Simultaneous Two-View Epipolar Geometry Estimation and Motion Segmentation by 4D Tensor Voting

Wai-Shun Tong †  
Dept. of Computer Science, Hong Kong University of Science & Technology, Clear Water Bay, Hong Kong.  
Email: cstws@cs.ust.hk

Chi-Keung Tang †  
Dept. of Computer Science, Hong Kong University of Science & Technology, Clear Water Bay, Hong Kong.  
Tel: +852-2358-8775  Fax: +852-2358-1477  Email: cktang@cs.ust.hk

Gérard Medioni *  
PHE 204, MC0273, Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, CA 90083-0273, USA  
Tel: +1-213-740-6440  Fax: +1-213-740-7877  Email: medioni@iris.usc.edu

† Hong Kong University of Science and Technology  
* University of Southern California

A preliminary version of this paper appears in the proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2001. In this submission, we use color codes in many figures. Reviewers are encouraged to review the color pdf file we submitted to TPAMI and view all figures on-line using Acrobat Reader.

February 6, 2004
SIMULTANEOUS TWO-VIEW EPIPOLAR GEOMETRY ESTIMATION AND MOTION SEGMENTATION BY 4D TENSOR VOTING

Wai-Shun Tong, Chi-Keung Tang, Gérard Medioni

Abstract

We address the problem of simultaneous two-view epipolar geometry estimation and motion segmentation from non-static scenes. Given a set of noisy image pairs containing matches of \( n \) objects, we propose an unconventional, efficient and robust method, 4D tensor voting, for estimating the unknown \( n \) epipolar geometries, and segmenting the static and motion matching pairs into \( n \) independent motions. By considering the 4D isotropic and orthogonal joint image space, only two tensor voting passes are needed, and a very high noise to signal ratio (up to five) can be tolerated. Epipolar geometries corresponding to multiple, rigid motions are extracted in succession. Only two uncalibrated frames are needed, and no simplifying assumption (such as affine camera model or homographic model between images) other than the pin-hole camera model is made. Our novel approach consists of propagating a local geometric smoothness constraint in the 4D joint image space, followed by global consistency enforcement for extracting the fundamental matrices corresponding to independent motions. We have performed extensive experiments to compare our method with some representative algorithms to show that better performance on non-static scenes are achieved. Results on challenging datasets are presented.

Keywords

Epipolar geometry, motion segmentation, non-static scene, robust estimation, higher dimensional inference

1 Introduction

In the presence of moving objects, image registration becomes a more challenging problem, as the matching and registration phases become interdependent. Many researchers assume a homographic model between images (e.g. [15]), and detect motion by residual, or more than two frames are used (e.g. [5]). Three views are used to define a trifocal tensor [9]\(^1\), which has a large number of parameters to be estimated. Torr and Murray [21] use epipolar geometry to detect independent motion. Wolf and Shashua [24] proposed the segmentation matrix that encapsulates two-body motion geometry from two views. Vidal et al. [23] also considered two views, and generalized the notion of fundamental matrix into a multibody fundamental matrix involving more than two independently moving objects. In this paper, we propose to perform epipolar geometry estimation for non-static scenes using 4D tensor voting, which runs in linear time in

\(^1\)In this paper, we use a geometric tensor to encode structure information in 4D. Our use of a 4D symmetric tensor is different from trilinear or trifocal tensors found in some epipolar geometry literature.
the number of image pairs. Our geometric approach addresses the motion segmentation and outlier rejection simultaneously for fundamental matrix estimation, via a novel voting approach.

In the presence of a large number of false matches, linear methods such as the Eight Point Algorithm [10] are likely to fail. Thanks to many previous attempts, which either remove most of the false matches by non-linear optimization [1], [25], or perform data normalization [8] before fundamental matrix estimation, we can robustly solve the epipolar geometry problem efficiently. However, when false matches and matches due to consistent motion are both present, the above methods become unstable since the single fundamental matrix assumption is violated. Depending on the size and number of the objects in motion, robust methods [25] may return a totally incorrect epipolar geometry of the scene. Also, even if the fundamental matrix corresponding to the static background can be recovered, correspondences for objects in motion are usually discarded as outliers. While outliers are still the main issue in the presence of moving objects (in static scene as well, as Hartley suggested [8]), the interdependence of motion segmentation, outlier rejection, and parameter estimation makes the problem particularly difficult to solve. Specifically, the most dominant motion (e.g. background) may treat other less salient motion matches as outliers.

In this paper, we propose an unconventional and non-iterative geometric approach that establishes strong geometric constraint in the 4D joint image space on true matching pairs (static or motion). Robust segmentation is performed in a non-parametric manner. Our method contrasts with many algebraic approaches which optimize certain functionals for estimating a large number of parameters. The 4D tensor voting method and the non-iterative Hough transform [4] share the similar idea that a voting technique is employed to output the solution receiving maximal support. However, as the dimensionality grows, Hough transform is extremely inefficient, and thus is impractical in higher dimensional detection problems. Technical differences between the two methods will be discussed later in depth, following a detailed description of our 4D algorithm. We show that 4D tensor voting is very effective in rejecting outlier noise resulted from wrong matches, and is capable of identifying and segmenting salient motions in succession, by using a traditional RANSAC approach to recover the epipolar geometry.

The rest of this paper is organized as follows. We first give an outline of the algorithm. In section 3 we review related work and provide background knowledge on epipolar geometry. In section 4 we give a contextual review on tensor voting. Section 5 presents the details of motion segmentation and outlier rejection. Section 6 evaluates the time complexity of our system (it is a linear-time algorithm in the number of matches). In sections 7–8, we present our result and
evaluate our system, where we discuss our method from the practitioners’ point of view. Finally, we conclude our work with future research direction in section 9.

A preliminary version of this paper appears in [20]. The present coverage makes the description complete by providing a quantitative comparison of our 4D approach with widely used techniques for epipolar geometry estimation, such as LMedS, M-estimators, and RANSAC. Experimental results were performed to evaluate the accuracy of estimation on both static and non-static scenes, and on both real and synthetic data (for the latter, ground truth data are available). An analysis of the applicability of our approach and adequate discussion are also given.

2 Outline of the algorithm

The key idea is that inliers, even if they belong to a small and independently moving object, reinforce each other after voting, whereas outliers do not support each other. In 4D tensor voting, strong geometric constraints for motion inliers are enforced, reinforcing the saliency of good matches and suppressing that of outliers simultaneously using a non-iterative voting scheme that gathers tensorial support. There are two stages for extracting each motion in succession:

- **Local smoothness constraint enforcement (outlier rejection and motion segmentation).** We first apply 4D tensor voting, by exploiting the fact that in the 4D joint image space, the corresponding 4D joint image of a point match lie on a 4D cone. We also show that independently moving objects in fact give rise to different 4D cones in the joint image space. We use 4D tensor voting, in exactly two passes (c.f. [18]), to enforce the local continuity constraint in 4D, so that false matches (which must not lie on any of the 4D cones) are discarded. Matching points that satisfy the continuity constraint are retained.

- **Global epipolar constraint enforcement (parameter estimation).** Using this filtered set of matches, we show that even a slight modification of a simple and efficient algorithm like the simple eight-point linear method, or random sub-sampling, can extract different motion components in a non-static scene. Epipolar geometries corresponding to multiple motions are also estimated.

The overall approach is summarized in Fig. 1. The input to the algorithm is a set of candidate point matches, which can be obtained manually, or by automatic cross-correlation.

Our 4D algorithm works by first normalizing the input point set (using Hartley’s approach [8]). After data normalization, we convert the dataset into a sparse set of 4D points (section 5.1).

Next, we convert the 4D coordinates into tensors, which will be discussed in section 5.2. Our outlier rejection requires estimating the normal to the 4D cone, if the joint image corresponds
Fig. 1. Flowchart of the 4D system. Three motions are identified.

to a point lying on it. To this end, we use 4D ball voting field (section 5.3) to vote for normals. The local continuity constraint is then enforced by the 4D stick voting field (section 5.4) to vote for a smooth surface.

Based on the hypersurface saliency information estimated after the above two voting passes, we discard points that are likely to be false matches (section 5.5). Then, methods such as the normalized Eight Point Algorithm, RANSAC, and LMedS, can be used to estimate the fundamental matrix from the filtered set of matches. This enforces the global constraint that the points should lie on a cone in 4D (section 5.6).

In a non-static scene, the estimated matrix can be used to identify the subset of matching points corresponding to a consistent (camera or object) motion. The above process is applied again on the remaining set of unclassified matched points. Thus, multiple and independent motions can be successively extracted.

3 Related Work

A comprehensive survey and recent development on multiple view geometry can be found in [6]. In [11], camera calibration for initially uncalibrated stereo images was proposed. It
also shows that other methods at that time are unstable when the points are close to planar. In the absence of false matches or moving objects, algebraic method can be used for an accurate estimation of fundamental matrix. One algebraic approach was proposed by Sturm [17]. In this classical approach, linear spanning and determinant checking using seven corresponding points are used. If more than seven points are available, the Eight-Point Algorithm [10] is often used. The algorithm estimates the essential (resp. fundamental) matrix from two calibrated (resp. uncalibrated) camera images, by formulating the problem as a system of linear equations. A minimum of eight point matches are needed. If more than eight are available, a least mean square minimization is often used. To make the resulting matrix satisfy the rank two requirement, the singularity enforcement [8] is performed. Its simplicity of implementation offers a major advantage. In Hartley [8], it is noted that after outlier rejection, the Eight-Point Algorithm performs comparably with other non-linear optimization techniques.

False matches are unavoidable in matching programs based on cross correlation. More complicated, non-linear iterative optimization methods were therefore proposed in [25]. These non-linear robust techniques use objective functions, such as the distance between points and their corresponding epipolar lines, or the gradient-weighted epipolar errors, to guide the optimization process. Despite the increased robustness of these methods, non-linear optimization methods require somewhat careful initialization for early convergence to the desired optimum. The most successful algorithm in this class is the LMedS proposed by Zhang et al. [25]. The algorithm uses the least median of squares, data sub-sampling, and certain adapted criterion to discard outliers, by solving a non-linear minimization problem. The fundamental matrix is then estimated.

Torr and Murray [22] proposed RANSAC, which randomly samples a minimum subset with seven pairs of matching points for parameter estimation. The candidate subset that maximizes the number of points and minimizes the residual is the solution. However, it is computationally infeasible to consider all possible subsets, which is exponential in number. Thereby, additional statistical measures are needed to derive the minimum number of sample subsets.

LMedS and RANSAC are considered to be some of the most robust methods. But, it is worth noting that these methods still require a majority (at least 40–50%) of the data be correct, or else some statistical assumption is needed (e.g. the approximate percentage of correct matches needs be known), while our proposed method can tolerate a higher noise to signal ratio. If both false matches and motion exist, these methods may fail, or become less attractive since many matching points due to motion are discarded as outliers.

Another class of robust algorithm is the M-Estimator [7], in which a weighting function is
designed to minimize the effect of outliers. The advantage of this approach is that various weight functions can be designed for different scenarios. But, for motion scenes with a large number of false matches, the weight functions may not be trivial.

In [15], Pritchett and Zisserman proposed the use of local planar homography. Homographies are generated by Gaussian pyramid techniques. Point matches are then generated by using homography. However, the homography assumption does not generally apply to the entire image (e.g. curved surfaces).

In [1], Adam, Rivlin, and Shimshoni addressed the problem of outlier rejection. They discover that with proper rotation of the image points, the correct matching pairs will create line segments pointing in approximately the same direction. However, this method requires some form of searching. Point matches due to motion are rejected.

In [18], the estimation of fundamental matrix is formulated as an 8D hyperplane inference problem. The eight dimensions are defined by the parameter space of the epipolar constraint. Each point match is first converted into a point in the 8D parameter space. Then, 8D tensor voting is applied to “vote” for the most salient hyperplane from the noisy matches. The resulting point matches are assigned a hypersurface saliency value. Matches with low saliency values are labeled as outlier and discarded. Matches with high saliency values are used for parameter estimation. Note that one major difference of this method is that geometric continuity constraint is used, instead of minimizing an algebraic error. While [18] reports good results, a multipass algorithm is proposed. In case of multiple motions, the stability of the system is reduced. The noise rejection ability is somewhat hampered by the non-orthogonality and anisotropy of the 8D parameter space. Also, in [18], motion correspondences are discarded as outliers. A comparison between 4D and 8D tensor voting for epipolar geometry estimation is shown in Table I. We shall

<table>
<thead>
<tr>
<th></th>
<th>4D</th>
<th>8D</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of passes</td>
<td>2</td>
<td>≥ 2</td>
</tr>
<tr>
<td>orthogonal parameter space</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>isotropic parameter space</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>motion pixels</td>
<td>classified</td>
<td>discarded</td>
</tr>
<tr>
<td>outlier noise</td>
<td>discarded</td>
<td>discarded</td>
</tr>
</tbody>
</table>

**TABLE I**

4D AND 8D TENSOR VOTING APPROACH COMPARISON
explain the comparative advantages of 4D tensor voting in this paper.

It is also worth noting that the tensor voting approach has also shown success in perceptual grouping of motion layers [14], and perceptual grouping from motion cues [13].

3.1 Review of joint image and epipolar geometry

Here, we briefly review epipolar geometry and its corresponding 4D cone in the joint image space. More details can be found in [2], [16]. Refer to Fig. 2. Given two images of a static scene taken from two camera systems $C_1$ and $C_2$, let $(u_l, v_l)$ be a point in the first image. Its corresponding point $(u_r, v_r)$ is constrained to lie on the epipolar line derived from $(u_l, v_l)$. This epipolar line is the intersection of two planes: one plane is defined by three points: the two optical centers $C_1$, $C_2$, and $(u_l, v_l)$, and the other plane is the image plane of the second image. Symmetric relation applies to $(u_r, v_r)$. This is known as the epipolar constraint. The fundamental matrix $F = [F_{ij}], 1 \leq i, j \leq 3$, that relates any matching pair $(u_l, v_l)$ and $(u_r, v_r)$ is given by $u_1^T F u_2 = 0$, where $u_1 = (u_l, v_l, 1)^T$ and $u_2 = (u_r, v_r, 1)^T$.

It was suggested by Anandan and Avidan in [2] that in the joint image space, the above equation involving $F_{ij}$ can be written as:

$$\frac{1}{2} \begin{pmatrix} q^T & 1 \end{pmatrix} C \begin{pmatrix} q \\ 1 \end{pmatrix} = 0$$

where $q = (u_l, v_l, u_r, v_r)$ is the stack vector of a corresponding point pair, or joint image coordinates, and $C$ is a $5 \times 5$ matrix defined as

$$C = \begin{bmatrix}
0 & 0 & F_{11} & F_{21} & F_{31} \\
0 & 0 & F_{12} & F_{22} & F_{32} \\
F_{11} & F_{12} & 0 & 0 & F_{13} \\
F_{21} & F_{22} & 0 & 0 & F_{23} \\
F_{31} & F_{32} & F_{33} & F_{32} & 2F_{33}
\end{bmatrix}$$
Note that the 4D joint image space is isotropic and orthogonal. In [2], $C$ is proved to be of rank four. It describes a quadric in the 4D joint image space $(u_l, v_l, u_r, v_r)$. According to [16], a quadric defined by rank four, $5 \times 5$ matrix represents a cone in 4D. Since the 4D cone is a geometric smooth structure except at the apex\(^2\), we use tensor voting to propagate the smoothness constraint in a neighborhood.

### 3.2 Epipolar geometry and motion segmentation

In this section, we review the relationship between epipolar geometry and motion segmentation. The epipolar constraint of a static scene describes the camera motion between two optical centers. Suppose the scene moves, not the camera. We can obtain exactly the same stereo pair, and therefore the same fundamental matrix, if the camera motion and the scene motion are inverse of each other.

Let $< \mathbf{R}_1, t_1^\top >$ be the geometric transformation <rotation, translation> from one camera coordinate system to another. Without loss of generality, assume that the former system is the world coordinate system. Suppose a moving object is seen by both cameras. Let the object move from 3D position $P_1$ to $P_2$. Let the object motion from $P_1$ and $P_2$ be $< \mathbf{R}_2, \ell_2 >$ w.r.t. the world coordinate system. Therefore, if we “compensate” the object motion with the camera motion, the epipolar constraint of the “motion-compensated” object describes a “compensated” camera motion equal to $< \mathbf{R}_2 \mathbf{R}_1, \mathbf{R}_2 t_1^\top + \ell_2 >$. Hence, there are altogether two epipolar constraints to describe the non-static scene: one fundamental matrix describes the static component of the scene due to camera motion $< \mathbf{R}_1, t_1^\top >$, and another fundamental matrix describes the “motion-compensated” object $< \mathbf{R}_2 \mathbf{R}_1, \mathbf{R}_2 t_1^\top + \ell_2 >$. Each epipolar constraint corresponds to a cone in the 4D joint image space. The above can be applied to three or more motions readily, leading to an approach to motion segmentation by using epipolar geometry:

1. Reject wrong matches and reinforce true matches simultaneously, by exactly two passes of 4D tensor voting.
2. Extract the most salient cone in the 4D joint image space, by parameter estimation technique (or simply normalized Eight Point Algorithm with singularity enforcement).
3. Remove point matches whose joint image lies on the 4D cone found above, and repeat step (1).

Fig. 3 shows the results on a synthetic example to further elaborate motion detection by epipolar geometry estimation. Three spheres with independent motion are captured in two frames.

---

\(^2\)The unfortunate case that all matches cluster around the apex is addressed in the discussion section.
Fig. 3. Top: two frames of a moving camera capturing three spheres with independent motion. There are a lot of wrong matches (noise to signal ratio is 1:1). Middle: outliers are discarded, and motion matches are identified and classified. Corresponding epipolar lines of the respective moving objects are drawn. Bottom: hypothetical 4D cones are drawn to illustrate the matching criterion.

by a moving camera. Using 4D tensor voting, we extract consistent motion components of the scene successively (first the largest sphere in the middle, then the sphere on the left, and finally the smallest sphere on the right), classify them into three distinct sets, and reject all outliers. A few corresponding epipolar lines of the respective moving spheres are also drawn. In the result section, we shall state more experimental results on real and synthetic data.
4 Contextual review of tensor voting

In this section, we provide a concise review of tensor voting [12] in the context of 4D inference. Detailed pseudocodes of the 4D voting algorithm are also available in the appendix. In tensor voting, tensor is used for token representation, and voting is used for non-iterative token-token communication. Tensor and voting are related by a voting field. Voting fields are tensor fields, which postulate the most likely directional information, by encoding a geometric smoothness constraint. 4D voting fields can be derived from the 2D stick voting field.

4.1 Voting fields for the basic case

Our goal is to encode the smoothness constraint that should effectively determine if a 4D point lies on some smooth surface in 4D. Let us consider a question related to 2D smoothness: Suppose a point \( P \) in the plane is to be connected by a smooth curve to the origin \( O \). Suppose also that the normal \( \vec{N} \) to the curve at \( O \) is given. What is the most likely normal direction at \( P \)? [12]. Fig. 4a illustrates the situation.

We claim that the osculating circle connecting \( O \) and \( P \) is the most likely connection, since it keeps the curvature constant along the hypothesized circular arc. The most likely normal is given by the normal to the circular arc at \( P \) (thick arrow in Fig. 4a). The length of this normal, which represents the vote strength, is inversely proportional to the arc length \( OP \), and also to the curvature of the underlying circular arc.

To encode proximity and smoothness (low curvature), the decay of the field takes the following form, which is a function of \( r \), \( \varphi \), \( c \), and \( \sigma \):

\[
DF(r, \kappa, \sigma) = e^{-(\frac{\kappa + c \varphi^2}{2\sigma})}
\]  

where \( r \) is the arc length \( OP \), \( \kappa \) is the curvature, \( c \) is a constant which controls the decay with high curvature. \( \sigma \) is the scale of analysis, which determines the effective neighborhood size. Note that \( \sigma \) is the only free parameter in the system.

If we consider all points in the 2D space, the whole set of normals thus derived constitutes the 2D stick voting field, Fig. 4b. Each normal is called a stick vote \([v_x, v_y]\), thus defined as

\[
\begin{bmatrix}
  v_x \\
  v_y
\end{bmatrix} = DF(r, \kappa, \sigma) \begin{bmatrix}
  \sin \varphi \\
  \cos \varphi
\end{bmatrix}
\]

Since we use a Gaussian decay function for \( DF(\cdot) \), the effective neighborhood size is about \( 3\sigma \). Given \( O \) and \( \vec{N} \), \( r \) and \( \kappa \) at \( P \) are known: refer to Fig. 4a. Using sine rule, \( \kappa = \frac{\sin \varphi}{|OP|} \). The arc length \( r = \frac{\kappa}{\pi} \). If \( \kappa = 0 \), \( r = |OP| \).
Given an input token $A$, how to use this field to cast a stick vote to another token $B$ for inferring a smooth connection between them? Let us assume the basic case that $A$’s normal is known, as illustrated in Fig. 4c. First, we fix the scale $\sigma$ to determine the size of the voting field. Then, we align the voting field with $A$’s normal (by translation and rotation). If $B$ is within $A$’s voting field neighborhood, $B$ receives a stick vote from the aligned field.

4.2 Vote collection and tensor representation

How does $B$ collect and interpret all received votes? Other input tokens cast votes to $B$ as well. Denote a received stick vote by $[v_x, v_y]^T$. To collect the majority vote, one alternative is to accumulate the vector sum of the the stick votes. However, since we only have sparse data (especially in the 4D joint image space), orientation information may be unavailable or wrong. Two consistent directions with different orientation will cancel out each other if we collect the vector sum, or first order tensor.

We therefore collect second order moments. The tensor sum of all votes collected by $B$ are accumulated, by summing up the covariance matrices consisting of the votes' second order
moments: \[ \begin{bmatrix} \sum v_x^2 & \sum v_x v_y \\ \sum v_y v_x & \sum v_y^2 \end{bmatrix} \]. This is a second order symmetric tensor. By decomposing this tensor into the corresponding eigensystem, we obtain the most likely normal at $B$, given by the eigenvector associated with the largest eigenvalue. Let us denote this unit vector by $\hat{e}_1$.

Geometrically, a second order symmetric tensor in 2D is equivalent to an ellipse. The major axis gives the general direction. The minor axis indicates the uncertainty: if the length of the minor axis is zero, the tensor is a stick tensor, representing absolute certainty in one direction given by the major axis. If the length of the minor axis is equal to that of the major axis, the tensor is a ball tensor, indicating absolute uncertainty in all directions.

At first glance, the use of tensor fields in vote collection resembles to that of radial basis function for scattered data interpolation, where a set of functions are linearly combined to fit the data. However, the tensor fields used here have fundamental difference. First, our output is a tensor, instead of a scalar. The accumulated tensor captures the statistical distribution of directions by its shape after voting, while a scalar cannot. Moreover, tensor voting is capable of performing extrapolation (inference) as well as interpolation. More importantly, geometric salience information is encoded in the resulting tensor, which is absent in any interpolation scheme. We will address feature saliency shortly in the vote interpretation section.

4.3 Voting fields for the general case

Now, consider the general case that no normal is available at $A$. We want to reduce this case to the basic case, so we need to estimate the normal at $A$. Without any a priori assumption, all directions are equally likely as the normal direction at $A$. Hence, we rotate the 2D stick voting field at $A$. During the rotation, it casts a large number of stick votes to a given point $B$. All stick votes received at $B$ are converted into second order moments, and the tensor sum is accumulated. This is exactly the same as casting stick votes as described in the previous sections.

Then, we compute the eigensystem of the resulting tensor to estimate the most likely normal at $B$, given by the direction of the major axis of the resulting tensor inferred at $B$, or $\hat{e}_1$.

Alternatively, for implementation efficiency, instead of computing the tensor sum on-the-fly at a given vote receiver $B$, we precompute and store tensor sums due to a rotating stick voting field received at each quantized vote receiver within a neighborhood. We call the resulting field a 2D ball voting field, which casts ball votes in $A$’s neighborhood. Fig. 4d shows the ball voting field, which stores the eigensystem of the tensor sum at each point. Note the presence of two eigenvectors at each site in Fig. 4d.
4.4 Vote interpretation

In 4D, we can define similar stick and ball voting fields. After collecting the second order moments of the received votes, they are summed up to produce a 4D second order symmetric tensor, which can be visualized as a 4D ellipsoid, represented by the corresponding eigensystem

$$\sum_{i=1}^{4} \lambda_i \hat{e}_i \hat{e}_i^T$$

where $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4 \geq 0$ are eigenvalues, and $\hat{e}_1$, $\hat{e}_2$, $\hat{e}_3$, and $\hat{e}_4$ are corresponding eigenvectors. The eigenvectors determine the orientation of the 4D ellipsoid. The eigenvalues determine the shape of the 4D ellipsoid.

Consider any point in the 4D space. It is either on a smooth structure, at a discontinuity or an outlier. If it is a hypersurface point, the stick votes received in its neighborhood reinforce each other, indicating a high agreement of tensor votes. The inferred tensor should be stick-like, that is, $\lambda_1 \gg \lambda_2, \lambda_3, \lambda_4$, indicating certainty in a single direction. On the other hand, an outlier receives a few inconsistent votes, so all eigenvalues are small. We can thus define surface saliencies by $\lambda_1 - \lambda_2$, with $\hat{e}_1$ indicating the normal direction to the hypersurface. Moreover, if it is a discontinuity or a point junction where several surfaces intersect exactly at a single point (e.g. the apex of a 4D cone), it indicates a high disagreement of tensor votes, indicating no single direction is preferred. Junction saliency is indicated by high values of $\lambda_4$ (and thus other eigenvalues). Outlier noise is characterized by low vote saliency and low vote agreement.

5 Motion segmentation and outlier rejection by 4D tensor voting

We use 4D tensor voting to perform motion segmentation and outlier rejection. The input to our algorithm consists of a set of potential point matches. False correspondences may be present. In a non-static scene where multiple motions exist, point matches that correspond to a single consistent motion should cluster and lie on a 4D cone in the joint image space. To reject false matches while retaining the matching points contributed by the background and/or salient motion, we make use of the property that outliers do not lie on any of these cones.

5.1 From point matches to joint image coordinates

Before we encode the data points into a 4D tensor (next section), we first normalize the 2D point matches. Since the matching points found by the correlation methods may be shifted to quantized positions, this data normalization step brings added stability to later steps.
For each potential matching point pair \((u_l, v_l), (u_r, v_r)\) in images 1 and 2 respectively, we normalize them to \((u'_l, v'_l), (u'_r, v'_r)\) by translation and scaling [8]: center the image data at the origin, and scale the data so the mean distance between the origin and the data points is \(\sqrt{2}\) pixels. The 4D joint image coordinates are then formed by stacking the normalized 2D corresponding points together: \((u'_l, v'_l, u'_r, v'_r)\).

5.2 Tensor encoding

The joint image coordinates are points in the 4D space. They are first encoded into a 4D ball tensor to indicate no orientation preference, since initially they do not have any orientation information. Geometrically, a 4D ball tensor can be thought as a 4D hypersphere which does not have preference in any direction. Mathematically, it can be represented by the equivalent eigensystem having four equal eigenvalues, and four orthonormal unit vectors:

\[
\begin{align*}
\sum_{i=1}^{4} \lambda_i \hat{e}_i \hat{e}_i^T &= \sum_{i=1}^{3} (\lambda_i - \lambda_{i+1}) \sum_{j=1}^{i} \hat{e}_j \hat{e}_j^T + \lambda_4 \sum_{j=1}^{4} \hat{e}_j \hat{e}_j^T \\
&= \lambda_4 \sum_{j=1}^{4} \hat{e}_j \hat{e}_j^T \\
&= \lambda_4 B
\end{align*}
\]  

if all \(\lambda_i\)'s are equal. \(B = \sum_{j=1}^{4} \hat{e}_j \hat{e}_j^T\) is the 4D ball tensor.

In our system, we simply set \(\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 1\). Hence, the first term in Equation (6) is zero to give a ball tensor. We use the world coordinate system to initialize \(\hat{e}_1, \hat{e}_2, \hat{e}_3,\) and \(\hat{e}_4\). So \(\hat{e}_1 = [1 0 0 0]^T, \hat{e}_2 = [0 1 0 0]^T,\) etc. Any four orthonormal vectors in 4D can do the job.

Thus, the joint image coordinates indicate the position of the encoded 4D tensor, whose shape will then be changed after the first voting pass, where a new tensor is produced at that location. This new tensor is no longer an isotropic hypersphere, but possesses orientation.

After the first pass voting with 4D ball voting field, the accumulated tensor is decomposed into the corresponding eigensystem. The \(\hat{e}_1\) component of the decomposed tensor will be pointing along the normal to the surface (section 5.3).

In the second pass, 4D stick voting field will be aligned with the normal to propagate the smoothness constraint (section 5.4), reinforcing true data points lying on a smooth (conical) surface, while suppressing false data points not lying on any smooth structure. Reinforcement and suppression are expressed in terms of the hypersurface saliencies \((\lambda_1 - \lambda_2)\), after decomposing the accumulated tensor obtained in the second pass voting.
5.3 Normal estimation by the ball voting field

GENTENSORVOTE in Algorithm 2 (appendix) is called here, where voter and votee are the encoded 4D ball tensors. This is the first of the two tensor voting passes. After the input has been encoded as a set of ball tensors, they communicate with each other by our voting fields to estimate the most likely normal at each 4D point.

Communication among all input ball tensors is achieved by the 4D ball voting field \( V_B(P) \) applied at a 4D point \( P \). \( V_B(P) \) is obtained by rotating and integrating vote contributions of the 4D stick voting field \( V_S(P) \) (defined shortly). The 4D ball voting field has the form

\[
V_B(P) = \int_0^\pi \int_0^\pi \int_0^\pi R_{\theta_1\theta_2\theta_3\theta_4} V_S(R_{\theta_1\theta_2\theta_3\theta_4}^{-1} P) R_{\theta_1\theta_2\theta_3\theta_4}^T d\theta_1 d\theta_2 d\theta_3 d\theta_4 |\theta_4=0
\]

where \( \theta_1, \theta_2, \theta_3, \theta_4 \) are rotation angles in \( w, z, y, x \) axis respectively, and \( R_{\theta_1\theta_2\theta_3\theta_4} \) is the rotation matrix that aligns \( V_S(P) \).

Like the 2D case, let a 4D point \( P_1 \) communicate with another 4D point \( P_2 \), by using the ball voting field, that is, by aligning \( V_B \) with \( P_1 \). Since \( V_B \) has non-zero volume, if \( P_2 \) is contained in \( V_B \)'s neighborhood, then \( P_2 \) is said to receive a ball tensor vote from \( P_1 \). Or, equivalently, \( P_1 \) has cast a ball vote to \( P_2 \). \( P_2 \) receives ball tensor votes cast by all other \( P_1 \)'s, and sums them up using tensor addition. The resulting symmetric tensor is decomposed into the corresponding eigenvalues and eigenvectors. The hypersurface saliency, which indicates the likelihood of \( P_2 \) lying on a (conical) surface, is given by \( \lambda_1 - \lambda_2 \). The normal to the surface in 4D is given by \( \hat{e}_1 \), which is the preferred direction given by the majority vote.

5.4 Enforcement of smoothness constraint by the stick voting field

GENTENSORVOTE in Algorithm 2 is called again, where voter and votee are 4D tensors obtained in the first pass. Let us first describe \( V_S \), the 4D stick voting field. The design rationale of the 4D stick voting field is the same as the 2D counterpart: to encode the proximity and smoothness (constancy of curvature) constraints, but now in 4D. By following the same arguments, the question we pose and the answer are the same as that of section 4, which are illustrated in Fig. 4, except now in 4D.

The 4D stick voting field \( V_S \) can in fact be derived from the 2D stick voting field: starting from the 2D stick voting field, denoted by \( S_{2D} \), we first rotate \( S_{2D} \) by 90°. Denote the rotated field by \( R_{\pi}S_{2D} \). Put the field in the 4D space, at the origin and aligned with \( [1 \ 0 \ 0 \ 0]^T \). Then,

\[\text{ADD TENSOR in the appendix.}\]

\[\text{The rotation can be clockwise or counterclockwise, since } S_{2D} \text{ is symmetric.}\]
rotate the field about the $x$ axis, which produces a sweeping volume in the 4D space. Thus, $V_S$ is the 4D stick voting field, which describes the direction along $[1 \ 0 \ 0 \ 0]^T$ and postulates the normal directions in a neighborhood. The size of the neighborhood is determined by the size of the voting field, or the scale $\sigma$ in Equation (3).

Recall that a 4D normal is obtained at each point after the first voting pass. The second pass is used to propagate the smoothness constraint, so as to reinforce true data points via smoothness propagation, and to suppress noisy points that are not lying on any smooth structures. We align the 4D stick voting field $V_S(P)$ with the normal (given by the $\hat{e}_1$ component of the inferred tensor) estimated at point $P$ to vote for smooth surface. The stick voting field $V_S(P)$ propagates the local continuity or smoothness constraint in a neighborhood. Votes are received and summed up using tensor addition.

Let a point receive a stick vote $[n_1 \ n_2 \ n_3 \ n_4]^T$. The covariance matrix consisting of the vote’s second order moments is calculated, which is a $4 \times 4$ symmetric matrix. This symmetric matrix is equivalent to a second order symmetric tensor in 4D. The given point accumulates this matrix by tensor sum, which essentially adds up the matrices. Vote interpretation by decomposing this matrix into the corresponding eigensystem follows.

Since false matching points do not form any smooth features with other true matching points, the resulting directional votes do not agree in orientation, and thus will not produce a consistent normal direction in the majority vote. Therefore, these points will have low hypersurface saliency. On the contrary, correct matching points reinforce each other to produce a high hypersurface saliency.

### 5.5 Rejection of wrong matches

Recall that after collecting and accumulating the tensor sum, the accumulated result is still a second order symmetric tensor in 4D. We compute the corresponding eigensystem $\sum_{i=1}^{4} \lambda_i \hat{e}_i \hat{e}_i^T = (\lambda_1 - \lambda_2)\hat{e}_1 \hat{e}_1^T + (\lambda_2 - \lambda_3)(\hat{e}_1 \hat{e}_1^T + \hat{e}_2 \hat{e}_2^T) + (\lambda_3 - \lambda_4)(\hat{e}_1 \hat{e}_1^T + \hat{e}_2 \hat{e}_2^T + \hat{e}_3 \hat{e}_3^T) + \lambda_4(\hat{e}_1 \hat{e}_1^T + \hat{e}_2 \hat{e}_2^T + \hat{e}_3 \hat{e}_3^T + \hat{e}_4 \hat{e}_4^T)$. True matching points will have a high hypersurface saliency, $\lambda_1 - \lambda_2$. The saliency value indicates if a point lies on some smooth structure such as a cone.

After this step, for high hypersurface saliency points, we still do not know which point cone the corresponding matching point belongs to. On the other hand, low hypersurface saliency value indicates that the point does not lie on any local smooth structure, and hence not on any cone. So, this point should be discarded as a false match.

In [18], we use the extrema detection to discard false matches. In this paper, we adopt a simpler
and faster technique, which classifies a point match as correct if the associated hypersurface saliency is above the mean hypersurface saliency value. We find the enforcement of the global constraint of fitting a cone is more efficient and effective for the purpose of parameter estimation. The thresholded set of good matches are used for fundamental matrix estimation.

5.6 Parameter estimation and motion extraction

Now we have a set of good matches. These good matches satisfy the local continuity constraint (i.e., a high likelihood of lying on a smooth structure). We want to enforce a global, point-cone constraint on this set of filtered matches. If there exist multiple and independent motions, the matching points should correspond to distinct epipolar geometries alongside with their respective fundamental matrices. To estimate the fundamental matrix that corresponds to the most salient motion (or motionless background), we apply RANSAC or the normalized linear method, either of which is sufficient since we now have a set of good matches.

We use a 2D illustration in Fig. 5 to explain the successive extraction of motion components. Let $S_i$ be the set of point matches after subtracting the $i$-th set of motion matches, Fig. 5a, and $S_0$ is the input point matches we obtained after tensor encoding (section 5.2). After 4D tensor voting, a filtered set of point matches $T_i$ is obtained, Fig. 5b. Note that $T_i$ may contain more
than one motion components. RANSAC or other techniques can be performed on \( T_i \) to estimate the dominating fundamental matrix, Fig. 5c. Let \( R_i \subseteq S_i \) be the maximal subset of matches that produces the minimal residual, according to the estimated fundamental matrix. We use the normalized Eight-Point Algorithm (eigen-analysis) with singularity enforcement to estimate the dominating fundamental matrix. Then, we apply the above parameter estimation to \( S_i - R_i \) to successively extract the next dominating fundamental matrix. The above process is repeated until \( T_i = R_i \), or, only one motion component is found by RANSAC in the inliers returned by 4D tensor voting.

6 Time complexity analysis

Let \( N \) be the total number of input matches, and \( k \approx 3\sigma \) be the size of the voting field used in tensor voting. \( k \) is often chosen to be half of the image dimension. Our experiments show that the results are not sensitive to \( k \). Note that we do not store an entire 4D voting field, which consumes storage. Instead, due to symmetry, we use a 1D array to store the ball voting field, and a 2D array to store the stick voting field, by making use of Algorithm 2 to precompute and store the votes.

Data conversion (section 5.1) and tensor encoding (section 5.2) take \( O(N) \) time respectively. The two passes of tensor voting (sections 5.3 and 5.4) take \( O(kN) \) time each. Outlier rejection and motion segmentation (section 5.5) takes \( O(N) \) time. Parameter estimation (section 5.6) takes \( O(K) \) time, where \( K \) is the total number of subsets we use in RANSAC. Therefore, if there are \( n \) motions, the total time complexity is \( O(n(kN + K)) \), which is essentially linear in \( N \). Note that for any \( n \), \( N \) only need to satisfy the minimum number of points necessary to define a fundamental matrix, that is, eight, which is independent of \( n \) (c.f. [23]).

In practice, for epipolar geometry estimation, we set \( K = 1000 \). A complete run takes only a few seconds. Since we already have a set of good matches by the time we run RANSAC, we do not need all possible combinations to find the maximal subset, which is exponential in number. Currently, the size of each subset used is 10 to 15. The estimation stability increases with the size of the subset, but up to a certain point. For instance, if the size of the filtered set of good matches is small, then, we need to use smaller subsets.

7 Results

We have performed extensive experiments to evaluate our 4D tensor voting approach. They are categorized in the followings:
7.1 Evaluation on motion segmentation (synthetic data)

First, we evaluate our system by using fourteen synthetic non-static scenes to exhaust all camera and object transformations (rotation and translation), as tabulated in Table II, where ground truths are available for direct verification. The objects are of different sizes. To evaluate noise robustness, five different noise-to-signal levels are tested. They are respectively 1, 1.5, 2, 2.5, 3. That is, there are three wrong correspondences for each correct pair of match, for instance, when the noise-to-signal ratio equals to 3. A total of 70 (= 14 × 5) experiments is run to enumerate all noise scenarios.

Each scene consists of three independent motions. The types of camera motion are also shown in Table II. Fig. 6 shows the two views of the moving objects for each case. For clarity of display, the noisy correspondences are not shown.

Three experimental conditions are worth noting here. First, since RANSAC is used in our parameter estimation, we run each RANSAC with different number of random subsets (iterations) to evaluate its effect on our results. In general, the results improve with increasing number of subsets. Second, results without noise rejection by 4D tensor voting (we call it RANSAC ONLY) is compared to justify the significance of our method (we call it 4DRANSAC). Since random permutations are used to estimate the epipolar geometries, in each experiment, we run our system 20 times, and produce the mean performance on motion pixel segmentation. Third, in order to provide a fair comparison basis for successive motion segmentation among all methods, the input is always updated by \( A \leftarrow A - \text{result}(4\text{DRANSAC}) \), not by \( A \leftarrow A - \text{result(METHOD)} \). Algorithm 1 summarizes the rundown of our experiment.

True (resp. false) positives, as defined in Algorithm 1, indicate the number of correctly (resp. incorrectly) labeled motion correspondences. True (resp. false) negatives, which indicate the performance on classifying noisy matches, can be derived similarly\(^7\).

In Algorithm 1, the METHODS we implemented and tested are: 4D ONLY, RANSAC ONLY, and our 4DRANSAC consisting of outlier rejection by 4D tensor voting followed by RANSAC estimation of epipolar geometries.

To facilitate analysis of our extensive experiments, the results of all 70 experiments are further categorized by object motions:

(a) Pure translation (scenes (1), (6)), Fig. 7a, Fig. 8a, and Fig. 9a.

\(^7\) TN \( \leftarrow \) count((input − result(METHOD)) \( \cap \) (input − correct)), FN \( \leftarrow \) count((input − result(METHOD)) \( \cap \) correct)
Algorithm 1 EVALUATESEGMENTATIONPERFORMANCE (input, correct, METHOD)

This function computes $TP$ and $FP$ (true positives and false positives, defined below) which indicate the performance of motion segmentation.

Input:

$\text{input} = \text{the set of correct matches plus noise.}$
$\text{correct} = \begin{cases} \text{allgt} & \text{if METHOD = 4D ONLY} \\ \text{gt}[\alpha] & \text{otherwise} \end{cases}$
allgt = the set of all correct matches (ground truths)
gt[\alpha] = correct matches for motion $M_\alpha$
METHOD = \{ 4D ONLY | RANSAC ONLY | 4DRANSAC \}

Output:

\text{result(METHOD) = inliers classified by METHOD}

\begin{algorithmic}
\FOR {each noise-to-signal ratio = 1, 1.5, 2, 2.5, 3}
\FOR {number of random subsets (iterations) = 1000, 5000, 10000}
\FOR {$i = 1$ to 20}
\STATE input$_i \gets \text{input}$
\FOR {each pass of motion extraction $\alpha$}
\FOR {each METHOD}
\STATE result(METHOD) $\gets$ METHOD(input$_i$)
\STATE $TP_{\alpha i \text{METHOD}} \gets \text{count(result(METHOD) \cap correct)}$
\STATE $FP_{\alpha i \text{METHOD}} \gets \text{count(result(METHOD) \cap (input$_i$ - correct))}$
\ENDFOR
\STATE input$_{i+1} \gets$ input$_i$ - result(4DRANSAC)
\ENDFOR
\ENDFOR
\FOR {each extracted motion $M_\alpha$ and METHOD}
\STATE $TP_{\alpha \text{METHOD}} \gets \sum_{i=1}^{20} TP_{\alpha i \text{METHOD}} / 20$
\STATE $FP_{\alpha \text{METHOD}} \gets \sum_{i=1}^{20} FP_{\alpha i \text{METHOD}} / 20$
\STATE output $TP_{\alpha \text{METHOD}}$ and $FP_{\alpha \text{METHOD}}$ at the current noise-to-signal ratio and number of random subsets
\ENDFOR
\ENDFOR
\ENDFOR
\end{algorithmic}
Fig. 6. Non-static synthetic scenes.

(b) Pure rotation (scenes (2), (4), (8), (10), (13)), Fig. 7b, Fig. 8b, and Fig. 9b.

c) Translation and rotation (scenes (3), (5), (7), (9), (11), (12)), Fig. 7c, Fig. 8c, and Fig. 9c.

The plots for 4D ONLY are interpreted as follows. In Fig. 7a, the plots for true positives or $TP$ (resp. false positives or $FP$) correspond to the mean $TP$ (resp. $FP$) for scenes (1) and (6) after running 4D ONLY. In Fig. 7b, the $TP$ (resp. $FP$) plots correspond to the mean $TP$ (resp. $FP$) after running 4D ONLY on scenes (2), (4), (8), (10), and (13). Fig. 7c shows the averaged result on segmenting the three motions for scenes (3), (5), (7), (9), (11), and (12). As there are three
TABLE II

NON-STATIC SYNTHETIC SCENES. EACH SCENE CONSISTS OF THREE INDEPENDENT MOTIONS. DIFFERENT NOISE-TO-SIGNAL RATIOS (1, 1.5, 2, 2.5, 3) WERE TESTED.

<table>
<thead>
<tr>
<th>Number of objects</th>
<th>Object motion</th>
<th>Camera motion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>translation</td>
<td>translation</td>
</tr>
<tr>
<td></td>
<td>rotation</td>
<td>rotation</td>
</tr>
<tr>
<td>(1) 3 0 0</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>(2) 3 0 0</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>(3) 3 0 0</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>(4) 0 0 3</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>(5) 0 0 3</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>(6) 0 0 3</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>(7) 3 0 0</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>(8) 3 0 0</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>(9) 3 0 0</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>(10) 0 0 3</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>(11) 0 0 3</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>(12) 2 1 0</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>(13) 1 1 1</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>(14) 1 1 1</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

motions ($M_1$, $M_2$, and $M_3$), three measures on $TP$ and $FP$ are reported. Plots for RANSAC ONLY and 4DRANSAC are interpreted similarly.

In each 3D plot, the three respective axes denote the noise level (in %), the number of RANSAC iterations (random subsets) used to estimate the fundamental matrix, and the $TP$/$FP$ percentages. Note that the $TP$/$FP$ percentages are defined differently for 4D ONLY and the other two methods: since 4D ONLY (not 4DRANSAC) is primarily used for noise rejection (no motion classification is done, which is performed in the parameter estimation step that follows, therefore allgt is used instead of $gt(\alpha)$). For each motion analysis $M_\alpha$, $1 \leq \alpha \leq 3$, the $TP$ percentage is defined as $TP_{\text{count(input,allgt)}}/100\%$. The $FP$ percentage is defined as $FP_{\text{count(input,allgt)}}/100\%$. That is, the inliers identified by 4D ONLY are not classified into the object motion to which they belong. In RANSAC ONLY and our 4DRANSAC, the $TP$ percentage for motion $M_\alpha$, $1 \leq \alpha \leq 3$, is defined as $TP_{\text{count(input,gt(\alpha))}}/100\%$. The $FP$ percentage for motion $M_\alpha$, $1 \leq \alpha \leq 3$, is defined as $FP_{\text{count(input,gt(\alpha))}}/100\%$. Given the above details, the following sentence summarizes well how to interpret the plots in Fig. 7–9: good performance is indicated by high values of $TP$ and low values of $FP$.

We have the following conclusions from our experiments:

1. RANSAC ONLY fails in identifying true positives in all the 70 experiments.
2. 4D RANSAC is very robust to incorrect matches.
Fig. 7. Performance on outlier rejection by 4D ONLY: (a) pure translation (b) pure rotation. (c) translation and rotation. Denote true positives by TP, and false positives by FP. Three motions $M_\alpha, 1 \leq \alpha \leq 3$ are segmented. The vertical axis denotes the TP/FP percentages, the left axis indicates the number of RANSAC iterations (random subsets) used to estimate the fundamental matrix, and the right axis correspond to the noise level (in %).

3. The segmentation results improve with increasing number of random subsets (iterations).

In practice, we can use fewer iterations for acceptable epipolar geometry estimation, except for the following scenario, which is further addressed in the limitation section (section 8.5).

4. For scenes (1) and (6) which involves pure object translations, the performance (e.g. the plot for $TP M_3$) for 4DRANSAC is not satisfactory.

Readers may demand a comparison of 4DRANSAC with other representative methods such as LMedS and M-Estimator, which are addressed in the next section. However, it is worth noting that the strength of 4DRANSAC is its noise robustness and its ability to segment motion pixels successively. The LMedS uses median as threshold. In noisy correspondences, the median assumption may not work well. The M-Estimator can extract the most salient motion/background. Given multiple motions, it is somewhat difficult to adjust the thresholds to segment them.
7.2 Evaluation of epipolar geometry estimation (real/synthetic data)

Refer to the quantitative results in Table III. Fig. 3 shows the results on a synthetic image pair of a non-static scene, THREE SPHERES, captured by a moving camera. Random, wrong matches are added to the set of input point matches. The number of wrong matches we added is equal to the number of true matching points, thus the noise to signal ratio is 1. This ratio will be increased up to 5.788 as we subtract matches corresponding to salient motion in subsequent passes. For example, after the first salient motion is extracted, we subtract 109 matches from the input, so leaving 83 correct data points in the second pass. The first two rows of figures in the table for FAN, UMBRELLA, CAR and TOYS are interpreted similarly.

When THREE SPHERES have been processed, wrong matches (blue crosses) are classified and discarded. The three salient motions (inliers) are identified. In Fig. 3, the epipolar lines and correspondences of the three extracted motions are colored red, green, and orange, respectively. Note that, in each pass, we can label all outliers (but one in the last run) correctly.
inliers are correctly classified. Three real examples are shown in Fig. 12. The FAN scene shown in Fig. 12a is an indoor scene. An electric fan is rotating about an axis. Note that we can identify the moving electric fan and the static scene behind it. In Fig. 12b, the UMBRELLA scene shows a walking man holding an umbrella. Our system can discard outliers, identify motion and background. Fig. 12c shows the CAR scene. In Fig. 13, the TOYS scene shows a forward camera motion and two toy cars moving in different directions. We can extract the forward camera motion and segment the two additional motions despite the large amount of noise we added. Our 4D system can reject outliers, retain motion matches, and produce the corresponding epipolar geometries for the non-static scene. In all examples, the camera as well as some objects in the scene are in motion. Note that the normalized Eight Point Algorithm and robust methods in [25] fail on all our noisy sets of matches.

*In practice, the correspondence establishment can be performed by automatic cross-correlation or manual picking (section 2) which produces less noisy point matches used in our experiments. Here, we test our approach to the extremes, in the presence of a large amount of noises.*
### Table III

<table>
<thead>
<tr>
<th></th>
<th>THREE SPHERES</th>
<th>FAN</th>
<th>UMBRELLA</th>
<th>CAR</th>
<th>TOYS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>most salient</td>
<td>2nd salient</td>
<td>3rd salient</td>
<td>most salient</td>
<td>2nd salient</td>
</tr>
<tr>
<td>No. of correct data points $S_j$</td>
<td>192</td>
<td>83</td>
<td>33</td>
<td>111</td>
<td>50</td>
</tr>
<tr>
<td>No. of incorrect data points</td>
<td>192</td>
<td>192</td>
<td>191</td>
<td>151</td>
<td>150</td>
</tr>
<tr>
<td>noise/signal ratio</td>
<td>1.000</td>
<td>2.313</td>
<td>5.788</td>
<td>1.360</td>
<td>5.000</td>
</tr>
</tbody>
</table>

#### (A) Results on 4D Tensor Voting

<table>
<thead>
<tr>
<th></th>
<th>No. of correct inliers $T_j$</th>
<th>No. of incorrect inliers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>150</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

#### (B) Results on Parameter Estimation

<table>
<thead>
<tr>
<th></th>
<th>No. of correct inliers $R_j$</th>
<th>No. of incorrect inliers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>109</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Scale used in 4D analysis $\sigma$</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>No. of random trials</td>
<td>10000</td>
<td>10000</td>
</tr>
<tr>
<td>No. of points in each subset</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

#### TABLE III

FOUR EXPERIMENTAL RESULTS ON NON-STATIC SCENES.

### 8 Discussion

In this section, we further compare our algorithm with representative algorithms of epipolar geometry estimation. Then, we state issues and limitations of our method.

#### 8.1 Comparison with representative algorithms

Our evaluation uses static and non-static scenes. For static scenes, we used 7 real data sets with noise provided in the Matlab toolkit written by Armangué [3][9], and generated the fundamental matrix using both our method and other algorithms provided in the toolkit. The average pixel distances between the matching points and the estimated epipolar lines are plotted in Fig. 10. Armangué’s toolkit provides the implementation of about 15 epipolar geometry estimation methods. Here, owing to space limitation, we report the results of five methods that give best performances. Readers can refer to a technical report [19] for results on other methods. As ground truth data for the non-static scenes are difficult to obtain by automatic means, we hand-picked good matches and added random false matches in our test cases. Then, the algorithms implemented in the toolkit and our algorithm are compared. The results are plotted in Fig. 11.

It is clear that our method produces reasonably better results than other algorithms for the non-static scenes we experimented. For static scenes, Torr’s M-estimator [22] performs slightly better than our method. This is understandable, as the false match filtering by 4D Tensor Voting is more tailored for non-static scenes, in which only local continuity constraint is maintained.

9http://eia.udg.es/~armangue/research/
As for static scenes, we can also enforce the global constraint, but that will require changing the scale of analysis in tensor voting. Since our aim is to have no free parameters for the whole system, we sacrifice the slight inaccuracy to complete automation.

### 8.2 Comparison with the 8D approach

Note that the advantages of 8D voting [18] are still inherited in this 4D formulation: non-iterative processing and robustness to a considerable amount of outlier noise. An additional advantage of dimensionality reduction is brought by the joint image space, which is an isotropic one parameterized by \((u_l, v_l, u_r, v_r)\). In [18], the 8D space is parameterized by \((u_l u_r, v_l u_r, u_r, u_l v_r, v_l v_r, v_r, u_l, v_l)\), which is neither isotropic nor orthogonal, nor independent. Hence, in the 8D case, in order to satisfy the isotropy assumption of tensor voting, we have to scale (for voting) and re-scale the parameter space (for computing the fundamental matrix). Therefore, some precision is lost as a result of this scaling and re-scaling operations. Multiple passes (more than two) are needed in practice to improve the accuracy. Now, in 4D, we only need two passes in all the experiments. Note further that in 8D, we need at least eight points to fix an 8D normal, while only four are sufficient to fix a 4D normal.
8.3 Tensor voting versus generalized Hough transform

As shown in section 6, the time and space complexities of 4D tensor voting are independent of dimensionality, unlike that of Hough transform. Our method is more efficient than Hough transform in high dimensions. Although tensor voting and Hough transform share the same idea that a voting scheme is employed, the computational frameworks are very different. In Hough transform [4], the voting occurs in high dimensional space, where each quantized location receives a number of votes cast from voters. The maximal support is given by the point receiving the maximum (scalar) counts. In 4D tensor voting used in this paper, the voting occurs in the high dimensional space, where only the points corresponding to the input joint images receive tensor votes from voters, efficiently cast by aligning voting fields, which are 1D or 2D arrays in implementation. Moreover, no quantization is necessary.

8.4 Estimating $n$ matrices versus a single multibody matrix

In terms of epipolar geometry estimation and motion detection, 4D tensor voting adopts a geometric approach, by propagating strong geometric smoothness constraint in the 4D joint image space for detecting smooth 4D structures. This novel approach is effective: when false matches and multiple motion matches co-exist, the problem of multiple motion segmentation is particularly challenging. This segmentation problem is complicated by the fact that matches belonging to a less salient motion may be mis-classified as outliers with respect to more salient motion. Our method is inspired by robust structure inference in 2D, and surface fitting 3D (for comparison of related work, see [12]). The 2D and 3D tensor voting reject outliers (not lying on any smooth objects) and extract multiple objects in succession, without any genus or topological limitations. In particular, the most salient object is segmented first, followed by less salient ones.

The 4D version considers joint image coordinates. Matches corresponding to independent multiple motions are manifested into multiple point cones in the 4D joint image space. False match is translated into an outlier in this 4D space, where the smoothness constraint is violated. Our method adopts a segmentation and estimation approach, which can extract salient motions in succession: first, the most salient “motion” (e.g. the background), followed by the second salient motion (e.g. the largest moving object in the scene), so on.

This approach is different from [24] and [23], in which the two-view two-body (resp. multibody) motion extraction and epipolar estimation are performed. The (multibody) fundamental matrix is estimated, which encapsulates all the fundamental matrices corresponding to motion. Individual matrices are extracted from this composite matrix. The saliency of each motion is
unknown when its fundamental matrix is extracted. A further step may be needed to understand the saliency of the extracted motion.

8.5 Limitations and future work

Although our approach performs better than some representative algorithms on our sample non-static scenes, 4D tensor voting on epipolar geometry estimation for non-static scenes has its own limitations.

We use epipolar geometry to extract motion components successively. One limitation is due to the inherent ambiguity that the epipolar constraint maps a point to a line. In certain cases, wrong matches can coincidentally lie on corresponding epipolar lines. Alternatively, if false matches happen to be consistent with a parametric model returned by RANSAC, they can be misclassified as a consistent motion instead of outliers. Our approach enforces the local smoothness constraint in the joint image space by 4D tensor voting. However, if false matches happen to form some smooth surface in the joint image space, our voting system either cannot detect them, or cannot reject them in the following parameter estimation step. Also, if two motion components have very similar epipolar geometry (e.g. their motions are relatively parallel to each other with respect to the viewpoint), our approach may fail to segment them into two distinct components. One single motion will be returned, which needs further processing to distinguish the motions.

If unfortunately all motion matches happen to cluster around the apex of the corresponding point cone, we need to detect the apex, instead of smooth surfaces. As shown in previous work [12] and section 4, regardless of dimensionality, tensor voting is well suited to detect smooth manifolds as well as point junctions, which are characterized by high disagreement of tensor votes. This difficult case does not pose any theoretical difficulty to our approach. If a high disagreement of tensor votes in the 4D space is detected in the close vicinity of a 4D location (indicated by large $\lambda_1$), the corresponding point matches should be segmented.

9 Conclusion

We propose a specialization of the tensor voting algorithm in 4D, and describe a novel, efficient, and effective method for robust estimation of epipolar geometry estimation and motion segmentation for non-static scenes. The key idea is the rejection of outliers which do not support each other, whereas inliers (correct matches), even if they are small in number, reinforce

However, in practice, we found that the corresponding motion is not very salient if such coincidental smooth structure occurs.
each other as they belong to the same surface (cone). This idea is translated into a voting-based computational framework, the main technical contribution of this paper. A geometric approach is adopted, in which salient motion are extracted in succession. By reducing the dimensionality, we solve the problems in [18], such as the non-orthogonality and anisotropy of the parameter space. Motion correspondences are classified into distinct sets, by extracting their underlying epipolar geometries. Only two passes of tensor voting are needed. It is shown that the new approach can tolerate a larger amount of outlier noise. Upon proper segmentation and outlier rejection, parameter estimation techniques such as RANSAC, LMedS, or even the normalized Eight Point Algorithm can be used to extract motion components and epipolar geometry for non-static scenes. Besides the issues to be addressed in section 8.5, in the future, we propose to perform reconstruction into a set of motion layers, based on the extracted epipolar geometries.

Acknowledgment

We are indebted to the Associate Editor and all anonymous reviewers for their very constructive comments and thoughtful discussion with the authors throughout the review process. This research is supported by the Research Grant Council of Hong Kong Special Administrative Region, China: HKUST6193/02E, the National Science Foundation under grant number 9811883, and the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152.

Appendix

A Algorithms for 4-D tensor voting

We detail the general tensor voting algorithm [18] in this section. In [18], an high dimensional tensor voting algorithm is presented. C++\textsuperscript{11} and Matlab\textsuperscript{12} source codes are available. The voter makes use of GEN\text{TEGR}VOTE to cast a tensor vote to vote receiver (votee). Normal direction votes generated by GEN\text{NORMAL}VOTE are accumulated using \text{COMBINE}. An $4 \times 4$ \texttt{outTensor} is the output. The votee thus receives a set of \texttt{outTensor} from voters within its neighborhood. The resulting tensor matrices can be summed up by ADD\texttt{ENSOR}, which performs ordinary $4 \times 4$ matrix addition. This resulting matrix is equivalent to a 4D ellipsoid.

References


\textsuperscript{11}http://www.cs.ust.hk/~cktang/TVLib.zip.

\textsuperscript{12}http://www.cs.ust.hk/~cstws/research/TensorVoting3D/


Fig. 12. Epipolar geometry for camera motion and one motion non-static scenes. Motion components and their corresponding epipolar lines are indicated by green. Background matches and epipolar lines are colored in red. Discarded outliers are in blue. The first two rows show the image pair and input noisy matches. The last two rows show the results.
Fig. 13. **TOYS** - epipolar geometry for forward camera motion and two additional independent motions in a non-static scene. Motion components and their corresponding epipolar lines are indicated by green and yellow. Background matches and epipolar lines are colored in red. Discarded outliers are in blue. The first row shows the image pair and input noisy matches. The second row shows the results.
Algorithm 2 \textsc{gentensorvote} (voter,votee)

It uses \textsc{gennormalvote} to compute the most likely normal direction vote at the votee. Then, plate and ball tensors are computed, by integrating the resulting normal votes cast by voter. They are used to vote for curves and junctions.

\begin{verbatim}
for all 0 \leq i, j < 4, outTensor[i][j] \leftarrow 0
for all 0 \leq i < 3,
    voterSaliency[i] \leftarrow voter[\lambda_i] - voter[\lambda_{i+1}]

voterSaliency[3] \leftarrow voter[\lambda_3]
if (voterSaliency[0] > 0) then
    vecVote \leftarrow \textsc{gennormalvote} (voter,votee)
    \{ Compute stick component \}
    \textsc{combine} (outTensor, vecVote)
end if
transformVoter \leftarrow voter
for i = 1 to 3 do
    if (voterSaliency[i] > 0) then
        // count[i] is a sufficient number of samples uniformly
        // distributed on a unit (i + 1)-D sphere.
        while (count[i] \neq 0) do
            transformVoter[direction] \leftarrow random[direction] \leftarrow \textsc{genrandomuniformpt}()
            if (i \neq 3) then
                /* Compute the alignment matrix, except the isotropic ball tensor */
                transformVoter[direction] \leftarrow voter[eigenvectorMatrix] \times random[direction]
            end if
            vecVote \leftarrow \textsc{gennormalvote} (transformVoter,votee)
            \textsc{combine} (outTensor, vecVote, voterSaliency[i])
            count[i] \leftarrow count[i] - 1
        end while
    end if
end for
return outTensor
\end{verbatim}
Algorithm 3 GENNORMALVOTE (voter, votee)
A vote (vector) on the most likely normal direction is returned.

\[ v \leftarrow \text{votee[position]} - \text{voter[position]} \]

/* voter and votee are connected by high curvature? */

if (angle(voter[direction],v) < \pi/4) then

    return ZeroVector \{ smoothness constraint violated \}

end if

/* voter and votee on a straight line, or voter and votee are the same point */

if (angle(voter[direction],v) = \pi/2) or (voter = votee) then

    return voter[direction]

end if

Compute center and radius of the osculating hemisphere in 4D, as shown in Fig. 4a.

*/ assign stick vote */

\[ \text{stickvote}[\text{direction}] \leftarrow \text{center} - \text{voter[position]} \]

\[ \text{stickvote}[\text{length}] \leftarrow e^{-\frac{r^2 + c^2}{2\sigma^2}} \{ \text{equation (3)} \} \]

\[ \text{stickvote}[\text{position}] \leftarrow \text{votee[position]} \]

return stickvote

Algorithm 4 COMBINE (tensorvote, stickvote, weight)
It performs tensor addition, given a stick vote.

\textbf{for all } i, j \textbf{ such that } 0 \leq i, j < 4 \textbf{ do}

\[ \text{tensorvote}[i][j] \leftarrow \text{tensorvote}[i][j] + \text{weight} \times \text{stickvote}[i] \times \text{stickvote}[j] \]

\textbf{end for}

Algorithm 5 ADDTENSOR (outTensor, inTensor, weight)
Add two second order symmetric tensors – simply matrix addition

\textbf{for all } i, j \textbf{ such that } 0 \leq i, j < 4 \textbf{ do}

\[ \text{outTensor}[i][j] \leftarrow \text{outTensor}[i][j] + \text{weight} \times \text{inTensor}[i][j] \]

\textbf{end for}