N-Dimensional Tensor Voting and Application to Epipolar Geometry Estimation

Chi-Keung Tang, *Member*, *IEEE Computer Society*, Gérard Medioni, *Senior Member*, *IEEE*, and Mi-Suen Lee, *Member*, *IEEE Computer Society*

Abstract—We address the problem of epipolar geometry estimation efficiently and effectively, by formulating it as one of hyperplane inference from a sparse and noisy point set in an 8D space. Given a set of noisy point correspondences in two images of a static scene without correspondences, even in the presence of moving objects, our method extracts good matches and rejects outliers. The methodology is novel and unconventional, since, unlike most other methods optimizing certain scalar, objective functions, our approach does *not* involve initialization or any iterative search in the parameter space. Therefore, it is free of the problem of local optima or poor convergence. Further, since no search is involved, it is unnecessary to impose simplifying assumption (such as affine camera or local planar homography) to the scene being analyzed for reducing the search complexity. Subject to the general epipolar constraint *only*, we detect wrong matches by a novel computation scheme, **8D Tensor Voting**, which is an instance of the more general *N*-dimensional Tensor Voting framework. In essence, the input set of matches is first transformed into a sparse 8D point set. Dense, 8D tensor kernels are then used to vote for the most salient hyperplane that captures all inliers inherent in the input. With this filtered set of matches, the normalized Eight-Point Algorithm can be used to estimate the fundamental matrix accurately. By making use of efficient data structure and locality, our method is both time and space efficient despite the higher dimensionality. We demonstrate the general usefulness of our method using example image pairs for aerial image analysis, with widely different views, and from *nonstatic* 3D scenes (e.g., basketball game in an indoor stadium). Each example contains a considerable number of wrong matches.

Index Terms—Tensor, hyperplane inference, epipolar geometry, matching, robust estimation.

1 Introduction

In computer vision, epipolar geometry is a fundamental constraint used whenever two images of a static scene are to be registered. Two issues needed to be addressed are: the *correspondence* problem and the *parameter estimation* problem given a set of correspondences. The main difficulty stems from the unavoidable outliers inherent in the given matches. Most robust techniques require that the majority of matches be correct or else some form of outlier detection and removal is usually performed before actual parameter estimation.

Both outlier detection and parameter estimation are often formulated as a nonlinear optimization and search process in the parameter space. In the case of the full perspective camera model, this search space can be prohibitively large. Consequently, gradient-based and other nonlinear heuristic search techniques have been proposed. The output, however, may not be initialization free and poor initialization may seriously affect convergence rate. Though simplifying assumptions, such as affine camera model and local planar homography [14], can reduce the search complexity, the class of transformations which can be represented is somewhat restricted.

In this paper, we propose an unconventional, effective, and efficient approach to solve the outlier detection problem for epipolar geometry estimation. The solution technique is unconventional since we do not formulate the problem into an iterative, optimization framework, as many other researchers already did. We demonstrate the effectiveness of our approach by using difficult examples and quantitatively justify our method. Our method is efficient since, as shown in the space and time complexity section, our algorithm is independent of the dimensionality of the parameter space.

A compact conference version of this paper has appeared in [16], which includes a brief description of the 8D instance of the more general, *N*-dimensional tensor voting formalism, the latter of which is detailed in Section 2 in this paper. The present coverage contains a more detailed description and illustration on our methodology.

As shown in Section 1.2, the epipolar constraint is a linear and homogeneous one that defines an 8D hyperplane in its parameter space. Therefore, given a candidate set of (usually noisy) matches, suppose that we could *visualize* in 8D, the subset of inlier matches should cluster themselves onto a hyperplane in the corresponding 8D space. Thus, analogous to line detection in a 2D point cloud, we can pose our outlier detection problem as one of hyperplane inference in 8D. Therefore, it is of great value and theoretical interest if we can extract this salient hyperplane feature from the given sparse and noisy 8D cluster, without performing any iterative or multidimensional parameter

- C.-K. Tang is with the Computer Science Department, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. E-mail: cktang@cs.ust.hk.
- G. Medioni is with the Institute for Robotics and Intelligent Systems, University of Southern California, CA 90089-0273. E-mail: medioni@iris.usc.edu.
- M.-S. Lee is with Philips Research USA, Briarcliff Manor, NY 10510-2099. E-mail: Mi-Suen.Lee@Philips.com.

Manuscript received 12 Apr. 2000; revised 12 Jan. 2001; accepted 16 Jan. 2001.
Recommended for acceptance by A. Shashua.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 111894.

search. Simplifying assumption will become unnecessary, as the size of the search space is no longer an issue.

An intuition to a noniterative solution for hyperplane inference is inspired by the Hough Transform [7]. It employs a *voting* technique that outputs the solution receiving maximal support. However, as the dimensionality grows, the Hough Transform is extremely inefficient. Thus, it is impractical in most higher-dimensional detection problems.

Therefore, the contributions of this paper are twofold: A formal framework, the *N*-dimensional tensor voting formalism, is proposed. It is generalized from the 2D version [12]. The *N*-D tensor voting resembles the Hough transform in that it uses a voting technique; but is different since the time and space complexities are independent of dimensionality. The second contribution consists of the application of this higher-dimensional methodology in outlier detection and removal for general epipolar geometry estimation. We demonstrate the general usefulness of this novel estimation methodology with a variety of image pairs as examples.

1.1 Previous Work

If the input set of correspondences is already very good, then, a linear method, such as Eight-Point Algorithm [10], can be used for accurate parameter estimation. This algorithm is probably the most cited method for computing the essential (respectively fundamental) matrix from two calibrated (respectively uncalibrated) camera images. With more than eight points, a least mean square minimization is used, then followed by the enforcement of the singularity constraint so that the rank of the resulting matrix is 2. Its obvious advantages are speed and ease of implementation. However, in practice, the input set of matches contains a considerable amount of outliers. The linear method is extremely sensitive to wrong matches.

In [8], Hartley normalizes the data before using the Eight-Point Algorithm and shows that this normalized version performs comparably well with more complicated iterative techniques, which are described below. Outlier rejection is performed before the algorithm is used. Note that the Eight-Point Algorithm is not optimal since it minimizes an algebraic distance. Therefore, a nonlinear geometric error measure should be used.

More complicated, iterative optimization methods are proposed to address the issues of noisy matches, some of them are described in [22]. These nonlinear, robust methods make use of certain optimization criteria, such as distance between points and their corresponding epipolar lines or gradient-weighted epipolar errors. Iterative methods, in general, require careful (or at least sensible) initialization for early convergence to the desired optimum (or, in other words, avoiding local optimum). In particular, the method proposed by Zhang [22] uses least median of squares, data subsampling, and certain adapted criterion, to discard outliers by solving a nonlinear minimization problem. The fundamental matrix is then estimated. Note that robust methods require a majority of the data be correct, whereas we can tolerate much higher outlier to inlier ratio, as shown in the Section 6.

Torr and Murray [19] proposed the use of *RANSAC*: Random sampling of a minimum subset (seven pairs) for parameter estimation is performed. The solution is given by the candidate subset that maximizes the number of consistent points and minimizes the residual. It is, however, computationally infeasible to consider all possible subsets, since it is exponential in number. Therefore, additional statistical measures are needed to derive the minimum number of sample subsets. Extra samples are also needed to avoid degeneracy. In our approach, we can afford to consider all input matches, since it is linear in time and space.

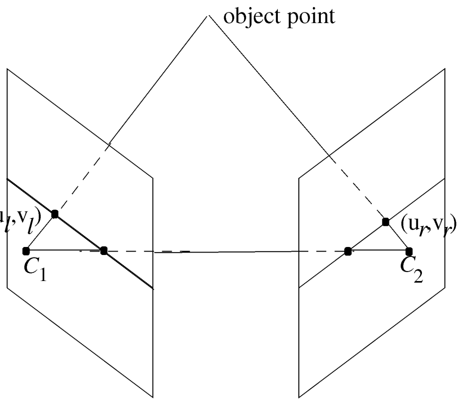

Fig. 1. Epipolar geometry.

Chai and Ma [2] proposed the use of genetic algorithm to avoid the problem of local minima, by using properly defined genetic operators. The optimization process can be sped up through incorporating the ideas of evolution that properly guides the search process.

Ke et al. [4] proposed the use of Reactive Tabu Search which avoids the prohibitive search complexity by guiding a local heuristic search among neighborhood configurations beyond local optima. A cost function is minimized while outliers are discarded, using a certain criterion.

In [14], Pritchett and Zisserman proposed the use of *local homography* (planar projective transformation). Homographies are generated by Gaussian pyramid techniques. Point matches are then generated using a homography. The set of matches is then enlarged, by using RANSAC for selecting a subset of initial matches consistent with a given homography. Besides its viewpoint invariance, homography drastically reduces the search space. However, the homography assumption, as noted, does not generally apply to the entire image (e.g., curved surfaces), although *local* homography applies in most situations.

1.2 Review of Epipolar Geometry

Here, we briefly review epipolar geometry (more details in [5]), and formulate the estimation problem as one of 8D hyperplane inference.

Given two images of a static scene taken from two camera systems (see Fig. 1), let (u_ℓ, v_ℓ) be the image coordinates of 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 line is the intersection of two planes: The first is defined by 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. A symmetric relation applies for (u_r, v_r). This is known as the *epipolar constraint*. The *fundamental matrix* $\mathbf{F}$ that relates any matching pair (u_ℓ, v_ℓ) and (u_r, v_r) is given by $\mathbf{u_1}^T\mathbf{F}\mathbf{u_2} = 0$, where $\mathbf{u_1} = (u_\ell, v_\ell, 1)^T$ and $\mathbf{u_2} = (u_r, v_r, 1)^T$. Note that $\mathbf{F}$ is a rank 2, 3×3 homogeneous matrix.

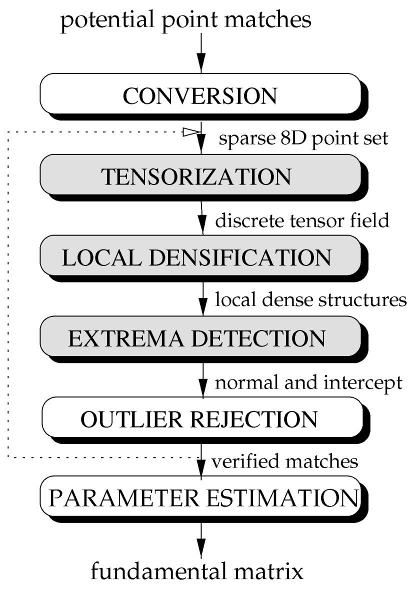

Fig. 2. The eight-dimensional tensor voting approach to the epipolar estimation problem.

The epipolar constraint can be rewritten as a linear and homogeneous equation in terms of the nine unknown coefficients in $\mathbf{F}$ or $\mathbf{u}^T\mathbf{f} + F_{33} = 0$, where

$$\mathbf{u} = [\, u_\ell u_r \quad v_\ell u_r \quad u_r \quad u_\ell v_r \quad v_\ell v_r \quad v_r \quad u_\ell \quad v_\ell \,]^T \qquad (1)$$

$$\mathbf{f} = [\, F_{11} \quad F_{12} \quad F_{13} \quad F_{21} \quad F_{22} \quad F_{23} \quad F_{31} \quad F_{32} \,]^T \qquad (2)$$

which defines a hyperplane equation in the 8D space parameterized by u_ℓ, u_r, v_ℓ, v_r. Note that $\mathbf{u}$ is our measurement, which is an 8D point defined by a match $\mathbf{u_1} \leftrightarrow \mathbf{u_2}$. Although we have nine unknowns $F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33}$ since $\mathbf{F}$ is homogeneous, the parameter space is still 8D.

The hyperplane normal (given by $\mathbf{f}$) and F_{33} are unknowns and they are estimated using our 8D version of tensor voting.

1.3 Overall Approach and Outline of this Paper

Using epipolar geometry estimation as an example, Fig. 2 shows our overall *N*-D approach, in which "Tensorization" and "Densification" are implemented as 8D voting processes (the more general *N*-dimensional tensor voting is described in Section 2).

The input set of point matches, obtained by automatic means such as cross-correlation techniques, is first converted into a sparse 8D point set as described in Section 1.2. This point set is then *"tensorized"* into a discrete tensor field, which encodes the most preferred normal direction at each point.

Then, this tensor field is locally *"densified,"* producing local dense structures suitable for *extrema detection*, from which the salient hyperplane containing all good matches can be estimated. Eight-dimensional tensorization, densification, and extrema detection are described in Section 3. Each input match is then checked against the inferred hyperplane, producing a set of filtered inliers. Other pertinent issues on implementation are described in Section 4.

Finally, the normalized eight-point algorithm (least mean square minimization followed by the enforcement of singularity constraint) is applied to the verified matches for fundamental matrix estimation. Complexity analysis and results on a variety of image pairs are described in Sections 5 and 6, respectively.

2 *N*-Dimensional Tensor Voting

In this section, we describe how to generalize the tensor voting formalism to *N*-dimension for any $N > 2$, using the 2D case as the basis. The 2D tensor voting formalism, as described in [12], can be generalized to *N*-D readily. For this reason, we shall relate the 2D version to facilitate our *N*-D discussion whenever appropriate. Table 1 summarizes the key concepts in this section:

- *N*-D tensor representation and interpretation,
- *N*-D tensor communication,
- *N*-D tensor voting implementation, and
- *N*-D feature extraction.

Before describing the above elements in *N*-D tensor voting, let us first motivate our study of *N*-dimensional voting.

2.1 Motivation

We first justify our approach in this section. While a more rigorous analysis is a subject of further research, the following should provide some insightful intuition to explain the efficacy of our approach. Our formalism consists of two important aspects: *tensor* for representation and *voting* for communication. We shall describe, in Section 2.3, the fundamental stick voting field (Fig. 5) which is the most important element. Given a point P with its finite neighborhood, what is the most likely direction at this point? Consider an analogy in particle physics in which two particles vibrate. Each emits a waveform to communicate with each other. If their frequencies agree, they will reinforce each other, producing a maxima (or resonance). If their frequencies do not agree, no maxima is produced. The fundamental stick voting field (and its derivations) is used to mimic this communication process, reporting a solution that produces the highest agreement, or the maximal response. Fig. 3 shows three scenarios:

1. A point at D receives vector votes from $A, B, C, E,$ and F, Fig. 3a. Suppose that points $A, B, C, E,$ and F are associated with a direction consistent with an underlying smooth curve (the dotted curve), on which D is lying. To propagate the continuity constraint (using the 2D fundamental stick field), each point casts a vector vote at D. Only one vector ($20°$) is consistent with the given curve tangent directions at $A, B, C, E,$ and F *simultaneously*, which are depicted as "$\times$" in Fig. 3a. If we plot the corresponding histogram, a maxima (resonance) exists at $20°$.

   Note that the situation is simplified in this illustration: in practice, other angles should receive some response. The dotted curve in the histogram is closer in appearance to the real situation.
2. A point at D receives votes from voters without any direction information, Fig. 3b. Now, suppose all the points are not associated with any direction information. Given that they are still lying on a smooth curve, when they "communicate" with each other (using the 2D ball voting field, derived from the 2D fundamental

TABLE 1
Generalization of 2D Tensor Voting to *N*-D

	2-D	*N*-D
feature to be inferred	curve	hypersurface
Representation	ellipse	hyperellipsoid
size of eigensystem	2×2	$N \times N$
number of basis tensors	2	N
stick basis	absolute certainty in *tangent* orientation along a single 2-D direction	absolute certainty in *normal* direction along a single N-D direction
ball basis	absolute uncertainty in both 2 directions	absolute uncertainty in in all N directions
quantization unit	2-D grid (i, j)	N-D hypercube $(i_1, i_2, \cdots, i_N)$
data structure for vote collection	2-D grid	N-D red-black tree (linearized)
neighborhood	grid distance = 1	Hemming distance = 1
local extremal feature	curve segment (defined by ordered zero-crossings on cuboid edges)	hypersurface patch (defined by ordered zero-crossings on hypercuboid edges)
discontinuities	intersection of lines (junctions)	intersection of hypersurfaces

stick field), only *one* direction at each site produces a maximum response to satisfy the continuity constraint. In other words, a strong maxima (resonance) at certain angle is detected at each location.

3. A point receives votes from many directions, Fig. 3c. At a point junction where more than one curves intersect, a set of votes with inconsistent directions is obtained. All orientations are equally likely. No maxima or resonance is produced.

2.2 Tensor Representation and Interpretation

In this section, we describe the components in N-D tensor voting in detail, which are used to realize the communication process described in Section 2.1.

2.2.1 Second Order Symmetric Tensor in N-D

Suppose we perform curve detection in a 2D point cluster. A point in this cluster either belongs to a *curve*, a *point junction* where intersecting curves meet, or is an *outlier*. Consider the two extremes in which a point on a curve is very certain about its *tangent* direction, whereas a point junction at which many curves intersect has absolute orientation uncertainty. Thus, we use a "stick" to indicate absolute orientation certainty and a "ball" to indicate absolute orientation uncertainty. This whole continuum can be abstracted as a second order symmetric tensor in 2D, which can be visualized geometrically as an ellipse (Fig. 4). This ellipse can be described by the

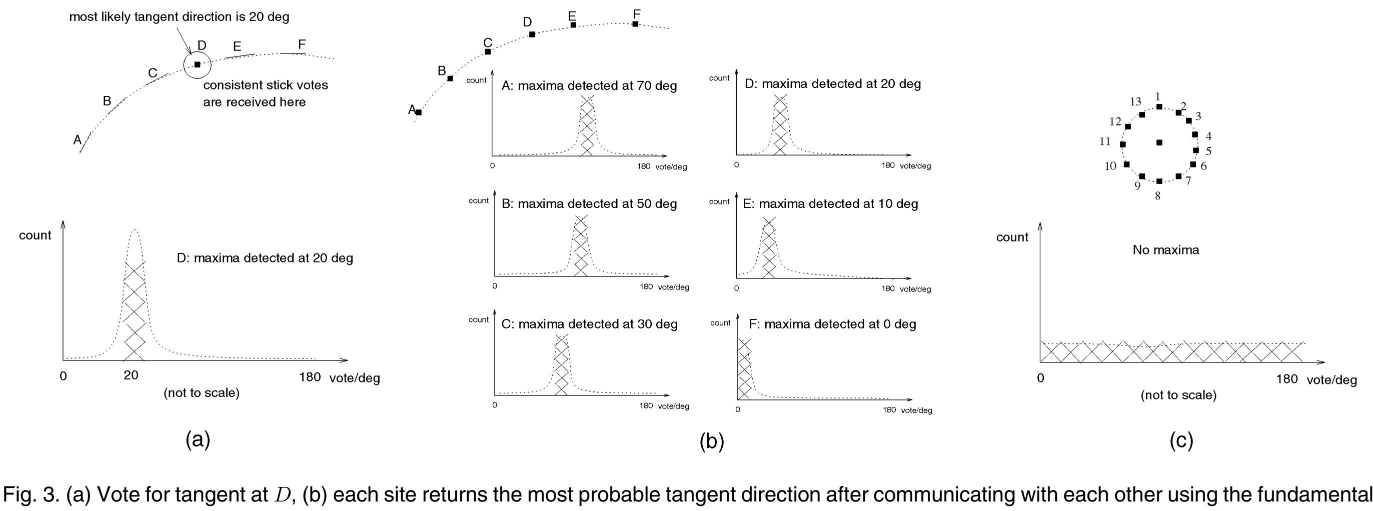
most likely tangent direction is 20 deg
A
B
C
D
E
F
consistent stick votes are received here
count
D: maxima detected at 20 deg
0
20
180
vote/deg
(not to scale)
(a)
A: maxima detected at 70 deg
B: maxima detected at 50 deg
C: maxima detected at 30 deg
D: maxima detected at 20 deg
E: maxima detected at 10 deg
F: maxima detected at 0 deg
(b)
1
2
3
4
5
6
7
8
9
10
11
12
13
No maxima
(c)

Fig. 3. (a) Vote for tangent at D, (b) each site returns the most probable tangent direction after communicating with each other using the fundamental stick field, and (c) no maxima is detected when there is no direction preference.

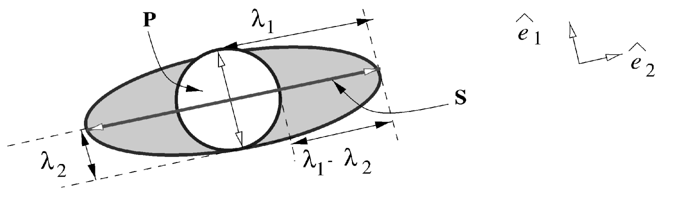

Fig. 4. A second order symmetric 2D tensor.

equivalent 2×2 eigensystem, with its two unit eigenvectors $\hat{e}_1$ and $\hat{e}_2$ and the two corresponding eigenvalues $\lambda_1 \geq \lambda_2$:

$$(\lambda_1 - \lambda_2)\mathbf{S} + \lambda_2\mathbf{B}, \tag{3}$$

where $\mathbf{S} = \hat{e}_1\hat{e}_1^T$ defines a *stick tensor*, and $\mathbf{B} = \hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T$ defines a *ball tensor*, in 2D. Note that these tensors define the two *basis* tensors for any 2D ellipse.

Analogously in N-D, in hypersurface detection, a point in the N-D space can either be: on *hypersurface* (smooth), at a *discontinuity* of order between two and N, or is an *outlier*. Consider the two extremes: an N-D point on a hypersurface is very certain about its *normal* direction, whereas a point at a junction has absolute orientation uncertainty. As in 2D, this whole continuum can be abstracted as a second order symmetric N-D tensor, or equivalently, a *hyperellipsoid*. This hyperellipsoid can be equivalently described by the corresponding eigensystem with its N eigenvectors $\hat{e}_1, \hat{e}_2, \cdots, \hat{e}_N$ and the N corresponding eigenvalues, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$. Rearranging the $N \times N$ eigensystem, the N-D ellipsoid is given by:

$$(\lambda_1 - \lambda_2)\mathbf{S} + \sum_{i=2}^{N-1}(\lambda_i - \lambda_{i+1})\sum_{j=1}^{i}\hat{e}_j\hat{e}_j^T + \lambda_N\mathbf{B}. \tag{4}$$

In particular, $\mathbf{S} = \hat{e}_1\hat{e}_1^T$ and $\mathbf{B} = \sum_{i=1}^N \hat{e}_i\hat{e}_i^T$ defines an N-D *stick* and *ball*, respectively, among all the N *basis* tensors. We call the rest of $N-2$ basis tensors $\sum_{j=1}^i \hat{e}_j\hat{e}_j^T$ *plate* tensors. Any hyperellipsoid in N-D can be represented by a linear combination of these N basis tensors.

2.2.2 N-D Tensor Interpretation

We now explain the geometric meaning of the eigensystem we derived in the previous section. Return to the 2D case. The *eigenvectors* encode *orientation (un)certainties*: *Tangent* direction is described by the *stick* tensor, indicating *certainty* along a single direction. At *point* junctions, where more than two intersecting lines converge, a *ball* tensor is used since there is no preferred orientation. The *eigenvalues* encode the *magnitude* of orientation (un)certainties since they indicate the size of the ellipse. Hence, given a generic ellipse and its equivalent eigensystem, we have the following geometric interpretation:

- $(\lambda_1 - \lambda_2)$ corresponds to 2D *curve saliency*, with a stick tensor $\hat{e}_1\hat{e}_1^T$ indicating the curve tangent direction,

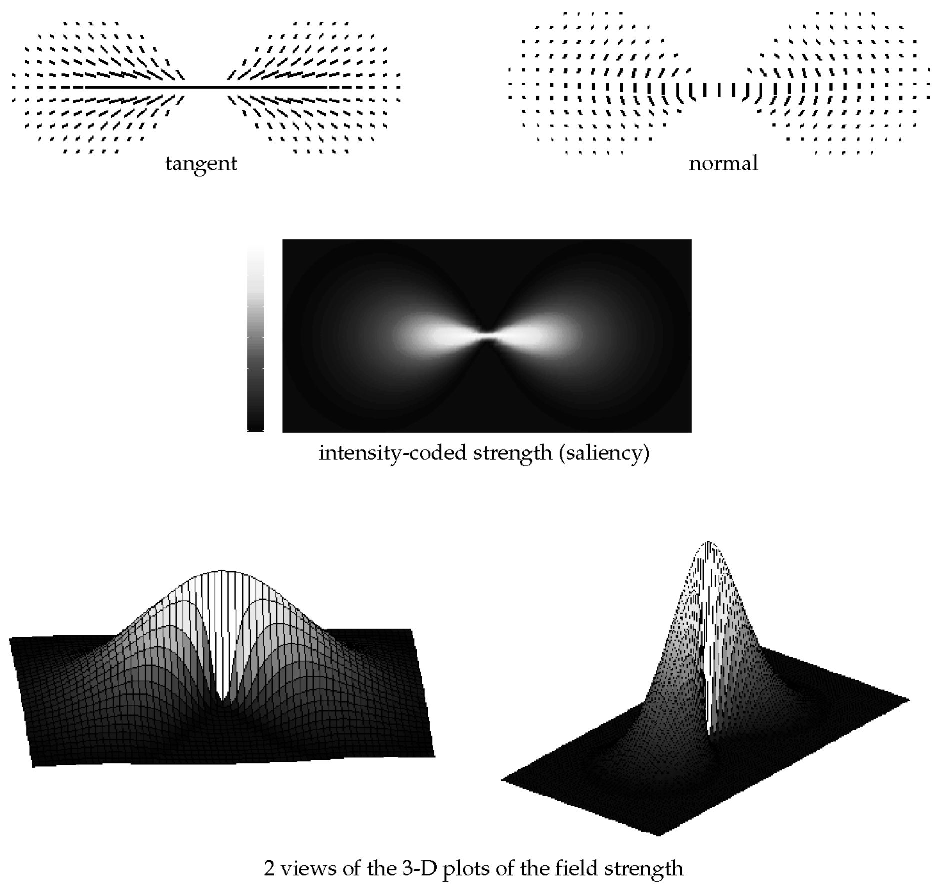

Fig. 5. The fundamental 2D stick voting field.

- λ_2 corresponds to 2D *junction saliency*, with total uncertainty in orientation as indicated by the ball tensor, or $(\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T)$.

In N-D, we have similar geometric interpretation: The eigenvectors effectively encode orientation (un)certainties: A normal to a *hypersurface* is described by the *stick* tensor, which indicates *certainty* along a single, N-D direction. *Orientation uncertainty* is indicated by the *ball* tensor, where many intersecting hypersurfaces are present and, thus, no single orientation is preferred. The *eigenvalues* encode the *magnitudes* of orientation (un)certainties:

- $(\lambda_1 - \lambda_2)$ corresponds to N-D *hypersurface saliency*, with $\hat{e}_1\hat{e}_1^T$ indicating the normal direction,
- for $2 \leq i < N$, $(\lambda_i - \lambda_{i+1})$, and $\sum_{j=1}^{i} \hat{e}_j\hat{e}_j^T$ correspond to orientation uncertainty in i directions, which actually defines a $(N-i)$-D feature whose direction(s) are given by $\hat{e}_{i+1}, \cdots, \hat{e}_N$. For example, given $N = 3, i = 2$, $(\lambda_2 - \lambda_3)(\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T)$ defines a plate tensor in 3D, which describes a 1D feature, a curve element, with tangent direction given by $\hat{e}_3$. Here, the normal orientation *uncertainty* only spans a plane perpendicular to $\hat{e}_3$, indicated by a plate tensor defined as $\sum_{j=1}^{2} \hat{e}_j\hat{e}_j^T$.
- $\lambda_N \sum_{j=1}^{n} \hat{e}_j\hat{e}_j^T$ corresponds to N-D *hyperjunction saliency*, with the N-D ball tensor specifying total orientation uncertainty.

2.2.3 Uniform Encoding

Therefore, given an N-D point, with or without orientation, we can unify the input into a tensor field by the following: If the input token is a point without any directional information, it is encoded as a N-D ball tensor $(\lambda_1 = \lambda_2 = \cdots = \lambda_N = 1)$ since initially there is no preferred orientation. $[\hat{e}_1\ \hat{e}_2\ \cdots \hat{e}_N]^T$ is an $N \times N$ identity matrix.

If the input token is a curve element in N-D, it is encoded as a plate tensor $(\lambda_1 = \lambda_2 = \cdots = \lambda_{N-1} = 1, \lambda_N = 0)$, with $\hat{e}_N$ equal to the direction of the curve tangent. Other plates are encoded accordingly.

If the input token is a N-D hypersurface patch element, then it is encoded as a stick tensor $(\lambda_1 = 1, \lambda_2 = \lambda_3 = \cdots = \lambda_N = 0)$, with $\hat{e}_1$ equal to the direction of the surface normal to the given hypersurface patch.

In implementation, we first encode the input uniformly into a tensor field. Each input tensor token then *communicates*, by a voting algorithm, in order to obtain a generic tensor, which describes the orientation preference (or no preference) at that site.

2.3 *N*-D Tensor Communication: *N*-D Tensor Voting

Each input token *votes*, or is made to align, with precomputed, discrete versions of the basis tensors in a convolution-like way, propagating preferred direction in a neighborhood. We call these precomputed basis tensors *voting fields*. As a result, preferred orientation information is propagated and gathered at each input site. This voting process consists of two phases:

- token refinement ("tensorization"): tensor votes are collected at input sites only and

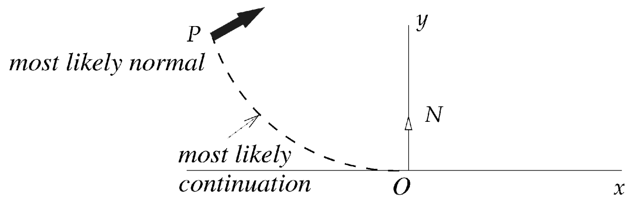

Fig. 6. The design of fundamental 2D stick voting field.

- dense extrapolation ("densification"): tensor votes fill the volume for feature extraction.

2.3.1 Token Refinement and Dense Extrapolation

Given a set of input tokens, they are first encoded as tensors as described in Section 2.2.3. These initial tensors communicate with each other by token refinement and dense extrapolation.

In essence, in token refinement, each token collects all the tensor values cast at its location by all the other tokens in a neighborhood. The resulting tensor value is the tensor sum of all the tensor votes cast at the token location. In dense extrapolation, each token is first decomposed into its corresponding N elements. By using an appropriate voting field, each token broadcasts the information in a neighborhood. The size of the neighborhood is given by the size of the voting field used. As a result, a tensor value is put at every location in the neighborhood. The resulting dense information can be used for feature extraction in which first derivatives are computed.

In fact, these two tasks can be implemented by a voting process, which in essence involves having each input token aligned with precomputed, dense, N-D voting fields. The alignment is simply a translation followed by rotation in the N-D space.

Therefore, it remains to describe the N-D voting fields, which can be derived from the most basic, fundamental 2D stick voting field.

2.3.2 Derivation of N-D Voting Fields

The fundamental 2D stick voting field. Voting fields of *any* dimensions can be derived from the 2D stick tensor and, therefore, it is called the *fundamental 2D stick voting field*. Fig. 5 shows this fundamental 2D stick voting field.

In 2D, a direction can be defined by either the tangent vector, or the normal vector, which are orthogonal to each other. We can therefore define two equivalent fundamental fields, depending on whether we assign a tangent or normal vector at the receiving site.

Here, we describe the normal version of the 2D stick kernel. The tangent version is similar. Given a point at the origin with a known normal ($\overline{N}$), we ask the following question: *For a given point P in space, what is the most likely normal (at P) to a curve passing through O and P and normal to* $\overline{N}$? Fig. 6 illustrates this situation. We claim that the *osculating circle* connecting O and P is the most likely one since it keeps the curvature constant along the hypothesized circular arc. For a detailed theoretical treatment, please refer to [12]. The most likely normal is given by the *normal to the circular arc at P*. The length of the normal vector at P, which represents the saliency of the vote, is inversely proportional to the arc length OP and also to the curvature of the underlying circular arc. In doing so, both the proximity and the smoothness (or lower curvature) constraints are effectively encoded as the corresponding saliency measure, representing the likelihood of a smooth curve passing through that point.

In spherical coordinates, the decay of the fundamental 2D stick field takes the following form:

$$\overline{DF}(r, \varphi, \sigma) = e^{-\left(\frac{r^2+c\varphi^2}{\sigma^2}\right)}, \tag{5}$$

where r is the arc length OP, φ is the curvature, and σ is the scale of analysis, the only free parameter in our formalism.

Note that the connection as given by the osculating circle becomes less likely if the angle subtended by $\overline{N}$ and $\overline{OP}$ is less than $45°$. Therefore, we only consider the set of orientations for which the angle defined above is not less than $45°$. See Fig. 5.

Derivation of *N*-D stick kernel. Without loss of generality, we derive the N-D stick kernel oriented at either $\pm[1\ 0\ 0\ \cdots\ 0]^T$ in world coordinate system (note we do not distinguish the polarity of orientation which is unknown). Other orientations can be achieved by a simple rotation in the N-D space. We first rotate the normal version of the 2D stick kernel so that it describes the orientation $\pm[1\ 0]^T$. Denote the rotated, normal version of the fundamental 2D stick voting field V'_F. We adopt the parameterization $\mathbf{T}(\lambda_1, \lambda_2, \cdots, \lambda_N, \alpha_N, \alpha_{N-1}, \cdots, \alpha_1)$, where α_i are angles of rotation about the axis i. Therefore, this parameterization characterizes the magnitude and orientation of the tensor $\mathbf{T}$. Using this parameterization, the N-D stick $S(\cdot)$ is:

$$\mathbf{S}(1, \underbrace{0, \cdots, 0}_{N-1}, \alpha_N, \alpha_{N-1}, \cdots, \alpha_1) = \int_0^\pi V'_F d\alpha_1|_{\alpha_N=\alpha_{N-1}=\cdots=\alpha_2=0}. \tag{6}$$

Derivation of *N*-D ball kernel. The N-D ball kernel can be obtained by rotating the above N-D stick kernel $\mathbf{S}$ about the remaining $N-1$ axes and integrating the contributions:

$$\begin{aligned}&\mathbf{B}(\underbrace{1, 1, \cdots, 1}_{n}, \alpha_N, \alpha_{N-1}, \cdots, \alpha_1)\\ &= \int_0^\pi \int_0^\pi \cdots \int_0^\pi \mathbf{S} d\alpha_2 d\alpha_3 \cdots d\alpha_N|_{\alpha_1=0}.\end{aligned} \tag{7}$$

Derivation of other *N*-D kernels. For the other i kernels, $2 \le i < N$, we have

$$\begin{aligned}&\mathbf{P}(\overbrace{1, 1, \cdots, 1}^{i}, \overbrace{0, 0, \cdots, 0}^{n-i}, \alpha_N, \alpha_{N-1}, \cdots, \alpha_1)\\ &= \int_0^\pi \int_0^\pi \cdots \int_0^\pi \mathbf{S} d\alpha_{n-i+2} \cdots d\alpha_N|_{\alpha_1=\alpha_2=\cdots=\alpha_{n-i+1}=0},\end{aligned} \tag{8}$$

which actually describes the rotation of the N-D stick field $\mathbf{S}$ about the i axes and the integration of the contributions from all possible angles of rotation.

2.4 Algorithms on *N*-Dimensional Tensor Voting

In Section 2.3.2, we give the continuous definitions for the stick, plate, and ball voting fields. In this section, we describe the discrete algorithms on tensor voting which use discrete voting fields. C++ source codes are available at http://www.cs.ust.hk/~cktang. The voter makes use of GENTENSORVOTE (Algorithm 1) to cast a tensor vote to vote receiver (votee), by integrating contribution using a stick rotated about an eigenvector, Section 2.3.2. Stick votes generated by GENSTICKVOTE (Algorithm 2) are accumulated using COMBINE (Algorithm 3). An $N \times N$ $outTensor$ is the output. The votee thus receives a set of $outTensor$ from voters within its neighborhood. The resulting tensor matrices can be summed up by ADDTENSOR (not shown here), which performs ordinary $N \times N$ matrix addition. These algorithms work for any dimension ≥ 2. To increase efficiency, a dense voting field is precomputed once, by calling GENTENSORVOTE k^N times, where k is the scale or size of the neighborhood. In particular, it is sufficient to use a 1D array of size k to store an N-D ball since an N-D ball is isotropic in all directions.

Algorithm 1 GENTENSORVOTE (dimension,voter,votee)

First, the stick component of a tensor vote is computed (if direction is given). Then, all other tensor components (plates and balls) are computed, by integrating the resulting stick votes cast by a rotating stick at the voter.

```
for all $0 \le i, j <$ dimension, outTensor[i][j] $\leftarrow$ 0
for all $0 \le i <$ dimension $- 1$,
        voterSaliency[i] $\leftarrow$ voter[$\lambda_i$] $-$ voter[$\lambda_{i+1}$]
voterSaliency[dimension-1] $\leftarrow$ voter[$\lambda_{dimension-1}$]
if (voterSaliency[0] $>$ 0) then
  vecVote $\leftarrow$ GENSTICKVOTE (dimension,voter,votee)
  COMBINE (dimension,outTensor,vecVote)
end if
transformVoter $\leftarrow$ voter
for $i = 1$ to dimension $- 1$ 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] $\leftarrow$
        GENRANDOMUNIFORMPT()
      if ($i \neq$ dimension$-1$) then
        /* Compute the alignment matrix, except the
           isotropic ball tensor */
        transformVoter[direction] $\leftarrow$
           voter[eigenvectorMatrix] $\times$ random[direction]
      end if
      vecVote $\leftarrow$ GENSTICKVOTE
           (dimension,transformVoter,votee)
      COMBINE (outTensor, vecVote, voterSaliency[i])
      count[i] $\leftarrow$ count[i] $-$ 1
    end while
  end if
end for
return outTensor
```

Algorithm 2 GENSTICKVOTE (dimension, voter, votee)

A stick vote (vector) is returned.

```
$v \leftarrow$ votee[position] $-$ voter[position]
/ * check if 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 $N$-dimensional
        hemisphere
stickvote[direction] $\leftarrow$ center $-$ voter[position]
stickvote[length] $\leftarrow e^{-\frac{s^2+c\rho^2}{\sigma^2}}$ {equation (5)}
stickvote[position] $\leftarrow$ votee[position]
return stickvote
```

Algorithm 3 COMBINE (dimension, tensorvote, stickvote, weight)

It performs tensor addition, given a stick vote.

```
for all $i, j$ such that $0 < i, j <$ dimension $- 1$ do
    tensorvote[i][j] $\leftarrow$ tensorvote[i][j] + weight $\times$ stickvote[i]
            $\times$ stickvote[j]
end for
```

2.5 Implementation of *N*-D Tensor Voting

The *N*-D tensor voting process aggregates tensor contribution from a neighborhood of voters by using tensor addition. Consider the case that we have only two input tokens. Initially, before any voting occurs, each token location encodes the local tensor information (i.e., Section 2.2.3). Denote these two tensors by $\mathbf{T}_{0,1}$ and $\mathbf{T}_{0,2}$. The following describes the token refinement step. The dense extrapolation step can easily be generalized from it.

Tensor Encoding. The input is encoded into a perfect ball, a perfect stick, or a perfect plate (note that, in the case, of *N*-D, we have a total of $N-2$ cases of plate tensors). One example is a "hypercurve": a space curve occupying in the *N*-D space.

- Stick. $\lambda_1 = 1, \lambda_2 = \lambda_3 = \cdots = \lambda_N = 0$, with $\hat{e}_1$ equals to the given orientation, $\hat{e}_2, \cdots, \hat{e}_N$ are unit vectors orthonormal to each other and also to the given $\hat{e}_1$. Therefore, a local, Cartesian coordinate system aligned with $\hat{e}_1$ can be used to determine $\hat{e}_2, \cdots, \hat{e}_N$.
- Plates. For $2 \leq i < N$, $\lambda_1 = \lambda_2 = \cdots = \lambda_i = 1$, $\lambda_{i+1} = \lambda_{i+2} = \cdots = \lambda_N = 0$. We set $\hat{e}_{i+1}, \hat{e}_{i+2}, \cdots, \hat{e}_N$ to be the given orientations, respectively. $\hat{e}_1, \hat{e}_2, \cdots, \hat{e}_N$ are unit vectors chosen such that they are orthonormal to each other, and to $\hat{e}_{i+1}, \hat{e}_{i+2}, \cdots, \hat{e}_N$. Again, a local, Cartesian coordinate system aligned with $\hat{e}_{i+1}, \hat{e}_{i+2}, \cdots, \hat{e}_N$ can be used to determine $\hat{e}_1, \cdots, \hat{e}_N$.

  (For example, for $N = 3, i = 2$, we use $\lambda_1 = \lambda_2 = 1$ and $\lambda_3 = 0$ to represent a curve element as a 3D plate tensor, with $\hat{e}_3$ being equal to the given curve tangent direction. Having determined $\hat{e}_3$, we can align $\hat{e}_3$ with a local 3D Cartesian coordinate system to determine $\hat{e}_1$ and $\hat{e}_2$.)
- Ball. $\lambda_1 = \lambda_2 = \cdots = \lambda_N = 1$, with $\hat{e}_1 = [1\ 0\ \cdots\ 0]^T$, and $\hat{e}_2 = [0\ 1\ \cdots\ 0]^T, \cdots$, and $\hat{e}_N = [0\ 0\ \cdots\ 1]^T$.

Therefore, the input is unified into a *N*-D second order symmetric tensor $\mathbf{T}_{0,j}, 1 \leq j \leq 2$ by

$$\mathbf{T}_{0,j} = [\hat{e}_1\quad \hat{e}_2\quad \cdots\quad \hat{e}_N] \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \lambda_N \end{bmatrix} \begin{bmatrix} \hat{e}_1^T \\ \hat{e}_2^T \\ \vdots \\ \hat{e}_N^T \end{bmatrix} \tag{9}$$

$$= (\lambda_1 - \lambda_2)\mathbf{S} + \sum_{k=2}^{N-1}(\lambda_k - \lambda_{k+1}) \sum_{l=1}^{k} \hat{e}_l \hat{e}_l^T + \lambda_N \mathbf{B} \tag{10}$$

$$= \mathbf{T}_{0,j}^S + \sum_{k=2}^{N-1} \mathbf{T}_{0,j}^{P_k} + \mathbf{T}_{0,j}^B, \tag{11}$$

where $\mathbf{S} = \hat{e}_1 \hat{e}_1^T$ and $\mathbf{B} = \sum_{l=1}^{N} \hat{e}_l \hat{e}_l^T$ defines an *N*-D *stick* and *ball*, respectively. The $N-2$ plate tensors are defined respectively by $(\lambda_k - \lambda_{k+1}) \sum_{l=1}^{k} \hat{e}_l \hat{e}_l^T,\ 2 \leq k < N$. These *N basis* tensors define any hyperellipsoid in *N*-D, by a linear combination of them.

Note that all the $N \times N$ matrices $\mathbf{T}_{0,j}^S, \mathbf{T}_{0,j}^{P_k}, \mathbf{T}_{0,j}^B$ are symmetric, semipositive definite and they describe a stick, plate, and ball tensor, respectively.

Tensor Voting. An input site j collects the tensor vote cast from the voter i. This vote consists of a stick component, $N-2$ plate components, and a ball component.

- Stick vote. Let $[v_1\ v_2\ \cdots\ v_N]^T$ be the stick vote collected at site j, which is cast by voter i after aligning the $\hat{e}_1$ component of the tensor $\mathbf{T}_{0,i}$ at i (obtained in the tensor encoding stage) with the *N*-D stick voting field, by translation and rotation. Then,

$$\mathbf{T}_{1,j}^S = \mathbf{T}_{0,j}^S + (\lambda_1 - \lambda_2) \begin{bmatrix} v_1^2 & v_1 v_2 & \cdots & v_1 v_N \\ v_2 v_1 & v_2^2 & \cdots & v_2 v_N \\ \cdots & \cdots & \cdots & \cdots \\ v_N v_1 & v_N v_2 & \cdots & v_N^2 \end{bmatrix} \tag{12}$$

$$= \mathbf{T}_{0,j}^S + (\lambda_1 - \lambda_2)\mathbf{T}, \tag{13}$$

  where $\mathbf{T}$ is $N \times N$ a symmetric, semipositive definite matrix, consisting of the second order moment collection of the vote contribution. That is, by using Algorithm 3: COMBINE.
- Plate votes. Let $\mathbf{T}^{P_k}$ be the plate vote, $2 \leq k < N$, collected at site j, which is cast by voter i. Then, $\mathbf{T}_{1,j}^{P_k} = \mathbf{T}_{0,j}^{P_k} + (\lambda_k - \lambda_{k+1})\mathbf{T}^{P_k}$. That is, by using ADDTENSOR.
- Ball vote. Let $\mathbf{T}^B$ be the ball vote, collected at site j, which is cast by voter i. Then, $\mathbf{T}_{1,j}^B = \mathbf{T}_{0,j}^B + \lambda_N \mathbf{T}^B$.

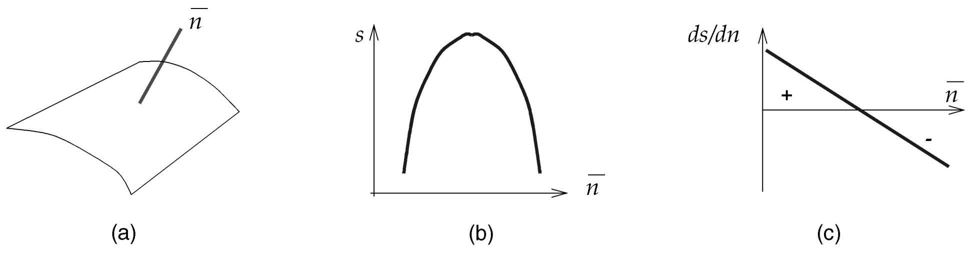

Fig. 7. *N*-D surface extremality. (a) An *N*-D normal (with an imaginary patch drawn), (b) saliency along normal, and (c) the derivative.

Therefore, $\mathbf{T}_{1,j} = \mathbf{T}^S_{1,j} + \sum_{k=2}^{N-1} \mathbf{T}^{P_k}_{1,j} + \mathbf{T}^B_{1,j}$ is obtained. That is, by using ADDTENSOR. Note that this tensor sum, $\mathbf{T}_{1,j}$, is still symmetric and semipositive definite since $\mathbf{T}^S_{1,j}$, $\sum_{k=2}^{N-1} \mathbf{T}^{P_k}_{1,j}$, and $\mathbf{T}^B_{1,j}$ are all symmetric and semipositive definite. Hence, $\mathbf{T}_{1,j}$ produced by the above is also a second order symmetric tensor.

Tensor Decomposition. After $\mathbf{T}_{1,1}$, $\mathbf{T}_{1,2}$ have been obtained by the above voting process implemented as tensor addition, we decompose each of them into the corresponding eigensystem.

Note that the same tensor sum applies to the tensor voting process for extrapolating directional estimates, with the following changes:

- First, site j may or may not hold an input token. For noninput site j, $\mathbf{T}_{0,j}$ is a zero matrix.
- Second, we do not propagate the ball component from voting sites since the ball component corresponds to junction information and thus should not be propagated.

2.6 *N*-D Feature Extraction

Recall that we need dense extrapolation to produce a dense structure for feature extraction. After the dense extrapolation stage, N dense structures, which are dense scalar or vector fields, are produced after the tensor decomposition step into the corresponding eigensystem. Here, we only consider the two extreme cases, the stick and the ball maps:

- hypersurface map: Each N-D voxel in this map consists of 2-tuple $(s, \hat{n})$, where $s = \lambda_1 - \lambda_2$ indicate the *hypersurface-ness*, or hypersurface saliency, and $\hat{n} = \hat{e}_1$ denotes the hypersurface normal direction.
- hyperjunction map: It is a dense scalar map $\{\lambda_N\}$ which denote, the *hyperjunction-ness*, or hyperjunction saliency.

Since the other $N-2$ vector maps are more complicated and we have not applied them in applications, we shall not study the feature extraction from these maps in this paper.

The extraction of maximal junction in the hyperjunction map is very straightforward: It is a local maxima of the scalar value λ_N.

2.6.1 *N*-D Hypersurface Extremality

Here, we generalize 3D extremal surface [18] to N-D extremal hypersurface. Recall that the hypersurface map is a dense vector field $\{(s, \hat{n})\}$ which encodes hypersurface normals $\hat{n} = \hat{e}_1$ associated with saliency values $s = \lambda_1 - \lambda_2$.

To illustrate, suppose the dense structure as obtained after the dense extrapolation stage, or densification, is dense and continuous, i.e., $\{(s, \hat{n})\}$ is defined for every point P in the N-D space.

Suppose that we could traverse and look at the s values along the N-D vector $\hat{n}$ (Fig. 7a). By the definition of the N-D stick kernel, if a patch exists, after tensor voting, a maxima in s (Fig. 7b) should be detected.

Therefore, we define an *extremal hypersurface* as the locus of points for which the saliency s is locally extremal along the direction of the N-D normal, i.e., $\frac{ds}{d\hat{n}} = 0$. This is only a necessary condition for extrema detection. The sufficient condition, which is used in the implementation, is defined in terms of *zero crossings* along the line defined by $\hat{n}$ (Fig. 7c). We first compute the saliency gradient along the N principal axes $x_1, x_2, \cdots, x_N$:

$$\overline{g} = \bigtriangledown s = \begin{bmatrix} \frac{\partial s}{\partial x_1} & \frac{\partial s}{\partial x_2} & \cdots & \frac{\partial s}{\partial x_N} \end{bmatrix}^T.$$

Then, we project $\overline{g}$ onto $\hat{n}$ by computing the inner product, i.e., $q = \hat{n} \cdot \overline{g}$. Thus, an extremal hypersurface is the locus of points with $q = 0$.

2.6.2 *Discrete Version*

In implementation, we have a discrete $\{(s_{i_1,i_2,\cdots,i_N}, \hat{n}_{i_1,i_2,\cdots,i_N})\}$ dense map. We can define the corresponding discrete versions of $\overline{g}$ and q, i.e.,

$$\overline{g_{i_1,i_2,\cdots,i_N}} = \begin{bmatrix} s_{i_1+1,i_2,\cdots,i_N} - s_{i_1,i_2,\cdots,i_N} \\ s_{i_1,i_2+1,\cdots,i_N} - s_{i_1,i_2,\cdots,i_N} \\ \vdots \\ s_{i_1,i_2,\cdots,i_N+1} - s_{i_1,i_2,\cdots,i_N} \end{bmatrix} \tag{14}$$

and $q_{i_1,i_2,\cdots,i_N} = \hat{n}_{i_1,i_2,\cdots,i_N} \cdot \overline{g_{i_1,i_2,\cdots,i_N}}$. Given an input point, we compute $q_{i_1,i_2,\cdots,i_8}$ at each vertex voxel (a total of 2^N that makes up the hypercube quantization unit, or hypercuboid, that contains the input site). Thus, the set of all $\{q_{i_1,i_2,\cdots,i_N}\}$ constitutes a scalar field. If the signs of the q's of two adjacent vertex voxels are different, a *zero crossing* occurs on the corresponding hypercuboid edge (there is a total of $N2^{N-1}$ of them). Implementation issues on N-D feature extractions which arose from the increase in N are described in Section 3.

2.6.3 *Extraction of* $(N-1)$-D *Entity*

When the zero crossings have been detected, in our N-D case, we need to group these zero crossings in order to find an $(N-1)$-D entity, or a hypersurface patch, that intersects with the N-D hypercuboid.

Weigle and Banks [20] describe a contour meshing procedure which generalizes well to N dimensions. We only

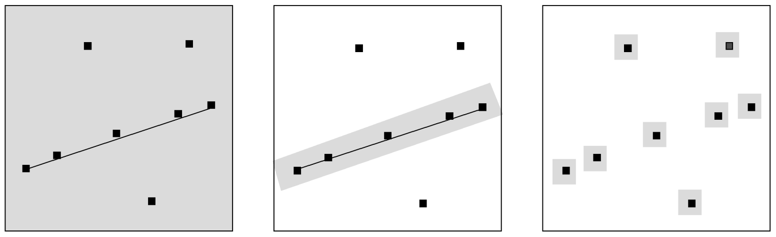

Fig. 8. Local densification in 2D. (a) Vote casting densifies everywhere; in vote gathering, we compute (b) a slab of votes only, or even perform (c) local densification.

outline the procedure here. Details (and terms in italics below) can be found in the paper and in a technical report [17].

- First, a splitting operation is performed, which divides a hypercuboid, made up of 2^N vertex voxels, into a set of $2^{N-1}N!$ *N-simplexes*.
- Zero crossings are then detected on the edges of these resulting simplexes.
- A *contouring algorithm* is applied recursively, starting by contouring 1-simplexes (edge) and then 2-simplexes (triangle), so on.

Given a hypercuboid, which is made up of 2^N vertices, a *contour*, made up of $(N-1)$-simplexes, should be produced if a hypersurface (an $(N-1)$-D entity) passes through that hypercuboid. Therefore, the detection of hypersurface is translated into the following verification: If the candidate contour as produced by the above contour meshing procedure can be triangulated into a set of $(N-1)$-simplexes and nothing more, i.e., without any "dangling" simplex of lower dimensions left behind, then, we can conclude that a hypersurface is detected.

We have now explained the generalized *N*-D version of tensor voting, and extraction of 0D entity (hyperjunction) and $(N-1)$-D entity (hypersurface), and cited the relevant reference on *N*-D feature extraction. In the following section, we specialize the *N*-D tensor voting and feature extraction into 8D and describe implementation details to cope with the complexity issues which arose in feature extraction, as a result of higher dimensionality.

3 Epipolar Geometry Estimation

In this section, we specialize the *N*-D version and apply it to the estimation of epipolar geometry. First, we transform each match as described in Section 1.2 into a point in an 8D space parameterized by u_ℓ, u_r, v_ℓ, v_r. Then, we perform the following steps, which are depicted as shaded processes in the flowchart of Fig. 2.

3.1 Tensorization in 8D

Each 8D point, which corresponds to a potential match, is first encoded as an 8D ball. Then, these input balls communicate with each other, propagating ball votes in a neighborhood. After each input site has collected all the 8D tensor votes in its neighborhood, the resulting tensor is decomposed into the corresponding eight components. Since we want to infer a hyperplane, the 8D ball component is discarded as it corresponds to junction information, which should not be propagated in the dense voting stage.

3.2 Local Densification in 8D

After the input has been tensorized, the stick component at each input tensor is made to align with the 8D stick voting field for obtaining a densified structure SMap $\{(s, \hat{n})\}$, which indicates hypersurface-ness, as defined in Section 2.6. This dense structure is used for extrema detection (Section 3.3), which discards outlier matches. This voting process is exactly the same as tensorization, except that directed votes are also collected at noninput sites in the volume.

We first give a 2D analogy to motivate the issue we need to address due to higher dimensionality. In 2D line extraction, we can afford to densify the whole 2D domain, i.e., votes are *cast everywhere*, Fig. 8a. Or, more efficiently, since we have obtained saliency information after tensorization (in 2D), densification starts out from the *most* salient site first. Votes are then propagated subject to connectivity (since a connected line should be extracted). Hence, only a slab of votes enveloping the line are computed during the extraction process, Fig. 8b.

In our 8D version, two implementation differences are made to avoid the drastic increase in time and space complexities in feature extraction due to the higher dimensionality.

For time efficiency, we do *not* even need to compute a slab of votes, since all we need is *outlier rejection*, for which an explicit hyperplane represented as connected hypersurface patches (analogous to connected line segments) is not necessary. We pose this outlier rejection problem as one of *extrema detection* in 8D (next section), which is performed at each input site only. Therefore, it is sufficient to perform *local* densification, Fig. 8c. Each input site *gathers* all the stick votes cast within the neighborhood (defined by the size of the voting field), performs smoothing, computes the eigensystem, interprets the vote, and produces a hypersurface patch (if any) for that site only, all on-the-fly. The result produced by vote gathering is the same as vote casting since they are reciprocal to each other.

For space efficiency, both local densification and vote gathering imply that it is unnecessary to keep the sparse input in an explicit 8D voxel array, which would be very expensive. Since we quantize the input, we use a "linearized 8D red-black tree" to store each input site. This data structure is an ordinary red-black tree, but we concatenate the eight integer coordinates as the search key. Its size is only $O(m)$, m is the input size. Hence, it is much less expensive than a whole 8D array. Please refer to a standard algorithm text (such as [4]) for details on red-black tree.

3.3 Extrema Detection and Outlier Rejection

Now, for each input site, we have computed a dense and local collection $\{(s, \hat{n})\}$ of the SMap that encodes surface normals $\hat{n}$ associated with saliency values s. We want to infer a salient hyperplane, with subvoxel accuracy, that contains the set of inliers. It is done by *extrema detection*, which indicates whether a salient hyperplane passes through that site.

3.3.1 8D Extremality

We specialize the N-D extremality to 8D extremality in this section. In the 8D implementation, we compute the saliency gradient

$$\overline{g} = \bigtriangledown s = \left[\frac{\partial s}{\partial x_1}\frac{\partial s}{\partial x_2}\cdots\frac{\partial s}{\partial x_8}\right]^T,$$

and then project $\overline{g}$ onto $\hat{n}$, i.e., $q = \hat{n} \cdot \overline{g}$. Thus, an extremal hypersurface is the locus of points with $q = 0$.

We have discrete $\{(s_{i_1,i_2,\cdots,i_8}, \hat{n}_{i_1,i_2,\cdots,i_8})\}$ in implementation. We can define the corresponding discrete versions of $\overline{g}$ and q, and $q_{i_1,i_2,\cdots,i_8} = \hat{n}_{i_1,i_2,\cdots,i_8} \cdot \overline{g_{i_1,i_2,\cdots,i_8}}$. Given an input point, we compute $q_{i_1,i_2,\cdots,i_8}$ at each vertex voxel (a total of $2^8 = 256$ voxels, making up a hypercuboid). Thus, the set of all $\{q_{i_1,i_2,\cdots,i_8}\}$ constitutes a scalar field. If the signs of the qs of two adjacent vertex voxels are different, a zero crossing occurs on the corresponding hypercuboid edge (there is a total of 1,024 of them).

3.3.2 Grouping of Detected Zero Crossings

The mere existence of zero crossings does not necessarily imply the presence of a salient hyperplane because outlier noise can produce local perturbations of the scalar field. Therefore, we need to group the zero crossings detected at each input site into meaningful entities.

In 2D, the Marching Squares algorithm can be used to link or order all zero crossings detected on a grid edge in order to produce a curve, which is a 1D entity.

In 3D, the classical Marching Cubes [11] algorithm orders detected zero crossings on the 3D cuboid edges (a total of 12 of them) to form nontrivial cycles. One cycle corresponds to a surface patch, which is a 2D entity. In practice, a surface patch can be detected by checking a precomputed lookup table of all feasible configurations: the set of all zero crossings detected on cuboid edges should *exactly* form a set of cycles without any zero crossing left.

The grouping of zero crossings detected on the hypercuboid edges in a discretized 8D space is analogous. We precompute *once* all feasible configurations (rotationally symmetric counterparts are counted as a single configuration), and store each template configuration as an ordered edge set in a lookup table. Then, the subset of hypercuboid edges with zero crossings (detected as described in Section 3.3.1) are matched against the stored templates. This template matching is very efficient, since a configuration can be quickly discarded by a simple check on the number of edges in the template, followed by the ordering of edges. If a match occurs, then we conclude that a salient hyperplane passes through this 8D site, or equivalently, an inlier is found.

Given the set of inliers found, we estimate the hyperplane normal and F_{33} as follows: The hyperplane normal is the saliency-weighted mean of the normals inferred at all classified inliers. F_{33} is the saliency-weighted mean of the F_{33}'s at all inliers, obtained using the estimated hyperplane normal.

4 Other Issues

In this section, we describe other implementation details which augment the 8D tensor voting system in order to be applied to the estimation of epipolar geometry.

4.1 Data Normalization

A data normalization (translation and scaling) step as described in Hartley [8] is performed. The normalization step is performed independently for the two image pairs. The set of image points on one image, obtained from all potential point matches, is first translated so that their centroid is at the origin. Then, the point coordinates are scaled so that the mean distance from the origin is $\sqrt{2}$. See [8] for the underlying reason on data normalization.

4.2 Scaling

In the epipolar case, the eight dimensions are independent, but not normalized or orthogonal. Since the fundamental stick voting field assumes isotropic and orthogonal dimensions, scaling of the dimensions of the 8D space is needed so that the bounding box of the input is a hypercube (normalized in all dimensions).

The smallest dimension of all the eight dimensions of the bounding box is first computed. Then, a scale factor is computed for the each of the remaining seven dimensions. The resulting bounding box is scaled to a bounding cube with its eight dimensions are equal to the smallest dimension of the box. The scale factor for each dimension can therefore be different.

Note that scaling does not make the spanning axes of the epipolar space orthogonal. But, since the eight dimensions are already independent, the voting kernels should still be used over a wide applicable range of scale. Because of the scaling, the normal inferred at each inlier needs to be rescaled back to the original space before the estimated hyperplane normal and F_{33} are computed.

4.3 Multipass Refinement

To improve accuracy, we need to run several passes to filter the output from the previous pass. The set of classified inliers is progressively refined as more outliers are rejected in each pass. Typically, only four to five passes are needed and the refinement stops when the output inlier set is the same as the input. Since no multidimensional search is involved, a single pass is not very time-consuming. The flow of all the working pieces described in this paper is summarized in Algorithm 4.

It is interesting to note that the multipass tensor voting has some similarities with the Kalman filter, in the sense that we also perform a prediction-correction process, using the notion of covariance matrix. The Kalman filter estimates the state vector of a discrete-time controlled process governed by a linear stochastic difference equation [21]. This estimation process uses a feedback control, which can be understood as a predictor-corrector algorithm, with a set of predictor and corrector equations to implement the

TABLE 2
Summary of the Results on Epipolar Geometry Estimation

		input			classification results and parameter accuracy						
		nb of matches	good matches	bad matches	true inliers	false outliers	true outliers	false inliers	% correct	parameter accuracy	nb of passes
Pentagon	512×512	436	178	258	172	6	252	6	97.25%	1.4809	5
Arena	817×591	486	197	289	196	1	280	9	97.94%	3.3948	2
Gable	534×556	368	174	194	165	9	189	5	96.20%	3.2929	4
House	768×576	48	16	32	14	2	32	0	95.83%	2.3973	5
Game-1	638×219	252	85	167	80	5	163	4	96.43%	1.5401	3
Game-2	643×288	150	95	55	90	5	48	7	92.00%	1.8977	3

projection and update measurement. It is well-known that the Extended Kalman Filter, needed in the nonlinear case, is difficult to implement, and suffers from issues such as convergence problems.

There are two important properties of our approach compared with the Kalman filter. First, we optimize a geometric measure, which is desirable since we are solving for a hyperplane, a geometric problem. Second, a nonlinear update is performed in the "corrector" step, which allows us to reject outliers very efficiently. These properties are largely due to our voting fields: They encode geometric constraints rather than propagating algebraic ones. The geometric constraint we used is the continuity constraint, which is propagated along preferred directions.

Algorithm 4 Epipolar Geometry Estimation

```
while ( $\text{inliers} \neq \text{outliers}$) do
  Normalize data points
  Convert each input pair into an 8D point (Section 1.2)
  Scale and quantize the 8D input point set
  Tensorize each input site (Section 3.1)
  Locally densify each input site (Section 3.2)
  Use 8D extremality for outlier rejection (Section 3.3)
  Rescale the input
  Estimate hyperplane from inliers
  Classify inliers based on inferred hyperplane
end while
Apply the Normalized Eight-Point Algorithm [8]
Enforce the singularity constraint [8]
return fundamental matrix
```

5 Space and Time Complexity

Except for tensorization, local densification, and extrema detection, other processes in Fig. 2 are clearly linear in time and space. Since we use an efficient data structure to store the input, and only local densification is performed, the space complexity of these three processes is also linear, or $O(m)$, where m is the input size.

The time complexity of local densification is only a constant factor of tensorization. The voting field alignment takes $O(1)$ time since it is only a translation followed by a rotation. Therefore, the total time complexity for tensorization and local densification is $O(mk)$, where k is the size of the voting field.

For extrema detection, since there is only a finite number of detected zero crossings and configurations, the total time is $O(m)$.

Therefore, unlike the Hough Transform, the time and space complexities of the 8D voting are independent of the dimensionality. The actual running time for an input of 100 matches is approximately one minute on a Pentium II (450 MHz) processor.

6 Results

We demonstrate the general usefulness of our method on a variety of image pairs. In terms of the accuracy of the estimated parameters, we note that all the methods reported in [22] (M-estimators, LMedS) fail on all of our set of input matches. This is the first quantitative evaluation. We provide a second measure on parameter accuracy in the form of "distance" between the "ground truth" and our estimated fundamental matrices (in pixels). This distance measure is computed by randomly generating points in the images, and computing the mean distance between points and epipolar lines. We use the program *Fdiff* provided by Zhang [22]. The "ground truth" is obtained by using either Zhang's implementation (in the case of aerial image pairs, we use the image pairs, not our noisy matches, as input), or the linear method by manually picked true correspondences. Table 2 summarizes the results of our experiments.

6.1 Aerial Image Pairs

In *Pentagon* (Fig. 9) and *Arena* (Fig. 10) experiments, we add a large number of wrong matches, by hypothesizing *all* pairs within 50 pixels of the correct matches in the corresponding images. We are still able to achieve a high correct percentage despite the large number of wrong matches. The resulting filtered matches are numbered in the corresponding images and a few corresponding epipolar lines are also drawn. In *Gable* (Fig. 11), we have inlier to outlier ratio approximately equals to one, i.e., one match out of two is incorrect. The lighter crosses in Fig. 11 denote classified outliers. The darker crosses, alongside with corresponding matching numbers, indicate the filtered set of good matches. This resulting set of match is passed into the normalized Eight-Point Algorithm. A few corresponding epipolar lines are shown.

To understand the effect of outliers on the distribution of points in 8D in the above examples, and also on the robustness on our 8D hyperplane extraction, let us visualize

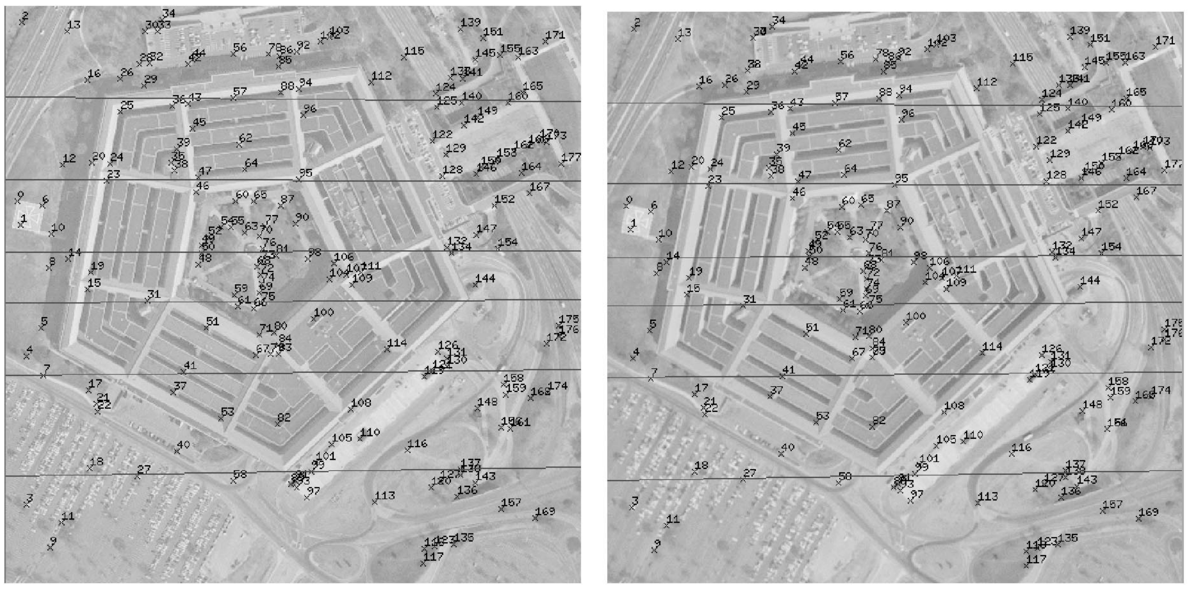
Fig. 9. *Pentagon.*

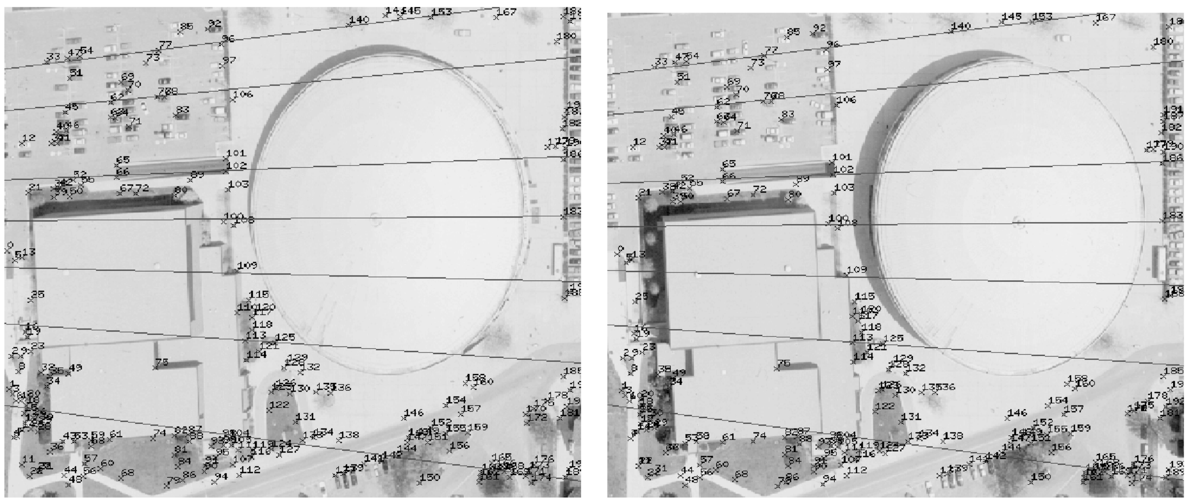
Fig. 10. *Arena.*

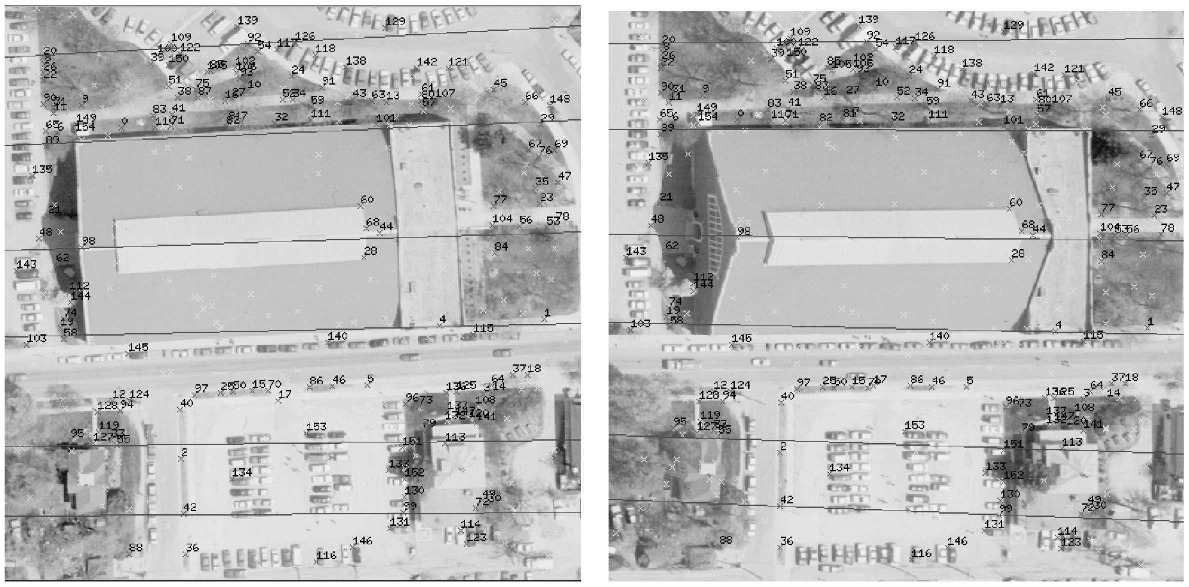
Fig. 11. *Gable.*

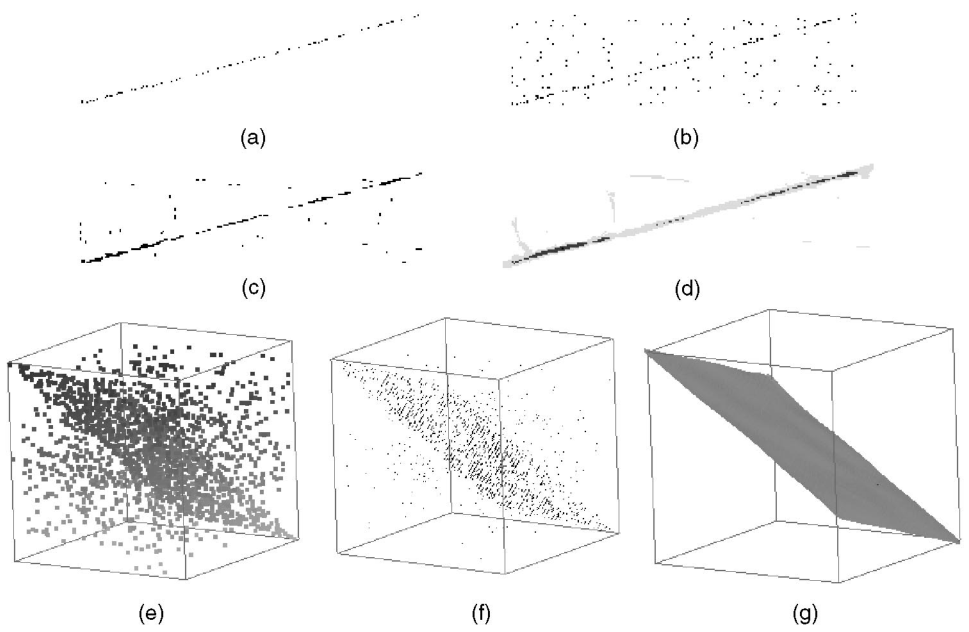

Fig. 12. The effect of manually added outliers on the distribution of points, in 2D and 3D. (a) Original set of 2D points, (b) noisy point set, (c) noise is attenuated after tensor voting, (d) the 2D curve saliency, (e) a noisy point cluster in 3D, (f) most noise is removed after tensor voting, and (g) the extracted planar surface.

the 2D and 3D counterparts, Fig. 12. In the 2D scenario, we have a set of 130 points $(x_i, y_i)^T$ sampled on a straight line, with 260 noises $(x_i, y_j)^T, i \neq j$, and j is randomly generated. Thus, it simulates wrong matches we produced above for *Pentagon* and *Arena*, and allows us to visualize the resulting noisy point distribution in a lower-dimensional space. In 3D, we have 653 points $(x_i, y_i, z_i)^T$ sampled on a plane $(4x + 3y + 5z - 2 = 0)$. A total of 1,306 incorrect data points are added, with coordinates $(x_i, y_i, z_j)^T$, where $i \neq j$, and j is also randomly generated. To make this 3D experiment more challenging, we scale the z by a factor of two, making the grid nonisotropic. Note the interesting observation: Given the same inlier to outlier ratio, noise robustness increases with dimensions. This is due to the nonlinearity in the update step of the tensor voting process, in which the continuity constraint has more dimensions to propagate in the preferred directions.

6.2 Image Pair with Widely Different Views

In the *House* pair (courtesy of A. Zisserman), Fig. 13, two very disparate views of the same static scene are taken. We manually pick only 16 correct matches and then add 32 wrong matches randomly. Our method rejects all outliers and produce the accurate epipolar geometry.

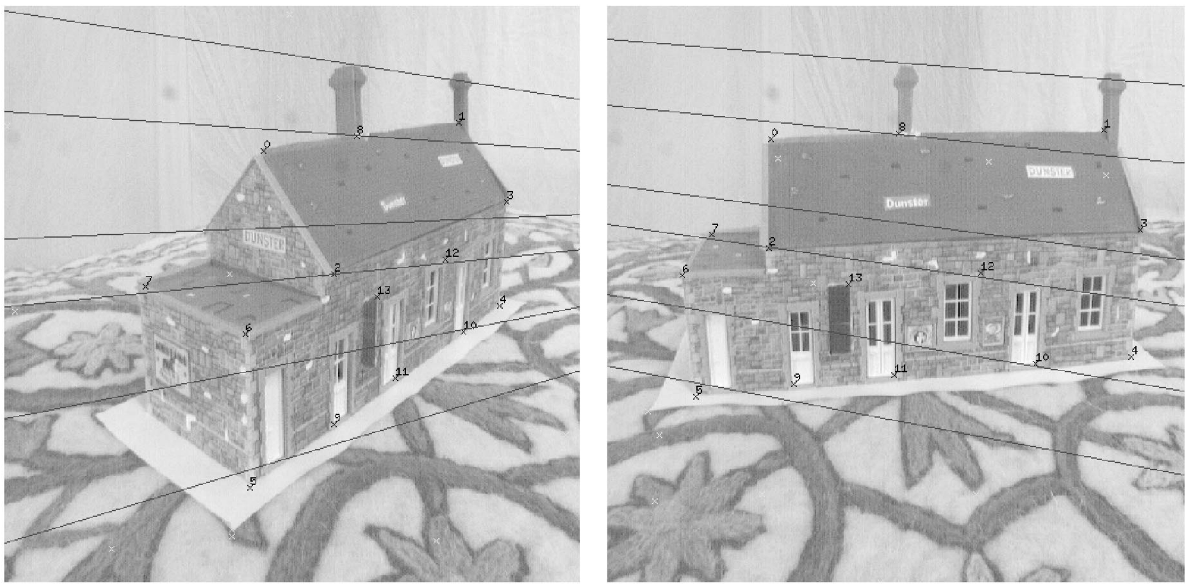

Fig. 13. *House.*

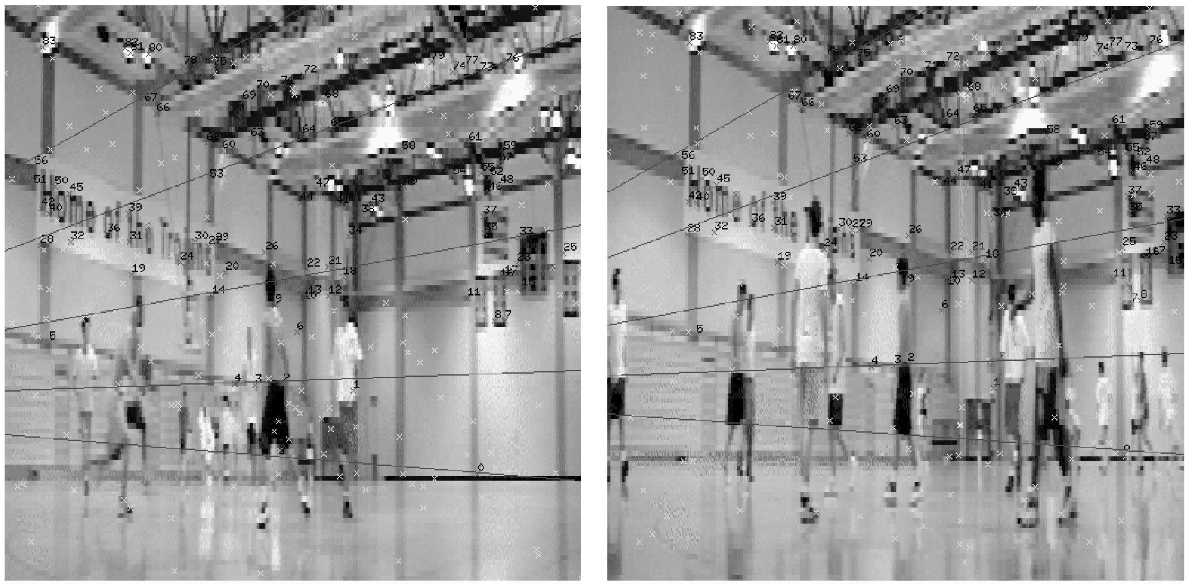

Fig. 14. *Game-1*.

6.3 Image Pairs with Nonstatic Scenes

In the presence of moving objects, image registration becomes a more challenging problem, as the matching and registration phases become interdependent. Most researchers assume a homographic model between images, and detect motion by residual, or more than two frames are used. Torr and Murray [18] use epipolar geometry to detect independent motion. We propose to perform true epipolar geometry estimation for nonstatic scenes by using tensor voting.

Two image pairs, *Game-1* (Fig. 14) and *Game-2* (Fig. 15), of a nonstatic basketball game scene are taken. The background of both image pairs is a 3D static indoor stadium. There is a lot of independent motion due to moving players. This produces many incorrect matches to the already noisy set of matches as given by cross correlation technique. In *Game-2*, we have some additional false matches on moving players. Since our method is designed to detect a *salient* hyperplane (contributed by the 3D background) from a noisy 8D cluster, and, in this case, the outliers are caused by nonstationary agents and their shadows cast on the floor, we should be able to pull out this hyperplane containing the inlier matches. The results of our experiments show that we can indeed discard such wrong matches, retain true matches coming from the static background, stationary players, and the audience. Therefore, we believe that our approach can extract multiple motions, mainly egomotion or possibly motion of large scene objects from an image pair, which is the subject of future research.

7 Conclusion and Future Work

In this paper, we have generalized the tensor voting formalism into any dimensions and described a novel approach to address the problem of outlier detection and removal, in the context of epipolar geometry estimation. The epipolar geometry estimation problem is posed as an 8D hyperplane inference problem. Our method is more efficient than the Hough Transform in high dimensions. The computation and the subsequent use of hyperplane saliency and extremal property in the spatial domain (versus parameter domain for orientation) are novel and effective and are completely different from the Hough Transform. Since the methodology avoids searching in the parameter space, it is free of the problems of local optima and poor

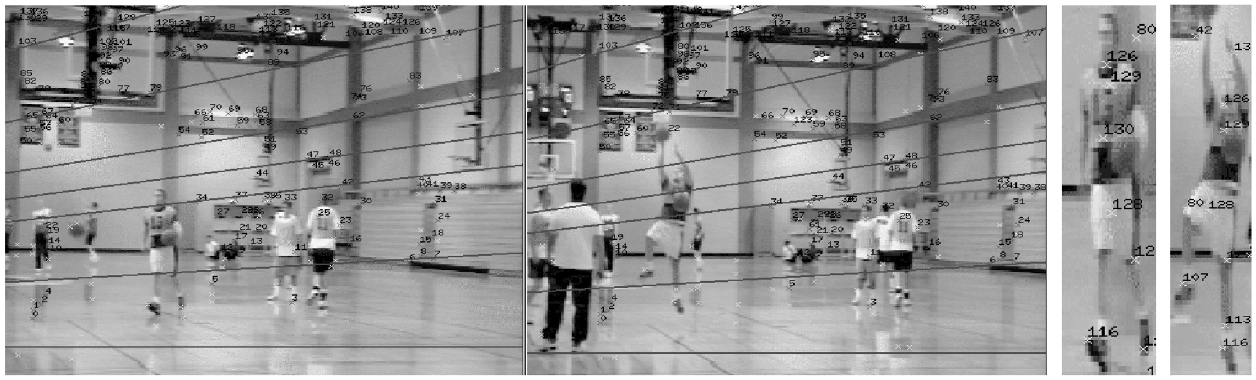

Fig. 15. *Game-2*.

iterative convergence. Our approach is initialization free (i.e., no initial fundamental matrix guess is needed). The pinhole camera model is the only assumption we make. No other simplifying assumption is made about the scene being analyzed. By using an adequate data structure, higher dimensionality translates into a constant factor in processing time. The future work of this research will, in addition to the application of multimotion analysis mentioned in Section 6.3, focus on the investigation of quantization effect, scale of analysis, and other problem domains.

Acknowledgments

This research is supported by the US National Science Foundation under grant no. 9811883 and the Research Grant Council under grant HKUST6246/00E. The authors would like to thank all the reviwers for their constructive comments that helped to improve the final manuscript.

References

[1] E.L. Allgower and S. Gnutzmann, "Simplicial Pivoting for Mesh Generation of Implicitly Defined Surfaces," *Computer Aided Geometric Design,* vol. 8, no. 4, pp. 305-325, 1991.

[2] J. Chai and S.D. Ma, "Robust Fundamental Matrix Estimation from Uncalibrated Images," *Pattern Recognition Letters,* vol. 19, no. 9, pp. 829-838, 1998.

[3] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms.* MIT Press, 1990.

[4] Q. Ke, G. Xu, and S.D. Ma, "Recovering Epipolar Geometry by Reactive Tabu Search," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 767-771, 1998.

[5] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint.* Cambridge, Mass.: MIT Press, 1993.

[6] G. Guy and G. Medioni, "Inference of Surfaces, 3D Curves and Junctions from Sparse, Noisy 3D Data," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 11, pp. 1265-1277, Nov. 1997.

[7] P.V.C. Hough, "Methods and Means for Recognizing Complex Patterns," US Patent 3 069 654, Dec. 1962.

[8] R.I. Hartley, "In Defense of the Eight-Point Algorithm," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 6, pp. 580-593, June 1997.

[9] M.-S. Lee, "Tensor Voting for Salient Feature Inference in Computer Vision," Ph.D. thesis, Univ. of Southern California, 1998.

[10] H.C. Longuet-Higgins, "A Computer Algorithm for Reconstructing a Scene from Two Projections," *Nature,* vol. 293, pp. 133-135, 1981.

[11] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm," *Computer Graphics,* vol. 21, no. 4, 1987.

[12] G. Medioni, M.-S. Lee, and C.-K. Tang, *A Computational Framework of Feature Extraction and Segmentation.* Elsevier Science, 2000.

[13] G.M. Nielson and B. Hamann, "The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes," *Proc. IEEE Visualization Conf.,* pp. 83-90, 1991.

[14] P. Pritchett and A. Zisserman, "Wide Baseline Stereo Matching," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 754-760, 1998.

[15] C.-K. Tang and G. Medioni, "Inference of Integrated Surface, Curve, and Junction Descriptions from Sparse, Noisy 3D Data," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 11, pp. 1206-1223, Nov. 1998.

[16] C.-K. Tang, G. Medioni, and M.-S. Lee, "Epipolar Geometry Estimation by Tensor Voting in 8D," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 502-509, 1999.

[17] C.-K. Tang, "Tensor Voting for Feature Extraction, Integration, and Higher Dimensional Inference," PhD thesis, Univ. of Southern California, 2000.

[18] P.H.S. Torr and D.W. Murray, "Statistical Detection of Independent Movement from a Moving Camera," *Image and Vision Computing,* vol. 1, no. 4, 1993.

[19] P.H.S. Torr and D.W. Murray, "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix," *Int'l J. Computer Vision,* vol. 24, no. 3, pp. 271-300, 1997.

[20] C. Weigle and D.C. Banks, "Complex-Valued Contour Meshing," *Proc. IEEE Visualization Conf.,* pp. 173-179, 1996.

[21] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," Technical Report TR 95-041, Univ. of North Carolina at Chapel Hill. 2001. http://www.cs.unc.edu/~welch/kalman/kalman_filter/kalman.html.

[22] Z. Zhang, "Determining the Epipolar Geometry and Its Uncertainty: A Review," *Int'l J. Computer Vision,* vol. 27, no. 2, pp. 161-195, 1998.

Chi-Keung Tang received the MS and PhD degrees in computer science from the University of Southern California (USC), Los Angeles, in 1999 and 2000, respectively. He has been with the Computer Science Department at the Hong Kong University of Science and Technology since 2000, where he is currently an assistant professor. He is also an adjunct researcher at the Visual Computing Group of Microsoft Research, China, working on various exciting research topics in computer vision and graphics. At USC, he had been working with Mi-Suen Lee, Professor Medioni, and his colleagues on *Tensor Voting*, a computational framework for segmentation and feature extraction. His research interests include low- to mid-level vision such as segmentation, correpondence, shape analysis, and vision and graphics topics such as image-based rendering and medical image analysis. He is a member of 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 and codirector of the Computer Vision Laboratory. 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 senior member of the IEEE. He has served on the program committees of many major vision conferences and was program cochairman of the 1991 IEEE Computer Vision and Pattern Recognition Conference in Maui, program co-chairman 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 co-chair of the 1998 International Conference on Pattern Recognition held in Brisbane, Australia, and general cochairman of the upcoming 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.

Mi-Suen Lee received the BSc degree from the Chinese University of Hong Kong in 1989, the MPhil degree from The University of Hong Kong in 1992, and the PhD degree from The University of Southern California in 1998, all in computer science. Since 1998, she has been a senior member of research staff at Philips Research-USA. Her current research interests include segmentation, perceptual grouping, robust techniques, shape analysis, and image-based rendering. She is a member of the IEEE Computer Society.