Copyright © 1980 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 59, No. 9, November 1980
Printed in U.S.A.

Motion-Compensated Coding: Some New Results

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

(Manuscript received March 19, 1980)

In earlier papers, we have presented motion-compensated television coding schemes in which the displacement of objects was recursively estimated using a steepest descent algorithm that minimized the square of the intensity prediction error at each picture element. In this paper, we present extensions in which displacement is estimated by considering the prediction error at several picture elements. These extensions are more complex, but they significantly improve the performance of the displacement estimation in those cases where the displacement is spatially uniform. However, in real scenes containing large spatial variations of displacement, only a small improvement is obtained. For one scene containing a head-and-shoulders view of a person engaged in active conversation, an improvement of about 10 percent in average bit rate was obtained over our previous motion-compensation scheme.

I. INTRODUCTION

The displacement estimation algorithms described in this paper estimate the displacement of objects in successive frames of a television scene. They are a generalization of the pel-recursive displacement estimation algorithm that we had introduced earlier.[1,2] Before describing the generalization, it is useful to first outline the pel-recursive displacement estimation algorithm. Let $I(\mathbf{x}_k, t)$ denote the intensity of a scene at the kth sample point $\mathbf{x}_k$ from a scan line and let $I(\mathbf{x}_k, t - \tau)$ denote the intensity at the same spatial location in the previous frame.* If the scene consists of an object that is undergoing pure translation under uniform illumination, then, disregarding the background,

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

* Subscript k in $\mathbf{x}_k$ is used to denote the sample number in the same order as scanning.

where $\mathbf{D}$ is the displacement (two-component vector) of the object in one frame interval τ. The pel-recursive algorithm obtains an estimate of $\mathbf{D}$ (i.e., $\hat{\mathbf{D}}$) by recursively minimizing the square of the displaced frame difference at the current pel location. The displaced frame difference $\text{DFD}(\cdot\ ,\ \cdot)$ is defined by

$$\text{DFD}(\mathbf{x}_k, \hat{\mathbf{D}}) = I(\mathbf{x}_k, t) - I(\mathbf{x}_k - \hat{\mathbf{D}}, t - \tau). \tag{2}$$

The minimization is done by a steepest descent algorithm of the form

$$\hat{\mathbf{D}}_{i+1} = \hat{\mathbf{D}}_i - \tfrac{1}{2}\, \epsilon\, \nabla_{\hat{\mathbf{D}}_i}[\text{DFD}(\mathbf{x}_k, \hat{\mathbf{D}}_i)]^2, \tag{3}$$

where $\nabla_{\hat{\mathbf{D}}_i}[\cdot]$ is the two-dimensional gradient with respect to $\hat{\mathbf{D}}_i$. Equation (3) can be expanded to

$$\hat{\mathbf{D}}_{i+1} = \hat{\mathbf{D}}_i - \epsilon \text{DFD}(\mathbf{x}_k, \hat{\mathbf{D}}_i)\ \nabla\ I(\mathbf{x}_k - \hat{\mathbf{D}}_i, t - \tau), \tag{4}$$

where $\nabla = \nabla_{\mathbf{x}}$ is the two-dimensional spatial gradient operator with respect to horizontal and vertical coordinates of vector $\mathbf{x}$.* Having computed displacement, the motion-compensated coder predicts intensity, $I(\mathbf{x}_k, t)$, by the displaced previous frame intensity $I(\mathbf{x}_k - \hat{\mathbf{D}}_i, t - \tau)$ using interpolation for nonintegral (in terms of pel distances) values of $\hat{\mathbf{D}}_i$. The displacement at either the previous pel or the previous line element is used to predict the intensity of the present pel. This allows the receiver to compute displaced previous frame intensity (or the prediction) without explicit transmission of the displacement. If the magnitude of the prediction error exceeds a predetermined threshold, the coder transmits a quantized version of $\text{DFD}(\mathbf{x}_k, \hat{\mathbf{D}}_i)$ and the necessary addressing information to the receiver.

The extensions of this paper consider the displaced frame differences at many picture elements to estimate $\mathbf{D}$. For example, $\hat{\mathbf{D}}$ can be updated from sample to sample by using a steepest-descent algorithm to minimize a weighted sum of the squared displaced frame differences at some previously transmitted neighboring picture elements. Thus,

$$\hat{\mathbf{D}}_{i+1} = \hat{\mathbf{D}}_i - \epsilon \cdot \tfrac{1}{2} \cdot \nabla_{\hat{\mathbf{D}}_i}\left[\sum_{j=0}^{p} W_j[\text{DFD}(\mathbf{x}_{k-j}, \hat{\mathbf{D}}_i)]^2\right], \tag{5}$$

where $W_j \geq 0$ and $\sum_{j=0}^{p} W_j = 1$.

We consider two variations of this algorithm. In one, the displacement is estimated by steepest descent on the weighted sum of the squared displaced frame differences. This corresponds to eq. (5) above. In the other algorithm, displacement is estimated by a least-mean-square approximation using a specified number of neighboring picture

* Subscript i is used to denote the iteration number. Since the displacement estimate may not be revised at each picture element, in general, i may not be the same as k at a given picture element. Also in some cases, there could be many iterations for the same k (or $\mathbf{x}_k$).

elements and a previous estimate of displacement. As in our previous pel-recursive estimator, these estimators can also be generalized to the transform domain.[3,4]

This paper is organized as follows. The next section gives a detailed description of the two algorithms. Section III gives results of simulations on synthetic computer-generated scenes as well as a real scene containing complex motion. It is seen that both the estimators significantly improve the performance of the displacement estimator for the synthetic scene. In the case of the real scenes, however, both the estimators give only about 10-percent improvement in the bit rate at a significant increase in the complexity.

II. DESCRIPTION OF THE ALGORITHMS

In this section, we develop the two algorithms for estimating displacement. Algorithm 1, called gradient of summed error (GSE), works as follows. Let $\mathbf{x}_k$ be the current picture element and $\hat{\mathbf{D}}_i$ be the estimate of displacement at the ith iteration. This estimate is revised by using the steepest descent on the summed error given by

$$\sum_{j=0}^{p} W_j[I(\mathbf{x}_{k-j}, t) - I(\mathbf{x}_{k-j} - \hat{\mathbf{D}}_i, t - \tau)]^2 \tag{6}$$

and, therefore, the$(i + 1)$th estimate of D, i.e., $\hat{D}_{i+1}$, is given by

$$\hat{\mathbf{D}}_{i+1} = \hat{\mathbf{D}}_i - \epsilon\left\{\sum_{j=0}^{p} W_j\text{DFD}(\mathbf{x}_{k-j}, \hat{\mathbf{D}}_i) \cdot \nabla[I(\mathbf{x}_{k-j} - \hat{\mathbf{D}}_i, t - \tau)]\right\}. \tag{7}$$

This can be generalized further using a different function of the errors (i.e., DFD$(\cdot\ ,\ \cdot)$) instead of the square function. The difference between eqs. (7) and (4) is that, to estimate a new value $\hat{\mathbf{D}}_{i+1}$ of $\mathbf{D}$ from the old value $\hat{\mathbf{D}}_i$, the displaced frame difference is evaluated at several neighboring picture elements rather than just one picture element. This has the effect of smoothing the update term [that is, the second term on the right-hand side of eq. (4)]. As in our earlier papers, the displacement is revised only at those picture elements where the magnitude of the frame difference, $I(\mathbf{x}_k, t) - I(\mathbf{x}_k, t - \tau)$, is higher than a threshold. Thus, the displacement is revised only in the "moving areas."

The second displacement algorithm is a least-mean-square estimator based on the intensity in the previous frame at a location displaced by the old estimate of displacement. Thus, assuming eq. (1), the displaced frame difference of eq. (2) can be written as

$$\begin{aligned}\text{DFD}\ (\mathbf{x}_k, \hat{\mathbf{D}}_i) &= I(\mathbf{x}_k - \mathbf{D}, t - \tau) - I(\mathbf{x}_k - \hat{\mathbf{D}}_i, t - \tau)\\ &= -(\mathbf{D} - \hat{\mathbf{D}}_i)^T\nabla I(\mathbf{x}_k - \hat{\mathbf{D}}_i, t - \tau) + \text{other terms,}\end{aligned} \tag{8}$$

where the superscript T on a vector or a matrix denotes its transpose.

Quantity $(\mathbf{D} - \mathbf{D}_i)$ can be estimated by standard techniques of linear least-mean square by using $\text{DFD}(\cdot,\cdot)$ on the left-hand side as an observation and treating the higher order terms on the right-hand side as noise.* This gives us an algorithm of the form

$$\hat{\mathbf{D}}_{i+1} = \hat{D}_i - \epsilon\left[\sum_{j=0}^{p} \nabla I(\mathbf{x}_{k-j} - \hat{\mathbf{D}}_i, t-\tau)\nabla I^T(\mathbf{x}_{k-j} - \hat{\mathbf{D}}_i, t-\tau)\right]^{-1} \cdot \left[\sum_{j=0}^{p} \text{DFD}(\mathbf{x}_{k-j}, \hat{\mathbf{D}}_i)\nabla I(\mathbf{x}_{k-j} - \hat{\mathbf{D}}_i, t-\tau)\right]. \quad (9)$$

The matrix inverse in the above equation can be approximated by

$$M_i^{-1} = \frac{1}{\Delta_i}\begin{bmatrix} \sum_{j=0}^{p} LDIF_j^2 & -\sum_{j=0}^{p} LDIF_j EDIF_j \\ -\sum_{j=0}^{p} LDIF_j EDIF_j & \sum_{j=0}^{p} EDIF_j^2 \end{bmatrix}$$

with

$$\Delta_i = \sum_{j=0}^{p} EDIF_j^2 \sum_{j=0}^{p} LDIF_j^2 - \left\{\sum_{j=0}^{p} EDIF_j\, LDIF_j\right\}^2, \quad (10)$$

where

$EDIF_j =$ Element difference at $(\mathbf{x}_{k-j} - \hat{\mathbf{D}}_i)$ in the previous frame (approximating the horizontal component of ∇I).
$LDIF_j =$ Line difference at $(\mathbf{x}_{k-j} - \hat{\mathbf{D}}_i)$ in the previous frame (approximating the vertical component of ∇I).

In evaluating the matrix inverse, one must be careful that the matrix is not singular. Singularity can be a result of not averaging enough samples. If, on the other hand, a large number of samples are averaged, then displacement averaging may result. To avoid this, whenever M_i came close to being singular (as indicated by its determinant), no update was performed. The second term of eq. (9) is given by

$$\psi_i = \begin{bmatrix} \sum_{j=0}^{p} \text{DFD}(\mathbf{x}_{k-j} - \hat{\mathbf{D}}_i, t-\tau)\, EDIF_j \\ \sum_{j=0}^{p} \text{DFD}(\mathbf{x}_{k-j} - \hat{\mathbf{D}}_i, t-\tau)\, LDIF_j \end{bmatrix}. \quad (11)$$

Since M_i is only a 2×2 matrix, no great savings may result in computations by using the matrix inversion lemma.[5,6]

* Linear least-mean-square estimation can be used by assuming that the second term of the right-hand side of eq. (9), i.e., $\nabla I(\mathbf{x}_k - \mathbf{D}_i, t-\tau)$, does not "strongly" depend upon $\mathbf{D}_i$.

The ϵ in both algorithms was kept fixed. For algorithm 1, the best ϵ was found to be 1/128 among the set of ϵ that were tried. Similarly for algorithm 2, the best ϵ was found to be ½. The update term for both the algorithms was also limited to make sure that no individual iteration changed the displacement estimate by more than a certain amount. In algorithm 1, the update term was limited to (0.2) pels/field, whereas for algorithm 2, it was limited to (0.08) pels/field.

III. SIMULATION RESULTS

We simulated the above two algorithms on two types of scenes. The first scene is a synthetic scene which was computer-generated. It is a damped radial cosine in intensity with a radius of 60 pels which translated from frame to frame by a given amount. The pattern is described mathematically by the intensity function

$$I(R) = 100 \cdot \mathrm{Exp}(-0.01R)\cos(2\pi R/P) + 128; \qquad 0 \le R \le 60, \qquad (12a)$$

where R is the radial distance from the center (taken to be (100, 100)) and

$$P = (1 - R/60)10 + 10. \qquad (12b)$$

This function is displayed on a 256 × 256 element raster in two interlaced fields of 128 lines each. This pattern is shown in Fig. 1. The

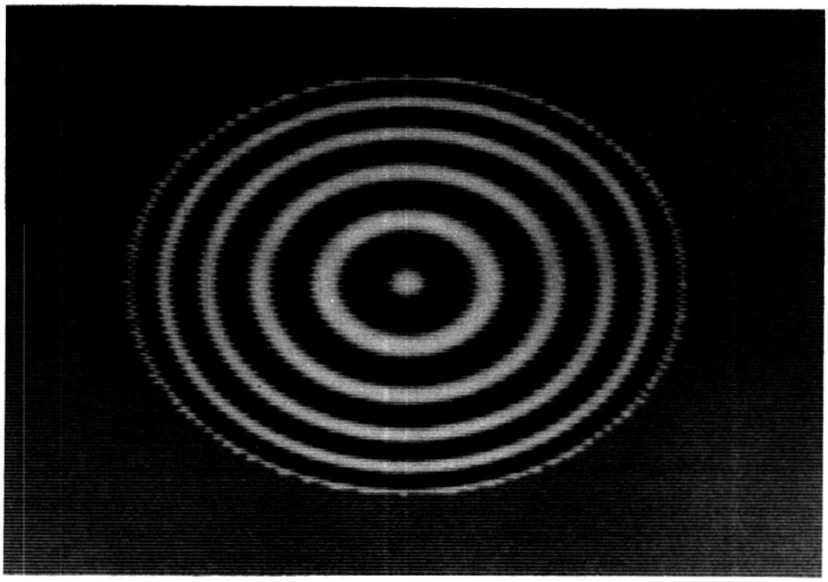

Fig. 1—Synthetic image used in simulations. This image is described by eq. (12) in Section III.

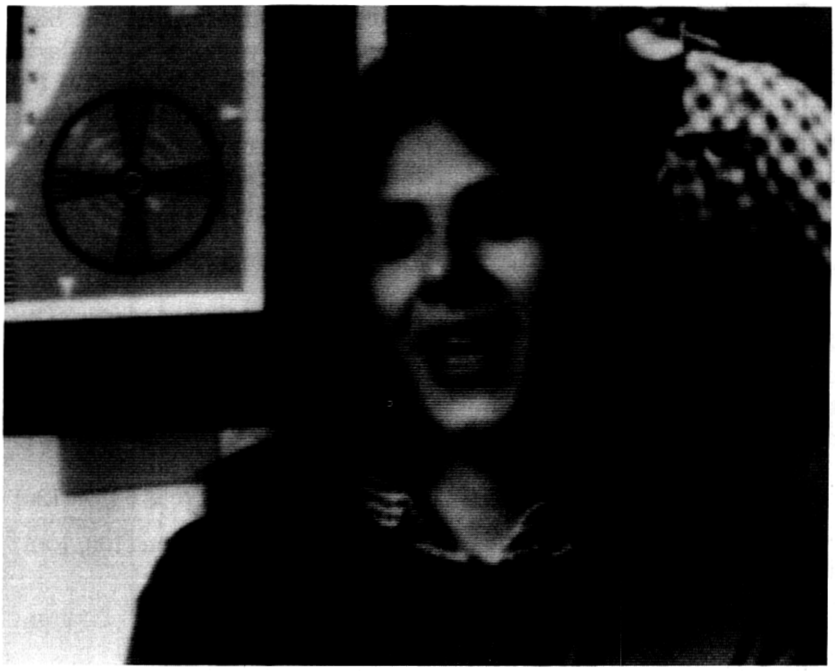

Fig. 2—Single frame from the scene Judy.

other scene, called Judy, is a head-and-shoulders view of a person engaged in active conversation. This consisted of 60 frames obtained by Nyquist rate sampling of a video signal having 1-MHz bandwidth. Each sample was quantized uniformly to 8 bits. One frame of this scene is shown as Fig. 2.

Various configurations of neighboring picture elements were used to evaluate the error terms. Referring to Fig. 3, let the prediction be evaluated at the picture element X. The error terms are evaluated at picture elements with the following five configurations.

Configuration 1—Error term consists of error only at element K. This is similar to our previous algorithm,[1,2] and is included here for the purpose of comparison.

Configuration 2—The error term is made up of the displaced frame differences at {C, D, J, K, L}.

Configuration 3—The error term is made up of displaced frame differences at {A, B, X, C, K}. This uses certain picture elements not yet available to the receiver and is included only to evaluate the effect of knowledge of such picture elements on the displacement estimator.

Configuration 4—The error term is made of displaced frame differences at {I, J, K, L, M, O, P, Q}. We note that this configuration uses picture elements only from the previous lines in the same field.

Configuration 5—Here the error term is made up of displaced frame differences at {C, D, E, F, H, I, J, K, L, M, N, O, P, Q}. This uses previously transmitted picture elements from the present line as well as from the last two lines.

Many other configurations were tried; however, no interesting conclusions could be reached for these others. In some of these cases, weighted errors with unequal weights were used.

3.1 Results from synthetic scenes

In this case, only the quality of the displacement estimators was judged with no reference to its usefulness for coding. Displacement estimators were initialized to zero at the leftmost element of each scan line, and recursions were carried out from pel to pel within a scan line. Figure 4 is a plot of the normalized displacement error

$$\| \hat{\mathbf{D}}_i - \mathbf{D} \| / \| \hat{\mathbf{D}}_0 - \mathbf{D} \|$$

for algorithm 1, and Fig. 5 shows it for algorithm 2. It is clear from these figures that inclusion of more picture elements in the error term improves the convergence and the steady-state error. For example, with algorithm 1, configuration 2 decreases the normalized displacement error to 0.06 compared to 0.5 for configuration 1.

For algorithm 1, configurations that use picture elements from only the previous line do not perform as well as configuratioins of picture elements from the present and previous line. For algorithm 2, in general, the convergence is much faster than algorithm 1, and low steady-state error is obtained. Configuration 4, for example, attains in 40 iterations a normalized displacement error of 0.09 for algorithm 1 and 0.05 for algorithm 2. Configuration 5 does significantly better than configuration 1 for algorithm 2. In 60 iterations, it reduces the nor-

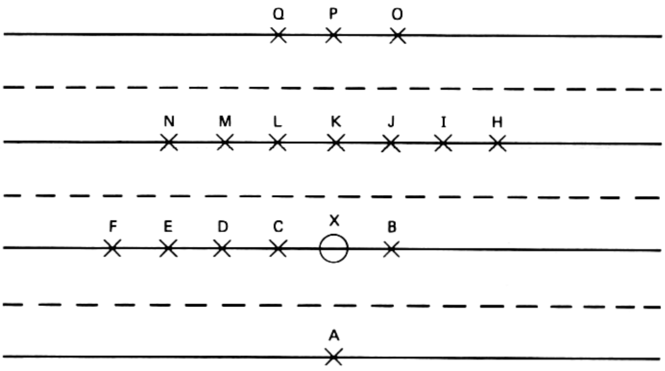

Fig. 3—Configuration of picture elements for weighted error calculation. Pel X is being predicted. Dotted lines denote scan lines from previous field.

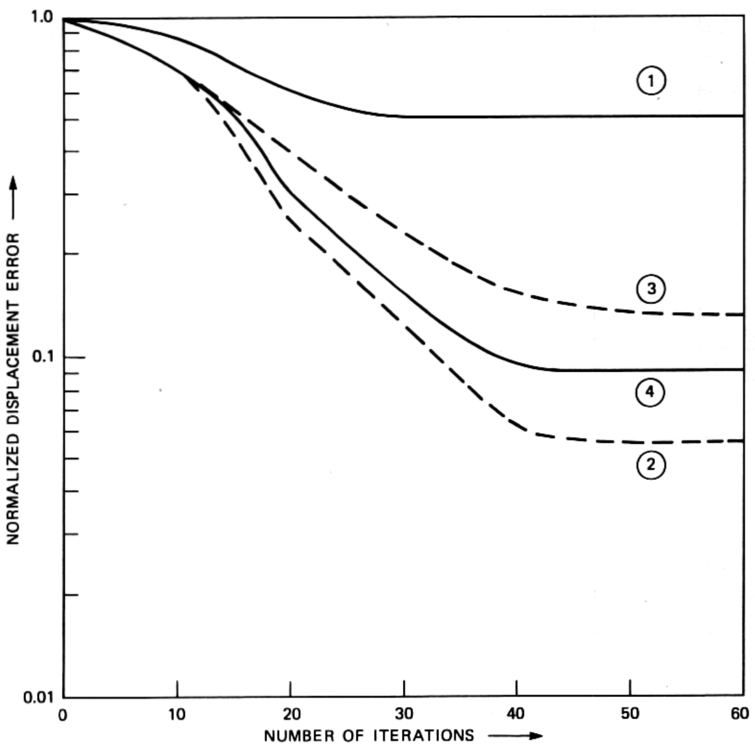

Fig. 4—Normalized displacement error ($\| \hat{\mathbf{D}}_i - \mathbf{D} \| / \| \hat{\mathbf{D}}_0 - \mathbf{D} \|$) is plotted against the iteration number for algorithm 1. Four configurations (1, 2, 3, 4) are shown. All iterations are started from the left-most pel in a scan line, $\hat{\mathbf{D}}_0$ is taken to be zero, and $\mathbf{D}$ is taken to be 2 pels per frame in the horizontal direction.

malized displacement error by a factor of more than 30 compared to configuration 1. It is seen, therefore, that for both algorithms the increase of number of picture elements in the error term is a significant advantage and appears to be a controlling factor. We also tried to use weights in the calculation of the error term which were inversely proportional to the exponential of the distance from the picture element X. It was found that there was no significant improvement using such a set of weights. One can conclude, then, that in the synthetic scene where the displacement is spatially uniform, proximity to the picture element X is not as important as the number of picture elements used in averaging the error. We also made some simulations in which the synthetic scene was corrupted by additive noise. In these cases, the convergence and the steady-state error generally became worse compared to the case of no noise. However, the additional improvement by using more picture elements was higher than in the case of no noise. Thus, the relative improvement by using configuration

5 compared to configuration 1 was higher (normalized error decreased by a factor of 40, compared to the factor of 30 for no noise case). It was also found that, for algorithm 2, the number of times matrix M_i came close to being singular was more in the case of configuration 2 than in configuration 5. This supplements our reasoning that averaging large number of pels prevented the matrix M from becoming nonsingular.

3.2 Results for real scenes

Both algorithms 1 and 2 were simulated with real scenes as the input. This was done to evaluate their usefulness for coding. Thus, a sequence of pictures was coded by a motion-compensated coder of the same type as in Ref. 2, except that the displacement was estimated by the new extensions. We used configurations 1, 3, and 5. Configuration 3 was used only for algorithm 1 and configuration 5 was used for both algorithms. The DPCM quantizer had 35 levels and is given in Fig. 10 of Ref. 1. Prediction error was sent in a quantized form only if it was higher than a threshold of 3 (out of 255 corresponding to 8-bit signals). This gave a picture quality which appeared reasonable; that is, the coding degradations were visible but not annoying.

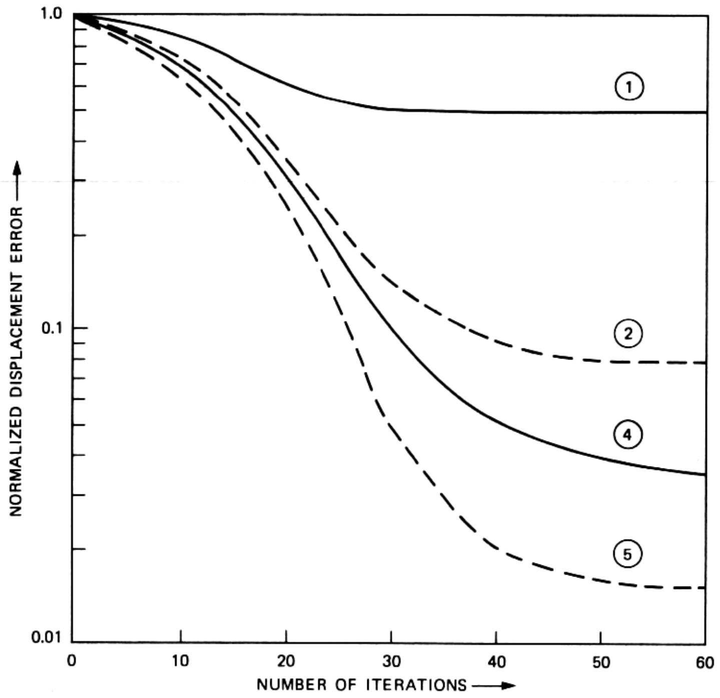

Fig. 5—Normalized displacement error against the iteration number for algorithm 2.

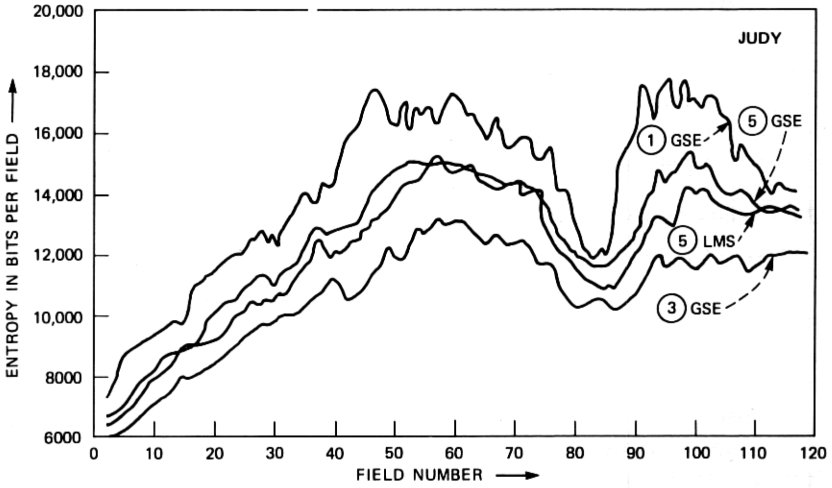

Fig. 6—Plots of bits per field are given for the scene Judy. Configuration 1 corresponds to our previous motion-compensation algorithm (Ref. 2). Configuration 5 is shown for the two algorithms (GSE and LMS). Configuration 3 uses some pels for displacement estimation that are not yet available to the receiver.

The total bits per frame were calculated by adding bit requirements for quantized prediction error (whenever it was sent) and addressing. Error bits were computed by evaluating the entropy of the quantized prediction error, and the address bits were computed by one-dimensional run length coding of the unpredictable picture elements along a scan line. To reduce the computational burden, only a part (240 × 240 array) of the frame was coded. Figure 6 shows the plots of bits per field as a function of the field number for the 120 fields of the scene Judy. Configuration 1, which is our old motion-compensation algorithm,[2] is plotted for comparison. It is seen from Fig. 6 that configuration 5 for both algorithms is about 10 percent better than configuration 1. There is a slight preference for algorithm 2 over algorithm 1. Configuration 3, which uses some picture elements not yet available to the receiver, does about 20 percent better than does configuration 1. It is seen that, although configuration 5 gives dramatic improvement in convergence of displacement iterations for synthetic scenes, it does not appreciably improve the performance of the coder. We feel this may be a result of spatially nonuniform displacements in the real scene. The use of a larger number of picture elements for evaluation of the error term has the effect of averaging displacements which might not be so useful if these displacements are nonuniform. Another scene, Mike and Nadine,[1,2] was also processed with both the algorithms using configuration 5. It was found that bit rates decreased by about 12 percent compared to the use of configuration 1 (our previous motion compensation algorithm).

IV. CONCLUSIONS

We have presented in this paper extensions of our previous displacement estimation algorithms. These extensions allow us to recursively estimate the displacement of objects by minimizing the intensity prediction error at several picture elements rather than only one, as in our previous algorithms. We have found that, for synthetic scenes where the displacement is spatially uniform, the extensions perform significantly better in terms of convergence and steady-state displacement error. However, in real scenes, where the displacement may be spatially nonuniform, only about 10-percent improvement in average bit rates is obtained compared to our previous motion-compensation schemes.

REFERENCES

1. A. N. Netravali and J. D. Robbins, "Motion-Compensated Television Coding: Part I," B.S.T.J., *58*, No. 3 (March 1979), pp. 631–670.
2. J. D. Robbins and A. N. Netravali, "Interframe Coding Using Movement Compensation," Int. Conf. on Communications, June 1979.
3. J. A. Stuller and A. N. Netravali, "Transform Domain Motion Estimation," B.S.T.J., *58*, No. 7 (September 1979) pp. 1673–1702.
4. A. N. Netravali and J. A. Stuller, "Motion-Compensated Transform Coding," B.S.T.J., *58*, No. 7 (September 1979), pp. 1703–1718.
5. D. D. Falconer and L. Ljung, "Application of Fast Kalman Estimation to Adaptive Equalization," IEEE Transactions on Communications, *COM-26*, No. 10 (October 1978), pp. 1439–1446.
6. M. J. Shensa, "Recursive Least Squares Lattice Algorithms—A Geometrical Approach," to appear in IEEE Trans. on Automatic Control.