Segmentation and Factorization-Based Motion and Structure Estimation for Long Image Sequences

Christian Debrunner, *Member, IEEE*,
and Narendra Ahuja, *Fellow, IEEE*

Abstract—This paper presents a computer algorithm which, given a dense temporal sequence of intensity images of multiple moving objects, will separate the images into regions showing distinct objects, and, for those objects which are rotating, will calculate the three-dimensional structure and motion.

Index Terms—Multiple motion segmentation, motion sequence segmentation, image segmentation, image motion factorization, motion understanding, motion estimation, trajectory detection, point tracking.

1 Introduction

THIS paper describes an algorithm for three-dimensional structure and motion estimation for scenes containing multiple independently moving rigid objects. The two-dimensional motion in an input image sequence is represented by the image plane motion of *feature points*, whose positions in successive images form *trajectories*. The algorithm partitions the trajectories into sets, each corresponding to a distinct rigid object, and computes the *motion* and *structure* of each rigid object. The motion is assumed to be described by one of several *motion models*; the algorithm chooses the simplest motion model which describes the motion accurately. Currently, the algorithm uses two rigid motion models: a *translational model* (arbitrary translation), and a *rotational model* (locally constant angular velocity with arbitrary translation).

Our method consists of two steps:

1) finding the trajectories of feature points in the input image sequence and
2) determining the subset of trajectories, the structure, and the motion corresponding to each object.

The first step involves the detection of the path of the feature (a *feature path*) over several frames using the algorithm described in [1], and then linking the feature paths to produce the trajectories used in the second step. The method is robust to missing features or weak feature detector responses since the detected paths generally extend over many frames and overlap the paths detected in several previous and successive frames. The second step integrates the task of segmenting the images into distinct moving objects with the task of estimating the motion and structure for each object. Groups of trajectories are tested for uniformity of motion, against the hierarchy of motion models, and the simplest model that yields acceptable fit error is treated as the correct model. The model parameters are used to determine the final segmentation and estimates of motion and structure. The processing of image frames is sequential: As new image frames become available, the image segmentation and motion and structure estimates are updated to accommodate the new data. The rotational model parameters are computed using the motion factorization method described in [2], [3], [4]. This method uses a Singular Value Decomposition (SVD) and a few eigendecompositions to efficiently compute the motion and structure from trajectory data.

- *C. Debrunner is with Lockheed Martin Corporation, Denver, CO 80201. E-mail: chris.h.debrunner@lmco.com.*
- *N. Ahuja is with the Department of Electrical and Computer Engineering and Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801.*

Manuscript received revised 5 Dec. 1997. Recommended for acceptance by A. Singh. For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 106040.

2 Review of Previous Research

Using trajectories and assumptions of motion continuity can not only help reduce errors in the motion and structure estimation, but it can also simplify the process of finding the feature point correspondences. Methods which consider correspondences between only two frames such as those described in [5], [6], [7] cannot take advantage of any continuity of motion over a long sequence of frames. Sethi et al. have also noted this [8], [9], [10], [11], and propose a method which uses the *Greedy Exchange Algorithm* to rearrange the trajectories to maximize their *path coherence*. In the results presented, however, these algorithms where not tested on data sets such as ours with frame-to-frame feature displacements similar to the distance between features. It is unclear whether Sethi's algorithms would converge to the correct result in these circumstances since the initial matchings would be largely incorrect.

Most previous motion estimation methods do not address the problem of segmenting the image sequence into regions corresponding to rigid objects with different 3D motions. Many that do ([12], [13], [14]) segment the image at 2D motion discontinuities. Adiv [15] and Bergen et al. [16] improve on this by instead segmenting the image based on how well the image flow fits affine motion models. Unlike our motion model, these motion models assume planar surfaces, and, hence, often lead to oversegmentation. In addition, they are based on instantaneous motion estimates rather than trajectories and are, hence, much more susceptible to noise. In [15], Adiv also introduces the notion of using multiple models of motion, similar to our use of a hierarchy of motion models.

In [17], Tomasi and Kanade present a method which generalizes the motion estimation method described in [2], [3], [4] to allow arbitrary frame-to-frame rotations. However some assumptions about the magnitude or smoothness of motion are still necessary to obtain feature trajectories. Kanade points out [18] that with our assumption of constant rotation we are absorbing the trajectory noise primarily in the structure parameters, whereas their algorithm absorbs them in both the motion and structure parameters. Tomasi and Kanade's work was extended in [19] where Boult and Brown show how the factorization of the measurement matrix can be used to split the measurement matrix into parts consisting of independently moving rigid objects. Their method relies on a detailed analysis of the rank properties of the measurement matrix, and shows promising results on the 2D motion examples presented in [19].

3 Finding Trajectories

The algorithm described in this section produces a set of trajectories, where each trajectory is a list of the positions of one visible feature over some subset of the frames in the image sequence. Unlike previous methods [8], we do not first detect features in each image and then track them over many frames. Instead, we detect *feature paths*, which represent the presence of a feature in motion over several consecutive images, which prevents the fragmentation of trajectories due to occlusions and weak feature detector responses. These feature paths are then linked together to form the long trajectories needed for input to the structure and motion estimation algorithm. The first step in this process is to produce a *feature map* for each feature type and each frame. A feature map has small values in areas where the input image has little evidence of the feature of interest and large values in areas where the input image has strong evidence of the feature of interest. The

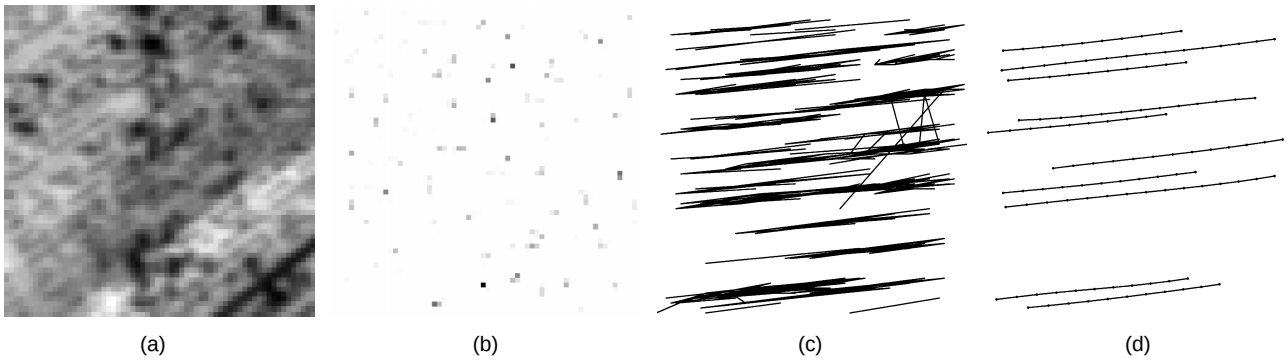


Fig. 1. This figure shows an example of the feature mapping algorithm on a sequence of 16, 64 × 64 pixel images. (a) shows the first frame of the sequence, and (b) shows the maxima feature map of (a). The feature map is shown inverted for clearer printing, i.e., high-intensity points are shown in black, and low-intensity points are shown in white.(c) shows feature paths detected by Blostein's algorithm from a maxima feature path sequence, and (d) shows the trajectories produced by the feature path linking algorithm from the feature paths shown in (c).

features maps used in this method are intensity minima and maxima, and the amount of evidence for these features is measured by fitting a quadratic surface to the image intensity surface. These feature maps are used as input to the feature path detection step which uses the method described by Blostein in [1]. This method efficiently checks all possible feature paths and uses sequential hypothesis testing to determine when enough evidence is present to declare a detection. The final step is to group the feature paths with other feature paths produced by the same image feature to produce trajectories. This is accomplished by partitioning the feature paths into disjoint sets, such that all feature paths within a set have high spatiotemporal overlap, whereas feature paths from different sets have low overlap. The trajectories are then computed by fitting trajectories to these feature path sets to minimize the least-square error. Fig. 1 shows an example of these steps for a small image region. Details of the algorithm are given in [3].

4 Structure and Motion Estimation

This section gives an overview of our motion models, with emphasis on the rotational motion model used for estimating the three-dimensional structure and motion of an object from the image plane trajectories. These algorithms are used in the motion segmentation and motion and structure estimation step described in Section 5. More detailed descriptions are presented in [2], [3], [4]. The two motion models used in our motion segmentation approach are the translational model and the rotational model, both of which assume orthographic projection. The translational model allows for arbitrary translational motion, and its parameters are computed using a linear least-squares fit to the trajectories. The rotational motion model allows a constant speed rotation around a fixed-direction axis and an arbitrary translation. The rotation parameters are extracted first, and then a linear least squares fit is used to compute the translation parameters. Aside from a small number of eigendecompositions and SVDs, the algorithm is closed form and requires no iterative optimization.

4.1 The Motion and Structure Models

We define the world coordinate system such that its **x**-**y** axes are the image plane **x**-**y** axes. The object on which the feature points are located rotates with a constant angular velocity of $\vec{\Omega} = \left[\omega_x\ \omega_y\ \omega_z\right]^{\mathrm{T}}$, and an arbitrary translation described by the sequence of locations $\vec{\mathbf{C}}_0(f) = \left[c_{0x}(f) c_{0y}(f) c_{0z}(f)\right]^{\mathrm{T}}$. We define an object-centered coordinate system whose **z** axis is parallel or antiparallel to $\vec{\Omega}$, and which translates but does not rotate with the object. We choose the direction of the object-centered coordinate system **z** axis, such that it does not point into the negative **z** half space of the world coordinate system, the negative **y** halfplane of the world **x**-**y** plane, or the world negative **x** axis. We also let $\vec{\mathbf{n}} = \left[n_x\ n_y\ n_z\right]^{\mathrm{T}}$ be the object coordinate system vector $\left[0\,0\,1\right]^{\mathrm{T}}$ expressed in world coordinates. Then, letting $\vec{\mathbf{O}}_p(f) = \left[O_{px}(f) O_{py}(f) O_{pz}(f)\right]^{\mathrm{T}}$ be the position of point p in object centered coordinates at the time frame f is recorded, the positions $\vec{\mathbf{O}}_p(f)$ change over time, due only to the rotation of the object and can be expressed in terms of $\vec{\mathbf{O}}_p(0)$ as $\vec{\mathbf{O}}_p(f) = \mathbf{G}^f \vec{\mathbf{O}}_p(0)$, where

$$\mathbf{G} = \begin{bmatrix} \cos\omega & -\sin\omega & 0 \\ \sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

and $\omega = \vec{\mathbf{n}} \cdot \vec{\Omega}$.

The rotation can be described by the angular speed ω and the unit vector $\vec{\mathbf{n}}$, but since $\vec{\mathbf{n}}$ contains only two degrees of freedom, we replace it with the two dimensional *foreshortening* vector $\vec{\mathbf{F}}$.

$$\vec{\mathbf{F}} = \begin{bmatrix} F_x \\ F_y \end{bmatrix} = \begin{cases} \pm \begin{bmatrix} \dfrac{n_x}{n_{xy}} \sqrt{1-n_z} \\ \dfrac{n_y}{n_{xy}} \sqrt{1-n_z} \end{bmatrix} & \text{if } n_{xy} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $n_{xy} = \sqrt{n_x^2 + n_y^2}$ and the ± is chosen such that $\vec{\mathbf{F}}$ lies in the first two quadrants ($F_y > 0$ or ($F_y = 0$ and $F_x \geq 0$)). $\vec{\mathbf{F}}$ is a vector in the image plane which is parallel to the image plane projection of the rotation axis, and whose length is $\sqrt{1-n_z}$ and, hence, is always between zero and one. This allows us to express the image plane coordinates of point p in frame f under the rotational motion model as

$$\vec{\mathbf{P}}_p(f) = \mathbf{H}\mathbf{G}^f \begin{bmatrix} O_{px}(0) \\ O_{py}(0) \\ O_{pz}(0)\sqrt{2 - F_x^2 - F_y^2} \end{bmatrix} + \vec{\mathbf{c}}_0(f) \tag{3}$$

where

$$\mathbf{H} = \begin{bmatrix} 1 - F_x^2 & -F_x F_y & F_x \\ -F_x F_y & 1 - F_y^2 & F_y \end{bmatrix} \tag{4}$$

and $\vec{\mathbf{c}}_0(f) = \left[c_{0x}(f)\, c_{0y}(f)\right]^{\mathrm{T}}$ (see [3] and [4] for more details of how (3) is derived). Under the translational motion model **G** is the identity matrix, $\vec{\mathbf{F}} = 0$, and (3) reduces to

$$\vec{\mathbf{P}}_p(f) = \begin{bmatrix} O_{px}(0) \\ O_{py}(0) \end{bmatrix} + \vec{\mathbf{c}}_0(f). \tag{5}$$

Equation (3) is the starting point for the next section, which briefly describes our algorithm for estimating the motion and structure parameters for the rotational model.

4.2 The Rotational Motion Estimation Algorithm

Our algorithm uses algorithms used in state-space singular value decomposition based sinusoidal retrieval methods, such as the Direct-Data Approximation method [20]. These methods extract multidimensional sinusoidal signals from noisy data streams. In our problem, the multidimensional sinusoidal signal is the rotational component of the motion, and the frequency of the recovered sinusoid is ω, while the amplitudes and phases determine the structure and orientation of the object. Since the translational component of the motion $\vec{\mathbf{c}}_0(f)$ in (3) is not a sinusoid, we start by taking the differences in point positions $\vec{\mathbf{y}}_{pq}(f) = \vec{\mathbf{P}}_q(f) - \vec{\mathbf{P}}_p(f)$ to eliminate this term. The resulting purely rotational motion can be expressed as a system of state equations

$$\vec{\mathbf{x}}_{pq}(f+1) = \mathbf{G}\vec{\mathbf{x}}_{pq}(f) = \mathbf{G}^{f+1}\vec{\mathbf{x}}_{pq}(0)$$

$$\vec{\mathbf{y}}_{pq}(f) = \mathbf{H}\vec{\mathbf{x}}_{pq}(f) \tag{6}$$

where

$$\vec{\mathbf{x}}_{pq}(f) = \begin{bmatrix} O_{qx}(f) - O_{px}(f) \\ O_{qy}(f) - O_{py}(f) \\ \left(O_{qz}(f) - O_{pz}(f)\right)\sqrt{2 - F_x^2 - F_y^2} \end{bmatrix}. \tag{7}$$

Now, given n points over m frames at positions $\vec{\mathbf{P}}_p(f)$ for $p = 0, \ldots, n-1$ and $f = 0, \ldots, m-1$, form a matrix **D** from the input data as follows:

$$\mathbf{D} = \left[\Upsilon_0\, \Upsilon_1 \cdots \Upsilon_{m-k-1}\right] \tag{8}$$

where

$$\Upsilon_f = \begin{bmatrix} \vec{\mathbf{y}}_{01}(f) & \vec{\mathbf{y}}_{02}(f) & \cdots & \vec{\mathbf{y}}_{0(n-1)}(f) \\ \vec{\mathbf{y}}_{01}(f+1) & \vec{\mathbf{y}}_{02}(f+1) & \cdots & \vec{\mathbf{y}}_{0(n-1)}(f+1) \\ \vdots & \vdots & \ddots & \vdots \\ \vec{\mathbf{y}}_{01}(f+k) & \vec{\mathbf{y}}_{02}(f+k) & \cdots & \vec{\mathbf{y}}_{0(n-1)}(f+k) \end{bmatrix} \tag{9}$$

and k is chosen to make the matrix **D** as nearly square as possible. We also define **C** as follows:

$$\mathbf{C} = \left[\chi_0\ \chi_1 \cdots \chi_{m-k-1}\right] = \left[\chi_0\ \mathbf{G}\chi_0\ \mathbf{G}^2\chi_0 \cdots \mathbf{G}^{m-k-1}\chi_0\right], \tag{10}$$

where

$$\chi_f = \left[\vec{\mathbf{x}}_{01}(f)\ \vec{\mathbf{x}}_{02}(f) \cdots \vec{\mathbf{x}}_{0(n-1)}(f)\right]. \tag{11}$$

Now, letting

$$\mathbf{O} = \begin{bmatrix} \mathbf{H} \\ \mathbf{HG} \\ \vdots \\ \mathbf{HG}^k \end{bmatrix}, \tag{12}$$

the relationship between **D**, **O**, and **C** can be expressed as

$$\mathbf{D} = \mathbf{OC}. \tag{13}$$

Note that (13) is a factorization of the trajectory data **D** into a matrix **O** containing the motion information and a matrix **C** containing the structure information. Taking the singular value decomposition of **D** and retaining only the first three left singular vectors and the first three right singular vectors produces a factorization of **D** into a $2k \times 3$ matrix **O** and a $3 \times (n-1)(m-k)$ matrix **C**. Given the two matrices **O** and **C**, we can extract **G** and **H** from **O** and χ_0 from **C**. We can then find the unknown motion parameters F_x and F_y from **H** and ω from **G**, and the structure parameters $\vec{\mathbf{O}}_p(f)$ from χ_0. But the matrices (**G**, χ_0, **H**) corresponding to this factorization will not in general be of the form given in (1), (11), (7), and (4). Thus, to find the motion and structure parameters, the algorithm finds a linear transformation of the factorization **OC** into a factorization of the proper form. Our method of finding this transformation is detailed in [2], [3], [4]. Having computed the rotational parameters $\vec{\mathbf{F}}$ and ω, the final step of the algorithm is to compute the translational component $\mathbf{c}_0(f)$ using a linear least squares fit to the trajectory data.

There are, however, some combinations of structure and motion parameters which will cause this algorithm to fail because of rank deficiencies in the intermediate matrices. We refer to algorithm for the general case as the *full rank* algorithm, but we also use a variation called the *rank two* algorithm (described in [3]) to handle the special cases. Assuming that enough input trajectory information is available (for general structure: at least three frames and at least four points, at least four frames and at least three points, or at least five frames and at least two points; for points lying in a plane, at least four frames are always required), the degeneracies can occur for four reasons:

1) there is no rotation ($\omega = 0$),
2) all the feature points producing the trajectories lie on a line parallel to the rotation axis,
3) the rotation axis lies along the world **z** axis ($\vec{\mathbf{F}} = 0$),
4) all the feature points producing the trajectories lie in a plane perpendicular to the rotation axis.

In the first two cases, there is no recoverable rotational motion, and, therefore, the relative depth components of the structure cannot be recovered and the translational motion algorithm extracts all the recoverable information. In the third case, the relative depth component of the structure is also not recoverable, but the rank two algorithm can recover all the remaining structure, rotation, and translation parameters, and, in case (4), it can recover all the parameters. See [3] for a detailed analysis of the rank degeneracies and for descriptions of the modified algorithms. Also see [3] for descriptions of how the algorithm is modified to allow for noisy inputs and for trajectories of varying starting and ending frames.

This section has introduced our structure and motion models and briefly described the structure and motion estimation algorithm used for the rotational motion model. The latter algorithm provides an efficient way to incorporate information from many frames and points into the calculation of the motion parameters, allowing accurate estimates of motion and structure parameters. When there is no rotational motion, we use a linear least squares fit to a translational motion model to determine the motion. The following section describes how this translational motion estimation algorithm and the rotational motion estimation algorithm are used to segment the input trajectories into groups corresponding to the different rigid objects in the scene and to determine the structure and motion parameters for each of these objects.

5 Image Sequence Segmentation

This section describes how our algorithm partitions the feature points detected in an image sequence into groups corresponding

to the multiple rigid objects visible in the sequence. The feature points are initially grouped according to spatial adjacency and motion compatibility between neighbors, but the groups of feature points, called *regions*, are also required to fit one of the *motion models*: the translational model and the rotational model. Although there are rigid three-dimensional motions which cannot be modeled by the translational or rotational models (e.g., a rotation with precession), for convenience all motions which do not fit our two rigid motion models and do not contain any local motion discontinuities are classified as *unmodeled motions*. The algorithm always tries to use the simplest region motion model: It uses the translational motion model as long as it accurately accounts for a region's feature point trajectories, and then it tries the rotational model. If the rotational model also does not accurately account for a region's trajectories, and there is no motion discontinuity, it classifies the motion as unmodeled. The input data are processed frame by frame, and initially, all feature points in the first frame are grouped into a single region with a region motion model of translational motion. Once the point positions from the second frame are obtained, some preliminary motion information is available, and the algorithm can try to split the single initial region into smaller ones.

The processing then continues in a uniform fashion: The new point positions in each new frame are added to the trajectories of the existing regions, and then the regions are processed to make them compatible with the new data. The processing of the regions is broken into four steps:

1) If the old region motion model gives too large a fit error, either split the region into smaller ones, find a new region motion model which works better, or apply the previous motion model over a subsequence of the image sequence,
2) Add any newly visible points or ungrouped feature points to a region with which they are compatible,
3) Merge adjacent regions with compatible motions,
4) Remove outliers from the regions.

Compatibility among feature points is checked by fitting one of the motion models described in Section 4, and a region's feature points are considered incompatible if the fit error returned by the appropriate motion estimation algorithm is above a threshold. We assume that the trajectory detection algorithm described in Section 3 can produce trajectories accurate to the nearest pixel, and, therefore, we use a threshold (which we call the *error threshold*) of one-half of a pixel per visible trajectory point per frame.

When a region is split in the first step, all the feature points contained in the original region should be assigned to one of the new regions, but, generally, some of the feature point trajectories do not fit the motion parameters of any of the new regions well. In this step and when removing outliers in the fourth step, feature points result which cannot be associated with any region. These are referred to as the *leftover* feature points, and they are stored on a list called the *leftovers list*.

5.1 Updating Regions

When processing a new frame, the new point positions are added to any previously existing trajectories, or if a new trajectory starts in this frame, it is added to the leftovers list. Then, each existing region is examined in turn. In processing the previous frame, the algorithm found a motion model and a set of parameters for that model which accounted for the motion described by the region's trajectories (unless the motion is unmodeled). As a region is processed in successive frames, it will not always have the same motion model. For example, if the region contains trajectories from an object rotating slowly about the world $\mathbf{y}$ axis, with only a few frames available the trajectories might fit the translational motion model very closely. Then as more frames become available, the trajectories will no longer fit the translational motion model, but they should fit the rotational motion model. In processing a region in a new frame, if the region's previous motion model is the translational or rotational model, the motion model and parameters used in the previous frame are applied to the new trajectory point positions. If the resulting error is then below the threshold, the next region is processed. Otherwise, the region must be split, a better fitting motion model must be found, the model must be applied over a subsequence of the frames, or the region's motion must be classified as unmodeled.

Regions are split along local translational motion discontinuities. If no such boundary is found, the algorithm tries to fit the trajectories to the other motion models. If the previous model was the rotational model, we assume the fit error was too large because the rotational motion is not constant, and apply the rotational model to a slightly shorter subsequence of the frames. If the subsequence produces an accurate fit, the rotational model is used, and, otherwise, the motion is classified as unmodeled. This process of testing the old model and possibly splitting the region or changing the motion model is repeated for every region, and then the algorithm proceeds to the next step, grouping the leftover feature points.

5.2 Grouping Leftover Feature Points

After updating the region motion models and the model parameters of all the regions, all feature points in the leftovers list are repeatedly considered for addition to any neighboring regions in the current frame until no more can be added to any of the regions. The leftovers list may contain feature points which have just become visible, feature points which were left over after region splitting, or feature points which were removed in the outlier removal step.

5.3 Region Merging

After adding leftover feature points to regions wherever possible, any adjacent regions with compatible motion parameters are merged. For every region, the algorithm checks whether it can be merged with any of its neighbors. Given a pair of regions, the higher-order of the two regions' motion models is used to test if the regions can be merged.

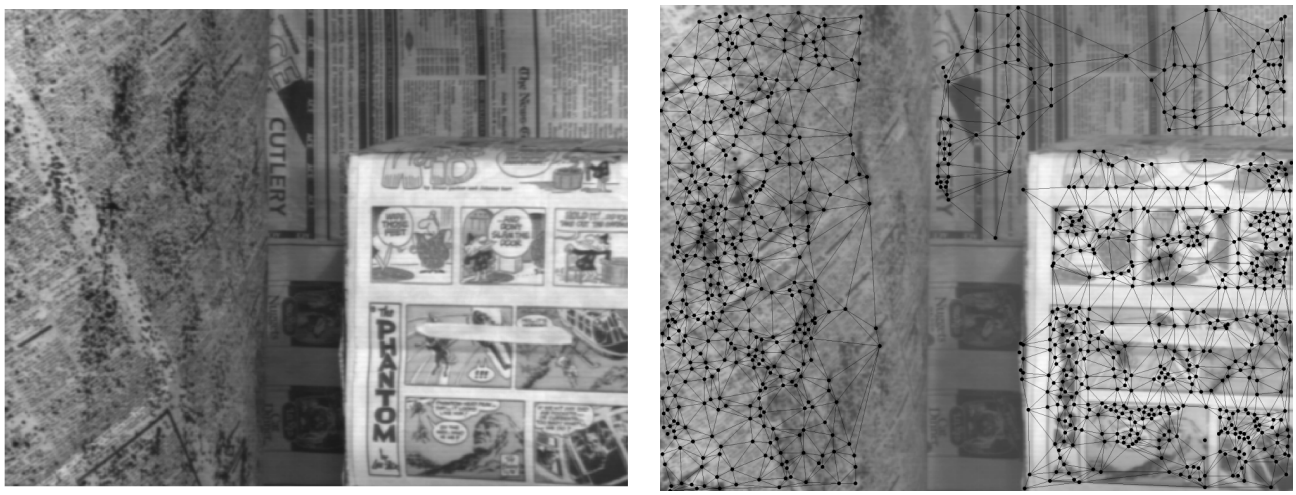
5.4 Outlier Removal

The final step in the image sequence segmentation loop is to examine each region and to remove any feature points which fit the region's motion parameters very poorly. Outliers arise due to incorrect trajectories produced by the trajectory detection step described in Section 3. Incorrect trajectories can arise from false feature points (such as specular reflection or the intersection of two independently moving occluding boundaries) which do not correspond to fixed points on a rigid object, or simply from incorrectly tracked feature points. Over a short trajectory these feature points might fit the motion parameters of a nearby region, but, as more frames become available, the trajectory will eventually diverge from the predicted motion. These outlier feature points are left in the leftovers list.

Thus, we have now described the complete process, starting with a sequence of gray level images, calculating the trajectories of feature points in the sequence, grouping these trajectories into different regions corresponding to different rigid objects, and calculating the structure and motion parameters of each region.

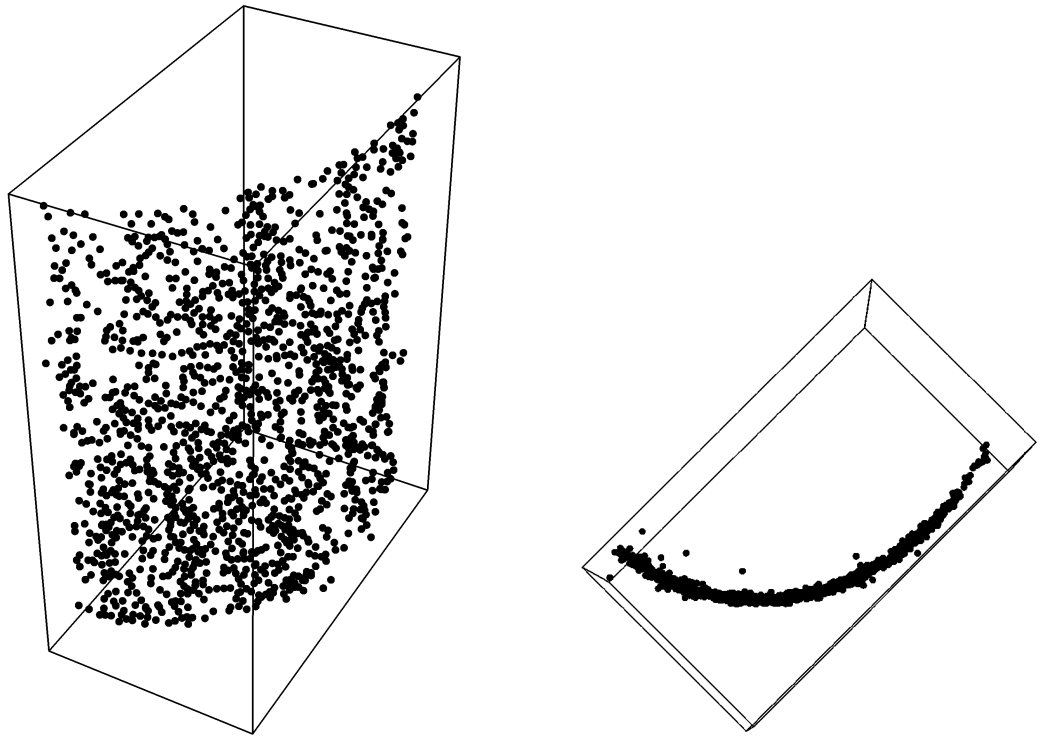
6 Experiments

In this section, we present the results of experiments which show the capabilities of the approach, and also demonstrate some of the inherent limitations of such approaches. The trajectory finding,



(a) (b)

Fig. 2. (a) The first image of the cylinder sequence. (b) The image sequence segmentation found for the cylinder sequence (the segmentation is superimposed on the last frame of the sequence).



(a) (b)

Fig. 3. (a) The view of the three-dimensional point positions calculated by our structure and motion estimation algorithm from point trajectories derived from cylinder image sequence. (b) A top view of (a) showing the shape of the surface formed by the points.

structure and motion estimation, and image sequence segmentation algorithms were tested on a real image sequence of 50 frames. This sequence, called the cylinder sequence, consists of images of a cylinder rotating at a rate of about one degree per frame around a nearly vertical axis and translating at about 0.14 pixel per frame to the right, a box moving right at 0.31 pixel per frame, and a background moving right at 0.14 pixel per frame. All the objects were covered in newspaper and splattered with paint to provide a large number of features to track. Fig. 2a shows the first image of the sequence.

The trajectory finding algorithm found 2,598 trajectories for the cylinder sequence. These trajectories then served as input to the image sequence segmentation algorithm described in Section 5, which partitioned the trajectories into groups corresponding to different rigid objects and estimated the motion and structure parameters using the algorithm described in Section 4.

The image sequence segmentation for the cylinder sequence is shown in Fig. 2b. For this image sequence, the algorithm separated out the three image regions: the cylinder, the box, and the background. Since the latter two are only moving with translational motion, their three-dimensional structure cannot be recovered by the motion and structure estimation algorithm. Instead, they are identified as translating regions, and their collective velocity is estimated. The cylinder, however, is rotating, and, consequently,

TABLE 1
COMPARISON OF THE PARAMETERS ESTIMATED BY THE ALGORITHM AND THE TRUE PARAMETERS FOR THE CYLINDER IMAGE SEQUENCE EXPERIMENT

Parameters	Estimated	Actual
ω	–0.022	–0.017
$\vec{\mathbf{F}}$	(0, 0.94)	(0, 0.90)
$\vec{\mathbf{v}}$	(0.29, –0.19)	(0.14, 0)

its structure can be recovered from the image sequence. To illustrate the structure recovered by the algorithm, Fig. 3 shows two projections of the three-dimensional point positions calculated for the 1,456 trajectories grouped together to form the cylinder. Fig. 3b shows an end-on view of the points, showing that the points lie very nearly on a cylindrical surface. In addition, Table 1 shows the estimated and the actual motion parameters. To understand the error in the ω estimate in Table 1, consider the motion of the cylinder in this experiment. The cylinder is rotating around an axis nearly parallel to the image plane and as pointed out in [21], a rotation about an axis parallel to the image plane is inherently difficult to distinguish from translation parallel to the image plane and perpendicular to the rotation axis (this would also explain the error in $\vec{\mathbf{v}}$ in Table 1). Note that, in spite of the error in the estimated motion parameters for the cylinder, the predicted trajectory point positions differ from the actual positions by an average of less than the error threshold of 0.5 pixel. Thus, we have chosen a particularly difficult and ambiguous motion in this example.

These experiments have shown that our algorithm can segment feature trajectories in real image sequences into groups corresponding to different rigid objects, and that it can accurately estimate the structure and motion of the objects.

7 CONCLUSIONS

This paper has presented a method for finding the structure and motion of multiple moving objects from a long sequence of intensity images by dynamically invoking a motion and structure estimation algorithm on different subsets of feature points under different motion models. The main features of our method are:

1) Motion and structure estimation and segmentation processes are integrated,
2) The motion and structure estimation algorithm factors the trajectory data into separate motion and structure matrices, and
3) The feature tracking algorithm accommodates briefly occluded points and weak or missing feature detector responses.

In the experiments, our method segmented the image sequence into separate rigid objects with few errors. The motions of the objects in the sequence represented very difficult to estimate motions (the cylinder). The difficulty is reflected in the accuracy of the parameter estimates, but, in all cases, the estimated parameters predicted the trajectories to within an average error of less than 0.5 pixels.

ACKNOWLEDGMENTS

The authors would like to thank Steve Blostein for providing the code for the feature path detection step and Lockheed Martin Corporation for providing the text processing facilities on which this paper was prepared.

This paper was supported by DARPA and the U.S. National Science Foundation under grant IRI-89-02728 and the State of Illinois Department of Commerce and Community Affairs under grant 90-103.

REFERENCES

[1] S.D. Blostein, "A Sequential Hypothesis Testing Approach to Detecting Small, Moving Objects in Image Sequences," *IEEE Trans. Signal Processing*, pp. 1,611-1,629, July 1991.
[2] C. Debrunner and N. Ahuja, "A Direct Data Approximation Based Motion Estimation Algorithm," *Proc. 10th Int'l Conf. Pattern Recognition*, pp. 384-389, Atlantic City, N.J., June 1990.
[3] C. Debrunner, "Structure and Motion From Long Image Sequences," PhD dissertation, Univ. of Illinois at Urbana-Champaign, Aug. 1990.
[4] C. Debrunner and N. Ahuja, "Estimation of Structure and Motion From Extended Point Trajectories," unpublished manuscript.
[5] S. Ullman, "The Interpretation of Structure From Motion," *Proc. Royal Soc. London*, vol. 203, pp. 405-426, 1979.
[6] S. Ranade and A. Rosenfeld, "Point Pattern Matching by Relaxation," *Pattern Recognition*, vol. 12, no. 4, pp. 269-275, 1980.
[7] J. Fang and T. S. Huang, "Some Experiments on Estimating the 3-D Motion Parameters of a Rigid Body From Two Consecutive Image Frames," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no. 5, pp. 545-554, Sept. 1984.
[8] I.K. Sethi and R. Jain, "Finding Trajectories of Feature Points in a Monocular Image Sequence," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 1, pp. 56-73, Jan. 1987.
[9] V. Salari and I.K. Sethi, "Correspondence in Presence of Occlusion," *Proc. IEEE CS Workshop Computer Vision*, pp. 327-330, Miami Beach, Fla., Nov. 1987.
[10] I.K. Sethi, V. Salari, and S. Vemuri, "Feature Point Matching Using Temporal Smoothness in Velocity," *Pattern Recognition Theory and Applications*, P.A. Devijver and J. Kittler, eds., vol. F30, NATO ASI Series, pp. 119-131. Berlin and Heidelberg: Springer-Verlag, 1987.
[11] I.K. Sethi, V. Salari, and S. Vemuri, "Image Sequence Segmentation Using Motion Coherence," *Proc. First Int'l Conf. Computer Vision*, pp. 667-671, London, June 1987.
[12] E.C. Hildreth, "The Measurement of Visual Motion," Cambridge, Mass.: M.I.T. Press, 1984.
[13] W.B. Thompson, K.M. Mutch, and V.A. Berzins, "Dynamic Occlusion Analysis in Optical Flow Fields," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 7, pp. 374-383, July 1985.
[14] B.G. Schunk, "Image Flow Segmentation and Estimation by Constraint Line Clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, pp. 1,010-1,027, Oct. 1989.
[15] G. Adiv, "Determining Three-Dimensional Motion and Structure From Optical Flow Generated by Several Moving Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 7, pp. 384-401, July 1985.
[16] J.R. Bergen, P.J. Burt, R. Hingorani, and S. Peleg, "Multiple Component Image Motion: Motion Estimation," *Proc. Int'l Conf. Computer Vision*, Osaka, Japan, Dec. 1990.
[17] C. Tomasi and T. Kanade, "Factoring Image Sequences Into Shape and Motion," *Proc. IEEE Motion Workshop*, pp. 21-28, Princeton, N.J., Oct. 1991.
[18] T. Kanade, personal communication, Oct. 1991.
[19] T.E. Boult and L.G. Brown, "Factorization-Based Segmentation of Motions," *Proc. IEEE Motion Workshop*, pp. 21-28, Princeton, N.J., Oct. 1991.
[20] S.Y. Kung, K.S. Arun, and D.V.B. Rao, "State-Space and Singular-Value Decomposition-Based Approximation Methods for the Harmonic Retrieval Problem," *J. Optical Soc. Am.*, vol. 73, pp. 1,799-1,811, Dec. 1983.
[21] J. Weng, T. S. Huang, and N. Ahuja, "Motion and Structure From Two Perspective Views: Algorithms, Error Analysis, and Error Estimation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, pp. 451-476, May 1989.