

Resource-Aware Video Processing Techniques for Ambient Multimedia Systems

Nicholas H. Zamora, Xiaoping Hu, Umit Ogras, and Radu Marculescu

Department of Electrical and Computer Engineering

Carnegie Mellon University

Pittsburgh, PA 15213-3890

{nhz, xh, uogras, radum} @ ece.cmu.edu

ABSTRACT

Ambient Intelligence (AmI) is the inconspicuous presence of computing into every facet of our lives. AmI systems of the future will contain devices with highly limited resources in terms of processing power, memory, and battery lifetime. Contrary to this are the memory, power, and cycle-hungry video processing applications which are required to provide the level of utility demanded by the users. This work introduces the idea of processing a portion of a video frame with the intent of achieving high levels of video processing performance while reducing the hardware requirements for that processing considerably. The techniques we propose operate only on a portion of the video frame and optionally adjust that portion's size dynamically to reasonably match the video content. Our results show that this technique can save roughly 75% in both memory and processing cycle requirements and up to 87% in energy consumption while inducing less than 10% error in XY processing of the video frame.

INTRODUCTION

We boldly, yet not rashly, claim that ubiquitous computing is the next truly transforming technology of our time [1, 2]. Engineers are currently attempting to realize these abilities by building systems consisting of dozens, hundreds, and eventually even thousands of tiny computing devices, sensors, and actuators, all connected through a dynamic and complex communication network. These systems will unobtrusively be present in our daily lives, and will require unique user interfaces to report environmental conditions and receive user commands.

1.1 Motivation

Cost and form factor (size) will be key variables in determining usability of AmI systems. As a result, the computational devices employed in these systems will have limited memory sizes, tight energy constraints, and modest processing capabilities. Most current DSP and microcontroller chips have memory ranges of 8KB up to 4MB, clock speeds of 5MHz up to 720 MHz, and have active power consumption rates between 1mW and 1W. The high performance DSPs have more memory and speed resources, however with power consumption rates on the order of 1W, these chips will require bulky batteries and this is definitely not acceptable for AmI systems. Still, we envision these systems to incorporate real-time video processing applications.

From a few kilobytes to several megabytes and between a few MIPS to several hundred MIPS, real-time video processing demands ample data storage and provides challenging hard and/or soft real-time deadlines for the system to meet. Therefore, new resource-aware techniques for real-time video processing are required to realize these applications in light of severe device resource limitations. Introducing a new set of resource-aware techniques for real-time video processing which results in desirable levels of performance is the primary objective of this paper.

One application of ambient intelligent systems is video-based object tracking. Object tracking is used in a multitude of applications including surveillance, teleconferencing, 3D positioning, and even virtual reality applications. For surveillance

applications, although the public will not be aware of the presence of these tiny computers behind the walls around us, the computers should be aware of what is moving around them. One method of tracking objects in the environment is by capturing video and performing motion estimation techniques on that video.

1.2 Related Work

Several techniques for video-based object tracking have already been studied [3, 4]. Hamamoto et. al. [5] have proposed an FPGA implementation of an object tracking technique to track multiple objects in a scene. Huang et. al. [6] have proposed an object tracking system used for both encoding applications and for surveillance systems. Sun et. al. [7] have used object tracking techniques to identify a portion of a frame taken from a panoramic camera, and the authors use this information to encode part of a video frame for transmission to remote viewers.

Although new techniques in video encoding, such as in MPEG-4, employ object-recognition in order to encode parts of the frame that have changed between consecutive frames, these techniques are fundamentally different from the algorithms proposed in this paper. The algorithms proposed herein consider only a *portion* of the video frame captured by the video device, and is a *best-effort* technique to perform video-based object tracking. In contrast to our approach, none of these studies considers limited resources such as memory, processing power, or battery lifetime. In the same spirit, the Sony EVI-D30 camera [8] provides object tracking, but is often criticized as not being a robust solution. The cameras we envision should provide similar functionality at only a fraction of the cost, weight, size, power consumption, and complexity of this device.

1.3 Contribution and Paper Organization

Our techniques, which are based on *region-of-interest* (ROI) processing, are novel in that the primary design objective is for memory, processing, and energy-limited devices. Our approach analyzes only a portion of each video frame to significantly reduce the hardware resource requirements while maintaining a good quality of results of the object tracking algorithm. To the best of our knowledge, the techniques we propose are the first viable option for highly resource-constrained devices to perform video object tracking accurately. The algorithms we propose are illustrated on a single processor system, although the benefits we show can be achieved by applying them to a multi-processor object tracking system as well.

The next section gives an overview of two video applications relevant to our new video processing techniques. Section 3 describes the techniques we are proposing. Section 4 shows our results and the accuracy we maintain in the video application, using these techniques, and Section 5 concludes the work.

2. DRIVER APPLICATIONS

We describe next example applications for the video processing optimization techniques we propose in this work.

2.1 Video Teleconferencing

Video teleconferencing is a common application for object tracking techniques. In order to meet the bandwidth requirements

and to guarantee a desirable image quality, efficient encoding is usually achieved by using object-based encoding [9]. The video processing techniques we propose here can be used with a motor controlling the tilt angle of the camera's focus to accurately track an object through a scene.

The system is presented in Fig. 1. In this system, the camera captures video data in its focus area, and sends it to both the video encoding software and the object tracking software, which may run on different computing devices. The object tracking software tracks the objects in the video, and sends data to the motor controller which then, in turn, activates a motor which changes the focus area of the camera. The goal is to steer the focus area to accurately follow moving objects in the environment.

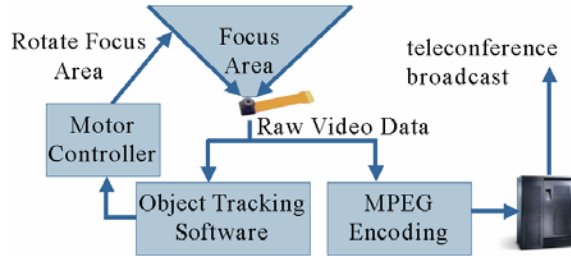


Figure 1: Diagram of a smart teleconferencing application, where the camera direction is controlled by an attached motor. This work focuses on the object-tracking software application.

2.2 Video Surveillance

Surveillance systems are widely deployed in banks, offices, airports, and highways for security and safety by employing a number of image sensors or cameras. However, the number of cameras used and regions covered is limited due to the costly video equipment currently on the market. The AmI systems of the future will have dozens and even hundreds of networked cameras constantly monitoring and analyzing scenes from different angles and positions. The processing devices attached to each camera should be small, cheap, and simple in order to keep the system unobtrusive and cheap. Each camera can be fixed and aimed in a particular direction all the time, or can watch a larger field using a controlling motor. Our work focuses on the setup with a motor to control the direction (tilt) of the video camera (Fig. 1).

3. REGION OF INTEREST: A NEW RESOURCE-AWARE VIDEO PROCESSING TECHNIQUE

Given a video stream, there are two general techniques a designer can use to fit video processing onto a relatively cheap, small, and simple microcontroller. The first category consists of *lossless techniques*, where the quality of the results is not diminished. The other set of techniques are *lossy techniques*, such as video sub-sampling in both time and space, pixel quantization or a disregard of a portion of the precision of the data. The techniques we propose are new lossy techniques for real-time video processing, and we title them *Region Of Interest (ROI) processing*.

3.1 ROI Processing

Large memory is usually an assumed requirement for video applications. Consider a CIF frame in NTSC, which is the size of 352x240 pixels. Using 1 byte to store one luminance value for each pixel, more than 84 Kbytes are needed to store a single frame. With 24-bit color, the memory requirement jumps to 253 Kbytes. In order to avoid storing so much video data, we propose ROI processing. This technique defines a region of each video frame in which useful processing will likely occur. Outside the region of interest, the video data is assumed to be statically constant (Fig. 2).

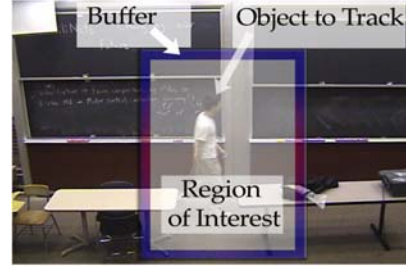


Figure 2: Region of interest video data processing.

The buffer area, shown in the figure, allows the ROI to change position between frames. The ROI can remain of *fixed size*, or can dynamically adjust its size to fit the properties of the video it is processing. When the ROI is fixed in size, the main goal of the processing is to identify the *movement of the dominant object* in the frame and update the center of the ROI so that it successfully follows the object around the scene. When the ROI is of *variable size*, successful object tracking is still important. In addition, however, the ROI can be reduced to a smaller size to further save the amount of memory and processing required in exchange for a slight degradation of quality of the object tracking results.

The combination of the size of this frame together with other lossy processing techniques should be chosen to match both the memory and speed capabilities of a chosen microcontroller. This technique accurately tracks moving objects when the size of those objects is a fraction of the size of the ROI, however our results show effective object tracking even when the moving objects are bigger than the ROI window.

3.2 Fixed-Size ROI Algorithm

The ROI algorithm for a fixed-size ROI can be described precisely as follows:

$$\begin{aligned}
 \text{prevROI}_x &\leftarrow \text{FrameWidth} / 2 & (1) \\
 \text{prevROI}_y &\leftarrow \text{FrameHeight} / 2 \\
 \text{currROI}_x &\leftarrow \text{FrameWidth} / 2 \\
 \text{currROI}_y &\leftarrow \text{FrameHeight} / 2 \\
 \text{prevROI} &\leftarrow \text{ROI of first video frame} & (2) \\
 \text{while (true)} & & (3) \\
 \text{currROI} &\leftarrow \text{ROI portion of next frame} \\
 [\text{obj}_x, \text{obj}_y, v_x, v_y, \text{reset}] = & \text{FindObject}(\text{prevROI}, \text{currROI}, \\
 & \text{prevROI}_x, \text{prevROI}_y, \text{currROI}_x, \text{currROI}_y) & (4) \\
 \text{prevROI} &\leftarrow \text{currROI} \\
 \text{prevROI}_x &\leftarrow \text{currROI}_x \\
 \text{prevROI}_y &\leftarrow \text{currROI}_y \\
 \text{currROI}_x &\leftarrow \text{obj}_x + v_x \bullet t_{\text{frame}} & (5) \\
 \text{currROI}_y &\leftarrow \text{obj}_y + v_y \bullet t_{\text{frame}}
 \end{aligned}$$

The variables in this algorithm are defined in Table 1. At the beginning of the algorithm, the ROI positions itself in the center of the frame (1). Then, the first frame's ROI data is stored (2). Then, (3) defines the core loop for the video processing. (4) is the technique used in conjunction with ROI optimization. The object tracking algorithm returns a position and velocity of the moving object detected, if any, and might optionally reset the ROI if no object has been detected for several frames. The ROI center point is updated according to the projected position of the moving object in (5), and the loop repeats.

The object tracking algorithm we use (4) detects a moving object using the following 3 major steps:

Variable	Description
FrameWidth	Total width of the video frame
FrameHeight	Total height of the video frame
prevROI _x	Center horizontal position of the previous ROI
prevROI _y	Center vertical position of the previous ROI
currROI _x	Center horizontal position of the current ROI
currROI _y	Center vertical position of the current ROI
prevROI	Previous frame's video data within ROI
currROI	Current frame's video data within ROI
obj _x	Center horizontal position of the moving object
obj _y	Center vertical position of the moving object
v _x	Horizontal velocity of the moving object
v _y	Vertical velocity of the moving object
t _{frame}	Time between consecutive frames

Table1: Variable descriptions for ROI fixed-size algorithm

- First, the ROI is broken into 8x8-pixel blocks; we compare these blocks in a pixel by pixel manner with the previous frame to determine which blocks contain movement.
- Second, the blocks are grouped into objects by grouping those moving blocks which neighbor each other. The dominant object is chosen to be the one with the highest number of moving blocks.
- Finally, to determine the direction of the dominant object's movement, we perform a 3-step search on each block. The position of the object is the average position of each block which comprises the object, and the velocity of the object is the average velocity of each block resulting from the 3-step search.

Additional steps can be added to make the object tracking more accurate, or the 3 steps above can be changed to customize the overall application for different video types and/or hardware limitations.

After the motion detection completes, the ROI position can be appropriately updated (step (5)). Now it should be clear that the buffer region in Fig. 2 is necessary since it allows the ROI to move between consecutive frames and contain the moving object throughout the video. A few details, such as making sure the next computed ROI position is within the bounds of the buffer region, are not shown in this fixed-size ROI algorithm for simplicity.

3.3 Dynamically-Adjusting ROI Algorithm

The ROI is not required to remain of fixed size. If the design goals are to minimize memory, processing cycles, and power consumed, then an ROI which adjusts its size to suit the object characteristics of the video frame can help achieve these goals. Here we show the basic algorithm for implementing a dynamic ROI in the form of a set of rules to govern that algorithm:

1. Expand or contract borders of ROI according to fraction of blocks "near" that border which are moving and are part of the object.

% blocks near border	0%	< 5%	< 15%	< 25%	> 25%
Action taken	E full	E half	DM	C half	C full

2. Expand borders based on the velocity (in pixels) of the moving object. Expand the top or left borders when the velocity is negative, and the right or bottom borders when the velocity is positive.

Object Velocity in pixels	< -6	[-6, -2]	[-2, 2]	[2, 6]	> 6
Action taken	E full	E half	DM	E half	E full

3. Sparsity rule: expand the border (E full) if the number of moving blocks is < 10% of the total blocks in the frame and more than half of them are "near" one particular border.

These rules use these shorthands:

E full	Expand that border the entire buffer region
E half	Expand that border half the buffer region
DM	Do not move the border
C half	Contract that border half the buffer region
C full	Contract that border the full buffer region
"near"	Within 3 blocks of the edge of the ROI

Each border is expanded or contracted by applying these rules for each frame. The size of the ROI is constrained to a maximum budgeted number of blocks. Also, the ROI is restricted to be at least a minimum of 5x5 blocks in size for each frame in order to prevent the ROI from shrinking to nothing when there are no moving objects in the video being processed.

4. RESULTS AND DISCUSSION

We compared our ROI video-based object tracking techniques to full-frame processing on 16 352x288 pixel-sized video clips involving either single or multiple objects of various sizes with varying velocities. The video clips ranged from 12 to 203 frames in length, at 5 fps. Also, our ROI techniques use a buffer region of 2 blocks in both X and Y directions (each block is an 8x8 pixel square). The accuracy of the object tracking is shown in Fig. 3.

As can be seen from Fig. 3A, using an ROI with a size of a quarter of the entire video frame (i.e. – data savings compared to full-frame processing is roughly 75%), the ROI position is always within 50 pixels of the optimal value in each direction, and on average is off the optimal value by 25 horizontal pixels and 23 vertical pixels (7% and 10% of the total frame size). The optimal value is the position the ROI would take if full-frame processing took place with the same object tracking algorithm. Additionally, careful manual examination of our results on all our video clips show that the object tracking with ROI processing always successfully tracked the object which is moving in the scene.

As one increases the size of the ROI as in Fig. 3B, the position of the ROI becomes more closely matched to the optimal value, when the full-frame is considered. This is expected because, with a bigger ROI, the object tracking algorithm is allowed to operate on a larger fraction of the data in the whole frame and therefore the ROI will more intelligently position itself in future frames to be centered on the moving object.

Figs. 3C-3D show the average error of the ROI, again compared to the optimal case, using the dynamic ROI technique described earlier. Fig. 3C uses a maximum ROI size of 25% of the video frame, similar to Fig. 3A, and Fig. 3D shares this same relationship to Fig. 3B. The dynamic ROI setups always process no more of the video frame than their fixed-size counterparts, but they also shrink their ROI size to save on limited resources when possible. Fig. 3C has an average error across all video clips of 32 and 21 pixels in the horizontal and vertical direction, respectively. This is less than 10% error with respect to the full size video frame.

Full frame processing considers 1232 blocks on each frame. Using fixed-size ROI processing, Figs. 3(A)-(B) operated on 308 and 468 blocks, respectively. These three sizes are the maximum allowed for the dynamic ROI technique, but *on average* only 240 and 303 blocks were processed. As such, 80% and 75% of the blocks are not stored for the ROI setups in Figs. 3(C)-(D). These blocks are never processed and energy is not wasted on them.

The *containment* of the object is another measure of the quality of the object tracking results. Fig. 4 gives a summary of the percentage of the moving object that is outside the ROI. Fig. 4(A)-(B) show that, for a fixed size ROI, 34% and 17% of the moving blocks on average were not completely contained in the ROI for the different sizes of ROI. For a dynamically adjusting ROI, on

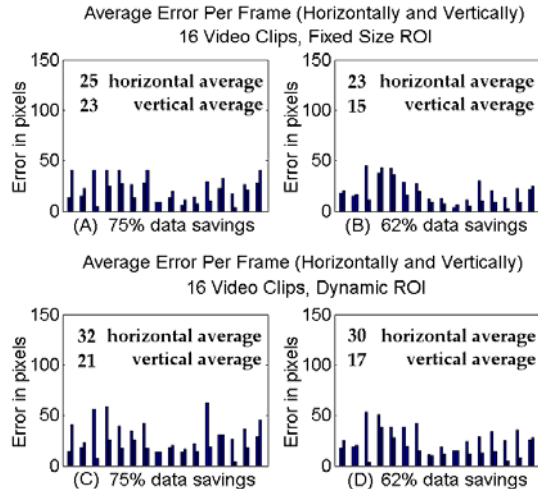


Figure 3: Performance results for ROI techniques using 2 different ROI sizes. (A) shows the difference between our ROI technique and the optimal position for the ROI based on full-frame processing, for a region with area 25% compared to the full frame. (B) has a larger ROI area. (A)-(B) use the fixed-size ROI technique, and (C)-(D) uses the dynamic ROI technique.

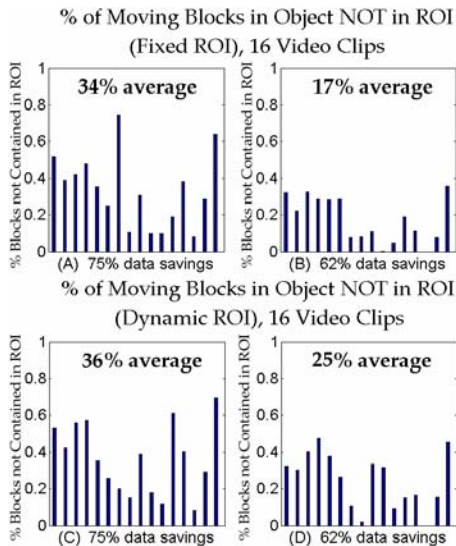


Figure 4: % of blocks in the moving object not contained in ROI. average 36% and 25% of the moving blocks of the moving object were not contained in the ROI. These containment errors might appear significant, but most of the error comes about only because the ROI is simply not large enough to contain the whole objects. Even in the worst case, Fig. 4(C), 64% of the object is within the ROI on average in each frame, and the object detection successfully tracks the objects at all times.

5. CONCLUSION

This paper introduced static-size and adjustable-size ROI techniques for a video-based object-tracking algorithm. The experiments showed the quality of the object-tracking results for various video clips and various ROI sizes.

Utilizing ROI processing in real-time video applications for resource-constrained devices allows the system designer to maintain accurate results while keeping equipment costs low. Table 2 gives a summary of the resources saved and the error introduced by using these ROI techniques.

	Full Frame	Fixed ROI			Dynamic ROI		
		(A)	(B)	46%	(C)	(D)	46%
Memory Savings	0%	75%	62%	46%	80%	75%	69%
Processing Savings	0%	74%	61%	35%	78%	73%	67%
Energy Savings	0%	86%	79%	55%	87%	86%	83%
Error (Pos)	0%	8%	7%	4%	9%	8%	7%
Error (Contain)	0%	34%	17%	10%	36%	25%	17%

Table2: Resource Savings and Error % for ROI Techniques compared to Full-Frame processing. Note the extra columns showing larger, more accurate ROI setups whose results were not included in Figs. 3-4 due to space constraints.

The energy dissipated is proportional to the voltage level of the circuit, $E_{\alpha V^2}$ [10]. Processing less data results in circuit-level slack which can be exploited to scale down the processor voltage supply according to gate-delay estimates in [11] for a 0.18 μ m process. This voltage scale down allows us to project significant energy savings, as shown in Table 2. Further work will investigate methods of implementing these techniques on various resource-limited hardware platforms.

This work was supported in part by the Semiconductor Research Corporation under Contract 2001-HJ-898 and by a Semiconductor Research Corporation Ph.D. Fellowship for Nicholas H. Zamora.

6. REFERENCES

- [1] M. Lindwer, D. Marculescu, T. Basten, R. Zimmerman, R. Marculescu, S. Jung, and E. Cantatore. Ambient Intelligence Visions and Achievements: Linking Abstract Ideas to Real-World Concepts. In *Proc. of the Design, Automation and Test in Europe*, (DATE '03), pp. 10-15, March 2003.
- [2] F. Boekhorst. Ambient Intelligence, the Next Paradigm for Consumer Electronics: How will it Affect Silicon? In *Proc. of the Solid-State Circuits Conf.*, (ISSCC '02), pp. 28-31, Feb. 2002.
- [3] A. Cavallaro, O. Steiger, and T. Ebrahimi. Multiple video object tracking in complex scenes. In *Proc of the ACM Int. Conf. on Multimedia*, (MM '02), pp. 523-532, Dec. 2002.
- [4] F. Ziliani and J. Reichel. Effective integration of object tracking in a video coding scheme for multisensor surveillance systems. In *Proc. of the IEEE Int. Conf on Image Processing*, (ICIP '02), vol. 1, pp. 521-524, June 2002.
- [5] T. Hamamoto, S. Nagao, and K. Aizawa. Real-time objects tracking by using smart image sensor and FPGA. In *Proc. of the IEEE Int. Conf. on Image Processing*, (ICIP '02), vol. 3, pp. 441-444, June 2002.
- [6] Y. Huang, B. Hsieh, S. Chien, and L. Chen. Simple and effective algorithm for automatic tracking of a single object using a pan-tilt-zoom camera. In *Proc. of the IEEE Int. Conf. on Multimedia and Expo*, (ICME '02), vol. 1, pp. 789-792, Aug. 2002.
- [7] X. Sun, J. Foote, D. Kimber, and B. Manjunath. Panoramic Video Capturing and Compressed Domain Virtual Camera Control. In *Proc. of the ACM Int. Conf. on Multimedia*, (MM '01), pp. 329-338, Oct. 2001.
- [8] Sony EVI-D30 Camera, <http://www.sony.com/>
- [9] Motion Pictures Experts Group. Overview of the MPEG-4 Standard. ISO/IEC JTC1/SC29/WG11 N4668, March 2002.
- [10] J. Rabaey, A. Chandrakasan, and B. Nikolic'. *Digital Integrated Circuits*, 2nd edn. pp. 213-231. Prentice Hall, Person Education, Inc., 2003, Upper Saddle River, NJ.
- [11] K. Chen and C. Hu. Performance and Vdd Scaling in Deep Submicrometer CMOS. In *IEEE Journal of Solid-State Circuits*, (JSSC '98), Vol. 33, No. 10, Oct. 1998.