

Epipolar Geometry Estimation by Tensor Voting in 8D*

Chi-Keung Tang, Gérard Medioni
Institute for Robotics & Intelligent Systems, USC
Los Angeles, CA 90089
{chitang,medioni}@iris.usc.edu

Mi-Suen Lee
Philips Research
Briarcliff Manor, NY 10510
msl@philabs.philips.com

Abstract

We present a novel, efficient, initialization-free approach to the problem of epipolar geometry estimation, 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 as obtained from two views of a static scene without correspondences, even in the presence of moving objects, our method pulls out inlier matches while rejecting outliers. Unlike most methods which optimize certain objective function, our approach does not involve initialization or any search in the parameter space, and therefore is free of the problem of local optima or poor convergence. 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 establishing salient “extremalities” via a novel approach, **8D Tensor Voting**: the input set of matches is first transformed into a sparse and discrete 8D point set. Dense tensor kernels are then applied to vote for the most salient hyperplane (normal and intercept) that captures all inliers inherent in the input. With this filtered set of matches, the normalized Eight-Point Algorithm suffices for the accurate estimation of the fundamental matrix. By using 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 (i) for aerial image analysis, (ii) with widely different views, and (iii) from non-static 3D scenes (e.g. basketball game in an indoor stadium). Each example contains a considerable amount of wrong matches.

1 Introduction

Epipolar geometry is a fundamental constraint used in computer vision 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 unavoidable *outliers* inherent in the given matches. Most robust techniques require that the majority of matches to be correct, or else some form of outlier detection and removal is usually performed before the actual parameter estimation.

Both the outlier detection and parameter estimation are often formulated as a non-linear optimization and search process

in the parameter space. In the case of full perspective camera model, this search space can be prohibitively large. Consequently, gradient-based and other non-linear heuristic search techniques have been proposed. The output, however, may not be initialization free, and poor initialization may seriously affect convergence rate. Simplifying assumptions, such as affine camera model and local planar homography [9], can drastically reduce the search complexity, but somewhat restrict the class of transformations which can be represented.

In fact, the general epipolar constraint is a linear and homogeneous one that defines an 8D hyperplane in its parameter space (section 1.2). Given a candidate set of (possibly noisy) matches, if we could *visualize* in 8D, the subset of inlier matches should cluster onto a hyperplane in the corresponding 8D space. Thus, analogous to line detection in 2D, we can pose this outlier detection problem as one of hyperplane inference. It is of great value and theoretical interest if we can pull out this *salient* hyperplane feature from the given sparse and noisy 8D cluster, without performing any iterative or multidimensional search. Simplifying assumption will become unnecessary as the search space is no longer an issue.

An intuition to a non-iterative solution for hyperplane inference is inspired by the Hough Transform [4]. It employs a *voting* technique that produces the solution receiving maximal support. However, as the dimensionality grows, the Hough Transform is extremely inefficient, and thus is impractical in most higher dimensional detection problems.

Therefore, the contributions of this paper is twofold: A formal framework, *8D tensor voting*, is proposed. It is extended from the 2D version [6], and modified for hyperplane detection. 8D 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 secondary contribution is the application of this hyperplane inference technique to outlier detection and removal for general epipolar geometry estimation, and we demonstrate the general usefulness of our method with a variety of image pairs.

1.1 Previous Work

If the input set of correspondences is already very good, then, the *linear method* such as Eight-Point Algorithm [7], can be used for accurate parameter estimation. This algorithm is probably the most cited method for computing the essential (resp. fundamental) matrix from two calibrated (resp. uncalibrated) camera images. With more than eight points, a least

*This research is supported by the National Science Foundation under grant no. 9811883.

mean square minimization is used, then followed by the enforcement of the singularity constraint. Its obvious advantages are speed and ease of implementation. However, in practice, the input set of matches contains a considerable amount of outliers. More complicated, iterative optimization methods are proposed, some of them are described in [12]. These non-linear *robust methods* use 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. In particular, the method proposed by Zhang et al. [12] use 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. Note that robust methods require that a majority of the data to be correct, whereas we can tolerate much higher outlier to inlier ratio, as shown in the result section.

Torr and Murray [11] propose 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 points and minimizes the residual. It is, however, computationally infeasible to consider all possible subsets, which are 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.

Chai and Ma [1] propose the use of genetic algorithm to avoid the problem of local minima, by proper definition of genetic operators. The optimization process can be sped up through incorporating the ideas of evolution that properly guides the search process.

Hartley [5] normalizes the data before using the Eight-Point Algorithm, and shows that this normalized version performs comparably with more complicated iterative techniques. Outlier rejection is performed before the algorithm is used.

In [9], Pritchett and Zisserman propose the use of *local planar homography* (plane 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 the epipolar geometry (more details in [2]), and formulate the estimation problem as one of 8D hyperplane inference. Given two images of a static scene taken from two camera systems, 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 line is the intersection of two planes: one plane is defined by three points: the two optical centers and (u_ℓ, v_ℓ) , and the other plane by the im-

age plane of the second image. 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 $\mathbf{u}_2 = (u_r, v_r, 1)^T$. This equation can be written as a linear and homogeneous equation in terms of the 9 unknown coefficients in \mathbf{F} , or $\mathbf{u}^T \mathbf{f} + F_{33} = 0$, where

$$\begin{aligned} \mathbf{u} &= [u_\ell u_r \quad v_\ell u_r \quad u_r \quad u_\ell v_r \quad v_\ell v_r \quad v_r \quad u_\ell \quad v_\ell]^T \\ \mathbf{f} &= [F_{11} \quad F_{12} \quad F_{13} \quad F_{21} \quad F_{22} \quad F_{23} \quad F_{31} \quad F_{32}]^T \end{aligned}$$

which exactly defines a hyperplane equation in the 8D space. Each vector \mathbf{u} is a point in an 8D space defined by a match $\mathbf{u}_1 \leftrightarrow \mathbf{u}_2$. The hyperplane normal (given by \mathbf{f}) and intercept (given by F_{33}) are to be estimated using 8D voting.

1.3 Our Approach and Outline of this Paper

Figure 1 shows our overall approach, in which the blue processes are implemented as 8D voting processes (described in section 2). The input set of point matches, obtained by automatic means such as cross-correlation technique, is first converted into a sparse 8D point set as described in the previous subsection. This point set is quantized and “*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 normal and intercept of the salient hyperplane containing all good matches can be estimated. Extrema detection (the pink process) is detailed in section 3. The input match is then checked against the inferred hyperplane, and a set of filtered inliers is produced. This process and other issues pertinent to epipolar geometry are described in section 4. Finally, the normalized eight-point algorithm (least mean square minimization followed by 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.

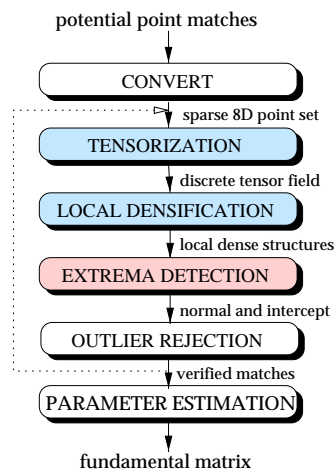


Figure 1 8D tensor voting approach to the epipolar estimation problem.

In essence, the basis of our method is grounded on two elements: local structures are uniformly represented by a second order symmetric **tensor**, which effectively encodes preferred direction. Data communication is accomplished by a **voting**

process, which simultaneously ignores outlier noise and corrects erroneous orientation (if given). The use of a voting process for feature inference from sparse and noisy data was introduced by Guy and Medioni [3], and then formalized into a unified tensor framework [6]. This methodology is non-iterative and robust to considerable amount of outlier noise. The only free parameter is the scale of analysis, which is indeed a property of human perception. Input data is made to align (votes) with precomputed dense tensor kernels, by propagating preferred information in a neighborhood. This preferred information includes proximity, smoothness, and continuity.

2 8D Tensor Voting

We describe tensorization and local densification (the blue processes in Figure 1) in this section. The 2D tensor voting formalism [6] can be generalized to 8D readily, except for some implementation changes, mainly for efficiency purposes. For this reason, we shall relate the 2D version to facilitate our 8D discussion. Table 1 summarizes the key generalization components. In this section, we focus on the multidimensional version of voting as follows:

- Representation and interpretation
- Voting kernels
- Tensorization
- Local densification

	2D	8D
feature to be inferred	line	hyperplane
Representation	ellipse	hyperellipsoid
size of eigensystem	2×2	8×8
stick basis	absolute certainty in <i>tangent</i> orientation along a single 2D direction	absolute certainty in <i>normal</i> direction along a single 8D direction
ball basis	absolute uncertainty in all 2 directions	absolute uncertainty in all 8 directions
quantization unit	2D grid (i, j)	8D hypercube (i_1, i_2, \dots, i_8)
data structure for vote collection	2D grid	8D red-black tree (linearized)
neighborhood	grid distance = 1	Hemming distance = 1
local extremal feature	line segment	hypersurface patch
discontinuities	intersection of lines (junctions)	intersection of hypersurfaces

Table 1 Generalization of 2D tensor voting to 8D.

2.1 Representation and Interpretation

Suppose we perform line detection in a 2D point cluster. A point in this cluster either belongs to a *line*, a *point junction* where intersecting lines meet, or is an *outlier*. Consider the two extremes, in which a point on a line is very certain about its *tangent* orientation, whereas a point junction at which many lines intersect has absolute orientation uncertainty.

This whole continuum can be abstracted as a second order symmetric 2D *tensor*, or geometrically as an **ellipse** (Figure 2). This ellipse can be described by the equivalent 2×2 eigensystem, with its two unit eigenvectors \hat{V}_{max} and \hat{V}_{min} , and the two

corresponding eigenvalues $\lambda_{max} \geq \lambda_{min}$, which is given by:

$$(\lambda_{max} - \lambda_{min})\mathbf{S} + \lambda_{min}\mathbf{B}, \quad (1)$$

where $\mathbf{S} = \hat{V}_{max}\hat{V}_{max}^T$ defines a *stick tensor*, and $\mathbf{B} = \hat{V}_{max}\hat{V}_{max}^T + \hat{V}_{min}\hat{V}_{min}^T$ defines a *ball tensor*, in 2D. These tensors define the two *basis tensors* for any 2D ellipse. Now, every point in this cluster is represented as tensor.

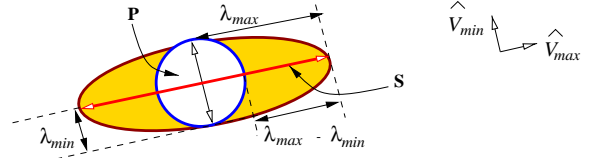


Figure 2 A second order symmetric 2D tensor.

Suppose we already have this tensor field, and we decompose it into the corresponding eigensystem. The *eigenvectors* thus encode *orientation (un)certainties*: *tangent* orientation 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*, on the other hand, effectively encode the *magnitude* of orientation (un)certainties, since they indicate the size of the corresponding ellipse. Hence, after the eigensystem analysis, two *saliency maps* are produced. Each point in these fields has a 2-tuple (s, \bar{v}) , where s is a scalar indicating feature *saliency* or strength, and \bar{v} is a unit vector indicating *direction*: (1) *Line saliency map*: $s = \lambda_{max} - \lambda_{min}$, and $\bar{v} = \hat{V}_{max}$ indicates the tangent direction, and (2) *Junction saliency map*: $s = \lambda_{min}$, \bar{v} is arbitrary.

Analogously, for 8D hyperplane detection, a point in the 8D space can either be: on *hyperplane* (planar smooth), at a *discontinuity* (junction), or is an *outlier*. Consider the two extremes: an 8D point on a hyperplane is very certain about its *normal* orientation, whereas a point at a junction has absolute orientation uncertainty. As in 2D, this whole continuum can be abstracted as a second order symmetric 8D tensor, or equivalently a **hyperellipsoid**. This hyperellipsoid can be equivalently described by the corresponding eigensystem with its eight eigenvectors $\hat{V}_1, \hat{V}_2, \dots, \hat{V}_8$ and the eight corresponding eigenvalues, $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_8$. Rearranging the 8×8 eigensystem, the 8D ellipsoid is given by:

$$(\lambda_1 - \lambda_2)\mathbf{S} + \sum_{i=2}^7 (\lambda_i - \lambda_{i+1}) \prod_{j=1}^i \hat{V}_j \hat{V}_j^T + \lambda_8 \mathbf{B}, \quad (2)$$

where $\mathbf{S} = \hat{V}_1 \hat{V}_1^T$ and $\mathbf{B} = \sum_{i=1}^8 \hat{V}_i \hat{V}_i^T$ defines an 8D *stick* and *ball*, respectively. Like the 2D version, the *eigenvectors* effectively encode *orientation (un)certainties*: *hyperplane* orientation (normal) is described by the *stick tensor*, which indicates *certainty* along a single, 8D direction. *Orientation uncertainty* is indicated by the *ball tensor*, where many intersecting hyperplanes are present and thus no single orientation is preferred. Again, the *eigenvalues* encode the *magnitudes* of orientation (un)certainties. This 8×8 eigensystem gives eight saliency maps $\{(s, \bar{n})\}$, where s indicates saliency and \bar{n} denotes direction. In the case of hyperplane inference for the epipolar geometry problem, we are only interested in the **surface saliency**

map (SMap), where surface saliency is $s = \lambda_1 - \lambda_2$, and normal direction is $\bar{n} = \hat{V}_1$.

Having defined the tensor formalism, we are now ready to describe the *voting* algorithm for obtaining the tensor representation that gives the above saliency maps. Each input token **votes**, or is made to align (by translation and rotation), with predefined, discrete versions of the two basis tensors (we shall call them voting *kernels*) in a convolution-like way. As a result of voting, preferred orientation information is propagated and gathered at each input site. We describe the design of voting kernels next, the key components used in voting.

2.2 The Design of 8D Voting Kernels

The design of the 8D voting kernels is analogous to its 2D counterpart. In the following derivation, since all vectors lie on a same hyperplane, w.l.o.g., refer to Figure 3 that shows a 2D depiction of a voting scenario, in which analogously normals (instead of tangents) are propagated as preferred directions. Suppose we have a normal N at the origin O , and a point P receiving vote from O . A *hyperspherical* connection is postulated as the “most likely” surface continuation between O and P , because this captures constraints such as smoothness and constancy of curvature. We say that O with its 8D normal N has cast a vote at P . This vote consists of an 8D stick tensor (a unit vector) indicating the preferred normal *direction*, and a scalar indicating its surface *saliency*. The direction is given by the 8D normal of the hypothetical hypersphere at P (i.e., the thick arrow). The saliency is attenuated by a Gaussian decay:

$$DF(P) = e^{-\left(\frac{s^2 + cp^2}{\sigma^2}\right)} \quad (3)$$

where s is the arc length connecting O and P (therefore, s encodes proximity), ρ is the curvature, c is a constant (chosen *a priori*, not a parameter) that controls the amount of attenuation with higher curvature, and σ is the scale of analysis (the only parameter). The set of all such votes in the 8D space is quantized and collected as the **stick kernel**. (Here, the curvature is represented by the angle subtended by the radii of the hypersphere incident at O and P .)

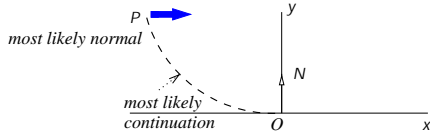


Figure 3 The design of the stick kernel.

Absolute uncertainty in normal orientation is represented by an 8D **ball tensor**. It can be derived *directly* from the stick kernel, by complete rotations along the seven principal directions (the remaining one has been fixed by the stick kernel). Thus, its shape is radiating with diminishing (Gaussian-decayed) strength in every direction. Please refer to [6] for a mathematical treatment of the analogous 2D and 3D cases. Figure 4 shows the projections of the stick and ball kernels.



Figure 4 Projections of (a) stick and (b) ball kernels.

2.3 Tensorization

Using these voting kernels, we tensorize the quantized input point set into a discrete tensor field, which encodes the “most likely” hyperplane normal direction at each input point.

First, each input site is *tensorized* into an 8D *ball* tensor, because initially there is no preferred information. We place the isotropic ball kernel at each input site in the volume. In doing so, each site casts ball votes to other sites in a neighborhood defined by $DF(\cdot)$. Directed votes received at each input site are collected, using tensor addition, as an 8×8 covariance matrix of the second order moment collection. A tensor field is thus produced.

This resulting tensor field then votes with the 8D *stick* kernel, by aligning with this kernel (by translation and rotation). Each site receives stick votes cast by other sites in a neighborhood defined by $DF(\cdot)$. Directed votes are collected using tensor addition, again as an 8×8 covariance matrix of the second order moment collection. The tensor voting formalism described in section 2.1 is then applied. Note that in tensorization, only votes received at *input sites* are collected.

2.4 Local Densification

After the input has been tensorized, the stick component at each input tensor is made to align with the stick kernel again for obtaining a densified structure SMap $\{(s, \bar{n})\}$ as defined in section 2.1, which is used for extrema detection (section 3) that discard outliers. This voting process, is *exactly* the same as tensorization, except that directed votes are also collected at non-input sites in the volume. The same tensor voting formalism also applies here. The only implementation differences are described in the following.

For 2D line extraction, we can afford to densify the whole 2D domain, i.e., votes are cast and collected *everywhere* (Figure 5(a)). Or, more efficiently, since we have obtained saliency information after tensorization, 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 (Figure 5(b)) during the extraction process.

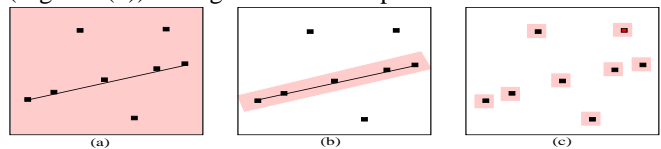


Figure 5 (a) vote casting densifies everywhere; in vote gathering, we compute (b) a slab of votes only, or even perform (c) local densification.

In this 8D version, two implementation differences are made to avoid the drastic increase in time and space complexities owing to the higher dimensionality.

For time efficiency, we do *not* even need to compute a slab of votes since we are performing *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, *local densification* (Figure 5(c)) suffices: a small volume centered at each input site *gathers* all the

stick votes cast within the neighborhood (defined by $DF(\cdot)$), performs smoothing, computes the eigensystem, interprets the vote, and produces a hypersurface patch (if any) for that site, 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 the input is quantized, we use a “linearized 8D red-black tree” (simply an ordinary red-black tree, but we concatenate the quantized (integer) coordinates i ’s as the search key) to store each input site. This data structure is only of size $O(n)$, where n is the input size, much smaller than a whole 8D array.

3 Extrema Detection and Outlier Rejection

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

3.1 8D Extremality

To illustrate, we suppose the local dense structure as obtained after densification is continuous (the discrete version for implementation will be described next), i.e. $\{(s, \bar{n})\}$ is defined for every point p in it. Imagine that we could traverse and look at the s values along the 8D vector \bar{n} (Figure 6(a)). By the definition of the stick kernel, after voting, an extrema (or maxima) in s (Figure 6(b)) should be observed if this input point lies on a salient hyperplane in the 8D point cluster. Therefore, we define *extremal hypersurface* as the locus of points for which the saliency s is locally extremal along the direction of the 8D normal, i.e., $\frac{ds}{d\bar{n}} = 0$. This is only a necessary condition for 8D extremality. The sufficient condition, which is used in the implementation, is defined in terms of *zero crossings* along the line defined by \bar{n} (Figure 6(c)). We therefore compute the saliency gradient $\bar{g} = \nabla s = \left[\frac{\partial s}{\partial x_1} \quad \frac{\partial s}{\partial x_2} \quad \dots \quad \frac{\partial s}{\partial x_8} \right]^T$, and then project \bar{g} onto \bar{n} , i.e., $q = \bar{n} \cdot \bar{g}$. Thus, *an extremal hypersurface is the locus of points with $q = 0$* . This definition is analogous to the 3D version in [3].

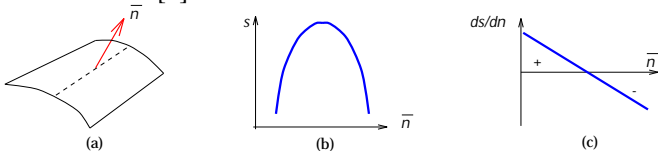


Figure 6 (a) an 8D normal (with an imaginary patch drawn), (b) saliency along normal, and (c) the derivative.

We have discrete $\{(s_{i_1, i_2, \dots, i_8}, \bar{n}_{i_1, i_2, \dots, i_8})\}$ in implementation. We can define the corresponding discrete versions of \bar{g} and q , and $q_{i_1, i_2, \dots, i_8} = \bar{n}_{i_1, i_2, \dots, i_8} \cdot \bar{g}_{i_1, i_2, \dots, i_8}$. Given an input point, we compute q_{i_1, i_2, \dots, i_8} at each vertex voxel (a total of $2^8 = 256$ that makes up the hypercube quantization unit containing that input point). Thus, the set of all $\{q_{i_1, i_2, \dots, i_8}\}$ constitutes a scalar field. If the signs of the q ’s of two adjacent vertex voxels are differ-

ent, a *zero crossing* occurs on the corresponding hypercuboid edge (there is a total of 1024 of them).

3.2 Grouping of Detected Zero Crossings

Mere existence of zero crossings does not necessarily imply the presence of a salient hyperplane, because outlier noise can produce local perturbation of the scalar field. Therefore, as in 2D and 3D cases, 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 (order) all zero crossings detected on a grid edge to produce a curve (1D entity).

In 3D, the classical Marching Cubes [8] algorithm orders the detected zero crossings on the 3D cuboid edges (a total of 12) to form non-trivial cycles, or surface patches (2D entities). It can be done by a lookup table of all *feasible configurations* (i.e., the set of all zero crossings detected on cuboid edges should *exactly* form a set of cycles without any “dangling” zero crossing left).

The grouping of zero crossings detected on the hypercuboid edges in a discretized 8D space is analogous to the 3D case: 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 (hypercycle) in a lookup table. Then, the subset of hypercuboid edges with zero crossings (detected as described in section 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 intercept as follows: the hyperplane normal is the saliency-weighted mean of the normals inferred at all classified inliers. The hyperplane intercept is the saliency-weighted mean of the intercepts at all inliers, obtained using the estimated hyperplane normal.

4 Other Issues

DATA NORMALIZATION. A data normalization (translation and scaling) step as described in [5] is performed.

SCALING. In the epipolar case, the eight dimensions are independent, but not normalized or orthogonal. Because the tensor voting formalism assumes isotropic dimensions, dimension scaling is needed so that the bounding box of the input is a hypercube (normalized in all dimensions). Note that scaling does not make the epipolar space orthogonal. But, since the eight dimensions are already independent, the voting kernels should still be used over a wide applicable range. Because of the scaling, the normal inferred at each inlier needs to be rescaled back before the estimated hyperplane normal and intercept are computed.

PROGRESSIVE 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 4 to

5 passes are needed, and the refinement stops when the output inliers is the same as the input. Since no multidimensional search is involved, a single pass is not very time-consuming.

5 Space and Time Complexity

Except for tensorization, local densification, and extrema detection, other processes in Figure 1 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 $O(n)$, where n is the input size. The time complexity of *local* densification is only a constant factor that of tensorization. In the *worse* case, the neighborhood of an input site is the input itself. Kernel alignment takes $O(1)$ time since it can be implemented as a table-lookup operation. Therefore, the total time complexity for tensorization and local densification is $O(n^2)$. For extrema detection, since there is only a finite number of detected zero crossings and configurations, the total time is $O(n)$. Table 2 gives the summary on complexities. Therefore, in contrast to the Hough Transform, the time and space complexities of the 8D voting are *independent* of the dimensionality. The running time for an input of 100 matches is approximately 1 min on a Pentium II (450 MHz) processor.

	time complexity	space complexity
Tensorization	$O(n^2)$	$O(n)$
Local densification	$O(n^2)$	$O(n)$
Extrema detection	$O(n)$	$O(n)$

Table 2 Space and time complexities.

6 Results

We demonstrate the general usefulness of our method by experimenting with a variety of image pairs. In terms of the accuracy of the estimated parameters, we note that all the methods reported in [12] (M-estimators, LMedS) fail on all of our set of input matches. This is the first quantitative evaluation. We provide a second one 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 [12]. 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 3 summarizes the results of our experiments.

6.1 Aerial Image Pairs

In *Pentagon* and *Arena* experiments (Figure 7), we add a large amount of outliers, by hypothesizing *all* pairs within 50 pixels of the correct matches in the corresponding images. We are still able to achieve high correct percentage despite the large number of wrong matches. The resultant filtered matches are numbered in the corresponding images, and a few corresponding epipolar lines are also drawn. In *Gable* (Figure 7), we have approximately 100% noisy matches, i.e., one match out of two is incorrect. The yellow crosses in Figure 7 denote classified outliers. The purple 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.

6.2 Image Pair with Widely Different Views

In the *House* pair (courtesy of Zisserman), Figure 8, two very disparate views of the same static scene are taken. We manually pick only 16 matches and then add 32 wrong matches. Our method rejects all outliers and produce the accurate epipolar geometry.

6.3 Image Pairs with Non-Static 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 [10] use epipolar geometry to detect independent motion. Here, we propose to perform true epipolar geometry estimation for non-static scenes using tensor voting in 8D.

Two image pairs, *Game-1* (Figure 8) and *Game-2* (Figure 9), of a non-static basketball game scene are taken. The background of both image pairs is a 3D static indoor stadium. There is independent motion due to moving players. This produces as much as 100% additional wrong matches to the already noisy set of matches due to moving players, as given by cross correlation technique. 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 non-stationary agents and their shadows cast on the floor, we should be able to pull out this hyperplane containing the inlier matches. In *Game-2* we have some additional false matches on moving players. The results of our experiments show that we can indeed discard such wrong matches, retain true matches arisen from the static background, stationary players and audience. Therefore, we believe that our approach can extract multiple motions, mainly egomotion or possibly motion of large scene objects from an image pair (the subject of future research).

7 Conclusion and Future Work

In this paper, we describe a novel approach to address the problem of outlier detection and removal, in the context of epipolar geometry estimation. The epipolar geometry 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 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 problems of local optima and poor 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 assumptions are made to the scene being analyzed. By using adequate data structure, higher dimensionality translates into a constant factor in processing time.

		input				classification results and parameter accuracy						nb of passes
		nb of matches	good matches	bad matches	% noise	true inliers	false outliers	true outliers	false inliers	% correct	parameter accuracy	
<i>Pentagon</i>	512 × 512	436	178	258	144.94%	172	6	252	6	97.25%	1.4809	5
<i>Arena</i>	817 × 591	486	197	289	146.70%	196	1	280	9	97.94%	3.3948	2
<i>Gable</i>	534 × 556	368	174	194	111.49%	165	9	189	5	96.20%	3.2929	4
<i>House</i>	768 × 576	48	16	32	200.00%	14	2	32	0	95.83%	2.3973	5
<i>Game-1</i>	638 × 219	252	85	167	196.47%	80	5	163	4	96.43%	1.5401	3
<i>Game-2</i>	643 × 288	150	95	55	57.89%	90	5	48	7	92.00%	1.8977	3

Table 3 Summary of results on epipolar geometry estimation.

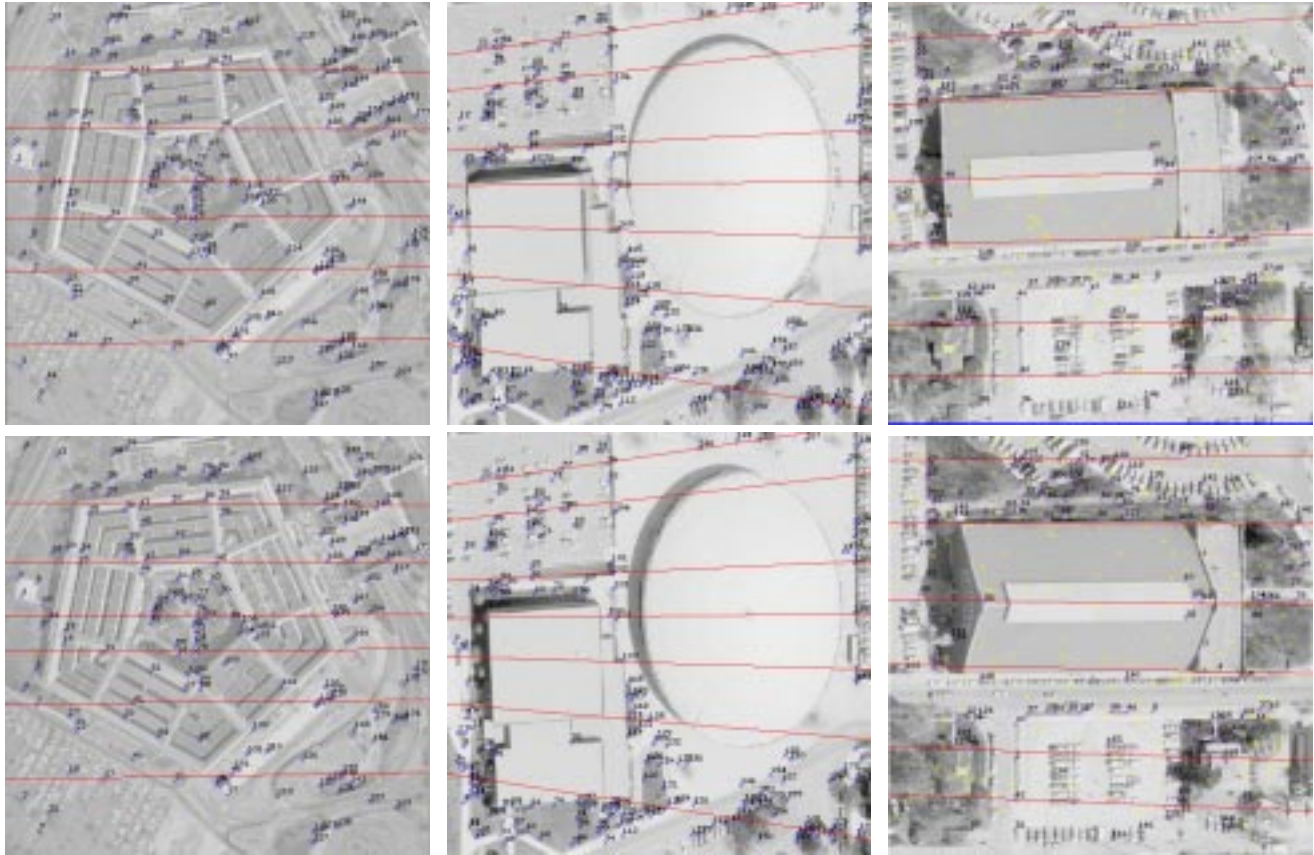


Figure 7 Left to right: Pentagon, Arena, and Gable pairs.

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 the application of this multidimensional voting approach for general hypersurface extraction, and possible applications to other problem domains.

References

- [1] J. Chai and S.D. Ma, "Robust Fundamental Matrix Estimation from Uncalibrated Images," *Pattern Recognition Letters*, vol. 19, no. 9, pp. 829–838, 1998.
- [2] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*. Cambridge, Mass. MIT Press, 1993.
- [3] G. Guy and G. Medioni, "Inference of Surfaces, 3-D Curves and Junctions from Sparse, Noisy 3-D Data," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 19, no. 11, pp. 1265–1277, 1997.
- [4] P.V.C. Hough, "Methods and Means for Recognizing Complex Patterns," U.S. Patent 3 069 654, Dec 1962.
- [5] R.I. Hartley, "In Defense of the Eight-Point Algorithm," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 19, no. 6, pp. 580–593, 1997.
- [6] M.-S. Lee, "Tensor Voting for Salient Feature Inference in Computer Vision," Ph.D. Thesis, University of Southern California, 1998.
- [7] H.C. Longuet-Higgins, "A Computer Algorithm for Reconstructing a Scene from Two Projections," *Nature*, vol. 293, pp. 133–135, 1981.
- [8] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm", *Computer Graphics*, 21(4), 1987.
- [9] P. Pritchett and A. Zisserman, "Wide Baseline Stereo Matching," in *Proc. IEEE Int'l Conf. Comput. Vision (Bombay, India)*, pp. 754–760, 1998.
- [10] P.H.S. Torr and D.W. Murray, "Statistical Detection of Independent Movement from a Moving Camera," *Image and Vision Comput.*, vol. 1, no. 4, 1993.
- [11] P.H.S. Torr and D.W. Murray, "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix," *Intl J. Comput. Vision*, vol. 24, no. 3, pp. 271–300, 1997.
- [12] Z. Zhang, "Determining the Epipolar Geometry and its Uncertainty: A Review," *Int'l J. Comput. Vision*, vol. 27, no. 2, pp. 161–195, 1998.

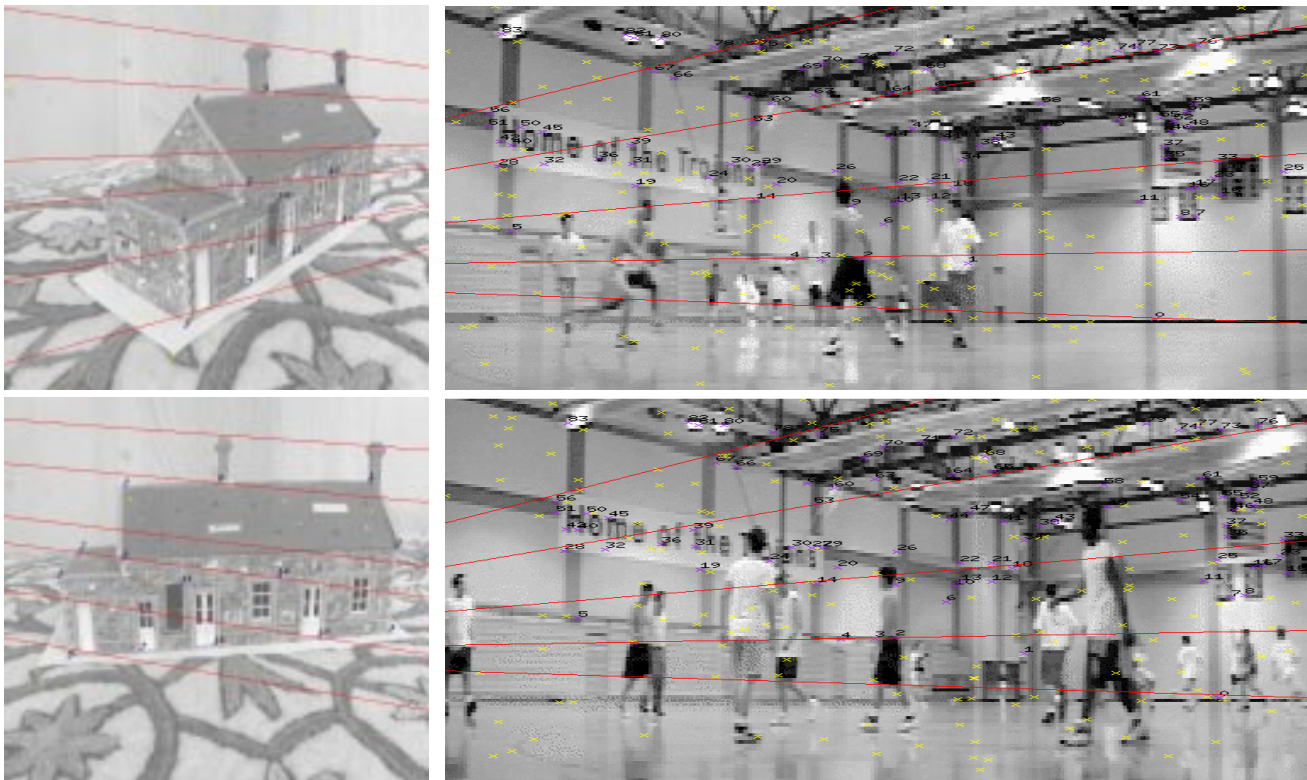


Figure 8 Left to right: House and Game-1 pairs.

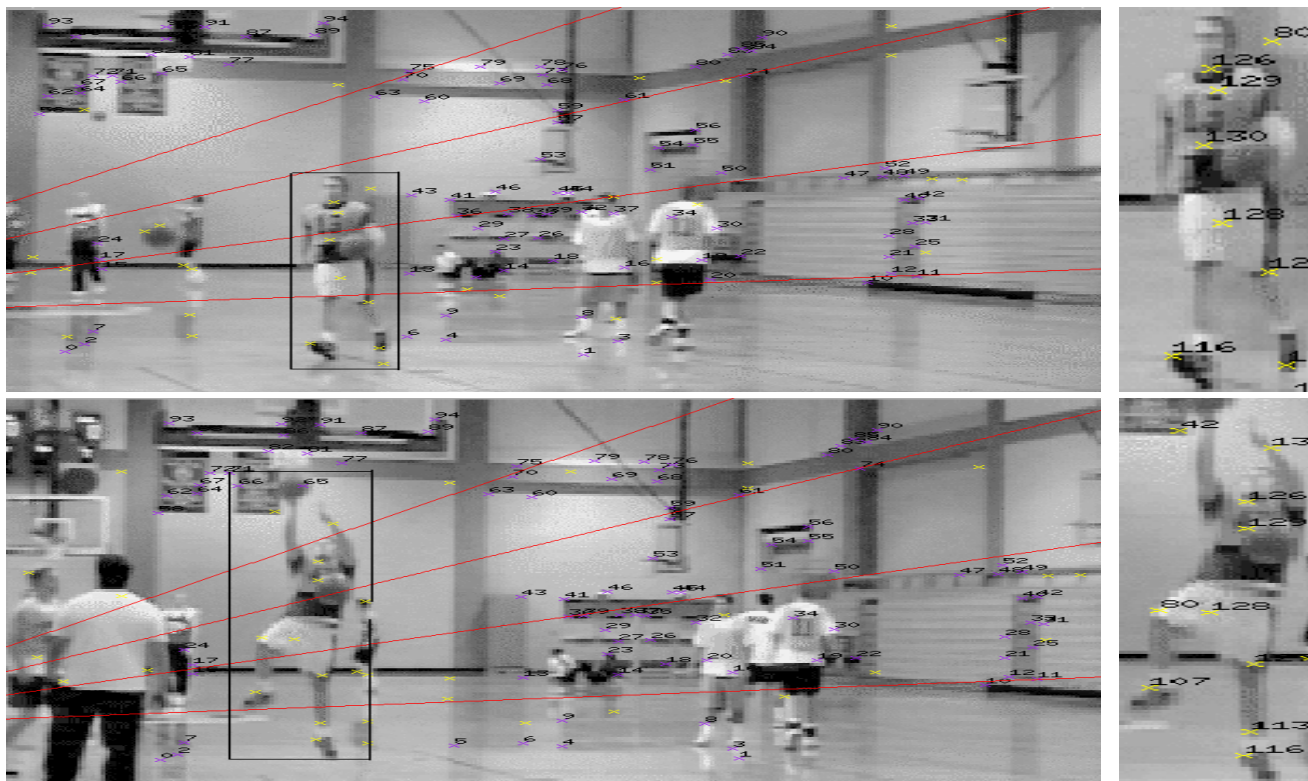


Figure 9 Game-2 pair. Incorrect matches are marked in yellow.