THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 58, No. 3, March 1979
Printed in U.S.A.

Motion-Compensated Television Coding: Part I

By A. N. NETRAVALI and J. D. ROBBINS

(Manuscript received August 29, 1978)

We present methods of estimating displacements of moving objects from one frame to the next in a television scene and using such displacements for frame-to-frame prediction. Displacement is estimated by a recursive algorithm which seeks to minimize a functional of the prediction error. Several simplifications of the algorithm are presented which make it attractive for hardware implementation. Performance of the algorithm is evaluated by computer simulations on two sequences of moving images containing various amounts and types of motion. In both cases, the use of displacement-based (or motion-compensated) prediction results in bit rates that are 22 to 50 percent lower than those obtained by simple "frame-difference" prediction, which is used commonly in the interframe coders.

I. INTRODUCTION

Television signals are generated by scanning a scene several times a second even though there may not be any change in the scene from one frame to the next. This results in a considerable frame-to-frame redundancy in the signal. Existence of this redundancy has long been recognized, and several measurements have been made to quantify it. However, the first real demonstration of a frame-to-frame coder which used redundancy between successive frames was made in 1969 by Mounts.[1] Since then, several improvements have been made to the basic frame-to-frame encoder resulting in prototypes or real implementations of coder-decoder pairs.[2-5] These are the subjects of two excellent surveys,[6,7] the first one covering material up to 1972 and the second one up to 1978. As is evident from these works, most frame-to-frame coders are based on the following:

(*i*) Segmenting each television frame into two parts, one part which is the same as the previous frame and the other part (called the moving area) which has changed from the previous frame.

(*ii*) Transmitting two types of moving area information: (*a*) addresses specifying the location of the picture elements in the moving

area, and (*b*) information by which the intensities of the moving area elements can be updated.

(*iii*) Matching the coder bit rate to the channel rate. Since the motion in a real television scene occurs randomly and in bursts, the amount of information about the moving area will change as a function of time. To transmit it over a constant bit rate channel; (*a*) smooth out the transmitted information rate by storing it in a buffer prior to transmission, and (*b*) use the buffer fullness to regulate the encoded bit rate by varying amplitude, spatial, and temporal resolution of the television signal.

Intensities of the moving area picture elements are transmitted by predictive coding which sends frame difference, element difference, or line difference (or their combination) as the differential signal. Attempts have been made to optimize the picture quality within the constraints of the buffer size and the channel rate.

Simultaneously with these implementations, computer simulations have been used to explore other improvements of the frame-to-frame coders. It has long been recognized[8,9] that, if an estimate of the translation of an object is obtained, more efficient predictive encoding can be performed by taking differences of elements in the moving areas with respect to elements in the previous frame that are appropriately translated. We refer to such schemes as "Motion-Compensated Coding Schemes." Their success depends obviously on the following: (*i*) the amount of purely translational motion of objects in a real television scene,* (*ii*) the ability of an algorithm to estimate the translation with an accuracy that is desirable for good prediction of intensity, and (*iii*) robustness of the displacement estimation algorithm when amplitude, spatial, and temporal resolution of the transmitted picture are lowered due to buffer fullness.

Several methods of estimating the displacement of an object in a television scene have been proposed. Methods of point-by-point correlation[9] or pattern matching used in scene analysis[10–12] appear to be too complex for present-day implementation, especially if displacement needs to be defined with resolution finer than the sampled grid of a television frame. Simpler displacement estimation techniques[13,14] utilize the relation between the spatial differential and temporal differential signals. They would be easier to implement. Another approach is adaptive linear prediction using elements in the previous frame (or field) which are displaced in all directions (horizontal left, right; vertical up, down) by a certain maximum amount and adapting the coefficients to minimize an intensity error function.[15] All these techniques assume

* So that the television camera sees the objects in pure translation, it is also necessary that the lighting be uniform in the camera field of view.

that the displacement is constant within a block of picture elements. This assumption presents difficulties in scenes with multiple moving objects, occluding objects, as well as different parts of the same object moving with different displacements. Of course, decreasing the size of the block makes this assumption more realistic, but then the quality of displacement estimate suffers.

The techniques proposed by Haskell[15] do not require an explicit estimate of translational displacement and are applicable even if the motion is not precisely translational. However, they also work on blocks of picture elements and require a matrix inversion in addition to other complicated operations and, therefore, do not appear to be easily implementable without a significant simplification.

We present several new techniques for estimating motion. They attempt to minimize recursively a measure of the motion-compensated prediction error. Thus, given an ith estimate of displacement, we obtain the $(i + 1)$th estimate such that, in general, the motion-compensated prediction error resulting from $(i + 1)$th estimate is lower than that using the ith estimate. The recursive minimization is performed by a gradient or steepest descent algorithm. Considerable freedom exists in the specific choice of this algorithm. Our choices have been guided primarily by a desire to implement this algorithm in real time.

In Section II, we present a derivation of several motion-estimation algorithms and describe some simulations to evaluate their performance. Section III contains modifications of one of the algorithms of Section II for use in a frame-to-frame coder. Here we evaluate the performance of the algorithm in the context of a frame-to-frame coder. Section IV contains a discussion and the conclusions of our study. Many enhancements of the algorithm are possible, and some of these will be presented in a companion paper.[16]

II. MOTION ESTIMATION

In this section, we derive some simple algorithms for estimating motion. They attempt to minimize recursively a certain quantity (function of the motion estimation error). If the changes in successive television frames are due to translation of an object, then the algorithm iterates in a gradient or steepest descent direction such that the consecutive estimates converge to an estimate of translation. A proof of convergence of such a scheme under certain assumptions is given in the appendix and is supported by a large number of computer simulations presented at the end of this section.

2.1 Motion estimation in a block of pels

We mentioned in the introduction that most algorithms in the literature for estimating translation of an object from a television scene

assume that the translation is constant within a block of picture elements (pels). We start with one such algorithm developed by Limb and Murphy[13] and Cafforio and Rocca[14] and show how it can be modified to obtain a better estimate of translation. This is done primarily to define a quantity which is fundamental to our recursive algorithm introduced in the next section.

Assuming a 2:1 interlaced raster format, let $I(\mathbf{x}, t-\tau)$ and $I(\mathbf{x}, t)$ be the intensity values of the two successive frames as a function of spatial location $\mathbf{x}$ (a two-dimensional vector) and time t. The time between the two frames is τ. If an object moves in translation, then in the moving area (disregarding the uncovered background):

$$I(\mathbf{x}, t) = I(\mathbf{x} - \mathbf{D}, t - \tau), \tag{1}$$

where $\mathbf{D}$ is the translation vector of the object during the time interval $[t-\tau, t]$. The frame difference at spatial position $\mathbf{x}$ is given by

$$\begin{aligned} FDIF(\mathbf{x}) &= I(\mathbf{x}, t) - I(\mathbf{x}, t-\tau) \\ &= I(\mathbf{x}, t) - I(\mathbf{x} + \mathbf{D}, t), \end{aligned} \tag{2}$$

which can be written, for small $\mathbf{D}$, by Taylor's expansion about $\mathbf{x}$ as

$$FDIF(\mathbf{x}) = -\mathbf{D}^T \nabla I(\mathbf{x}, t) + \text{Higher Order Terms in } \mathbf{D}, \tag{3}$$

where ∇ is the gradient with respect to $\mathbf{x}$ and superscript T on a vector or matrix denotes its transpose. If the translation of the object is constant over the entire moving area (except for the uncovered background) and if the higher order terms in $\mathbf{D}$ can be neglected, then both sides of the above equation can be summed over the entire moving area to obtain a good estimate for translation. We recognize that ∇I can be taken to be a vector of element and line differences (*EDIF*, *LDIF*) if the intensity is available on a discrete grid as is the case in most coding situations. Using linear regression, we get $\hat{\mathbf{D}}$, an estimate of $\mathbf{D}$ as:

$$\hat{\mathbf{D}} = -\left[\sum_{\substack{\text{moving} \\ \text{area}}} \nabla I(\mathbf{x}, t) \cdot \nabla I(\mathbf{x}, t)^T\right]^{-1} \left[\sum_{\substack{\text{moving} \\ \text{area}}} FDIF(\mathbf{x}) \cdot \nabla I(\mathbf{x}, t)\right],$$

which can be written as

$$\hat{\mathbf{D}} = -\begin{bmatrix} \sum EDIF^2(\mathbf{x}) & \sum EDIF(\mathbf{x}) \cdot LDIF(\mathbf{x}) \\ \sum EDIF(\mathbf{x}) \cdot LDIF(\mathbf{x}) & \sum LDIF^2(\mathbf{x}) \end{bmatrix}^{-1} \cdot \begin{bmatrix} \sum FDIF(\mathbf{x}) \cdot EDIF(\mathbf{x}) \\ \sum FDIF(\mathbf{x}) \cdot LDIF(\mathbf{x}) \end{bmatrix},$$

where all the summations are over the entire moving area. This can be approximated* by assuming that

$$\sum_{\substack{\text{moving}\\ \text{area}}} EDIF(\mathbf{x}) \cdot LDIF(\mathbf{x}) = 0$$

and then

$$\hat{\mathbf{D}} = -\begin{bmatrix} \sum FDIF(\mathbf{x}) \cdot EDIF(\mathbf{x}) / \sum EDIF^2(\mathbf{x}) \\ \sum FDIF(\mathbf{x}) \cdot LDIF(\mathbf{x}) / \sum LDIF^2(\mathbf{x}) \end{bmatrix}. \tag{4}$$

This can be further approximated* by avoiding the multiplications in the sums as:

$$\hat{\mathbf{D}} = -\begin{bmatrix} \dfrac{\sum FDIF(\mathbf{x}) \text{sign}(EDIF(\mathbf{x}))}{\sum |EDIF(\mathbf{x})|} \\ \dfrac{\sum FDIF(\mathbf{x}) \text{sign}(LDIF(\mathbf{x}))}{\sum |LDIF(\mathbf{x})|} \end{bmatrix}, \tag{5}$$

where

$$\text{sign}(z) = \begin{cases} 0, & \text{if } z = 0 \\ \dfrac{z}{|z|}, & \text{otherwise.} \end{cases} \tag{6}$$

This algorithm is identical to one of the algorithms given by Limb and Murphy.[13] Its accuracy may be improved† by several modifications suggested by Limb and Murphy[13] and Cafforio and Rocca.[14]

We suggest another modification which improves the above motion estimator further. First, we note that the above estimates are good as long as $\mathbf{D}$ is small. As $\mathbf{D}$ increases, the quality of the approximation becomes poor. This can be overcome to some extent by linearizing the intensity function around an initial estimate of $\mathbf{D}$. This is possible in a television situation, where there is an estimate of $\mathbf{D}$ for every field. Thus, for the ith field, displacement estimate $\hat{\mathbf{D}}^i$ can be obtained by linearizing the intensity function around the displacement estimate for the $(i-1)$th field. This process results in the following recursion:

$$\hat{\mathbf{D}}^i = \hat{\mathbf{D}}^{i-1} + \mathbf{U}^i, \tag{7}$$

---

* We make no attempts to justify these two approximations. They are made merely to derive the algorithm given in Ref. 13. Perhaps, instead of the least squares, some other criterion may lead to the same result (i.e., eq. (5)) with fewer assumptions.

† The improvement suggested by Limb (Ref. 13) was simulated for comparison purposes. However, for the type of pictures used in Section 3.3, the performance did not change noticeably by incorporating the improvement.

where $\hat{\mathbf{D}}^{i-1}$ is an initial estimate of $\hat{\mathbf{D}}^i$ and $\mathbf{U}^i$ is the update of $\hat{\mathbf{D}}^{i-1}$ to make it more accurate, i.e., an estimate of $\mathbf{D} - \hat{\mathbf{D}}^{i-1}$.

We now define the quantity DFD $(\mathbf{x}, \hat{\mathbf{D}}^{i-1})$, called the displaced frame difference, which is analogous to $FDIF(\mathbf{x})$ used in (4) and (5),

$$\text{DFD}(\mathbf{x}, \hat{\mathbf{D}}^{i-1}) = I(\mathbf{x}, t) - I(\mathbf{x} - \hat{\mathbf{D}}^{i-1}, t - \tau) \tag{8}$$

DFD is defined in terms of two quantities: (i) the spatial location $\mathbf{x}$ at which it is evaluated and (ii) the displacement $\hat{\mathbf{D}}^{i-1}$ with which it is evaluated. Obviously, in the case of a two-dimensional grid of discrete samples, an interpolation process would be used to evaluate $I(\mathbf{x} - \hat{\mathbf{D}}^{i-1}, t - \tau)$ for nonintegral values of $\hat{\mathbf{D}}^{i-1}$. As defined, DFD has the property of converging to zero as $\hat{\mathbf{D}}^i$ converges to the actual displacement, $\mathbf{D}$, of the image. Following the same steps as were used in the derivation of eq. (5), we get:

$$\begin{aligned}\text{DFD}(\mathbf{x}, \hat{\mathbf{D}}^{i-1}) &= I(\mathbf{x}, t) - I(\mathbf{x} + \mathbf{D} - \hat{\mathbf{D}}^{i-1}, t) \\ &= -(\mathbf{D} - \hat{\mathbf{D}}^{i-1})^T \cdot \nabla I(\mathbf{x}, t) + \text{Higher Order Terms.}\end{aligned} \tag{9}$$

Neglecting higher order terms and making approximations similar to the above results in an estimate of $\mathbf{D} - \hat{\mathbf{D}}^{i-1}$ which, when combined with eq. (7), yields:

$$\hat{\mathbf{D}}^i = \hat{\mathbf{D}}^{i-1} - \begin{bmatrix} \dfrac{\sum \text{DFD}(\mathbf{x}, \hat{\mathbf{D}}^{i-1}) \text{sign}(EDIF(\mathbf{x}))}{\sum |EDIF(\mathbf{x})|} \\ \dfrac{\sum \text{DFD}(\mathbf{x}, \hat{\mathbf{D}}^{i-1}) \text{sign}(LDIF(\mathbf{x}))}{\sum |LDIF(\mathbf{x})|} \end{bmatrix}, \tag{10}$$

where the summations are again carried over the entire moving area.

This may be simplified slightly by noting that if the initial estimate of displacement $\hat{\mathbf{D}}^{i-1}$ has only integral components, then DFD$(\cdot, \cdot)$ can be computed without interpolation. Let $[\mathbf{D}]$ denote an integer approximation to $\mathbf{D}$. This can be obtained either by truncating or rounding both the components of the vector $\mathbf{D}$.

$$\hat{\mathbf{D}}^i = [\hat{\mathbf{D}}^{i-1}] - \begin{bmatrix} \dfrac{\sum \text{DFD}(\mathbf{x}, [\hat{\mathbf{D}}^{i-1}]) \cdot \text{sign}(EDIF(\mathbf{x}))}{\sum |EDIF(\mathbf{x})|} \\ \dfrac{\sum \text{DFD}(\mathbf{x}, [\hat{\mathbf{D}}^{i-1}]) \cdot \text{sign}(LDIF(\mathbf{x}))}{\sum |LDIF(\mathbf{x})|} \end{bmatrix}. \tag{11}$$

We describe the performance of both the above algorithms later in this section.

2.2 Pel-recursive estimation of motion

We mentioned in the introduction that there is an advantage in recursive algorithms which iterate on a pel-by-pel (or on a small block

of pels) basis, i.e., they revise their displacement estimate at every moving area pel. Such recursive algorithms overcome, to a large extent, the problems of multiple moving objects, as well as different parts of an object undergoing different displacements, provided the recursion has sufficiently rapid convergence. Since we intend to use the displacement estimator for predictive coding, our algorithm should in some manner attempt to minimize the resulting prediction error. Also, since the prediction error is calculated for transmission anyway, its use in the recursive estimation of displacement does not result in extra computations and is therefore advantageous. Thus, if a pel at location $\mathbf{x}_a$ is predicted with displacement $\hat{\mathbf{D}}^{i-1}$ and intensity $I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t - \tau)$, resulting in prediction error $\mathrm{DFD}(\mathbf{x}_a, \hat{\mathbf{D}}^{i-1})$, the estimator should try to produce a new estimate, $\hat{\mathbf{D}}^i$, such that $|\mathrm{DFD}(\mathbf{x}_a, \hat{\mathbf{D}}^i)| \leq |\mathrm{DFD}(\mathbf{x}_a, \hat{\mathbf{D}}^{i-1})|$. To this end, we attempt to recursively minimize $[\mathrm{DFD}(\mathbf{x}, \hat{\mathbf{D}})]^2$ at each moving area element using a gradient type of approach. For example,

$$\begin{aligned}\hat{\mathbf{D}}^i &= \hat{\mathbf{D}}^{i-1} - (\epsilon/2)\nabla_{\mathbf{D}}[\mathrm{DFD}(\mathbf{x}_a, \hat{\mathbf{D}}^{i-1})]^2 \\ &= \hat{\mathbf{D}}^{i-1} - \epsilon\mathrm{DFD}(\mathbf{x}_a, \hat{\mathbf{D}}^{i-1})\nabla_{\mathbf{D}}\mathrm{DFD}(\mathbf{x}_a, \hat{\mathbf{D}}^{i-1}),\end{aligned}$$

where $\nabla_{\mathbf{D}}$ is the gradient with respect to displacement $\mathbf{D}$ and ϵ is a positive scalar constant. The gradient $\nabla_{\mathbf{D}}$ may be evaluated using the definition of DFD and noting that

$$\nabla_{\mathbf{D}}(\mathrm{DFD}(\mathbf{x}_a, \hat{\mathbf{D}}^{i-1})) = + \nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t - \tau), \tag{12}$$

where ∇ is the gradient with respect to x. This gives us

$$\hat{\mathbf{D}}^i = \hat{\mathbf{D}}^{i-1} - \epsilon\mathrm{DFD}(\mathbf{x}_a, \hat{\mathbf{D}}^{i-1})\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t - \tau), \tag{13}$$

where DFD and ∇I are evaluated by interpolation for nonintegral $\mathbf{D}^{i-1}$. A significant reduction in computation of ∇I is achieved by quantizing $\hat{\mathbf{D}}^{i-1}$ to an integral value. Thus, if $[\hat{\mathbf{D}}^{i-1}]$ represents a rounded or truncated value of each of the components of $\hat{\mathbf{D}}^{i-1}$, then the estimator of eq. (13) can be simplified to

$$\hat{\mathbf{D}}^i = \hat{\mathbf{D}}^{i-1} - \epsilon\mathrm{DFD}(\mathbf{x}_a, \hat{\mathbf{D}}^{i-1})\nabla I(\mathbf{x}_a - [\hat{\mathbf{D}}^{i-1}], t - \tau). \tag{14}$$

It should be pointed out that ∇_D could have been evaluated using (9), resulting in an estimator in which ∇I is evaluated at $(\mathbf{x}_a, t)$ instead of $(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t - \tau)$ as above. This second method implies an assumption regarding the expansion of $I(\cdot, \cdot)$, which may not be valid if $D - \hat{\mathbf{D}}^{i-1}$ is large. Also, there is no difference in the computational complexity if it is assumed that a linear interpolation of $I(\mathbf{x}, t - \tau)$ is used to compute DFD, and the resulting displaced line and element differences are used to define ∇I of eq. (14).

It is interesting to observe that at every iteration we add to our old estimate a vector quantity parallel to the direction of the spatial gradient of image intensity and whose magnitude is proportional to

the motion-compensated prediction error. It may be seen from eq. (9) that if the displacement error ($\mathbf{D} - \hat{\mathbf{D}}^{i-1}$) is orthogonal to the intensity gradient ∇I, the displaced frame difference DFD $(\cdot, \cdot)$ is zero, giving a zero update for recursion of eq. (14). This may happen even though the object may have actually moved. However, this is not a failure of the algorithm, but rather is identical to the situation in which an intensity ramp is translated and only motion parallel to the ramp direction (∇I) is perceived. Motion perpendicular to the ramp direction is unobservable, and as such is arbitrary. It is only through the occurrence of edges with differing orientations in real television scenes that convergence of $\hat{\mathbf{D}}^i$ to actual $\mathbf{D}$ is possible.

The quantities involved in the above algorithm are shown in Fig. 1. An initial estimate of the displacement at pel $\mathbf{x}_a$, $\hat{\mathbf{D}}^{i-1}$, is to be updated using (14), yielding $\hat{\mathbf{D}}^i$. Using the initial estimate $\hat{\mathbf{D}}^{i-1}$ and $\mathbf{x}_a$, the samples in the previous frame in the neighborhood of spatial position $\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}$ are located (for example: samples b, c, d, e, and f). The

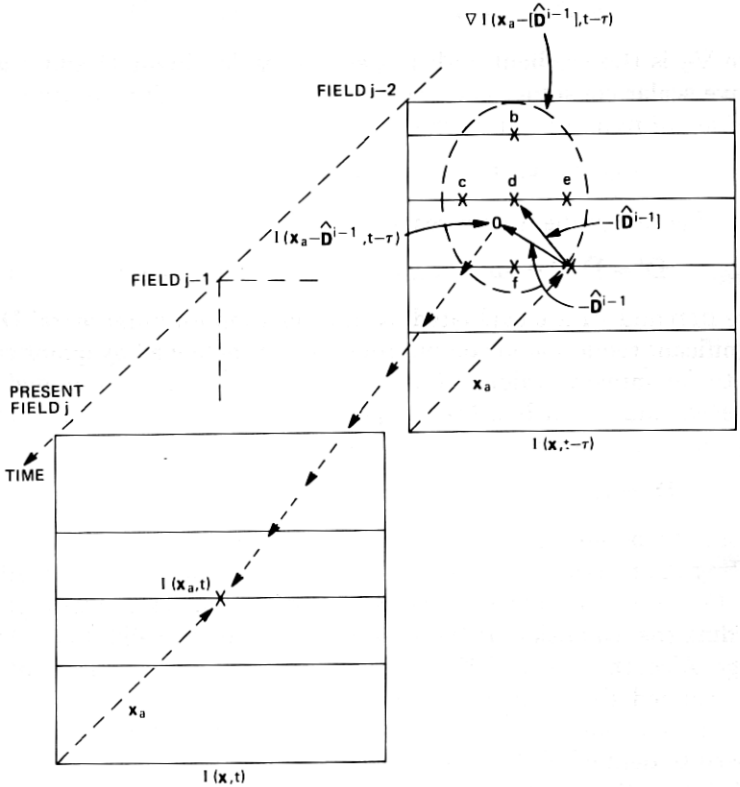

Fig. 1—Recursive motion estimation. Displacement estimate $\hat{\mathbf{D}}^{i-1}$ is updated at pel a. Gradient of intensity, $\nabla I(\mathbf{x}_a - [\hat{\mathbf{D}}^{i-1}], t - \tau)$, is obtained by using intensities at pels b, c, d, e, f in the field $(j - 2)$.

samples of this neighborhood are then used to compute the update term in conjunction with $I(\mathbf{x}_a, t)$. Thus, in Fig. 1, the components of intensity gradient may be approximated by

$$EDIF = (I_e - I_c)/2 \tag{15}$$

$$LDIF = (I_b - I_f)/2. \tag{16}$$

Similarly, $I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t - \tau)$ may be approximated by a two-dimensional linear interpolation using the intensities of the neighborhood. In the next section, we present several simplifications of this basic algorithm and adapt it for frame-to-frame coding. Recursive algorithms, using a model of the video process for motion estimation in a different context, are described in Ref. 17.

2.3 Motion estimator performance

In this section, we give simulation results for the first two motion estimators, algorithm I (eq. (5)) and algorithm II (eq. (11)). Obviously, the performance depends upon the type of scene and, even then, it is not clear how to measure the performance. We have used two types of scenes. The first is produced synthetically using the following formula:

$$I(\mathbf{x}, t) = \begin{cases} 127 & \text{if} \quad \|\mathbf{R}\| > 100 \\ 127\,(1 + e^{-.05\|\mathbf{R}\|}\cos(0.2 \cdot \pi \cdot \|\mathbf{R}\|), & \text{otherwise,} \end{cases} \tag{17}$$

where $\mathbf{R} = \mathbf{x} - (\mathbf{x}_o + \mathbf{D}t)$, 127 is the background intensity (on a scale of 0 to 255), $\mathbf{x}_o$ is the location of the center of the moving pattern at $t = 0$, $\mathbf{D}$ is the displacement of the pattern per frame (i.e., velocity), and $|\cdot|$ denotes the Euclidean norm. This formula produces a series of alternating light and dark concentric rings with exponentially decreasing radial intensity variation. This pattern was chosen because it contains a distribution of edges with different direction and height. $I(\mathbf{x}, t)$ was sampled both in time t and space $\mathbf{x}$ to produce a set of four frame sequences with strictly horizontal translation ranging from 0.5 to 6.0 pels per frame.

The second type of scene is also a collection of four frame sequences containing an object approximately in horizontal translation. These are the same as shown in Fig. 1b of Ref. 15. They contain a mannequin's head which was moved at various nominal speeds from 0.4 to 4.7 pels per frame.

The first type of scene was chosen so that an exact velocity was known and, therefore, the performance of motion estimators could be evaluated rather easily by measuring the deviation from this known velocity. A second measure of performance was obtained by computing the "match entropy" of the elements in the moving area of each field.

That is, using the displacement estimate $\hat{\mathbf{D}}^i$, obtained from two consecutive fields, the entropy of DFD($\mathbf{x}$, $\hat{\mathbf{D}}^i$) was computed using a linear two-dimensional interpolation over the moving area elements of frame $I(\mathbf{x}, t)$. This quantity is similar to the entropy of the prediction error; however, some future information is used in its calculation. Due to the presence of shadows and nonuniform illumination, the second type of scene does not possess a precisely defined velocity, and therefore only match entropy was used to evaluate the motion estimators for this type of scene.

In all the simulations of the results of algorithms I and II of this section and the next section, the moving area elements were determined by an algorithm similar to that described in Ref. 13. Likewise, the definition of $EDIF(\mathbf{x})$ and $LDIF(\mathbf{x})$ used in the simulations of algorithm I may be found in Ref. 13. For $EDIF(\mathbf{x})$, this involved the averaging of the element differences at $I(\mathbf{x}, t)$ and $I(\mathbf{x}, t-\tau)$. $LDIF(\mathbf{x})$ was computed in a similar manner. To extend this definition for use in algorithm II, the corresponding differences at $I(\mathbf{x}, t)$ and $I(\mathbf{x} - [\hat{\mathbf{D}}^{i-1}], t-\tau)$ were averaged.

The performance of algorithms I and II is given in Figs. 2a and 2b for the synthetic scenes. As seen in Fig. 2a, the error in the estimated velocity* using algorithm II is considerably smaller than that of algorithm I, especially at higher velocities. The peak observed in the estimates of algorithm I is perhaps due to the insensitivity of the algorithms to a per-frame shift equal to the period (10.0 units) of the synthetic moving image. The initial estimate of displacement for algorithm II was assumed to be 0. It is interesting to note that the curve of algorithm I is approximately the same as that of algorithm II after the first iteration; but after only three iterations, algorithm II converges to its curve in Fig. 2a. In Fig. 2b, the match entropies resulting from the estimates of Fig. 2a are shown. Again, the superiority of algorithm II over algorithm I is clearly seen. Also for comparison, we have included the entropies of $FDIF(\mathbf{x})$ of picture elements in the moving area. As expected, both algorithms I and II show significant improvements compared to frame differences. These conclusions for synthetic pictures remained generally unchanged when the direction of motion was changed and when random frame-to-frame noise of −40 dB (s/n ratio) was added to the scenes.

Performance of algorithms I and II for the second type of scene is given in Fig. 3. Here, since we do not have available an exact velocity, we only give the match entropy at the various nominal horizontal velocities used to create the scenes. While both algorithms result in match entropies considerably smaller than the frame difference en-

* Average of last four out of six estimates.

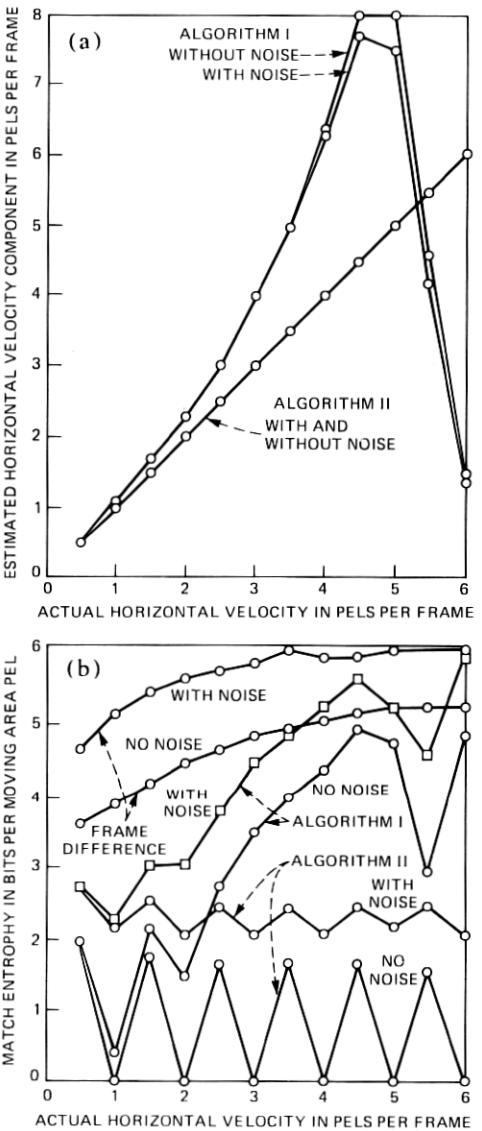

Fig. 2—Performance of two motion estimators which obtain one displacement estimate per field on synthetic pictures. Algorithm II results in considerable improvement over algorithm I. Zero entropy indicates that, in addition to a perfect estimate, there was no interpolation error in evaluating the prediction error. Estimated vertical velocity components (in lines/frame) were ± 0.3 for algorithm I and ± 0.02 for algorithm II.

tropy, we see that the performance of the two appears about the same. This result is perhaps due to the averaging of the displacement components over a large block (i.e., the moving area of an entire field). It should be mentioned that the performance of both the algorithms

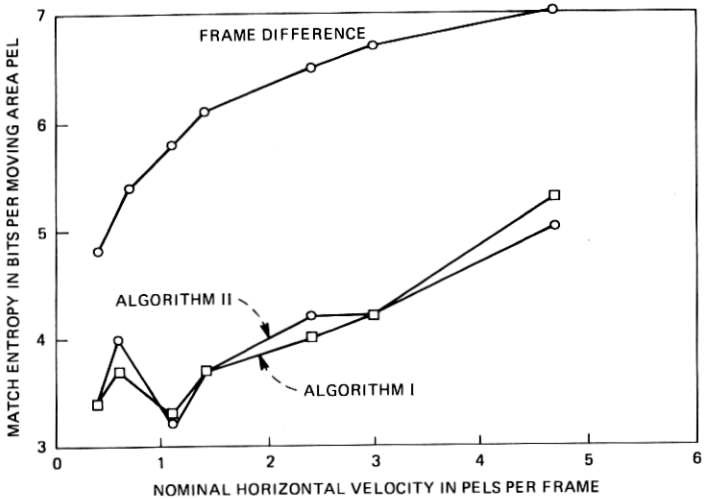

Fig. 3—Performance of two motion estimators which obtain one displacement estimate per field on moving mannequin.

can be improved by separating the uncovered background, as suggested by Cafforio and Rocca.[14]

III. CODER PERFORMANCE USING RECURSIVE DISPLACEMENT ESTIMATION

In the previous section, we developed motion estimators and discussed the performance of estimators which obtain one velocity estimate per field. We used scenes which had only one object, moving with a nearly uniform translational displacement. However, such scenes are unrealistic and, therefore, estimators which can dynamically adjust to motion of objects are more desirable. In this section, we first describe our recursive estimator in more detail and then evaluate its performance on scenes that are much more realistic. We then modify our basic estimator so that it can be incorporated in a frame-to-frame encoder and evaluate the coder performance. We note that we have paid little attention to a very important facet of frame-to-frame coders: resolution control using the contents of the buffer. It is important that motion estimation does not suffer immensely in lower resolution modes of the coder. Although some of these issues will be considered in part II,[16] more realistic performance evaluation can only be done using a hardware coder working on real scenes.

The first scene, called Judy, consists of 64 frames (2:1 interlaced fields) of 256 × 256 samples each, obtained at 30 times a second and sampled at Nyquist rate from a video signal of 1 MHz bandwidth, and contains head-and-shoulders view of a person engaged in a rather active conversation. The portion of a frame classified as moving area

varies from 15 to 51 percent. Also, the motion is not translational, and different parts of the scene move differently (such as lips, eyes, and head). Four frames of this scene are shown in Fig. 4.

The second scene, called Mike and Nadine, consists of 64 frames with the same resolution as the scene Judy and contains a panned full body view of two people briskly walking around each other on a set with severe nonuniform illumination. The percentage of a frame classified as moving area varied from 92 to 96 percent. Four frames from this sequence are shown in Fig. 5.

3.1 Basic estimator performance

The basic recursive estimator consists of the following: the displacement estimate is updated at each moving area picture element using eq. (14), i.e.,

$$\hat{\mathbf{D}}^i = \hat{\mathbf{D}}^{i-1} - \frac{1}{1024} \cdot \mathrm{DFD}(\mathbf{x}_j, \hat{\mathbf{D}}^{i-1}) \cdot \nabla I(\mathbf{x}_j - [\hat{\mathbf{D}}^{i-1}], t - \tau), \qquad (18)$$

where $\hat{\mathbf{D}}^i$ is the estimated displacement of moving area pel $\mathbf{x}_j$, $\hat{\mathbf{D}}^{i-1}$ is the last estimate formed prior to pel $\mathbf{x}_j$ during the pel-by-pel, line-by-line (interlaced raster scan order) iteration through the moving area, ∇I is the spatial gradient of the intensity, approximated by *EDIF* and *LDIF* defined in (15) and (16), and $[\hat{\mathbf{D}}^{i-1}]$ denotes rounded $\hat{\mathbf{D}}^{i-1}$. A two-dimensional linear interpolation is used to evaluate DFD (interpolation is discussed in detail in Section 3.3.3). Instead of using the previous frame for the intensity $I(\mathbf{x}, t - \tau)$, we have used the previous field. Relative advantages of this choice are discussed later. Also, both the horizontal and vertical components in the displacement error estimate (i.e., the second term of the right-hand side of eq. (18)) are clipped at a magnitude of (8/128), so that the displacement from pel to pel is not allowed to change by more than 1/16 pel/field and 1/16 line/field. This avoids the possibility of rapid oscillations in displacement due to noise. The accuracy used in computation of displacement and interpolation is 1/128 pel (or line) per field.

In all the simulations of this section, with the exception of the results of algorithm I appearing in Fig. 7, the moving area picture elements were determined by the following rule: pel z (Fig. 6) is classified as a moving area pel if either

$$(i) |FDIF(z)| > T_2$$
or
$$(ii) |FDIF(z)| > T_1$$
and
$$|FDIF| \text{ at a, b, c, or d} > T_1$$
and
$$|FDIF| \text{ at A, B, C, or D} > T_1,$$

Fig. 4—Four frames of the scene Judy.

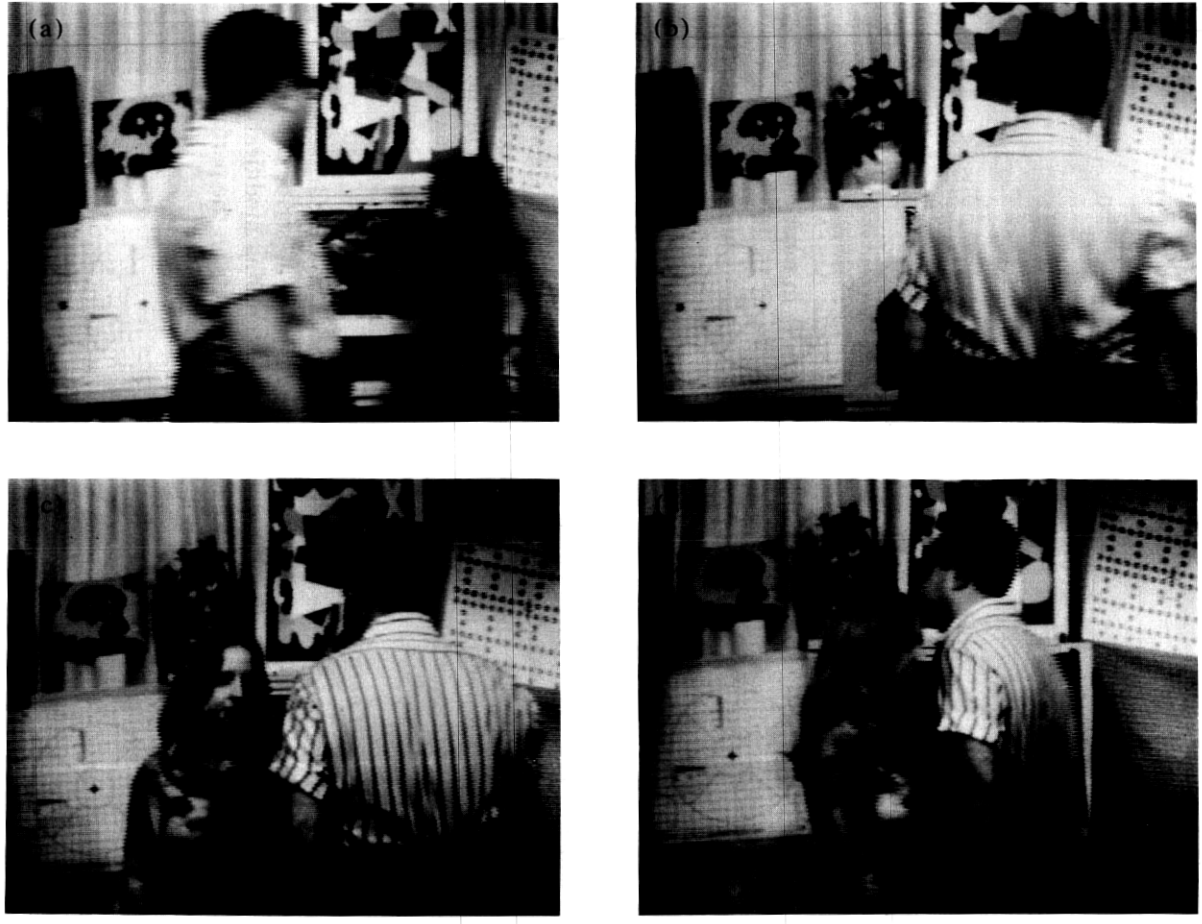

Fig. 5—Four frames of the scene Mike and Nadine.

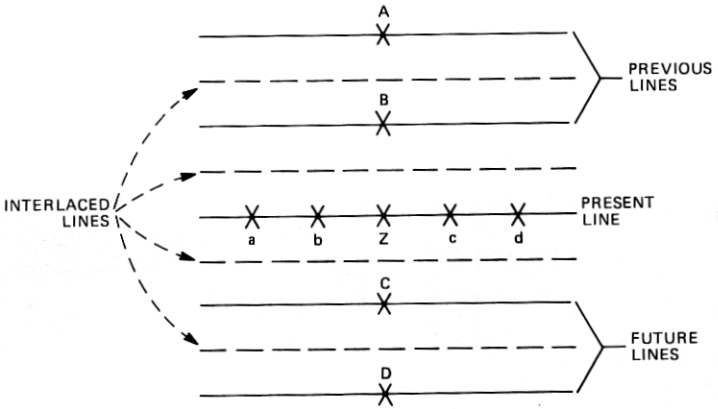

Fig. 6—Configuration of pels used in the moving area segmentor.

where T_1 and T_2 are two thresholds, and $T_2 \geq T_1$ in most cases. This segmenter is quite similar to the one in Refs. 4 and 15. It overcomes, to a large extent, the effects of frame-to-frame noise which otherwise produce a large number of isolated moving area elements. For the basic estimator, the moving area was segmented with thresholds $T_1 = 4$ and $T_2 = 255$ (on a scale of 0 to 255).

The performance of this basic estimator is given in Fig. 7 for the sequence Judy. Also for comparison, we have included the entropies of the moving area frame differences and the match entropy of algorithm I of Section II. On the average, match entropy for the recursive estimator was lower by about 1.4 bits/moving area pel than that of the frame differences, while algorithm I only resulted in approximately half of this decrease.*

3.2 Basic coder performance

In order to incorporate the estimator as a part of a coder, several other choices have to be made. In this section, we describe a basic coder and its performance. The next section contains modifications and simplifications of the basic coder and their effects on the performance. The basic coder consists of the following:

3.2.1 Displacement estimator

In the predictive coder, since the displacement estimate is not transmitted to the receiver, it has to be derived from the previously encoded and transmitted data. We derive a displacement estimate for a previously transmitted pel and use it for computation of the predic-

* Results in Fig. 7 for algorithm I were obtained by using a slightly different moving area segmenter given in Ref. 13.

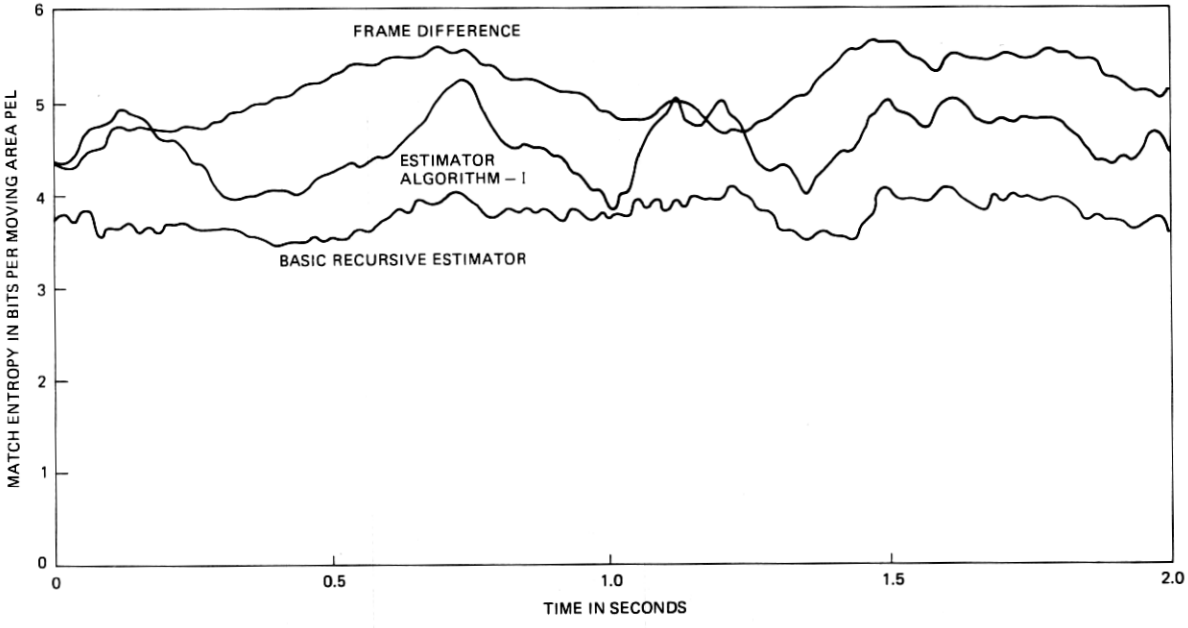

Fig. 7—Performance of basic recursive estimator on the scene Judy. Compared to frame difference and the estimator algorithm I, which obtains one displacement estimate per field, the recursive estimator results in a lower and relatively constant entropy.

tion of the present pel. The two natural choices for the relative location of the displacement estimate are the pel above and the pel to the left of the predicted pel. We have chosen the previous line (i.e., the pel above in the same field), since it relieves hardware timing constraints that would have otherwise resulted. If the previous line pel is not a moving area pel, we use the initial displacement estimate that would have been used by the above pel if it were moving. Such use of displacement estimates for prediction is shown in Fig. 8. We see that several elements (e.g., elements $j + 1, j + 2, \cdots, j + 4$ of the present line) may be predicted with the same displacement estimate if the corresponding elements of the previous line are not moving area elements. Other details of the estimator remain the same as those given in Section 3.1 for the basic estimator.

3.2.2 Segmentation

Every field was first divided into two segments: moving area and background, using the segmentation strategy described in Section 3.1. For a good picture quality, the thresholds T_1 and T_2 were chosen to be 1 and 3 (on a scale of 0 to 255), respectively. Moving area was further divided into two segments: (*i*) compensable regions, where motion-compensated prediction is adequate and, therefore, no update information need be sent; and (*ii*) uncompensable regions which require an update (i.e., transmission of prediction error). This segmentation of the moving area was performed using the following rule:

(*i*) If the magnitude of the motion-compensated prediction error for a moving area pel is greater than 3 (out of 255), then it is called uncompensable.

(*ii*) To reduce the occurrence of isolated compensable pels, compensable pel runs of length 1, 2, and 3 between uncompensable pels were bridged by calling them uncompensable.

This segmentation is shown in Fig. 9.

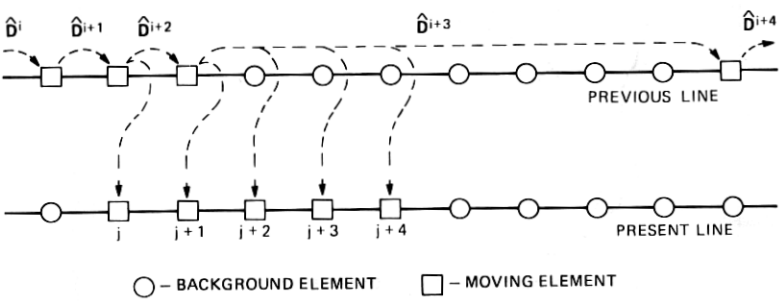

Fig. 8—Use of displacement estimates ($\hat{D}^i$) in prediction process. Displacement estimates are formed at every moving area pel in the same order as the scanning process. Displacement estimate of the previous line element is used for prediction of the present line picture elements.

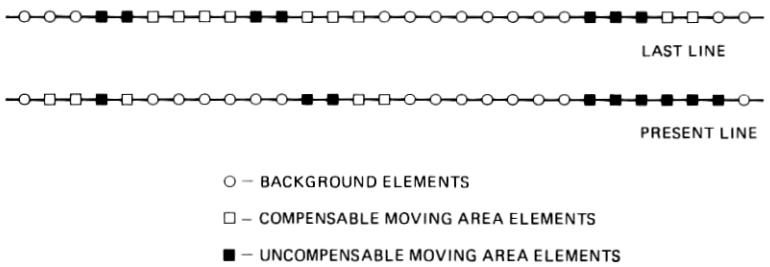

Fig. 9—Motion-compensated coder segmentation. Elements whose motion-compensated prediction error is lower than a threshold are called compensable and are not transmitted.

DECISION LEVELS	1,7,13,21,30,39,49,60,71,82,95,108,121,134,147,160,173,255
REPRESENTATIVE LEVELS	0,4,10,17,26,35,44,55,66,77,87,102,115,128,141,154,167,179

Fig. 10—Quantizer for basic coder. Only the positive side of the symmetric quantizer is shown.

3.2.3 Quantization

The prediction error was quantized using a 35-level symmetric quantizer shown in Fig. 10. Other quantizers were also investigated. The relative performance of the coder did not depend heavily on the quantizer. However, the picture quality degraded as would be expected by using a very coarse quantizer. The quantizer of Fig. 10 was chosen since it gave good picture quality, although the quantization error was clearly visible to a trained eye.

The performance of this basic coder was evaluated by computing the following quantities: (*i*) the entropy of the prediction errors of uncompensable pels, and the entropy of the run lengths defining the (*ii*) background pels, (*iii*) compensable pels, and (*iv*) uncompensable pels. The last three quantities are the addressing costs associated with not sending the prediction errors for certain pels. Figure 11 shows the overall bit rate associated with frame difference conditional replenishment encoding and motion-compensated encoding for both of our test sequences. Conditional replenishment encoding was done using the quantizer of Fig. 10 and moving area segmentation similar to that of the motion-compensated coder. The quality of pictures for both cases appeared to be approximately the same. Two characteristics of the motion-compensated coding are worth noting: (*i*) the average bits/field is reduced by about 44 percent; (*ii*) more importantly, during the times when large amounts of frame difference data are generated due to either rapid movement or a high percentage of moving area, the motion-compensated coder does significantly better. This type of behavior may help to decrease the occurrence of temporal overload

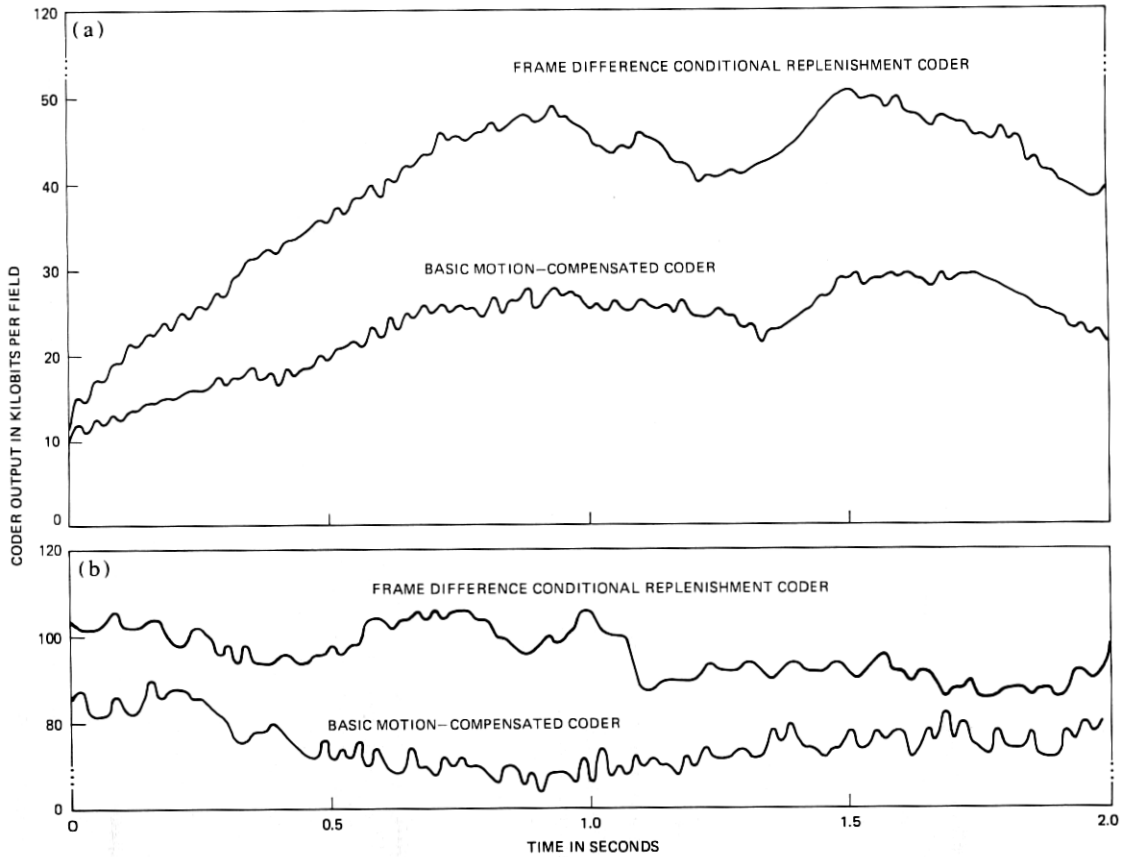

Fig. 11—Performance of basic motion-compensated coder and its comparison with a conditional replenishment coder for (a) the scene Judy and for (b) the scene Mike and Nadine.

associated with most conditional replenishment coders. Part of the reason for such a significant reduction in bit rate due to movement compensation is seen from Figs. 12 and 13 which show the results for conditional replenishment and motion-compensated coding, respectively. In these figures, the pel intensity is proportional to the bits required to code the prediction error for that pel. Uncompensable pels form a very small portion of the moving area and their prediction error is also small. For the scene Mike and Nadine, the output of each encoder is more than twice that obtained for Judy. This increase is mainly due to the panning of the camera. For this scene, intended as a severe test for the encoders, motion-compensated encoding resulted in a 22-percent reduction in average encoded bits.

Figure 14 shows a breakdown of the bits required for transmission of addresses and prediction error for scene Judy. It is interesting to note that, on the average, for the conditional replenishment coder the addressing bits comprise only about 15 to 30 percent of the total transmitted bits, but in the movement-compensated coder, since the magnitude of the prediction error is decreased, the addressing bits account for a much larger fraction of the total bits (50 to 65 percent). More efficient addressing methods are therefore desirable with motion compensation.

3.3 Variation in coder structure

In this section, we simplify the coder and evaluate the effect of various modifications on the coder performance.

3.3.1 Coder prediction

It is known[15] that, in conditional replenishment encoders, instead of using a frame differential prediction, line or element difference of the frame difference can be used with advantage. Thus in Fig. 8, for picture element $\mathbf{x}_{j+3}$, the conditional replenishment coder would send $[FDIF(\mathbf{x}_{j+3}) - FDIF(\mathbf{x}_{j+2})]$ as an element difference of frame difference prediction error. Similar modifications can be made to the motion-compensated coder. Here we transmit line or element differences of the displaced frame differences. Thus for picture element $\mathbf{x}_{j+3}$, we transmit $[\text{DFD}(\mathbf{x}_{j+3}, \mathbf{D}^{i+3}) - \text{DFD}(\mathbf{x}_{j+2}, \mathbf{D}^{i+3})]$ as an element difference of displaced frame difference prediction error. In cases such as pel $\mathbf{x}_j$ of Fig. 8, where the previous element is not in the moving area, their FD as well as DFD is assumed to be zero. The performance of this predictor modification is given in Fig. 15. As seen from this figure, element differencing improves both the frame differential and motion-compensated coders by about 5 to 15 percent.

3.3.2 Addressing of moving area

In motion-compensated coding, the addressing information takes up a large fraction of the total transmitted bits. We have therefore used

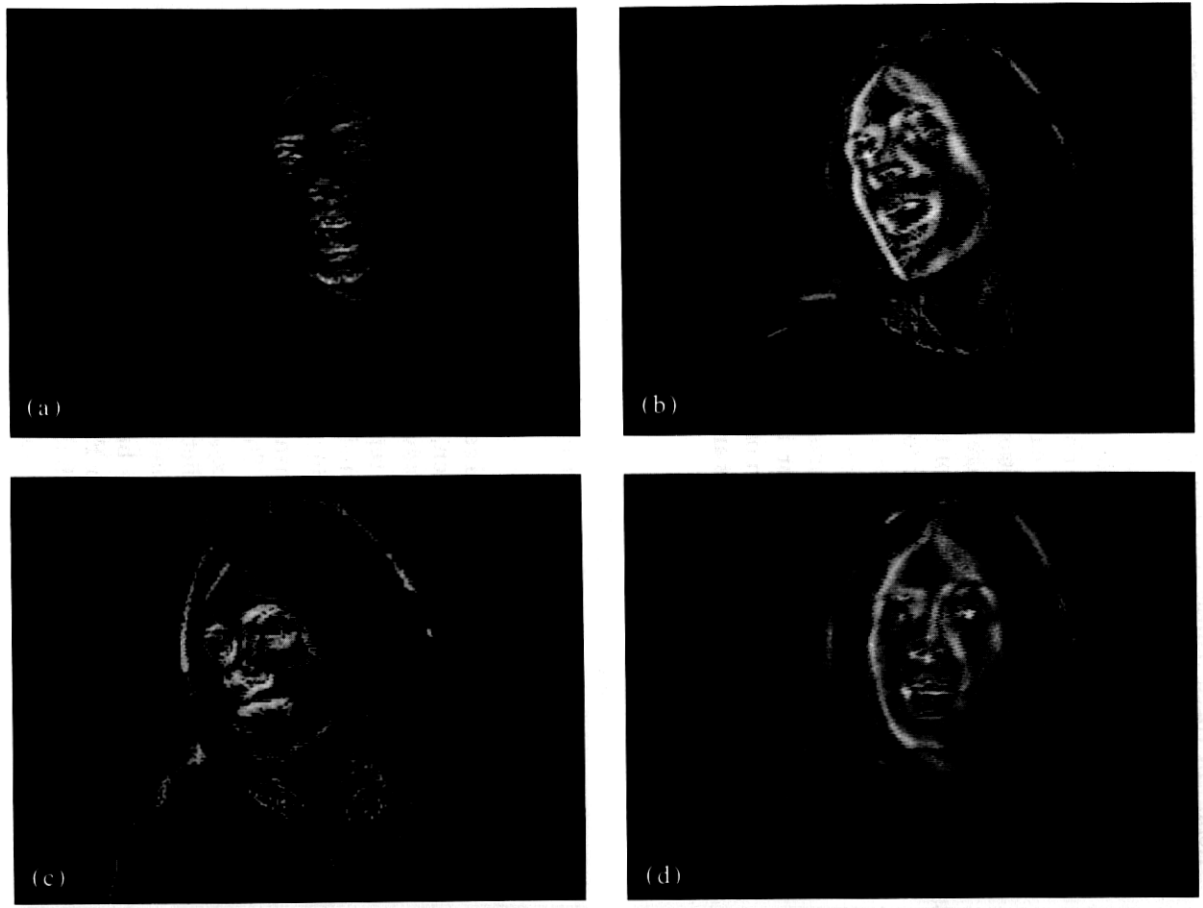

Fig. 12—Transmitted amplitude information for a conditional replenishment coder. Intensity of a picture element is proportional to the bits required to code the prediction error using frame difference prediction for four frames of scene Judy (Fig. 4). Addressing information is not included.

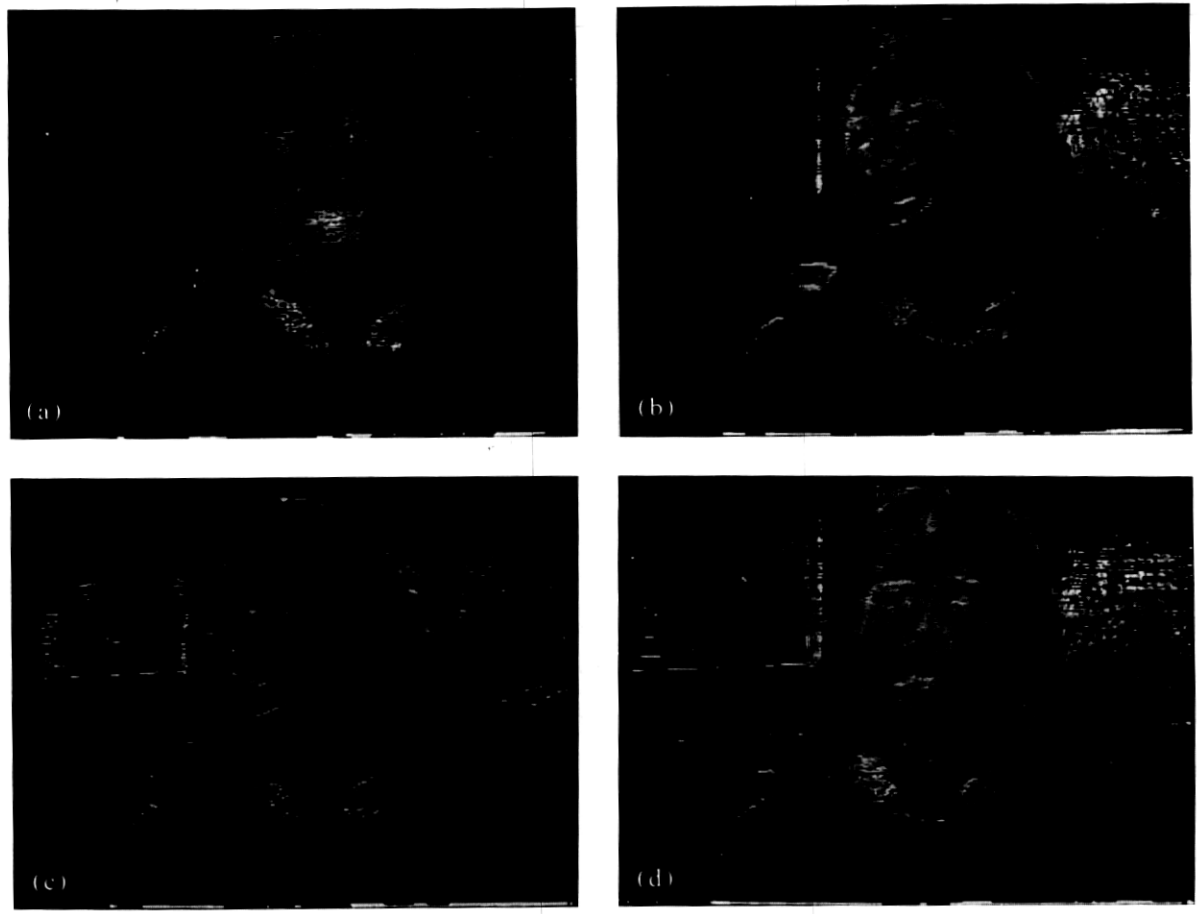

Fig. 13—Transmitted amplitude information for a motion-compensated coder. Intensity of a pel is proportional to the bits required to code the prediction error using the basic coder of Section 3.2 for four frames of scene Judy (Fig. 4). Addressing information is not included.

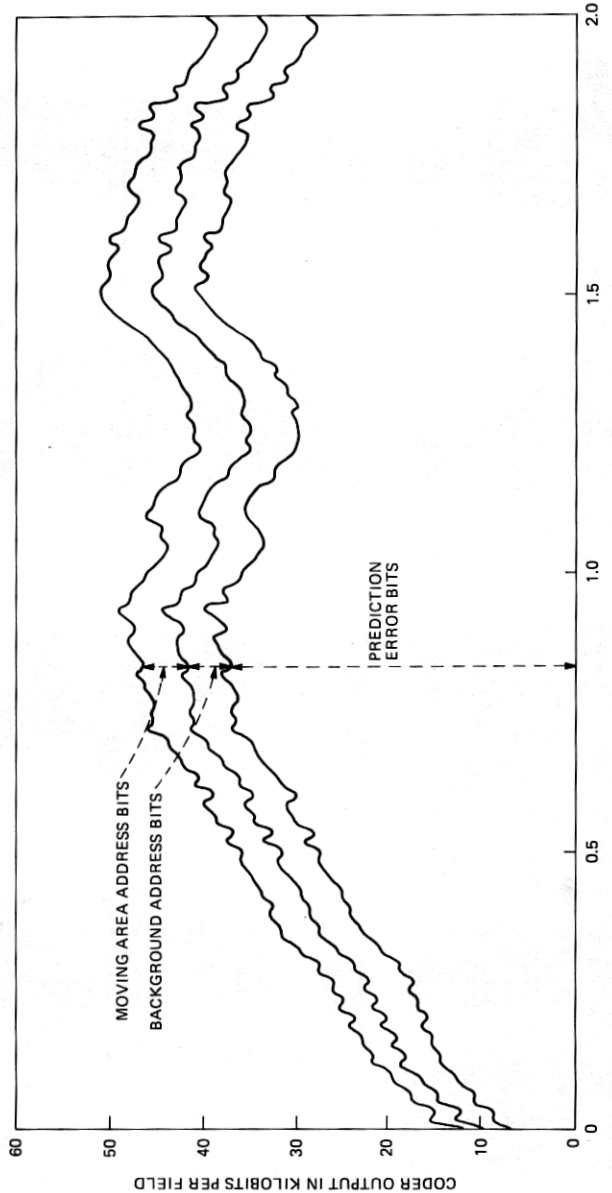

Fig. 14a—Components of conditional replenishment coder output. Approximately 80 percent of output is devoted to prediction error.

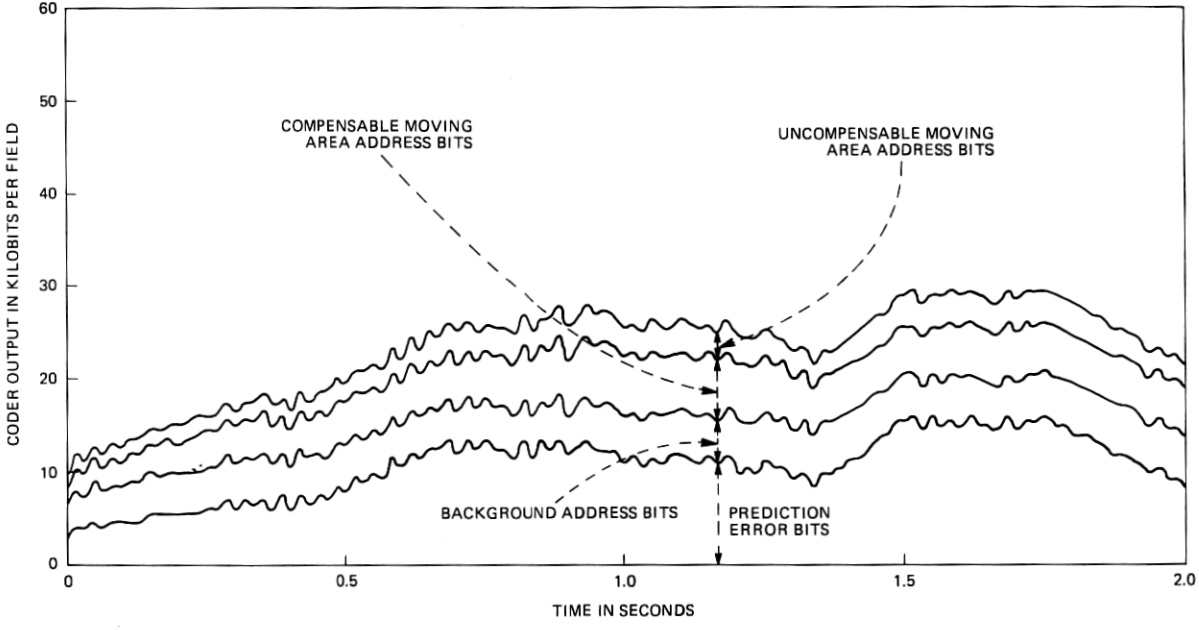

Fig. 14b—Components of motion-compensated coder output. Note that addressing information accounts for more than 50 percent of the coder's output, whereas the conditional replenishment output is mostly prediction error.

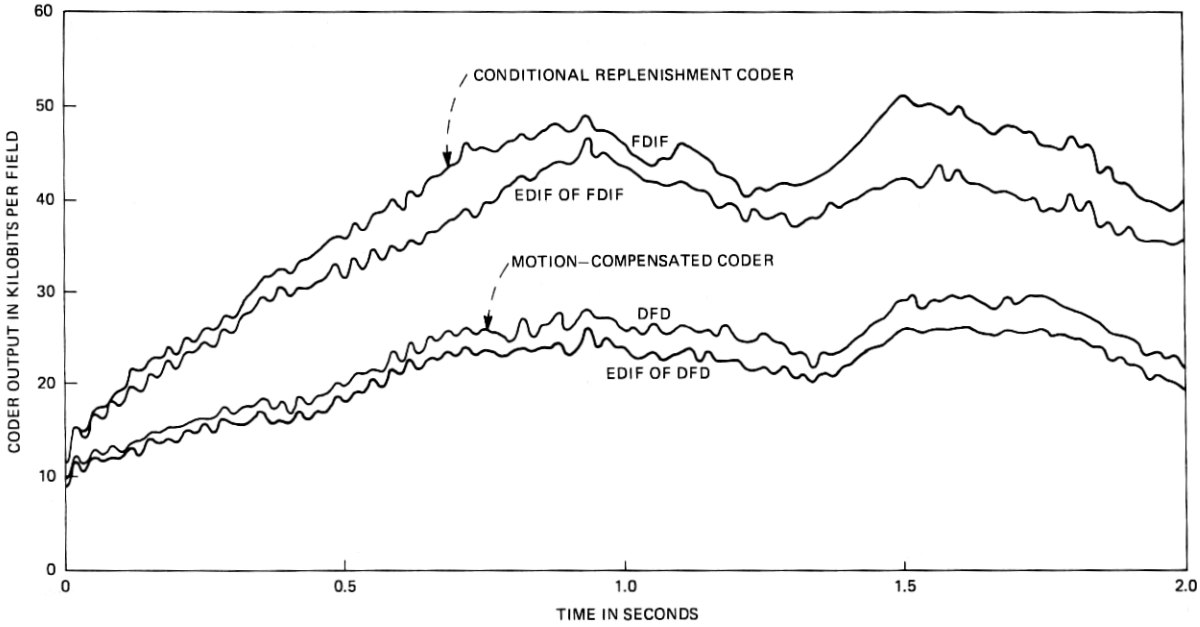

Fig. 15—Performance with predictor modification for sequence Judy. Element difference of either frame difference or displaced frame difference lower the entropy.

some known[18] addressing techniques and evaluated their effects on the total bit rate. As in Ref. 18, three techniques for addressing the moving area were tried. In the first technique, the beginning of each moving area was given an absolute address (8 bits) with respect to the beginning of the scan line, and the end of moving area cluster was given a special flag word different from all the prediction error codes. This is similar to the technique used in Ref. 2. The second technique[18] was to address the beginning of each cluster of moving area with respect to a similar cluster in the previous line as long as the magnitude of this differential address was less than ±8; otherwise, the cluster was addressed with respect to the beginning of the line. Maximum differential address of 8 was chosen after experiments with 2, 4, 16, and 32. The final technique is to address all the clusters as one-dimensional run lengths, as in our previous sections. The performance of these three addressing schemes for both the conditional replenishment and motion-compensated coders is given in Fig. 16 for sequence Judy. As expected, absolute addressing requires more bits than either differential or run-length addressing. Run-length addressing is the most efficient and decreases the overall bit rate by about 10 percent compared to the absolute addressing in a motion-compensated coder.

It should be noted that, to precisely evaluate the addressing cost associated with motion-compensated coding, it is not adequate to use only the run-length entropies of the three types of segments. Knowing the type of the last run, the receiver must be told, in some fashion, which of the two possible types the next run is. This information has been included in the simulations of Fig. 16. Comparing the motion-compensated coder using run-length addressing for moving area as in Fig. 16 with the results shown in Fig. 11 which does not include the next segment type of information, we see that this information accounts for about 8 percent of the bits.

3.3.3 Interpolation

The motion-compensated coder discussed above uses previous field intensities for interpolation rather than previous frame intensities. Use of previous field has two advantages: (*i*) the displacement values are generally smaller, since fields occur at every 1/60th of a second, whereas frames occur at 1/30th of a second, and (*ii*) in a hardware coder, since only a certain number of elements from the previous field or frame would be made available for interpolation with the same number of adjacent available elements, almost twice the amount of motion can be accommodated. On the other hand, one disadvantage of using the previous field intensities is that, for a perfectly horizontal motion, intensities have to be obtained from the previous field by interpolation (due to the interlace) which introduces error in the computation of DFD. Another factor, which appears to be a disadvan-

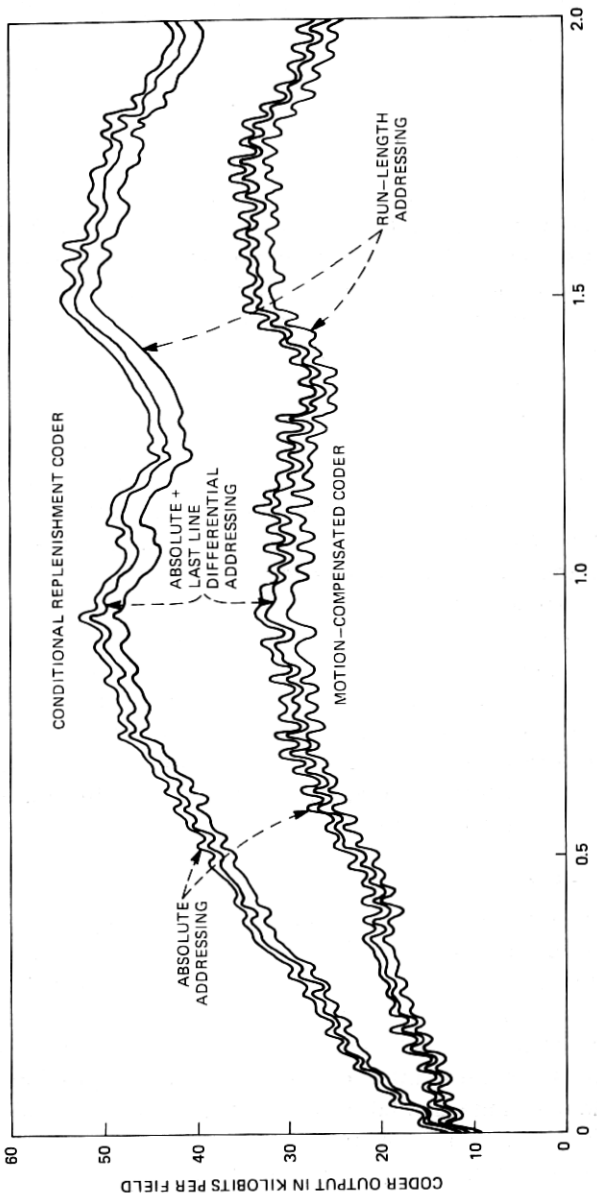

Fig. 16—Performance of moving area addressing schemes. Run-length addressing does better than absolute or differential addressing.

tage but which can be overcome, is the use of an alternate field dropping coder mode. To relieve a buffer overflow condition caused by a peak in coder output, it is advantageous to drop alternate fields as is done in conditional replenishment coders,[4,5] however; the previous field is then no longer available for motion estimation. In any case,we experimented with both the previous field and frame intensities and found that the use of previous field intensities resulted in entropies of unquantized moving area prediction errors which were 5 to 15 percent lower than those of previous frame intensities.

Of the many interpolation techniques possible, we restricted our attention to linear interpolation since our goal was a coder that could be implemented in real time. Thus, given the four surrounding pels as in Fig. 17, if the displacement $\mathbf{D}$ is written as the sum of an integral part $\mathbf{D}^I$, and a fractional part $\mathbf{D}^f$ with component magnitudes D_1^f and D_2^f, then the intensity can be interpolated by a standard two-dimensional linear interpolation as:

$$I = (1 - D_2^f)[(1 - D_1^f) I_D + D_1^f I_C] + D_2^f[(1 - D_1^f) I_B + D_1^f I_A], \tag{19}$$

where I is the interpolated intensity. It should be noted that eq. (19) has been used for all simulations previously described in this paper. Since this interpolation formula requires a large number of time-consuming multiplications, we tried another approach. We first select the three nearest (in the sense of the Euclidean norm) neighboring pels out of the four (e.g., pels B, C, and D in Fig. 17). This may be done by rounding $\mathbf{D}$ to form $\mathbf{D}^I$, thus locating the nearest neighbor, pel D, which is taken to be the corner of the three, then testing the sign of the rounding errors to locate the other two pels. The following formula is then used for interpolation:

$$I = I_D + D_1^f(I_C - I_D) + D_2^f(I_B - I_D). \tag{20}$$

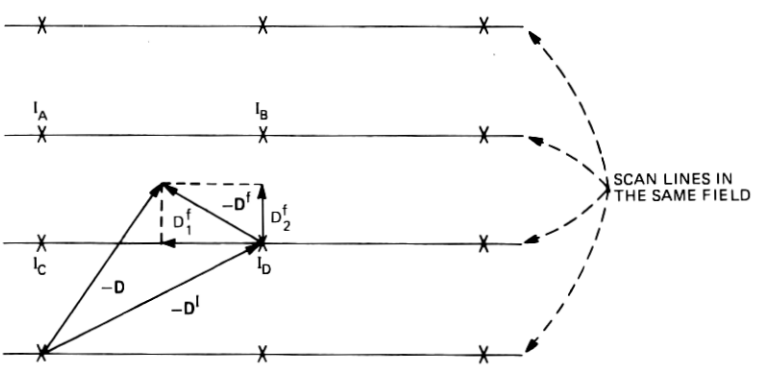

Fig. 17—Two-dimensional linear interpolation. Displacement $\mathbf{D}$ is decomposed into integral part $\mathbf{D}^I$ and fractional part $\mathbf{D}^f$.

Simulations using sequence Judy, comparing eqs. (19) and (20), showed very little differences in entropy, and since the interpolation of eq. (20) is significantly simpler, it is more suitable for a real-time implementation.

The precision with which the interpolation computation is performed was also varied. In particular, we used precisions of 2^{-n}; $n = 1, 2 \cdots, 7$ pel (or line) per field for displacement estimates, and found, as expected, that the average bits/field increased as the precision was decreased. The average increase in bits/field relative to a precision of 1/128 pel/field was as follows: negligible increase for precisions greater than 1/8 pel/field, 2 to 3 percent for 1/8 pel/field, 6 to 10 percent for 1/4 pel/field, and 15 to 25 percent for 1/2 pel/field. Thus, it appears that for interpolation, a resolution of 1/8 pel/field does an adequate job with much lower complexity compared to resolution of 1/128 pel/field. It should be noted that the recursion of eq. (18) was calculated by using a precision which is higher than the one used for the calculation of interpolation.

3.3.4 Estimator parameters

In eq. (18), the ϵ was taken to be 1/1024 and the update term was clipped to (1/16) pels/field. Both these quantities were varied over a wide range. Performance due to such variations is given in Fig. 18. This shows the robustness of the estimator. Of course, increase of both ϵ and the clipping level to a very high value results in a noisy estimator, but it gives the ability to adjust quickly to rapid changes in motion. On the other hand, a small value of ϵ allows us to converge to a finer value of displacement, thereby allowing low prediction error. A compromise between these two conflicting goals appears to be the values of ϵ and clipping level used in (18).

We mentioned earlier that the recursion of eq. (18) was calculated at a resolution of 1/128 pel/field. Simulation results on Judy indicate that, when the interpolation is calculated with resolution of 1/8 pel/field, the recursion of eq. (18) does not need to be calculated at resolutions higher than 1/16 pel/field, i.e, there is no significant improvement in performance by increasing the resolution beyond 1/16 pel/field in the calculation of displacement. To make the simulations realistic, all the computations were performed using integer arithmetic (on the computer).

Within our investigation into the effect of precision, the following simplification of the update term was simulated:

$$\hat{\mathbf{D}}^i = \hat{\mathbf{D}}^{i-1} - \epsilon \cdot \text{sign}(\text{DFD}(\mathbf{x}_j, \hat{\mathbf{D}}^{i-1})) \cdot \text{sign}(\nabla I(\mathbf{x}_j, [\hat{\mathbf{D}}^{i-1}])), \quad (21)$$

where sign of a vector quantity is the vector of signs of its components. The sign function, defined by (6), avoids the multiplication necessary for computation of the update term. Thus the update of each displace-

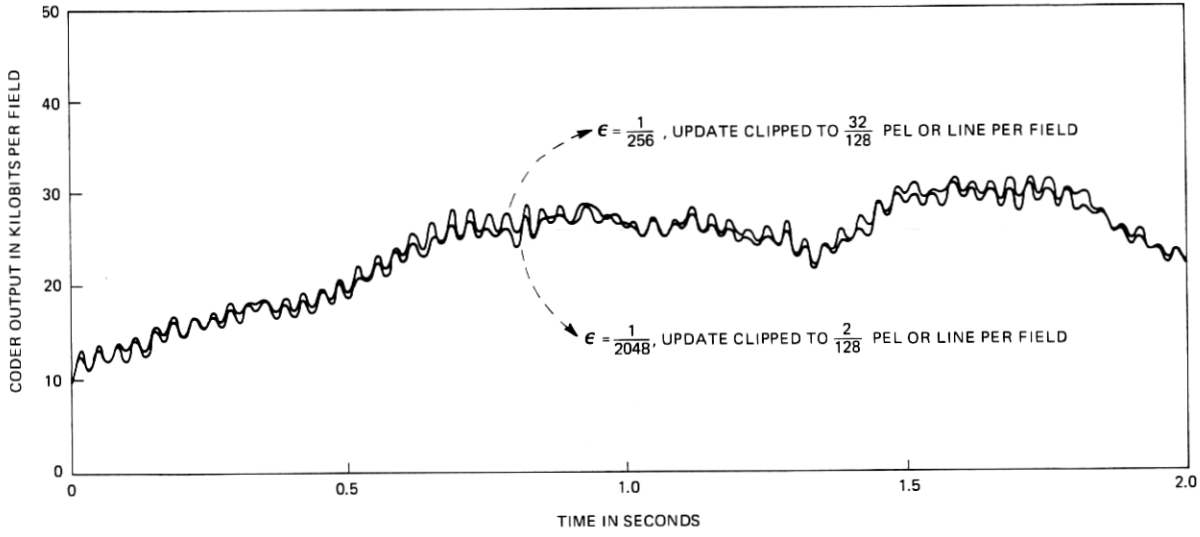

Fig. 18—Performance change due to the variation of rate and the maximum level of displacement update in motion estimator for sequence Judy. Results show robustness to variation of these parameters.

ment component from one picture element to the next consists of only three possibilities; 0 or $\pm\epsilon$. The performance using such a simplification is given in Fig. 19. It is clear from this figure that the performance in terms of bits/field is virtually unaffected by using the simplified update term of eq. (21).

Earlier we mentioned that we computed the displacement from the previous line of video and used it for computation of the motion-compensated prediction of the present line. This was done so that the displacement computation would have enough time to finish in a real-time encoder. It is possible, at least in simulations, to use the displacement of the previous moving area element in the same line for computation of the motion-compensated prediction of the present pel. This variation improves the performance of the encoder by about 6 to 10 percent in terms of bits/field.

The last simplification we attempted was in the calculation of the intensity gradient. Our hope was to use the same element and line differences which would be used in the computation of the DFD, so that this computation could be shared. Figure 20 shows three possible configurations of picture elements and the corresponding intensity gradients that were simulated. In the first and second configuration, the elements are those used for the interpolation of eqs. (19) and (20), respectively, while configuration III replaces pel D of configuration II with two pels further away. Coder performance using the three interpolation configurations was similar, and since the second configuration is the simplest, it is more suitable for hardware implementation.

3.4 Combined effects of simplifications

We have so far evaluated the changes in coder performance due to each individual simplification. In this section, we evaluate the combined effects of the following simplifications:

(*i*) The moving area is specified with absolute addresses and the next segment-type information is transmitted as in part 2 of the last section.

(*ii*) The sign $(\cdot)\cdot\text{sign}\,(\cdot)$ estimator of eq. (21) is used for displacement estimation.

(*iii*) The three-point interpolation of eq. (20) is used for intensity prediction and displacement estimation at a precision of ⅛ pel/field.

(*iv*) The line and element differences of interpolation eq. (20) are used as the intensity gradient components of eq. (21).

As seen from the simulated performance of the coder in Fig. 21, the modifications have had only a minor effect on the encoded bits/field. The simplified basic coder represents what we believe to be a practical algorithm for real-time implementation. A simplified block diagram of this coder appears in Fig. 22. The inner loop, which comprises the

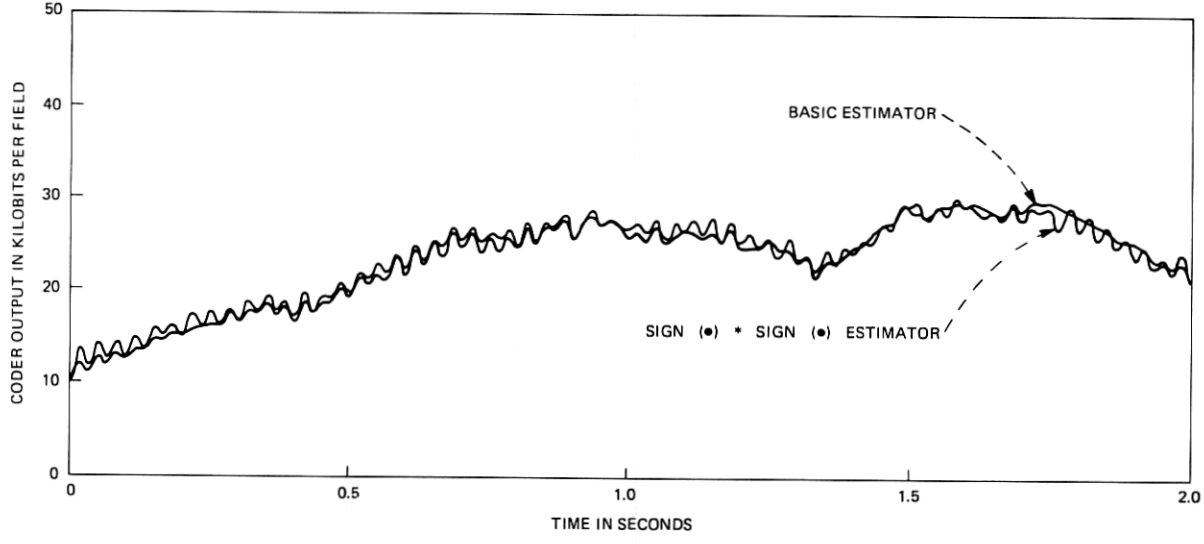

Fig. 19—Coder performance using a sign (·) * sign (·) motion estimator update for sequence Judy.

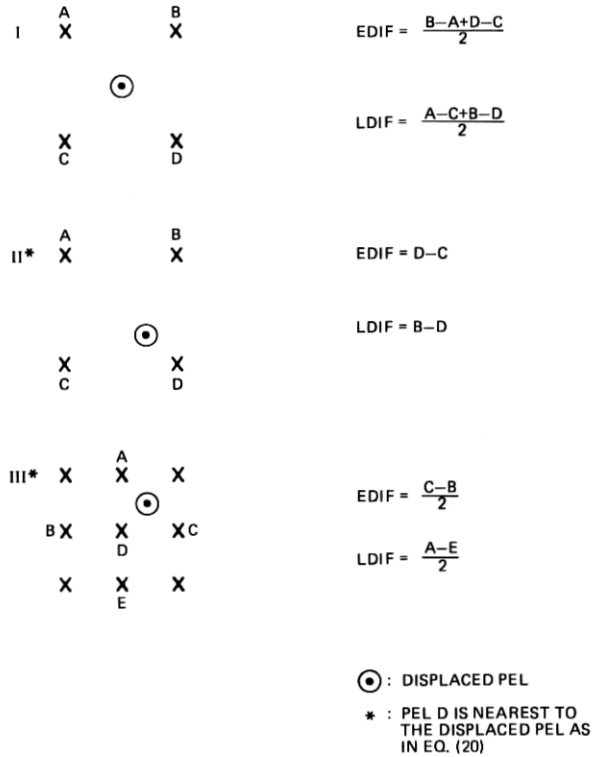

Fig. 20—Possible pel configurations for computing intensity gradient.

displacement estimator, attempts to match interpolated intensities of the last field with those of the last line by adjusting the displacement. The outer loop uses the displacement estimates to predict the video input with interpolated last field intensities. Based on frame difference, the segmentor limits displacement estimation (switch A) and motion-compensated prediction (switch B) to the moving areas. The segmentor also controls the selection of errors for transmission with switch C. Hardware for controlling the coder's output and matching it to a constant rate channel has been omitted in this representation. However, it has to be included for any real-time implementation.

IV. DISCUSSIONS AND CONCLUSIONS

We have developed recursive estimators in this paper and simplified them so that a real-time implementation would be possible. Some characteristics of our final estimator are that it needs computation of interpolation intensities at resolution no more than ⅛ pel/field (which can be done without multiplication). We believe that this is in the realm of present-day circuit speeds. Also, by the very nature of the

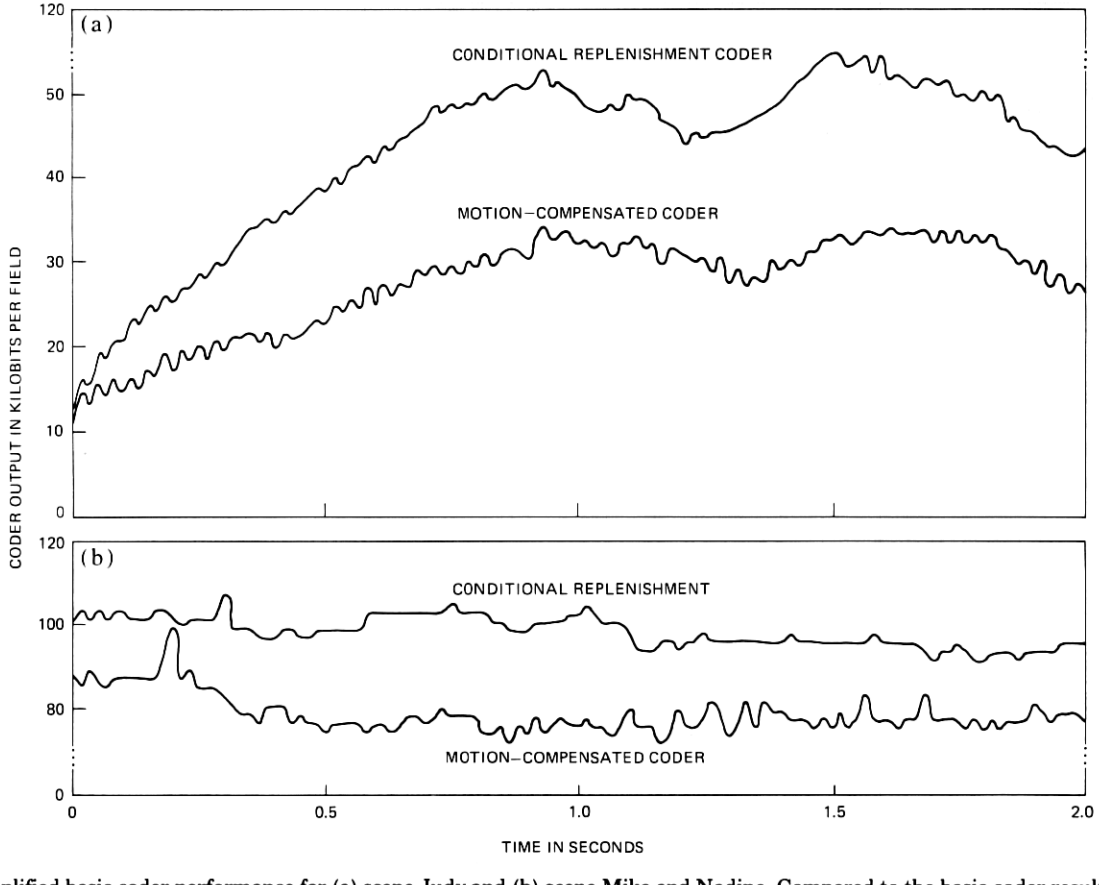

Fig. 21—Simplified basic coder performance for (a) scene Judy and (b) scene Mike and Nadine. Compared to the basic coder results of Fig. 11, the increase in output bit rate is mainly due to absolute addressing of moving area elements and the inclusion of the next segment type of information.

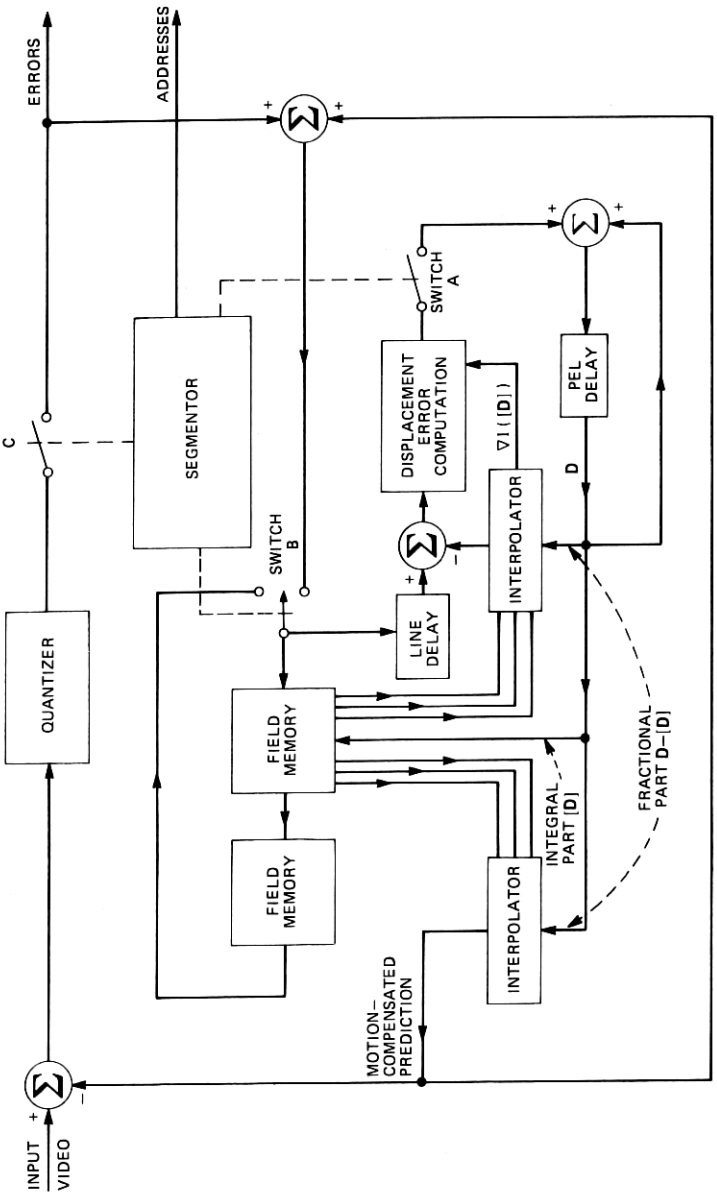

Fig. 22—A simplified block diagram of motion-compensated coder.

recursive estimation, we do not need the assumption of uniform translational motion of a single object, which appears to be necessary for most of the heretofore known estimates. Performance of the simple estimator on the sequence Judy shows that a 30-to-50 percent improvement in bits/field is possible compared to frame-difference conditional replenishment. We also evaluated the performance of our simple estimator on a more difficult scene: Mike and Nadine. Both conditional replenishment and motion compensation generate significantly more data for this scene than for Judy due to the high percentage of moving area pels. However, the motion-compensated coding decreases the transmitted bits/field by approximately 22 percent. Thus the improvement is somewhat lower for this scene. Some of the modifications suggested in Part II[16] improve the performance for this type of scene.

V. ACKNOWLEDGMENTS

The authors would like to thank Ed Bowen for his help in creating the scene Mike and Nadine and Mike Lukacs and Nadine Pinkerton for being subjects of this scene.

APPENDIX

In this appendix, we show that the gradient algorithm of eq. (13) which attempts to minimize $\mathrm{DFD}(\cdot\,,\cdot)$ converges, under suitable conditions, to true displacement. As in Section II, let us assume that an object is moving with pure translation in the field of view of the camera. Neglecting the uncovered background and assuming a constant and uniform displacement per unit time, we get

$$I(\mathbf{x}, t) = I(\mathbf{x} - \mathbf{D}, t - \tau), \tag{22}$$

where $\mathbf{D}$ is the true displacement and τ is the frame time. The pel-recursive motion estimation algorithm of eq. (13) can be written as:

$$\hat{\mathbf{D}}^i = \hat{\mathbf{D}}^{i-1} - \epsilon \nabla \mathrm{DFD}(\mathbf{x}_a, \hat{\mathbf{D}}^{i-1}) I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t - \tau). \tag{23}$$

Using the definition of $\mathrm{DFD}(\cdot\,,\cdot)$ from eq. (8), w get

$$\begin{aligned}\hat{\mathbf{D}}^i = \hat{\mathbf{D}}^{i-1} - \epsilon \{ & I(\mathbf{x}_a, t) \\ & - I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t - \tau)\} \nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t - \tau).\end{aligned} \tag{24}$$

Substituting for $I(\mathbf{x}_a, t)$ from (22),

$$\begin{aligned}\hat{\mathbf{D}}^i = \hat{\mathbf{D}}^{i-1} - \epsilon \{ & I(\mathbf{x}_a - \mathbf{D}, t - \tau) \\ & - I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t - \tau)\} \cdot \nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t - \tau).\end{aligned} \tag{25}$$

Using Taylor's series, we can write

$$\begin{aligned}I(\mathbf{x}_a - \mathbf{D}, t - \tau) - I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t - \tau) \\ = (\hat{\mathbf{D}}^{i-1} - \mathbf{D})^T \cdot \nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t - \tau) \\ + \text{higher order terms in } (\hat{\mathbf{D}}^{i-1} - \mathbf{D}).\end{aligned}$$

Substituting in (25),

$$\hat{\mathbf{D}} = \hat{\mathbf{D}}^{i-1} - \epsilon\{(\hat{\mathbf{D}}^{i-1} - \mathbf{D})^T \nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)\}\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)$$
$$+ \text{higher order terms in } (\hat{\mathbf{D}}^{i-1} - \mathbf{D}).$$
$$= \hat{\mathbf{D}}^{i-1} - \epsilon\{\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)\cdot\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)^T\}(\hat{\mathbf{D}}^{i-1} - \mathbf{D})$$
$$+ \text{higher order terms in } (\hat{\mathbf{D}}^{i-1} - \mathbf{D}).$$

Subtracting $\mathbf{D}$ from both sides, we get

$$(\hat{\mathbf{D}}^i - \mathbf{D}) = (\hat{\mathbf{D}}^{i-1} - \mathbf{D}) - \epsilon\{\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)$$
$$\cdot\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)^T\}(\hat{\mathbf{D}}^{i-1} - \mathbf{D})$$
$$+ \text{higher order terms in } (\hat{\mathbf{D}}^{i-1} - \mathbf{D}).$$

Neglecting higher order terms,* for small $(\hat{\mathbf{D}}^{i-1} - \mathbf{D})$,

$$(\hat{\mathbf{D}}^i - \mathbf{D})$$
$$= [J - \epsilon\{\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)\cdot\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)^T\}](\hat{\mathbf{D}}^{i-1} - \mathbf{D})$$

where J is an identity matrix of appropriate size. We now take statistical averages of both sides, assume statistical independence of the two right-hand terms,† and then apply Schwartz inequality to get

$$\|\bar{\hat{\mathbf{D}}}^i - \bar{\mathbf{D}}\|$$
$$\leq \|J - \epsilon\{\overline{\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)^T}\}\|$$
$$\cdot \|\bar{\hat{\mathbf{D}}}^{i-1} - \bar{\mathbf{D}}\| \quad (26)$$

where a bar on top denotes a statistical average. This can be written as:

$$\|\bar{\hat{\mathbf{D}}}^i - \bar{\mathbf{D}}\| \leq |1 - \epsilon\lambda_{\max}| \cdot \|\bar{\hat{\mathbf{D}}}^{i-1} - \bar{\mathbf{D}}\|, \quad (27)$$

where $\lambda_{\max}$ is the maximum eigenvalue of the positive semidefinite, symmetric matrix $\overline{\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)^T}$. For convergence of the algorithm, we need

$$|1 - \epsilon\lambda_{\max}| < 1,$$

i.e.,

$$\frac{2}{\lambda_{\max}} > \epsilon > 0. \quad (28)$$

Since the maximum eigenvalue is hard to compute, and since it is upper bounded by the trace, we get

---

* This may be a severe assumption in certain cases. In such cases, the ability of the algorithm to converge may depend upon the goodness of the initial estimate $\hat{D}$.

† This is traditionally done in stochastic gradient algorithms. See, for example, Refs. 19 to 21.

$$\frac{2}{tr[\overline{\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)^T}]} > \epsilon > 0. \quad (29)$$

But since

$$tr[\overline{\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)\nabla I(\mathbf{x}_a - \hat{\mathbf{D}}^{i-1}, t-\tau)^T}] \cong \frac{1}{N}\sum_{i=1}^{N} EDIF_i^2 + LDIF_i^2,$$

we get the following condition for convergence

$$\frac{2N}{\sum_{i=1}^{N} EDIF_i^2 + LDIF_i^2} > \epsilon > 0, \quad (30)$$

where the summations are carried over the entire moving area containing N pels.

REFERENCES

1. F. W. Mounts, "A Video Encoding System Using Conditional Picture-Element Replenishment," B.S.T.J., *48*, No. 7 (September 1969), pp. 2545–2554.
2. J. C. Candy, M. A. Franke, B. G. Haskell, and F. W. Mounts, "Transmitting Television as Clusters of Frame-to-Frame Differences," B.S.T.J., *50*, No. 6 (July–August 1971), pp. 1889–1917.
3. J. O. Limb, R. F. W. Pease, and K. A. Walsh, "Combining Intraframe and Frame-to-Frame Coding for Television," B.S.T.J., *53*, No. 6 (August 1974), pp. 1137–1173.
4. B. G. Haskell, P. L. Gordon, R. L. Schmidt, and J. V. Scattaglia, "Interframe Coding of 525-Line, Monochrome Television at 1.5 MBS," IEEE Trans. Commun., *COM-25*, No. 10 (October 1977), pp. 1339–1344.
5. T. Ishiguro, K. Iinuma, Y. Iijima, T. Koga, S. Azami, and T. Mune, "Composite Interframe Coding of NTSC Color Television Signals," 1976 National Telecommunications Conference Record (Dallas, Texas, November, 1976), *1*, pp. 6.4-1 to 6.4-5.
6. B. G. Haskell, F. W. Mounts, and J. C. Candy, "Interframe Coding of Videotelephone Pictures," Proc. IEEE, *60*, No. 7 (July 1972), pp. 792–800.
7. B. G. Haskell, "Frame Replenishment Coding of Television," a chapter in *Image Transmission Techniques*, W. K. Pratt, Ed., New York: Academic Press, 1978.
8. F. Rocca, "Television Bandwidth Compression Utilizing Frame-to-Frame Correlation and Movement Compensation," *Symposium on Picture Bandwidth Compression* (M.I.T. Cambridge, Mass., 1969), Gordon and Breach, 1972.
9. B. G. Haskell and J. O. Limb, "Predictive Video Encoding Using Measured Subject Velocity," U.S. Patent 3,632,865, January 1972.
10. W. N. Martin and J. K. Aggaraval, "Dynamic Scene Analysis; The Study of Moving Images," Technical Report No. 184, Information Systems Research Laboratory, University of Texas, Austin, Texas, January 1977.
11. J. L. Potter, "Scene Segmentation Using Motion Information," Computer Graphics and Image Processing, *6* (1977), pp. 558–581.
12. L. Dreschler and H.-H. Nagel, "Using 'Affinity' for Extracting Images of Moving Objects from TV-Frame Sequences," Technical Report IFI-HH-B-44/78, The University of Hamburg, Hamburg, Germany, February 1978.
13. J. O. Limb and J. A. Murphy, "Estimating the Velocity of Moving Images from Television Signals," Computer Graphics and Image Processing, *4* (1975), pp. 311–327.
14. C. Cafforio and F. Rocca, "Methods for Measuring Small Displacements of Television Images," IEEE Trans. Inform. Theory, *IT-22*, No. 5 (September 1976), pp. 573–579.
15. B. G. Haskell, "Entropy Measurements for Nonadaptive and Adaptive, Frame-to-Frame, Linear Predictive Coding of Video Telephone Signals," B.S.T.J., *54*, No. 6 (August 1975), pp. 1155–1174.

16. J. D. Robbins and A. N. Netravali, "Motion-Compensated Television Coding: Part II," unpublished work.
17. A. O. Aboutalib, M. S. Murphy, and L. M. Silverman, "Digital Restoration of Images Degraded by General Motion Blurs," IEEE Trans. on Automatic Control, *AC-22*, No. 3 (June 1977), pp. 294–302.
18. B. G. Haskell, "Differential Addressing of Clusters of Changed Picture Elements for Interframe Coding of Videotelephone Signals," IEEE Trans. Commun. (January 1976), pp. 140–144.
19. B. Widrow, J. Glover, J. McCool, et al., "Adaptive Noise Canceling: Principles and Applications," Proc. IEEE, *63* (December 1975), pp. 1692–1716.
20. L. Ljung, "Analysis of Recursive Stochastic Algorithms," IEEE Transactions on Automatic Control, *AC-22*, No. 4 (August 1977), pp. 551–575.
21. J. Salz, "On the Behavior of Stochastic Gradient Algorithms," unpublished work.