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

Koichi Sato[*†], Brian L. Evans[*‡] and J. K. Aggarwal[*†]

[†]Computer and Vision Research Center
[‡]Embedded Signal Processing Laboratory
[*]Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin, TX 78712, USA
{ koh@mail.utexsas.edu , bevans@ece.utexas.edu , aggarwaljk@mail.utexas.edu }

Abstract

This paper describes the design and implementation of a hybrid intelligent surveillance system consisting 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 a 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 systems in real-time hardware and software using a 16-bit CISC microprocessor. This embedded system processes one frame image in 89 ms, which is within three frame-cycle periods for a 30Hz video system. In addition, the real-time embedded system prototype uses 5.7K bytes of program memory, 854K bytes of internal data memory and 2M bytes external DRAM.

1. Introduction

Tracking, recognizing and detecting objects using a video sequence are topics of significant interest in computer vision. Cai and Aggarwal [1] reported a variety of methods for analyzing human motion. In [2-3], Haritaoglu, Harwood and Davis tracked both single and multiple humans in outdoor image sequences using a PC. In [4], Nuria, Rosario and Pentland 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 computation in floating point faster than 30 frames per second. In this system, a DSP is located between the two memories. The DSP computes and transfers the image data from the video memory to the other memory, which is connected to the next processing stage.

In [8], we proposed a surveillance system that recognizes interactive human activities using a side-view image sequence. The system consists of a single monochrome video camera, a video-capturing device and a PC. The system is located along a sidewalk and it thus captures human movement from a side-view. 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. The external conditions for our system create potential difficulties. First, outdoor image sequences tend to contain shadows and moving trees and leaves, which makes segmentation more 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 is more stable because it uses both spatial and velocity information.

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 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 in 10 frame/s. In this paper, Section 2 describes the algorithms used in our system and discusses a division of tasks for the embedded systems. Section 3 models and simulates the embedded system using dataflow models in Ptolemy [10]. Section 4 discusses the hardware design. Section 5 draws conclusions.

2. Assignment of tasks to the embedded system

In designing the surveillance system, we considered the optimum division of tasks between the embedded system and the PC-based system to take full advantage the unique characteristics of each. For example, a statically schedulable, 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 by a PC-based system. 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, followed by a discussion and evaluation of 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. To determine suitability, we consider the code size, computational complexity within a loop and the number of branches in the process flow.

Figure 1 presents on overview of the system. In the figure, letters A-I represent process stages and numbers 1-9 denote data transfer stages.

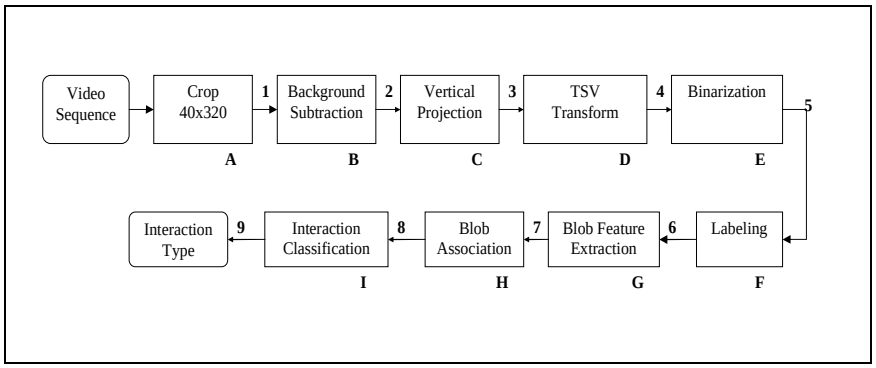

Figure 1. Overview of single camera system

Crop (A) and Background Subtraction (B)

In order to extract persons, we crop an S_HxS_V image into an S_HxR_V region. We set that region at the level of an average human torso so that all humans are extracted by this operation. As a result, noise outside the region is eliminated. Then we perform a simple background subtraction and binarization to segment the foreground region using a threshold *Th*,

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

To avoid missing human blobs that have intensities similar to the background, we use a low threshold, thereby accepting some noise. S_HxR_Vx2 is chosen as an estimate of the computational complexity per frame, while S_HxR_V [bits/frame] is an estimate of the data transfer rate.

Vertical projection (C)

In order to reduce the noise, we perform a vertical projection over the entire image (S_HxR_V size) and re-binarize the projection value by a 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 that is constant in all situations and $H(x)$ is the object extracted binary image. As a result, we get a sequence of one-dimensional binary images. For this process, S_Hx(R_V+1)x2 is chosen as an estimate of the computational complexity per frame, while S_H [bits/frame] is an estimate of the data transfer rate.

The TSV Transform (D) and Binarization (E)

The TSV transform, which was first proposed in [8,9], estimates the pixel velocity from binary image sequences. Here, a one-dimensional binary image sequence $H_n(x)$ is converted into an S_HxV_V size TSV image $V_n(x,v)$. S_H is the horizontal size of the original image and the vertical size V_V is determined by the resolution and range of the velocities.

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

For the TSV transform, we compute two multiplications and one addition for each TSV point. Thus, we have S_HxV_Vx3 computations per frame. The transfer rate is S_HxV_Vx8 [bits/frame].

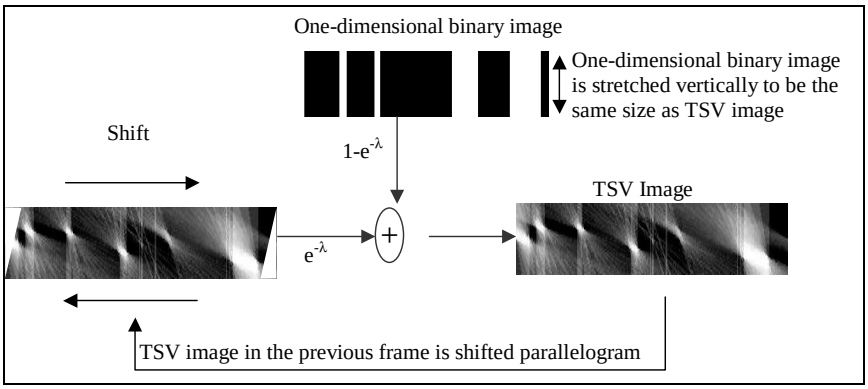

Figure 2. Computing the TSV Transform

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

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

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

Labeling(F), Feature Extraction(G), Blob Association(H) and Interaction Classification(I)

Once $V_n(x,v)$ is binarized, labeling over $V_n(x,v)$ yields human blobs (stage F). Features of the human blobs (stage E) are used to associate the human blobs over frames (stage H). Finally, the system determines the interaction using the human trajectories (stage I). The computation time for each process depends on the content of the images. We have assumed a reasonable number for each process. For labeling, we performed two comparisons at each point. For feature extraction, we assumed 20 blobs are labeled in the previous process, and we performed 300 computations for each blob.

Evaluating Processes by Data Transfer Rate and Computational Complexity

Figure 3 is a graph depicting the variation of the data transfer rate (left axis) and the computational complexity (right axis) as functions of the various processes involved in our system. Table 1 summarizes the suitability of the various stages for implementation on the embedded system.

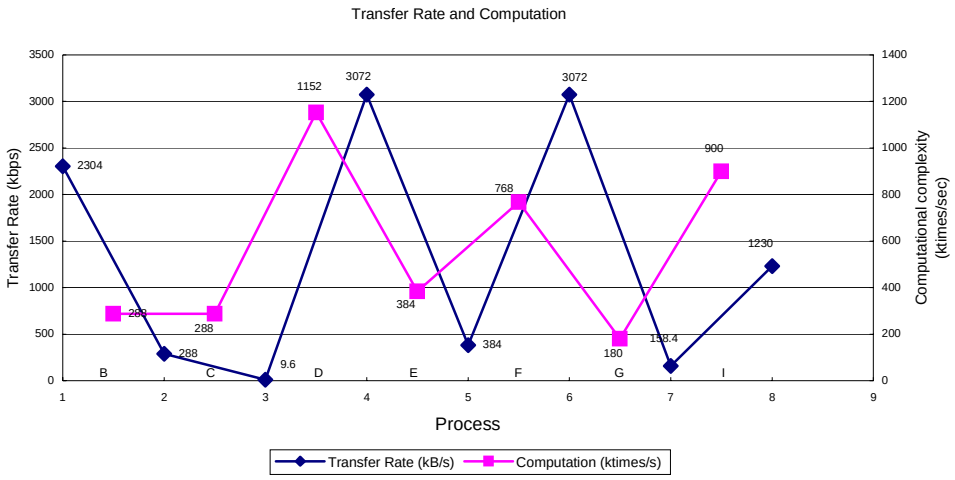

Figure 3. Data transfer rate and computational complexity

Step		Code size	Iterative computation	Number of branches	Suitability
A	Crop	Small	Simple	Small	Suitable
B	Subtraction	Small	Simple	Small	Suitable
C	V Projection	Small	Simple	Small	Suitable
D	TSV	Small	Simple	Small	Suitable
E	Binarization	Small	Simple	Small	Suitable
F	Labeling	Small	Medium	Medium	Fair
G	Feature Ext.	Medium	Medium	Small	Fair
H	Association	Large	Complex	Large	Unsuitable
I	Classification	Large	Complex	Large	Unsuitable

Table 1. Suitability of each process for implementation on the embedded system

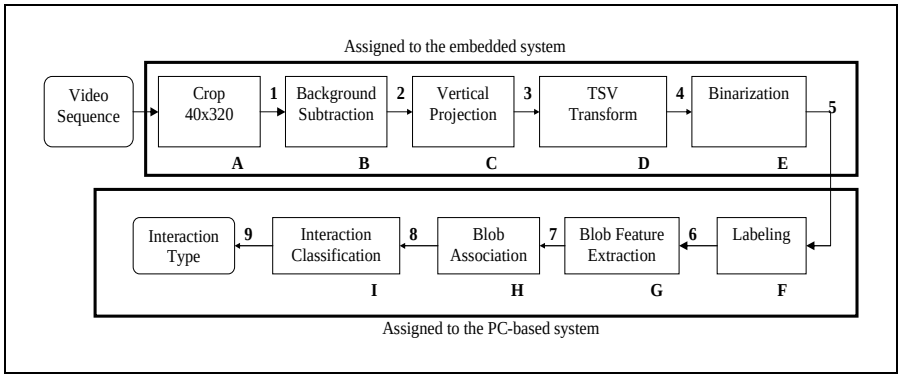

Figure 4. Division of tasks between the embedded and PC-based system

For the actual computation, we assumed that the system digitized a 30 [fps] monochrome video signal into a 320x240 image sequence. Also, we used R_V=30 [pixels] and V_V=40 [pixels]. Based on these considerations, we assigned stages A through E to the embedded system, 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 data transfer from the embedded system to the PC

3. Dataflow Modeling and Ptolemy Simulation

We determined the tasks to be assigned to embedded system in the previous section. In this section, we discuss the design and simulation of the embedded system using Ptolemy [10]. Figure 5 shows a block diagram of the embedded system on Ptolemy and Figure 6 gives the simulation result. Figure 5 follows the same steps as the block diagram discussed in Figure 1.

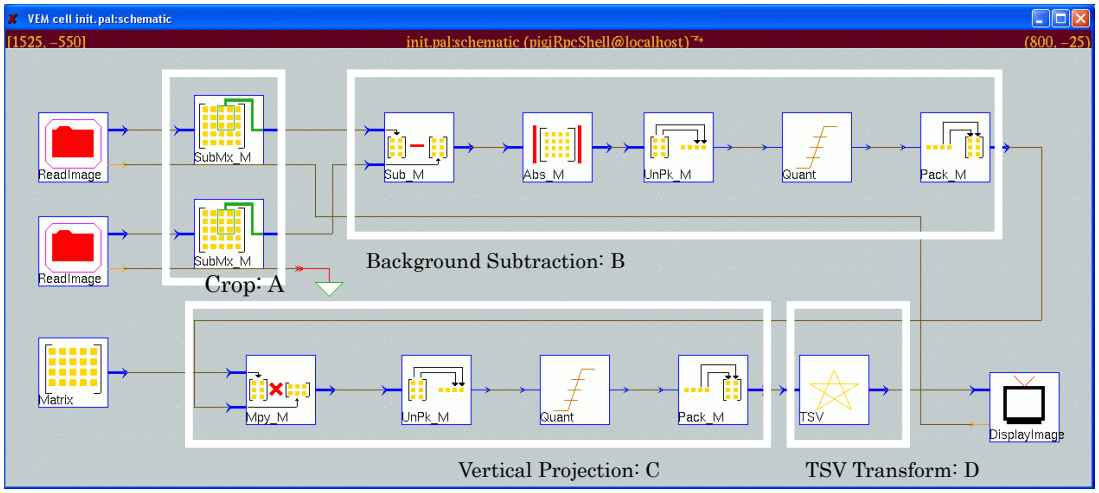

Figure 5. System design on Ptolemy

Figure 6. Results of the Ptolemy simulation

In Figure 6, 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 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 that the bright blobs are located in the same horizontal coordinates as the persons. We validated the Ptolemy simulation results by comparing them with the previous PC implementation, and the two sets of results were the same.

4. Hardware Design

With the specifications described in Table 2, we designed the hardware to perform the tasks that we simulated in Ptolemy. Figure 7 is a diagram of the embedded system. Figure 8 is a top view picture of the hardware.

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

Table 2. Hardware Specification

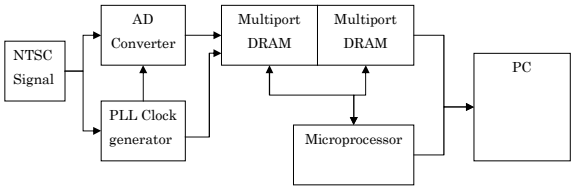

Figure 7. Diagram of the hardware

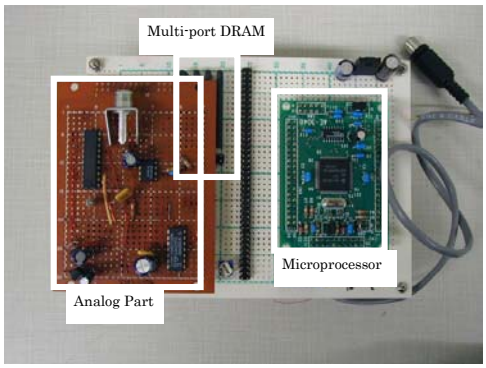

Figure 8. Picture of hardware design

Table 3 shows the results of the hardware simulation. This system processes one frame within 89 ms, which means that this system computes the data corresponding to one frame within three image frame cycles and performs 10 frames/s. 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 [ms] < 100 [ms] = 3 frame-cycle
Code size	5732 bytes
Internal RAM usage	854 bytes
Data rate to PC	128kbps

Table 3. Result of Hardware simulation

5. 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 huge 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.7, no.3, pp. 428-440, March, 1999.

[2] I. Haritaoglu, D. Harwood and L. Davis, "W4: Who, When, Where, What: A real time system for detecting and tracking people", *Proc. Int. Conf. on Automatic Face and Gesture, Nara*, pp 222-227, April 1998.

[3] I. Haritaoglu and L. Davis, "Hydra: Multiple people detection and tracking using silhouettes", *Proc. IEEE Work. on Visual Surveillance*, pp.6-13, Fort Collins, Colorado, June 1999.

[4] N. Oliver, B. Rosario and A. Pentland, "A Bayesian computer vision system for modeling human interactions", *Proc. Int. Conf. on Vision Systems*, Gran Canaria, Spain, pp. 255-272, January 1999.

[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*, 1999, pp 1090-1095

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

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

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

[10] Ptolemy Project, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, http://ptolemy.eecs.berkeley.edu/.