Edge-Based 3-D Camera Motion Estimation with Application to Video Coding

A. Zakhor, *Member, IEEE,* and F. Lari

***Abstract*—The evolution of an image sequence obtained by a real camera from a real scene can be conceptually separated into two parts: 1) motion of the camera and 2) motion of the objects in a scene. Most existing motion estimation algorithms use the block matching algorithm (BMA) to model both the camera motion and local motion due to the objects. In doing so, successive frames are divided into small blocks and the movement of each block is approximately modeled by a translation, thus resulting in one motion vector per block. In this paper, we propose two classes of algorithms for modeling camera motion in video sequences captured by a camera. The first class can be applied in situations where there is no camera translation and the motion of camera can be adequately modeled by zoom, pan, and rotation parameters. The second class is more general in that it can be applied to situations where the camera is undergoing a translational motion, as well as a rotation and zoom and pan. This class uses seven parameters to describe the motion of the camera and requires the depth map to be known at the receiver. The salient feature of both of our algorithms is that the camera motion is estimated using binary matching of the edges in successive frames. In doing so, we show that unlike local motion estimation, edge matching can be sufficient in estimating camera motion parameters. Finally, we compare the rate distortion characteristics of our algorithms with that of the BMA and show that we can achieve similar performance characteristics as BMA, with reduced computational complexity.**

I. Introduction

MOTION COMPENSATION (MC) plays an important role in image compression applications such as video conferencing, video telephony, medical imaging, and CD-ROM storage. The basic idea is to take advantage of temporal redundancies between adjacent frames in an image sequence to reduce the information transmission rate. This is accomplished by estimating the displacement between frames of picture elements, which may be either uniformly sized blocks or individual pixels.

Ideally, the apparent motion in most moving sequences taken with a real camera can be attributed to either camera motion or the movement of the objects in a scene. The movement due to the camera is generally referred to as global motion, whereas the movement of the objects is called local motion. There are a number of advantages in separating these two classes of motion in MC algorithms. First, if there is no local movement in the scene and only the camera is moving, the dynamics of the resulting video sequence can be adequately modeled by only estimating camera motion parameters. Second, in more realistic situations where there is both local and global motion present in the video sequence, a great deal of redundancy between successive frames can be removed by estimating the parameters related to camera motion. This is because the number of degrees of freedom of camera movement is small compared to the complex motion of the objects in typical scenes. Specifically, camera motion parameters can be effectively used to predict the stationary parts of the scene, thus saving the bandwidth needed to otherwise transmit motion vectors for them.

Most existing video coding techniques such as CCITT's Recommendation H.261, also referred to as $p \times 64$, and the MPEG standards only use local motion estimation to form a prediction of the current frame [1]. Even though global motion estimation is not officially part of these two standards, several authors have exploited camera motion estimation in the context of video sequence coding. For example, Adolph and Buschmann [2] propose a coder for television video signals at 1.15 Mbps, where global MC is carried out before local one. In doing so, they assume the global motion to consist of only zoom and pan, and estimate it using a frame-matching algorithm. Their experimental results indicate that for the given rate of 1.15 Mbps, the quantization step size for coding error frames can be reduced by a factor of three if global MC is exploited, and that the quality of the coded scenes is considerably improved if global MC is applied.

In addition to [2], Baker [3] has shown that a two-stage global/local motion compensation approach improves motion prediction and reduces the amount of motion side information. Keesman [4], Hoetter [5], and Wu and Kittler [6] also show the advantages of global motion estimation schemes. Hoetter models zoom and pan, while Keesman, Wu, and Kittler model rotation as well as zoom and pan parameters. The techniques used for estimating the global motion parameters vary considerably. For instance [4]–[6] use pel recursive algorithms, while [2], [3] use luminance block matching.

From the above survey, it is clear that most of the existing global MC techniques used for video compression applications only consider zoom, pan, and possibly rotation as global motion parameters. Thus, an important parameter that is missing from these approaches is camera translation. This is not surprising, since there are some intrinsic difficulties involved with using camera translation parameters in generating motion

Manuscript received October 24, 1992; revised February 13, 1993. This work was supported by National Science Foundation PYI award MIP-9057466, ONR young investigator award N00014-92-J-1732, and Sun Microsystems. The associate editor coordinating the review of this paper and approving it for publication was Dr. H. H. Chen.

A slightly modified version of this material has appeared previously as a chapter in the book *Motion Analysis and Image Sequence Processing* (M. I. Sezan and R. L. Lagendijk, eds., Kluwer Academic Pubs.)

The authors are with the University of California, Berkeley, CA 94720.

IEEE Log Number 9210830.

compensated frames. As we will see in Section II, this has to do with the fact that unlike other global parameters such as zoom, pan, and rotation, translation parameters require the depth map of the pixels in a scene before they can be used to derive the motion compensated frames.

In this paper, we will consider a seven parameter camera model, including rotation and translation for global motion compensation of video sequences. In doing so, we will exploit the results on the problem of "structure and motion" recovery in the Computer Vision (CV) literature [7]–[10], and "camera calibration" in the photogrammetry [11]–[13]. These problems appear in diverse applications such as autonomous navigation, stereo reconstruction, robot vision, object recognition, scene analysis, and cartography [11], [14], [15]. The most basic formulation of "structure and motion" problem in CV consists of recovering translation and rotation parameters and structure of an object in three-dimensional (3-D) space. This problem is in fact equivalent to the problem of recovering the rotation and camera translation parameters of a camera, and the 3-D depth map of a stationary scene from its video sequence. Indeed, this is the motivation behind applying results on "structure from motion" to global camera parameter estimation.

The basic approach to structure from motion in CV literature consists of three steps [10]: extracting feature points, establishing correspondence by matching the features, and finally computing the structure and motion parameters based on the feature matches. Our basic approach to global motion estimation in this paper is similar to this. Since the number of matched features in our applications is large enough, we need not be concerned about uniqueness issues typically considered in CV applications [8]. Another inherent advantage of the large number of matches is that we can use simple linear algorithms, rather than iterative, or more complex nonlinear optimization techniques [8]–[10].

The two major steps in most existing global motion estimation algorithms, including ours are 1) estimating local motion vectors and 2) using the motion vectors in linear least-squares estimation of global parameters. One of our goals in this paper is to demonstrate that edge matching can be successfully used in step (a) of most global motion estimation algorithms. Specifically, we will show that edges alone are sufficient to determine the seven parameters in rotation/translation model as well as the parameters in the zoom/pan model. This can have important practical consequences since it implies that for camera motion estimation, the eight-bit luminance information in video sequences can be collapsed to one-bit edge information without loss of performance.

Edge matching is a subclass of token tracking algorithms used formotion estimation. More generally, in token tracking schemes, distinctive image features are detected and their correspondences tracked from frame to frame in a sequence. The features, such as corners, blobs, and straight lines, are assumed to arise from distinctive scene features. There are a number of motivations behind the use of edges as tokens for matching: First, it has been argued that the human visual system computes motion by temporal filtering of edge signal [16]. Second, it has been suggested that one of the advantages of using edges over raw irradiance schemes is that they are tied more closely to physical features [17].

In spite of these motivations, it has been found that the locations of edges by themselves are not sufficient for local motion estimation [18]. As a result, a number of other attributes such as orientation, strength, and curvature are used in conjunction with the location of edges in order to overcome the aperture ambiguity problem [19]–[21].

In this paper, we will demonstrate that unlike local motion estimation, global estimation can be accomplished by matching the location of edges in consecutive frames of a video sequence. In doing so, we compare the performance of our edge-based global motion estimation techniques with the well known block matching algorithm (BMA), which is extensively used in today's video compression standards. Note that BMA can also be considered a subclass of token tracking and matching algorithms. Specifically, the particular token which is matched in BMA is the intensity of raw images.

Compared to intensity matching, the major payoff in using edge-matching techniques is the simplicity involved in using one-bit edge information rather than eight-bit intensity information: this simplicity can potentially have implications on both hardware and software implementation of global motion-estimation techniques. In addition, since edges are more closely tied to physical features in a scene than individual pixel intensities, they are likely to be useful in other parts of typical video compression systems. Examples of these would be edge-based vector quantization, edge-based local MC, local MC in conjunction with global MC, segmented video coding, model-based coding, etc. [36].

To summarize, our goal in this paper is twofold: First, we will show that the seven-parameter camera model consisting of rotation and translation is applicable and useful in video compression applications. Second, we show that unlike local motion estimation, edges matching can be used to replace intensity-based block matching for global motion estimation. The outline of the paper is as follows: In Section II, we describe the global motion models used by our proposed algorithms presented in Sections III-A and III-B. In Section IV, simulations are used to examine the performance of our global motion estimation algorithms in the context of video compression. Section V compares the computational complexity of our algorithms with traditional intensity-based BMA algorithms, and Section VI includes conclusion.

II. Global Motion Models

In this section, we describe the specific model parameters for our global motion estimation algorithms. In general, camera movement can be decomposed into the following categories:

- Change of the camera focal length: zoom.
- Rotation around an axis normal to the camera axis: pan.
- Rotation around the camera axis.
- Translation along the camera axis.
- Translation in the plane normal to the camera axis.

Each of the above affects the temporal appearance of the resulting video sequence in a specific way. In this section, we describe the analytic relationship between the above pa-

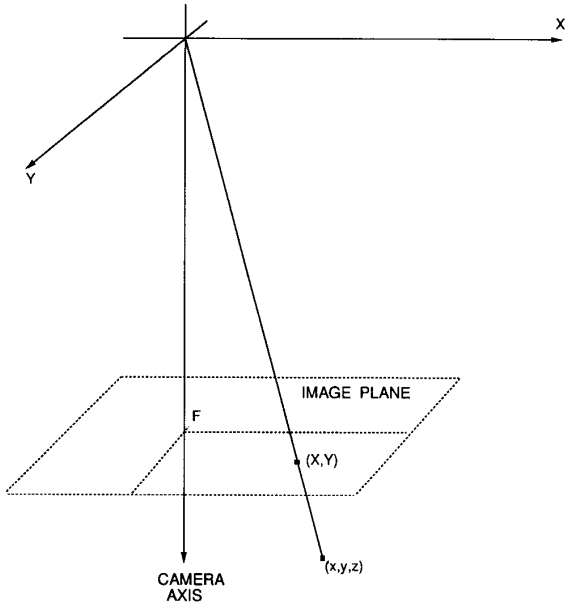


Fig. 1. Projection of the point (x, y, z) in 3-D space onto (X, Y) in the camera image plane.

rameters and pixel intensities of resulting video sequences of stationary scenes. These relationships can be exploited in video compression applications to predict future frames based on current frames and camera motion parameters.

As we will see in this section, the first three parameters—i.e., zoom, pan, and rotation—by themselves are enough to determine the apparent intensity change in a video sequence. On the other hand, for translation along, or in the plane normal to the camera axis, the translation parameters by themselves are not sufficient, and the depth map of the scene under consideration is also needed to estimate the changes in a video sequence. To distinguish between these two cases, we consider two classes of models: the first class, discussed in Section II-A, includes zoom, pan, and rotation, while the second class, discussed in Section II-B, considers translation and rotation.

A. Zoom, Pan, and Rotation Model

A video camera uses central projection to map the 3-D space onto the image plane at the focal point [22]. Using the notation in Fig. 1, the object space coordinates (x, y, z) of a point in the 3-D space and the image plane coordinates, (X, Y), of its image are related by the perspective transformation

$$X = \frac{Fx}{z} \qquad Y = \frac{Fy}{z}. \tag{1}$$

We now describe mathematical models for zoom and pan. A zoom is achange in the camera focal length. If X_1, Y_1 denote the image plane coordinates of a point (x, y, z)before zoom, and X_2, Y_2 the image plane coordinates of the same point after zoom, it is easily shown that

$$X_2 = \frac{F_2}{F_1}X_1 \qquad Y_2 = \frac{F_2}{F_1}Y_1 \tag{2}$$

where F_1 and F_2 are the focal lengths before and after the zoom. We define $c_1 = F_2/F_1$ to be the zoom parameter. A pan is a rotation of the camera around an axis parallel to the image plane. As shown in [3], if the rotation is small enough, the entire frame is displaced uniformly by the same vector:

$$\begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} = \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{3}$$

where t_x, t_y are the rotation angles around the x and y axes respectively. In arriving at the above equations, we implicitly assume that the rotation angles t_x and t_y are small so that $cos(t_x) \approx= 1$, $cos(t_y) \approx 1$, $sin(t_x) \approx t_x$ and $sin(t_y) \approx t_y$ [5]. Furthermore, we assume the rotation to be small enough so that $x_{1,2}t_x << z$ and $y_{1,2}t_y << z$ [5]. This implies that for a given rotation angle and field of view, the approximation becomes increasingly more valid for points with larger depth, z.

In addition to rotation around an axis parallel to the image plane,we also consider rotation around the camera axis, i.e., z axis in Fig. 1. A rotation of t_z around the camera axis can be described by

$$\begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} = \begin{pmatrix} \cos t_z & \sin t_z \\ -\sin t_z & \cos t_z \end{pmatrix} \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} \approx \begin{pmatrix} 1 & t_z \\ -t_z & 1 \end{pmatrix} \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} \tag{4}$$

where the above approximation assumes that the rotation angle t_z is small.

Combining zoom, pan, and rotation models described above, we arrive at the following model [22]:

$$\begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} = \begin{pmatrix} c_1 & c_2 \\ -c_2 & c_1 \end{pmatrix} \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} + \begin{pmatrix} c_3 \\ c_4 \end{pmatrix} \tag{5}$$

where c_1, c_2, c_3 and c_4 are given by

$$\begin{aligned} c_1 &= \frac{F_2}{F_1} \\ c_2 &= t_z c_1 \\ c_3 &= F_2 t_y + t_z F_2 t_x \\ c_4 &= F_2 t_x - t_z F_2 t_y. \end{aligned}$$

Thus, assuming there is no translation, the four parameters c_i completely describe all the motion in a video sequence of a stationary scene resulting from camera zoom, pan, and rotation.

B. Rotation and Translation Model

In this section, we will consider a seven-parameter model consisting of camera rotation and translation for global motion estimation. Before dealing with camera motion, consider the problem of recovering the motion parameters of a 3-D rigid body from the video sequence captured by a stationary camera.

$$R \equiv \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix} = \begin{pmatrix} n_1^2 + (1-n_1^2)\cos\theta & n_1 n_2(1-\cos\theta) - n_3\sin\theta & n_1 n_3(1-\cos\theta) + n_2\sin\theta \\ n_1 n_2(1-\cos\theta) + n_3\sin\theta & n_2^2 + (1-n_2^2)\cos\theta & n_2 n_3(1-\cos\theta) - n_1\sin\theta \\ n_1 n_3(1-\cos\theta) - n_2\sin\theta & n_2 n_3(1-\cos\theta) + n_1\sin\theta & n_3^2 + (1-n_3^2)\cos\theta \end{pmatrix} \quad (7)$$

This problem is extensively discussed in the CV literature [7]–[10]. It has been shown that any 3-D rigid body motion is equivalent to a rotation by an angle θ around an axis through the origin with directional cosines n_1, n_2, n_3 followed by a translation $(\Delta x, \Delta y, \Delta z)$,

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = R \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} + \vec{T} \quad (6)$$

where R is a 3×3 rotation matrix, shown as (7) at the top of the page.

$\vec{T}$ is the translation vector defined by

$$\vec{T} \equiv \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}$$

and (x_1, y_1, z_1) and (x_2, y_2, z_2) correspond to the coordinates of a point on the object before and after motion respectively [22]. It has been shown that once the elements of R are computed, n_1, n_2, n_3, θ can be found using a technique described in [8].

Our problem is slightly different from the above formulation in that the scene is assumed to be stationary while the camera is translating and rotating in the 3-D space. Thus, we must interpret the variables in the above formulation slightly differently. Specifically, we change the coordinate system so that the center of the camera is at origin, and therefore stationary at all times. Under these circumstances, the coordinates of the points in the 3-D scene will appear to be moving with respect to the camera. Under these conditions, (6) still holds, except for a change in the definition of the variables: the coordinates of a fixed point in 3-D space with respect to the camera is denoted by (x_1, y_1, z_1) before motion, and by (x_2, y_2, z_2) after motion, and its projection in the image plane moves from (X_1, Y_1) before motion to (X_2, Y_2) after motion. In this new interpretation, the rotation matrix R and translation vector T will correspond to the rotation and translation of the camera rather than the rigid body in the scene.

Based on (6), the seven parameters $n_1, n_2, n_3, \theta\ \Delta x, \Delta y, \Delta z$ are sufficient for recovering (x_2, y_2, z_2) from (x_1, y_1, z_1). However, in our video application, the ultimate goal is to recover, estimate, or predict the *image plane* coordinates in frame $n+1$ from those in frame n. Specifically, our modeling goal is to relate (X_1, Y_1) to (X_2, Y_2) in terms of camera motion parameters. This way, once the camera motion is estimated from the image sequence, it can be effectively used for predictive coding.

Since the image and 3-D space coordinates are related to each other through (2), using (6), we can relate image coordinates before and after motion. Using the third line of (6) to compute the ratio $\frac{z_1}{z_2}$, using (2) to express x_1, x_2, y_1, and y_2 in the first two equations of (6) in terms of X_1, Y_1, X_2 and Y_2, and assuming for simplicity $F = 1$, we obtain [8]

$$X_2 = \frac{(r_1 X_1 + r_2 Y_1 + r_3) z_1 + \Delta x}{(r_7 X_1 + r_8 Y_1 + r_9) z_1 + \Delta z} \quad (8)$$

$$Y_2 = \frac{(r_4 X_1 + r_5 Y_1 + r_6) z_1 + \Delta y}{(r_7 X_1 + r_8 Y_1 + r_9) z_1 + \Delta z}. \quad (9)$$

Thus, camera rotation and translation results in a relationship between image points (X_1, Y_1) and (X_2, Y_2) which is not only dependent on their coordinates, but also on the depth z_1 of the point in 3-D space whose projection in the image plane is at coordinates (X_1, Y_1). Note that if there is no translation, i.e., $\Delta x = \Delta y = 0$, then from (8) and (9), (X_2, Y_2) can be easily related to (X_1, Y_1) without requiring any depth information. Thus, it is the translation, but not the rotation that requires the depth information.

III. Algorithm Description

In this section, we will propose algorithms for each of the camera models discussed in Section II. Specifically, Section III-A describes a global motion estimation and compensation algorithm based on zoom, pan, and rotation, and Section III-B deals with a seven-parameter global motion estimation and compensation algorithm for camera translation and rotation. As will be seen, there is a great deal of similarity between the two algorithms.

A. Zoom, Pan, and Rotation Algorithm

In this section, we describe our edge-based algorithm for finding zoom, pan, and rotation parameters of the camera. As mentioned earlier, there are a number of existing zoom and pan detection algorithms [2], [3], [23], most of which consist of two steps: 1) estimate local motion vector for each block; 2) use the location of blocks in the current frame and their corresponding motion compensated location in the next frame in (5) in order to find the linear least-squares estimate of zoom and pan parameters [3]. Since zoom and pan are smoothly varying processes, in many practical situations it is sufficient to use the motion vectors between every few frames in finding the least-squares estimate of zoom and pan parameters. In addition, since displacement vectors between every few frames are likely to be larger than between consecutive frames, pel recursive algorithms [5], [6] are not suitable; block matching algorithms can be used, but since their search area has to be large, they are computation intensive.

In this section, we propose an edge-based approach to global motion estimation of zoom, pan, and rotation parameters to be

used in predictive coding of video. One motivation behind our algorithm is to reduce the computational intensity of existing luminance-based estimation algorithms. This will be discussed in more detail in Section V. The flowchart of our approach is shown in Fig. 2. If A and B denote the two frames between which zoom and pan parameters are to be estimated, then the steps of our algorithms can be described in the following way:

1) Find the edges in frames A and B: We have chosen an edge detection algorithm described in Section V-A for its speed and precision. As will be seen, our edge-detection algorithm consists of smoothing and finding directional derivatives [24]–[26].
2) Search for "features" in frame A: We refer to an 8×8 block as a "feature" if it contains more than 8 edge pixels. This is done in order to prevent false matches in the next step. To avoid aperture ambiguity, blocks with strictly vertical and horizontal edges are discarded. Note that the minimum distance between the coordinates of the location of two features is $(8, 8)$.
3) Search for perfect matches of the frame A features in frame B: Two edge features are said to be perfectly matched if they are identical in the binary domain. To maximize the computation speed, the search area is centered around the point the feature would have moved to, given the previously detected values of zoom and pan. If no previous information is known, the search begins around the feature position in frame A. If a match is found the coordinates of the feature in frame A, $(X_{i,A}, Y_{i,A})$ and the matched coordinates in frame B, $(X_{i,B}, Y_{i,B})$ are stored for use in the next step.
4) Find linear least-squares estimate of c_1, c_2, c_3 and c_4 by fitting the model shown in (5) to the K pairs of coordinates obtained in the previous step. This can be accomplished by minimizing the following error expression:

$$Error = \sum_{n=1}^{K} [(X_{n,B} - c_1 X_{n,A} - c_2 Y_{n,A} - c_3)^2 + (Y_{n,B} - c_1 Y_{n,A} + c_2 X_{n,A} - c_4)^2] \quad (10)$$

setting $\frac{\partial E}{\partial c_i}$ for $i = 1, 2, 3$ to zero,and solving a linear system of equations to estimate c_1, c_2, c_3, and c_4.
5) Remove "outlier" coordinates and repeat the previous step until the iterations converge: by outlier coordinates, we mean the ones whose residual errors are more than one standard deviation away from the average error of all the coordinate pairs. We consider the algorithm to have converged if the parameter estimates do not change substantially from one iteration to the next. For all the examples we present in this paper, the above procedure converges in 2 to 3 steps.
6) Apply the global motion parameters obtained in the above step to the interpolated version of frame A in order to obtain "motion"-compensated prediction for frame B. This is necessary because even though the motion vectors used in estimating global motion parameters have pixel accuracy, the effective motion vectors for each pixel does not necessarily have pixel accuracy.

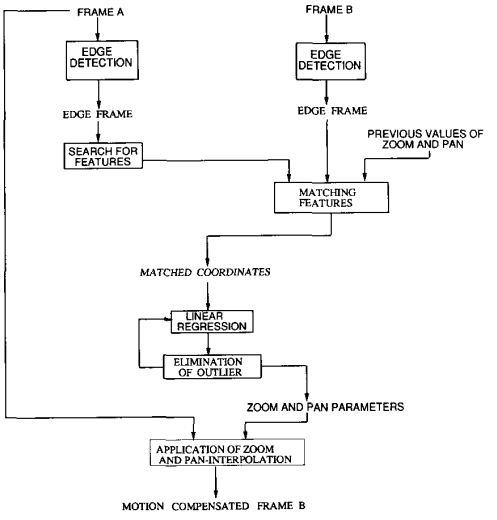


Fig. 2. Flowchart of the zoom/pan algorithm of Section III-C1.

As a result, the displaced pixels according to the global parameters do not necessarily coincide with the sampling grid. Thus, we are forced to carry out interpolation to derive a motion-compensated prediction of future frames. We adopt a very simple interpolation scheme in which the intensity value of a pixel (i, j) on the sampling grid is given by

$$I(i,j) = \frac{\sum_{k \in A_{ij}} d_k I_k}{\sum_{k \in A_{ij}} d_k} \quad (11)$$

where A_{ij} is a $2\Delta \times 2\Delta$ area centered around (i, j) pixel, Δ is the spacing on the sampling grid, d_k is the distance of the kth point with intensity I_k from the (i, j) pixel.

Note that the linear least squares approach of step 4 has already been reported in the literature [3], [6], [5], [4], [28]. Thus the major difference between our approach and existing global motion estimation techniques is the edge-based feature matching, i.e., steps 1-3.

B. Rotation and Translation Algorithm

In this section, we will propose an algorithm for estimating rotation and translation parameters of a camera based on the model described in Section II-B. As mentioned earlier, camera translation parameter requires the knowledge of the depth map in order for it to be effectively used in predictively coding. Thus, our proposed algorithm in this section must recover both the seven parameters corresponding to translation and rotation of the camera and the depth map.

There are a number of algorithms in the CV literature for computing translation and rotation of rigid bodies [7], [8], [10]. The algorithm we propose in this section is heavily based on the approach proposed by Tsai and Huang in 1984 [8]. While the main objective in [8] is to derive conditions for unique

$$E = \begin{pmatrix} \Delta z \cdot r_4 - \Delta y \cdot r_7 & \Delta z \cdot r_5 - \Delta y \cdot r_8 & \Delta z \cdot r_6 - \Delta y \cdot r_9 \\ \Delta x \cdot r_7 - \Delta z \cdot r_1 & \Delta x \cdot r_8 - \Delta z \cdot r_2 & \Delta x \cdot r_9 - \Delta z \cdot r_3 \\ \Delta y \cdot r_1 - \Delta x \cdot r_4 & \Delta y \cdot r_2 - \Delta x \cdot r_5 & \Delta y \cdot r_3 - \Delta x \cdot r_6 \end{pmatrix} \equiv \begin{pmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{pmatrix} \tag{15}$$

recovery of rigid body objects in scene analysis, our objective is to use the camera parameters in predictive video coding.

Recall from Section II-B that if the projection of a point P in 3-D space onto the image plane is (X_1, Y_1) before camera motion and (X_2, Y_2) after camera motion, then the relationship between these coordinates is given by (8) and (9). Rewriting z_1 in each of these equations in terms of motion parameters and image plane coordinate, we obtain

$$z_1 = \frac{\Delta x - \Delta z X_2}{X_2(r_7 X_1 + r_8 Y_1 + r_9) - (r_1 X_1 + r_2 Y_1 + r_3)} \tag{12}$$

from (8) and

$$z_1 = \frac{\Delta y - \Delta z Y_2}{Y_2(r_7 X_1 + r_8 Y_1 + r_9) - (r_4 X_1 + r_5 Y_1 + r_6)} \tag{13}$$

from (9). Equating the right-hand sides of the above equations, we obtain [8]

$$[X_2 \quad Y_2 \quad 1] \; E \begin{bmatrix} X_1 \\ Y_1 \\ 1 \end{bmatrix} = 0 \tag{14}$$

where the matrix E is referred to as the "essential matrix" and is given by (15) at the top of the page.

From (14) it is clear that the matrix E can be determined only to within a scale factor. Since elements of E are linear in $\Delta x, \Delta y$, and Δz, this implies that the translation parameters can be determined to within a scale factor.This can also be seen by noting that z_1 in (12) and (13) is linear in the translation parameters. This ambiguity is not merely a mathematical artifact; rather, it is inherent to the problem at hand, and can be physically explained by considering that an object at depth $d_1$translating at T_1 appears the same to the camera as one at depth βd_1translating at βT_1.

To avoid this ambiguity, without loss of generality, we set one of the e_i to 1. The only potential problem this might cause is in situations where the particular e_i set to 1, is actually zero. However as we will see later, such problems can be easily detected numerically, and can therefore be avoided.

The flowchart of our proposed global MC algorithm is shown in Fig. 3. Note that the flowchart implicitly assumes that the depth map for the first frame of the scene is available at the receiver. In spite of this, the depth map need not necessarily be *transmitted* to the receiver. For instance, if the first two frames of each scene are intra-frame coded, the transmitter and the receiver could compute the same depth map independently. At the end of this section, we will describe a way of determining the depth map for the first frame.

If A and B denote the two frames between which the global motion is to be estimated, then the steps of our algorithms

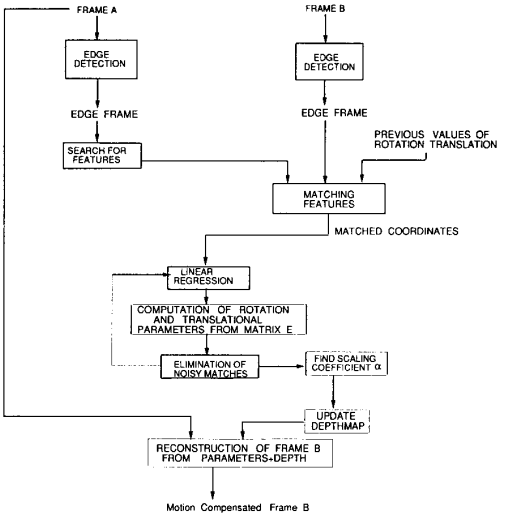


Fig. 3. Flowchart of the rotation/translation algorithm in Section III-C2.

shown in the flowchart of Fig. 3 can be described in the following way:

1) Detect edges similar to step 1 in Section III-A.
2) Detect features similar to step 2 in Section III-A, except that the minimum distance between the location of features can be $(1, 1)$ rather than $(8, 8)$. This is because, even though the window size for detecting features is 8×8, unlike in Section III-A, we allow them to overlap, thus resulting in a larger number of features and reduced noise sensitivity.
3) Match features similar to step 3 in Section III-A.
4) Find linear least-squares estimates of e_i by fitting the model shown in (14) to the K pairs of coordinates obtained in step 3. This can be accomplished by minimizing the following error expression:

$$Error = \sum_{i=1}^{K} [(X_{i,B} X_{i,A} e_1 + X_{i,B} Y_{i,A} e_2 + X_{i,B} e_3 + Y_{i,B} X_{i,A} e_4 + Y_{i,B} Y_{i,A} e_5 + Y_{i,B} e_6 + X_{i,A} e_7 + Y_{i,A} e_8 + e_9)^2] \tag{16}$$

where $(X_{i,B}, Y_{i,B})$ denotes coordinates of a feature in frame B which is matched to a feature in frame A with coordinates $(X_{i,A}, Y_{i,A})$.

As mentioned earlier, in solving the above linear least-squares problem, we have to set one of the e_i to one in order to remove a scaling ambiguity. As an example, consider the case where e_8 is chosen to be one. Considering (15), this choice is only problematic when $e_8 \equiv \Delta y r_2 - \Delta x r_5 = 0$, i.e., in either of the following two situations: 1) the only non-zero translation

is along the z direction, i.e, $\Delta z \neq 0$ and $\Delta x = \Delta y = 0$; 2) translation along the x direction is zero and there is no camera rotation, i.e., $\Delta x = 0$, $\theta = 0$. We have found numerically that in physical situations where the choice of $e_i = 1$ is invalid, then the resulting translation parameters become unrealistically large, signaling an unappropriate choice of $e_i = 1$. Under these conditions, one simply switches from one e_i to another in order to remove the scaling ambiguity.

Once an appropriate parameter e_j has been set to one,we can find the remaining parameters by minimizing the error expression in (16), or equivalently by setting $\frac{\partial E}{\partial e_i}$ for $i = 1,..9, i \neq j$ to zero, and solving a linear system of equations.

5) Once the matrix E is computed to within a scale factor, we find the seven rotation and translation parameters by computing the singular value decomposition (SVD) of the 3×3 matrix E[8]. If the SVD of E is given by $E = U\Sigma V^T$, then there are two solutions for the rotation matrix, R, given by

$$R = U \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & s \end{pmatrix} V^T \tag{17}$$

or

$$R = U \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & s \end{pmatrix} V^T \tag{18}$$

where $s = \det(U)\det(V)$, and one solution for the translation vector, up to a scale factor α, given by

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \alpha \begin{pmatrix} (-\phi_1^T\phi_1 + \phi_2^T\phi_2 + \phi_3^T\phi_3)^{\frac{1}{2}} \\ (\phi_1^T\phi_1 - \phi_2^T\phi_2 + \phi_3^T\phi_3)^{\frac{1}{2}} \\ (\phi_1^T\phi_1 + \phi_2^T\phi_2 - \phi_3^T\phi_3)^{\frac{1}{2}} \end{pmatrix} \tag{19}$$

where ϕ_i is the ith row of E for $i = 1, 2, 3$. Only one of the two solutions for R together with the appropriate sign for α, will yield positive z_1 and z_2. Since the object must be in front of the camera, and therefore have a positive depth, the solution is unique to within a positive scale factor.

6) Remove the outliers using the following steps: a) recover rotation and translation parameters from the e_i; b) use the matches in (12) and (13) to find z_1; c) use (8) and (9) to compute new values for (X_2, Y_2); d) find the error between these newly computed (X_2, Y_2) and those that were actually used in the matching process; e) remove outliers by discarding matched features which have errors of more than one standard deviation.

7) Compute the scaling factor α: At this point, an important observation needs to be made. While both the depth z_1 and the coefficients $\Delta x, \Delta y, \Delta z$ are defined to within a common scale factor for any pair of frames, consecutive scale factors should be chosen in such a way that they result in consistent depths in consecutive frames. This implies that there is effectively only one degree of freedom for the entire sequence, as far as the scaling factor is concerned. Thus, changing the value of this degree of freedom, does not affect the relative depth of various frames, but only fixes their absolute depth. For the sake of argument, we assume that this degree of freedom hasbeen fixed to a specific, but arbitrary value, so that the absolute depth map of frame $n-1$ is given by $\vec{z}_{n-1}$. Also, assume that step 5 has been applied to find the translation motion parameters between frames $n-1$ and n up to scaling factor, α_n. In addition to motion parameters, a by product of step 5, is α_n dependent depth map for frame $n-1$, which we will refer to as $\vec{z}_{n-1}(\alpha_n)$. Our approach is to choose α_n in such a way that $\vec{z}_{n-1}(\alpha_n)$ becomes as close as possible to its absolute depth map $\vec{z}_{n-1}$. This is done by solving another linear regression problem in which the depth of the pixels in $\vec{z}_{n-1}$ are matched to those in $\vec{z}_{n-1}(\alpha_n)$. Thus, we estimate

$$\alpha_n = \frac{\sum_{i \in K} \vec{z}^i_{n-1}}{\sum_{i \in K} \vec{z}^i_{n-1}(\alpha_n)}$$

where the set K consists of the pixels for which depth is defined in both $\vec{z}_{n-1}$ and $\vec{z}_{n-1}(\alpha_n)$. Once α_n has been determined, the motion parameters between frames $n-1$ and n are uniquely determined, and can therefore be applied to the absolute depth map of frame $n-1$ in order to compute the absolute depth map of frame n through (6). The absolute depth map of frame n can then be used to determine subsequent scaling factors such as α_{n+1}.

8) Transmit the seven parameters to the receiver so that the decoder can update the existing depth map based on the new set of rotation and translation camera parameters. Clearly, the depth map associated with the uncovered points in frame $n+1$ cannot be determined based on the depth map of frame n and the camera parameters.

9) Apply the rotation and translation parameters and the updated depth map to frame A to obtain a global motion compensated prediction for frame B. Equations (8) and (9) are used to compute the new coordinates, (X_2, Y_2), of the point in frame B which corresponds to the point (X_1, Y_1) in frame A. In other words, the intensity value of pixel (X_2, Y_2) in frame B is assigned the same value as that of pixel (X_1, Y_1) in frame A, where the coordinates are related through (8) and (9). Since this mapping does not guarantee (X_2, Y_2) to fall on a rectangular sampling grid, interpolation is necessary. Our approach to interpolation is similar to the one described in Section III-A for the zoom, pan, rotation algorithm.

10) Once the motion-compensated version of B is computed, the error between the actual and compensated frame B is quantized and coded for transmission to the receiver. We will describe the particular quantization and coding scheme we consider in our experimental results in Section IV.

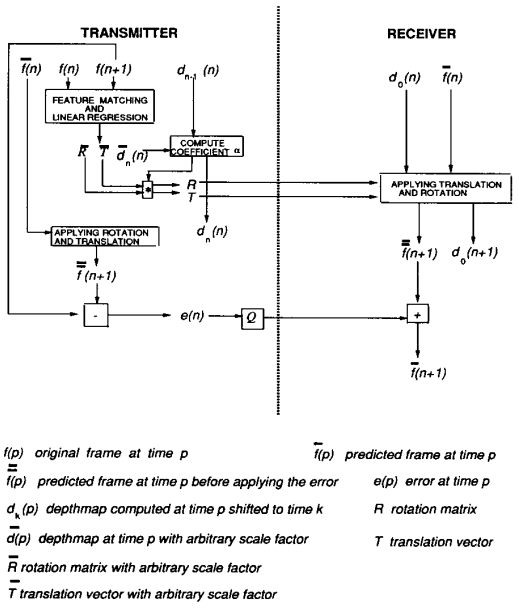


Fig. 4. Block diagram of a transmitter/receiver system employing the 3-D camera motion estimation algorithm of Section III-B.

Having described our basic algorithm, we now explain our approach to arriving at a depth map for the first frame. If we use the above algorithm to recover the translation and rotation parameters between frames 1 and 2, then (12) or (13) can be used to arrive at the depth z_1 for the pixel (X_1, Y_1) in frame 1. Since the depth map is only explicitly calculated for the first frame in a given scene, it is important for it to be dense; otherwise, the number of predicted pixels in motion compensated frames will not be dense, resulting in large error frames to be quantized and coded. However since the dense depth map need only be calculated once for each scene, this is only a one time cost.

The block diagram of a transmitter/receiver system employing the above motion estimation algorithm is shown in Fig. 4.

IV. Experimental Results

In this section, we will describe experimental results on the two algorithms described in Section III. Specifically, Section IV-A includes results on the algorithm of Section III-A, and Section IV-B includes results on the algorithm of Section III-B.

A. Zoom and Pan Results

We have applied the zoom and pan algorithm to three 720x480 pixel frame sequences. The "HDTV" sequence consists of 28 luminance frames with synthetic zoom and pan. The other two sequences are two pieces of the "Ping-Pong" sequence: "Ping-Zoom" with 40 frames of pure zoom along with random foreground motion, and "Ping-Pan" with 80 frames of pure pan along with random foreground motion. The zoom and pan parameters for "HDTV," "Ping-Pan," and "Ping-Zoom" are determined every 3, 10, and 3 frames respectively. Typical frames for the HDTV and Ping-Zoom and their corresponding edge maps are shown in Figs. 5 and 6. Note that edges corresponding to the ping-pong table in Ping-Zoom are jagged. This can be attributed to camera motion and to the fact that the edge-detection algorithm was applied to a frame, rather than to a field. Even though the found edges are imperfect, as we will see later, they do not affect the accuracy with which we can determine the zoom and pan parameters.

Figures 7, 8, and 9 show the predictive mean squared error (MSE) for globally motion compensated frames obtained via traditional luminance-based BMA and our edge-based algorithm for the "HDTV," "Ping-Pan," and "Ping-Zoom" sequences respectively. By globally motion compensated frames via traditional BMA, we mean frames obtained by applying steps 4, 5, and 6 of our algorithms to the motion vectors obtained by full search luminance match. The solid curves in all the three figures denote the non-compensated frame difference (FD) without any zoom or pan compensation. As seen, our edge-based algorithm has comparable MSE performance to traditional intensity-based BMA.

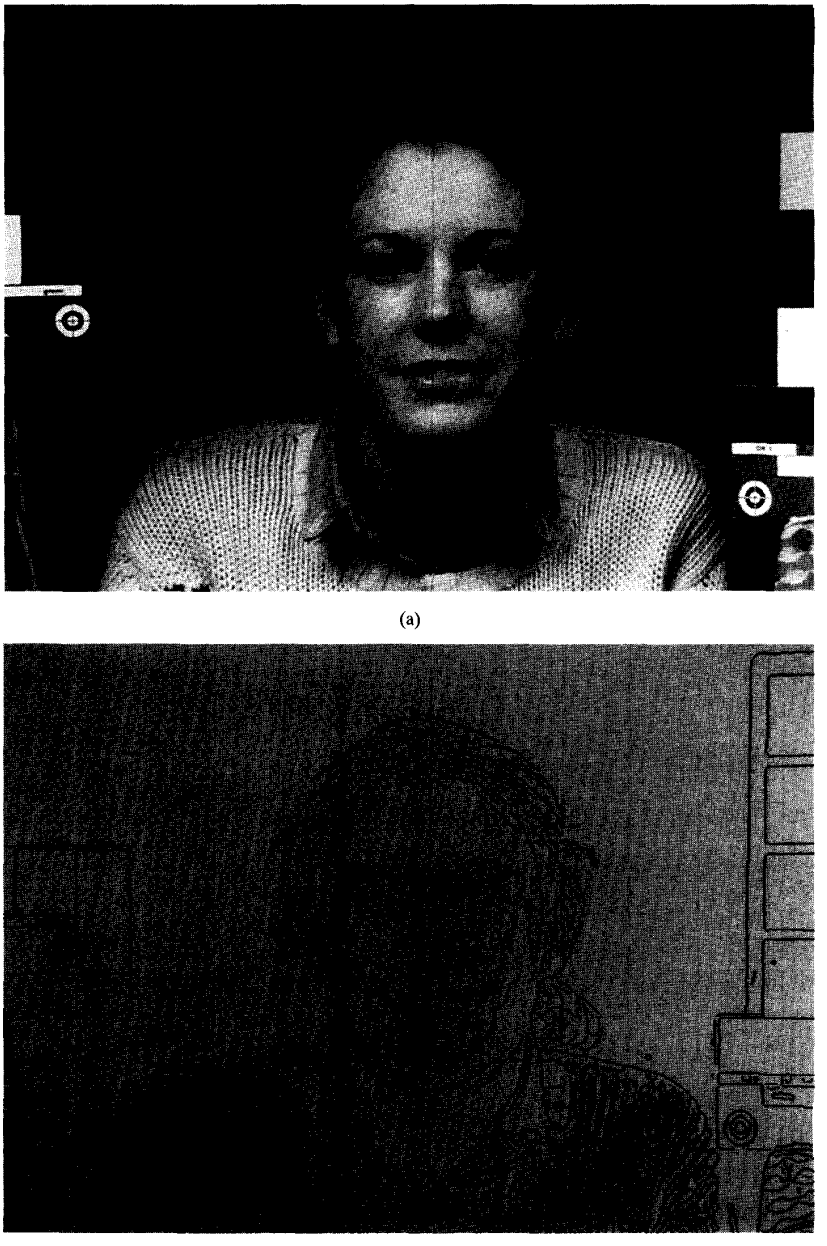
A few comments about the above results are in order: First, the sudden luminance change between frames 9 and 10 or the "HDTV" sequence does not affect the performance of our edge-based technique, while the luminance method yields a large number of false matches resulting in substantially lower MSE performance. Second, we have found that iterations do improve the performance of our edge-based techniques.[1]

Visual results shown in Fig. 10 fully confirm our results based on MSE plots. This figure shows the FD of intensity-based BMA, our edge-BMA-based technique, and the non-compensated FD for one frame of the Ping-Zoom sequence. As seen, the edge-based zoom-and-pan compensation results in approximately the same FD as conventional intensity-based BMA. Furthermore, the stationary objects such as the picture on the wall, and the ping-pong table result in small FD, while the region corresponding to the player results in larger FD due to the player movement in addition to camera zoom. Similar results have been obtained for the Ping-Pan and HDTV sequence.

B. Rotation and Translation Results

We have applied our algorithm of Section III-B to the first 40 frames of the "Flowers" sequence, which seem to only exhibit translational movement along the x direction. The dimensions of each frame is 486×720. This sequence exhibits a great deal of depth variation, featuring a tree close to the camera, a row of houses further away, and flowers in between. The first frame for this sequence and its corresponding edges are shown in Fig. 11, and its depth map is shown in Fig. 12. To visualize the depth map, we have heavily quantized it. As seen, the depth associated with the tree is smaller than that of the houses; in addition, the depth associated with the flowers is in between that of the houses and the tree. Finally, the sky has the largest depth as it is to be expected.

[1] Experiments show that this is also the case for traditional intensity-based BMA-based zoom-and-pan algorithms.



(a)

(b)

Fig. 5. (a) A typical frame of the "HDTV" sequence. (b) Edges of the image shown in (a).

We compare the compression capability of our algorithm with that of an intensity-based BMA with local motion compensation for each block. In our algorithm, the quantities to be quantized and coded for transmission are the rotation and translation parameters and the error frames. The corresponding quantities for a BMA-based algorithm, with forward prediction only, is the motion vectors and the error frames. The error frames in both cases are quantized and coded in a fashion similar to MPEG [29]. Specifically, we apply DCT to 8×8 blocks of error frames, discard coefficients below a certain threshold, quantize with a uniform quantizer, and variable-length-code the quantized coefficients in conjunction with the location of discarded coefficients. The motion vectors in the local motion estimation technique are also treated in a similar way to MPEG. Specifically, the motion vectors are first computed for each 16×16 block, and then differentially coded with variable length coder described in MPEG documents [29].

In applying our rotation/translation algorithm of Section III-B to the flower sequence, the algorithm successfully finds all the rotation and translation parameters, except for Δx

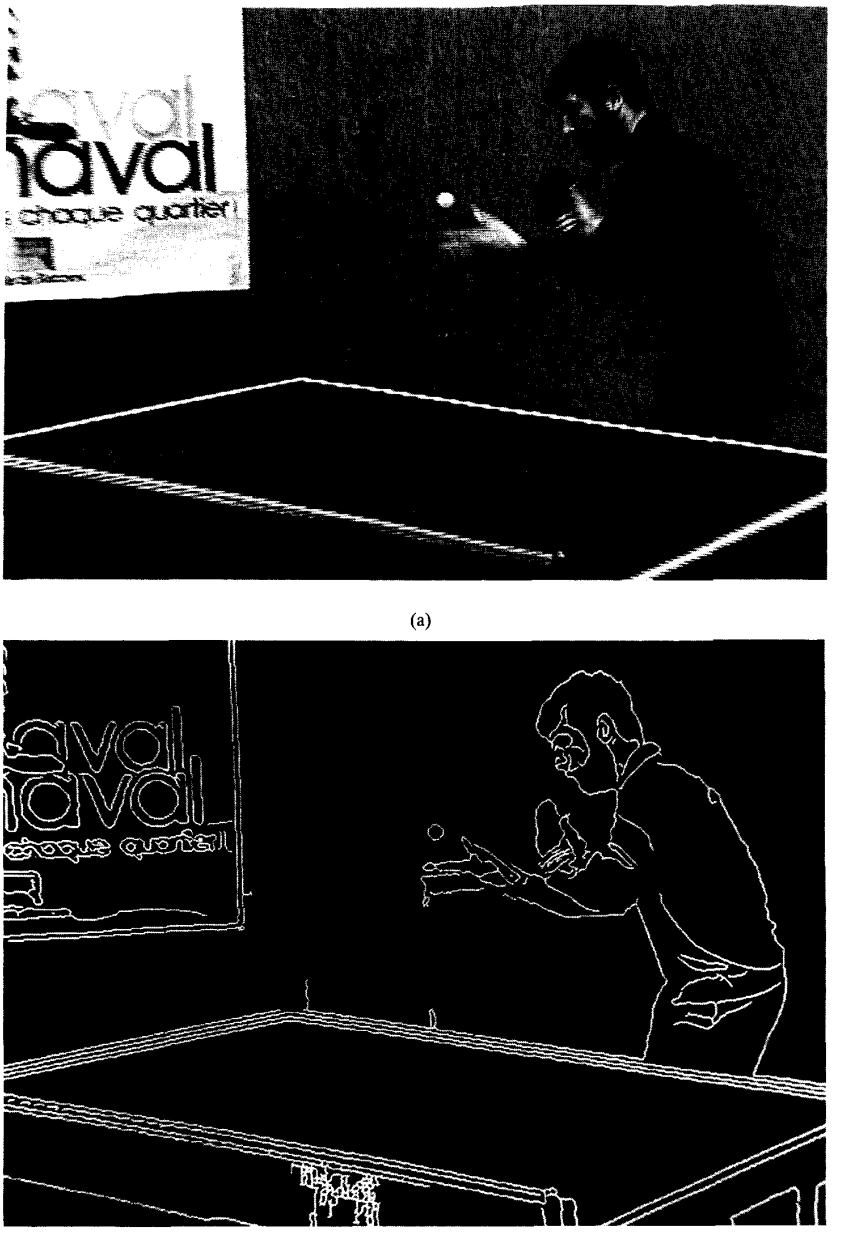


Fig. 6. (a) A typical frame of the "Ping-Zoom" sequence. (b) Edges of the image shown in (a).

to be zero.[2] The motion-compensated residual error based on rotation/translation algorithm is found to have almost the same energy as that of local motion estimation based on BMA. Figures 13 and 14 show the resulting bit rates in bits per pixel, and distortion in intensity per pixel for 1) the edge-based rotation, translation scheme of Section III-B; 2) intensity-based BMA with local MC; 3) edge-based zoom, pan, rotation technique of Section III-A. As seen, the bit rate of the three schemes is almost identical. The distortion for the rotation/translation model is similar to that of BMA, while that of the zoom/pan/rotation model is slightly higher particularly for earlier frames. The visual impression of the reconstructed frames at the receiver seem to be more or less the same for all three techniques. It is interesting to note that even though the zoom/pan/rotation model is inappropriate for this sequence, its rate/distortion characteristics is comparable to those of the BMA-based algorithm and rotation/translation

[2]Since the number of rotation and translation parameters needed per frame is only seven, we can assume that these parameters are quantized arbitrarily finely in our experimental results.

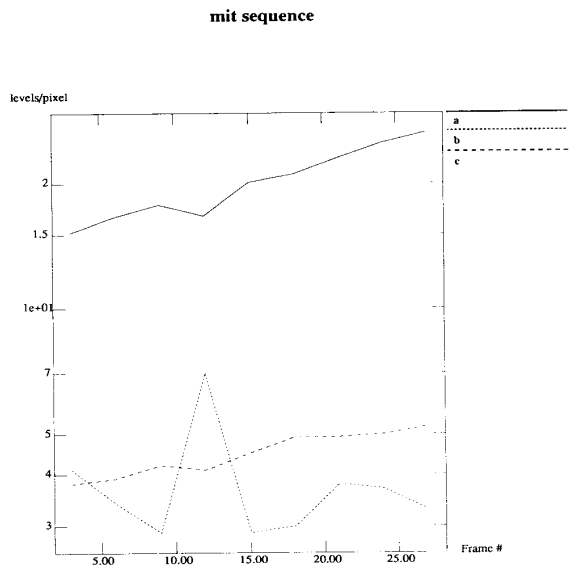


Fig. 7. Zoom/pan compensated FD for the "HDTV" sequence: (a) Non-compensated FD. (b) Global motion compensated FD with luminance matching. (c) Global motion compensated FD with edge matching.

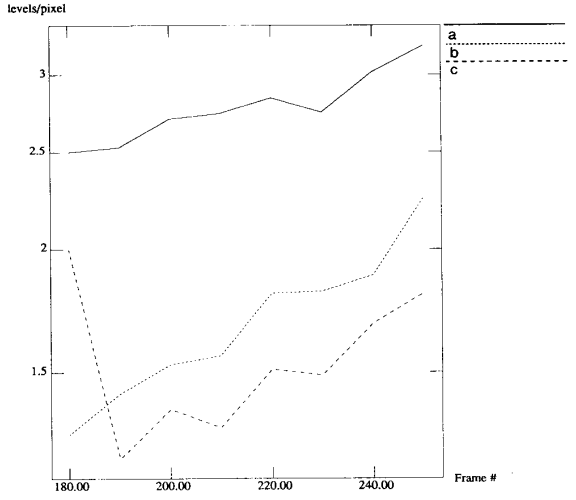


Fig. 8. Zoom/pan compensated FD for the "Ping-Pan" sequence: (a) Non-compensated FD. (b) Global motion compensated FD with luminance matching. (c) Global motion compensated FD with edge matching.

model. This is true even when we remove the rotation part of the zoom/pan/rotation algorithm and apply only a zoom/pan model. This can be partially explained by considering that the computed pan parameter along the x direction is around 3 for most frames, which is approximately the average motion experienced by the most pixels in the scene, such as the pixels associated with houses.

Another observation to be made is that the peak bit rate exhibited by BMA-based algorithm and the rotation/translation algorithm around frame 31 is absent in the zoom/pan algorithm. A similar trend was observed at other bit rates for these three algorithms.

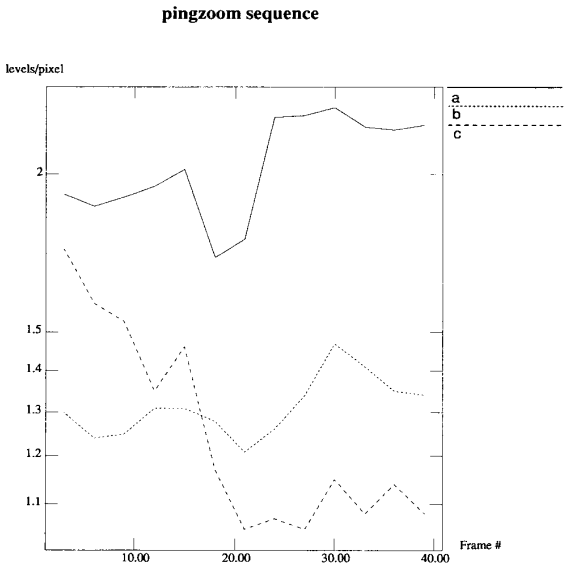


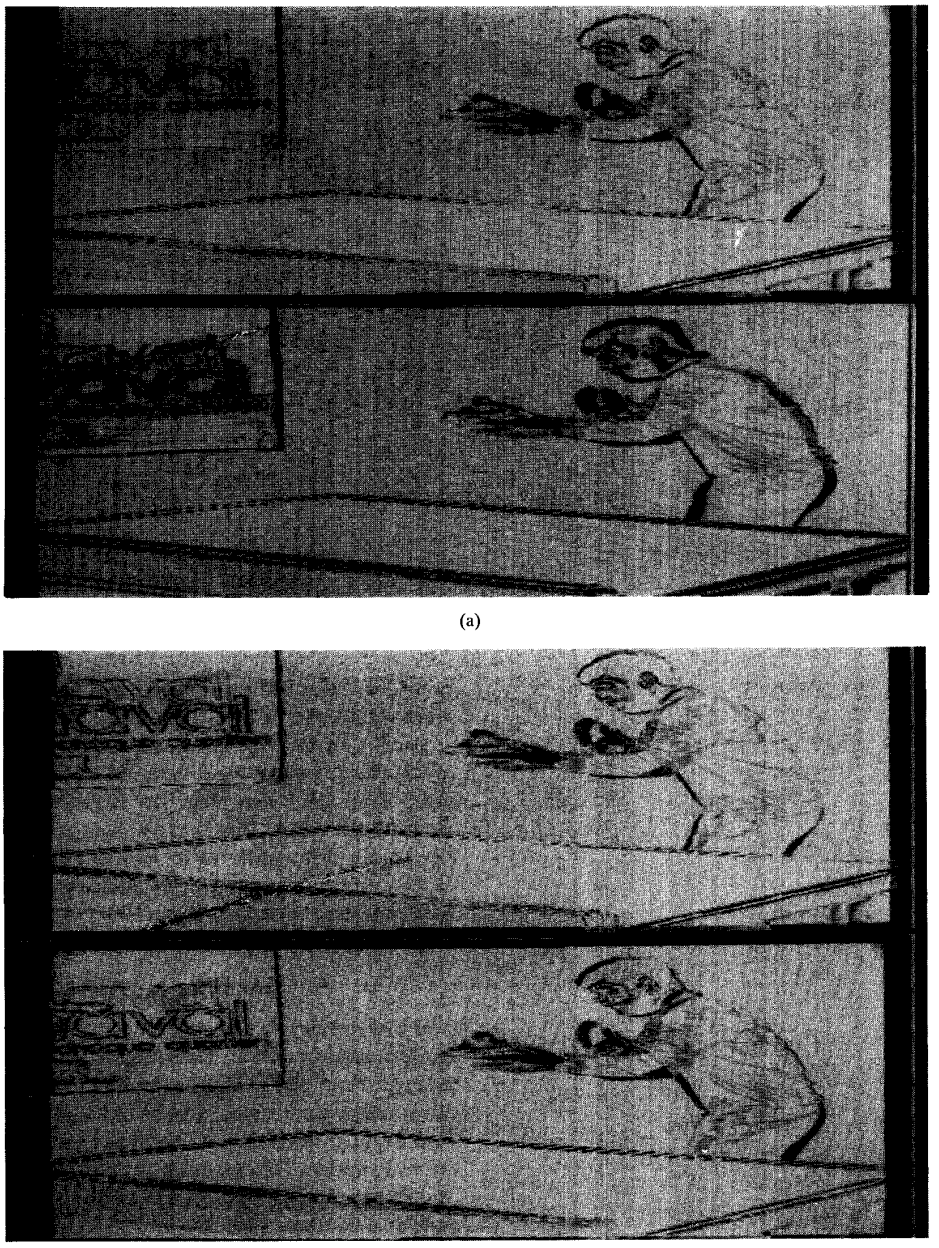
Fig. 9. Zoom/pan compensated FD for the "Ping-Zoom" sequence: (a) Non-compensated FD. (b) Global motion compensated FD with luminance matching. (c) Global motion compensated FD with edge matching.

Finally, we have found experimentally that the performance of the model-based algorithms of Section III remains the same if the edge matching part is replaced with traditional intensity matching. This is encouraging since it confirms our hypothesis that unlike local motion estimation, for global motion estimation, binary edge matching is sufficient and that 8 bit luminance information can be replaced with one bit edge information.

V. Computational Complexity

In this section, we compare the computational complexity of our techniques based on edges to that of intensity-based BMA for both fixed and floating point arithmetic. In doing so, we only compare the complexity of our algorithm to that of full search BMA, rather than hierarchical algorithms such as three-step, logarithmic, or conjugate gradient algorithms. There are two reasons behind this: First, in contrast with the hierarchical motion algorithms [30], [31], a major portion of our edge-based approach does not require intermediate decisions and therefore does not require programmable hardware, and can potentially be implemented with highly parallel architectures. Specifically, as we will see, the major part of edge detection consists of convolution, which can be mapped into regular structures [32], while in hierarchical techniques the number of intermediate decisions *per block* is proportional to the logarithm of the dimension of the search area. The second disadvantage of hierarchical algorithms in intensity-based local motion estimation algorithms is their inherent inaccuracy.

The outline of this section is as follows: in Section V-A, we describe the particular edge-detection algorithm we used for the experimental results of the previous section and review its computational complexity. In Section V-B, we



(a)

(b)

Fig. 10. "Ping-Zoom" sequence: comparison of edge BMA-based global motion compensated frame difference to (a) non-compensated frame difference; (b) luminance BMA-based global motion compensated frame difference. Edge-based method is the top half for both (a) and (b).

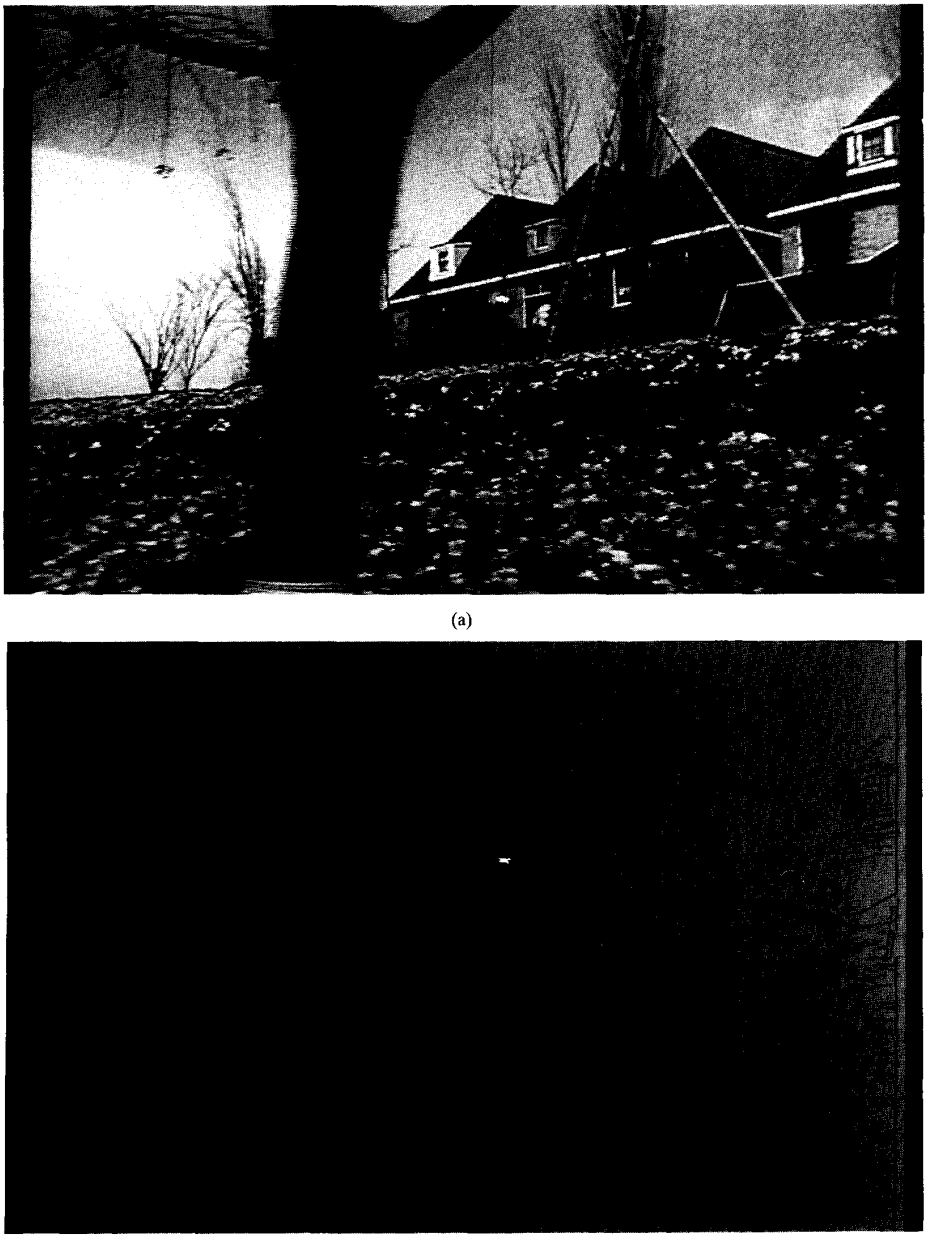
discuss the computational complexity of linear regression part of our algorithms and argue that the cost of regression can be neglected in comparison with edge detection for most practical purposes. In Sections V-C and V-D, we compare the computational complexity of our algorithms to that of full search intensity-based BMA for floating and fixed point arithmetic respectively.

Finally, it is worthwhile to mention that the edge-detection algorithm for our experimental results in Section IV uses fixed point arithmetic.

A. Edge Detection

The edge detection algorithm we have used in Section IV is described in detail in [24]–[26]. In this section, we will briefly review the algorithm so that we can characterize its computational complexity.

The edge-detection algorithm consists of two parts: smoothing, and taking directional derivatives. For smoothing purposes, we convolve our images with one-dimensional Gaussians along both x and y directions. By separating the two-



(a)

(b)

Fig. 11. (a) A typical frame of the "Flowers" sequence. (b) Edges of the image shown in (a).

dimensional convolution kernel into a pair of one-dimensional ones, we can save on the computations. The impulse response of the particular Gaussian filter we use has 8 taps and is of the form

$$h(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}. \tag{20}$$

To detect edges, we need to locate the maxima of the gradient or zeros of the second directional derivative along the gradient [24]–[26]. In doing so, we use a normalized symmetric exponential one-dimensional filter implemented in the image processing software package KHOROS [33]. Specifically, the first and second directional derivatives of the low pass version of $I(n_x, n_y)$ obtained by convolution of $I(n_x, n_y)$ with exponentially symmetric filter $f(n_x, n_y)$ is given by[24]–[27]

$$I_x(n_x, n_y) = I(n_x, n_y) * f_1(n_y) * f_2(n_y) * [f_2(n_x) - f_1(n_x)] \tag{21}$$

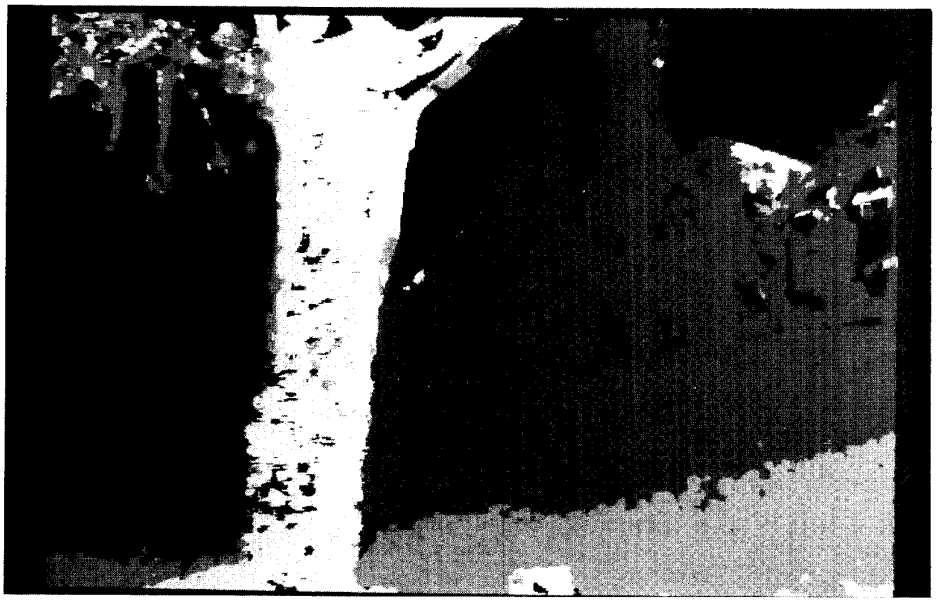


Fig. 12. Depth map of a typical frame of the "Flowers" sequence.

$$I_{xx}(n_x,n_y) = I(n_x,n_y) * f_1(n_y) * f_2(n_y) * [f_2(n_x) + f_1(n_x)] - 2I(n_x,n_y) * f_1(n_y) * f_2(n_y) \quad (22)$$

$$I_y(n_x,n_y) = I(n_x,n_y) * f_1(n_x) * f_2(n_x) * [f_2(n_y) - f_1(n_y)] \quad (23)$$

$$I_{yy}(n_x,n_y) = I(n_x,n_y) * f_1(n_x) * f_2(n_x) * [f_2(n_y) + f_1(n_y)] - 2I(n_x,n_y) * f_1(n_x) * f_2(n_x) \quad (24)$$

where $*$ stands for convolution, and f_1 and f_2 are defined as

$$f_1(n) = \begin{cases} 0 & n < 0 \\ a_0(1-a_0)^n & n \geq 0 \end{cases} \quad (25)$$

$$f_2(n) = \begin{cases} 0 & n > 0 \\ a_0(1-a_0)^{-n} & n \leq 0 \end{cases} \quad (26)$$

$f(n_x,n_y)$ is defined as

$$f(n_x,n_y) = \hat{f}(n_x)\hat{f}(n_y)$$

where

$$\hat{f}(n) = f_1(n) * f_2(n).$$

Since f_1 and f_2 are infinite impulse response (IIR) filters, they can be implemented in a recursive fashion via a first-order difference equation:

$$I_1(n_x,n_y) \equiv I(n_x,n_y) * f_1(n_x) = I_1(n_x-1,n_y) + a_0[I(n_x,n_y) - I_1(n_x-1,n_y)] \quad (27)$$

$$I_2(n_x,n_y) \equiv I(n_x,n_y) * f_2(n_x) = I_2(n_x+1,n_y) + a_0[I(n_x,n_y) - I_2(n_x+1,n_y)] \quad (28)$$

Based on the above, we can conclude that to find I_x and I_{xx},we need to perform four convolutions. Specifically, we need two convolutions to compute $\hat{I}(n_x,n_y) \equiv I(n_x,n_y) * f_1(n_y) * f_2(n_y)$, one convolution to form $\hat{I}(n_x,n_y) * f_2(n_x)$, and one convolution to form $\hat{I}(n_x,n_y) * f_2(n_y)$. Once these four convolutions are done, we need one addition[3] to compute $I_x(n_x,n_y)$ and two additions and a multiplication by 2 to compute $I_{xx}(n_x,n_y)$. Note that multiplication by 2 in fixed point arithmetic can be done at no almost no cost. Using (27) and (28), each convolution needs one multiplication and two additions. However, by choosing $a_0 = \frac{1}{2}$, the multiplications in fixed point computation can be done with shifting, and therefore their costs can be neglected.

Putting all of these together, we conclude that computing $I_x(n_x,n_y)$ and $I_{xx}(n_x,n_y)$ requires 5 multiplies and 11 adds per pixel for floating point implementation and 11 adds per pixel for fixed point arithmetic. Similar statments can be made for $I_y(n_x,n_y)$ and $I_{yy}(n_x,n_y)$. We summarize the operation count per pixel in Table I.

Once the gradient vector and the second derivative in the gradient direction for every point in the image is computed, the actual edge points are found by a series of thresholding operations [24]–[26]. The values of first and second hysteresis thresholds we have used for our various experiments are as follows: Ping-Pan $(1,10)$, Ping-Zoom, HDTV, and flowers $(1,4)$.

[3] We consider the complexity of a subtraction to be the same as an addition.

TABLE I
OPERATION COUNT FOR THE EDGE-DETECTION ALGORITHM

Task	Multiply count	Add count
8 Tap 1-D filtering along x	8	8
8 Tap 1-D filtering along y	8	8
IIR filtering for I_{xx} and I_x:		
Fixed point		11
Floating point	5	11
IIR filtering for I_{yy} and I_y:		
Fixed point		11
Floating point	5	11

In Sections V-C and V-D, we will use Table I to compare the operation count of our algorithm with traditional intensity-based techniques.

B. Linear Regression

In this section, we argue that the computational complexity associated with linear regression can be neglected in comparison with edge detection. In doing so, we are primarily concerned with the complexity of computing the matrices involved in the linear regression, rather than that of solving the linear system of equations. The computation associated with solving linear systems of equation can be justifiably ignored because 1) the matrices under consideration are either 3×3 or 4×4 and therefore their solution can be computed in a closed form; 2) unlike computing the matrices themselves, the cost associated with solving these systems is independent of the number of matched features; 3) on average, we solve only two systems per frame to remove outliers.

We begin with the zoom/pan/rotation algorithm of Section III-A, and then move on to the translation/rotation algorithm of Section III-B.

Zoom, Pan, and Rotation Algorithm For this algorithm, we solve a linear least-squares problem shown in (5) in order to determine c_i for $i = 1, 2, 3, 4$. Minimizing the error expression in (10) requires setting its derivatives with respect to c_i equal to zero, or equivalently solving the following system of equations:

$$A\vec{c} = \vec{d} \tag{29}$$

where $\vec{c}$ denotes the motion parameters, $\vec{d}$ is given by

$$\vec{d} \equiv \begin{bmatrix} \sum_{n=1}^{K} [X_{1n}X_{2n} + Y_{1n}Y_{2n}] \\ \sum_{n=1}^{K} [X_{2n}Y_{1n} - X_{1n}Y_{2n}] \\ \sum_{n=1}^{K} X_{2n} \\ \sum_{n=1}^{K} Y_{2n} \end{bmatrix}$$

matrix A is given by

$$A \equiv \begin{pmatrix} \sum_{i=1}^{K}(X_{1i}^2 + Y_{1i}^2) & 0 & \sum_{i=1}^{K} X_{1i} & \sum_{i=1}^{K} Y_{1i} \\ 0 & \sum_{i=1}^{K}(X_{1i}^2 + Y_{1i}^2) & \sum_{i=1}^{K} Y_{1i} & -\sum_{i=1}^{K} X_{1i} \\ \sum_{i=1}^{K} X_{1i} & \sum_{i=1}^{K} Y_{1i} & K & 0 \\ \sum_{i=1}^{K} Y_{1i} & -\sum_{i=1}^{K} X_{1i} & 0 & K \end{pmatrix}$$

and K denotes the number of matched features. To form matrix A and vector $\vec{d}$, there are 6 multiplications (mults) and 7 additions (adds) per feature match. In addition, the removal of outliers in step 5 of the algorithm, involves 1) applying the model shown in (5) which requires 4 mults and 4 adds per match; 2) computing two standard deviations which requires a total of 2 mults and 6 adds per match; 3) recomputing matrix A and vector $\vec{d}$ shown above which requires 6 mults and 7 adds per match. Thus, there are 13 operations per match for finding the parameters, and 29 operations per match for each iteration of removing the outliers. Assuming that our algorithm converges in two steps, i.e., the outliers are only removed once, there are a total of $29 + 13 = 42$operations per match. The average number of matches found for the three sequences considered in Section IV-A is one per 3500 pixels.[4] Putting all of these together, we get a total of .01 operations per pixel for the linear regression. This is negligible compared to the operations per pixels needed for computing the edge, regardless of whether fixed or floating point arithmetic is used.

Finally, if a global motion estimation algorithm uses intensity-based BMA, the number of "matches" used in linear regression is by definition one per 64 for 8×8 blocks. This results in 0.65 operations per pixel, which can be neglected for most practical purposes.

Rotation and Translation Algorithm Similar arguments can be applied to the rotation/translation algorithm of Section III-B. In that algorithm, the linear regression is carried out to minimize the error function of (16). For convenience, assume that the particular e_i chosen to be one is e_9. Minimizing the error expression in (16) requires setting its derivative with respect to c_i equal to zero, or equivalently solving the following system of equations:

$$Q\vec{e} = \vec{w} \tag{30}$$

where $\vec{e}$ denotes the vector containing the e_i parameters, $\vec{w}$ is an 8 dimensional vector given by

$$\vec{w} = \sum_{n=1}^{K} [\rho\gamma, \dot{\rho}\gamma, \gamma, \dot{\gamma}\rho, \dot{\gamma}\dot{\rho}, \dot{\gamma}, \rho, \dot{\rho}] \tag{31}$$

[4]As an example, of all the features in a 720×480 Ping-Zoom frame, only 93 matches were found; the corresponding numbers for the Ping-Pan and HDTV sequences are 113 and 81 respectively.

and Q is an 8×8 matrix given by

$$\sum_{n=1}^{K} \begin{pmatrix} \rho^2\gamma^2 & \rho\gamma^2\dot{\rho} & \rho\gamma^2 & \rho^2\gamma\dot{\gamma} & \rho\gamma\dot{\gamma}\dot{\rho} & \rho\gamma\dot{\gamma} & \rho^2\gamma & \rho\gamma\dot{\rho} \\ \rho\gamma^2\dot{\rho} & \dot{\rho}^2\gamma^2 & \dot{\rho}\gamma^2 & \rho\gamma\dot{\gamma}\dot{\rho} & \dot{\rho}^2\gamma\dot{\gamma} & \dot{\rho}\gamma\dot{\gamma} & \dot{\rho}\gamma\rho & \dot{\rho}^2\gamma \\ \rho\gamma^2 & \dot{\rho}\gamma^2 & \gamma^2 & \dot{\gamma}\rho\gamma & \dot{\gamma}\dot{\rho}\gamma & \dot{\gamma}\gamma & \rho\gamma & \dot{\rho}\gamma \\ \rho^2\gamma\dot{\gamma} & \rho\gamma\dot{\gamma}\dot{\rho} & \dot{\gamma}\rho\gamma & \dot{\gamma}^2\rho^2 & \dot{\gamma}^2\dot{\rho}\rho & \dot{\gamma}^2\rho & \dot{\gamma}\rho^2 & \dot{\gamma}\rho\dot{\rho} \\ \rho\gamma\dot{\gamma}\dot{\rho} & \dot{\rho}^2\gamma\dot{\gamma} & \dot{\gamma}\dot{\rho}\gamma & \dot{\gamma}^2\dot{\rho}\rho & \dot{\gamma}^2\dot{\rho}^2 & \dot{\gamma}^2\dot{\rho} & \dot{\gamma}\dot{\rho}\rho & \dot{\gamma}\dot{\rho}^2 \\ \rho\gamma\dot{\gamma} & \dot{\rho}\gamma\dot{\gamma} & \dot{\gamma}\gamma & \dot{\gamma}^2\rho & \dot{\gamma}^2\dot{\rho} & \dot{\gamma}^2 & \dot{\gamma}\rho & \dot{\gamma}\dot{\rho} \\ \rho^2\gamma & \dot{\rho}\gamma\rho & \rho\gamma & \dot{\gamma}\rho^2 & \dot{\gamma}\dot{\rho}\rho & \dot{\gamma}\rho & \rho^2 & \rho\dot{\rho} \\ \rho\gamma\dot{\rho} & \dot{\rho}^2\gamma & \dot{\rho}\gamma & \dot{\gamma}\rho\dot{\rho} & \dot{\gamma}\dot{\rho}^2 & \dot{\gamma}\dot{\rho} & \rho\dot{\rho} & \dot{\rho}^2 \end{pmatrix}$$

with $\rho \equiv X_{1n}, \dot{\rho} \equiv Y_{1n}, \gamma \equiv X_{2n}, \dot{\gamma} \equiv Y_{2n}$. Since Q is a symmetric matrix, 40 mults and 44 adds are needed in computing it for each match, resulting in a total of 84 operations per match. In addition, computing $\vec{w}$ requires 12 operations per match. In removing the outliers, we also need to re-compute matrix Q and vector $\vec{w}$. Before doing so however, we need to compute the following: 1) find the singular value decomposition of the E matrix which takes 18 operations; 2) use the singular vectors in (17) and (18) to compute the rotation parameters; this step requires 45 operation for (17) and 45 operations for (18); 3) use the e_i parameters in (19) to compute the translation parameters; this step requires 24 operations; 4) use the computed rotation and translation parameters in either (12) or (13) to compute z_1; this step takes 13 operations per feature match; 5) use (8) and (9) to compute (X_2, Y_2) coordinates for each match; this step requires 20 operations per match; 6) compute the standard deviation of the error using 8 operations; 7) re-compute matrix Q and vector $\vec{w}$ above using 96 operations per match.

Since steps 1), 2), and 3) are independent of the number of matched features, their contribution to the complexity of outlier removal can be ignored. Thus, the total number of operations per match needed for outlier removal is 136. Assuming the algorithm converges in two steps, i.e., we only remove the outliers once, there is a total of 232 operations per match. Furthermore, assuming there is approximately one match every 350 pixels, the number of operations per pixel becomes 0.66. This is again negligible compared to that needed for edge detection, regardless of whether fixed or floating point arithmetic is used.

Finally, if a global motion estimation algorithm uses intensity-based BMA, the number of "matches" in linear regression is by definition one per 64 for 8×8 blocks. This results in $\frac{232}{64} = 3.6$ operations per pixel, which can be neglected for most practical purposes.

C. Floating Point Implementation

In this section, we compare the computational complexity of traditional full search BMA and that of our edge-based technique for floating point implementation. For simplicity, we assume that the time/complexity required to match edges is negligible compared to the time/complexity required to find them. Furthermore, we assume that the amount of time needed for a floating point addition is roughly equal to that of a multiplication, and we will refer to either of those floating point operations as a flop.

Under these conditions, based on Table I, we conclude that the number of operations per pixel for our edge-based technique is 64. Similarly, the number of flops for full search BMA is $2S_A$ since there are $2S_A$ additions. S_A in the above expressions stands for search area used for finding the matching vector. We now compare computational complexity of our algorithms in Sections III-A and III-B.

Based on the above reasoning, we conclude that for global motion estimation algorithms based on the zoom/pan/rotation model of Section II-A, the edge-based algorithm results in lower computational complexity than full search intensity matching techniques, provided the search area is larger than 4×4. A similar statement can be made about global motion estimation based on rotation/translation model.

Since in most camera systems, the zoom and pan parameters vary slowly, it is enough to compute the parameters every P frames or so. As a result, we must use a fairly large search area. For instance in our experimental results in Section IV-A, we subsample the frames by as much as 10 before computing the zoom and pan parameters, and as a result the search area for the Ping-Pan and Ping-Zoom sequences in Section IV-A was chosen to be 41×41. For this particular value of search area, the floating point implementation of the full search version of our algorithm requires 53 times fewer operations than full search BMA.

For the rotation/translation algorithm, the search area used forluminance matching was 17×17 resulting in a saving factor of 17.

D. Fixed Point Implementation

Unlike floating point implementation, the cost of multiplication in fixed point implementation is higher than that of addition. In general, several factors need to be taken into account when comparing the costs associated with various algorithms. These factors include speed, chip area, and power and can usually be traded off with each other depending on the architecture used. For example, parallel processing architectures allow one to trade off chip area (or hardware complexity) with speed. Thus, comparing costs associated with various algorithms is not an easy task.

In spite of these, it is possible to choose simple ways of defining complexity. Specifically, in comparing eight-bit fixed point implementations, we assume that 1) cost of an eight-bit-by-eight-bit multiplication is 4.5 times that of eight-bit addition [34], and 2) the cost of an eight-bit addition is 16 times that of one-bit matchings [34]. The word cost in the previous sentence reflects either time or the hardware required to do a certain operation.

We now compare fixed point implementation of our edge-based algorithm with that of intensity-based full search BMA. The complexity of a full search block matching algorithm per pixel in units of cost of one-bit matching is $S_A(2 \times 16)$. This is because full search matching requires 2 adds, and the price of an eight-bit add is 16 times that of one-bit matching with a logic gate. Thus, the cost associated with full search block matching is proportional to $32S_A$.

The complexity of an edge-based full search matching algorithmis proportional to $(16 \times 4.5 \times 16)$ for the multiplications needed for edge detection, 38×16 for the additions needed

for edge detection and S_A for one-bit matching. Thus, the cost associated with edge-based full search matching algorithm is $1760 + S_A$.

Comparing the above two techniques, we can conclude that for both global motion estimation based on zoom/pan/rotation and rotation/translation, as long as the search area is larger than 8×8, the cost associated with edge-based matching is smaller than that of intensity-based full search matching. For example, for search area with the size of 41×41 used in the examples of Section IV-A, the intensity-based matching is 13 times more expensive than edge-based matching.

VI. Discussion

We have used the results on recovery of 3-D rigid body motion to develop a new algorithm for global motion estimation. Based on this approach, global parameters consisting of camera translation and rotation are first estimated from a video sequence, and then used in conjunction with the depth map of a scene to form the MC prediction for future frames. Our approach is different from existing global motion estimation techniques in that camera translation, as well as rotation is modeled. The performance of our approach is comparable to intensity-based local MC, at least in situations where the scene only consists of camera motion, such as in the flower sequence.

A salient feature in our algorithm is the use of edges in estimating camera parameters. There are several motivations behind edge-based global motion estimation: First, the availability of VLSI edge detection chips and convolution chips [32], [35] make the possibility of using edges in motion estimation quite realistic and potentially rewarding. Second, since a number of edge-based video coding schemes have been recently proposed [36], the additional use of edges for motion estimation is lucrative, particularly since it results in lower computational complexity as compared to traditional intensity-based techniques.

During the course of this paper, we have compared the complexity and performance of our edge-based global MC algorithms to two different classes of MC algorithms: 1) full search intensity-based BMA, with local MC; 2) full search intensity-based BMA with global MC. As far as performance goes, we have found that if the only motion in the sequence is due to the camera, then our edge-based technique does as well as 1). An example of this is shown in the flower sequence curves in Figs. 13 and 14, where the only variation in the video is due to camera translation. On the other hand, if there is additional object movement in the scene such as in Ping-Zoom and Ping-Pan, then our edge-based technique does as well as any global MC technique such as 2). An example of this is shown in the plot of Fig. 8 for camera pan and Fig. 9 for camera zoom. As for complexity, we showed in Section V that our edge-based global MC algorithms are less complex than both 1) and 2) for both floating point and fixed point implementations.

An important point to keep in mind is that the degree to which global MC algorithms work is highly dependent on the relative motion of camera compared to the objects in the scene. Applying any global model estimation technique, including ours, to a sequence that does not correspond to any camera motion at all will be unsatisfactory. In fact, this is true for all model-based techniques in image processing. An area for future research is the sensitivity of our proposed algorithms to various factors such as local motion, error in matching edges, and the values of camera parameters. For instance, our zoom determination algorithm is likely to encounter difficulties as

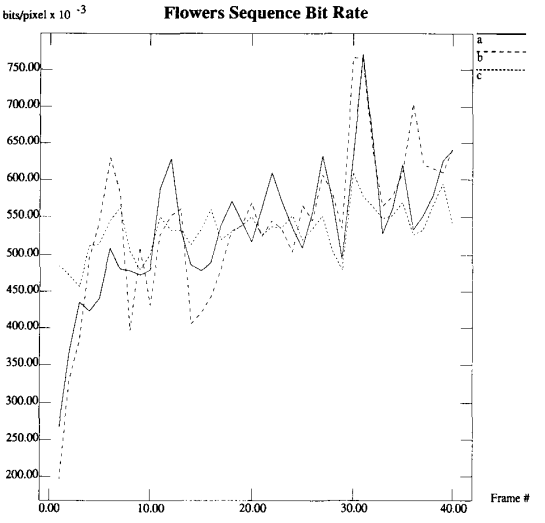


Fig. 13. Bit rate for flowers sequence: (a) Local motion estimation with intensity-based BMA with quantization step of the residue set to 1.6. (b) Edge-based global motion estimation based on rotation/translation model with quantization step of the residue set to 1.8. (c) Edge-based global motion estimation based on zoom/pan/rotation model with quantization step parameter of the residue set to 2.0.

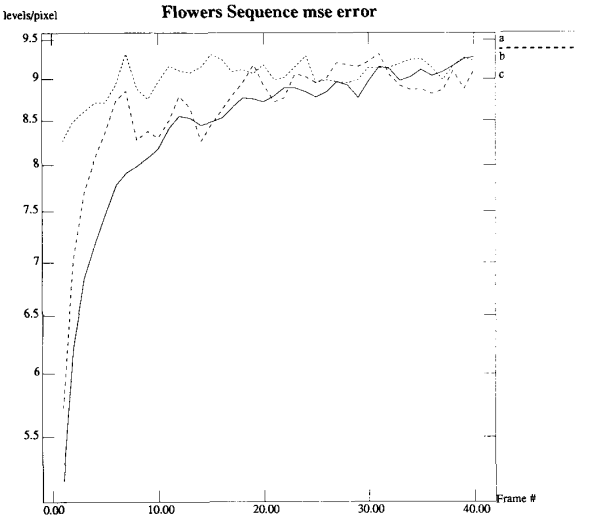


Fig. 14. Mse for flowers sequence: (a) Local motion estimation with intensity-based BMA with quantization step of the residue set to 1.6. (b) Edge-based global motion estimation based on rotation/translation model with quantization step of the residue set to 1.8. (c) Edge-based global motion estimation based on zoom/pan/rotation model with quantization step parameter of the residue set to 2.0.

the zoom parameter becomes large. Along the same lines, it is well known that computing the translation parameter becomes ill conditioned as the motion becomes small [8]. However, this is of little consequence in the context of video coding since it results in an E matrix with a large condition number which can be easily detected. One possible strategy to deal with such a situation with little or no loss in coding performance is to avoid motion compensation as long as the translation parameters are too small.

The work presented in this paper can be potentially applied in many practical situations with video sequences resulting from stationary scenes and moving cameras. One application of such a scenario might be 3-D video data bases in which 3-D stationary objects are captured on a video sequence by a camera moving around them [37]. In this situation, camera motion estimation techniques can not only be useful for efficient representation of the video sequence, but also for 3-D interpolation of the views not directly captured by the video camera. Other interesting application might include 3-D scene representation and reconstruction, and transmission.

REFERENCES

[1] J. Jurgen, "An abundance of video formats," *IEEE Spectrum*, vol. 29, pp. 26-28, Mar. 1992.

[2] D. Adolph and R. Buschmann, "1.15 Mbit/s coding of video signals including global motion compensation," *Signal processing: Image communication*, vol. 3, 1991, pp. 259-274.

[3] Y. T. Tse and R. L. Baker, "Global zoom/pan estimation and compensation for video compression," *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, Toronto, Canada, Apr. 1991, pp. 2725-2728.

[4] G. Keesman, "Motion estimation based on a motion model incorporating translation, rotation and zoom," *Signal processing*, vol. 4, 1988, pp. 31-34.

[5] M. Hoetter, "Differential estimation of the global motion parameters zoom and pan," *Signal processing*, vol. 16, Mar.1989, pp. 249-265.

[6] S. F. Wu and J. Kittler, "A differential method for simultaneous estimation of rotation, change of scale and translation," *Signal processing: Image communication* , vol. 2, 1990, pp. 69-80.

[7] G. Adiv, "Determining three-dimensional motion and structure from optical flow generated by several moving objects" *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 7, July 1985, pp. 384-401.

[8] R. Y. Tsai and T. S. Huang, "Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, Jan. 1984, pp. 13-26.

[9] T. S. Huang and A. N. Netravali, "3D motion estimation," in *Machine Vision for Three-Dimensional Scenes*. New York: Academic Press, 1990, pp. 195-218.

[10] J. Weng, N. Ahuja, and T. S. Huang, "Motion and structure from point correspondences with error estimation: Planar surfaces," *IEEE Trans. Signal Processing*, vol. 39, Dec. 1991, pp. 2691-2717.

[11] S. Y. Chen and W. H. Tsai, "A systematic approach to analytic determination of camera parameters by line features," *Image Recognition*, vol. 23, no. 8, pp. 859-877, 1990.

[12] P. R. Wolf, *Elements of Photogrammetry*. New York: McGraw Hill, 1983.

[13] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381-395, 1981.

[14] D. Vernon and M. Tistarelli, "Using camera motion to estimate range for robotic parts manipulation," *IEEE Trans. Robotics and Automation*, vol. 6, Oct. 1990, pp. 509-521.

[15] H. H. Chen and T. S. Huang, "Matching 3-D line segments with applications to multiple-object motion estimation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, October 1990, pp. 1002-1008.

[16] D. Marr and E. C. Hildreth, "Theory of edge detection," *Proc. Royal Society London B.*, vol. 207, pp. 187–217, 1980.

[17] E. C. Hildreth, "Computations underlying the measurement of visual motion," *Artificial Intelligence*, vol. 23, pp. 309-354, 1984.

[18] David W. Murray and Bernard F. Buxton, *Experiments in the machine interpretation of visual motion*. Cambridge, MA: MIT Press, 1990.

[19] G. L. Scott, *Local and Global Interpretation of Visual Motion*. London: Pitman and Morgan Kaufmann, 1987.

[20] L. A. Spacek, "Edge detection and motion detection," *Image and Vision Computing*, vol. 4, no. 1, pp. 43-56, 1986.

[21] C. Wang, H. Sun, S. Yada, and A. Rosenfeld, "Some experiments in relaxation image matching using corner features," *Pattern Recognition*, vol. 16, no. 2, pp. 167-182, 1983.

[22] D. F. Rogers and J. A. Adams, *Mathematical Elements for Computer Graphics*. New York: McGraw-Hill, 1976.

[23] G. Kummerfeldt, F. May and W. Wolf, "Coding television signals at 320 and 64 kbit/s," *Proc. SPIE on Image Coding*, vol. 594, 1985, pp. 119-128.

[24] J. Shen and S. Castan, "An optimal linear operator for edge detection," *Proc. Computer Vision and Pattern Recognition*, Miami, FL, 1986.

[25] J. Shen and S. Castan, "Edge detection based on multi-edge model," *Proc. SPIE*, Cannes, 1987.

[26] J. Shen and S. Castan, "Further results on DRF method for edge detection," *9th International Conference on Pattern Recognition*, Rome, 1988.

[27] J. Shen and S. Castan, "An optimal linear operator for step edge detection," *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 2, March 1992, pp. 112-133.

[28] P. J. Burt, J. R. Bergen, R. Hingorani, R. Kolczynski, W. A. Lee, A. Leung, J. Lubin, and H. Shvaytser, "Object tracking with a moving camera," *Proc. IEEE Workshop on Visual Motion*, 1989, pp. 2-12.

[29] "MPEG Video Committee Draft," December 1990.

[30] T. Komarek and P. Pirsch, "VLSI architectures for hierarchical block matching algorithms," in *IEEE Int. Symp. Circuits and Systems*, New Orleans, LA, May 1990, pp. 45-48.

[31] T. H. Y. Meng and A. C. Hung, "Parallel array architectures for motion estimation," *Proc. Application Specific Array Processors*, September 1991.

[32] Zoran 2-D Convolver ZL33771, Zoran 1-D convolver ZR33891, and ZR33288.

[33] John Rasure and Danielle Argiro, "Visual programming system and software development environment for data processing and visualization (Khoros)," user manual, University of New Mexico, 1992.

[34] J. Rabaey, Private communication.

[35] C. Lee, F. V. M. Catthoor, and H. J. De Man, "An efficient ASIC Architecture for al time edge detection," *IEEE Trans. CAS*, vol. 36, pp. 1350-1360, Oct. 1989.

[36] M. J. Biggar and A. G. Constantinides, "Segmented video coding," *Proc. International Conference on Acoustics, Speech and Signal Processing*, 1988, New York, NY, pp 1108–1111.

[37] A. Zakhor and F. Lari, "3D camera motion estimation with applications to video compression and 3D scene reconstruction." *Proc. SPIE Symp. on Electronic Imaging*, San Jose, CA, Feb. 1993.

Avideh Zakhor (S'87–M'87), for a photograph and biography please see page 508 of this issue.

F. Lari, photograph and biography were not available at the time of publication.