ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Special Issue 6, May 2015

Smart Cameras in Embedded Systems

N.Mohanapriya [1], R.Saranya[2], and T.Kowsalya[3]

PG Scholars, Muthayammal Engineering College, Rasipuram, Namakkal, Tamilnadu, India[1,2]

Professor Head, Muthayammal Engineering College, Rasipuram, Namakkal, Tamilnadu, India[3]

ABSTRACT: A smart camera performs real-time analysis to recognize scenic elements. Smart cameras are useful in a variety of scenarios: surveillance, medicine, etc. We have built a real-time system for recognizing gestures. Our smart camera uses novel algorithms to recognize gestures based on low-level analysis of body parts as well as hidden Markov models for the moves that comprise the gestures. These algorithms run on a Trimedia processor. Our system can recognize gestures at the rate of 20 frames/second. The camera can also fuse the results of multiple cameras.

I. OVERVIEW

Recent technological advances are enabling a new generation of smart cameras that represent a quantum leap in sophistication. While today's digital cameras capture images, smart cameras capture high-level descriptions of the scene and analyze what they see. These devices could support a wide variety of applications including human and animal detection, surveillance, motion analysis, and facial identification.Video processing has an insatiable demand for real-time performance. Fortunately, Moore's law provides an increasing pool of available computing power to apply to real-time analysis. Smart cameras leverage very large-scale integration (VLSI) to provide such analysis in a low-cost, low-power system with substantial memory. Moving well beyond pixel processing and compression, these systems run a wide range of algorithms to extract meaning from streaming video.

Because they push the design space in so many dimensions, smart cameras are a leading-edge application for embedded system research.

II. DETECTION AND RECOGNITION ALGORITHMS

Although there are many approaches to real-time video analysis, we chose to focus initially on human gesture recognition—identifying whether a subject is walking, standing, waving his arms, and so on. Because much work remains to be done on this problem, we sought to design an embedded system that can incorporate future algorithms as well as use those we created exclusively for this application.

Our algorithms use both low-level and high-level processing. The low-level component identifies different body parts and categorizes their movement in simple terms. The high-level component, which is application-dependent, uses this information to recognize each body part's action and the person's overall activity based on scenario parameters.

Human detection and activity/gesture recognition algorithm has two major parts: Low-level processing (blue blocks in Figure 1) and high-level processing (green blocks in Figure 1).

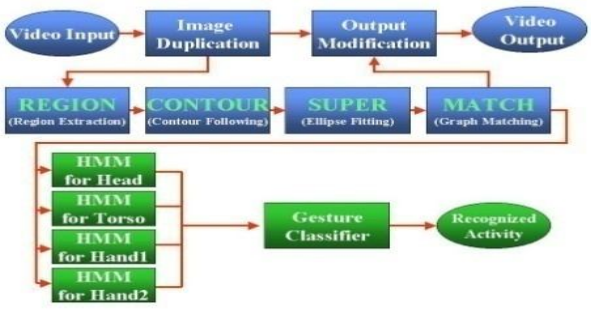


Figure 1: Algorithm.

ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Special Issue 6, May 2015

A) Low-level processing

The system captures images from the video input, which can be either uncompressed or compressed (MPEG and motion JPEG), and applies four different algorithms to detect and identify human body parts.

Region extraction: The first algorithm transforms the pixels of an image like that shown in Figure 2.a, into an M ¥ N bitmap and eliminates the background. It then detects the body part's skin area using a YUV color model with chrominance values down sampled

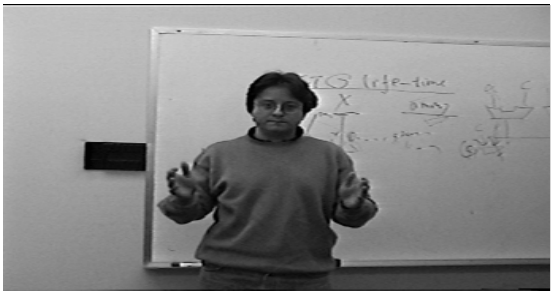


Figure 2a

Next, as Figure 2b illustrates, the algorithm hierarchically segments the frame into skin-tone and non-skin-tone regions by extracting foreground regions adjacent to detected skin areas and combining these segments in a meaningful way.

Figure 2.b

Contour following: The next step in the process, shown in Figure 2c, involves linking the separate groups of pixels into contours that geometrically define the regions. This algorithm uses a 3 ¥ 3 filter to follow the edge of the component in any of eight different directions.

Figure 2c

ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Special Issue 6, May 2015

Ellipse fitting: To correct for deformations in image processing caused by clothing, objects in the frame, or some body parts blocking others, an algorithm fits ellipses to the pixel regions as Figure 2d shows to provide simplified part attributes. The algorithm uses these parametric surface approximations to compute geometric descriptors for segments such as area, compactness (circularity), weak perspective invariants, and spatial relationships.

Figure 2d

Graph matching: Each extracted region modeled with ellipses corresponds to a node in a graphical representation of the human body. A piecewise quadratic Bayesian classifier uses the ellipses parameters to compute feature vectors consisting of binary and unary attributes. It then matches these attributes to feature vectors of body parts or meaningful combinations of parts that are computed offline. To expedite the branching process, the algorithm begins with the face, which is generally easiest to detect.

B) High-level processing

The high-level processing component, which can be adapted to different applications, compares the motion pattern of each body part—described as a spatiotemporal sequence of feature vectors—in a set of frames to the patterns of known postures and gestures and then uses several hidden Markov models in parallel to evaluate the body's overall activity. We use discrete HMMs that can generate eight directional code words that check the up, down, left, right, and circular movement of each body part.

Human actions often involve a complex series of movements. We therefore combine each body part's motion pattern with the one immediately following it to generate a new pattern. Using dynamic programming, we calculate the probabilities for the original and combined patterns to identify what the person is doing. Gaps between gestures help indicate the beginning and end of discrete actions.

A quadratic Mahalanobis distance classifier combines HMM output with different weights to generate reference models for various gestures. For example, a pointing gesture could be recognized as a command to "go to the next slide" in a smart meeting room or "open the window" in a smart car, whereas a smart security camera might interpret the gesture as suspicious or threatening.

To help compensate for occlusion and other image-processing problems, we use two cameras set at a 90-degree angle to each other to capture the best view of the face and other key body parts. We can use high-level information acquired through one view to switch cameras to activate the recognition algorithms using the second camera. Certain actions, such as turning to face another direction or executing a predefined gesture, can also trigger the system to change views Soft-tissue reconstruction.

We can use ***Mat Lab*** to develop our algorithms. This technical computation and visualization programming environment runs orders of magnitude more slowly than embedded platform implementations, a speed difference that becomes critical when processing video in real time. We can therefore port our ***Mat Lab*** implementation to C code

ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Special Issue 6, May 2015

running on a very long instruction word (VLIW) video processor, which lets us make many architectural measurements on the application and make the necessary optimizations to architect a custom VLSI smart camera.

III. REQUIREMENTS

At the development stage, we can evaluate the algorithms according to accuracy and other familiar standards. However, an embedded system has additional real-time requirements:

Frame rate: The system must process a certain amount of frames per second to properly analyze motion and provide useful results. The algorithms we use as well as the platform's computational power determine the achievable frame rate, which can be extremely high in some systems.

Latency: The amount of time it takes to produce a result for a frame is also important because smart cameras will likely be used in closed-loop control systems, where high latency makes it difficult to initiate events in a timely fashion based on action in the video field.

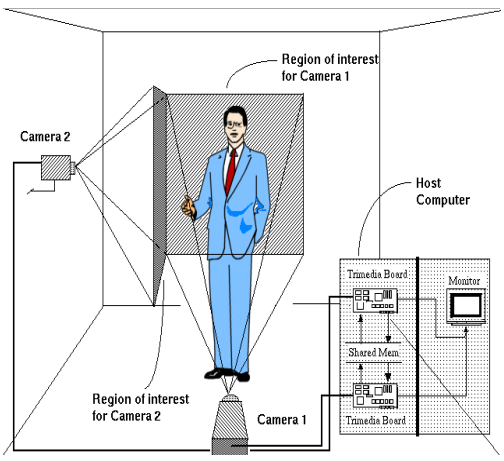
Moving to an embedded platform also meant that we have to conserve memory. Looking ahead to highly integrated smart cameras, we want to incorporate as little memory in the system as possible to save on both chip area and power consumption. Gratuitous use of memory also often points to inefficient implementation.

IV. COMPONENTS

Our development strategy calls for leveraging off-the-shelf components to process video from a standard source in real time, debug algorithms and programs, and connecting multiple smart cameras in a networked system. We use the 100-MHz Philips TriMedia TM-1300 as our video processor. This 32-bit fixed- and floating-point processor features a dedicated image coprocessor, a variable length decoder, an optimizing C/C++ compiler, integrated peripherals for VLIW concurrent real-time input/output, and a rich set of application library functions including MPEG, motion JPEG, and 2D text and graphics.

V. TEST BED ARCHITECTURE

Our test bed architecture, shown in Figure 3, uses two TriMedia boards attached to a host PC for programming support. Each PCI bus board is connected to a Hi8 camera that provides NTSC composite video. Several boards can be plugged into a single computer for simultaneous video operations. The shared memory interface offers higher performance than the networks likely to be used in VLSI cameras, but they let us functionally implement and debug multiple-camera systems with real video data.



ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Special Issue 6, May 2015

VI. EXPERIMENTS AND OPTIMIZATIONS

As data representation becomes more abstract, input/output data volume decreases. The change in required memory size, however, is less predictable given the complex relationships that can form between abstract data. For example, using six single-precision, floating-point parameters to describe 100 ellipses requires only 2.4 Kbytes of memory, but it takes 10 Kbytes to store information about two adjoining ellipses.

Based on these early experiments, we optimize our smart camera implementation by applying techniques to speed up video operations such as substituting new algorithms better suited to real-time processing and using TriMedia library routines to replace C-level code.

VII. ALGORITHMIC CHANGES

We originally fit super ellipses (generalized ellipses) to contour points, and this was the most time-consuming step. Rather than trying to optimize the code, we decided to use a different algorithm. By replacing the original method developed from principal component analysis with moment-based initialization, we reduced the Levenberg-Marquardt fitting procedure, thus decreasing the execution time.

After converting the original Mat lab implementation into C, we performed some experiments to gauge the smart camera system's effectiveness and evaluate bottlenecks. The un optimized code took, on average, 20.4 million cycles to process one input frame, equal to a rate of 5 frames per second.

We first measure the CPU times of each low-level processing step to determine where the cycles were being spent. Microsoft Visual C++ is more suitable for this purpose than the TriMedia compiler because it can collect the running time of each function as well as its sub functions' times.

Figure 4a shows the processing-time distribution of the four body-part-detection algorithms Figure 4b shows the memory characteristics of each low-level processing stage.

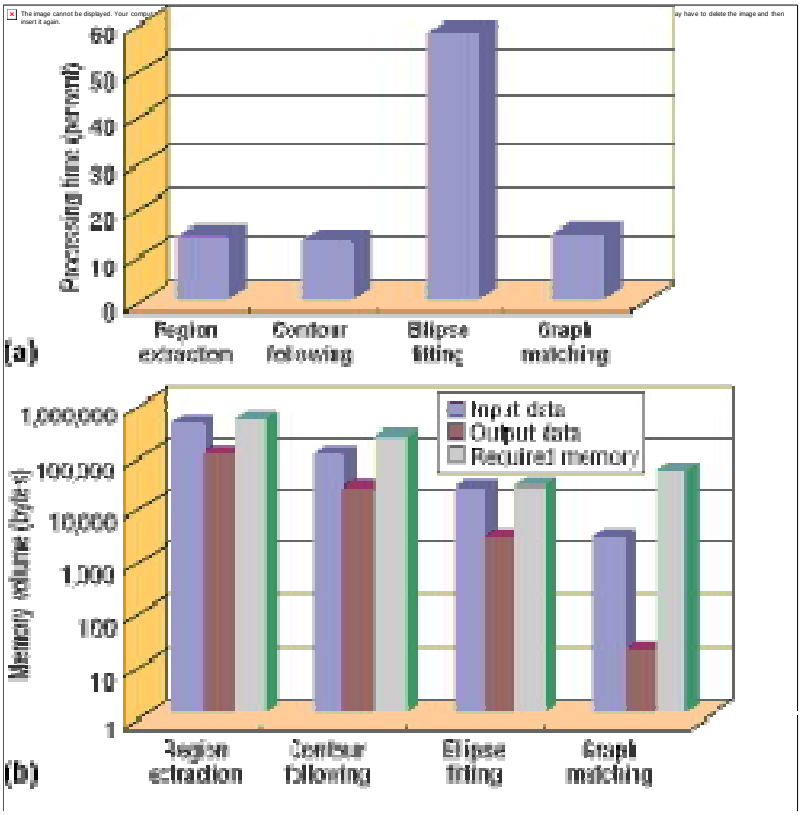


Figure 4.a, 4.b

ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Special Issue 6, May 2015

VIII. CONTROL –TO –DATA TRANSFORMATION

Increasing the processor's issue width can exploit the high degree of parallelism that region extraction offers. Using a processor with more functional units could thus reduce processing time during this stage. However, contour following, which converts pixels to abstract forms such as lines and ellipses, consumes even more time. The algorithm also operates serially: It finds a region's boundary by looking at a small window of pixels and sequentially moving around the contour; at each clockwise step it must evaluate where to locate the contour's next pixel. While this approach is correct and intuitive, it provides limited ILP.

We evaluate all possible directions in parallel and combined the true/false results into a byte, which serves as an index to look up the boundary pixel in a table. We also manipulate the algorithm's control-flow structure to further increase ILP. These optimizations double the contour-following stage's running speed

IX. OPTIMIZATION RESULTS AND CONCLUSION

The combination of these methods radically improves CPU performance for the application. Optimization boosts the program's frame rate from 5 to 31 frames per second. In addition, latency decreases from about 340 to 40-60 milliseconds per frame. We can add HMMs and other high-level processing parts, and that makes the program now runs at about 25 frames per second.

Our board-level system is a critical first step in the design of a highly integrated smart camera. Although the current system is directly useful for some applications, including security and medicine, a VLSI system will enable the development of high-volume, embedded computing products.

Because the digital processors and memory use advanced small-feature fabrication and the sensor requires relatively large pixels to efficiently collect light, it makes sense to design the system as two chips and house them in a multichip module. Separating the sensor and the processor also makes sense at the architectural level given the well-understood and simple interface between the sensor and the computation engine.

The advantages of leveraging existing sensor technology far outweigh any benefits of using pixel-plane processors until they become more plentiful. However, attaching special-purpose SIMD processors to the multiprocessor can be useful for boundary analysis and other operations. Such accelerators can also save power, which is important given the cost and effort required to deploy multiple cameras, especially in an outdoor setting. High-frame-rate cameras, which are useful for applications ranging from vibration analysis to machinery design, will likely require many specialized processing elements that are fast as well as area efficient.

REFERENCES

1. T.Kowsalya and Dr.S.Palaniswami (2014)'A Clock Control Strategy Based clustering Method For Peak Power And Rms Current Reduction' in Journal of Theoretical and Applied Information Technology Vol. 63 No.2 © 2005 - 2014 JATIT & LLSISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195 459
2. T.Kowsalya and Dr.S.Palaniswami(2012)' Decoupled SRAM Cell with Bit Line Decoupled Current Mode Sense Amplifier' Published in European journal of Scientific Research in volume 84 issue 2 Aug 2012
3. .J.M. Siskind, "Visual Event Classification via Force Dynamics," *Proc. 17th Nat',l Conf. Artificial Intelligence* and *12th Conf. Innovative Applications of Artificial Intelligence* (AAAI/IAAI 00), AAAI Press/MIT Press, Menlo Park, Calif., 2000, pp. 149-155.
4. K.Gopi and Mrs.T.Kowsalya" "A Direct Injection-locked QPSK Modulator based on ring VCO" published in International Journal of Innovative Research in Computer".Dec 2014
5. M. Lucente, G.-J. Zwart and A.D. George, "Visualization Space: A Testbed for Deviceless Multimodal User Interface," *Computer Graphics*, vol. 31, no. 2, 1997,
6. T.B. Moeslund and E. Granum, "A Survey of Computer Vision-Based Human Motion Capture," *Computer Vision and Image Understanding*, vol. 81, no. 3, 2001, pp. 231-268.
7. M. Nicolescu and G.G. Medioni, "Electronic Pan-Tilt-Zoom: A Solution for Intelligent Room Systems," *Proc. IEEE Int',l Conf. Multimedia and Expo* (ICME 00), IEEE CS Press, Los Alamitos, Calif., 2000, pp. 1581-1584.
8. J. Foote and D. Kimber, "FlyCam: Practical Panoramic Video and Automatic Camera Control," *Proc. IEEE Int',l Conf. Multimedia and Expo* (ICME 00), IEEE CS Press, Los Alamitos, Calif., 2000, pp. 1419-1422.
9. Pentland and T. Choudhury, "Face Recognition for Smart Environments," *Computer*, Feb. 2000, pp.50-55

ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Special Issue 6, May 2015

10. A.D. Wilson and A.F. Bobick, "Realtime Online Adaptive Gesture Recognition," *Proc. 15th Int',l Conf. Pattern Recognition* (ICPR 00), IEEE CS Press, Los Alamitos, Calif., 2000, pp.270-275.
11. Pentland, "Smart Rooms, Smart Clothes," *Proc. 14th Int',l Conf. Pattern Recognition* (ICPR 98), IEEE CS Press, Los Alamitos, Calif., 1998, pp.949-953.
12. Pentland, "Looking at People: Sensing for Ubiquitous and Wearable Computing," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, 2000,pp.107-119.
13. S. Stillman and I. Essa, "Towards Reliable Multimodal Sensing in Aware Environments," *Proc. Workshop on Perceptive User Interfaces* (PUI 01), ACM Press, New York, 2001,
14. S.M. Chai et al., "Focal Plane Processing Architectures for Real-Time Hyperspectral Image Processing," *J. Applied Optics: Special Issue on Information Processing*, vol.39, no.5, 2000,pp.835-849.
15. J. Watlington and V.M. Bove Jr., "A System for Parallel Media Processing," *Parallel Computing*, vol. 23, no. 12, 1997, pp.1793-1809