A Voting-Based Computational Framework for Visual Motion Analysis and Interpretation

Mircea Nicolescu, *Member*, *IEEE*, and Gérard Medioni, *Fellow*, *IEEE*

Abstract—Most approaches for motion analysis and interpretation rely on restrictive parametric models and involve iterative methods which depend heavily on initial conditions and are subject to instability. Further difficulties are encountered in image regions where motion is not smooth—typically around motion boundaries. This work addresses the problem of visual motion analysis and interpretation by formulating it as an inference of motion layers from a noisy and possibly sparse point set in a 4D space. The core of the method is based on a *layered 4D representation* of data and a *voting scheme* for affinity propagation. The inherent problem caused by the ambiguity of 2D to 3D interpretation is usually handled by adding additional constraints, such as rigidity. However, enforcing such a global constraint has been problematic in the combined presence of noise and multiple independent motions. By decoupling the processes of matching, outlier rejection, segmentation, and interpretation, we extract accurate motion layers based on the smoothness of image motion, then locally enforce rigidity for each layer in order to infer its 3D structure and motion. The proposed framework is noniterative and consistently handles both smooth moving regions and motion discontinuities without using any prior knowledge of the motion model.

Index Terms—Motion, image models, scene analysis.

1 Introduction

This work addresses a difficult and fundamental problem in computer vision, the analysis and interpretation of visual motion. An illustration is given in Fig. 1—given two image frames that contain general motion, the goal is twofold:

- to analyze the image changes in order to establish correspondences between pixels across frames, as a dense velocity field, and to group them into motion regions separated by motion boundaries, and
- to interpret these changes in order to recover the scene 3D structure and 3D motion.

This formulation that divides the visual process into motion analysis and interpretation is inspired from various perceptual studies [1], [2], which show that establishing correspondences is a low-level process that takes place prior to interpretation, where matches are established between elementary tokens, based on built-in affinity measures. The tokens involved in matching are not complex structures, but, rather, primitive image elements, such as points, fragments of edges, or blobs. In this study, we only focus on analysis from point tokens.

1.1 Motion Analysis

The examination of some simple configurations in motion indicates that the visual system incorporates a certain affinity measure between tokens, which can be roughly considered as a measure of similarity. This affinity is involved in both processes of matching and region grouping. Indeed, establishing a correspondence between two tokens implies that their mutual affinity (or preference to each other) is greater than the affinity to other tokens. In grouping, to determine that a token belongs to a certain region is equivalent to establishing that it has a stronger affinity to the tokens in that region than to tokens in other regions.

In order to solve the problem of motion analysis, a successful computational framework must define a way to express the affinities between tokens while also taking into account and handling the difficulties inherent to this process, such as the aperture problem and the presence of regions lacking texture. To this purpose, additional constraints need to be introduced. More specifically, any such constraint must satisfy two requirements: 1) define a practical measure of affinities between tokens so that they can be matched and grouped successfully and 2) allow the computation of motion (pixel velocities) where local measurements could not provide a complete solution—this being the case of one velocity component missing due to the aperture problem or the case of unreliable motion due to lack of texture. In this context, the constraints are needed in order to generate a dense output from a sparse and/or noisy input.

Several types of constraints have been commonly considered in the literature:

- constant velocity over an area of the image (valid for pure translation),
- constant velocity over small time intervals,
- velocity consistent with 2D rigid motion (valid for rotation and translation of objects in the image plane),
- velocity consistent with 3D rigid motion, and
- smooth velocity within image areas that represent distinct objects.

- *M. Nicolescu is with the Department of Computer Science, University of Nevada, 1664 N. Virginia St., Reno, NV 89557. E-mail: mircea@cs.unr.edu.*
- *G. Medioni is with the Integrated Media Systems Center and with the Institute for Robotics and Intelligent Systems, University of Southern California, Powell Hall, 3737 Watt Way, Los Angeles, CA 90089-0273. E-mail: medioni@usc.edu.*

Manuscript received 4 Aug. 2004; revised 20 Sept. 2004; accepted 21 Sept. 2004; published online 11 Mar. 2005.
Recommended for acceptance by G. Sapiro.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0401-0804.

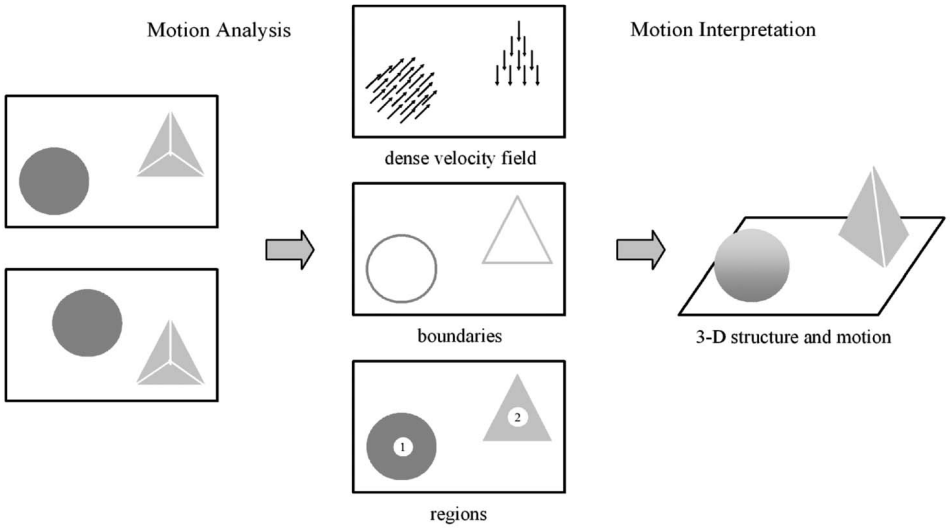


Fig. 1. Motion analysis and interpretation.

Methods that are based on assumptions of constant velocity or rigid motion are not sufficient for analyzing the two-dimensional motion that arises from the projection of arbitrary three-dimensional surfaces undergoing general motion in space. Therefore, it is the last constraint—*smoothness of image motion*—that is used in this work.

Our computational framework addresses the problem of visual motion analysis from a perceptual organization perspective, where saliency is used as an affinity measure. An essential property is that corresponding pixels in the two images form coherent perceptual structures in the 4D space of image coordinates and pixel velocities, while erroneous matches generate outlier tokens. More precisely, each potential match is seen as a token characterized by four attributes—the image coordinates (x, y) in the first image and the velocity with the components (v_x, v_y). Tokens are encapsulated as (x, y, v_x, v_y) points in the 4D space, this being a natural way of expressing the spatial separation of tokens according to *both* velocities and image coordinates. In general, for each pixel (x, y), there can be several candidate velocities, so each 4D point (x, y, v_x, v_y) represents a potential match.

Within this representation, smoothness of motion is embedded in the concept of surface saliency exhibited by the data. A computational methodology that successfully enforces the smoothness constraint in a consistent and unified manner, while preserving discontinuities is tensor voting [3]. This technique also benefits from the fact that it is noniterative and it does not depend on critical thresholds. By letting the tokens communicate their mutual affinity through voting, noisy matches are eliminated as they receive little support and distinct moving regions are extracted as smooth *salient surface layers*, in the 4D space.

1.2 Handling Uncertainty along Motion Boundaries

However, further difficulty in motion analysis is caused by the presence of motion boundaries themselves. The very source of information used for segmentation—pixel velocities—is mostly unreliable exactly at the motion boundaries, where the segmentation takes place. The example in Fig. 2, showing a truck moving from left to right over a static background, is used to illustrate the problem. From Area A that appears in the first image, only half is visible in the second image, the other half being occluded by the moving region. At the opposite side, Area B is still visible in the second image, but is now split into two regions, with new, unoccluded pixels in between. Even where no occlusion takes place, such as at the upper boundary, Area C is also split in the second image due to the motion between regions.

Consequently, the apparent motion around boundaries cannot be precisely determined by using any similarity criteria since the areas being compared must have a finite extent. Moreover, it is not realistic to assume that all the wrong matches can be later removed as noise. Due to the similarity of partial areas, wrong correspondences are often assigned in a consistent manner, resulting in overextended image regions.

The key observation is that one should not only rely on *motion cues* in order to perform motion segmentation. Examining the original images reveals a multitude of *monocular cues*, such as intensity edges, that can aid in identifying the true object boundaries. In this context, we formulate the problem of motion analysis as a two-component process that:

- enforces the smoothness of motion, except at its discontinuities, and
- enforces the smoothness of such discontinuities, aided by monocular cues.

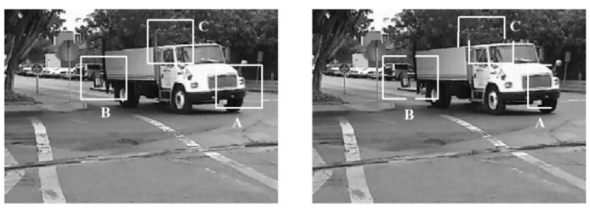


Fig. 2. Nonsimilarity at motion boundaries.

The first component is responsible for extracting the motion layers as salient, smooth surfaces in the 4D space of image coordinates and pixel velocities through a 4D voting process. The layers are then refined by a subsequent 2D voting process that integrates motion with monocular cues (intensity edges) for accurate inference of motion boundaries.

1.3 Motion Interpretation

Given two views of a static scene, the problem of recovering the 3D camera and scene structure has been intensively studied and it is considered well-understood. A set of matching points—typically corresponding to salient image features—are first obtained by methods such as cross-correlation. Assuming that matches are perfect, a simple Eight Point Algorithm [4] can be used for estimating the fundamental matrix and, thus, the epipolar geometry of the cameras is determined. A dense set of matches can then be established, aided by the epipolar constraint, and, finally, the scene structure is recovered through triangulation.

The simplistic approach described above performs reasonably well only in the case when 1) the set of matches contains no outlier noise and 2) the scene is rigid—i.e., without objects having independent motions.

However, the first assumption almost never holds since image measurements are bound to be imperfect and matching techniques will never produce accurate correspondences, mainly due to occlusion or lack of texture. In the presence of incorrect matches, linear methods, such as the Eight Point Algorithm, are very likely to fail. The problem can be reliably solved by robust methods, which involve nonlinear optimization [5], [6] and normalization of data before fundamental matrix estimation [7].

If the second assumption is also violated by the presence of multiple independent motions, even the robust methods may become unstable as the scene is no longer a static one. Even if the dominant epipolar geometry is recovered (for example, the one corresponding to the static background), it is not very clear how to handle misfits—they may be caused by outlier noise, independent motions, or even nonrigid motion.

The core inadequacy of most existing methods is that they attempt to enforce a global constraint—such as the epipolar one—on a data set which may include, in addition to noise, independent subsets that are subject to separate constraints. In this context, it is indeed very difficult to recover structure from motion and segment the scene into independently moving objects if these two tasks are performed simultaneously.

In order to address these difficulties, our approach decouples the above operations, allowing for explicit and separate handling of matching, outlier rejection, grouping, and recovery of camera and scene structure. During motion analysis, we determine an accurate representation in terms of dense velocities (equivalent to point correspondences), segmented motion regions, and boundaries by using only the *smoothness of image motion*. Next, we proceed with the extraction of scene and camera 3D geometry, separately on each rigid component of the scene. Note that our approach follows Ullman's interpretation of visual motion [1] in that the correspondence process takes place prior to 3D interpretation. Furthermore, we also perform segmentation before 3D interpretation based on the smoothness of image motion only.

The main advantage of our approach is that, at the interpretation stage, noisy matches have already been rejected and matches have been grouped according to the distinct moving objects present in the scene. Therefore, standard methods can be reliably applied on each data subset in order to determine the 3D camera and scene structure.

1.4 Method Overview

The first step of the proposed method formulates the motion analysis problem as an inference of motion layers from a noisy and possibly sparse point set in a 4D space. In order to compute a dense set of matches (equivalent to a velocity field) and to segment the image into motion regions, we use an approach based on a *layered 4D representation* of data and a *voting scheme* for communication. First, we establish candidate matches through a multiscale, normalized cross-correlation procedure. Each potential match is encoded as a (x, y, v_x, v_y) point in the 4D space of pixel coordinates and velocities. Within this representation, distinct moving regions in the image correspond to smooth, *salient surface layers* in 4D, which are extracted through a 4D voting process.

Although noisy correspondences are rejected as outliers, there are also wrong matches that are consistent with the correct ones. This mostly occurs at the motion boundaries, where the occluding layer is typically overextended toward the occluded area. In a subsequent stage, the correct motion boundaries are inferred by adding monocular information from the original images. First, we define zones of boundary uncertainty along the margins of layers. Within these zones, we create a 2D saliency map that combines the following information: the position and overall orientation of the layer boundary and the strength and orientation of the intensity edges from the images. The smoothness and continuity of the boundary is then enforced through a 2D voting process, allowing the true boundaries to be extracted as the most *salient curves* within these zones of uncertainty.

The second step interprets the image motion by estimating the 3D scene structure and camera geometry. A rigidity test is performed on the matches within each object to identify potential nonrigid (deforming) objects and, also, between objects, to merge those that move rigidly together but have separate image motions due to depth discontinuities. Finally, the epipolar geometry is estimated separately for each rigid component by using standard methods for parameter estimation (such as the normalized Eight Point Algorithm, LMedS, or RANSAC) and the scene structure and camera motion are recovered by using the dense velocity field.

Our early voting-based approach to grouping from motion that appeared in [8] mainly focused on the use of motion cues only. The integration of motion and monocular cues for boundary extraction through voting has been proposed in conference version in [9]. In this paper, we provide a full coverage of the framework, extend it to the problem of interpretation of image motion, and present many new results.

1.5 Related Work

Barron et al. [10] provide a detailed review of the computational methodologies used in motion analysis. Optical flow techniques—such as differential methods [11], [12], [13], [14], region-based matching [15], [16], [17], or energy-based methods [18]—rely on local, raw estimates of the optical flow field to produce a partition of the image. However, the flow estimates are very poor at motion boundaries and cannot be obtained in uniform areas.

Past approaches have investigated the use of Markov Random Fields (MRF) in handling discontinuities in the optical flow [19], [20], [21], [22]. While these methods give some good results, they rely heavily on a proper spatial segmentation early in the algorithm, which will not be realistic in many cases. Another research direction uses regularization techniques which preserve discontinuities by weakening the smoothing in areas that exhibit strong intensity gradients [23], [24]. Here, an incorrect assumption is also made that the motion boundaries can always be detected in advance, based on intensity only.

Significant improvements have been achieved by using layered representations and the Expectation-Maximization algorithm [25], [26], [27], [28], [29], [30]. There are many advantages of this formalism—mainly because it represents a natural way to incorporate motion field discontinuities and it allows for handling occlusion relationships between regions in the image. While these techniques provide a basis for much subsequent study, they still suffer from some major defects—the procedure requires an initialization step, which is essentially arbitrary, the algorithm is iterative, subject to stability concerns, and the description of the optical flow is parameterized and does not permit a general description as would be desirable. Some methods perform an iterative fitting of data to parametric models [31], [32]. The difficulties involved in this estimation process range from a severe restriction in motion representation (as rigid or planar) to overfitting and instability due to high-order parameterizations. Black and Anandan [33] address the issue of multiple motions through a robust estimation framework based on the Lorentzian estimator and a Graduated Non Convexity algorithm seeking an optimal solution. The approach relies on the assumption that the dominant (background) motion appears throughout a significant part of the image.

An example of using basis set methods (as steerable flow fields) is the work of Fleet et al. [34]. The results are good, but the use of a gradient descent solution is heavily dependent on initial conditions and parameters governing movement in the coefficient space.

Shi and Malik [35] have approached the problem of motion segmentation in terms of recursive partitioning of the spatio-temporal space through normalized cuts within a weighted graph. Grouping pixels based on local measurements does not pose initialization issues, but the approach tends to make early commitments to noisy local measurements. Also, it is difficult to decide where to stop the recursive partitioning process—a region with homogeneous motion may be further segmented due to areas with different intensity patterns.

Wu et al. [36] have applied wavelet techniques to the problem of optical flow determination. Optical flow is described as a linear combination of 2D wavelets. A coarse to fine adjustment is enabled by using different velocity space resolutions in a hierarchical pyramid. This permits capturing a wide range of velocity magnitudes without the instability created by applying coarse to fine adjustment in the intensity space, which can create poor optical flow values at low-resolution where image structure is lost. Instead, full image resolution is used at all levels of the velocity space pyramid. Based on the optical flow constraint, the sum of squared difference is minimized. Poor flow estimation along motion discontinuities is not explicitly addressed. The presence of iteration also leaves open the possibility of instability.

In order to avoid the computational expense of iterative minimizations of functionals, Little et al. [37] developed a parallel algorithm for computing the optical flow by using a local voting scheme based on similarity of planar patches. However, motion boundaries are poorly modeled due to similarity between partial areas in the presence of occlusion.

Tensor voting techniques have been previously applied to structure estimation from motion or stereo. Lee et al. [38] use a 3D voting scheme for stereo in order to eliminate false matches and infer disparity values in regions with low texture. Tong et al. [39] employ tensor voting to identify independently moving regions in stereo, as they form distinct cones in the 4D joint image space. Our work uses a different 4D representation, where motion layers are explicitly modeled, and a subsequent 2D voting pass to estimate curve saliency in order to refine motion boundaries, according to monocular cues (intensity edges).

2 The Tensor Voting Framework

2.1 Tensor Representation and Voting

The use of a voting process for feature inference from sparse and noisy data was introduced by Guy and Medioni [40] and then formalized into a unified tensor framework [3].

Input data is encoded as *elementary tensors*, then support information (including proximity and smoothness of continuity) is propagated by voting. Tensors that lie on smooth, salient features (such as curves or surfaces) strongly support each other and deform according to the prevailing orientation, producing *generic tensors*. Each such tensor encodes the local orientation of features (given by the tensor orientation) and their saliency (given by the tensor shape and size). Features can be then extracted by examining the tensors resulting after voting.

In 3D, a generic tensor can be visualized as an ellipsoid (Fig. 3). It is described by a 3×3 eigensystem, where eigenvectors e_1, e_2, e_3 give the ellipsoid orientation and eigenvalues $\lambda_1, \lambda_2, \lambda_3$ give its shape and size. The tensor is represented as a matrix S:

$$S = \lambda_1 \cdot e_1 e_1^T + \lambda_2 \cdot e_2 e_2^T + \lambda_3 \cdot e_3 e_3^T. \quad (1)$$

There are three types of features in 3D—surfaces, curves, and points—that correspond to three elementary tensors, also shown in Fig. 3. A surface element can be intuitively encoded as a *stick tensor*, where one dimension dominates (along the surface normal), while the length of the stick represents the surface saliency (confidence in this knowledge). A curve element appears as a *plate tensor*,

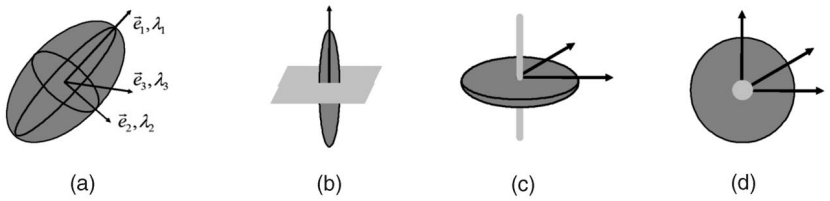


Fig 3. Tensor representation in 3D: (a) generic tensor, (b) stick tensor (surface element), (c) plate tensor, and (d) ball tensor (point element).

where two dimensions codominate (in the plane of curve normals). A point element appears as a *ball tensor*, where no dimension dominates, showing no preference for any particular orientation.

Input tokens are encoded as such elementary tensors. A point element is encoded as a *ball tensor*, with e_1, e_2, e_3 being any orthonormal basis, while $\lambda_1 = \lambda_2 = \lambda_3 = 1$. A curve element is encoded as a *plate tensor*, with e_1, e_2 being normal to the curve, while $\lambda_1 = \lambda_2 = 1$ and $\lambda_3 = 0$. A surface element is encoded as a stick tensor, with e_1 being normal to the surface, while $\lambda_1 = 1$ and $\lambda_2 = \lambda_3 = 0$.

Tokens communicate through a voting process, where each token casts a vote at each token in its neighborhood. The size and shape of this neighborhood and the vote strength and orientation are encapsulated in predefined voting fields (kernels), one for each feature type—there is a stick, a plate, and a ball voting field in the 3D case.

At each receiving site, the collected votes are combined through simple tensor addition, producing generic tensors that reflect the saliency and orientation of the underlying smooth features. Local features can be extracted by examining the properties of a generic tensor, which can be decomposed in its stick, ball, and plate components:

$$S = (\lambda_1 - \lambda_2) \cdot e_1 e_1^T + (\lambda_2 - \lambda_3) \cdot (e_1 e_1^T + e_2 e_2^T) + \lambda_3 \cdot (e_1 e_1^T + e_2 e_2^T + e_3 e_3^T). \quad (2)$$

Each type of feature can be characterized as follows:

- *Surface:* Saliency is $(\lambda_1 - \lambda_2)$, normal orientation is e_1.
- *Curve:* Saliency is $(\lambda_2 - \lambda_3)$, normal orientations are e_1, e_2.
- *Point:* Saliency is λ_3, no preferred orientation.

Therefore, the voting process infers surfaces, curves, and junctions simultaneously, while also identifying outliers (tokens that receive little support). The method is noniterative and robust to considerable amounts of outlier noise. It does not depend on critical thresholds because the only free parameter is the scale factor σ, which defines the voting fields.

Vote generation. For simplicity of illustration, we describe the vote generation process in the 2D case. Tensors in 2D are ellipses (represented by 2×2 eigensystems) and the features are curves and points, corresponding to stick and ball tensors.

Vote strength $VS(\vec{d})$ decays with distance $|\vec{d}|$ between voter and recipient and with curvature ρ:

$$VS(\vec{d}) = e^{-\left(\frac{|\vec{d}|^2 + \rho^2}{\sigma^2}\right)}. \quad (3)$$

Vote orientation corresponds to the smoothest local continuation from voter to recipient—see Fig. 4. A tensor P where curve information is locally known (illustrated by curve normal $\vec{N}_P$) casts a vote at its neighbor Q. The vote orientation is chosen so that it ensures a smooth curve continuation through a circular arc from voter P to recipient Q. To propagate the curve normal $\vec{N}$ thus obtained, the vote $V_{stick}(\vec{d})$ sent from P to Q is encoded as a tensor according to:

$$V_{stick}(\vec{d}) = VS(\vec{d}) \cdot \vec{N}\vec{N}^T. \quad (4)$$

Also, note that the vote strength at Q′ and Q″ is smaller than at Q—because Q′ is farther and Q″ requires a higher curvature than Q.

Fig. 4b shows the 2D stick field with its color-coded strength. When the voter is a ball tensor, with no information known locally, the vote is generated by rotating a stick vote in the 2D plane and integrating all contributions, according to (5). The 2D ball field is shown in Fig. 2c.

$$V_{ball}(\vec{d}) = \int_0^{2\pi} R_\theta \, V_{stick}(R_\theta^{-1}\vec{d}) \, R_\theta^T \, d\theta. \quad (5)$$

2.2 Extension to 4D

The issues to be addressed here are the *tensor representation* of the features in the 4D space and the generation of *voting fields*. Table 1 shows all the geometric features that appear in a 4D space and their representation as *elementary* 4D tensors, where n and t represent normal and tangent

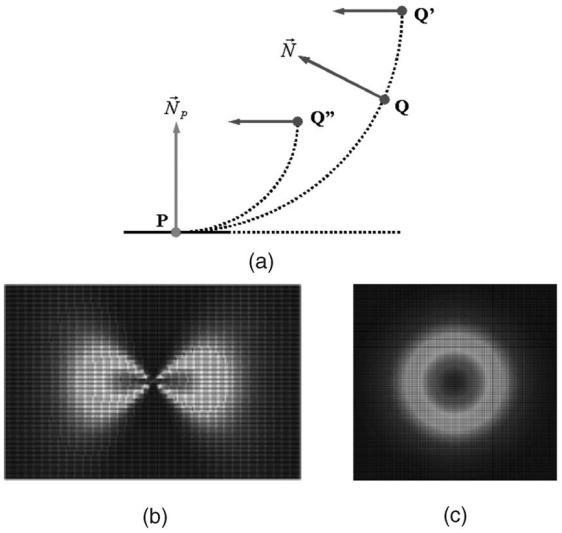


Fig. 4. Vote generation in 2D: (a) vote orientation, (b) 2D stick field, and (c) 2D ball field.

TABLE 1
Elementary Tensors in 4D

Feature	λ_1 λ_2 λ_3 λ_4	e_1 e_2 e_3 e_4	Tensor
point	1 1 1 1	Any orthonormal basis	Ball
curve	1 1 1 0	n_1 n_2 n_3 t	C-Plate
surface	1 1 0 0	n_1 n_2 t_1 t_2	S-Plate
volume	1 0 0 0	n t_1 t_2 t_3	Stick

TABLE 2
A Generic Tensor in 4D

Feature	Saliency	Normals	Tangents
point	λ_4	none	none
curve	λ_3 - λ_4	e_1 e_2 e_3	e_4
surface	λ_2 - λ_3	e_1 e_2	e_3 e_4
volume	λ_1 - λ_2	e_1	e_2 e_3 e_4

vectors, respectively. Note that a surface in the 4D space can be characterized by two normal vectors or by two tangent vectors. From a *generic* 4D tensor that results after voting, the geometric features can be extracted as shown in Table 2.

The voting fields are a key part of the formalism—they are responsible for the size and shape of the neighborhood where the votes are cast and also control how the votes depend on distance and orientation. The 4D voting fields are obtained as follows: First, the 4D stick field is generated in a similar manner to the 2D stick field, as explained in Section 2.1 and illustrated in Fig. 4. Then, the other three voting fields are built by integrating all the contributions obtained by rotating a 4D stick field around appropriate axes. In particular, the 4D ball field—the only one directly used here—is generated according to:

$$V_{ball}(\vec{d}) = \int_0^{2\pi}\int\int R_{\theta_{xy}\theta_{xu}\theta_{xv}}\, V_{stick}(R^{-1}_{\theta_{xy}\theta_{xu}\theta_{xv}}\vec{d})\, R^{T}_{\theta_{xy}\theta_{xu}\theta_{xv}}\, d\theta_{xy}d\theta_{xu}d\theta_{xv}, \tag{6}$$

where x, y, u, v are the 4D coordinates axes, R is the rotation matrix with angles $\theta_{xy}, \theta_{xu}, \theta_{xv}$, and the stick field corresponds to the orientation (1 0 0 0).

The data structure used to store the tensors is an *approximate nearest neighbor (ANN) k-d tree* [41]. The space complexity is $O(n)$, where n is the input size (the total number of candidate tokens). The average time complexity of the voting process is $O(\mu n)$, where μ is the average number of candidates in the neighborhood. Therefore, in contrast to other voting techniques such as the Hough Transform, both the time and space complexities of the Tensor Voting methodology are independent of the dimensionality of the desired feature. The running time for 120×120 image size and three candidates per pixel is about 2 minutes on a Pentium 4 (2 GHz) processor.

Scaling. In our application for motion analysis, the 4D representation (which includes the image coordinate metric and the velocity metric) is not isotropic, while the voting fields are. Therefore, after computing the set of candidate matches, we scale their 4D bounding box so that its size is the same along v_x and x and along v_y and y. After voting, the token coordinates are scaled back to the original space, together with the inferred surface normal at each tensor.

3 Generating Candidate Matches

We take as input two image frames that involve general motion—that is, both the camera and the objects in the scene may be moving. For illustration purposes, we give a description of our approach by using a specific example —the two images in Fig. 5a are taken with a handheld moving camera, where the candy box and the background exhibit distinct image motions due to their different distances from the camera.

For every pixel in the first image, the goal at this stage is to produce candidate matches in the second image. We use a normalized cross-correlation procedure, where all peaks of correlation are retained as candidates. When a peak is found, its position is also adjusted for subpixel precision according to the correlation values of its neighbors. Finally,

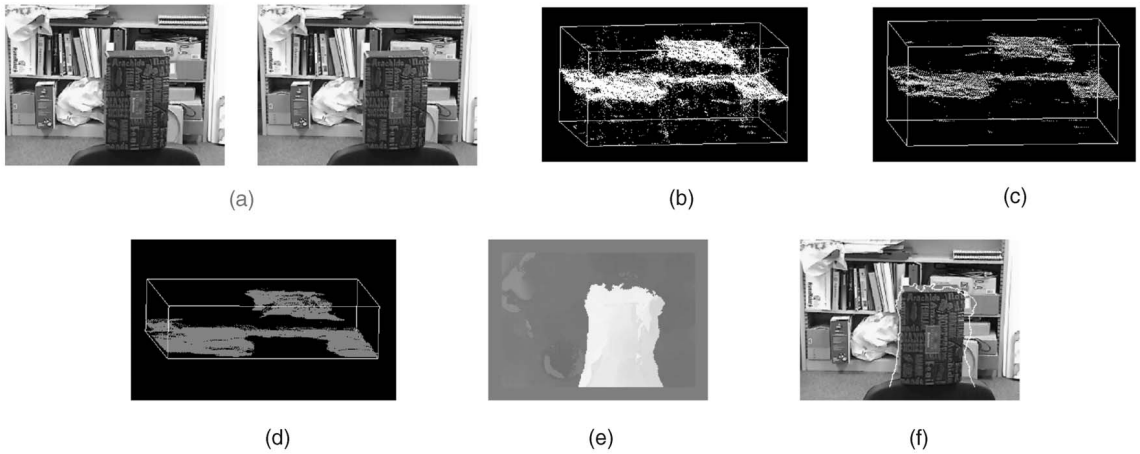


Fig. 5. CANDY BOX sequence—extraction of motion layers: (a) input images, (b) matching candidates, (c) selected velocities, (d) dense layers, (e) layer velocities, and (f) layer boundaries.

each candidate match is represented as an (x, y, v_x, v_y) point in the 4D space of image coordinates and pixel velocities, with respect to the first image.

Since we want to increase the likelihood of including the correct match among the candidates, we repeat this process at multiple scales by using different correlation window sizes. Small windows have the advantage of capturing fine detail and are effective close to the motion boundaries, but produce considerable noise in areas lacking texture or having small repetitive patterns. Larger windows generate smoother matches, but their performance degrades in large areas along motion boundaries. We have experimented with a large range of window sizes and found that the best results are obtained by using only two or three different sizes that should include at least a very small one. Therefore, in all the examples described in this paper, we used three correlation windows, with 3×3, 5×5, and 7×7 sizes.

The resulting candidates appear as a cloud of (x, y, v_x, v_y) points in the 4D space. Fig. 5b shows the candidate matches. In order to display 4D data, the last component of each 4D point has been dropped—the three dimensions shown are x and y (in the horizontal plane) and v_x (the height). The motion layers can already be perceived as their tokens are grouped in two layers surrounded by noisy matches.

Extracting statistically salient structures from such noisy data is very difficult for most existing methods. Because our voting framework is robust to considerable amounts of noise, we can afford to use the multiple window sizes in order to extract the motion layers.

4 Extraction of Motion Layers in 4D

Selection. Since no information is initially known, each potential match is encoded into a 4D *ball tensor*. Then each token casts votes by using the 4D *ball voting field*. During voting, there is strong support between tokens that lie on a smooth surface (layer), while communication between layers is reduced by the spatial separation in the 4D space of both image coordinates and pixel velocities. For each pixel (x, y), we retain the candidate match with the highest surface saliency $(\lambda_2 - \lambda_3)$, and we reject the others as wrong matches. By voting, we also estimate the normals to layers at each token as e_1 and e_2. Fig. 5c shows a 3D view of the recovered matches (the height represents v_x).

Outlier rejection. In the selection step, we kept only the most salient candidate at each pixel. However, there are pixels where all candidates are wrong, such as in areas lacking texture. Therefore, now we eliminate all tokens that have received very little support. Typically, we reject all tokens with surface saliency less than 10 percent of the average saliency of the entire set.

Densification. Since the previous step created holes (i.e., pixels where no velocity is available), we must infer them from the neighbors by using a smoothness constraint. For each pixel (x, y) without an assigned velocity, we try to find the best (v_x, v_y) location at which to place a newly generated token. The candidates considered are all the discrete points (v_x, v_y) between the minimum and maximum velocities in the set, within a neighborhood of the (x, y) point. At each candidate position (x, y, v_x, v_y), we accumulate votes according to the same tensor voting framework that we have used so far. This time, all four voting fields are used, since the previous voting step produced generic tensors. After voting, the candidate token with maximal surface saliency $(\lambda_2 - \lambda_3)$ is retained and its (v_x, v_y) coordinates represent the most likely velocity at (x, y). By following this procedure at every (x, y) image location, we generate a *dense velocity field*. A 3D view of the dense layers is shown in Fig. 5d.

Segmentation. The next step is to group tokens into *regions* by again using the smoothness constraint. We start from an arbitrary point in the image, assign a region label to it, and try to recursively propagate this label to all its image neighbors. In order to decide whether the label must be propagated, we use the smoothness of both velocity and layer orientation as a grouping criterion. Fig. 5e illustrates the recovered v_x velocities within layers (dark corresponds to low velocity) and Fig. 5f shows the layer boundaries superimposed over the first image.

5 Boundary Inference in 2D

At this stage, the extracted motion layers can still be over or underextended along the motion boundaries. This situation typically occurs in areas subject to occlusion, where the initial correlation procedure may generate wrong matches that are consistent with the correct ones and, therefore, could not be rejected as outlier noise.

However, now it is known how many moving objects are present in the scene and where they are. The margins of the layers provide a good estimate for the position and overall orientation of the true motion boundaries. We combine this knowledge with monocular cues (intensity edges) from the original images in order to build a boundary saliency map along the layers margins. Next, we enforce the smoothness and continuity of the boundary through a 2D voting process and extract the true boundary as the most salient curve within the map.

For simplicity of implementation, this procedure is performed in two successive passes—by separately using the horizontal and vertical components of the image gradient. In fact, during the first pass, all edges are found, with the exception of the ones "perfectly" horizontal. The second pass is actually used only to detect the remaining edges. Note that the two steps are interchangeable and their order is not important.

5.1 The Boundary Saliency Map

In the first pass, we start by finding the points that belong to the layer boundaries, identified by changes in region labels along horizontal lines. For each such point (x_c, y_c), we define a horizontal *zone of boundary uncertainty*, centered at (x_c, y_c). Since the over or underextension of motion layers is usually within the limits of the correlation window size, we chose the largest size used in correlation as the zone width. The zone height is one pixel.

Next, we make use of the monocular cues by computing the image gradient (from the intensity I in first image) at each location within the zones of boundary uncertainty:

$$\begin{aligned} G_x(x, y) &= I(x, y) - I(x - 1, y), \\ G_y(x, y) &= I(x, y) - I(x, y - 1). \end{aligned} \tag{7}$$

(a) (b) (c)

Fig. 6. CANDY BOX sequence—boundary inference: (a) boundary saliency map, (b) refined velocities, and (c) refined boundaries.

Since, at this pass, we are looking for nonhorizontal edges, we initialize our saliency map with the horizontal component of the gradient:

$$sal = |G_x(x, y)|. \tag{8}$$

This choice is made in order not to be influenced in the analysis by purely horizontal edges, which will be detected during the second pass. Diagonal edges that exhibit a significant horizontal gradient contribute to the saliency map and they are detected in the first pass.

Finally, we incorporate our estimation of the boundary position and orientation, as resulted from motion cues, by introducing a bias toward the current layer boundaries. Within each zone, we define a weight function W that is 1 at x_c and decays exponentially by:

$$W = e^{-\frac{(x-x_c)^2}{\sigma_W^2}}, \tag{9}$$

where σ_W corresponds to a weight of 0.2 at the zone extremities.

The saliency map is then updated by multiplying each existing value with the corresponding weight.

5.2 Detecting the Boundary

At this stage, we have a saliency value and an orientation at each location within the zones of uncertainty. However, in order to extract the boundaries, we need to examine how neighboring locations agree upon their information through a voting process.

We proceed by encapsulating all the existing information within a 2D tensor framework. Since we have boundary orientations at each location in the uncertainty zones, we create a 2D stick tensor, with the orientation (eigenvectors e_1 and e_2) given by the image gradient and the size taken from the saliency map:

$$\begin{array}{ll} e_1 = (G_x G_y) & \text{(normal to edge)} \\ e_2 = (-G_y G_x) & \text{(tangent to edge)} \\ \lambda_1 = sal & \\ \lambda_2 = 0. & \end{array} \tag{10}$$

The tensors then communicate through a 2D voting process, where each tensor is influenced by the saliency and orientation of its neighbors. After voting, the curve saliency values are collected at each tensor as $(\lambda_1 - \lambda_2)$ and stored back in the saliency map. Fig. 6a shows the tensors after voting, with the local curve tangent given by the eigenvector e_2. The curve saliency $(\lambda_1 - \lambda_2)$ is illustrated here as the length of the tangent vector. Note that, although strong texture edges are present in the uncertainty zone, after voting their saliency is diminished by the overall dominance of saliency and orientation of the correct object edges.

The true boundaries are extracted as follows: First, the token with global maximal curve saliency is chosen as a (starting) boundary point. Given a current boundary point C, the neighboring uncertainty zones are visited; in each zone, the next boundary point N is chosen as the one that maximizes the directional saliency $s_d = s \cdot \cos\alpha$, where s is the curve saliency at N and α is the angle between the curve tangent at C and the segment CN. Note that this procedure identifies all edges, except for purely horizontal ones. After marking the detected boundaries, the entire process is repeated in a similar fashion in the second pass, this time using the vertical component of the gradient, in order to detect any horizontal boundaries that have been missed during the first pass.

Finally, within each zone of uncertainty, all pixels between the old and the new boundary points are reassigned to the correct region. In addition to changing the region label, their velocities are recomputed in a 4D voting process similar to densification. However, since region labels are now available, the votes are collected only from points *within the same layer*.

Fig. 6b shows the refined velocities within layers (dark represents small velocity) and Fig. 6c shows the refined motion boundary that, indeed, corresponds to the actual object.

We have also analyzed several other image sequences and we present the results obtained here. In all experiments we used three correlation windows, with 3×3, 5×5, and 7×7 sizes, and, for each window, we retained all peaks of correlation. Therefore, each pixel in the image had at least three candidate matches, among which at most one was correct. For both the 4D and 2D voting processes, in all examples we used the same scale factor, corresponding to an image neighborhood with a radius of 16 pixels.

FISH sequence (Fig. 7). To quantitatively estimate the performance of our approach, we created a synthetic sequence from real images. The silhouette of a fish was cropped from its image and pasted at different locations over a second image in order to generate a motion sequence with ground truth. The average angular error we obtained is 0.42 degrees ± 1.2 degrees for 100 percent field coverage, which is very low, despite the multitude of texture edges from the cluttered background that were competing with the true object edges. This example is also used to show that we can successfully handle more detailed and nonconvex motion boundaries.

BARRIER sequence (Fig. 8). We analyzed the motion from two frames of a sequence showing two cars moving away from the camera. The analysis is difficult due to the large ground area with very low texture and because the

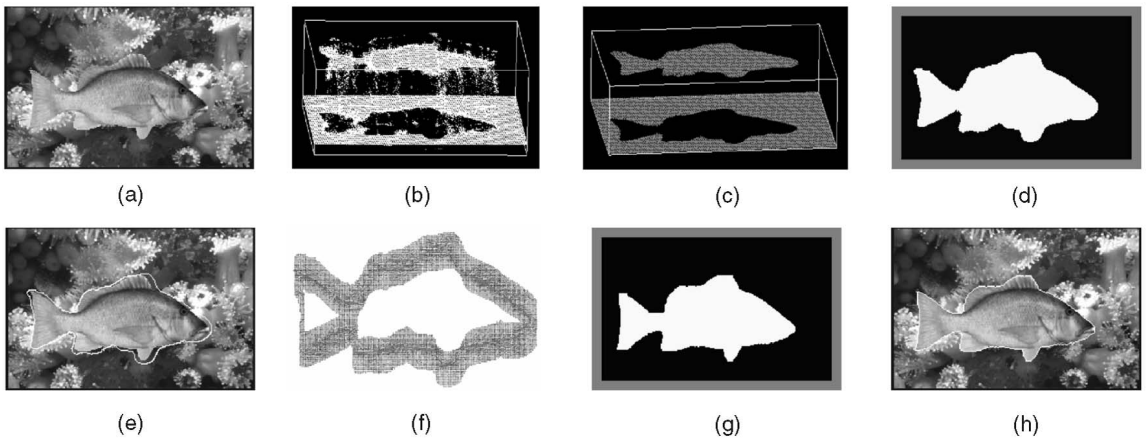


Fig. 7. FISH sequence: (a) one input image, (b) candidate matches, (c) dense layers, (d) layer velocities, (e) layer boundaries, (f) boundary saliency map, (g) refined layers, and (h) refined boundaries.

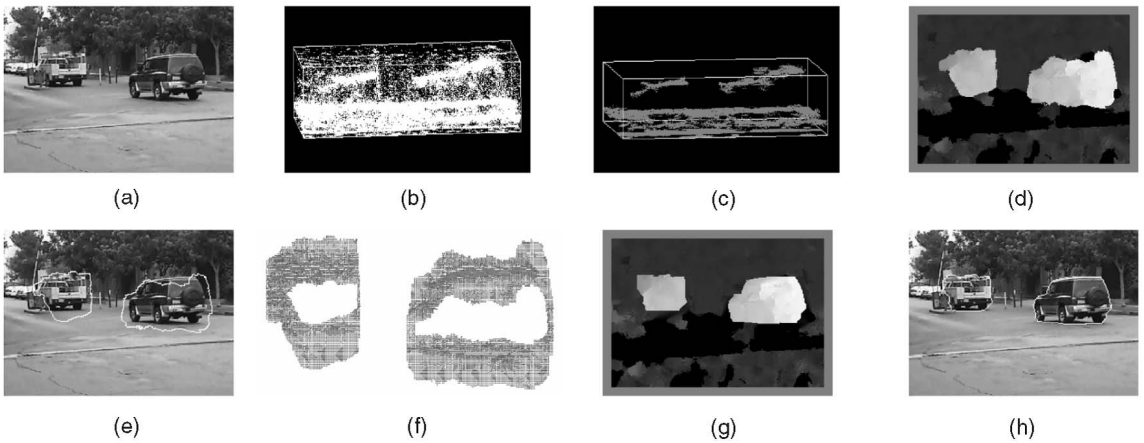


Fig. 8. BARRIER sequence: (a) one input image, (b) candidate matches, (c) dense layers, (d) layer velocities, (e) layer boundaries, (f) boundary saliency map, (g) refined layers, and (h) refined boundaries.

two moving objects have relatively small sizes in the image. Also note that the *image* motion is not translational—the front of each car has a lower velocity than its back. This is visible in the 3D view of the motion layers, which appear as tilted surfaces. In fact, our framework does not make any assumption regarding the type of motion—such as translational, planar, or rigid motion. The *only* criterion used is the smoothness of *image* motion.

6 Three-Dimensional Interpretation

So far, we have not made any assumption regarding the 3D motion and the only constraint used has been the smoothness of image motion. The observed image motion could have been produced by the 3D motion of objects in the scene or the camera motion or both. Furthermore, some of the objects may be subject to nonrigid motion.

For classification, we used an algorithm introduced by McReynolds and Lowe [42] that verifies the potential rigidity of a set of minimum six point correspondences from two views under perspective projection. The rigidity test is performed on a subset of matches within each object to identify potential nonrigid objects and also across objects, to merge those that move rigidly together but have distinct image motions due to depth discontinuities. It is also worth mentioning that the rigidity test is actually able to only guarantee the *nonrigidity* of a given configuration. Indeed, if the rigidity test fails, it means that the image motion is not compatible to a rigid 3D motion and, therefore, the configuration *must* be nonrigid. If the test succeeds, it only asserts that a possible rigid 3D motion exists that is compatible to the given image motion. However, this computational approach corresponds to the way human vision operates—as shown in [1], human perception solves this inherent ambiguity by always choosing a rigid interpretation when possible.

The remaining task at this stage is to determine the object (or camera) motion and the scene structure. Since wrong matches have been eliminated and correct matches are already grouped according to the rigidly moving objects in the scene, standard methods for reconstruction can be reliably applied. For increased robustness, we chose to use RANSAC [43] to recover the epipolar geometry for each rigid object, followed by an estimation of camera motion and projective scene structure.

The following discussion describes each case, illustrated with experimental results.

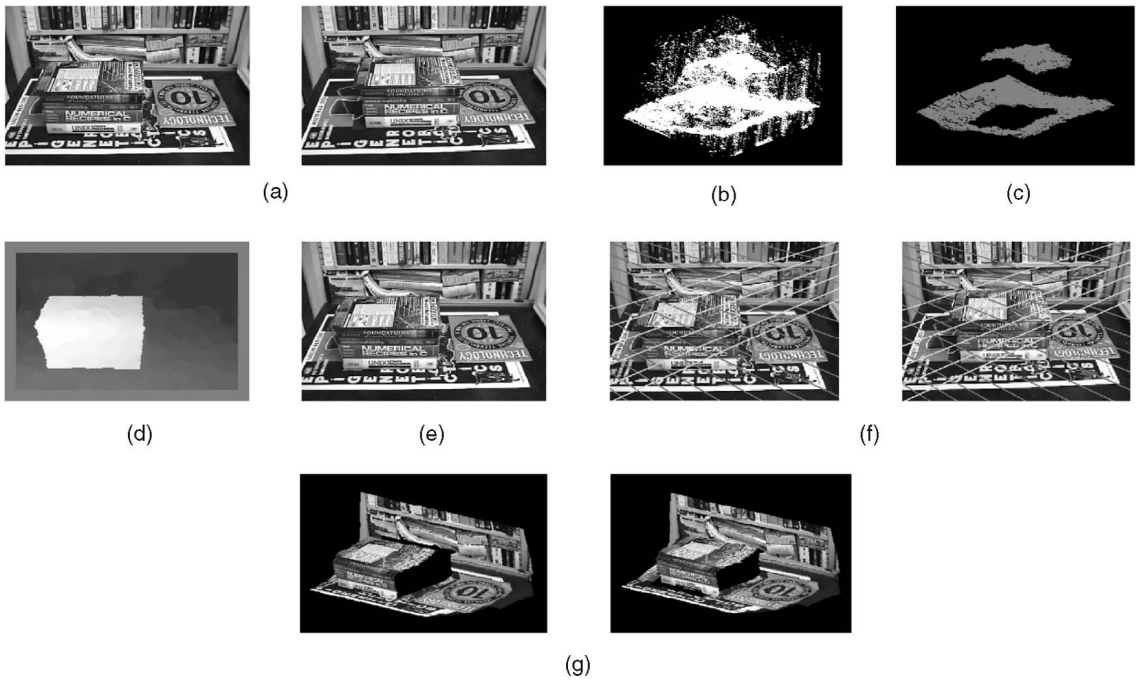


Fig. 9. BOOKS sequence: (a) input images, (b) candidate matches, (c) dense layers, (d) layer velocities, (e) layer boundaries, (f) epipolar lines, and (g) 3D structure and motion.

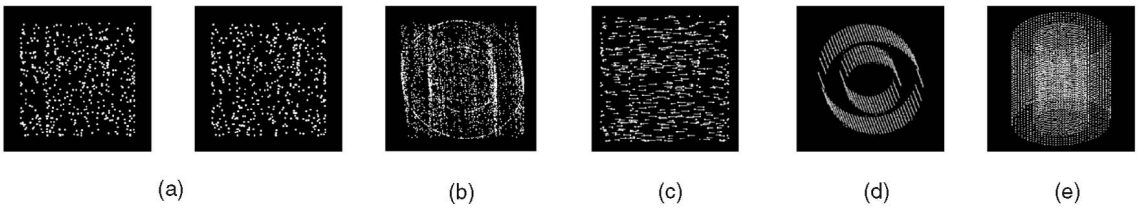


Fig. 10. CYLINDERS sequence: (a) input images, (b) candidate matches, (c) velocities, (d) dense layers, and (e) 3D structure.

Multiple rigid motions. This case is illustrated by the BOOKS example in Fig. 9, where two sets of matches have been detected, corresponding to the two distinct objects—the stack of books and the background. The rigidity test shows that, while each object moves rigidly, they cannot be merged into a single rigid structure. The recovered epipolar geometry is illustrated in Fig. 9f, while the 3D scene structure and motion are shown in Fig. 9g.

The CYLINDERS example, shown in Fig. 10, is adapted from Ullman [1] and consists of two images of random points in a sparse configuration, taken from the surfaces of two transparent coaxial cylinders rotating in opposite directions. This extremely difficult example clearly illustrates the power of our approach, which is able to determine accurate point correspondences and scene structure—even from a sparse input, based on motion cues only (without any monocular cues), and for transparent motion.

In the CAR example, shown in Fig. 11, the sign and the background correspond to a rigid configuration and can be merged, while the car exhibits an independent motion.

Single rigid motion. This is the stereo case, illustrated by the CANDY BOX example in Fig. 12, where the scene is static and the camera is moving. Due to the depth disparity between the box and the background, their image motions do not satisfy the smoothness constraint together and, thus, they have been segmented as two separate objects. However, the rigidity test shows that the two objects form a rigid configuration and, therefore, are labeled as a single object. The epipolar geometry estimation and scene reconstruction are then performed on the entire set of matches. Along with the 3D structure, Fig. 12c also shows the two recovered camera positions.

Nonrigid motion. The FLAG example, shown in Fig. 13, is a synthetic sequence where sparse random dots from the surface of a waving flag are displayed in two frames. The configuration is recognized as nonrigid and, therefore, no reconstruction is attempted. However, since the *image motion* is smooth, our framework is still able to determine correct correspondences, extract motion layers, segment nonrigid objects, and label them as such.

7 Experimental Comparison

7.1 Motion Segmentation

We use two different image pairs for quantitative estimation of motion segmentation—the TENNIS sequence (Fig. 14) and the COASTGUARD sequence (Fig. 15). For those two image pairs, the edge-based motion segmentation [44], [45], graph cuts [35], and tensor voting approaches are compared. In the illustrations, pixels correctly assigned to

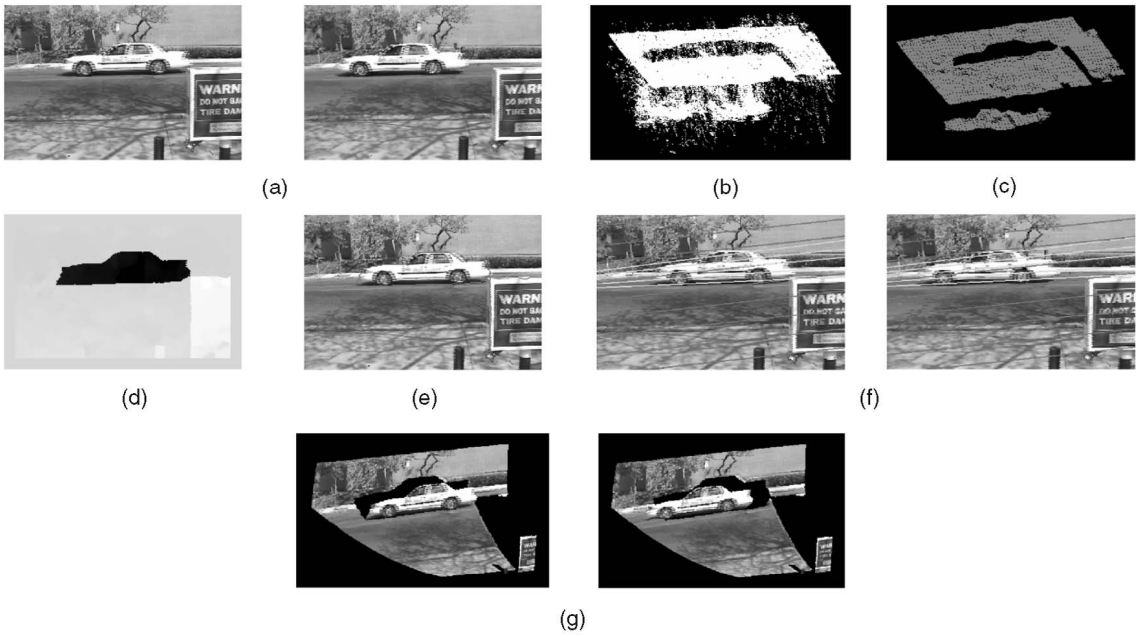


Fig. 11. CAR sequence: (a) input images, (b) candidate matches, (c) dense layers, (d) layer velocities, (e) layer boundaries, (f) epipolar lines, and (g) 3D structure and motion.

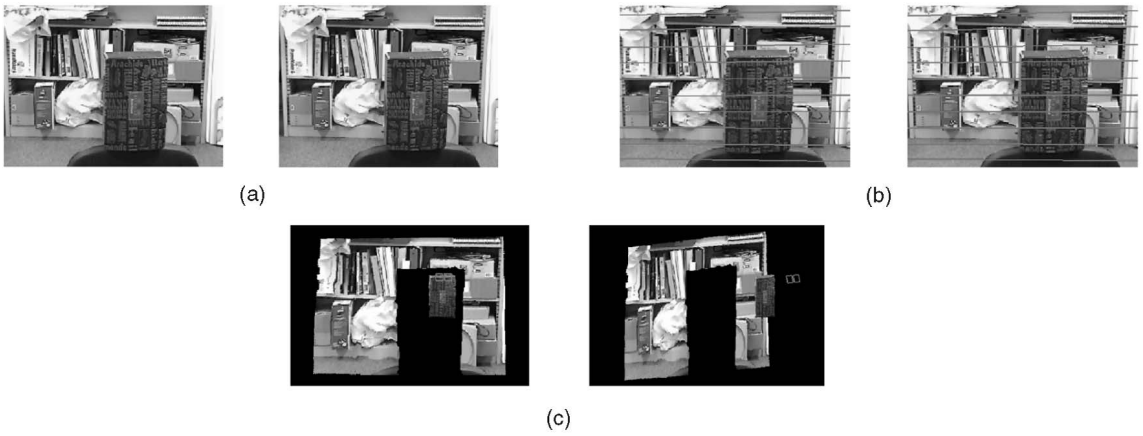


Fig. 12. CANDY BOX sequence: (a) input images, (b) epipolar lines, and (c) 3D structure and motion.

the foreground are shown in a darker shade of gray. The error rate results (representing the percentage of pixels assigned to erroneous motion regions) are presented in Table 3.

For the TENNIS sequence, all methods produce quite similar quantitative results, with the voting approach having a slightly better error rate. From a qualitative perspective, the tensor voting method appears to be the best as it is the only one able to correctly segment the ball. Also note the incorrect segmentation of the upper arm due to the articulated motion, a limitation that affects all three methods considered here.

For the COASTGUARD sequence, the quantitative results are again quite similar, but the graph cuts and voting-based methods produce a qualitatively better output, being able to recover the thin mast and the boat's fore and aft sections.

Furthermore, we applied the graph cuts technique to the CYLINDERS sequence described in Section 6. The results are shown in Fig. 16, which illustrates the recovered motion layers for the graph cuts and voting methods (without

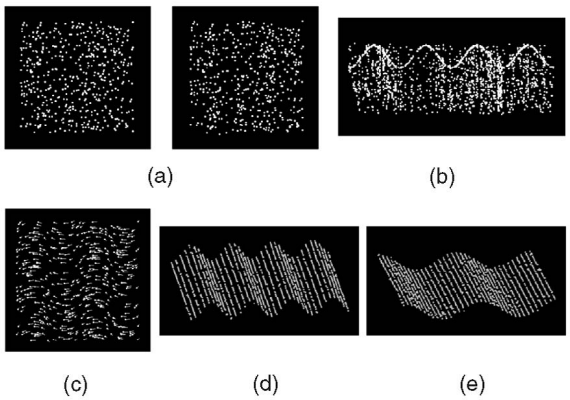
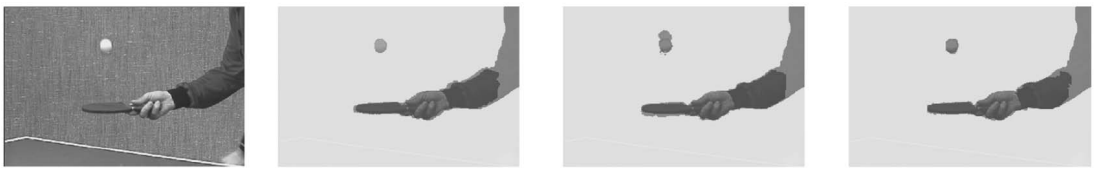
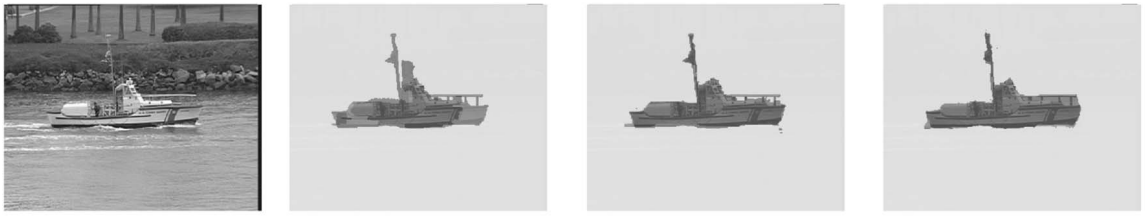


Fig. 13. FLAG sequence: (a) input images, (b) candidate matches, (c) velocities, (d) dense layers (v_x), and (e) dense layers (v_y).



(a) (b) (c) (d)

Fig. 14. TENNIS sequence: (a) one input image, (b) edge-based segmentation, (c) graph cuts, and (d) tensor voting.



(a) (b) (c) (d)

Fig. 15. COASTGUARD sequence: (a) one input image, (b) edge-based segmentation, (c) graph cuts, and (d) tensor voting.

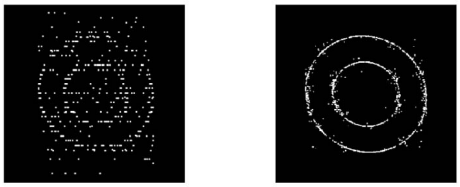
densification, for a fair comparison). This challenging example suggests that the voting-based method is better suited for the case of transparent motion and sparse data. It is also worth mentioning that our method does not assume any motion models or particular shapes for image regions, so that it can consistently provide good results over a wide range of configurations.

7.2 Motion Interpretation

We also analyzed a standard sequence (the TEDDY example—Fig. 17) with ground truth available to provide a quantitative estimate for the performance of our approach as compared to stereo methods. As shown in Table 4 (partially reproduced from [46]), our voting-based approach has the smallest error rate (percentage of pixels with a disparity error greater than 1) among the techniques mentioned.

TABLE 3
TENNIS and COASTGUARD Sequences—Results

Methods	Error Rate (TENNIS)	Error Rate (COASTGUARD)
Edge-based motion segmentation	5.6%	4.1%
Graph cuts	5.2%	1.4%
Tensor voting	5.1%	2.3%



(a) (b)

Fig. 16. CYLINDERS sequence: (a) graph cuts—layers and (b) tensor voting—layers.

8 Conclusions and Future Work

We have presented a novel approach that decouples grouping and interpretation of visual motion, allowing for explicit and separate handling of matching, outlier rejection, grouping, and recovery of camera and scene structure. The proposed framework is able to handle data sets containing large amounts of outlier noise, as well as multiple independently moving objects, or nonrigid objects.

Our methodology for extracting motion layers is based on a *layered 4D representation* of data and a *voting scheme* for token communication. Monocular cues—represented by intensity edges in the original images—are subsequently used to refine the layer boundaries through a *2D voting process*. The *boundary saliency map* proves to be an appropriate representation for this problem. It encodes the position, orientation, and strength of both the layer

(a) (b) (c)

Fig. 17. TEDDY sequence: (a) one input image, (b) ground truth disparity map, and (c) tensor voting disparity map.

TABLE 4
TEDDY Sequence—Results

Methods	Error Rate
Graph cuts	29.3%
Sum of squared differences	26.5%
Dynamic programming	30.1%
Tensor voting	15.4%

boundaries and image edges, all contributing to accurate inference of the motion boundaries.

The proposed method allows for structure inference without using any prior knowledge of the motion model, based on the smoothness of motion only, while consistently handling both smooth moving regions and motion discontinuities. The method is also computationally robust, being noniterative, and does not depend on critical thresholds, the only free parameter being the scale of analysis. We plan to extend our approach by incorporating information from multiple frames and to study the possibility of using an adaptive scale of analysis in the voting process.

ACKNOWLEDGMENTS

This research has been funded in part by the Integrated Media Systems Center, a US National Science Foundation (NSF) Engineering Research Center, Cooperative Agreement No. EEC-9529152, and by NSF Grant 9811883. Changki Min provided help with the experimental comparison to other methods.

REFERENCES

[1] S. Ullman, *The Interpretation of Visual Motion.* MIT Press, 1979.
[2] E. Hildreth, *The Measurement of Visual Motion.* MIT Press, 1983.
[3] G. Medioni, M.-S. Lee, and C.-K. Tang, *A Computational Framework for Segmentation and Grouping.* Elsevier Science, 2000.
[4] H.C. Longuet-Higgins, "A Computer Algorithm for Reconstructing a Scene from Two Projections," *Nature,* vol. 293, pp. 133-135, 1981.
[5] A. Adam, E. Rivlin, and L. Shimshoni, "Ror: Rejection of Outliers by Rotations," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 1, pp. 78-84, Jan. 2001.
[6] Z. Zhang, "Determining the Epipolar Geometry and Its Uncertainty: A Review," *Int'l J. Computer Vision,* vol. 27, no. 2, pp. 161-195, 1998.
[7] R.I. Hartley, "In Defense of the 8-Point Algorithm," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 6, pp. 580-593, June 1997.
[8] M. Nicolescu and G. Medioni, "Layered 4D Representation and Voting for Grouping from Motion," *IEEE Trans. Pattern Analysis and Machine Intelligence,* special issue on perceptual organization in computer vision, vol. 25, no. 4, pp. 492-501, Apr. 2003.
[9] M. Nicolescu and G. Medioni, "Motion Segmentation with Accurate Boundaries—A Tensor Voting Approach," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 382-389, 2003.
[10] J. Barron, D. Fleet, and S. Beauchemin, "Performance of Optical Flow Techniques," *Int'l J. Computer Vision,* vol. 12, no. 1, pp. 43-77, Feb. 1994.
[11] B. Horn and B. Schunck, "Determining Optical Flow," *Artificial Intelligence,* vol. 17, pp. 185-204, 1981.
[12] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proc. DARPA Image Understanding Workshop,* pp. 121-130, 1981.
[13] E. Simoncelli, E. Adelson, and D. Heeger, "Probability Distributions of Optical Flow," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 310-315, 1991.
[14] H.-H. Nagel and W. Enkelmann, "An Investigation of Smoothness Constraints for the Estimation of Displacement Vector Fields from Image Sequences," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, pp. 565-593, 1986.
[15] P. Anandan, "A Computational Framework and an Algorithm for the Measurement of Visual Motion," *Int'l J. Computer Vision,* vol. 2, pp. 283-310, 1989.
[16] P. Burt and E. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Trans. Comm.,* vol. 31, pp. 532-540, 1983.
[17] A. Singh, *Optical Flow Computation: A Unified Perspective.* IEEE Press, 1992.
[18] D. Heeger, "Optical Flow Using Spatiotemporal Filters," *Int'l J. Computer Vision,* vol. 1, pp. 279-302, 1988.
[19] F. Heitz and P. Bouthemy, "Multimodel Estimation of Discontinuous Optical Flow Using Markov Random Fields," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 12, pp. 1217-1232, Dec. 1993.
[20] M. Gelgon and P. Bouthemy, "A Region-Level Graph Labeling Approach to Motion-Based Segmentation," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 514-519, June 1997.
[21] C. Kervrann and F. Heitz, "A Markov Random Field Model-Based Approach to Unsupervised Texture Segmentation Using Local and Global Spatial Statistics," *IEEE Trans. Image Processing,* vol. 4, no. 6, pp. 856-862, 1995.
[22] Y. Boykov, O. Veksler, and R. Zabih, "Markov Random Fields with Efficient Approximations," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 648-655, 1998.
[23] S. Ghosal, "A Fast Scalable Algorithm for Discontinuous Optical Flow Estimation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 2, pp. 181-194, Feb. 1996.
[24] R. Deriche, P. Kornprobst, and G. Aubert, "Optical Flow Estimation while Preserving Its Discontinuities: A Variational Approach," *Proc. Asian Conf. Computer Vision,* pp. 290-295, 1995.
[25] T. Darrell and A. Pentland, "Robust Estimation of a Multilayered Motion Representation," *Proc. IEEE Workshop Visual Motion,* pp. 173-178, 1991.
[26] S. Ayer and H. Sawhney, "Layered Representation of Motion Video Using Robust Maximum Likelihood Estimation of Mixture Models and MDL Encoding," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 777-784, 1995.
[27] G. McLachlan and K. Basford, *Mixture Models Inference and Applications to Clustering.* Marcel Dekker, Inc., 1988.
[28] S. Hsu, P. Anandan, and S. Peleg, "Accurate Computation of Optical Flow by Using Layered Motion Representation," *Proc. Int'l Conf. Pattern Recognition,* 1994.
[29] A. Jepson and M. Black, "Mixture Models for Optical Flow Computation," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 760-761, 1993.
[30] Y. Weiss, "Smoothness in Layers: Motion Estimation Using Nonparametric Mixture Estimation," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 520-526, 1997.
[31] M. Irani and S. Peleg, "Image Sequence Enhancement Using Multiple Motions Analysis," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 216-221, 1992.
[32] J. Wang and E. Adelson, "Representing Moving Images with Layers," *IEEE Trans. Image Processing,* special issue on image sequence compression, vol. 3, no. 5, pp. 625-638, Sept. 1994.
[33] M. Black and P. Anandan, "The Robust Estimation of Multiple Motions: Parametric and Piecewise-Smooth Flow Fields," *Computer Vision and Image Understanding,* vol. 63, no. 1, pp. 75-104, 1996.
[34] D. Fleet, M. Black, and A. Jepson, "Motion Feature Detection Using Steerable Flow Fields," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 274-281, June 1998.
[35] J. Shi and J. Malik, "Motion Segmentation and Tracking Using Normalized Cuts," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 1154-1160, Jan. 1998.
[36] Y.-T. Wu, T. Kanade, J. Cohn, and C.-C. Li, "Optical Flow Estimation Using Wavelet Motion Model," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 992-998, Jan. 1998.
[37] J. Little, H. Bulthoff, and T. Poggio, "Parallel Optical Flow Using Local Voting," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 454-459, 1988.
[38] M.S. Lee, G. Medioni, and P. Mordohai, "Inference of Segmented Overlapping Surfaces from Binocular Stereo," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 6, pp. 824-837, June 2002.
[39] W.S. Tong, C.K. Tang, and G. Medioni, "Simultaneous Two-View Epipolar Geometry Estimation and Motion Segmentation by 4D Tensor Voting," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 9, pp. 1167-1184, Sept. 2004.
[40] G. Guy and G. Medioni, "Inference of Surfaces, 3-D Curves, and Junctions from Sparse, Noisy 3-D Data," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 11, pp. 1265-1277, Nov. 1997.
[41] S. Arya, D. Mount, N. Netanyahu, R Silverman, and A. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions," *J. ACM,* vol. 45, no. 6, pp. 891-923, 1998.

[42] D. McReynolds and D. Lowe, "Rigidity Checking of 3D Point Correspondences Under Perspective Projection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 12, pp. 1174-1185, Dec. 1996.
[43] P.H.S. Torr and D.W. Murray, "A Review of Robust Methods to Estimate the Fundamental Matrix," *Proc. Int'l J. Computer Vision,* 1997.
[44] P. Smith, T. Drummond, and R. Cipolla, "Layered Motion Segmentation and Depth Ordering by Tracking Edges," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 4, pp. 479-494, Apr. 2004.
[45] P. Smith, "Edge-Based Motion Segmentation," PhD dissertation, Univ. of Cambridge, 2001.
[46] D. Scharstein and R. Szeliski, "High-Accuracy Stereo Depth Maps Using Structured Light," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 195-202, 2003.

Mircea Nicolescu received the BS degree from the Polytechnic University Bucharest, Romania, in 1995, the MS degree from the University of Southern California in 1999, and the PhD degree from the University of Southern California in 2003, all in computer science. He is currently an assistant professor of computer science at the University of Nevada, Reno, and codirector of the Computer Vision Laboratory. His research interests include panoramic video systems, real-time segmentation and tracking, motion analysis, and perceptual grouping. In 1999 and 2003, he received the USC Academic Achievements Award and, in 2002, the Best Student Paper Award at the International Conference on Pattern Recognition in Quebec City, Canada. He is a member of the IEEE and the IEEE Computer Society.

Gérard Medioni received the Diplôme d'Ingénieur Civil from the Ecole Nationale Supérieure des Télécommunications, Paris, France, in 1977, and the MS and PhD degrees in computer science from the University of Southern California, Los Angeles, in 1980 and 1983, respectively. He has been with the University of Southern California (USC) in Los Angeles since 1983, where he is currently a professor of computer science and electrical engineering, codirector of the Computer Vision Laboratory, and chairman of the Computer Science Department. He was a visiting scientist at INRIA Sophia Antipolis in 1993 and chief technical officer of Geometrix, Inc., during his sabbatical leave in 2000. His research interests cover a broad spectrum of the computer vision field and he has studied techniques for edge detection, perceptual grouping, shape description, stereo analysis, range image understanding, image to map correspondence, object recognition, and image sequence analysis. He has published more than 100 papers in conference proceedings and journals. Dr. Medioni is a fellow of the IEEE and the IAPR. He has served on the program committees of many major vision conferences and was program chairman of the 1991 IEEE Computer Vision and Pattern Recognition Conference in Maui, program cochairman of the 1995 IEEE Symposium on Computer Vision held in Coral Gables, Florida, general cochair of the 1997 IEEE Computer Vision and Pattern Recognition Conference in Puerto Rico, program cochair of the 1998 International Conference on Pattern Recognition held in Brisbane, Australia, and general cochairman of the 2001 IEEE Computer Vision and Pattern Recognition Conference in Kauai. Professor Medioni is an associate editor of the *Pattern Recognition and Image Analysis* journal and one of the North American editors for the *Image and Vision Computing* journal.