NETWORK DRIVEN MOTION ESTIMATION FOR PORTABLE VIDEO TERMINALS

Wendi B. Rabiner and Anantha P. Chandrakasan

Massachusetts Institute of Technology, Dept. of EECS
Room 38-107
Cambridge, MA 02139, USA

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 terminals. This paper presents an approach to reducing the power dissipation of wireless video terminals in a networked environment by exploiting the predictability of object motion. Since the location of an object in the current frame can be predicted from its location in previous frames, it is possible to optimally partition the motion estimation computation between battery operated portable devices and high powered compute servers on the wired network. This can achieve 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.

1. 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 motion estimation and motion compensation at the encoder to decorrelate successive frames and achieve high compression ratios. However, motion estimation is extremely computationally intensive, requiring over 65 thousand operations/pixel/second for ± 16 pel full search motion estimation and 30 fps video. 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, the computationally simple algorithms can produce very inaccurate motion estimates.

A standard method for reducing the temporal correlation without performing motion estimation is called conditional replenishment (CR). CR consists of representing each frame as the difference between it and the previous frame. This method requires very little computation and does not require the transmission of motion vectors. However, CR will only be effective if there is little or no scene motion.

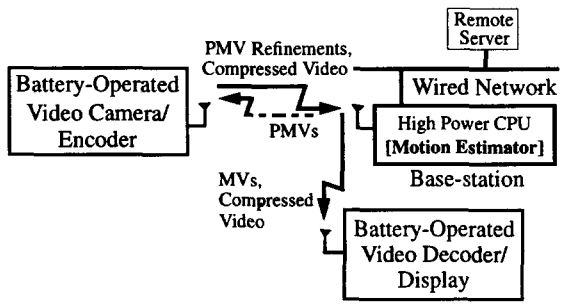

Figure 1. Video terminals in a networked environment.

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 an alternative method of performing motion estimation which achieves the bandwidth efficiency of full search encoder-based motion estimation and the power efficiency of CR.

2. USING NETWORK RESOURCES

2.1. 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. Similarly, 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 [1]. This predictability of the motion vectors can be exploited by removing the motion estimation function from the encoder and performing it at a high powered remote compute server, as described below.

Figure 1 shows a block diagram of wireless video terminals in a networked (wired and wireless) environment. 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 continuous from one frame to the next, the predicted motion vector for a macroblock

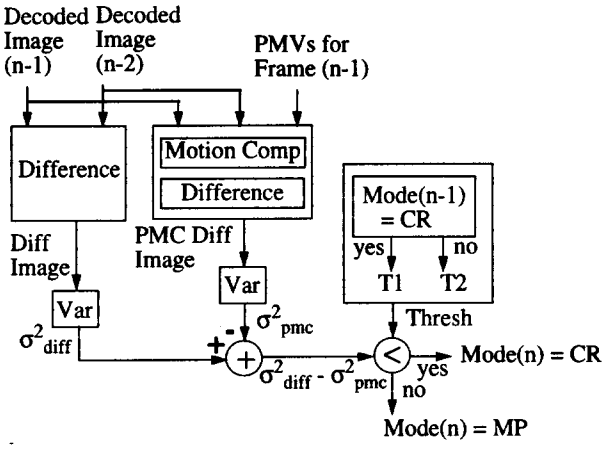

Figure 2. Algorithm for determining NDME system mode.

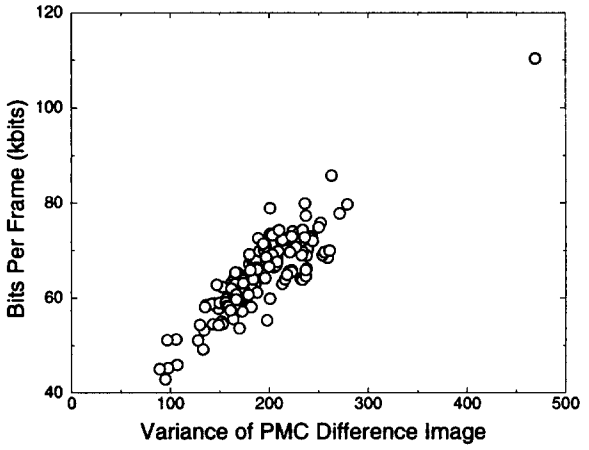

Figure 3. Bus Sequence. Number of bits required to code each frame versus the variance of that frame.

of the current frame is the "true" motion vector (found using a full search method on the decoded frames at the base-station) for that macroblock from the previous frame. These predicted motion vectors (PMVs) are sent to the encoder. Since the predicted motion vectors represent only an approximation of the motion of each macroblock, the encoder performs a local search (±1 pel) for the locally optimum motion vectors and performs motion compensation using these refined vectors. The encoder transmits the vector refinements and compressed video back to the base-station, which re-transmits the compressed video and the refined predicted motion vectors to the portable video decoder. The total data bandwidth for this system includes the PMVs being sent from the base-station to the encoder, the corrections to the PMVs being sent back to the base-station, and the compressed video. Employing motion prediction at a high powered network resource rather than motion estimation at the encoder greatly reduces the computational burden of the encoder.

2.2. Network Driven Motion Estimation

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 motion prediction and CR in order to obtain the maximum amount of compression for each frame. This method of reducing temporal correlation using an optimal combination 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 (PMC) difference image of the past frame, as shown in Figure 2. This metric is used because the variance of an image is a good measure of the amount of activity in that image [2]. Since the amount of data needed to code each of the difference images is proportional to the amount of activity in these images, as shown in Figure 3, the respective variances provide an efficient means of determining which mode should be used.

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 PMVs, when the system is operating in the motion prediction (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 to the base-station.

The first frame of a sequence is intra-coded, and the second frame is coded using CR, since the system has no previous frames from which to predict the sequence motion. After these initial frames, the variance metric described above determines which mode is used to code each frame. However, the CR mode is slightly preferenced over the MP mode when there is little scene motion, and, in order to keep the modes relatively consistent from frame to frame, the current mode is preferenced over switching modes. This results in thresholding the difference between the two variances, as shown in Figure 2[1]. This method of deciding the system mode continues until a scene change. Since network motion prediction breaks down at a scene change (which is unpredictable), the encoder is required to have a simple mechanism for detecting scene changes. When a scene change has been detected, the encoder must force an intra coded frame, and the entire network driven motion estimation process begins again.

The advantage of using NDME rather than CR can be seen in Figure 4, 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

[1]T1 was empirically chosen to be 100 and T2 was empirically chosen to be 0. Thus whenever the variance of the difference image is less than the variance of the PMC difference image, the system codes the next frame using CR.

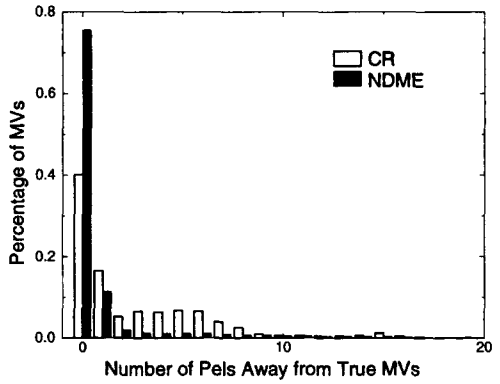

Figure 4. Bus Sequence. Distance between NDME/ CR MVs and "true" MVs.

close to the "true" motion vectors for the MPEG sequence "Bus". 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 network driven motion estimation.

3. EXPERIMENTAL RESULTS

Network driven motion estimation 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 × 240 pels), and "Suzie" (176 × 144 pels) as well as 150 frames of the sequences "Horse" and "Gym" (both 352 × 240 pels). Each of these sequences was coded with a constant quantization step size (MPEG QP = 10) using NDME, encoder-based motion estimation (EBME), and CR with 1 pel refinement. For all three coding methods, motion vectors were obtained to full pel accuracy and there was one motion vector generated for each 16 × 16 pixel macroblock. The average number of bits required to code each of the sequences using the different coding methods is shown in Table 1. These results show that, while coding with NDME does not achieve bit rates as low as coding with ideal EBME, it does achieve over a 30% decrease in average bit rate compared to coding with CR. Since the encoder performs the same computations for CR and NDME, this savings in bandwidth does *not* come at the expense of an increase in encoder power.

The decision about which mode to use to code the current frame is based on the variances of the CR and PMC difference images of the past frame. Figure 5a 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 motion prediction.

Figure 5b shows the number of bits required to code each frame of the "Bus" sequence using a constant quantization step size. The bit rate per frame of the NDME system is, on average, only 5-10 kbits worse than the ideal EBME

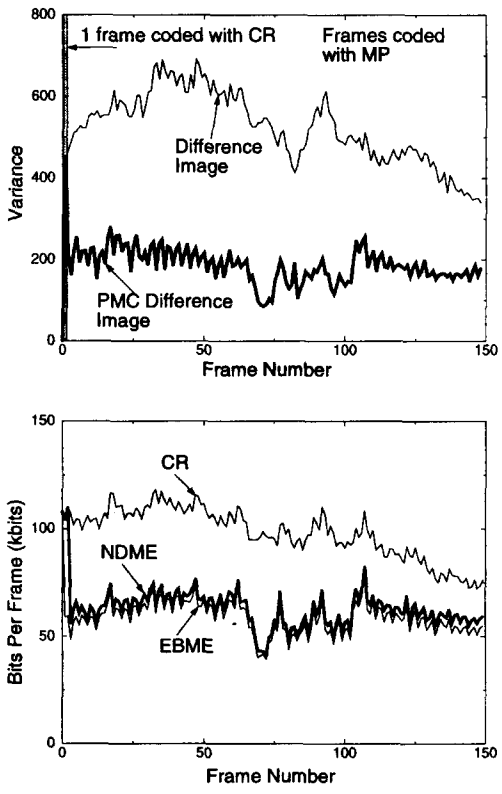

Figure 5. Bus Sequence. (a) Variance of CR and PMC difference images. (b) Number of bits to code each frame with fixed QP.

system and 25-50 kbits *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. Similar results were obtained for the "Flower" and "Horse" sequences, where, after the initial frames, the NDME system codes the remainder of these sequences using motion prediction.

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 Figure 7a for the "Tennis" sequence. For the first few frames, the variances of both difference images are very similar. In this case, the NDME system preferences using the CR mode since no motion vectors need to be transmit-

Table 1. Average number of kilobits per frame.

Sequence	EBME	NDME		CR	
Bus	59.0	65.3	(x1.11)	98.2	(x1.66)
Flower	71.1	74.9	(x1.05)	113.1	(x1.59)
Tennis	23.3	29.1	(x1.25)	34.5	(x1.48)
Suzie	2.5	2.6	(x1.04)	2.8	(x1.12)
Horse	23.5	26.5	(x1.13)	46.9	(x2.00)
Gym	23.3	27.7	(x1.19)	38.8	(x1.66)
Average	33.5	36.7	(x1.10)	55.6	(x1.66)

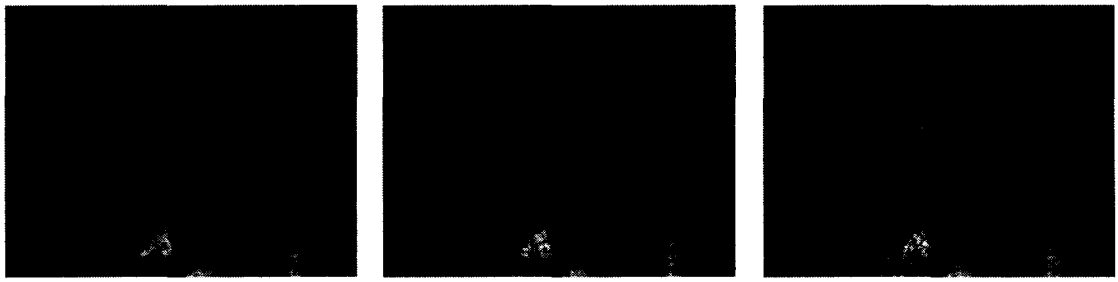

Figure 6. Suzie Sequence. (a) EBME (SNR=32.4 dB) (b) NDME (SNR=30.2 dB) (c) CR (SNR=27.5 dB)

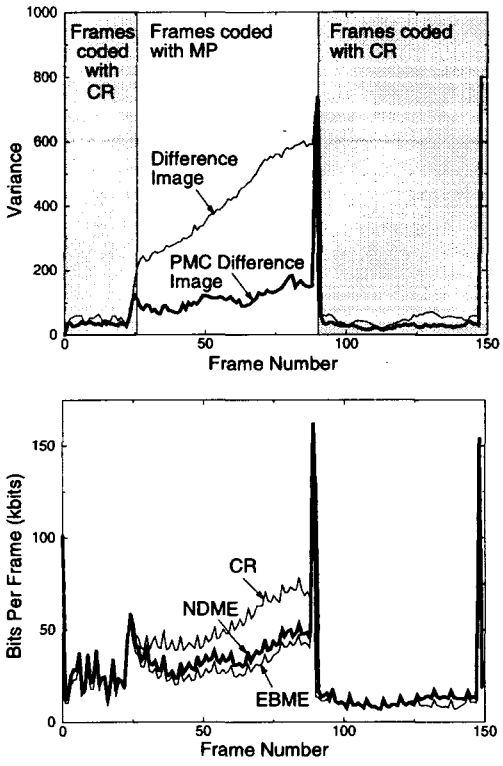

Figure 7. Tennis Sequence. (a) Variance of CR and PMC difference images. (b) Number of bits to code each frame with fixed QP.

ted 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 motion prediction, so the NDME system switches to this mode. Around frame 90 there is a scene change and the encoder automatically forces an intra-coded frame and a CR-coded frame. The remainder of the sequence has little motion, so the system remains in the CR mode. By adaptively switching between CR and motion prediction, 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.

Figure 7b shows the number of bits required to code each frame of the "Tennis" sequence. Only the middle frames of the sequence are coded using motion prediction; for these frames, the NDME system codes the images using approximately 5-10 kbits more than the ideal EBME system and 10-20 kbits fewer than the CR system.

Similar results were obtained for the "Suzie" and "Gym" sequences. The NDME system switches from CR to motion prediction back to CR for these sequences as well, since only the middle sections contain large amounts of motion.

These sequences were also coded using a constant bit rate to see the effects of the different coding algorithms on image quality. Figure 6, a frame from the "Suzie" sequence, shows the relatively small degradation in image quality obtained by using NDME rather than EBME and the large improvement in image quality obtained by using NDME rather than CR for this image.

4. CONCLUSION

Minimizing computation is the key to maximizing the battery life of portable devices. NDME achieves this minimization by reducing the encoder search window size from ± 16 pixels for full search encoder-based motion estimation to ± 1 pixel for the vector refinement. This results in a significant savings in power at the encoder with only a small increase in data bandwidth. By transferring the majority of the computation required for motion estimation from the power-limited portable video terminals to a high powered system on the network, efficient video coding can be achieved using a minimum amount of encoder power.

ACKNOWLEDGMENTS

This work is funded by the ARL Federated Lab program under contract #DAAL01-96-2-0001. Wendi Rabiner is supported by an NSF Fellowship.

REFERENCES

[1] S. Kim and C.-C. Kuo, "A Stochastic Approach for Motion Vector Estimation in Video Coding," *Neural and Stochastic Methods in Image and Signal Processing III, Proc. SPIE*, Vol. 2304, July 1994, pp. 111-122.

[2] A. Puri and R. Aravind, "Motion-Compensated Video Coding with Adaptive Perceptual Quantization," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 1, No. 4, December 1991, pp. 351-361.