1 Introduction

Interframe predictive coding is one of the most powerful image-coding techniques that can eliminate redundancy in natural scenes. In a typical interframe coder, the input picture is subtracted from the motion-compensated prediction of the previous frame and the resulting differential signal is coded and transmitted. At the other end, the decoder builds the same difference signal and adds it to the reconstruction of the previous frame to reconstruct the current frame. The key to the success of an interframe predictive coder is the ability to predict the current frame based on the previous ones. The better the prediction, the smaller is the error signal and hence the transmission bit rate.

For still parts of the picture, the best and simplest predictor is the one that uses the information from the previous frame. Real scenes, however, usually contain moving objects. For those, a more efficient prediction can be constructed by using picture elements (pixels) from the previous frame that are appropriately displaced. The prediction is then called motion-compensated prediction and the actual process is called motion compensation. In the past, various algorithms for motion estimation have been successfully used.

Traditionally, motion estimation techniques are categorized into two broad types: (1) block matching and (2) pixel recursive. Both methods are based on the 2-D information extracted from successive picture frames. The block-matching method estimates the displacement vectors by comparing the gray levels of successive frames on a block-by-block basis. The pixel-recursive method, on the other hand, predicts the displacement of each pixel recursively from its neighboring pixels, which have been coded. Both methods rely on the following assumptions:

1. The motion of moving objects is purely translational.
2. The illumination is uniform in both spatial and temporal domains.
3. Masking between objects and uncovered background is neglected.

In recent years, new methods of motion estimation have been devised that incorporate 3-D motion constraints into displacement estimation. Such techniques are very popular in new image coding schemes (e.g., model-based coding) and present a more general approach to motion estimation. Unfortunately the procedure of extracting the motion parameters remains quite complicated and computationally expensive.

At present, the block-matching algorithms attract most of the attention because of their simplicity and effectiveness. The method segments an image into fixed-size rectangular blocks and assumes that each block undergoes independent uniform translation given by the displacement vector $V = (dx, dy)$, as shown in Fig. 1. To maintain the validity of the assumption, in practice, relatively small square blocks are used (e.g., $8 \times 8$ or $16 \times 16$ pixels). For each block in the current frame, a thorough comparison is performed with all possible corresponding blocks within a search area in the previous frame. The best match is found by minimizing a distortion measurement, such as the mean-square error (MSE), or by maximizing a correlation function (e.g., cross-correlation) of the two blocks. To reduce the computational load a variety of fast algorithms have been proposed.

Subject terms: visual communication; video coding; motion estimation; motion analysis; block matching.
Despite its increasing popularity, block-matching motion compensation has its drawbacks. Most of them result from the assumption that the motion of the moving objects within a block is a uniform translation that can be approximated by a displacement vector. In reality, motion is a complex combination of translation and rotation that cannot be estimated by conventional block-matching techniques, because they inherently estimate only translation.

To cope with rotation as well as other nonlinear deformations, a general approach to the block-matching motion estimation is proposed here. It approximates the deformation of the real objects by deforming the corresponding blocks in the picture. Then for each block, extra information is extracted, corresponding to the complex motion within the picture. The calculation of the extra information requires additional operations that increase the computational load, but the improved prediction reduces the bit rate, especially when complicated motion exists in the scene.

2 Pixel Block Classification

For an efficient interframe predictive coding, it is important to know the distribution of the motion-compensated (MC) errors in each segmented block. If the analytical model of the MC errors is known, then only very few characteristics of the model have to be transmitted for the perfect reconstruction, resulting in an enormous reduction in bit rate. Unfortunately, MC errors are difficult to express in an analytical form because of their complicated character originated from a variety of factors such as noise, luminance changes, and the texture of the moving objects.6

In the past, an effort was made to qualitatively separate the blocks in an MC-predictive transform coder into several classes.7,8 This is important because it offers a simple method to define the main causes of the frame difference signal and thus to improve the coding performance by eliminating them. The same idea has been successfully used in international video coding standards9 such as the CCITT H.261. Here a similar categorization is introduced. Referring to Fig. 2, the first-class blocks include still areas of the picture, such as the background. For these blocks, the MC prediction error has small values caused mainly by noise or small luminance changes and are treated as insignificant blocks that are not coded. We call these blocks class A blocks. The second class contains blocks from moving objects on which the conventional motion estimation works well. After the motion compensation, the interframe difference signal contains small values and the block information requires only a few bits to be coded, as long as the motion vector is transmitted as side information. We name the blocks in this category as class B blocks.

The blocks with no appropriate corresponding blocks in the previous frame form the class C blocks. These blocks result mainly from the masking between the moving and stationary objects, the deformation of the objects, and luminance changes. MC prediction errors may consist of rather large values because they contain areas close to the edges of the moving objects. The motion compensation for the class C blocks often fails in several pixels because of the deformation of the objects, subpixel movements, and differences in pixels to sampling points. It is evident that the third type blocks are the ones that mostly contribute to the overall bit rate. In hybrid differential pulse code modulation/discrete cosine transform (DPCM/DCT) coders (e.g., H.261, MPEG), class C blocks yield to many significant coefficients that require a large number of transmitted bits. The justification of this statement is given in the following with a simple example.

Figures 3(a) and 3(b) show two successive frames of an artificial image sequence, where a dark-squared object rotates by 30 deg/frame around the perpendicular axis to the image plane, which passes through the center of the object. The frame difference signal before and after the motion compensation can be seen in Figs. 3(c) and 3(d), respectively. For the motion estimation, the full-search method, within a search window of ±8 pixels, was employed. The overlapped grid shows the position of the 16 × 16 pixel blocks used to estimate the motion vectors. The figure demonstrates that the motion compensation significantly reduces the interframe difference because it successfully eliminates the contribution of the class B blocks.

Although the motion compensation works quite well for the class B blocks, it fails to eliminate the contribution of the class C blocks. This can be seen in Fig. 4, where the mean-square error before and after the motion compensation for the changed blocks is drawn against their block number. The graph reveals that the motion compensation significantly reduces the MSE of some blocks, but leaves the majority of them (12 blocks) in the same level, without any compression gain.

To reduce the output bit rate of the coder while retaining the same quality picture, a new method for the prediction of these blocks must be used. In Sec. 3 a general approach to block-matching motion estimation is presented. It is a superset of the conventional block-matching algorithms and handles the class C blocks more successfully.
3 Generalized Block Matching

3.1 Introduction

Motion estimation can be viewed as a coordinate transformation that is applied to every pixel of the picture frame. Based on this concept, the coordinates of a pixel in the previous frame are transformed by the translation vector \( \mathbf{V} = (dx, dy) \) to obtain the coordinates of the corresponding pixel in the current frame. The chosen transformation satisfies a predefined criterion (for example, minimizes the MSE). To simplify the description of motion estimation in three dimensions, a notation borrowed from the computer graphics is used.

Let \( g_{i,j} \) and \( g_{i+dx,j+dy} \) represent the gray values of two blocks, \( N \times N \) pixels each, of the current and previous frames, respectively. If we represent their corresponding \( x \) and \( y \) coordinates with the symbols \( x_i \), \( y_i \), and \( x_i^{\prime} \), \( y_i^{\prime} \), respectively, where \( i = 1, 2, ..., N^2 \), then the conventional block-matching motion estimation can be expressed as: Find the motion parameters \( dx \) and \( dy \) that satisfy the following requirements:

\[
\begin{align*}
    x_i^{\prime} &= x_i + dx \\
    y_i^{\prime} &= y_i + dy
\end{align*}
\]

Any other distortion or correlation measure can be substituted for the mean-square error in Eq. (2). A general method to find the optimum values of \( dx \) and \( dy \) is to take all the possible combinations within a predefined region (full-search method) and choose the one that minimizes the distortion measure in Eq. (2). Alternatively, a faster method can be employed to search only some representatives of the whole set of combinations, which hopefully achieve the same accuracy, but with a fraction of the computational load (fast motion estimation algorithms). Whatever method used, it is evident from Eq. (1) that each pixel from the previous frame block is related to only one pixel of the current frame block. Thus, there is a one-to-one relationship between the corresponding pixels from the previous and the current frames.

An interesting generalization of this formulation is to employ a different geometric transformation instead of pure...
GENERAL APPROACH TO BLOCK-MATCHING MOTION ESTIMATION

Fig. 4 Mean absolute difference for changed blocks.

Fig. 5 Generalized block-matching principle.

Fig. 6 Example of successful rotational type motion estimation with pixel block transformation.

3.2 Affine Transformations

The simplest transformation to consider is the affine transformation. In this case, the coordinates of the corresponding pixels in the two frames are related with the following mapping:

\[ x'_f = a_0 x^p_f + a_1 y^p_f + a_2 \]  \[ y'_f = a_3 x^p_f + a_4 y^p_f + a_5 \]  (4)

The affine transform preserves the parallel lines and the equispaced points. Examples of affine transformed blocks are shown in Figs. 7(b), 7(c), and 7(d). Conventional block matching is a special case of the affine transformation with \( a_1 = a_3 = 0 \) and \( a_0 = a_4 = 1 \). Although the affine transformation offers only six degrees of freedom and thereby facilitates only triangle-to-triangle mappings, recent results have shown that a considerable improvement in the interframe coder efficiency can be achieved with the use of similar mappings.\(^{12,13}\)

3.3 Perspective Transformations

A more general transformation can be obtained by considering the perspective transformation. This transformation preserves the parallel lines only when they are parallel to the projection plane. Otherwise, the transformed lines converge to a vanishing point. This property, common to the human eye and the video camera, has been used in various 3-D motion estimators\(^{14,15}\) and as part of a knowledge-based video coding system.\(^{16}\) In the perspective transformation, the mapping functions are given by
Without loss of generality, the mappings can be normalized so that $a_8 = 1$. The perspective transformation shares several important properties with affine transformation. They are both planar mappings, and thus their forward and inverse transforms are single valued. They also preserve lines in all orientations. The affine transformation is a special case of the perspective transformation with $a_6 = a_7 = 0$ and $a_8 = 1$. The nine degrees of freedom are sufficient to permit planar quadrilateral-to-quadrilateral mappings. An example of perspective transformation is shown in Fig. 7(e). Note that the intersections along edges are not equispaced.

### 3.4 Bilinear Transformations

A third type of transformation, which can be applied to the pixel coordinates of a block, is the bilinear mapping. It is most commonly used in the interpolation of unknown pixel values from the surrounding ones. In general, the bilinear transformation handles the four-corner mapping problem for nonplanar quadrilaterals. Bilinear mapping preserves lines that are horizontal or vertical in the source image, such as the affine transform. However, lines not oriented along these two directions are not preserved as lines but rather as quadratic curves. The bilinear transformation of the pixel coordinates are given by the following equations:

$$
\begin{align*}
    x_i^c &= a_0 x_i^p + a_1 y_i^p + a_2 \\
    y_i^c &= a_3 x_i^p + a_4 y_i^p + a_5 \\
    x_i^f &= a_6 x_i^p + a_7 y_i^p + a_8 \\
    y_i^f &= a_9 x_i^p + a_{10} y_i^p + a_{11}
\end{align*}
$$

An example of bilinear transformation is shown in Fig. 7(f). Note that the intersections along the edges are equispaced. It is easy to verify that the affine transformation is a special case of the bilinear transformation with $a_6 = a_7 = 0$.

### 3.5 Comments on the Transformations

The presented nonlinear mapping functions improve prediction by deforming the matching blocks between successive frames. They provide extensions to the conventional block matching that retain the backward compatibility with the current algorithms. The actual implementation requires only simple modifications of the existing systems to conform to the new transformations.

The fact that the proposed method is applied to each block independently has the additional advantage of being suitable for parallel-processing schemes. This partly alleviates the heavy computational load of the high-precision arithmetic needed for the calculation of the mapping parameters. However, for better results, the required antialiasing and subsampling processes increase the computational load even further. In Sec. 4, the implementation techniques that can be used to overcome these problems are discussed.

### 4 Implementation Issues

#### 4.1 Estimation of the Mapping Parameters

To start with, we must specify a method for estimating the mapping parameters. Without loss of generality, we assume that the current frame is divided into square blocks of $N \times N$ pixels and for each block the best-matched quadrilateral is searched in the previous frame. Although the mapping func-
tions in Eq. (3) are continuous, the digital images are discrete, and thus the number of the searched quadrilaterals is enormously reduced.

Assuming that the position of each corner of a matching pixel block can vary no more than \( \pm N \) pixels/frame horizontally and vertically, there are \((2N + 1)^2\) possible displacements for each corner. The combination of all four corners yields a total number of \((2N + 1)^8\) possible quadrilateral candidates for each block in the current frame. Then the mapping parameters for each candidate can be found by solving the linear system formed by the corresponding corner coordinates. Fortunately this linear system can be simplified to a fast and simple formula.\(^\text{17}\) If, for example, a 16 \(\times\) 16 pixel block from the current frame is matched with the quadrilateral \(EFGH\) from the previous frame (see Fig. 5), then the mapping parameters for bilinear and perspective transformations are given respectively as:

**Bilinear:**

\[
\alpha_0 = \frac{x_E - x_E}{15}, \quad \alpha_1 = \frac{x_H - x_E}{15},
\]

\[
\alpha_2 = \frac{x_E - x_E + x_G - x_H}{225}, \quad \alpha_3 = x_E,
\]

\[
\alpha_4 = \frac{y_F - y_E}{15}, \quad \alpha_5 = \frac{y_H - y_E}{15},
\]

\[
\alpha_6 = \frac{y_F - y_F + y_G - y_H}{225}, \quad \alpha_7 = y_E.
\]  \hfill (7)

**Perspective:**

\[
\alpha_0 = \frac{x_F - x_F}{15} + \alpha_0 x_F, \quad \alpha_1 = \frac{x_H - x_E}{15} + \alpha_1 x_H, \quad \alpha_2 = x_E,
\]

\[
\alpha_3 = \frac{y_F - y_E}{15} + \alpha_3 y_F, \quad \alpha_4 = \frac{y_H - y_E}{15} + \alpha_4 y_H, \quad \alpha_5 = y_E,
\]

\[
\alpha_6 = \frac{\Delta x_E}{\Delta x_G}, \quad \alpha_7 = \frac{\Delta y_E}{\Delta y_G},
\]  \hfill (8)

where

\[
\Delta x_F = x_F - x_G, \quad \Delta x_G = x_H - x_G, \quad \Delta x_H = x_E - x_F + x_G - x_H,
\]

\[
\Delta y_F = y_F - y_G, \quad \Delta y_G = y_H - y_G, \quad \Delta y_H = y_E - y_F + y_G - y_H.
\]

Knowing the mapping parameters, the current block is scanned and the mean-square error is calculated for each mapping based on the differences of individual corresponding pixels. An even simpler method can be used, taking advantage of the repetitive nature of the deformations introduced for each quadrilateral. Evidently the mapping parameters can be calculated only once, because each block is matched with quadrilaterals of specific shapes. This approach requires the availability of a large memory, i.e., a lookup table (LUT), to store the mapping parameters of all the possible quadrilaterals.

The decision of which method to use depends on the characteristics of the particular application. Regardless of the adopted method, however, accelerating techniques can be used to simplify the whole process.

### 4.2 Computational Load

It was shown earlier that for a \( \pm N \) pixels/frame search area in the previous frame, the algorithm has to check \((2N + 1)^8\) different mappings. Even for small values of \(N\) the number of search operations is many times higher to that of the conventional full search method. To make the computational load affordable, a fast searching technique is proposed here.

The method can employ any fast block-matching algorithm, but instead of searching the entire window corner by corner, all four corners are tested before proceeding to the next stage of the algorithm. For example, using the three-step search method,\(^\text{4}\) which searches eight positions at each stage, the number of search points for a maximum motion speed of \(N = 8\) pixels/frame is reduced from \((25)^4 = 390,625\) to \((94)^4 + 8^4 = 14,753\). The number of operations can be reduced further if a faster motion estimator is used. The cross-search algorithm,\(^\text{5}\) for example, requires \((4 + 44)44 = 1393\) search operations. Figure 8 illustrates the use of the three-step search method for one of the corners. The shape of the best-matched quadrilateral in each stage is shown with different shading. The same search pattern applies to all four corners of the matching quadrilateral, but their positions can be different from each other.

Further reduction can be achieved by adaptively checking only those blocks that exhibit large interframe differences after conventional motion compensation. In this case, generalized block matching is applied only on the class C blocks, which cannot be compensated accurately by conventional block matching. Block discrimination is based on a distortion measure (e.g., MSE) and an arbitrary predefined threshold.

Other reductions can also be obtained by taking advantage of the intra/interdependency of the neighboring blocks esti-
The two mapping functions in Eq. (3) define a spatial transformation that establishes a spatial relationship between the pixels in the current and previous frame. The spatial transformation is only one of the three components that comprise the digital picture warping. The other two are the resampling and antialiasing, which are introduced as a result of the discrete nature of the digital images.

The resampling is introduced because the previous and current frame pixel grids, related to each other with the functions in Eq. (3), do not generally coincide. In fact, the positions of the grid points may take on any of the continuous values assigned by the mapping functions. Then an interpolation must be performed to reconstruct the continuous signal from its samples. A further sampling of the reconstructed signal gives the values that are assigned to the transformed pixels. The accuracy of the interpolation has a significant impact on the efficiency of the mapping. Popular interpolation functions include nearest neighbor, bilinear, and cubic convolution.

The antialiasing is the filtering used to eliminate aliasing. The aliasing is caused by the many-to-one nature of the non-linear mappings. These mappings require an appropriate filtering to properly integrate all the information from the source pixels to the target pixel. The most common solution to aliasing cancellation is the smoothing of the input picture prior to sampling.

Although resampling and antialiasing are very important components of the image-warping process, they require a large number of computations. To optimize the processing speed versus accuracy, the nearest neighbor interpolation is employed for the resampling, and a simple smoothing filter is introduced on the current frame block before the estimation of the mapping parameters. The filter is identical to that used in the H.261 standard coder feedback loop and requires relatively few computations.

An important factor influencing the performance of a motion-compensated video-coding system is the overhead information. As the number of class C blocks increases, the entropy also increases, reducing the compression gain of the source coder. Acknowledging this problem, most video coders employ a coding scheme for the motion overhead information. Favorite methods are entropy coding, DPCM in the spatial and/or temporal domains, and vector quantization.

In generalized block matching, the motion vector can be formed in different ways. It may consist of the eight motion parameters given in Eqs. (7) or (8), which define the appropriate mapping for each pixel block. Alternatively the horizontal and vertical displacements for each corner of the deformed quadrilateral can be used for the same purpose. The advantage is the use of integer arithmetic compared to floating point arithmetic needed for the mapping parameters.

A third method uses an index to the list of all possible mappings. With this method, both coder and decoder have access to a memory that holds the solutions of the linear systems tested in the motion estimation process. When the best-matched quadrilateral is found, the coder sends the corresponding index to the decoder, avoiding the transmission of the actual mapping parameters. The adoption of the fast
search method presented in Sec. 4.2 indicates that only 14 bits are required to represent any of the 14,753 different quadrilaterals matched to the block in the current frame. If a more sophisticated algorithm is adopted for the coding of the motion vectors, the motion overhead can be further reduced. In our experiments, no attempt was made to code the motion information to demonstrate the worst possible case in relation to bit rate.

5 Experimental Results

5.1 Experimental Setup

In the experiments, the generalized block matching was performed in two stages. In the first stage, the conventional full-search method was applied to $16 \times 16$ pixel blocks of the successive frames within a $\pm 8$ pixels search window. The resulting motion-compensated frame was compared with the input frame to discriminate the blocks that exhibit large interframe differences even after the first stage. The decision for the continuation to the second stage was then taken for each block, based on a predefined threshold. This threshold not only separates the active blocks with complex motion but also excludes the search for blocks with small frame differences (due to luminance changes or noise), thus reducing the total number of calculations.

For the second stage, the three-step search method introduced in Sec. 4.2 was adopted with the search steps of $\pm 4$, $\pm 2$, and $\pm 1$ pixels. Furthermore, the convex quadrilateral protection was used to avoid degenerated conditions and to reduce the number of searched quadrilaterals.

5.2 Artificial Images (Graphics)

Table 1 contains the mean absolute difference between the two image frames of Figs. 3(a) and 3(b). The table shows that there is a large reduction in the frame difference signal because of the improved prediction of the generalized motion estimation. Thus, the contribution of the class C blocks has been eliminated or at least reduced to a minimum. The results can be verified by the motion-compensated frames shown in Fig. 11. For comparison, Fig. 11(a) shows the result obtained by the conventional full-search method, whereas Fig. 11(b) shows the result using only simple affine transformations, as described in Ref. 12. Figures 11(c) and 11(d) illustrate the performance of the generalized method with perspective and bilinear transformations, respectively.

The improved motion-compensated prediction results in a large reduction of the bit rate. Figure 12 shows the bit rate of an H.261-type coder for different quantization step sizes using the picture frames of Fig. 3. It is interesting to note that there is a small improvement with the introduction of the conventional full-search method but a considerably larger improvement with the generalized block matching, resulting from the nature of the motion in the scene (e.g., pure rotation). Another interesting point is the better performance of the perspective transformation over the bilinear one. However, further experiments are needed to justify the superiority of the perspective transform.

Assuming 14 bits/block overhead information for the transmission of the generalized motion vectors, there are 280 extra bits that must be added to the bit rate of Fig. 12 for accurate comparisons. However, with real images, the significance of entropy reduction is too high such that 14 bits/block overhead does not inflate the bit rate.

5.3 Real Image Sequences

An H.261-type coder with a fixed quantization step size was used to code the "Claire" sequence at 10 Hz. Figure 13 illustrates the SNR of the motion-compensated error signal of the sequence using the motion-estimation algorithms presented above. In this sequence, on the average there were only 29 blocks/frame out of 396 (common intermediate format picture frame) that the first stage of the conventional full-search method failed to produce significantly lower error signals and hence were further processed by generalized block matching. Despite the small number of compensated blocks, the overall improvement is always more than 1 dB.

The superiority of the generalized block-matching method in terms of both the prediction quality and bit rate becomes apparent in the last few frames of the sequence when the rotational type of motion of the head causes more blocks to be missed by the conventional full-search method. Figure 14 shows the quality of the reconstructed pictures with the conventional full search [Fig. 14(a)] and generalized block matching using perspective transformation [Fig. 14(b)]. The improved prediction is also reflected in the total bit rate, shown in Fig. 15. The overhead information for the transmission of the motion parameters is included in this graph. Both the perspective and the bilinear mappings reduce the total bit rate by almost 10%, and they have similar performance.

5.4 Broadcast Television Signals

To test the performance of generalized block matching in real-world sequences a new image sequence was recorded off the air from a commercial satellite channel. The sequence was taken from a shampoo advertisement with complex mo-
Fig. 11 Motion-compensated predictors: (a) full-search method, (b) affine, (c) perspective, and (d) bilinear transformed.

Fig. 12 Bit rate for different quantization step sizes.

Fig. 13 Motion-compensated picture quality for "Claire" sequence.

The sequence was coded with an H.261-type coder with variable-bit-rate output (i.e., without buffer control and a fixed quantization step size of 32). Three different algorithms were tested in the motion estimation process, namely conventional full-search block matching with a search window of ±8 pixels and two versions of the generalized block matching with bilinear and perspective mapping functions. Figure 16 shows the quality of the prediction picture in relation to the input frames (SNR). As expected, conventional block-matching tracks quite efficiently the translational motion caused by camera panning in frames 25 to 41, but it fails...
to compensate accurately the rotational motion found in the first frames.

Generalized block matching, on the other hand, retains an accurate prediction throughout the sequence because it works successfully with both translational and rotational motion. The perspective transformation seems to have a slightly better performance than the bilinear transformation especially in the highly active frames (i.e., frames 6 to 16). The better prediction reduces the bit rate of the coder, as shown in Fig. 17. The overhead information was around 700 bits/frame, which is included in the total bit rate.

Early experiments indicate that better results can be obtained by reducing the size of the matching blocks (e.g., using 8 × 8 pixel blocks). On the other hand, the small block size increases the overhead information (i.e., motion parameters) that must be transmitted. A possible solution to this problem might be to segment the moving object(s) and use a variable block size scheme.22

Fig. 14 Enlarged areas of motion-compensated prediction images: (a) full search and (b) perspective transform.

Fig. 15 Total bit-rate profile for “Claire” sequence.

Fig. 16 Motion-compensated picture quality for broadcast TV.

Fig. 17 Total bit-rate profile for broadcast TV.
6 Conclusions

In this paper a new approach to block-matching motion estimation has been introduced that uses general geometric transformations of the pixel coordinates. Based on this approach, a generalized block-matching method was designed. The method can effectively estimate not only translational motion but also other deformation of the moving objects.

The main advantages of the proposed method are its simplicity and the ability for easy and parallel implementation with only a few modifications to existing video-coding systems. The only drawback of the proposed method is the enormous computational load needed for the estimation of the motion parameters. The computational load can be significantly reduced if fast-search algorithms are used and geometric characteristics of the moving objects are exploited.

Early experimental results clearly indicate that the proposed method can be successfully applied to real image sequences. It is strongly believed that further investigations will justify the practical value of the method not only for bit rate reduction of the video coders but also as a basis for a more sophisticated motion analysis system.

References


Vassilis Seferidis received the BSc in 1986 and the MSc in 1989 from Aristotle University of Thessaloniki, Greece, and the PhD in 1993 from University of Essex, England. Since October 1989 he has been working within the Image Processing Group of Essex University as senior research officer on research projects funded by British Telecom and the European Community. His current research interests include video coding, motion analysis and packet video. He is a member of the Hellenic Physical Society, the European Physical Society, SPIE, IEE, and IEEE.

Mohammad Ghanbari received the BSc degree in electrical engineering from Aryamehr University of Technology, Tehran, Iran, in 1970 and the MSc degree in telecommunication and the PhD degree in electronics engineering from the University of Essex, Colchester, England, in 1976 and 1979, respectively. From 1970 to 1975 and 1979 to 1986 he worked at the Iranian Radio and Television Broadcasting organization in charge of various groups and sections. Since July 1986 he has been working in the Department of Electronic Systems Engineering, University of Essex, first as a research fellow studying the design of video codecs for asynchronous transfer mode (ATM) networks, then as a lecturer. His work is concerned with bandwidth compression of digital TV signals, motion estimation, packet video, and variable-bit-rate-coding techniques for transmission of digital TV signals over packet-switched networks.