Network-Driven Motion Estimation for Wireless Video Terminals

Wendi B. Rabiner, *Student Member, IEEE,* and Anantha P. Chandrakasan, *Member, IEEE*

***Abstract*— Motion estimation has been shown to help significantly in the compression of video sequences. However, since most motion estimation algorithms require a large amount of computation, it is undesirable to use them in power constrained applications, such as battery-operated wireless video encoders. This paper describes a new compression algorithm, termed network-driven motion estimation (NDME), which reduces the power dissipation of wireless video devices in a networked environment by exploiting the predictability of object motion. Since the location of an object in the current frame can often be predicted accurately from its location in previous frames, it is possible to optimally partition the motion estimation computation between the portable devices and high powered compute servers on the wired network. In network-driven motion estimation, a remote high-powered resource at the base-station (or on the wired network), predicts the motion vectors of the current frame from the motion vectors of the previous frames. The base-station sends these predicted motion vectors to a portable video encoder, where motion compensation proceeds as usual. Network-driven motion estimation adaptively adjusts the coding algorithm based on the amount of motion in the sequence, using motion prediction to code portions of the video sequence which contain a large amount of motion and conditional replenishment to code portions of the sequence which contain little scene motion. This algorithm achieves a reduction in the number of operations performed at the encoder for motion estimation by over *two orders of magnitude* while introducing minimal degradation to the decoded video compared with full search encoder-based motion estimation.**

***Index Terms*—Energy-efficient, low power, motion estimation, motion prediction, networking.**

I. Introduction

As the use of multimedia devices becomes more prevalent, there is an increasing need to transmit video over low bandwidth networks such as wireless channels. Due to the large amount of data contained in these signals, efficient compression techniques are extremely important. Conventional video systems use some form of scene motion estimation/motion compensation at the encoder to achieve high compression ratios because it is very effective in decorrelating successive frames. However, most motion estimation algorithms require an extensive search to minimize a distortion metric, which is extremely computationally intensive (e.g., a ± 16 pel full-search motion estimation for 30 f/s video requires over 65 000 ops/pixel/s). The high computational requirement of this algorithm makes it unsuitable for power constrained portable video terminals. While work has been done on algorithms which reduce the amount of computation required to perform motion estimation (e.g., hierarchical search, successive elimination algorithm [1], [2]), there is an inherent tradeoff between computation and motion vector accuracy for conventional motion estimation algorithms—the ones which produce accurate motion vectors are extremely computationally complex, whereas the computationally simple algorithms often produce inaccurate motion vectors and hence do not achieve large amounts of compression.

A standard method for compressing the video stream without performing motion estimation is called conditional replenishment (CR). CR consists of representing each frame as the difference between itself and the previous frame. This method requires very little computation and does not require motion vectors—the image is represented entirely by the difference image. However, CR will only be effective in reducing the temporal correlation if there is little or no motion in the image. If there is a large amount of motion, performing CR and sending the difference image may not achieve an acceptable amount of compression of the original image. This paper describes a compression method which, rather than reducing the computation for motion estimation, transfers the majority of the computation from the power constrained video encoder to a high-powered server on the network.

II. Low-Power System Design

Reducing power consumption is the key to extending the battery life of portable devices and thus should be one of the driving forces when designing motion estimation algorithms for portable video devices. At a high level, the average power P dissipated by a digital signal processing (DSP) algorithm is given by

$$P = \sum_i N_i C_i V_{dd}^2 f_s \tag{1}$$

where N_i is the number of operations of type i (addition, multiplication, storage, or bus access) performed per sample, C_i is the capacitance switched per operation of type i, V_{dd} is the supply voltage, and f_s is the sample frequency. The main focus of low-power design has been the reduction of the supply voltage, as this results in a quadratic reduction in P. A variety of voltage scaling strategies have been developed to

Manuscript received September 30, 1996; revised January 31, 1997. This work was supported by the ARL Federated Lab program under Contract DAAL01-96-2-0001. The work of W. Rabiner is supported by an NSF Fellowship.

The authors are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

Publisher Item Identifier S 1051-8215(97)05890-4.

reduce the supply voltage without any loss in performance [3]. However, average power can also be lowered by reducing the number of operations per sample, N_i, through architectural and algorithmic optimizations. For example, approximate signal processing algorithms, such as the one described in [4] which dynamically changes the order of digital filters based on signal statistics, obtain large reductions in N_i. Algorithmic optimizations such as computing with powers of two (i.e., converting multiplications, which require a large amount of power, to low-complexity shifts and adds) also lower N_i [5]. At a system level, the computation can be optimally partitioned between the portable devices and the high-powered network servers to reduce the computation in the portable devices. An example of this is the InfoPad terminal [3], which partitions the text/graphics functions such that the portable terminal only performs display updates while a remote high-powered server performs the computationally intensive portions of the X-server functions. Network-driven motion estimation (NDME) uses a similar nonconventional system-level approach which exploits network resources to reduce the power consumption of portable video devices.

III. Motion Estimation

Motion estimation (ME) is used to reduce the amount of temporal correlation that exists in most video sequences. Estimating scene motion can be done on a pixel level [6], [7], a block level [8], or a global level [9]. The MPEG standard [10] employs block-based motion estimation, as this is the simplest motion estimation algorithm and most easily implemented in hardware. For this technique, each image is broken into 16 $\times$ 16 pixel blocks, called macroblocks. These macroblocks are small enough that there (generally) exists a good match for each macroblock compared with the pixels from the previous frame. This best match is found by comparing each macroblock of the current image to the pixels from the previous image within a certain specified area, called a search window. The size of the search window depends on the sampling rate of the video—typically, for 30 f/s video, objects do not move more than $\pm$16 pels from one frame to the next so the search window is set to 33 $\times$ 33 pixels. The best match is the set of pixels within the search window from the previous image which is closest, according to some error metric, to the pixels of the macroblock. Frequently, the sum of absolute differences (SAD) is the error metric used. If $f(x, y, t)$ represents the luminance value of the pixel at location (x, y) at time t, the SAD between macroblock i at time t and time $t-1$ is defined as

$$\mathrm{SAD}(i) = \sum_{y \in MB_i} \sum_{x \in MB_i} |f(x, y, t) - f(x, y, t-1)|. \quad (2)$$

A motion vector (MV), which specifies the location of this best match and, hence, the motion of the macroblock, is generated for each macroblock in the image.

A motion-compensated image is created by assembling all the best match macroblocks into an image. This motion-compensated image represents a prediction of the current image based on the previous image and the motion of the sequence. The original image is thus represented by the set of motion vectors (one for each macroblock) and the motion-compensated difference image, which is the residual between the original current image and the motion-compensated image. Typically, much less data is required to code the motion vectors and the motion-compensated difference image than is needed to code the original image, so compression is achieved. Fig. 1 shows a conventional video encoder which uses motion estimation.

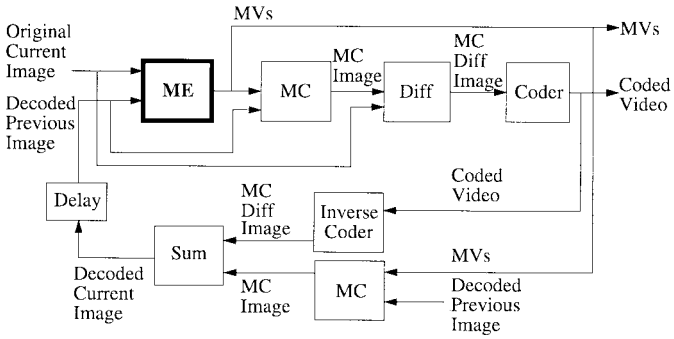


Fig. 1. Video encoder which uses motion estimation.

Motion estimation/motion compensation is very effective in decorrelating successive frames. However, it is extremely computationally intensive since every macroblock of the image must be compared to every location within the search window. For each comparison, the SAD, which requires two addition operations (subtracting the pixel values and adding this difference to the running sum) is performed for all 256 pixels in the macroblock. For a search window of size $w \times h$ (e.g., 33 $\times$ 33 for a $\pm$16-pel search) and 30-f/s video, this requires $60wh$ ops/pixel/s (e.g., 65 340 ops/pixel/s). Due to the high computational requirement of this full-search ME algorithm, it is not suitable for battery-operated encoders.

Much work has been done on algorithms which reduce the computational intensity of motion estimation. Many fast algorithms such as the two-dimensional (2-D)-logarithmic search, three-step search, and modified conjugate direction search [11] use only a few locations within the search window to find the MV, and algorithms like hierarchical ME [1] reduce the motion vector search space by subsampling the original image. However, these algorithms do not guarantee that the "true" MV has been found, only that a locally optimum MV has been found. These algorithms decrease the computation but typically increase the bandwidth of the coded video compared with full search ME. Other fast ME algorithms rely on the correlation that typically exists between MV's of spatially and temporally neighboring macroblocks in order to reduce the motion estimation computation [12]–[14]. These algorithms can produce fairly accurate MV's, but often they do not substantially reduce the motion estimation computation. Finally, algorithms such as the SEA [2] and spiral search use mathematical inequalities to reduce the computation while still guaranteeing that the "true" MV is found.

Motion estimation is a necessary step in the coding of video sequences, but it also requires an excessive amount of computation. Table I summarizes various fast motion estimation

TABLE I
MOTION ESTIMATION ALGORITHMS (SEARCH WINDOW $w \times h$, 33×33)

Algorithm	Computation (ops/pixel/frame)		Accuracy
Full Search	$2wh$	1	High
Spiral Search	Variable, $\leq 2wh$	≤ 1	High
Successive Elimination Algorithm [2]	Variable, $\leq 2wh$	≤ 1	High
Predicted MV Algorithm [13]	$0.1(2wh) - 0.2(2wh)$	$0.1 - 0.2$	Medium-High
Interblock Correlation Method [14]	Variable, $\ll 2wh$	$\ll 1$	Medium
MV Field Subsampling [12]	$wh + 4$	0.5	Medium
N-Level Hierarchical ME [1]	$2\sum_{n=0}^{N}\lceil\frac{w}{2^{2N-n}}\rceil\lceil\frac{h}{2^{2N-n}}\rceil$	$(N = 2)$ 0.1	Medium
2D-logarithmic Search [11]	$\geq 8\min(\log_2(\frac{w}{2}), \log_2(\frac{h}{2}))$ $\leq 14\min(\log_2(\frac{w}{2}), \log_2(\frac{h}{2}))$	≥ 0.015 ≤ 0.026	Medium
3 Step Search [11]	50	0.023	Low
Modified Conjugate Direction Search [11]	≥ 10 $\leq 2[(\frac{w}{2}+1)+(\frac{h}{2}+1)+1]$	≥ 0.005 ≤ 0.032	Low
Network-Driven ME	**18**	**0.008**	**Medium-High**

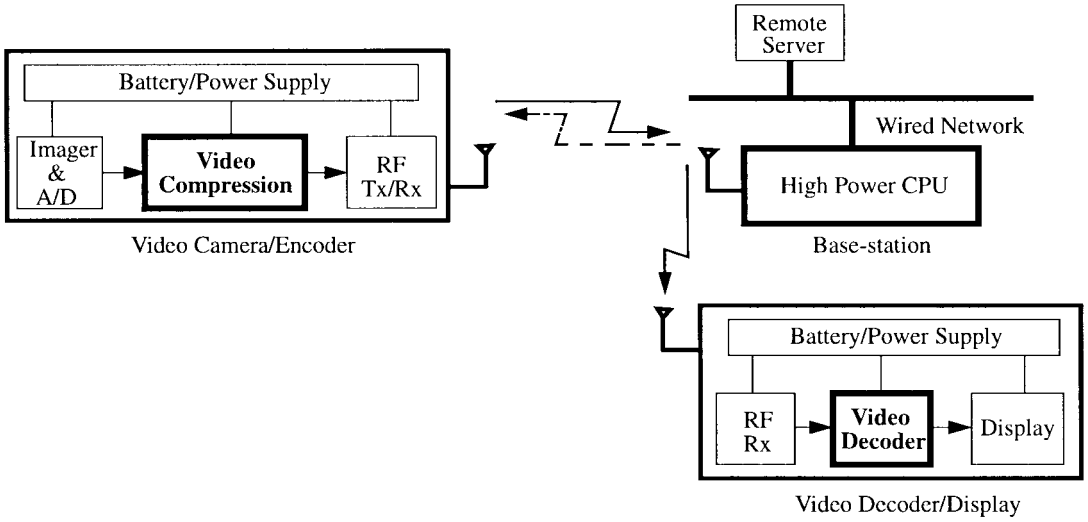


Fig. 2. Wireless video terminals in a networked environment.

algorithms and shows the tradeoff between computation and MV accuracy when ME is performed at the encoder.

IV. THE NETWORK AS A COMPUTATIONAL RESOURCE

A. Networked Wireless Video Systems

A block diagram of a wireless video system in a networked (wired and wireless) environment is shown in Fig. 2. This system contains a battery-operated video encoder, which acquires, digitizes, and encodes a video signal and transmits the coded video over a wireless link to a base-station. The base-station sends the coded video to a battery-operated video decoder, which decodes and displays the received images. In addition, the base-station can communicate with the encoder through a low bandwidth reverse wireless link, and it has access to a number of high-powered compute servers through a high bandwidth wired backbone. In conventional wireless video systems, the motion estimation is performed at the encoder, the coded video is sent to the base-station, and the base-station retransmits the coded video to the decoder. However, since there are stringent power constraints in the portable encoder, simple motion estimation algorithms must be used, which reduce the accuracy of the motion vectors and increase the bandwidth of the coded video. Since the wireless channel is a relatively low bandwidth channel, the quality of the decoded images may be highly degraded in order to reduce the bandwidth of the coded video.

B. Motion Prediction

The motion of objects in natural scenes is mostly continuous from one frame to the next. Therefore, by knowing the location of an object in all previous frames, it is possible to predict its location in the current frame. For example, the location of any object which moves with a constant velocity, such as a moving car, is predictable from two previous frames. Since most objects which do not move with constant velocity accelerate very slowly across the span of a frame in 30 f/s video, the constant velocity assumption is a good approximation to the motion of all objects. Since motion vectors represent the flow of motion from one frame to the next, it is possible to roughly predict the motion vectors of a frame from the motion vectors of the previous frames. The authors in [13] and [14] develop various algorithms to predict the MV's of the current frame from the MV's of the previous frame (or the neighboring MV's

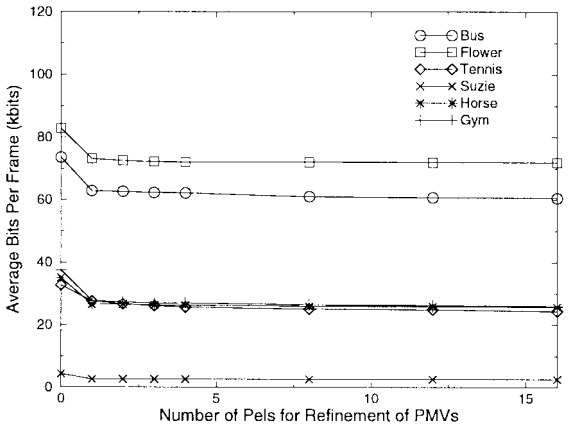


Fig. 3. Average number of bits per frame versus the number of pixels used in the PMV refinement search.

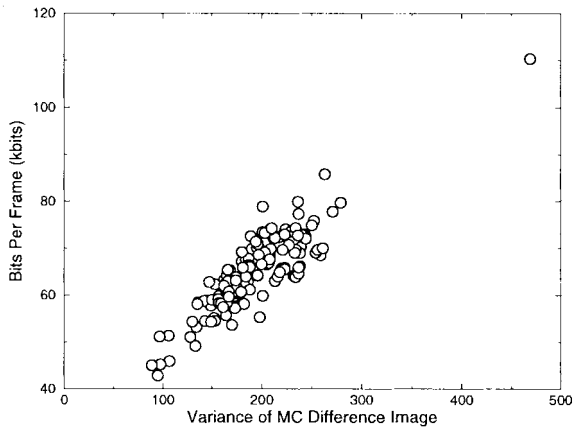


Fig. 5. Bit rate required to code each frame versus the variance of that frame.

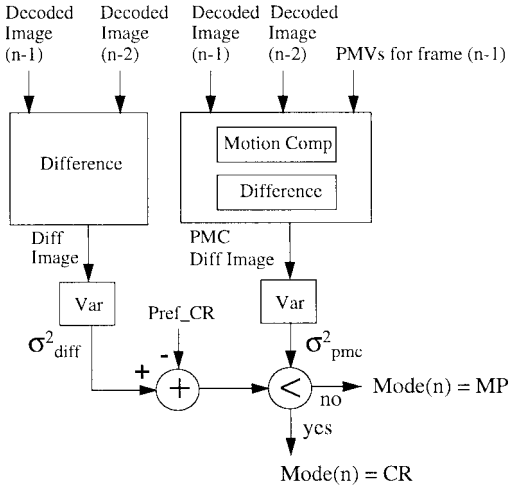


Fig. 4. Algorithm for determining NDME system mode at the base-station.

in the current frame) in order to reduce the search space for encoder motion estimation and thus reduce the computational complexity of motion estimation. However, these algorithms still require that the encoder perform the reduced computation and are still unsuitable for battery-operated video encoders.

A majority of the computation for motion estimation can effectively be *removed* from the encoder by exploiting a high-powered remote compute server to perform motion prediction (MP). A resource on the wired network with no power constraints (either at the base-station or a remote location) can estimate the motion of the sequence based on previous decoded frames and use this information to predict the motion vectors of the current frame. Since the motion of each macroblock is assumed to be relatively continuous from one frame to the next (for natural sequences at 30 f/s), the predicted motion vector for each macroblock of the current image is simply the motion vector for that macroblock in the previous reconstructed frame, which is found using full-search ME on the decoded previous frames either at a high-powered server at the base-station or at a remote high-powered server on the wired network.

These predicted MV's (PMV's) are then sent to the encoder, where, since these motion vectors represent only an approximation of the motion of each macroblock, the encoder performs a local search to improve the MV estimates and does motion compensation using these refined PMV's. Fig. 3 shows the average number of bits required to code each frame of the standard MPEG sequences "Bus," "Flower," "Tennis," and "Suzie" as well as the sequences "Horse" and "Gym" (using a constant quantization step size) versus the number of pels used to refine the PMV's (i.e., k pels translates into a $(2k+1) \times (2k+1)$ pixel search window and $60 \times (2k+1)^2$ ops/pixel/s). This plot shows that there is a large decrease in the average number of bits required to code each frame when the PMV's are refined by ± 1 pel compared with no PMV refinement. However, increasing the search beyond ± 1 pel drastically increases the amount of encoder computation required for the refinement search but does not reduce the data bandwidth significantly. The PMV refinement search is therefore set to ± 1 pel, as this requires only 540 ops/pixel/s and represents the best tradeoff between reducing encoder computation and reducing the data bandwidth.

The encoder transmits the PMV refinements and compressed video back to the base-station. A high-powered server at the base-station (or a remote server) reconstructs the image and performs motion estimation using the current decoded image and the previous decoded image to find the optimal MV's for the current reconstructed frame and hence the PMV's for the next frame. The base-station also retransmits the coded video as well as the refined predicted motion vectors to the decoder.

C. Adaptive Coding

Motion prediction minimizes the tradeoff between encoder computation and coded video bandwidth by optimally partitioning the motion estimation computation between portable devices and network resources. However, when there is little or no scene motion, it is advantageous to use CR to code the image, rather than predicting the motion (or lack of motion), since CR does not require the transmission of motion vectors. Therefore, it is advantageous to adaptively switch between coding using MP and CR in order to obtain the maximum amount of compression for each frame. This method of reducing temporal correlation using an optimal combination

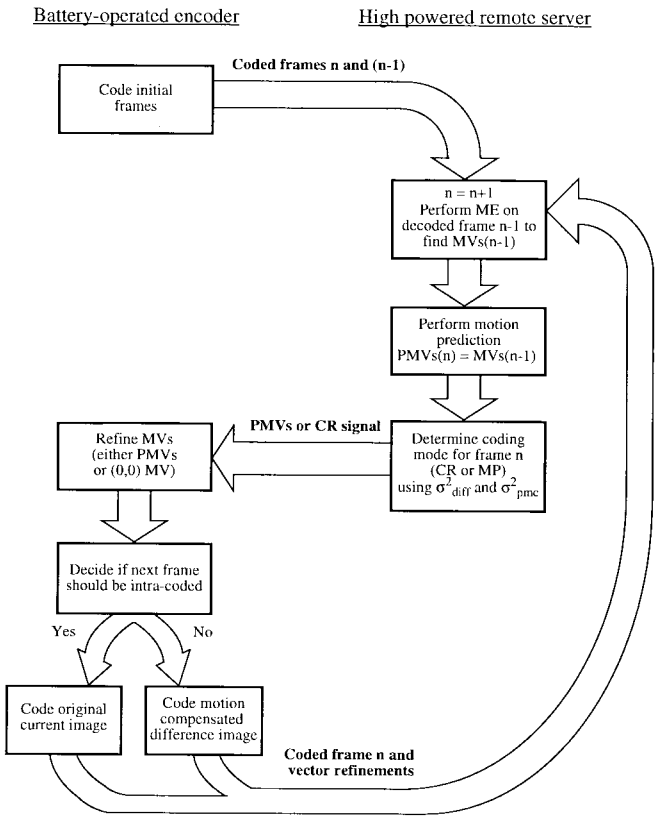


Fig. 6. Flow graph showing the network-driven motion estimation algorithm.

of CR and motion prediction is called *network-driven motion estimation*.

The decision about which mode to use for a given frame is made at the base-station (or a remote server) based on a comparison of the variance of the difference image and the variance of the predicted motion-compensated difference image of the past frame, as shown in Fig. 4. This metric is used because the variance of an image is a good measure of the amount of activity in that image [15]. Since the amount of data needed to code each of the difference images is proportional to the amount of activity in these images, the respective variances provide an efficient means of determining which mode should be used. Fig. 5 shows a plot of the motion-compensated difference image variance versus the bit rate required to code the frames of the "Bus" sequence (using a constant quantization step size). This shows that there is a definite correlation between the variance of these images and the number of bits required to code them.

The base-station sends the encoder either a signal to use the zero motion vector for each macroblock, when the system is operating in the CR mode, or the PMV's, when the system is operating in the MP mode. In either case, the encoder performs a local search centered around the specified motion vectors (either predicted or zero) and sends the motion vector refinements along with the compressed image (i.e., quantized DCT coefficients of the motion-compensated difference image) to the base-station. The image is reconstructed at the base-station (or at a remote high-powered server), and motion estimation is performed on the reconstructed image, even if the frame was

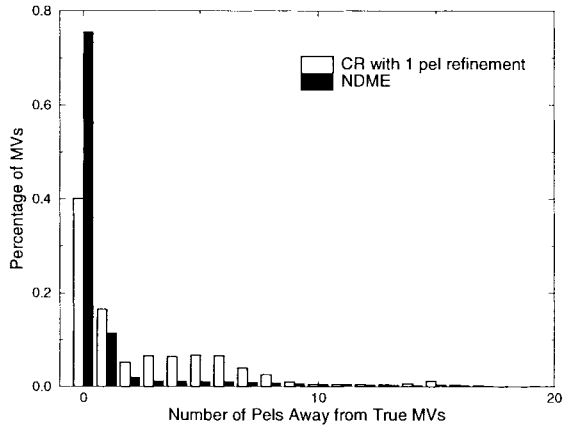
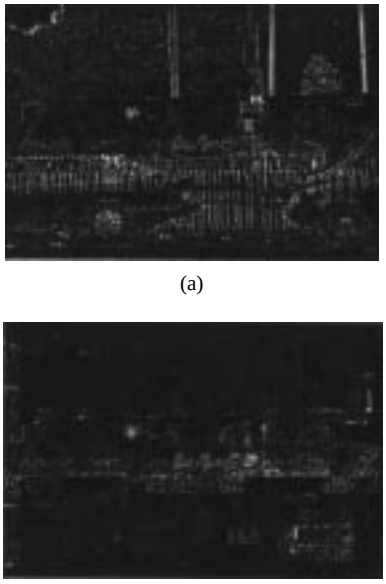


Fig. 7. Distance between predicted/CR MV's and true MV's.



(a)

(b)

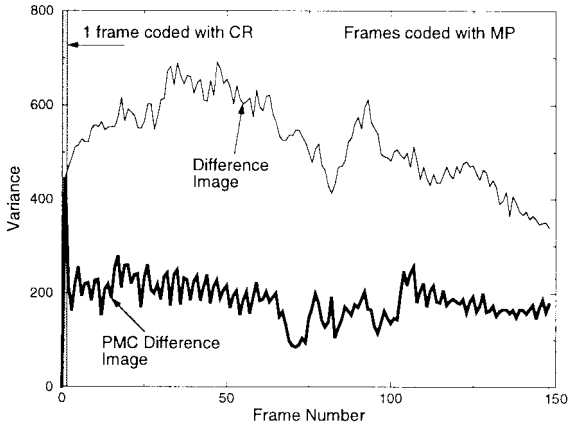
Fig. 8. Bus sequence: (a) CR with 1-pel refinement difference image and (b) NDME motion-compensated difference image.

TABLE II
AVERAGE NUMBER OF KILOBITS PER FRAME WITH FIXED QUANTIZATION STEP SIZE

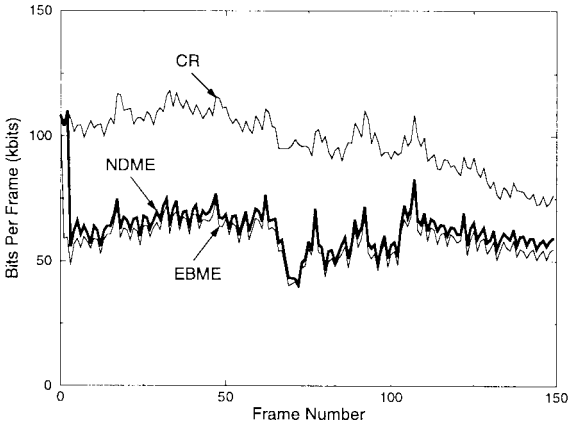
Sequence Name	Encoder-Based Motion Estimation		Network-Driven Motion Estimation		Conditional Replenishment	
Bus	58.6	(x1)	62.8	(x1.07)	98.1	(x1.67)
Flower	70.5	(x1)	73.2	(x1.04)	112.8	(x1.60)
Tennis	22.8	(x1)	27.6	(x1.21)	34.0	(x1.49)
Suzie	2.4	(x1)	2.6	(x1.08)	2.7	(x1.13)
Horse	23.5	(x1)	26.5	(x1.13)	46.9	(x2.00)
Gym	23.3	(x1)	27.7	(x1.19)	38.8	(x1.66)
Average	33.5	(x1)	36.7	(x1.10)	55.6	(x1.66)

coded using CR rather than MP. It is necessary to perform motion estimation on each reconstructed frame in order to determine the PMV's of the next frame, which are needed to determine the predicted motion-compensated difference image and hence the coding mode for the following frame.

The first frame of a sequence is intracoded, and the second frame is coded using CR, since the system has no previous

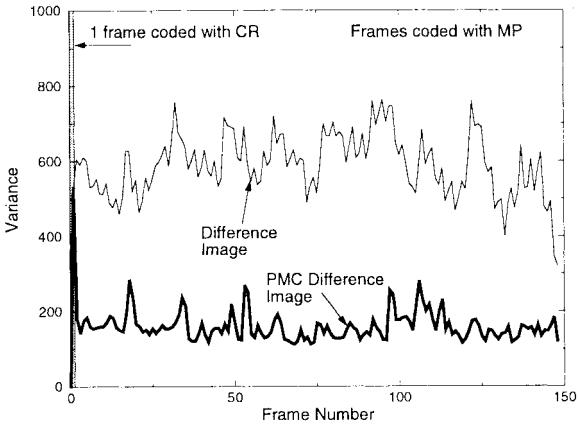


(a)

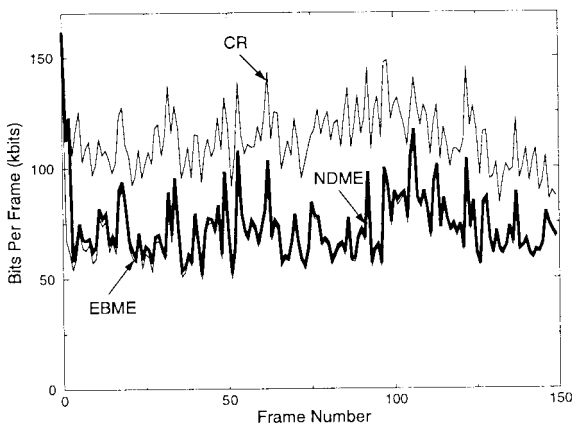


(b)

Fig. 9. Bus sequence: (a) variance of CR and PMC difference images and (b) number of bits required for each frame with fixed QP.



(a)



(b)

Fig. 10. Flower sequence: (a) variance of CR and PMC difference images and (b) number of bits required for each frame with fixed QP.

frames from which to perform motion estimation and predict the sequence motion. After these initial frames, the base-station determines which mode to use to code each frame based on the variance metric described above. However, since no PMV's need to be transmitted when the system is operating in the CR mode, this mode is slightly preferenced over MP when there is little scene motion. Thus σ^2_{diff} is reduced by Pref_CR before it is compared with σ^2_{pmc}, as shown in Fig. 4.[1] If $\sigma^2_{\text{diff}} -$ Pref_CR is less than σ^2_{pmc}, the next frame is coded using CR; otherwise, the next frame is coded using MP. This method of deciding the mode for each frame continues until the encoder forces an intra frame, which occurs if 1) the encoder detects a scene change, 2) a transmission error causes the encoder and base-station to lose synchronization, or 3) enough frames have passed that the encoder decides to refresh the sequence and begin with an intracoded frame.[2] Once the encoder forces an intra frame, the entire network-driven motion estimation process begins again.

This NDME system uses the same video encoder as the conventional encoder shown in Fig. 1 except the NDME system motion estimation is limited to a ± 1-pel search centered around the PMV's (or the (0, 0) vector) received from the base-station. Thus, *any* type of video coder (e.g., DCT-based, subband) can be used to code the motion-compensated difference image. The entire network-driven motion estimation algorithm can be seen in the flow-chart in Fig. 6. The flow-chart shows that the majority of the computation for motion estimation has been shifted from the power-constrained encoder to a remote high-powered server. This requires a low latency network (both wired and wireless) and a high-performance remote server in order to do the following in the time span of a single frame period: compute the PMV's, determine the coding mode, transmit the PMV's or the CR signal from the remote server through the wired and wireless networks to the encoder, refine the vectors at the encoder, compress the frame using the refined vectors, and transmit the vector refinements and coded video back to the remote server.[3]

[1] Pref_CR was empirically determined to be 50 if the previous frame was coded using CR and zero if the previous frame was coded using MP. This was done in order to keep the coding modes relatively consistent from one frame to the next.

[2] The maximum number of frames allowed before an intra frame is transmitted can be set for the particular application.

[3] In order to meet the delay requirements, the system will need to exploit current research in low latency protocols. In addition, the delay can be reduced by placing a high-powered server at the base-station to avoid transmission through the wired network.

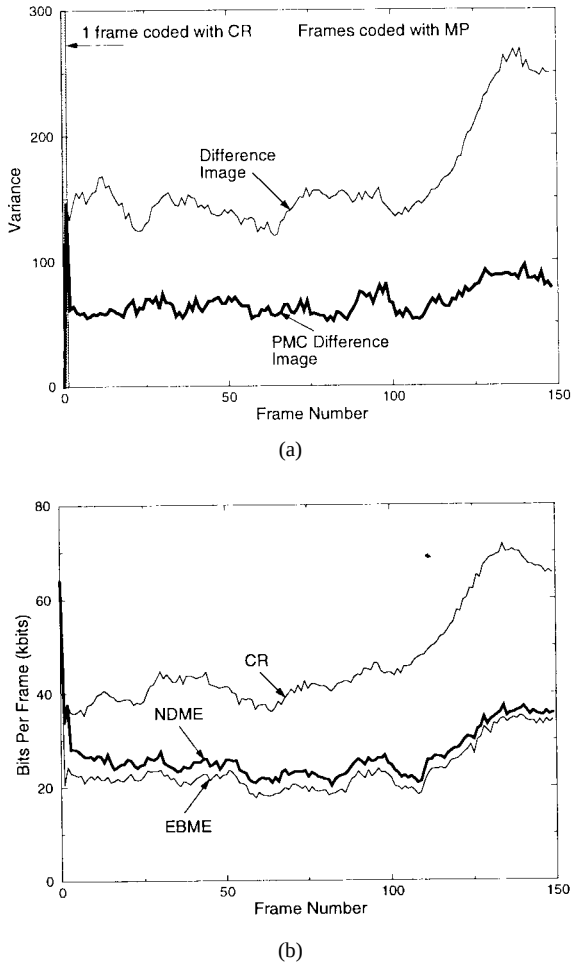


Fig. 11. Horse sequence: (a) variance of CR and PMC difference images and (b) number of bits required for each frame with fixed QP.

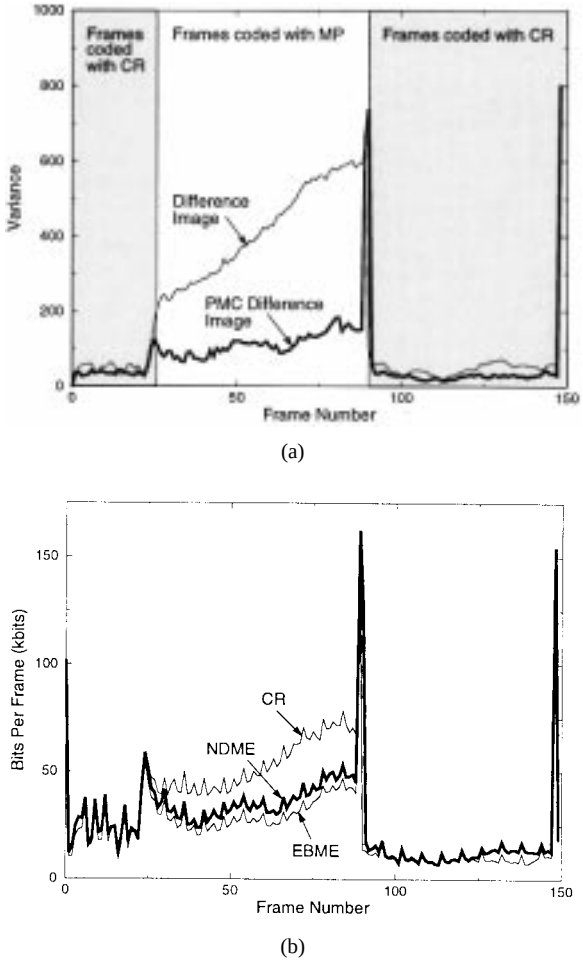


Fig. 12. Tennis sequence: (a) variance of CR and PMC difference images and (b) number of bits required for each frame with fixed QP.

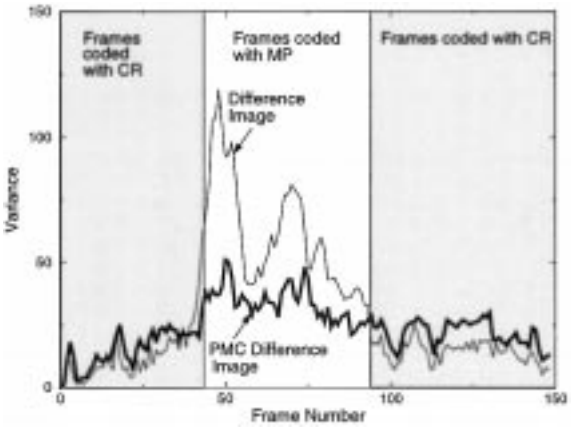
The total data bandwidth when the system is operating in the MP mode includes the PMV's being sent from the base-station to the encoder, the corrections to the PMV's being sent back to the base-station, and the compressed video. When the system is operating in the CR mode, the total data bandwidth includes the MV refinements and the compressed video.

The advantage of using NDME rather than CR can be seen in Fig. 7, the percentage of motion vectors from the NDME system that are close to the "true" motion vectors as compared with the percentage of motion vectors from the CR system (1-pel refined (0, 0) motion vectors) that are close to the "true" motion vectors for the "Bus" sequence. There are substantially more motion vectors from the NDME system than from the CR system that are the same as the "true" motion vectors. This shows the relatively high accuracy of this prediction method. Fig. 8 also shows the relatively high accuracy of NDME. Fig. 8(a) shows a CR with 1-pel refinement motion-compensated difference image, and Fig. 8(b) shows an NDME motion-compensated difference image. The NDME motion-compensated difference image contains substantially less perceptually relevant information than the CR motion-compensated difference image, and thus the NDME system will achieve greater compression for this frame than the CR system.

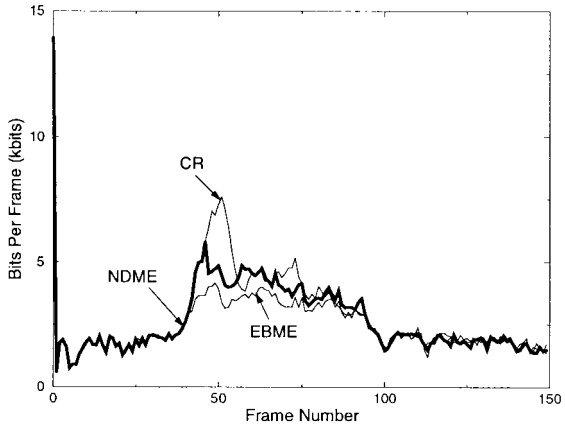
V. Experimental Results

This system was implemented in software and incorporated into the H.263 platform. To test this system, several experiments were run using 150 frames of the standard sequences "Bus," "Flower," "Tennis" (all 352 $\times$ 240 pels), and "Suzie" (176 $\times$ 144 pels), as well as 150 frames of the sequences "Horse" and "Gym" (both 352 $\times$ 240 pels). Each of these sequences was coded at a constant quantization step size (MPEG QP $=$ 10) using NDME, encoder-based ME (EBME),[4] and CR with 1-pel refinement. For all three coding methods, motion vectors were obtained at full pel accuracy, and there was one motion vector generated for each macroblock. The average number of bits required to code each of the sequences using the different coding methods is shown in Table II. These results show that coding with NDME requires only a 10% higher average bit rate than coding with ideal encoder-based ME and achieves over a 33% decrease in average bit rate compared to coding with CR. It is interesting to note that for the sequences which contain a large amount of motion and thus use MP to code all the frames (i.e., the sequences "Bus," "Flower," and "Horse"), NDME requires only a 6% higher average bit rate than coding with ideal encoder-based ME and

[4] For these experiments, only forward ME was used, rather than bidirectional ME, since bidirectional ME requires a substantial increase in computation.

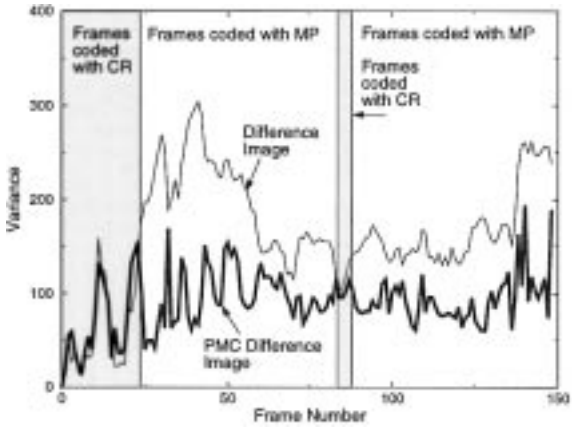


(a)

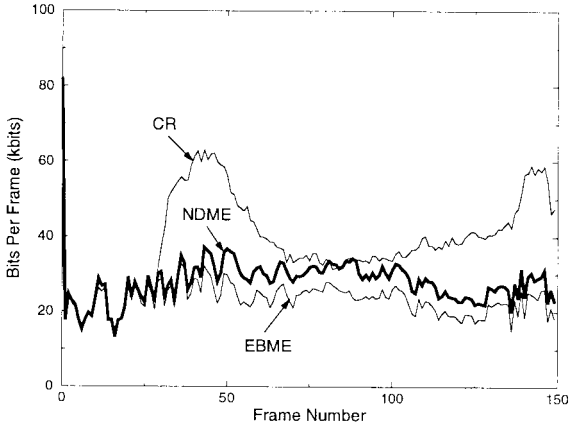


(b)

Fig. 13. Suzie sequence: (a) variance of CR and PMC difference images and (b) number of bits required for each frame with fixed QP.



(a)



(b)

Fig. 14. Gym sequence: (a) variance of CR and PMC difference images and (b) number of bits required for each frame with fixed QP.

achieves over a 36% decrease in average bit rate compared to coding with CR. Since the encoder performs the same computation for CR and NDME, this savings in bandwidth does *not* come at the expense of an increase in encoder power.

As described in Section IV-C, the decision about which mode to use to code the current frame is made at the base-station based on the variances of the CR and predicted motion-compensated (PMC) difference images of the previous frame. Fig. 9(a) shows these variances for the "Bus" sequence. After the initial frames, which must be intra-coded and CR-coded, the variance of the PMC difference image is substantially lower than the variance of the CR difference image, and the NDME system codes the remainder of the sequence using the MP mode.

Fig. 9(b) shows the number of bits required to code each frame of the "Bus" sequence at a constant QP. The bit rate per frame of the NDME system is, on average, only 2.5–5 kb worse than the ideal encoder-based ME system and 25–50 kb *better* than the CR system. By coding this sequence with NDME rather than CR, the bandwidth decreases by as much as 50% with no increase in encoder power dissipation. Similar results were obtained for both the "Flower" and "Horse" sequences, shown in Figs. 10 and 11, where, after the initial frames, the NDME system codes the remainder of these sequences using MP.

Comparatively, the "Tennis," "Suzie," and "Gym" sequences have much less motion. This can be seen by comparing the variances of the CR and PMC difference images, shown in Fig. 12(a) for the "Tennis" sequence. For the first few frames, the variances of both difference images are very similar. In this case, the NDME system prefers using the CR mode since no motion vectors need to be transmitted to the encoder. Scene motion begins around frame 25, which can be seen by the large increase in the variance of the CR difference image. At this point, it is advantageous to code the images using MP, so the NDME system switches to this mode. Around frame 90 there is a scene change and the encoder automatically sends an intra frame and forces the system into the CR mode. The remainder of the sequence has little motion, so the system remains in the CR mode. By adaptively switching between CR and MP, the NDME system is able to obtain bit rates as low as the CR system for frames with little or no motion and substantially lower than the bit rates from the CR system for frames with large amounts of motion.

Fig. 12(b) shows the number of bits required to code each frame of the "Tennis" sequence. Only the middle frames of the

(a)

(b)

(c)

Fig. 15. Tennis sequence coded with constant bit rate of 400 kb/s: (a) encoder-based ME, SNR = 27.3 dB, (b) network-driven ME, SNR = 25.4 dB, and (c) conditional replenishment, SNR = 22.3 dB.

(a)

(b)

(c)

Fig. 16. Suzie sequence coded with constant bit rate of 70 kb/s: (a) encoder-based ME, SNR = 32.4 dB, (b) network-driven ME, SNR = 30.2 dB, and (c) conditional replenishment, SNR = 27.5 dB.

sequence are coded using MP; for these frames, the NDME system codes the images using approximately 5–10 kb more than the ideal encoder-based ME system and 15–25 kb fewer than the CR system. Similar results were obtained for the "Suzie" and "Gym" sequences, shown in Figs. 13 and 14, where NDME adaptively switches the coding mode for these sequences in order to obtain the greatest possible amount of compression for each frame.

The sequences were also coded using a constant bit rate to see the effects of these different coding algorithms on image quality. Figs. 15 and 16 show the relatively small degradation in image quality obtained by using NDME rather than encoder-based ME and the large improvement in image quality obtained by using NDME rather than CR for these images.

The search window using NDME has parameters $w = 3$ and $h = 3$ (1-pel refinement), which requires 18 ops/pixel/s, whereas the search window using encoder-based ME has

parameters $w = 33$ and $h = 33$ (± 16 pel search window), which requires 2178 ops/pixel/s. Thus, the computation at the encoder is reduced by a factor of 121 per macroblock using NDME rather than encoder-based ME, which results in a significant savings in power at the encoder.

When the video sequences are coded using a constant bit rate, there is no increase in RF transmit power using NDME rather than encoder-based ME. There is, however, a slight increase in the RF power dissipation in order to receive the predicted motion vectors from the base-station. Since the bandwidth of the reverse channel is minimal, this loss in RF receive power is negligible compared to the large savings in encoder power obtained by reducing the encoder computation. The results described above show that NDME can obtain this reduction in encoder power with minimal degradation in the decoded images.

VI. Conclusion

Video is becoming an integral part of many portable devices, such as wireless cameras, personal communication devices, and video cellular phones. Due to the massive amount of data contained in video signals and the limited bandwidth of the wireless channel, compression techniques for these applications are extremely important. However, most algorithms which produce substantial compression are computationally intense. For portable video devices, this computational complexity translates into a shorter battery lifetime. If, on the other hand, computationally simple compression algorithms are used, there may not be an acceptable amount of compression of the video signal. This translates into using a larger quantization step size in order to compress the data to fit into the limited bandwidth of the channel, which causes the quality of the decoded images to be degraded. This illustrates the tradeoff between encoder computation and amount of compression when motion estimation is performed at the video encoder. By transferring the majority of the computation required for motion estimation from the power-limited portable video encoder to a high-powered system on the network, network-driven motion estimation can minimize this tradeoff to achieve over two orders of magnitude reduction in encoder power for motion estimation with only a slight increase in data bandwidth over conventional encoder-based motion estimation. Thus, network-driven motion estimation can achieve the bandwidth efficiency of encoder-based motion estimation and the power efficiency of conditional replenishment.

References

[1] V. Seferidis and M. Ghanbari, "Hierarchical motion estimation using texture analysis," in *IEE Int. Conf. Image Processing and Its Applications*, Apr. 1992, pp. 61–64.

[2] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. Image Processing*, vol. 4, no. 1, pp. 105–107, Jan. 1995.

[3] A. Chandrakasan and R. Broderson, *Low Power Digital CMOS Design.* Boston, MA: Kluwer, 1995.

[4] J. Ludwig, S. Nawab, and A. Chandrakasan, "Low-power digital filtering using approximate processing," *IEEE J. Solid-State Circuits*, vol. 31, no. 3, pp. 395–400, Mar. 1996.

[5] T. Meng, B. Gordon, E. Tsern, and A. Hung, "Portable video-on-demand in wireless communication," *Proc. IEEE*, vol. 83, no. 4, pp. 659–680, Apr. 1995.

[6] J. Biemond, L. Looijenga, D. Boekee, and R. Plompen, "A pel-recursive Wiener-based displacement estimation algorithm," *Signal Processing*, vol. 13, pp. 399–412, 1987.

[7] J. Huang and R. Merserau, "Multi-frame pel-recursive motion estimation for video image interpolation," in *Proc. ICIP'94*, vol. 2, 1994, pp. 267–271.

[8] H.-M. Hang, A. Puri, and D. Scilling, "Motion-compensated transform coding based on block motion-tracking algorithm," in *IEEE Int. Conf. Communications'87*, vol. 1, pp. 136–140.

[9] Y. T. Tse and R. L. Baker, "Global zoom/pan estimation and compensation for video compression," in *Proc. ICASSP'91*, 1991, pp. 2725–2728.

[10] D. Gall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, vol. 34, pp. 46–58, Apr. 1991.

[11] A. Netravali and B. Haskell, *Digital Pictures: Representation, Compression, and Standards.* New York: Plenum, 1995, ch. 5.

[12] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 148–157, Apr. 1993.

[13] S. Kim and C.-C. Kuo, "A stochastic approach for motion vector estimation in video coding," in *Neural and Stochastic Methods in Image and Signal Processing III, Proc. SPIE*, July 1994, vol. 2304, pp. 111–122.

[14] C.-H. Hsieh, P.-C. Lu, and J.-S. Shyn, "Motion estimation using interblock correlation," in *1990 IEEE Int. Symp. Circuits Syst.*, May 1990, vol. 2, pp. 995–998.

[15] A. Puri and R. Aravind, "Motion-compensated video coding with adaptive perceptual quantization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, pp. 351–361, Dec. 1991.

Wendi B. Rabiner (S'96) received the B.S. degree in electrical engineering from Cornell University, Ithaca, NY, in 1995 and the M.S. degree in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge, in 1997.

She is currently working toward the Ph.D. degree at MIT in the Research Laboratory of Electronics. Her research interests include image processing and coding, specifically algorithms designed for power constrained video systems.

Ms. Rabiner is a member of Eta Kappa Nu, Tau Beta Pi, and Sigma Xi.

Anantha P. Chandrakasan (S'87–M'95) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, in 1989, 1990, and 1994, respectively.

Since September 1994, he has been the Analog Devices Career Development Assistant Professor of Electrical Engineering at the Massachusetts Institute of Technology, Cambridge. His research interests include the ultra low power implementation of custom and programmable digital signal processors, wireless sensors and multimedia devices, emerging technologies, and CAD tools for VLSI. He is a co-author of the book titled *Low Power Digital CMOS Design* (Kluwer, 1995). He has served on the technical program committee of various conferences including ISSCC, DAC, ISLPED, ICCD, and VLSI Circuits Symposium. He is the technical program co-chair for the 1997 International Symposium on Low-Power Electronics and Design and for VLSI Design'98.

Dr. Chandrakasan received the NSF Career Development Award in 1995, the IBM Faculty Development Award in 1995, and the National Semiconductor Faculty Development Award in 1996. He received the IEEE Communications Society 1993 Best Tutorial Paper Award for the IEEE Communications Magazine paper titled, "A Portable Multimedia Terminal."