YIDDISH Vol 12 Issue 03,Sep 2022
ISSN NO: 0364-4308

Modifying an Intelligent Video Surveillance System with Multiple Processors

G.SUMAN[1],S.MUNI RAJA[2],
ASSISTANT PROFESSOR[1,2],
DEPARTMENT OF EEE
PBR VISVODAYA INSTITUTE OF TECHNOLOGY AND SCIENCE::KAVALI

Abstract

With the help of two specialized embedded RISC processors, the authors of this research report on the creation of a fully automated embedded video surveillance system. The software is broken up into many modules, such as those responsible for object tracking and video stream encoding. Using video footage captured by fixed cameras, the real-time object tracker can identify and follow relocating items in a scene. The block-matching algorithm serves as its foundation. Optimizing an ITU-T H.263 baseline video encoder for quarter common intermediate format (QCIF) and common intermediate format (CIF) resolutions is part of the process of encoding the video stream. The automated video surveillance system was realized by the integration of two separate subsystems, each using one of the two processing cores. The experimental results demonstrate the system's potential for real-time object detection, tracking, and encoding in pictures of QCIF and CIF resolutions. The system's low cycle count, low number of transistors, and low power consumption make it well-suited for use in outlying areas.

INTRODUCTION

Thanks to recent advancements in computer technology, fully automated video monitoring in real time is now a reality. The use of automated video surveillance allows for the monitoring of expansive spaces with complicated sceneries, increases the likelihood of detecting a particular occurrence, and simultaneously reduces the amount of data supplied to security personnel. A system like this would include a video compression part as well as an object detection/tracking part. Automated video surveillance systems begin with the identification of moving objects inside the field of view of a video camera, and the findings of this detection are then utilized for further processing, such as object tracking. Captured scenes may be stored more easily and sent over a less bandwidth thanks to video compression. The difference approach and block-matching algorithms are only two examples of the real-time tracking techniques used in [1]. To the best of our knowledge, no performance statistic is known for the hardware-implemented object tracker, despite the fact that a parallel hardware implementation of the tracking algorithm is described in [2].

The processing demands of video compression mean that it is still a difficult issue to solve, particularly for real-time embedded devices. Different strategies using hardware accelerators, parallel processing, and programmable CPUs have been suggested to fulfil these computing needs. There have been several reports of real-time H.263/MPEG-4 video

encoder implementations. Using several TMS320C6201 DSPs, the authors of [3] were able to encode CIF (352 288) images at a real-time frame rate of 30 fps. Due to both fine-grained and coarse-grained parallelism, H.263/MPEG-4 is well-suited for parallel processing. High encoding performance may be achieved with the help of hardware acceleration units, as stated in [4, 5], although hardware is less versatile and not well suited for rapid updates. Using the same RISC embedded processor as we use, recent work by [6] has accomplished real-time MPEG-4 video encoding of the 15 fps QCIF size pictures needing 65.7 Cycles. Our video encoder can encode CIF pictures at 15 frames per second with a lower cycle count than QCIF can, but this comes at the expense of more hardware. Because of its lower power requirements, it is more suited for low-power uses.

TELEVISION MONITORING THAT DOES ITSELF

In this part, we detail the video compression algorithms we utilized, the object identification and tracking methods we implemented, and the design choices that went into our surveillance system.

Detecting and following objects.

Both background subtraction (which compares each new frame with a representation of the scene backdrop) and block-matching algorithms (which compare each new frame with the previous frame) may be used to detect moving objects in video sequences. The next step is to use object tracking to follow the path of a moving item in a movie. backdrop model techniques avoid being fooled by static textures, but they can't pick up on foreground items that have a comparable intensity level with the backdrop. Background modelling strategies fall into either the no recursive or recursive camps [7]. In order to estimate the background picture based on the temporal fluctuation of each pixel in the buffer, no recursive methods must first buffer the previous N video frames. Buffering the frames takes up space in storage. Frame differencing, the median filter [8, 9], and the nonparametric background model [10] are all examples of popular no recursive methods. Since recursive methods iteratively update a single backdrop model based on each input frame, they eliminate the requirement to buffer a collection of frames for background estimate. Although recursive methods use less memory, input frames from the past may influence the present backdrop model. This means that any flaw in the base model will have more time to manifest itself. Recursive methods such as ap are often used.

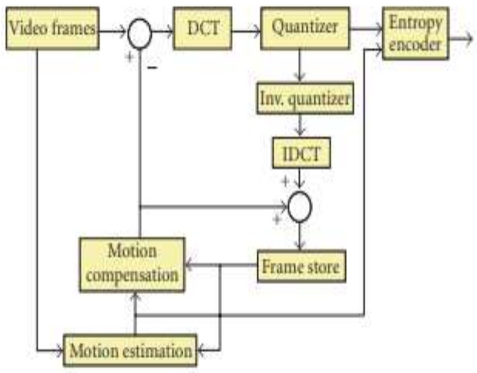


Figure 1: Schematic of the H.263 encoder.

proximate median filter [11], Kalman filter [12, 13], and adaptive mixture of Gaussians [14, 15]. Block-matching motion estimation algorithms widely used in video compression applications have also been used for moving object detection. The determined motion vectors can be used for object tracking by grouping or associating some of the motion vectors into a meaningful scene representation. Some real-time tracking methods based on blockmatching algorithms have been proposed [1, 16, 17].

The H.263 encoder overview

The H.263 video compression standard was defined by the ITU [18] for use in a range of low bit-rate video applications over wireless and public-switched telephone networks. A generic H.263/MPEG-4 encoder showing transform and predictive coding is depicted in Figure 1. The main operations that bring about compression include discrete cosine transform (DCT), inverse discrete cosine transform (IDCT), quantization (Q), inverse quantization (IQ), variable length coding (VLC), and motion estimation (ME). A compressed video sequence is made up of a series of INTRA- and INTER-frames. An INTRA-frame is used as a reference frame for INTER-frame encoding. It is coded using the DCT, Q, and VLC. INTER-frames are coded using the ME, and built-on INTRA-frames (or previous INTERframes) with motion vectors. Thus, an INTER-frame is not viewable on its own. In situations where the video sequence features a scene change, motion vectors will not generate a valid image. Thus, to improve error resilience, H.263 coding standard calls for at least one INTRA-coded frame every 132 frames [19] or when a scene change takes place.

MULTIPROCESSOR SYSTEM

The previous section described the implementation of the individual components that comprise the automated video surveillance application. In this section we discuss the integration of two heterogeneous processors, building of a simulator for the MPSoC system, synchronization mechanism adopted, and simulation of a system composed of these application-specific processors and their applications.

Extensamodelling protocol

The Extensa ISS is an instruction-cycle accurate instruction set simulation model which is appropriate for simulating and verifying the behaviour of a single Extensa processor connected to simple memories. The Extensamodelling protocol (XTMP) extends the ISS application programming interface (API) to allow for simulation of designs with multiple processors or custom hardware devices. XTMP models communication between cores and devices as transactions, not as signals, with a positive effect on the simulation speed and ease of development of the model, but it affects the accuracy of the developed model. The XTMP simulator runs faster than a hardware description language (HDL) simulator as the simulator and device models written in C do not need to model every signal transition for every gate and register.

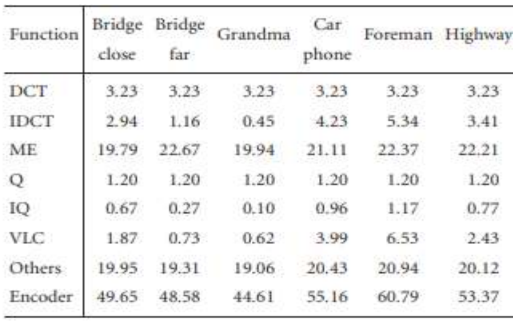
System memory map

The automated video surveillance application has two tasks that are executed on two processors, the object detection/tracking application runs on core 1 (called u1 s1) while the video encoding application runs on core 2 (u1 s2). A shared memory module of 64 kB was created as data is shared between the processors. When an object has been detected by the object tracking processor, the raw input pixels need to be shared with the video encoding processor. In the system a single global address space does not exist; instead, two separate memory maps are established and each processor has its own address space. Each processor has its own private system ROM and RAM, and the processors share a common memory module which appears at different addresses of individual processors. The processors and memory modules are connected via an intermediary object called a connector using the XTMP connect() function. The connector is connected to the cores via the cores' PIF, and allows multiple cores and multiple devices to be attached to it. It routes processor read/write transactions to memories and provides an address mapping capability. The XTMP multiAddressMapConnector has been used to define a processor-specific address space so that each processor can use the same address to access a different memory module as well as allowing each processor to use a different address to access a shared memory module

RESULTS AND DISCUSSION

The cache explorer has been used to analyse different possible cache configurations and determine which of them is optimal for the application. The H.263 video encoder core was configured with a direct-mapped, 16 KB instruction cache with cache line size of 64 bytes and a 2-way set associative 32 KB data cache with cache line size of 64 bytes. Cache configuration for the object tracker is not as crucial as the video encoder is the limiting factor in terms of performance. Therefore, the object tracker was configured with 8 KB direct mapped instruction cache with line size of 64 bytes, and 8 KB 2-way set associative data cache with line size of 64 bytes. The software optimizations and addition of video compression-specific TIE instructions to the customized processor core resulted in performance improvement of 41.2 times over the original TMN encoder version 1.7. This improvement has allowed real-time H.263 video encoding of

Table 1: Profile information of video encoder compressing QCIF video sequences with little (first three columns) and substantial (last three columns) motion (Cycles).



Function	Bridge close	Bridge far	Grandma	Car phone	Foreman	Highway
DCT	3.23	3.23	3.23	3.23	3.23	3.23
IDCT	2.94	1.16	0.45	4.23	5.34	3.41
ME	19.79	22.67	19.94	21.11	22.37	22.21
Q	1.20	1.20	1.20	1.20	1.20	1.20
IQ	0.67	0.27	0.10	0.96	1.17	0.77
VLC	1.87	0.73	0.62	3.99	6.53	2.43
Others	19.95	19.31	19.06	20.43	20.94	20.12
Encoder	49.65	48.58	44.61	55.16	60.79	53.37

15 fps QCIF and CIF size images requiring 49 and 205 million clock cycles, respectively. Table 4 shows the performance results for both the software optimized and TIE optimized encoders. The Extensa C and C++ compiler (XCC) provides better execution performance and smaller size of the compiled code when compared with the GCC compiler. Among the slowest operations in ANSI C are copying arrays of data. These are shown in Table 4 under the "Others" category. Further results which illustrate performance of the optimized processor for video encoding of some standard video sequences [22] are shown in Table 5, for QCIF video sequences, and in Table 6, for CIF video sequences. The results shown in Table 7 were obtained using the standalone Extensa ISS built for the object tracking processor. The video sequences with internal names kwbB, rheinhafen, taxi, bad, and dtneu-nebel were downloaded from [32] and resized to 352 × 288 using Virtual Dub version 1.65 [33]. Table 5 shows that 15 CIF resolution images can be processed in 128 million clock cycles for all video sequences tested,

Table 2: Profile information of video encoder compressing CIF resolution sequences (MCycles).

Function	Bridge close	Bridge far	Highway	Hall Monitor
DCT	12.95	12.96	12.95	12.95
IDCT	10.18	6.00	12.83	10.02
ME	82.23	98.67	95.01	82.78
Q	4.95	4.95	4.94	4.95
IQ	2.29	1.36	2.87	2.26
VLC	7.89	3.44	8.72	5.18
Others	77.56	75.96	78.51	77.44
Encoder	198.05	203.34	215.83	195.58

Table 3: Profile information of object tracker processing 15 CIF resolution frames (MCycles)

Function	kwbB	Rheinhafen	Taxi	Bad	Dtneu_nebel
Block matching	84.62	81.09	84.51	95.30	91.59
Connected component labeling	0.68	0.55	0.46	0.83	0.35
Output file generation (evaluation)	3.04	4.20	4.20	4.62	2.99
Finding motion within macroblocks	0.30	0.32	0.66	0.33	0.30
Others	26.88	26.26	26.61	26.56	27.11
Object tracker	115.52	112.42	116.44	127.64	122.34

significantly less than the number of clock cycles required by the video encoder to encode 15 CIF resolution frames. The final processor core configurations of both the video encoder and the object tracker are shown in Table 8. The 200 MHz processor core configured for object tracking has lower gate count and power dissipation compared to the 205 MHz processor core. This is due to the V1620-8 DSP configured for the video encoder processor core, which added approximately 75.000 to the gate count and 34 mW to power dissipation. The power consumption is an estimation provided by Xtensa Xplorer tool. It is based on the synthesis results of placed and routed units from library of components used in Xtensa processor. An accurate estimation of power consumption should be done after the physical synthesis of the processor core and running the target application. More TIE instructions also had to be added to the video encoder application to achieve real-time performance of 15 CIF frames per second. The video encoding core has been configured with larger instruction and data caches, thus explaining the big difference in area (including caches/local memories).

CONCLUSIONS

Using the Xtensa platform, a reactive real-time automated visual surveillance application was shown. Two processors are used to run the application's object tracking and video stream encoding subsystems, respectively. To achieve real-time QCIF and CIF encoding, the video stream encoding subsystem was implemented by optimizing a software H.263 video encoder running on the Xtensa flexible and extensible embedded CPU. The experimental findings reveal that the software and Xtensa-specific improvements result in 7.1 and 41.2 times the performance boost over the baseline, respectively.The blockmatching-based object tracker was successfully tested for a variety of items utilizing stationary camera video sequences. Some TIE instructions originally developed for the video compression application were repurposed to enable real-time tracking of in-motion objects. In the test video sequences, the object tracker takes 130 million clock cycles to analyze 15 CIF quality frames. The dual FIFO and shared memory between the two Xtensa cores allowed for the implementation of the automated video surveillance program. Tensilica's instruction-set simulator API was used to create a multiprocessor simulator software for simulating the automated video surveillance application on the system-on-chip architecture.

REFERENCES

[1] S. A. El-Azim, I. Ismail, and H. A. El-Latiff, "An efficient object tracking technique using block-matching algorithm," in Proceedings of the 9th National Radio Science Conference (NRSC '02), pp. 427–433, Alexandria, Egypt, March 2002.

[2] I. I. Mahmoud, H. A. Abd El-Halym, and S. E.-D. Habib, "Hardware development and implementation of an object tracking algorithm," in Proceedings of the 15th International Conference on Microelectronics (ICM '03), pp. 330–334, Cairo, Egypt, December 2003.

[3] O. Lehtoranta, T. Ham¨ al¨ ainen, and J. Saarinen, "Parallel im- ¨ plementation of H.263 encoder for CIF-sized images on quad DSP system," in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '01), vol. 2, pp. 209–212, Sydney, Australia, May 2001.

[4] Y.-C. Chang, W.-M. Chao, and L.-G. Chen, "LSI design for MPEG-4 coding system," in Proceedings of 47th Midwest Symposium on Circuits and Systems (MWSCAS '04), vol. 2, pp. 453– 456, Hiroshima, Japan, July 2004.

[5] M. J. Garrido, C. Sanz, M. Jimenez, and J. M. Meneses, "A flex- ´ ible architecture for H.263 video coding," Journal of Systems Architecture, vol. 49, no. 12–15, pp. 641–661, 2003.

[6] J.-G. Kim and C.-C. Jay Kuo, "MPEG-4 video codec IP design with a configurable embedded processor," in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '03), vol. 2, pp. 776–779, Bangkok, Thailand, May 2003.

[7] S.-C. S. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in Visual Communications and Image Processing, vol. 5308 of Proceedings of SPIE, pp. 881–892, San Jose, Calif, USA, January 2004.

[8] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 10, pp. 1337–1342, 2003.

[9] R. Cutler and L. S. Davis, "View-based detection and analysis of periodic motion," in Proceedings of the 14th International Conference on Pattern Recognition (ICPR '98), vol. 1, pp. 495–500, IEEE Computer Society Press, Brisbane, Queensland, Australia, August 1998.

[10] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," Proceedings of the IEEE, vol. 90, no. 7, pp. 1151–1163, 2002.

[11] N. J. B. McFarlane and C. P. Schofield, "Segmentation and tracking of piglets in images," Machine Vision and Applications, vol. 8, no. 3, pp. 187–193, 1995.

[12] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: real-time tracking of the human body," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 780–785, 1997.

[13] J. Heikkila and O. Silven, "A real-time system for monitoring of cyclists and pedestrians," in Proceedings of the 2nd IEEE Workshop on Visual Surveillance (VS '99), pp. 74–81, Collins, Colo, USA, July 1999.

[14] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 747–757, 2000.

[15] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '99), vol. 2, pp. 246–252, Fort Collins, Colo, USA, June 1999.

[16] K. Hariharakrishnan, D. Schonfeld, P. Raffy, and F. Yassa, "Video tracking using block matching," in Proceedings of IEEE International Conference on Image Processing (ICIP '03), vol. 3, pp. 945–948, Barcelona, Spain, September 2003.

[17] K. Hariharakrishnan, D. Schonfeld, P. Raffy, and F. Yassa, "Object tracking using adaptive block matching," in Proceedings of the International Conference on Multimedia and Expo (ICME '03), vol. 3, pp. 65–68, Baltimore, Md, USA, July 2003.

[18] ITU-T experts group on very bitrate visual telephony, ITU-T Recommendation H.263 version 2: video coding for low bitrate communication, January 1998.

[19] N. Yu, K. Kim, and Z. Salcic, "A new motion estimation algorithm for mobile real-time video and its FPGA implementation," in Proceedings of IEEE Region 10 International Conference.