_V$ [bits/frame] (Figure 3).

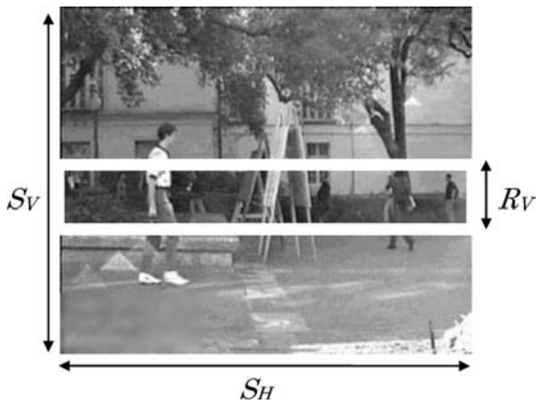

Figure 3. Image size and cropping area.

4.2. *Background Subtraction*

Then we perform a simple background subtraction and binarization to segment the foreground region using a threshold *Th*,

$$S(x, y) = \begin{cases} 1 & |I(x, y) - B(x, y)| > Th \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

To avoid missing human blobs that have intensities similar to the background, we use a low threshold, thereby accepting some noise. Since this process consists of one subtraction and one thresholding process for each pixel, $S_H \times R_V \times 2$ is chosen as an estimate of the computational complexity per frame. Since the output data is an $S_H \times R_V$ binary image, data transfer rate to the next stage is estimated as $S_H \times R_V$ [bits/frame].

4.3. *Vertical Projection*

In order to reduce the noise, we perform a vertical projection over the entire image of the size $S_H \times R_V$ and re-binarize the projection value by a threshold T_H. This process yields a one-dimensional image H(x):

$$H(x) = \begin{cases} 1 & \sum_{y=a}^{b} S(x, y) > T_H \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Here, T_H is a threshold that is constant in all situations and H(x) is the extracted binary image for the object. As a result, we get a sequence of one-dimensional binary images. For this process, the system performs R_V additions and one comparison for each vertical line, so $S_H \times (R_V + 1)$ could be the estimate of the computational complexity per frame, while S_H [bits/frame] is the estimate of the data transfer rate.

4.4. *The TSV Transform*

Here the TSV transform here is used for segmenting the moving persons. Since the TSV transform consists of two multiplications and one addition for each pixel in $S_H \times V_H$ image, we estimate $S_H \times V_V \times 3$ computation per frame as the computational complexity. The TSV transform generates an 8-bit grayscale image, so the transfer rate is $S_H \times V_V \times 8$ [bits/frame].

4.5. Binarization of the TSV Image

To group pixels with a similar velocity together, we binarize the TSV image by a fixed threshold Th_V.

$$\tilde{V}_n(x, v) = \begin{cases} 1 & \text{if } V_n(x, v) \geq Th_V \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

For binarization, we compute one comparison at each TSV point. This translates into $S_H \times V_V$ computations per frame and results in a data transfer rate of $S_H \times V_V$ bits/frame.

4.6. Labeling

A labeling method connects the surrounding '1'-valued pixels in the binary image. For each pixel, the four points surrounding it are the candidates for the connection. The process involves two comparisons per pixel in deciding the connection. Since the image size is $S_H \times V_V$, we estimate $2S_H \times V_V$ for the computational complexity. The process outputs an $S_H \times V_V$ matrix array of an 8-bit code. The 8-bit code represents the labeling number. The data transfer rate is estimated as $8S_H \times V_V$ [bits/frame].

4.7. Feature Extraction

In identifying each object blob, we use several features such as blob area, blob width, position and velocity. We estimate that there is one computation for each feature when scanning the entire matrix array. Thus, our estimation is 8 $S_H \times V_V$ operations per frame. The output of the feature extraction is the array of the feature values. We assume that the system extracts 20 blobs and that the feature size is 8 bytes per blob. The data transfer rate is thus 1280 bits/frame.

4.8. Blob Association

In order to track human blobs, we associate the blobs across the frames using feature similarities. We compute a feature similarity between possible combinations of the blobs in consecutive frames. Since we assume that there are 20 blobs in the frame, the possible combination is 20×20 pairs. For each blob similarity computation, we have 4 subtractions and 4 multiplications. So the computational complexity would be 6400 multiply-accumulates. Since the output of the blob association is the positional information, the data transfer rate is the 320 bits/frame.

4.9. Interaction Classification

The system determines an interaction using the human trajectories. This process classifies an interaction type using trajectory shapes. It compares predetermined interaction models with the current model and determines its classification. This process has more than 10 k operations per frame.

4.10. Evaluating Processes by Data Transfer Rate and Computational Complexity

Figure 4 depict the variation of the data transfer rate (left axis) and the computational complexity (right axis) as functions of the processes involved in our system.

For the actual computation, we assumed that the system digitized a 30 [fps] monochrome video signal into a 320×240 image sequence. Applicable computation cycle of the system ranges from 10 Hz to 30 Hz. Also, we used $R_V = 30$ [pixels] and $V_V = 40$ [pixels]. Based on these assumptions, we assigned to the embedded system stages A through E, while stages F through I were handled by the PC based system. The overall surveillance system is feedforward, and we want to partition the blocks to maximize embedded computations and minimize the amount of data transfer from the embedded system to the PC.

4.11. Constraints Based on Computational Complexity and Transfer Rate

We seek an optimal division of tasks in the surveillance system to provide the maximum overall performance. "Maximum overall performance" means that the system covers the maximum total view-range in real-time. We assume that the system consists of one PC and several embedded cameras, so the total view-range depends on the number of embedded cameras.

The number of connectable embedded camera N_c depends on several variables and constants shown in Table 1.

First, a task of the embedded system should be less than or equal to the embedded system's capability. So

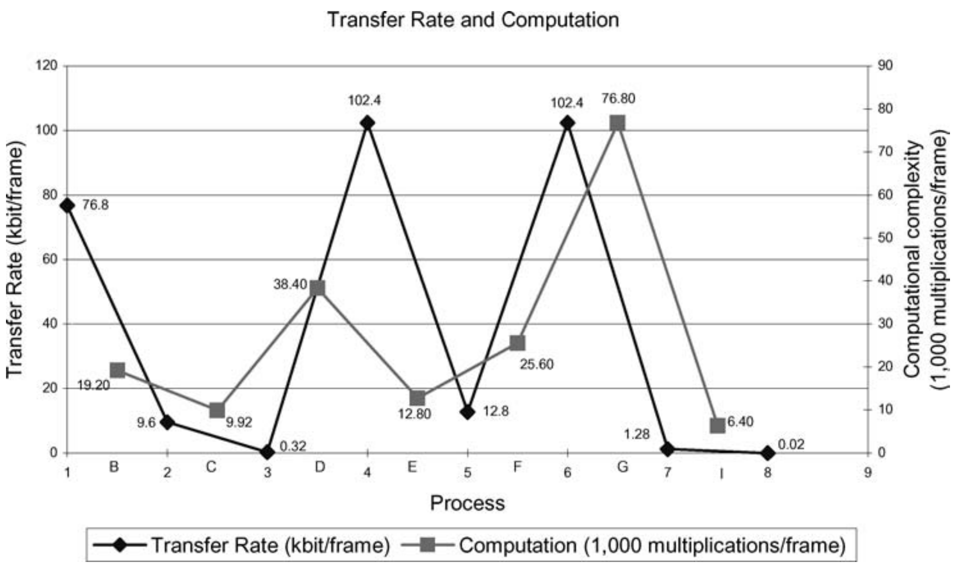

Figure 4. Data transfer rate and computational complexity.

we have the constraint.

$$fc \leq C_E. \tag{7}$$

Next, the task of PC $N_c f\{C_{\text{total}} \times c\}$ is also required to be less than or equal to the PC's processing capability. So

$$N_c f\{C_{\text{total}} - c\} \leq C_{PC}. \tag{8}$$

Finally, we need to consider the line capacity between the embedded system and the PC. We assume that the data transfer line is connected to every embedded system and the PC, such as an Ethernet connection. The total data transfer rate is necessarily less than or equal to the data line capability. Thus,

$$N_c f T \leq T_{EP}. \tag{9}$$

By rewriting the formulae (7), (8) and (9),

$$N_c \leq \frac{C_{PC}}{f\,(C_{\text{total}} - c)}, \tag{10}$$

$$c \leq \frac{C_E}{f}, \tag{11}$$

$$N_c \leq \frac{T_{EP}}{fT}. \tag{12}$$

We estimate C_E, C_P and T_{EP} as shown in Table 2. C_E is obtained based on an experiment. C_P is approximated from the executing time of a whole system. We assume the transfer line is 10 Mbps and we set 30% as T_{EP}.

Figures 5 and 6 are graphs showing the number of connectable embedded cameras N_c with respect to the boundary of the embedded system and the PC. We consider two values for f, namely 30 Hz or 10

Table 1 Variables and constants affect the number of connectable cameras

c	The total computational complexity for embedded system [operations/frame]
C_E	The upper limit of computational ability of the embedded system [operations/sec]
C_P	The upper limit of computational ability of the PC [operations/sec]
C_{total}	Total computational complexity of whole system [operations/frame]
T_{EP}	The upper limit of transfer rate from the embedded system to the PC [bits/sec]
T	Transfer rate from the embedded system to the PC [bits/frame]
f	Computational cycle [Hz]

Table 2 The estimation of the C_E, C_P and T_{EP}.

C_E	100,000 [operations/sec]
C_P	500,000 [operations/sec]
T_{EP}	3M [bits/sec]

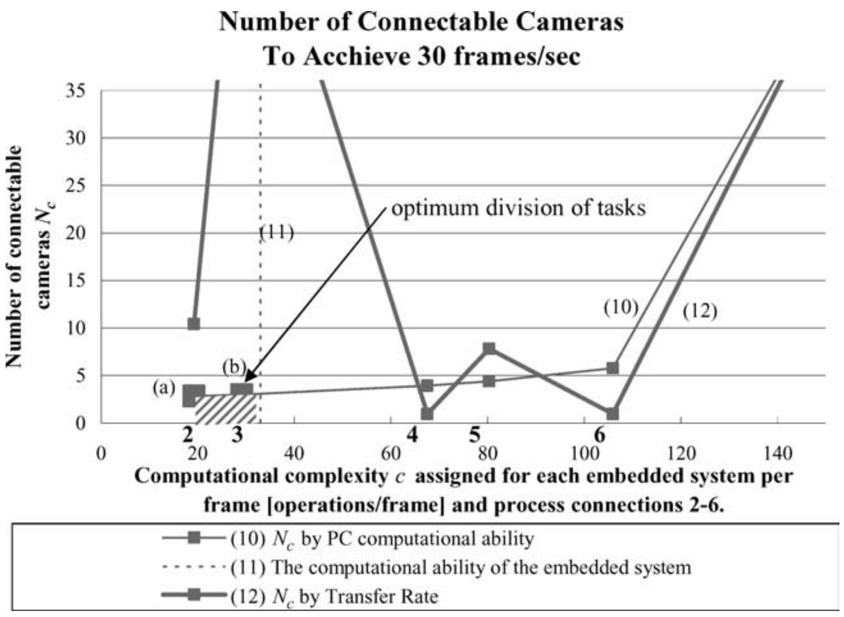

Figure 5. The number of connectable cameras N_c and the computational complexity c assigned for the embedded systems to achieve 30 frames/sec. The shaded area is the region meeting constraints (10)–(12).

Hz in Figures 5 or 6 respectively. An embedded-PC boundary may be set at any Process Connection **2–8**, we consider Process Connection **2–6** and represent them on the horizontal axis. On this axis we also illustrate the computational complexity c assigned to the embedded camera. The vertical axis is the number of connectable embedded camera N_c.

We have three conditions (10)–(12) to have a valid N_c. Each condition boundary represents as (10)–(12) respectively. In particular, we need to take into account the computational complexity c and the Process Connection **2–6** because condition (10) and (11) are based on c while condition (12) is based on Process Connection.

We present c and Process connection **2–6** on the same axis so that we can show (10) through (12) in a single graph. For each value of c to which we have a corresponding process connection, the numbers of the connection **2–6** are added. (10) and (11) are written based on c, while we use the number of the corresponding Process Connection **2–6** in writing (12).

The valid area for c and N_c is shown as the shaded area, that is the cross section area of conditions (10)–(12). Process Connections **2–6** that meet the conditions (10)–(12) are marked with a large triangle and square for 30 Hz and 10 Hz respectively. The maximum n, namely f for the 30 Hz system and f for the 10 Hz, is thus obtained.

We determined the computational frequency as 10 Hz, and then we find the optimal division of tasks (d) as shown Figure 7.

Table 3. The possible division of task and the number of connectable cameras.

30 Hz		10 Hz	
Each division of task	The number of connectable embedded cameras	Each division of task	The number of connectable embedded cameras
(a)	2	(a)	8
(b)	3	(b)	9
		(c)	2
		(d)	12

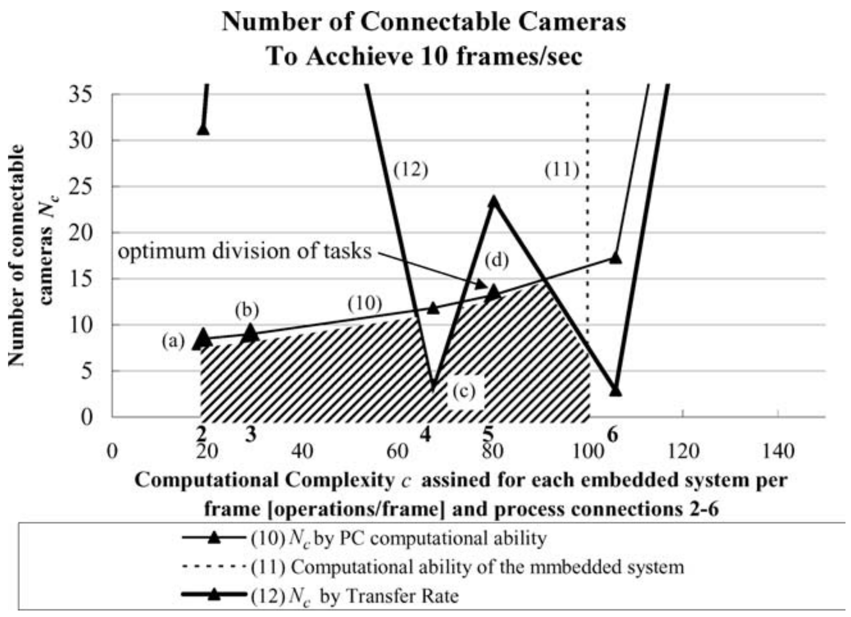

Figure 6. The number of connectable cameras N_c and the computational complexity c assigned for the embedded systems to achieve 10 frames/sec. The shaded area meets constraints (10)-(12).

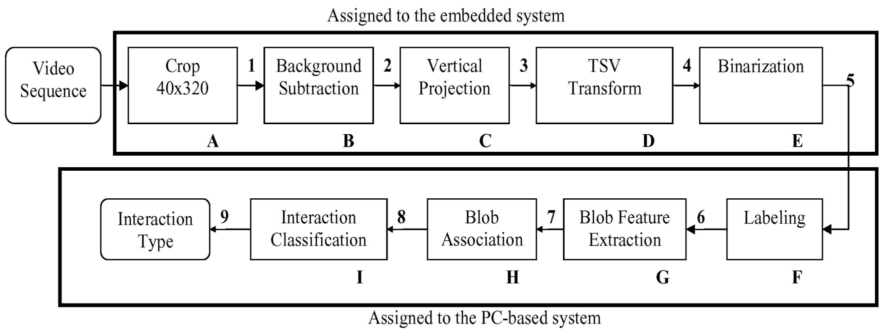

Figure 7. Division of tasks between the embedded and PC-based platforms.

5. Dataflow Modeling and Ptolemy Simulation

In the previous section, we determined the tasks to be assigned to the embedded system. In this section, we discuss the design and simulation of the embedded system using Ptolemy Classic [10]. Figure 8 shows a block diagram of the embedded system on Ptolemy Classic. Figure 9 gives the result of simulation. Figure 8 follows the same steps as the block diagram discussed in Figure 2.

In Figure 9, the input image sequence at the 20th, 40th, 60th, 80th and 100th frame (top rows) and the resulting TSV image at each corresponding frame (bottom rows) are shown. The horizontal axis of the original images and the TSV images are compatible. The vertical axes of the TSV images are velocity axes. The intensity indicates the measure of existence in a given 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 that the bright blobs are located in the same horizontal coordinates as the persons. We validated the Ptolemy Classic simulation results by comparing them with a previous PC implementation. We obtained the same result for the two experiments.

6. Hardware Design

With the specifications described in Table 4, we designed hardware to perform the tasks that we simulated

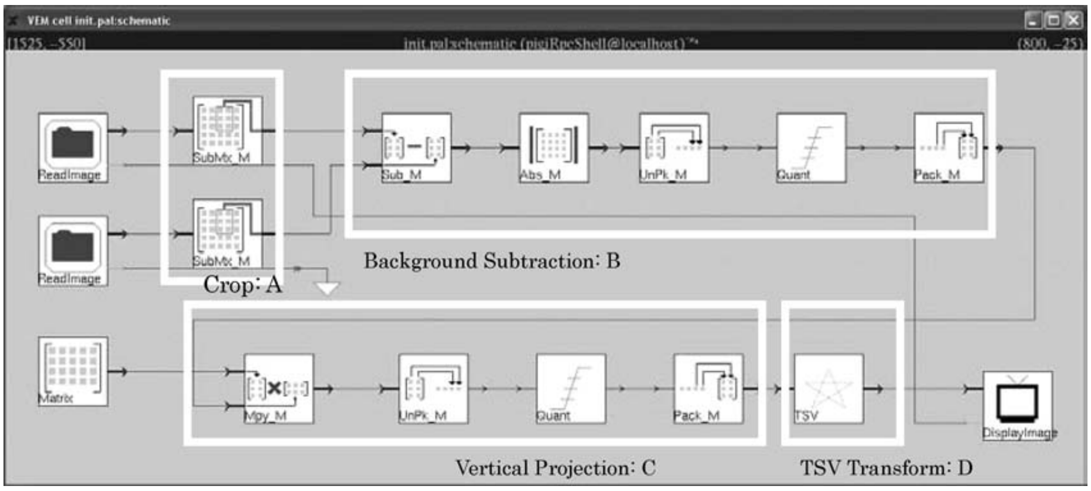

Figure 8. System design in Ptolemy Classic 0.7.1.

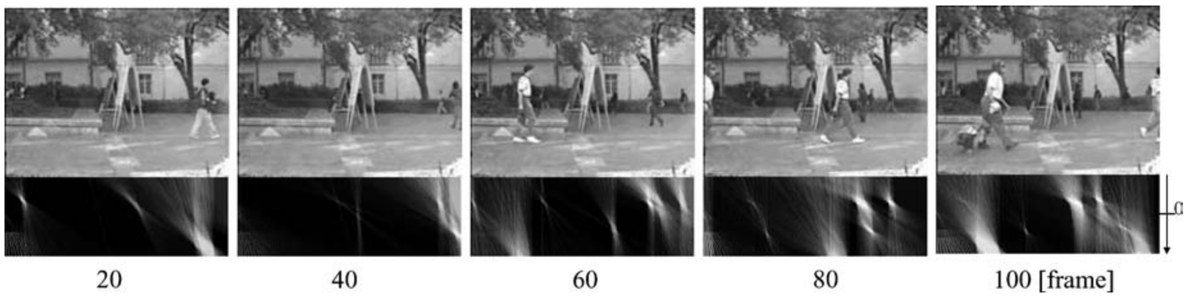

Figure 9. Results of the Ptolemy Classic 0.7.1 simulation.

in Ptolemy Classic. Figure 10 is a diagram of the embedded system. Figure 11 is a top view picture of the hardware.

The design consists of a video digitizing circuit, two 4 M-bit multi-port DRAM and a 16-bit microprocessor. The video clock is generated using a synchronization signal composed in the video signal. The data rate to the multi-port DRAM is 2 M bits/frame. The multi-port DRAMs obtain image data using their serial bus, and export memory data using DRAM interface. The microprocessor has a DRAM access port that facilitates coding.

Table 5 shows the results of a hardware simulation. This system processes one frame within 89 ms, which means that this system computes the amount of data corresponding to one frame within three image frame cycles and outputs 10 frames per second. Also, we can see that the code size, internal RAM usage, and the data transfer rate to a PC are all very small.

Table 4. Hardware specification.

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

Table 5. Result of hardware simulation.

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

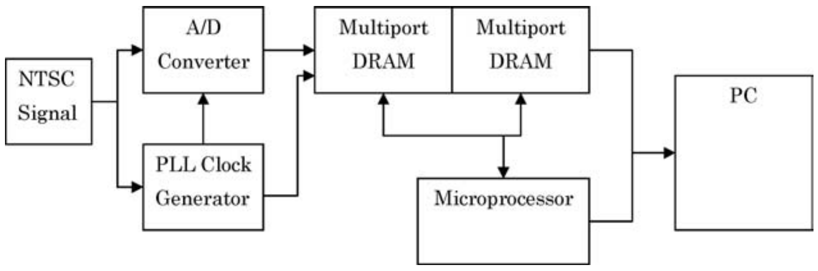

Figure 10. Diagram of the hardware design: NTSC National Television System Committee, PLL phase locked loop, and A/D Analog-to-digital converter.

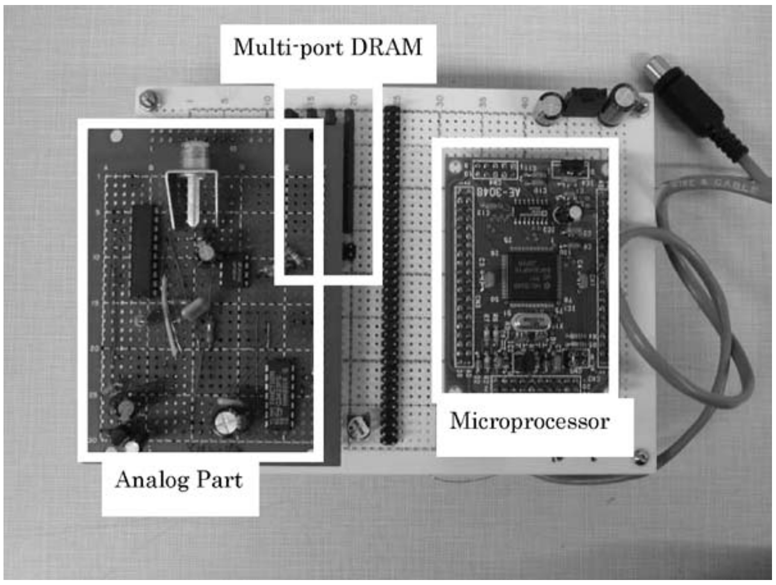

Figure 11. Picture of hardware design.

7. Conclusion

We have designed an embedded system that is a part of a hybrid system consisting of an embedded system and a PC-based system. We used a low-power CISC microprocessor and four multi-port DRAMs in this application. The execution time of the system is 89 ms per frame, that is, 10 frames/s. The performance bottleneck of the system is the DRAM access time, because a large amount of image data has to be transferred between the CPU and the DRAMs. We used several techniques to reduce the access time, including (i) designing an efficient code that minimizes the DRAM access time, and (ii) scheduling the DRAM access order so that the CPU can use the DRAM in the block transfer mode.

We have discussed a surveillance system using a 16-bit general purpose processor. However, a DSP is generally 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. We leave it as a future research project.

References

1. J.K. Aggarwal and Q. Cai, “Human Motion Analysis: A Review,” *Computer Vision and Image Understanding*, vol. 73, no. 3, 1999, pp. 428–440.
2. I. Haritaoglu, D. Harwood, and L. Davis, “W4: Who, When, Where, What: A Real Time System for Detecting and Tracking People,” in *Proc. Int. Conf. on Automatic Face and Gesture*, Nara, April 1998, pp. 222–227.

3. I. Haritaoglu and L. Davis, “Hydra: Multiple People Detection and Tracking Using Silhouettes,” in *Proc. IEEE Work. on Visual Surveillance*, Fort Collins, Colorado, June 1999, pp. 6–13.
4. N. Oliver, B. Rosario, and A. Pentland, “A Bayesian Computer Vision System for Modeling Human Interactions,” *IEEE Transactions on Pattern Analysis and Machine Interlligence*, vol. 22, no. 8, Aug. 2000, pp. 831–843.
5. A. Pentland, “Looking at People: Sensing for Ubiquitous and Wearable Computing” in *IEEE Transactions on Pattern Analysis and Machine Intel.*, vol. 22, no. 1, Jan. 2000, pp. 107–119.
6. P. Mahonen, “Wireless Video Surveillance: System Concepts,” in *Proc. Int. Conf. on Image Analysis and Processing*, Sep. 1999, pp. 1090–1095.
7. Y. Shirai, J. Miura, Y. Mae, M. Shiohara, H. Egawa, and S. Sasaki, “Moving Object Perception and Tracking by use of DSP,” in *Proc. of Computer Architectures for Machine Perception*, Nov. 1993, pp. 251–256.
8. K. Sato and J.K. Aggarwal, “Temporal Spatio-Velocity Transform and its Application to Tracking and Interaction,” *Computer Vision and Image Understanding*, vol. 96, issue 2, Nov. 2004, pp. 100–128.
9. K. Sato and J.K. Aggarwal, “Tracking Objects Using Temporal Spatio-Velocity Transform,” in *Proc. IEEE Work. on Performance Eval. of Tracking and Surveillance*, Kauai, Hawaii, Dec. 2001.
10. Ptolemy Project, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, http://ptolemy.eecs.berkeley.edu/.
11. K. Sato, B.L. Evans, and J.K. Aggarwal, “Designing an Embedded Video Processing Camera Using a 16-bit Microprocessor for Surveillance System,” in *Int. Work. on Digital and Computational Video*, Clearwater, FL, Nov. 2002, pp. 151–158.

Koichi Sato is a Ph.D. student in the Department of Electrical and Computer Engineering at The University of Texas at Austin. He earned his B.S. in University of Tokyo, Japan in 1993. He worked for Automotive Development Center in Mitsubishi Electric Corporation where he was involved in lane and automobile recognition in vehicle video processing products such as automatic cruise control and drowsiness detection systems. He enrolled in the current University at 1998 and received an M.S in 2000. In his Master’s thesis he worked on human tracking and human interaction recognition. His current work includes velocity extraction using the TSV transform, object tracking, and 3D object reconstruction.
koichisato@alumni.utexas.net

Brian L. Evans is a tenured Associate Professor in the Department of Electrical and Computer Engineering at The University of Texas at Austin. His research and teaching efforts are in embedded real-time signal and image processing systems. In signal processing, his research group is focused on the design and real-time software implementation of ADSL and VDSL transceivers, for high-speed Internet access. In image processing, his group is focused on the design and real-time software implementation of high-quality halftoning for desktop printers, smart image acquisition for digital still cameras, and 3-D sonar imaging systems. In signal and image processing, Dr. Evans has published over 100 refereed conference and journal papers. Dr. Evans is the primary architect of the Signals and Systems Pack for Mathematica, which has been on the market since October 1995. He was a key contributor to UC Berkeley’s Ptolemy Classic electronic design automation environment for embedded systems, which has been successfully commercialized by Agilent and Cadence. His BSEECS (1987) degree is from the Rose-Hulman Institute of Technology, and his MSEE (1988) and PhDEE (1993) degrees are from the Georgia Institute of Technology. From 1993 to 1996, he was a post-doctoral researcher in the Ptolemy project at UC Berkeley. He is a member of the Design and Implementation of Signal Processing Systems Technical Committee of the IEEE Signal Processing Society, and a Senior Member of the IEEE. He is the recipient of a 1997 National Science Foundation CAREER Award.
bevans@ece.utexas.edu

J.K. Aggarwal has served on the faculty of The University of Texas at Austin College of Engineering since 1964 and is currently Cullen Professor of Electrical and Computer Engineering and Director of the Computer and Vision Research Center. His research interests include computer vision and pattern recognition focusing on human motion. A Fellow of IEEE since 1976 and IAPR since 1998, he received the Senior Research Award of the American Society of Engineering Education in 1992, the 1996 Technical Achievement Award of the IEEE Computer Society and the graduate teaching award at The University of Texas at Austin in 1992. He has served as Chairman of the IEEE Computer Society Technical Committee on Pattern Analysis and Machine Intelligence (1987–1989); Director of the NATO Advanced Research Workshop on Multisensor

Fusion for Computer Vision, Grenoble, France (1989); Chairman of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (1993), and President of the International Association for Pattern Recognition (1992–1994). He is a Life Fellow of IEEE and Golden Core member of IEEE Computer Society. He has authored and edited a number of books, chapters, proceedings of conferences, and papers.
aggarwaljk@mail.utexas.edu