Trajectory Triangulation: 3D Reconstruction of Moving Points from a Monocular Image Sequence

Shai Avidan, *Member*, *IEEE*, and Amnon Shashua, *Member*, *IEEE*

Abstract—We consider the problem of reconstructing the 3D coordinates of a moving point seen from a monocular moving camera, i.e., to reconstruct moving objects from line-of-sight measurements only. The task is feasible only when some constraints are placed on the shape of the trajectory of the moving point. We coin the family of such tasks as "trajectory triangulation." We investigate the solutions for points moving along a straight-line and along conic-section trajectories. We show that if the point is moving along a straight line, then the parameters of the line (and, hence, the 3D position of the point at each time instant) can be uniquely recovered, and by linear methods, from at least five views. For the case of conic-shaped trajectory, we show that generally nine views are sufficient for a unique reconstruction of the moving point and fewer views when the conic is of a known type (like a circle in 3D Euclidean space for which seven views are sufficient). The paradigm of trajectory triangulation, in general, pushes the envelope of processing dynamic scenes forward. Thus static scenes become a particular case of a more general task of reconstructing scenes rich with moving objects (where an object could be a single point).

Index Terms—Structure from motion, multiple-view geometry, dynamic scenes.

1 Introduction

We wish to remove the static scene assumption in 3D-from-2D reconstruction by introducing a new paradigm we call *trajectory triangulation* that pushes the envelope of processing dynamic scenes from segmentation to 3D reconstruction.

Consider the situation in which a 3D scene containing a mix of static and moving objects is viewed from a moving monocular camera. The typical question addressed in this context is that of "segmentation": Can one separate the static from dynamic in order to calculate the camera ego-motion (and 3D structure of the static portion)? This question is basically a robust estimation issue and has been extensively (and successfully) treated as such in the literature (cf. [9], [6]).

However, consider the next (natural) question in this context: Can one reconstruct the 3D coordinates of a (single) point on a moving object? The measurements available in this context are *line-of-sight* only, thus, in a general situation the task of reconstruction *is not feasible*, unless further constraints are imposed. In order to reconstruct the coordinates of a 3D point, the point must be static in at least two views (to enable triangulation)—if the point is moving generally then the task of triangulation is not feasible. Note that the feasibility issue arises regardless of whether we assume the ego-motion of the camera to be known or not. Knowledge of camera ego-motion does not change the feasibility of the problem.

- *S. Avidan is with the Vision Technology group, Microsoft Research, Microsoft, Redmond, WA 98052. E-mail: avidan@microsoft.com.*
- *A. Shashua is with the Insititute of Computer Science, Hebrew University of Jerusalem, Israel. E-mail: shashua@cs.huji.ac.il.*

Manuscript received 15 Apr. 1999; accepted 18 Jan. 2000.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 109610.

We propose a stratified approach starting from a straight-line trajectory (the simplest, yet very practical) up to a general planar conic trajectory. The problem definition is as follows (see Fig. 1):

Problem Definition 1 (Trajectory Triangulation). *Consider a 3D point moving along some unknown trajectory, where the trajectory satisfies a parametric form (a straight-line or a conic section). The moving point is seen by a moving camera (2D projection) whose motion is general but known. The problem is to reconstruct the 3D positions of the point, at each time instant, from the known 2D matches across the views.*

Note that, we have not placed constraints on the laws of motion along the trajectory. The point can move arbitrarily along the trajectory, thus, the only assumption/constraint we are making is that the trajectory is either a 3D line or conic section. The straight-line assumption is reasonable for a range of applications as people, cars, and airplanes tend to move largely along straight-lines and is also valid in general situations for relatively small-time intervals (as an approximation). The conic-section parametric form is the next level up in the stratification hierarchy and, which, provides a wider range of applications (albeit at the price of careful considerations on numerical stability). Note that in the problem definition we assumed knowledge of camera ego-motion (projection matrices). We acknowledge the difficulty of recovering the camera ego-motion in general and under dynamic scene conditions, in particular, but believe it to be reasonable in view of the large body of theoretical and applied literature on the subject. Thus, we treat the problem of ego-motion as a "black-box" and a first layer in a hierarchy of tasks that are possible in a "3D-from-2D" family of problems.

The case of conic-based trajectories is reminiscent to the classic problem of *orbit determination* in astrodynamics (cf. [3]). Orbit determination is about calculating the orbit

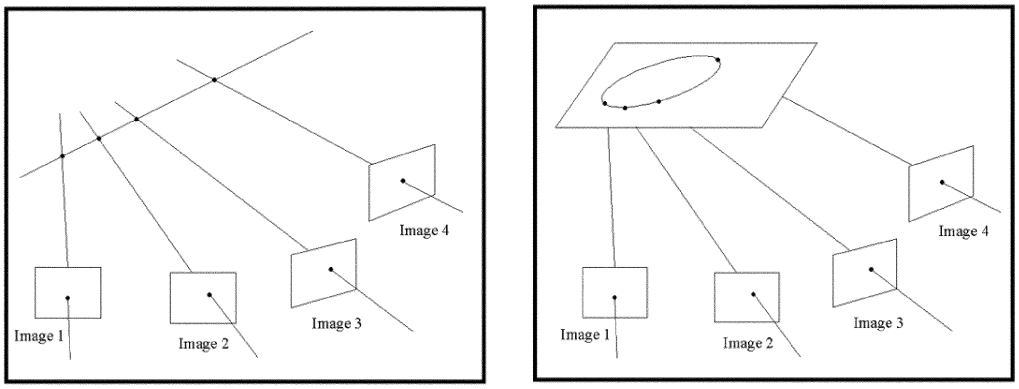


Fig. 1. (a) "Trajectory Triangulation" along a line. A line is moving along a line while the camera is moving. (b) "Trajectory Triangulation" along a planar conic. A point is moving along a planar conic while the camera is moving.

(conic section, typically elliptic) of body A around body B under a gravitational field. A branch of this problem includes the determination of an orbit from directional measurements only (lines of sight). However, the assumption of motion under a gravitational field constrains not only the shape of the trajectory (conic section), but also the law of motion *along* the trajectory—in this case, the motion is Keplerian, which is to say that equal areas are swept during equal times.

Our work on determining a conic trajectory from line of sight measurements (trajectory triangulation over conics) differs from the classic work on orbit determination by that *the motion of the point along the trajectory is arbitrary*. In other words, the only assumption we make is about the *shape* of the trajectory (conic section) while the motion of the point along the trajectory is unconstrained.

The paper proceeds as follows: We start with the case of straight-line trajectories (linear trajectory) and show that the solution boils down to determining a unique linear-line-complex from a set of five or more views. The algebraic solution involves linear equations over the Plucker coordinates of the trajectory—is remarkably simple and practical. We then proceed to the case of conic trajectories and we introduce two approaches for solving for the conic parameters (the position of the plane on which the conic resides and the location, shape, and type of the conic on that plane) from line-of-sight measurements. Both methods require nonlinear estimation techniques, but, are shown to be practical through experimentation on real image sequences. We end the paper with suggestion on future research on the general topic of *trajectory triangulation*. Parts of this work, as it evolved, have appeared in the meetings found in [1], [10].

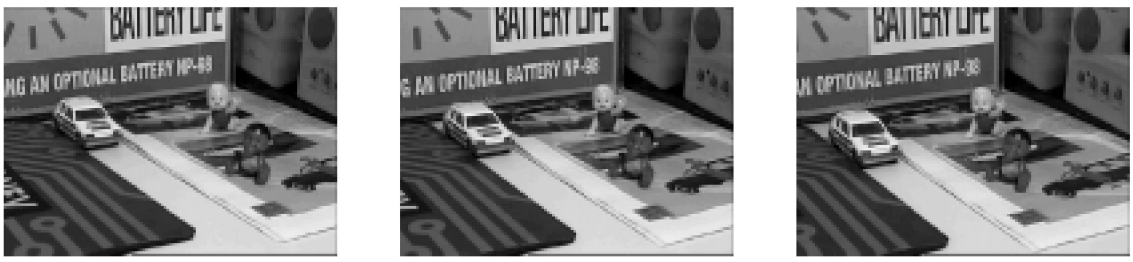
2 Trajectory Triangulation Over Straight Lines

Fig. 2 and Fig. 3 provide a sketch of the problem. A target point is moving along some unknown straight-line in 3D and is viewed by a number of cameras (or a moving single camera) with known projection matrices. In other words, the line-of-sights as rays in 3D are known (the position of the target point along the line of sights is unknown). The problem is to find the parameters of the 3D line from the observations, and once that is known, it becomes a simple problem (intersecting rays) to reconstruct the position of the target at each time instant.

The first question to consider is *how many views are necessary for a unique solution for the parameters of the line*? One can easily show that four views in general provide two solutions, thus, five views provide a unique solution. We first address this issue geometrically and then algebraically, as follows:

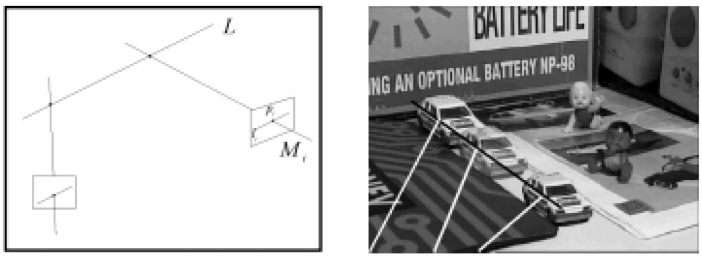
2.1 Geometric Interpretation

The geometric picture behind the problem of trajectory triangulation over straight lines is a *linear line complex*, i.e., a set of 3D lines that have a common intersecting line L. The intersecting line is the straight-line trajectory of the moving point (unknown) and the set of lines are the optical rays



(a) (b) (c)

Fig. 2. Three frames from a sequence in which the car is moving independently of the scene. The camera is moving to the right.



(a) (b)

Fig. 3. Illustration of Trajectory Triangulation. (a) Schematic illustration. A point moving along a 3D line and is projected on the image plane of a moving camera. Since the 3D point is moving, one can not use triangulation to recover its coordinates. A sketch of this principle is seen in (b) where the car is moving along a straight line while the camera is moving. The lines drawn from the car are the optical rays of a single point on the car as seen by the moving camera.

from the camera at each time instant (known). Let there be k views of the moving point. The question, therefore, is what is the minimal k that form a *unique* linear line complex?

Clearly, if $k = 2$, we have infinite lines L intersecting the two rays. For $k = 3$, for each point along the first ray there is a unique line incident to it and to the other two rays (the point and the second ray define a plane which intersects the third ray uniquely)—hence, we still have infinite lines L (see Fig. 4). For $k = 4$, three of the rays define a ruled quadric (the collections of lines swept by the point moving along the first ray, as considered above), which intersects the fourth ray at two distinct points—thus, we have two solutions for L, and thus, for $k = 5$ we have a unique solution.

The argument that one needs at least four lines for a finite number of solutions for a common intersecting line is well-known and is also used in graphics algorithms for synthetic illumination and visibility computations (cf. [11]).

2.2 Solving for Plucker Representation

We wish to recover the piercing line L given $k \geq 5$ views, known 3×4 projection matrices (describing camera motion) M_i and the projections p_i, $i = 1, \ldots, k$ of the moving point along the line L.

Let P, Q be any two points on the line L, and let l_i be the projection of L on view i. Clearly, $p_i^\top l_i = 0$ because p_i is incident to the line l_i in the image plane. We can represent l_i by the cross product of the projections of P and Q:

$$l_i \cong (M_i P) \times (M_i Q),$$

because M_i projects 3D points onto view i. A convenient way to simplify this expression is to represent the line L using Plucker coordinates: $L = P \wedge Q$ which is a vector of six components defined as follows:

$$\begin{aligned} L &= P \wedge Q \\ &= [1, X_p, Y_p, Z_p] \wedge [1, X_q, Y_q, Z_q] \\ &= [X_p - X_q, Y_p - Y_q, \\ &\quad Z_p - Z_q, X_p Y_q - Y_p X_q, \\ &\quad X_p Z_q - Z_p X_q, Y_p Z_q - Z_p Y_q]. \end{aligned} \tag{1}$$

The Plucker representation of the line is defined up to scale and is independent of the choice of P, Q. The entries of the vector L are the 2×2 minors of

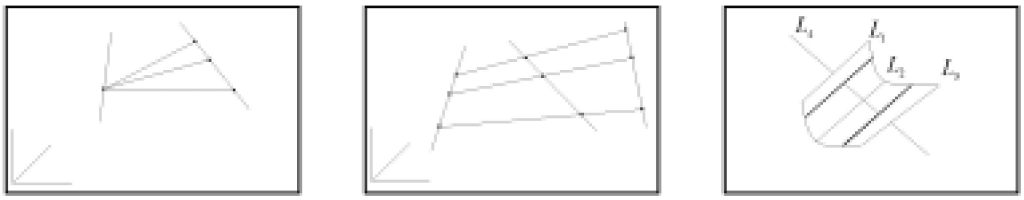
$$\begin{bmatrix} 1 & X_p & Y_p & Z_p \\ 1 & X_q & Y_q & Z_q \end{bmatrix}. \tag{2}$$

The entries of L also satisfy a quadratic constraint

$$L_1 L_6 - L_2 L_5 + L_3 L_4 = 0,$$

for all Plucker coordinates. The space of all 3D lines is therefore embedded in a five-dimensional projective space and subject to a quadratic constraint (i.e., not every six tuple corresponds to a real line). Thus, the six Plucker coordinates describe a four parameter space which confirms basic intuition that a 3D line could be parameterized by four numbers (such as by slope and intercept on two standard planes).

In [4], it was pointed out that by using the Plucker representation one can readily transform M_i into a 3×6 matrix $\tilde{M}_i$ that satisfies a line projection matrix relation: $l_i \cong \tilde{M}_i L$, where L is represented by its six Plucker coordinates. The rows of $\tilde{M}_i$ are defined by the $\wedge$ ("meet")



(a) (b) (c)

Fig. 4. Figures (a) and (b) show that one can define infinite number of lines that intersect two or three given lines, respectively. Given three lines (b), there is a unique intersecting line for every point along the first line, thus the set of intersecting lines form a ruled surface (quadric); the fourth line (c) intersects the surface at two points, thus there are two intersecting lines for a general set of four lines.

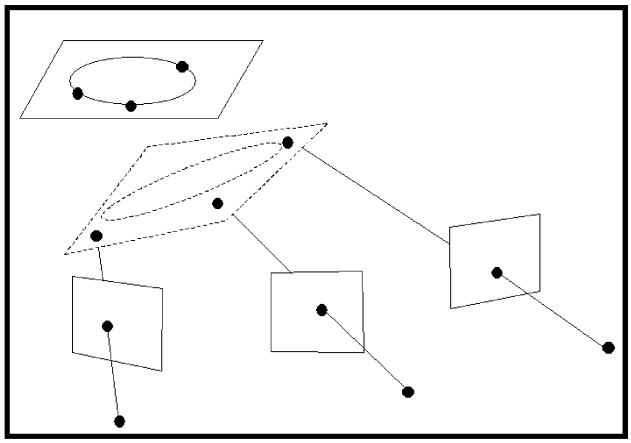


Fig. 5. Projecting the 3D line defined by the moving car on one of the images of the sequence.

operation (1) on the pairs of rows of M_i. (M^j stands for the jth row of camera matrix M)

$$\tilde{M} = \begin{bmatrix} M^2 \wedge M^3 \\ M^3 \wedge M^1 \\ M^1 \wedge M^2 \end{bmatrix}. \quad (4)$$

We have, therefore, established the following linear constraint on the unknown Plucker vector L:

$$p_i^\top \tilde{M}_i L = 0$$

which is linear in the parameters of L. Thus, five views in general provide a unique solution and more than five views provide a least-squares solution. Generally, the rank of the estimation matrix is five and L is the null space of the estimation matrix.

The quadratic constraint comes into play when the rank of the estimation matrix is four. This situation arises when the number of views is four (as we have seen in the previous section, we expect to have two solutions) or when the camera center of projection traces a straight line during camera motion. In these cases, we have a two-dimensional null space spanned by v_1, v_2, thus, $L \cong \lambda_1 v_1 + \lambda_2 v_2$. The scalars λ_1, λ_2 can be found up to a common scale from the quadratic constraint (3), thus, we obtain a second-order constraint on λ_1, λ_2 which provides two solutions for L.

The degenerate situations occur when the moving point and the camera center trace trajectories that live in the same ruled quadric surface. For example, when the camera center traces a straight line coplanar with L, we have a rank deficient situation (any line on that plane is a solution).

2.3 Multiple Points

So far, we have discussed the case of a single moving point along a straight line viewed by a projective camera. On an object moving along a straight-line path one may possibly track a number of points—those would correspond to a family of *parallel* lines. Note from (1), that, the coefficients L_1, L_2, L_3 denote the direction of the line L, thus, a set of k parallel lines are determined by $3 + 3k$ Plucker coordinates (up to scale). For example, when two points are tracked across four views, we have eight equations for a unique solution for the two parallel lines. More views and/or more points would give rise to an over-determined system of equations. In practice, due to the proximity of the points compared to the field of view of the camera, the added equations would make a relatively small contribution to the numerical stability of the system—but in any case, if the information exists (of multiple points on the straightly moving object), it is worthwhile making use of it.

2.3.1 Reconstructing the 3D Point

Note that once the 3D line L is recovered, it is a simple matter to reconstruct the actual 3D points as they were

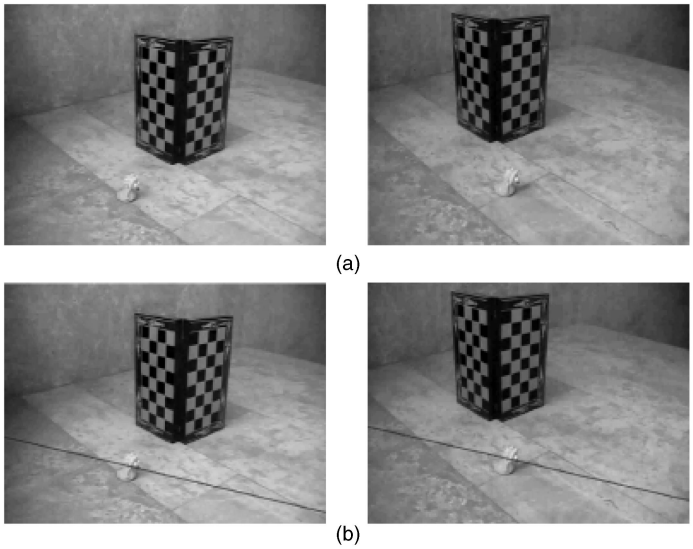


Fig. 6. The top row (a-1,a-2), shows the two extreme frames from a 30-frames sequence. We have computed the camera matrices of all the 30 frames using the chess-board. The bottom row (b-1,b-2), shows the recovered 3D line L as projected on the extreme two frames. The tip of the nose of the moving doll was used for the tracking. The last 10 frames were not used in computing the 3D line, yet, the distance between the tracked point and the projected 3D line L is about one pixel for all frames in the sequence.

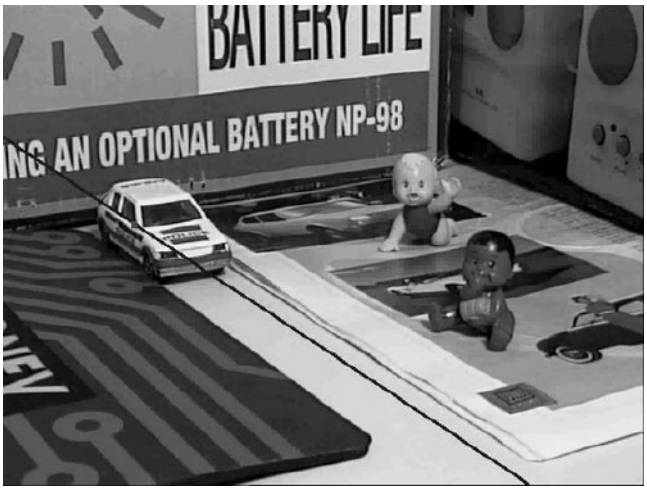


Fig. 7. Sketch of Method I. The true plane is shown in bold, the guessed plane is shown with dashed lines. Choosing a different plane affects the projection of the points on the first image (or common plane in general). If the plane is not the correct one, then the points on the first image will not form a conic.

moving in space, simply by intersecting each ray with the line L. This is done as follows. Represent the line L as a linear combination of two points

$$Q_1 = [1, X, Y, Z], Q_2 = [1, X + \delta X, Y + \delta Y, Z + \delta Z],$$

where $(\delta X, \delta Y, \delta Z) = (L_1, L_2, L_3)$, the first three components of the line L, and X, Y, Z can be solved from the following set of linear equations:

$$\begin{bmatrix} L_2 & -L_1 & 0 \\ L_3 & 0 & -L_1 \\ 0 & L_3 & -L_2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} L_4 \\ L_5 \\ L_6 \end{bmatrix}. \quad (5)$$

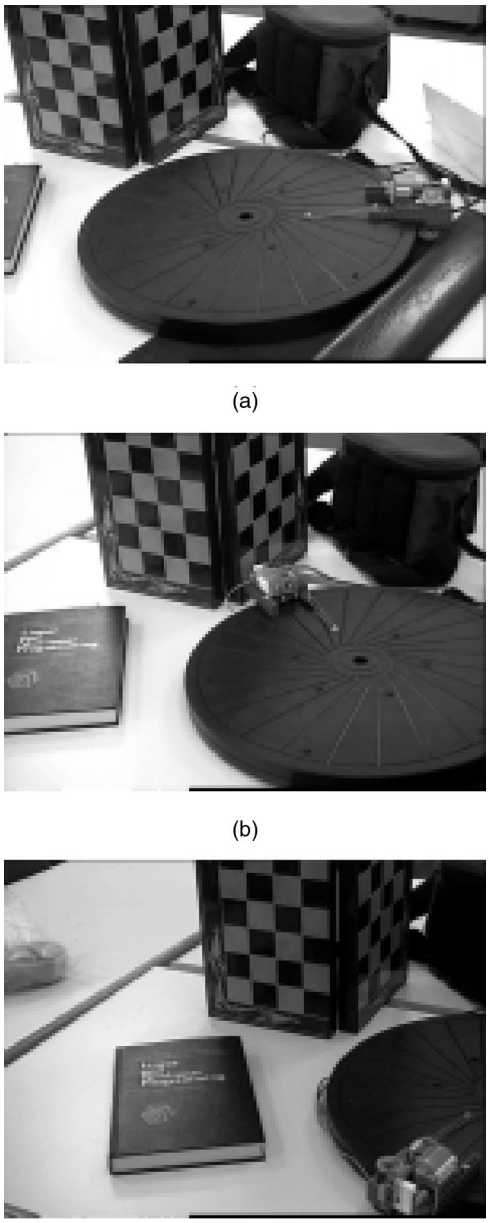
Next, find λ such that the point $P_i = Q_1 + \lambda Q_2$ satisfies the equation $p_i \cong M_i P_i$. P_i is the 3D position of the object at time instant i.

3 Trajectory Triangulation Over Conics

The next level up in the stratification hierarchy, yet still maintaining a simple solution, is when the target point in moving along some unknown conic section in 3D. In this case, we wish to recover—from measurements of line-of-sight only—eight parameters in general: Three for the position of the plane on which the conic resides on, and five for the conic itself. Once these parameters are recovered, the 3D coordinates of the moving point can be recovered by intersecting the line-of-sight with the conic section.

We propose two methods for recovering the parameters. The first method performs a 2D optimization (based on conic fitting) on some arbitrary virtual common plane. The method is very simple, but can only deal with general conics—a priori constraints on the shape of the 3D conic cannot be enforced due to the projective distortion from the conic plane to the virtual common plane.

The second method is slightly more complex as the optimization is performed in 3D (projective or Euclidean) but enables the enforcement of a priori constraints on the shape of the conic when the cameras are calibrated.



(a)

(b)

(c)

Fig. 8. The original image sequence. (a), (b) and (c) are the first, middle and last images, respectively in a sequence of 16 images. The camera is moving mainly to the left, while the Lego cube traces a circle on the turntable.

Numerical stability is greatly enhanced when a priori information is integrated into the estimation process. We will derive the second method for the case of calibrated cameras and when the conic in 3D is a circle. The extension to general conics follows in a straightforward manner, but will not be derived here.

3.1 Method I: 2D Optimization on a Common Plane

We denote the 3D position of the moving point and the camera matrix (projection matrix) at time $i = 1 \ldots k$ by $P_i = [X_i, Y_i, Z_i, 1]^T$ and $M_i = [H_i \; ; \; t_i]$, respectively. The image measurements are, thus, $p_i \cong M_i P_i$. Our goal is to recover the 3D points P_i, given the uncalibrated camera matrices M_i and the image measurements p_i. This can be

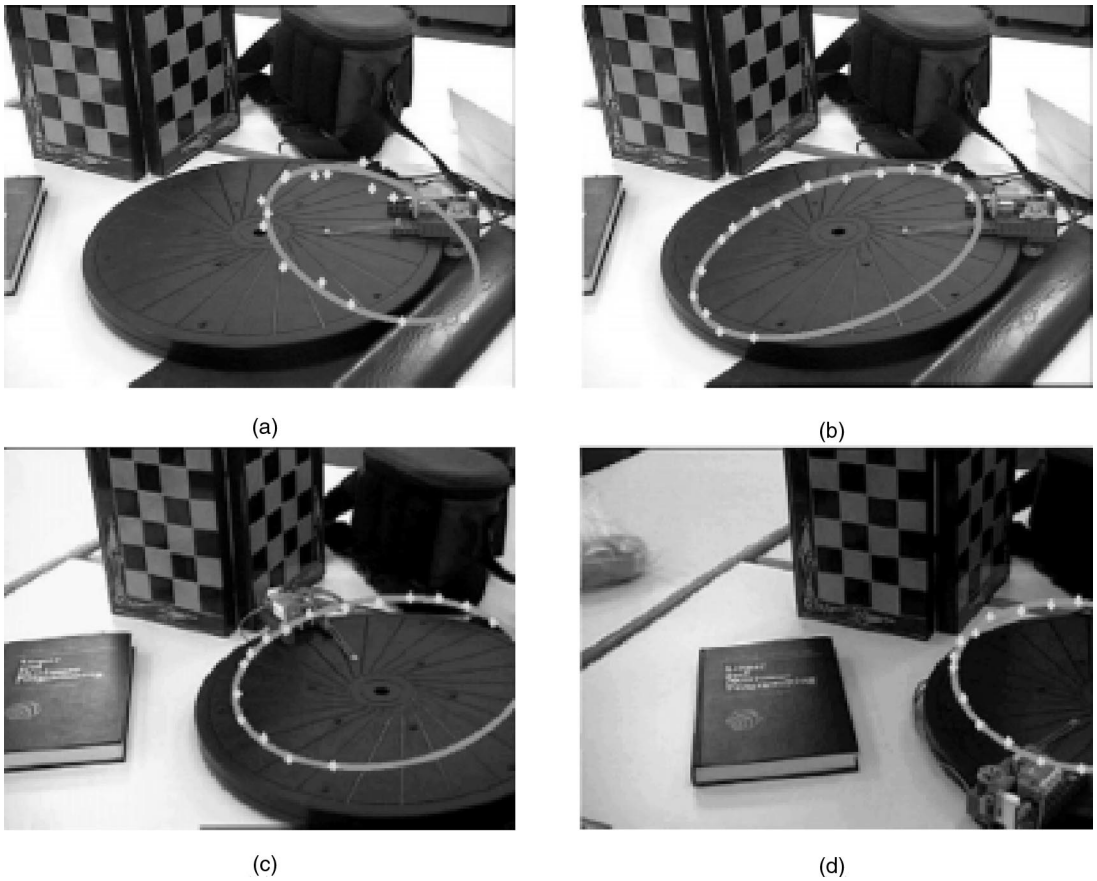


Fig. 9. Using 2D conic fitting (Method I) to recover the planar conic section. The results are shown by projecting the recovered planar conic (and the 3D points traced along the conic) on several refernce images from the sequence. (a) Shows the initial guess with the first image as the reference image, (b), (c), and (d) shows the initial results of the 2D conic fitting when the reference image is the first, middle, and the last images of the sequence, respectively. The resulting conic had an aspect ratio of 0.9, radius roughly 8 percent off, and orientation of the plane was 6^o off.

formulated as a nonlinear optimization problem in which eight parameters are to be estimated. The three parameters of the normal to the plane n and the five parameters of the conic as defined (up to scale) by a symmetric 3×3 matrix C.

Let the sought-after plane on which the conic resides on be denoted by π. Let A_i be the 2D homography from image i to some common arbitrary plane (image plane $i = 1$ if $M_1 = [I; 0]$) through the plane π, i.e., $A_i p_i$, $i = 1, \ldots, k$ must be a conic on the common plane (see Fig. 7). The following relation must hold:

$$A_i = (\| \mathbf{n} \| H_i + t_i \mathbf{n}^\top)^{-1}.$$

Therefore, each view provides one (nonlinear) constraint:

$$p_i^\top A_i^\top C A_i p_i = 0, \qquad i = 1, \ldots, k.$$

Since the total number of parameters are eight and each view contributes one (nonlinear) equation, then eight views are necessary for a solution (up to a finite-fold ambiguity) and nine views for a unique solution. It is possible to solve for n and C by means of numerical optimization, or to use an interleaving approach described below:

1. Start with an initial estimate of $\mathbf{n}$.
2. Compute $\hat{p}_i = A_i p_i$, where $A_i = (\| \mathbf{n} \| H_i + t_i \mathbf{n})^{-1}$.
3. Fit a conic C to the points $\hat{p}_i$.
4. Search over the space of all possible n to minimize the error term:

$$min_{\mathbf{n}} err(C, \hat{p}_i)).$$

There are a number of points worth mentioning. The minimization is over three parameters only due to Step 3 of conic fitting. A large body of literature is devoted to conic fitting and the numerical biases associated with this problem (cf. [6], [2], [5]). The error term in Step 4 is also an important choice: The algebraic error $p^\top C p$ between a point p and a conic C is least recommended because of numerical biases. In our implementation, for example, we have chosen to minimize the distance to the polar line Cp, i.e., $err(C, p) = dist(Cp, p)$. Finally, the search in Step 4 is achieved (in our implementation) by Levenberg-Marquardt optimization using numerical differentiation. Using Matlab, the optimization step consists of simply calling the *leastsq* function.

To summarize, this approach has the two advantages. It is simple and is carried over the 2D plane only. The disadvantages are, first, that the method does not facilitate a priori constraints on the shape of the conic, and second, the method involves a conic fitting (and evaluation) stage which could be challenging on the numerical front.

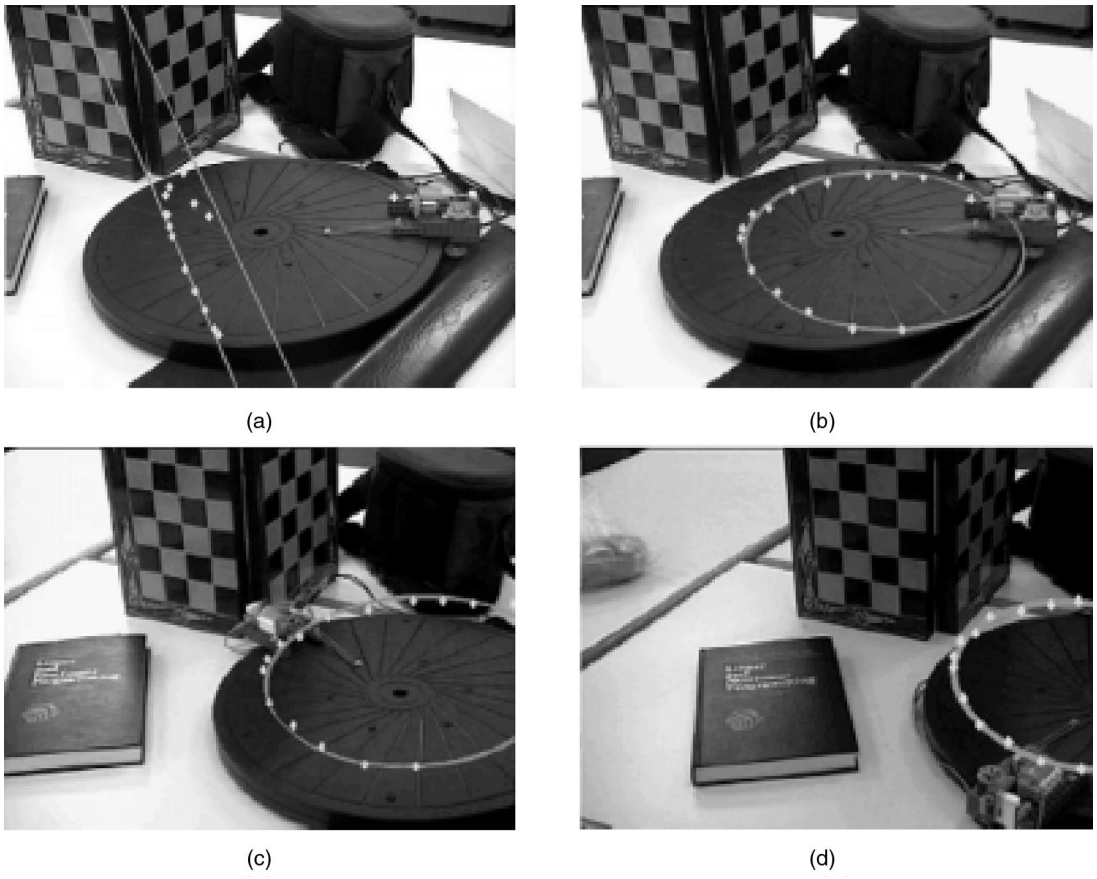


Fig. 10. Using 3D sphere fitting (Method II) to recover a planar conic section. The results are shown by projecting the recovered planar conic (and the 3D points traced along the conic) on several reference images from the sequence. (a) Shows an extreme initial guess with the first image as the reference image, (b), (c), and (d) shows the results of the 3D sphere fitting when the reference image is the first, the middle, and the last images of the sequence, respectively. The resulting radius of the circular path was 5 percent off from the ground truth, and around 4^o off in orientation.

3.2 Method II: Conic Fitting in 3D

In this method, the objective function is minimized in 3D space and is designed such that it can express a priori shape constraints, when available and when cameras are calibrated. The general idea is that a conic in 3D is represented by the intersection of the plane π and a quadric surface. By defining a suitable coordinate system of the quadric surface, one can obtain an eight parameter objective function. In case of calibrated projection matrices and if a priori information about the type of conic is given, say a circle, then the quadric surface representation can be simplified further. We will derive here, a special case in which the sought-after conic is a *circle* in 3D.

In the case of a circle, we wish to represent the arrangement of a sphere and a cutting plane. We expect the total number of parameters to be six (three for $\mathbf{n}$ and three for representing a circle in the plane), yet, a sphere is defined by four parameters. Therefore, an additional constraint is necessary and this is obtained by constraining the plane π to coincide with the center of the sphere. The details are below.

Let p_i and M_i be the projection and camera matrices of frame $i = 1, \ldots, k$, as defined previously. In case the cameras are calibrated, then the projection matrices represent the mapping from an Euclidean coordinate system to the image plane, i.e., $M_i = K_i[R_i \ ; \ u_i]$, where R_i, u_i are the rotational and translational components of the mapping and K_i is an upper-diagonal matrix containing the internal parameters of the camera (focal length, aspect ratio, and principle point). For our needs, since we assume M_i to be known, we can still denote M_i by the composition $M_i = [H_i \ ; \ t_i]$ as was done previously (thus, at this juncture it doesn't really matter whether the camera is calibrated or not). Let the 3D coordinates of the moving point P be denoted (as before) by $P_i = [X_i, Y_i, Z_i]^T$ at time $i = 1, \ldots, k$. We first represent P_i as a function of $\mathbf{n}$ as follows:

$$\lambda_i p_i = M_i P_i. \tag{6}$$

Which after substitution becomes:

$$\lambda_i p_i = [H_i \quad t_i] \begin{bmatrix} P_i \\ 1 \end{bmatrix} \tag{7}$$

$$\lambda_i H_i^{-1} p_i = P_i + H_i^{-1} t_i \tag{8}$$

thus, P_i as a function of M_i and p_i becomes:

$$P_i = \lambda_i H_i^{-1} p_i - H_i^{-1} t_i. \tag{9}$$

Next, we know that the moving point resides on the plane π, thus

$$P_i \mathbf{n} + 1 = 0. \tag{10}$$

After substitution, we obtain

(a) (b) (c)

Fig. 11. Three images from a video sequence of 100 images. We manually track the tip of one of the wings of the propeller. The camera motion was recovered from the static background.

$$\lambda_i = \frac{(H_i^{-1}t_i)^T\mathbf{n} - 1}{(H_i^{-1}p_i)^T\mathbf{n}}. \tag{11}$$

Taken together, (9) and λ_i above, give rise to:

$$P_i = \begin{bmatrix} X_i \\ Y_i \\ \frac{n_1X_i + n_2Y_i + 1}{-n_3} \end{bmatrix}$$

in which X_i, Y_i are functions of $\mathbf{n}$ (and Z_i is eliminated by being expressed as a function of $X_i, Y_i, \mathbf{n}$).

Let the center of sphere be at the coordinates $P_c = [X_c, Y_c, Z_c]$ and its radius R, thus, the points P_i satisfy the constraint:

$$(X_i - X_c)^2 + (Y_i - Y_c)^2 + (Z_i - Z_c)^2 - R^2 = 0 \tag{12}$$

which can be written as,

$$[\,P_i^T \quad 1\,]Q\begin{bmatrix} P_i \\ 1 \end{bmatrix}. \tag{13}$$

The 4×4 symmetric matrix Q is given by:

$$Q = \begin{pmatrix} 1 & 0 & 0 & q_1 \\ 0 & 1 & 0 & q_2 \\ 0 & 0 & 1 & q_3 \\ q_1 & q_2 & q_3 & q_4 \end{pmatrix},$$

where

$$\begin{aligned} q_1 &= X_c, \\ q_2 &= Y_c, \\ q_3 &= Z_c, \\ q_4 &= X_c^2 + Y_c^2 + Z_c^2 - R^2. \end{aligned} \tag{14}$$

Since the center of the circle P_c is on the plane π we have:

$$Z_c = \frac{n_1X_c + n_2Yc + 1}{-n_3} \tag{15}$$

so we need to solve for the three parameters q_1, q_2, q_4. Taken together, each view provides one (nonlinear) constraint (12) over six parameters $\mathbf{n}$ and q_1, q_2, q_4. Thus, seven views are necessary for a unique solution. As with Method I, it is possible to solve for the system over six parameters or to adopt an interleaving approach:

1. Start with an initial estimate of $\mathbf{n}$.
2. Compute the point $\hat{P}_i$ from (9) and (11).
3. Solve for q_1, q_2, q_4 (linear least-squares). P_c, R follow by substitution.
4. Search over the space of all possible $\mathbf{n}$ to minimize the error term:

$$min_{\mathbf{n}}, (dist(\hat{P}_i, P_c) - R)^2, \quad i = 1, \ldots, k,$$

where the search is done using numerical optimization (*leastsq* function of Matlab).

4 Experiments

We have conducted a number of experiments on the straight-line and conic trajectories.

4.1 Line Trajectory

We show the results of two of the experiments conducted. In the first experiment Fig. 5, a sequence of seven images of the moving car was taken with a moving camera (Fig. 5). The static points in the scene were used to compute the camera matrices and we have manually selected a point on the car in the first image and have automatically tracked it through the sequence. Then, we used the algorithm presented in this paper to recover the parameters of the 3D line L. We then projected the line L on one of the images (using the recovered camera matrix M_i).

In the second experiment, Fig. 6, we used a video sequence of 30 frames. Again, the camera was moving an object, a little doll, was moving as well. The camera matrices M_i were recovered using the chess-board appearing in the scene, and a point on the tip of the nose of the doll was manually selected and automatically tracked along the sequence. However, in this experiment we have used just the first 10 even-numbered frames $(2, 4 \ldots, 20)$ to recover the 3D line L, which was then projected on all of the frames of the sequence (using the corresponding M_i camera matrices), thus showing the ability of the method to *predict* the line on which the doll will appear in the last 10 frames. The average distance between the tracked point and the projected line was about one pixel. In addition, we have used the chess-board in the scene to obtain a Euclidean reconstruction and since we recovered the position of the doll at each time instant, we were able to produce a virtual movie where the camera moves freely in 3D space, capturing the doll from different viewpoints at different time instances.

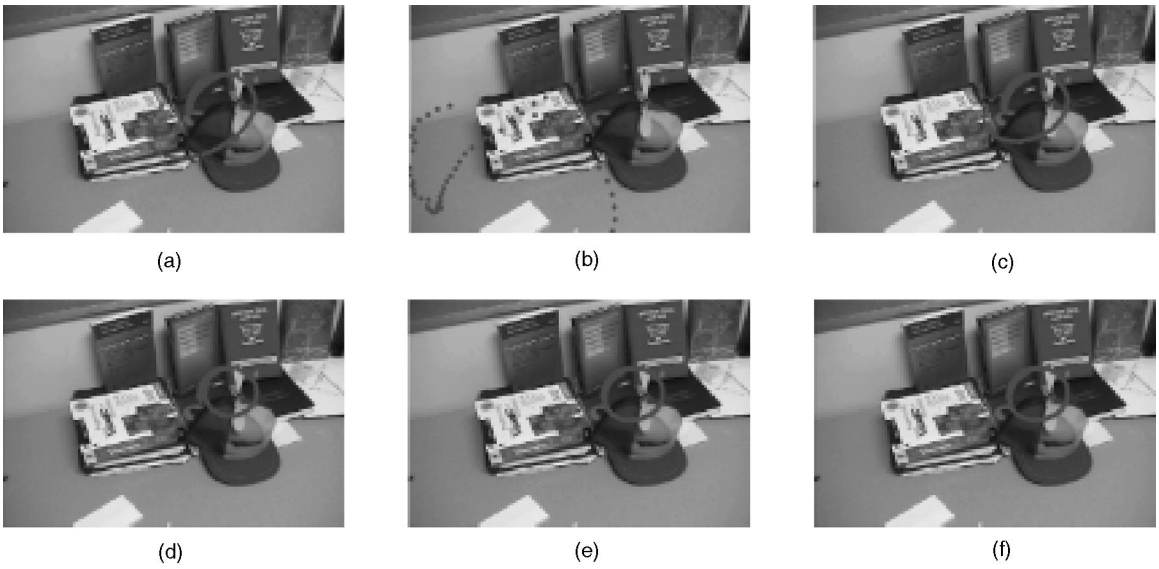


Fig. 12. Convergence of the 2D conic fitting algorithm. (a) show the reprojection of all tracked points on a particular image, given the initial guess of plane parameters. As the algorithm converges ((b)-(f)) the reprojected points form a conic. In this case, it took the algorithm 18 iterations to converge.

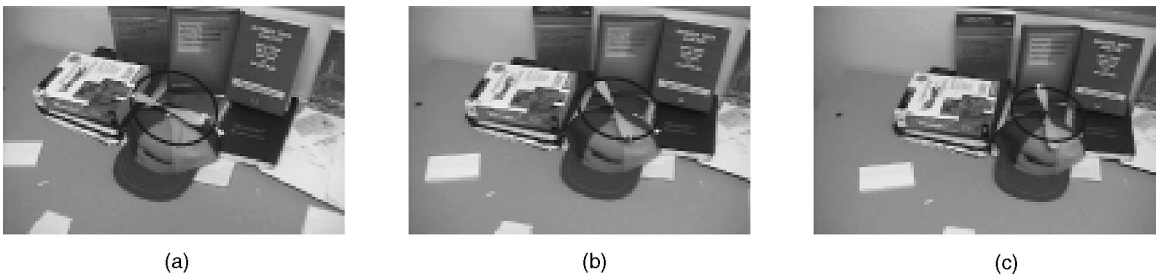


Fig. 13. Result of the convergence. We show the reprojected points and the fitted conic super imposed on three of the original images in the sequence.

4.2 Conic Trajectory

We have conducted a number of experiments on both synthetic and real image sequences. We report here, a typical example of a real image sequence experiment.

A sequence of 16 images was taken with a hand-held moving camera viewing a small Lego piece on a turntable. The Lego piece is therefore moving along a circular path. The first, middle, and last images of the sequence are shown in Fig. 8. The projection matrices were recovered from matching points on the static calibration object (the folded chess-board in the background). The corners of the chess-board were the control points for a linear system for solving for M_i for each image. The linear solution is not optimal, but, was good enough for achieving reasonable results for the trajectory triangulation experiments. A point on the Lego cube was then (manually) tracked over the sequence and its image positions p_i was recorded.

We tested both Methods I and II. In general, the 3D-based optimization (Method II) always converged from any initial guess of $\mathbf{n}$ (the position of the plane π).Fig. 10a shows the conic due to the initial guess that was used for this experiment, for example. The 2D-based optimization (Method I) was more sensitive to the initial guess of $\mathbf{n}$, and Fig. 9a shows a typical initial guess. The remaining displays in Fig. 9 and Fig. 10 show the projection of the final conic (following convergence of the numerical optimization) on the first, middle, and last images of the sequence. In Method II, the reconstructed points in 3D define a circle (as it was constrained to begin with) of a radius 5 percent off from the ground truth and around 4^o in orientation. In Method I, the resulting conic had an aspect ratio of 0.9 (recall that we solved for a general conic), radius roughly 8 percent off, and orientation of the plane was 6^o off.

Another example (using Method I) is shown in Fig. 11, Fig. 12 and Fig. 13. In this case, the moving point is the tip of a rotating wing (on top of the hat) and it took 18 iterations for convergence starting from some arbitrary position of the conic plane. In this case, we do not have ground truth for quantitative evaluation of accuracy, but the superimposed images show a reasonable qualitative fit to the true path.

To summarize, both methods generally behave well in terms of convergence from reasonable initial guesses. Method II was much less sensitive to the initial guess (converged in all our experiments) and generally produced more accurate results. The superiority of Method II most likely is a result of the added constraints which reduces the number of parameters (from eight to six), but at the expense of requiring calibrated cameras.

5 Summary and Future Research

We have introduced a new approach for handling scenes with dynamically moving objects viewed by a monocular moving camera. In a general situation, when both the camera and the target are moving without any constrains, the problem is not solvable, i.e., one cannot recover the 3D position of the target even when the camera ego-motion is known. We have proposed a stratified approach on parametric trajectories and begun the hierarchy with straight-line trajectories. In that case, we have shown that the parameters of the path are uniquely solved given at least five views of the moving target. We then addressed second-order forms which are conic shaped trajectories. In this context, we have introduced two methods. The first method performs the optimization on some arbitrary virtual plane and is very simple. However, it can only deal with general conics only—a priori constraints on the shape of the 3D conic cannot be enforced due to the projective distortion from the conic plane to the virtual common plane. The second method performs the optimization in 3D. The advantage of the second method is that under calibrated cameras it is possible to enforce a priori constraints on the shape of the conic. For example, we have derived the equations necessary for recovering a 3D circular path.

We believe that future work on the family of trajectory triangulation tasks may include the following directions:

- Sliding-window linear or conic trajectory fitting. A reconstruction of a generally moving point can be decomposed onto smaller subproblems in case many (dense) samples of the moving point are available (like in continuous motion).
- A unification of static and dynamic reconstruction. It is possible to estimate whether a point is static or moving simply by the size of the kernel in the case of linear trajectory triangulation. A one-dimensional kernel corresponds to a straight-line path, whereas higher dimensional kernels correspond to a single point (static situation).
- The possibility of recovering both the camera ego-motion and the trajectory (linear or conic) of the point. The task is of a multi-linear nature (for the linear trajectory triangulation) and thus, there may be an elegant way of decoupling the system as is done in the static case. Preliminary work in this direction can be found in [8].
- Unifying second and first order trajectories. Currently, the methods for conic trajectory would fail for a straight-line trajectory as a straight-line path is a singular event for Methods I and II. It is possible, however, to unify both straight and conic paths into a single computational framework if one observes the local tangent to the path along the image sequence [7].
- Handling more complex trajectories by tracking multiple points. If a sufficient number of points are tracked on a rigid body than the full motion of the object (relative to the camera ego-motion) can be recovered. It may be interesting to investigate the possible trajectory shapes when fewer points are available—such as two points.

Acknowledgments

This research was partially funded by DARPA through the U.S. Army Research Labs under grant DAAL01-97-R-9291. We would like to thank Yair Ramati for pointing our attention to the literature on orbit determination and reference [3].

References

[1] S. Avidan and A. Shashua, "Triangulation of lines: Reconstruction of a 3D Point Moving Along a Line from a Monocular Image Sequence," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* June 1999.
[2] F.L. Bookstein, "Fitting Conic Sections to Scattered Data," *Computer Graphics and Image Processing,* no. 9, pp. 56–71, 1979.
[3] P.R. Escobal, *Methods of Orbit Determination.* Krieger Publishing, 1976.
[4] O.D. Faugeras and B. Mourrain, "On the Geometry and Algebra of the Point and Line Correspondences Between N Images," *Proc. Int'l Conf. Computer Vision,* June 1995.
[5] K. Kanatani, "Statistical Bias of Conic Fitting and Renormalization," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 3, pp. 320–326, Mar. 1994.
[6] Y. Leedan and P. Meer, "Estimation with Bilinear Constraints in Computer Vision," *Proc. the Int'l Conf. Computer Vision,* pp. 733–738, Jan. 1998.
[7] D. Segal and A. Shashua, "3D Reconstruction from Tangent-of-Sight Measurements of a Moving Object Seen from a Moving Camera," *Proc. European Conf. Computer Vision (ECCV),* June 2000.
[8] A. Shashua and L. Wolf, "Homography Tensors: On Algebraic Entities that Represent Three Views of Static or Moving Planar Points," *Proc. European Conf. Computer Vision,* June 2000.
[9] P.H.S. Torr, A. Zisserman, and D. Murray, "Motion Clustering Using the Trilinear Constraint Over Three Views," *Workshop on Geometrical Modeling and Invariants for Computer Vision,* 1995.
[10] A. Shashua, S. Avidan, and M. Werman, "Trajectory Triangulation Over Conic Sections," *Proc. Int'l Conf. Computer Vision (ICCV),* pp. 330–337, 1999.
[11] S. Teller, "Computing the Antipenumbra Cast by an Area Light Source," *SIGGRAPH,* vol. 26, no. 2, pp. 139–148, 1992.

Shai Avidan received his BSc degree in mathematics and computer science from Bar-Ilan University, Ramat-Gan, Israel, in 1993, and his PhD from the Hebrew University, Jerusalem, Israel in 1999. He is currently a postdoctoral researcher at the Vision Group at Microsoft Research, Redmond, WA.His research interests are in computer vision and computer graphics with an emphasis on problems in structure from motion and image-based rendering. In addition, he has been working for the past 10 years in the industry in the fields of CAD, GIS, and Photogrammetry.

Amnon Shashua received his BSc. degree in mathematics and computer science from Tel-Aviv, Israel, in 1986, his MSc degree in mathematics and computer science from the Weizman Insititute of Science, Rehovot, Israel, in 1989, and his PhD degree in computational neuroscience, working at the Artificial Intelligence Laboratory, from the Massachusetts Institute of Technology, in 1993. He is currently a senior lecturer at the Institute of Computer Science, The Hebrew University of Jerusalem. His research interests are in computer vision and computational modeling of human vision. His previous work includes early visual processing of saliency and grouping mechanisms, visual recognition, image synthesis for animation and graphics, and theory of computer vision in the areas of three-dimensional processing from a collection of two-dimensional views.