*Proceedings of the International Workshop on Spatial Coherence for Visual Motion Analysis (in conjunction with the European Conference on Computer Vision), Prague, Czech Republic, May 2004.*

# Analysis and Interpretation of Multiple Motions through Surface Saliency

Mircea Nicolescu

Department of Computer Science
University of Nevada, Reno
Reno, NV 89557
mircea@cs.unr.edu

Changki Min Gérard Medioni

Integrated Media Systems Center
University of Southern California
Los Angeles, CA 90089-0273
{cmin, medioni}@iris.usc.edu

**Abstract[1].** The problem of recovering the 3-D camera and scene structure has been intensively studied and is considered well understood. Starting with two images, a process of establishing point correspondences is usually followed by the estimation of epipolar geometry while also rejecting outlier matches, and finally by 3-D structure estimation. However, most existing methods tend to fail in the combined presence of noise and multiple motions, since no single constraint applies to the entire set of matches. Hence, image registration becomes a more challenging problem, as the matching and registration phases become interdependent. We propose a novel approach that decouples the above operations, allowing for separate handling of matching, outlier rejection, grouping and 3-D interpretation. Our method first determines an accurate representation in terms of dense velocities, segmented motion regions and boundaries, by enforcing only the smoothness of image motion, followed by the extraction of 3-D camera and scene geometry.

## 1 Introduction

Most existing methods for recovering the camera and 3-D scene structure from a set of correspondences are usually based on the assumption that a single constraint (e.g., rigidity, or the epipolar constraint) can be enforced on the entire set. Given two views of a static scene, a set of matching points – typically corresponding to salient image features – are first obtained by methods such as cross-correlation. Assuming that matches are perfect, a simple Eight Point Algorithm [1] can be used for estimating the fundamental matrix, and thus the epipolar geometry of the cameras is determined. A dense set of matches can be then established, and finally the scene structure is recovered through triangulation. The simplistic approach described above performs reasonably well only when (i) the set of matches contains no outlier noise, and (ii) the scene is rigid – i.e., without objects having independent motions.

The first assumption almost never holds, since image measurements are bound to be imperfect, and matching techniques will never produce accurate correspondences, mainly due to occlusion or lack of texture. In the presence of incorrect matches, linear

[1] This research has been funded in part by the Integrated Media Systems Center, an NSF Engineering Research Center, Cooperative Agreement No. EEC-9529152, and by NSF Grant 9811883.

methods are very likely to fail. The problem can be reliably solved by robust methods, which involve non-linear optimization [2][3], and normalization of data before fundamental matrix estimation [4].

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

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

In order to address these difficulties, we propose a novel approach that decouples the above operations, allowing for explicit and separate handling of matching, outlier rejection, grouping, and recovery of camera and scene structure. In the first step, we determine an accurate representation in terms of dense velocities (equivalent to point correspondences), segmented motion regions and boundaries, by using only the smoothness of image motion [5]. In the second step we proceed with the extraction of scene and camera 3-D geometry, separately on each rigid component of the scene. Note that our approach follows Ullman's interpretation of visual motion [6], in that the correspondence process takes place prior to 3-D interpretation.

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

### 1.1 Previous Work

Linear methods [1][7][8] can be used for estimation of the fundamental matrix, in the absence of noisy matches or moving objects. The Eight Point Algorithm [1] recovers the essential/fundamental matrix from two calibrated/uncalibrated images, by solving a system of linear equations. A minimum of eight points is needed – if more are available, a least mean square minimization is used. To ensure that the resulting matrix satisfies the rank two requirement, its singularity is usually enforced [4][9].

In order to handle outlier noise, more complex, non-linear iterative optimization methods are proposed [3][10][11]. These techniques use objective functions, such as distance between points and corresponding epipolar lines, or gradient-weighted epipolar errors, to guide the optimization process. Despite their increased robustness, iterative optimization methods in general require somewhat careful initialization for early convergence to the correct optimum. One of the most successful algorithms in this class is LMedS [3], which uses the least median of squares and data sub-sampling to discard outliers by solving a non-linear minimization problem.

RANSAC [12] consists of random sampling of a minimum subset with seven pairs of matching points for parameter estimation. The candidate subset that maximizes the

number of inliers and minimizes the residual is the solution. Statistical measures are used to derive the minimum number of sample subsets. Although LMedS and RANSAC are considered to be some of the most robust methods, it is worth noting that these techniques still require a majority of the data to be correct, or else some statistical assumption is needed. If false matches and independent motions exist, these methods may fail or become less attractive, since in the latter case, many matching points on the moving objects are discarded as outliers.

In [13], Pritchett and Zisserman propose the use of local planar homographies, generated by Gaussian pyramid techniques. However, the homography assumption does not generally apply to the entire image.

### 1.2 Outline of the Approach

The first step of the proposed method formulates the motion analysis problem as an inference of motion layers from a noisy and possibly sparse point set in a 4-D space. In order to compute a dense set of matches (equivalent to a velocity field) and to segment the image into motion regions, we use an approach based on a *layered 4-D representation* of data, and a *voting scheme* for communication. First we establish candidate matches through a multi-scale, normalized cross-correlation procedure. Following a perceptual grouping perspective, each potential match is seen as a token characterized by four attributes – the image coordinates $(x,y)$ in the first image, and the velocity with the components $(v_x,v_y)$.

Tokens are encapsulated as $(x,y,v_x,v_y)$ points in the 4-D space, this being a natural way of expressing the spatial separation of tokens according to *both* velocities and image coordinates. In general, for each pixel $(x,y)$ there can be several candidate velocities, so each 4-D point $(x,y,v_x,v_y)$ represents a potential match.

Within this representation, smoothness of motion is embedded in the concept of surface saliency exhibited by the data. By letting the tokens communicate their mutual affinity through voting, noisy matches are eliminated as they receive little support, and distinct moving regions are extracted as smooth, *salient surface layers* in 4-D.

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

## 2 The Voting Framework

### 2.1 Voting in 2-D

The use of a voting process for feature inference from sparse and noisy data was formalized into a unified tensor framework by Medioni, Lee and Tang [14]. The input

data is encoded as tensors, then support information (including proximity and smoothness of continuity) is propagated by voting. The only free parameter is the scale of analysis, which is indeed an inherent property of visual perception.

In the 2-D case, the salient features to be extracted are points and curves. Each token is encoded as a second order symmetric 2-D tensor, geometrically equivalent to an ellipse. It is described by a 2×2 eigensystem, where eigenvectors $e_1$ and $e_2$ give the ellipse orientation and eigenvalues $\lambda_1$ and $\lambda_2$ are the ellipse size. The tensor is represented as a matrix $S = \lambda_1 \cdot e_1 e_1^T + \lambda_2 \cdot e_2 e_2^T$.

An input token that represents a curve element is encoded as a *stick tensor*, where $e_2$ represents the curve tangent and $e_1$ the curve normal, while $\lambda_1$=1 and $\lambda_2$=0. A point element is encoded as a *ball tensor*, with no preferred orientation, and $\lambda_1$=$\lambda_2$=1.

The communication between tokens is performed through a voting process, where each token casts a vote at each site in its neighborhood. The size and shape of this neighborhood, and the vote strength and orientation are encapsulated in predefined voting fields, one for each feature type – there is a stick voting field and a ball voting field in the 2-D case. The fields are generated based only on the scale factor $\sigma$. Vote orientation corresponds to the smoothest continuation from voter to recipient, while vote strength $VS(\vec{d})$ decays with distance $|\vec{d}|$ between them, and with curvature $\rho$:

$$VS(\vec{d}) = e^{-\left(\frac{|\vec{d}|^2+\rho^2}{\sigma^2}\right)} \qquad (1)$$

Fig. 1 shows how votes are generated to build the 2-D stick field. A tensor P where curve information is locally known (illustrated by curve normal $\vec{N}_P$) casts a vote at its neighbor Q. The vote orientation is chosen so that it ensures a smooth curve continuation through a circular arc from voter P to recipient Q. To propagate the curve normal $\vec{N}$ thus obtained, the vote $V_{stick}(\vec{d})$ sent from P to Q is encoded as a tensor according to:

$$V_{stick}(\vec{d}) = VS(\vec{d}) \cdot \vec{N}\vec{N}^T \qquad (2)$$

Fig. 2 shows the 2-D stick field, with its color-coded strength. When the voter is a ball tensor, with no information known locally, the vote is generated by rotating a stick vote in the 2-D plane and integrating all contributions. The 2-D ball field is shown in Fig. 3.

At each receiving site, the collected votes are combined through simple tensor

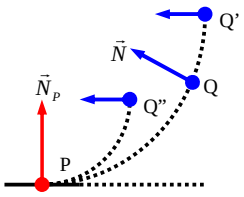



**Fig. 1.** Vote generation

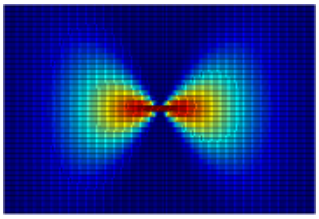


**Fig. 2.** 2-D stick field

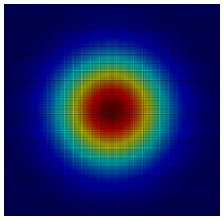


**Fig. 3.** 2-D ball field

**Table 1.** Elementary tensors in 4-D

| Feature | $\lambda_1$ $\lambda_2$ $\lambda_3$ $\lambda_4$ | $e_1$ $e_2$ $e_3$ $e_4$ | Tensor |
|---|---|---|---|
| point | 1 1 1 1 | Any orthonormal basis | Ball |
| curve | 1 1 1 0 | $n_1$ $n_2$ $n_3$ t | C-Plate |
| surface | 1 1 0 0 | $n_1$ $n_2$ $t_1$ $t_2$ | S-Plate |
| volume | 1 0 0 0 | n $t_1$ $t_2$ $t_3$ | Stick |

**Table 2.** A generic tensor in 4-D

| Feature | Saliency | Normals | Tangents |
|---|---|---|---|
| point | $\lambda_4$ | none | none |
| curve | $\lambda_3$ - $\lambda_4$ | $e_1$ $e_2$ $e_3$ | $e_4$ |
| surface | $\lambda_2$ - $\lambda_3$ | $e_1$ $e_2$ | $e_3$ $e_4$ |
| volume | $\lambda_1$ - $\lambda_2$ | $e_1$ | $e_2$ $e_3$ $e_4$ |

addition, producing generic 2-D tensors. During voting, tokens that lie on a smooth curve reinforce each other, and the tensors deform according to the prevailing orientation. Each tensor encodes the local orientation of geometric features (given by the tensor orientation), and their saliency (given by the tensor shape and size). For a generic 2-D tensor, its curve saliency is given by ($\lambda_1$-$\lambda_2$), the curve normal orientation by $e_1$, while its point saliency is given by $\lambda_2$. Therefore, the voting process infers curves and junctions simultaneously, while also identifying outlier noise.

### 2.2 Extension to 4-D

Table 1 shows all the geometric features that appear in a 4-D space and their representation as *elementary* 4-D tensors, where *n* and *t* represent normal and tangent vectors, respectively. Note that a surface in the 4-D space can be characterized by two normal vectors, or by two tangent vectors. From a *generic* 4-D tensor that results after voting, the geometric features are extracted as shown in Table 2.

The 4-D voting fields are obtained as follows. First the 4-D stick field is generated in a similar manner to the 2-D stick field (see Fig. 1). Then, the other three voting fields are built by integrating all the contributions obtained by rotating a 4-D stick field around appropriate axes. In particular, the 4-D ball field – the only one directly used here – is generated according to:

$$V_{ball}(\vec{d}) = \int_0^{2\pi}\int\int R\, V_{stick}(R^{-1}\vec{d})\, R^T \, d\theta_{xy} d\theta_{xu} d\theta_{xv} \qquad (3)$$

where $x$, $y$, $u$, $v$ are the 4-D coordinates axes and $R$ is the rotation matrix with angles $\theta_{xy}$, $\theta_{xu}$, $\theta_{xv}$.

The data structure used to store the tensors is an *approximate nearest neighbor (ANN) k-d tree* [15]. The space complexity is $O(n)$, where $n$ is the input size (the total number of candidate tokens). The average time complexity of the voting process is $O(\mu n)$ where $\mu$ is the average number of candidate tokens in the neighborhood. Therefore, in contrast to other voting techniques, such as the Hough Transform, both time and space complexities of the Tensor Voting methodology are *independent* of the dimensionality of the desired feature.

## 3 Motion Segmentation

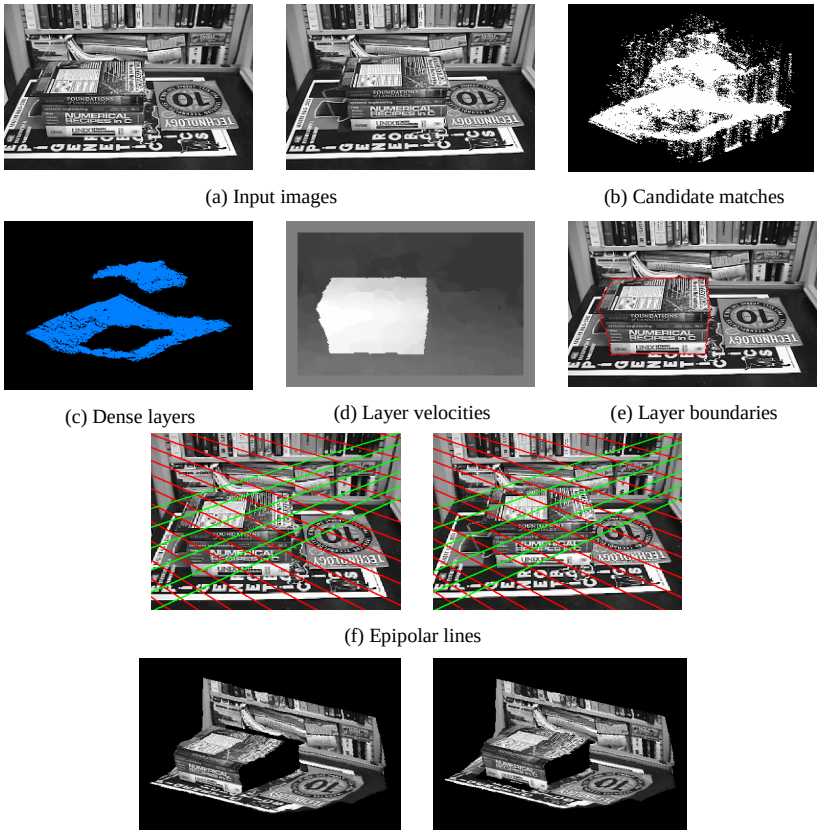
We take as input two image frames that involve general motion – that is, both the camera and the objects in the scene may be moving. For illustration purposes, we give a description of our approach by using a specific example – the two images in Fig. 4(a) are taken with a handheld moving camera, while the stack of books has been moved between taking the two pictures.

**Matching**. For every pixel in the first image, the goal at this stage is to produce candidate matches in the second image. We use a normalized cross-correlation procedure [16], where all peaks of correlation are retained as candidates. When a peak is found, its position is also adjusted for sub-pixel precision according to the correlation values of its neighbors. Finally, each candidate match is represented as a $(x,y,v_x,v_y)$ point in the 4-D space of image coordinates and pixel velocities, with respect to the first image.

Since we want to increase the likelihood of including the correct match among the candidates, we repeat this process at multiple scales, by using different correlation window sizes. Small windows have the advantage of capturing fine detail, and are effective close to the motion boundaries, but produce considerable noise in areas lacking texture or having small repetitive patterns. Larger windows generate smoother matches, but their performance degrades in large areas along motion boundaries. We have experimented with a large range of window sizes, and found that best results are obtained by using only two or three different sizes, that should include at least a very small one. In practice we used three correlation windows, with 3x3, 5x5 and 7x7 sizes.

The resulting candidates appear as a cloud of $(x,y,v_x,v_y)$ points in the 4-D space. Fig. 4(b) shows the candidate matches. In order to display 4-D data, the last component of each 4-D point has been dropped – the 3 dimensions shown are $x$ and $y$ (in the horizontal plane), and $v_x$ (the height). The motion layers can be already perceived as their tokens are grouped in two layers surrounded by noisy matches.

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



(a) Input images (b) Candidate matches

(c) Dense layers (d) Layer velocities (e) Layer boundaries

(f) Epipolar lines

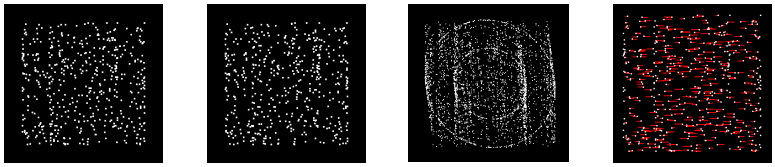
(g) 3-D structure and motion

**Fig. 4.** BOOKS sequence

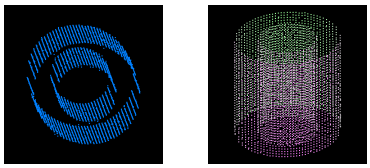
**Selection**. Since no information is initially known, each potential match is encoded into a 4-D *ball tensor*. Then each token casts votes by using the 4-D *ball voting field*. During voting there is strong support between tokens that lie on a smooth surface (layer), while communication between layers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

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

**Densification**. Since the previous step created holes (i.e., pixels where no velocity is available), we must infer them from the neighbors by using a smoothness constraint.



(a) Input images (b) Candidate matches (c) Velocities



(d) Dense layers (e) 3-D structure

**Fig. 5.** CYLINDERS sequence

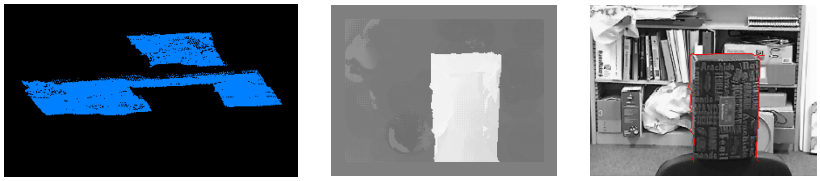
For each pixel $(x,y)$ without an assigned velocity we try to find the best $(v_x,v_y)$ location at which to place a newly generated token. The candidates considered are all the discrete points $(v_x,v_y)$ between the minimum and maximum velocities in the set, within a neighborhood of the $(x,y)$ point. At each candidate position $(x,y,v_x,v_y)$ we accumulate votes, according to the same Tensor Voting framework that we have used so far. After voting, the candidate token with maximal surface saliency ($\lambda_2$-$\lambda_3$) is retained, and its $(v_x,v_y)$ coordinate represent the most likely velocity at $(x,y)$. By following this procedure at every $(x,y)$ image location we generate a *dense velocity field*. Note that in this process, along with velocities we simultaneously infer layer orientations. A 3-D view of the dense layers is shown in Fig. 4(c).

**Segmentation**. The next step is to group tokens into *regions*, by using again the smoothness constraint. We start from an arbitrary point in the image, assign a region label to it, and try to recursively propagate this label to all its image neighbors. In order to decide whether the label must be propagated, we use the smoothness of both velocity and layer orientation as a grouping criterion. Fig. 4(d) illustrates the recovered $v_x$ velocities within layers (dark corresponds to low velocity).

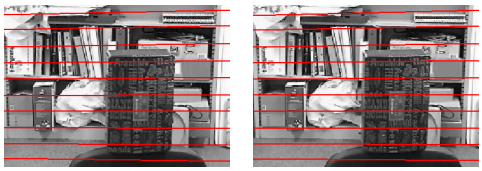
**Boundary inference**. The extracted layers may still be over or under-extended along the true object boundaries. This situation typically occurs in areas subject to occlusion, where the initial correlation procedure may generate wrong matches that are consistent with the correct ones, and therefore could not be rejected as outlier noise. The boundaries of the extracted layers give us a good estimate for the position and overall orientation of the true boundaries. We combine this knowledge with monocular cues (intensity edges) from the original images in order to build a boundary saliency map within the uncertainty zone along the layers margins. At each location in this area, a 2-D stick tensor is generated, having an orientation normal to the image gradient, and a saliency proportional to the gradient magnitude.

The smoothness and continuity of the boundary is then enforced through a 2-D voting process, and the true boundary is extracted as the most salient curve within the saliency map. Finally, pixels from the uncertainty zone are reassigned to regions according to the new boundaries, and their velocities are recomputed. Fig. 4(e) shows the refined motion boundaries, that indeed correspond to the actual object.

(a) Input images (b) Candidate matches



(c) Dense layers (d) Layer velocities (e) Layer boundaries



(f) Epipolar lines

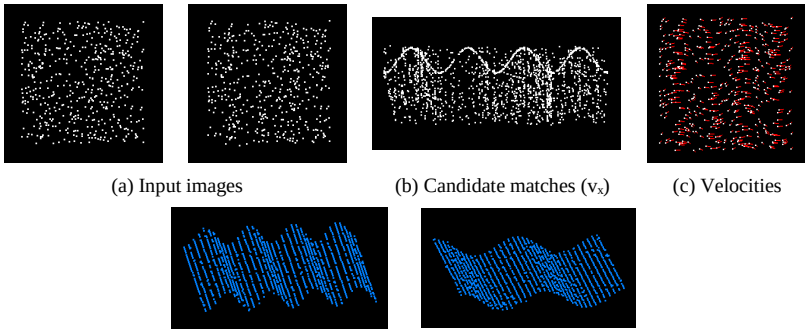
(g) 3-D structure and motion

**Fig. 6.** CANDY BOX sequence

## 4 Interpretation of Image Motion

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

For classification we used an algorithm introduced by McReynolds and Lowe [17], that verifies the potential rigidity of a set of minimum six point correspondences from two views under perspective projection. The rigidity test is performed on a subset of matches within each object, to identify potential non-rigid objects, and also across objects, to merge those that move rigidly together but have distinct image motions due to depth discontinuities. It is also worth mentioning that the rigidity test is actually



(a) Input images (b) Candidate matches ($v_x$) (c) Velocities

(d) Dense layers ($v_x$) (e) Dense layers ($v_y$)

**Fig. 7.** FLAG sequence

able to only guarantee the *non-rigidity* of a given configuration. Indeed, if the rigidity test fails, it means that the image motion is not compatible to a rigid 3-D motion, and therefore the configuration *must* be non-rigid. If the test succeeds, it only asserts that a possible rigid 3-D motion *exists*, that is compatible to the given image motion. However, this computational approach corresponds to the way human vision operates – as shown in [6], human perception solves this inherent ambiguity by always choosing a rigid interpretation when possible.

The remaining task at this stage is to determine the object (or camera) motion, and the scene structure. Since wrong matches have been eliminated, and correct matches are already grouped according to the rigidly moving objects in the scene, standard methods for reconstruction can be reliably applied. For increased robustness, we chose to use RANSAC [12] to recover the epipolar geometry for each rigid object, followed by an estimation of camera motion and projective scene structure.

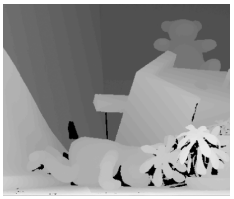
The following discussion describes each case, illustrated with experimental results.

**Multiple rigid motions**. This case is illustrated by the BOOKS example in Fig. 4, where two sets of matches have been detected, corresponding to the two distinct objects – the stack of books and the background. The rigidity test shows that, while each object moves rigidly, they cannot be merged into a single rigid structure. The recovered epipolar geometry is illustrated in Fig. 4(f), while the 3-D scene structure and motion are shown in Fig. 4(g).

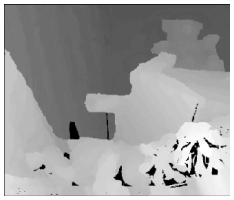
The CYLINDERS example, shown in Fig. 5, is adapted from Ullman [6], and consists of two images of random points in a sparse configuration, taken from the surfaces of two transparent co-axial cylinders, rotating in opposite directions. This extremely difficult example clearly illustrates the power of our approach, which is able to determine accurate point correspondences and scene structure – even from a sparse input, based on motion cues only (without any monocular cues), and for transparent motion.

**Single rigid motion**. This is the stereo case, illustrated by the CANDY BOX example in Fig. 6, where the scene is static and the camera is moving. Due to the depth disparity between the box and the background, their image motions do not satisfy the smoothness constraint together, and thus they have been segmented as two separate objects. However, the rigidity test shows that the two objects form a rigid

(a) An input image



(b) Ground truth disparity map



(c) Tensor Voting disparity map

**Fig. 8.** TEDDY sequence

**Table 3.** TEDDY sequence – results [18]

| Methods | Error Rate |
|---|---|
| Tensor Voting | **15.4%** |
| Sum of Squared Differences | 26.5% |
| Dynamic Programming | 30.1% |
| Graph Cuts | 29.3% |

configuration, and therefore are labeled as a single object. The epipolar geometry estimation and scene reconstruction are then performed on the entire set of matches. Along with the 3-D structure, Fig. 6(g) also shows the two recovered camera positions.

**Non-rigid motion**. The FLAG example, shown in Fig. 7, is a synthetic sequence where sparse random dots from the surface of a waving flag are displayed in two frames. The configuration is recognized as non-rigid, and therefore no reconstruction is attempted. However, since the *image motion* is smooth, our framework is still able to determine correct correspondences, extract motion layers, segment non-rigid objects, and label them as such.

We also analyzed a standard sequence (the TEDDY example – Fig. 8) with ground truth available, to provide a quantitative estimate for the performance of our approach, as compared to other methods. As shown in Table 3 (partially reproduced from [18]), our voting-based approach has the smallest error rate (percentage of pixels with a disparity error greater than 1), among the techniques mentioned.

## 5 Conclusions

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

Our methodology for extracting motion layers is based on a *layered 4-D representation* of data, and a *voting scheme* for token communication. It allows for structure inference without using any prior knowledge of the motion model, based on the smoothness of motion only, while consistently handling both smooth moving regions and motion discontinuities. The method is also computationally robust, being non-iterative, and does not depend on critical thresholds, the only free parameter being the scale of analysis.

We plan to extend our approach by incorporating information from multiple frames, and to study the possibility of using an adaptive scale of analysis in the voting process.

## References


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

[2] A. Adam, E. Rivlin, L. Shimshoni, "Ror: Rejection of Outliers by Rotations", Trans. PAMI, 23(1), pp. 78-84, 2001.

[3] Z. Zhang, "Determining the Epipolar Geometry and Its Uncertainty: A Review", IJCV, 27(2), pp. 161-195, 1998.

[4] R. I. Hartley, "In Defense of the 8-Point Algorithm", PAMI, 19(6), pp. 580-593, 1997.

[5] M. Nicolescu, G. Medioni, "Layered 4D Representation and Voting for Grouping from Motion", Trans. PAMI – Special Issue on Perceptual Organization in Computer Vision, vol. 25, no. 4, pp. 492-501, 2003.

[6] S. Ullman, "The Interpretation of Visual Motion", MIT Press, 1979.

[7] T. Huang and A. Netravali, "Motion and Structure from Feature Correspondences: A Review", P-IEEE, vol. 82, pp. 252-268, 1994.

[8] R. Hartley, "Projective Reconstruction and Invariants from Multiple Images", Trans. PAMI, vol. 16, no. 10, pp. 1036-1040, 1994.

[9] O. Faugeras, "Stratification of 3-D Vision: Projective, Affine, and Metric Representations", J. Optical Society of America, 12(3), pp. 465-484, 1995.

[10] Q.-T. Luong, O. Faugeras, "The Fundamental Matrix: Theory, Algorithms, and Stability Analysis", IJCV, vol. 17, pp. 43-76, 1996.

[11] R. Mohr, F. Veillon, L. Quan, "Relative 3D Reconstruction Using Multiple Uncalibrated Images", CVPR, pp. 543-548, 1993.

[12] P.H.S. Torr, D.W. Murray, "A Review of Robust Methods to Estimate the Fundamental Matrix", IJCV, 1997.

[13] P. Pritchett, A. Zisserman, "Wide Baseline Stereo Matching", ICCV, pp. 754-760, 1998.

[14] G. Medioni, Mi-Suen Lee, Chi-Keung Tang, "A Computational Framework for Segmentation and Grouping", Elsevier Science, 2000.

[15] S. Arya, D. Mount, N. Netanyahu, R Silverman, A. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions", Journal of the ACM, 45:6, pp. 891-923, 1998.

[16] P. Anandan, "A Computational Framework and an Algorithm for the Measurement of Visual Motion", IJCV, vol. 2, pp. 283-310, 1989.

[17] D. McReynolds, D. Lowe, "Rigidity Checking of 3D Point Correspondences Under Perspective Projection", Trans. PAMI, 18(12), pp. 1174-1185, 1996.

[18] D. Scharstein, R. Szeliski, "High-Accuracy Stereo Depth Maps Using Structured Light", CVPR, pp. 195-202, 2003.