Trajectory Triangulation of Lines: Reconstruction of a 3D point Moving along a Line from a Monocular Image Sequence

Shai Avidan Amnon Shashua
Institute of Computer Science,
The Hebrew University,
Jerusalem 91904, Israel
e-mail: {avidan, shashua}@cs.huji.ac.il

Abstract

We consider the problem of reconstructing the location of a moving 3D point seen from a monocular moving camera, i.e., to reconstruct moving objects from line-of-sight measurements only. Since the point is moving while the camera is moving, then even if the camera motion is known, it is impossible to reconstruct the 3D location of the point under general circumstances. However, 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 instance) can be uniquely recovered, and by linear methods, from at least 5 views. Consequently, we propose a new approach for dealing with dynamic scenes (rich with moving objects) in which once the camera motion is recovered, the 3D trajectory (straight line) of the moving target can be recovered — even when the moving target consists of a single point.

1 Introduction

Consider the situation in which a 3D scene containing a mix of static and moving objects (points) 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. [4, 3]).

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.

Nevertheless, when the shape of the trajectory of the moving point is constrained, say straight line trajectory, then the problem of 3D reconstruction from line-of-sights is solvable - and for some trajectory shapes in a very simple manner. We call the problem of reconstruction of a moving point from line of sight measurements as *trajectory triangulation* and we focus in this paper on straight-line trajectories. Hence, we define our problem as follows (see Fig. 2):

Problem Definition 1 (Linear Trajectory Triangulation) *Given a scene of dynamically moving 3D points, each point moving along some unknown 3D line, seen from a moving camera whose motion is general but known (or can be recovered from static scene points or through other means), reconstruct the trajectory (the 3D line) of each moving point from the known 2D matches across the views.*

Note that we have not placed constraints on the laws of motion along the line. The point can move arbitrarily along a 3D line, thus the only assumption/constraint we are making is that the point is a moving along a straight line. The straight-line assumption is reasonable for a range of applications as people, cars, planes tend to move largely along straight-lines and is also valid in general situations for relatively small time intervals (as an approximation). 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.

2 Trajectory Triangulation of Straight Lines

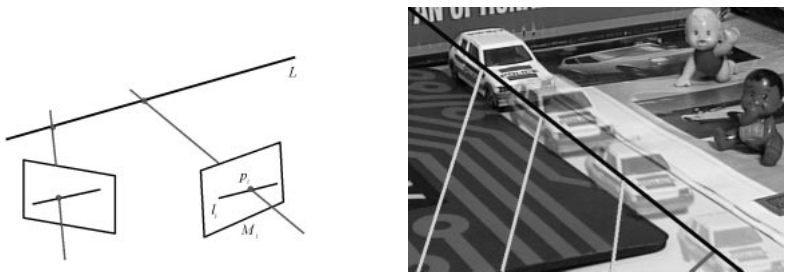
The first question to consider is *how many views are necessary for a unique solution*? One can easily show that 4 views provide two solutions, thus 5 views provide a unique solution. We first address this issue geometrically and then algebraically, as follows.

2.1 A Geometric Interpretation

The geometric picture behind the problem of trajectory triangulation of straight lines is a *linear line complex*, i.e., a set

(a) (b) (c)

Figure 1. Three frames from a sequence in which the car is moving independently of the scene. The camera is moving to the right.



(a) (b)

Figure 2. Illustration of Trajectory Triangulation. Schematic illustration (a) of 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.

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 from the camera at each time instance (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 2 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. 3). 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 4 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. [6]).

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, $i = 1, ..., k$ 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:

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

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 determinants of 2×2 sub-matrices 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

$$L_1 L_6 - L_2 L_5 + L_3 L_4 = 0, \tag{3}$$

for all Plucker coordinates. The space of all 3D lines is therefore embedded in a 5-dimensional projective space and subject

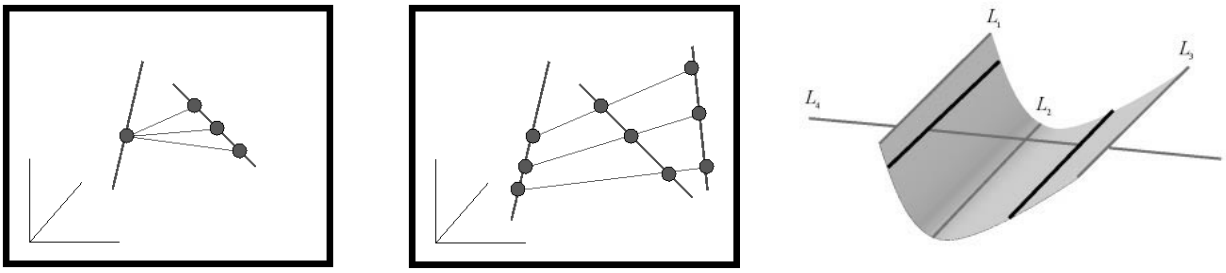


Figure 3. Figures (a) and (b) show that one can define infinite number of lines that intersect two or three given lines, respectively. Given 3 lines (b), there is a unique intersecting line for every point along the first line, thus the set of intersecting lines form a ruled surface (quadric); the fourth line (c) intersects the surface at two points, thus there are two intersecting lines for a general set of 4 lines.

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 [2] 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") operation (eqn. 1) on the pairs of rows of M_i. (M^j stands for the $j-th$ row of camera matrix M)

$$\tilde{M} = \begin{bmatrix} M^2 \wedge M^3 \\ M^3 \wedge M^1 \\ M^1 \wedge M^2 \end{bmatrix} \tag{4}$$

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 5 views provide a unique solution and more than 5 views provide a least-squares solution. Generally, the rank of the estimation matrix is 5 and L is the null space of the estimation matrix.

The quadratic constraint comes into play when the rank of the estimation matrix is 4. This situation arises when the number of views is 4 (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 $L = v_1 + \lambda v_2$. The scalar λ can be found from the quadratic constraint eqn. 3, thus we obtain a second-order constraint on λ 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

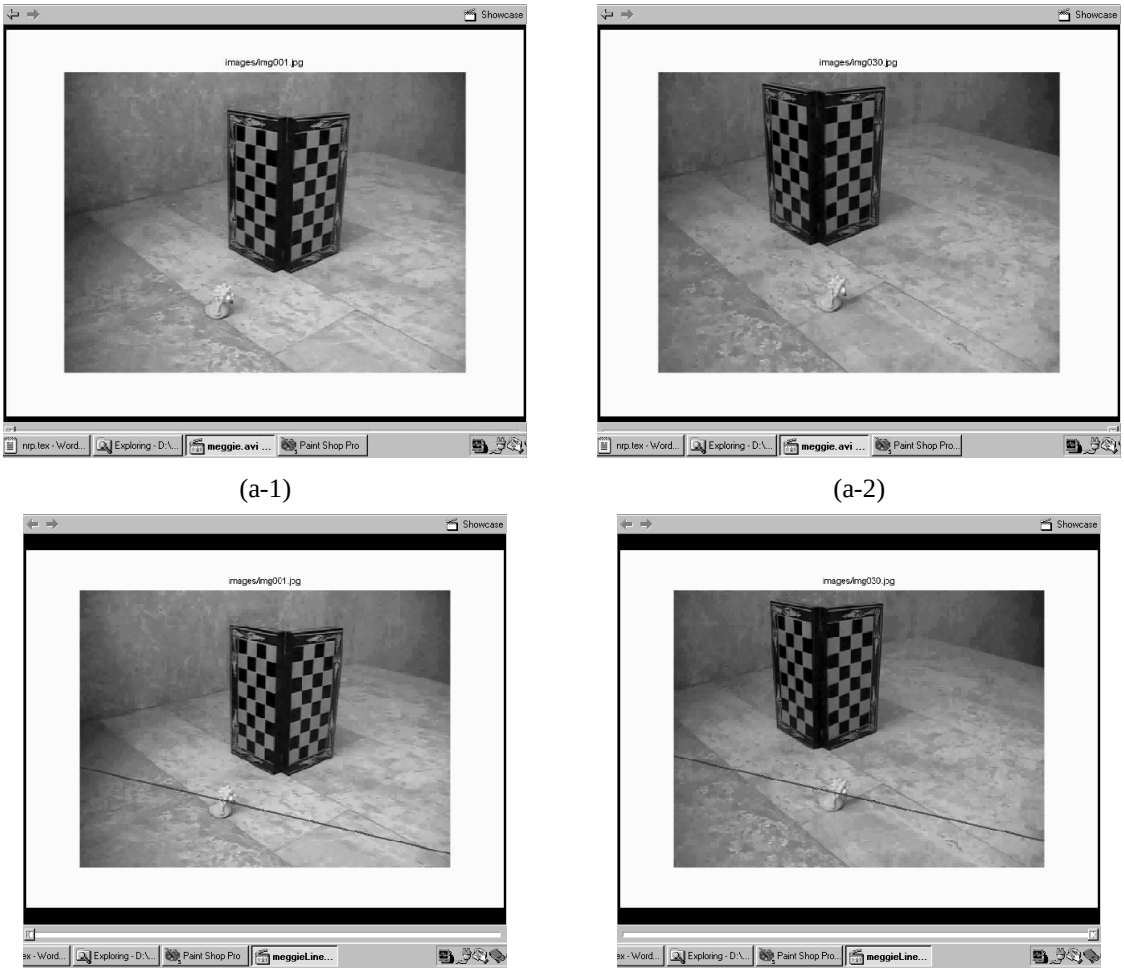
So far we have discussed the case of a single moving point along a straight line viewed by a projective camera. On an object moving along a straight-line path one may possibly track a number of points — those would correspond to a family of *parallel* lines in case the camera matrices M_i are Euclidean. Note from eqn. 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 8 equations for a unique solution for the two parallel lines. More views and/or more points would give rise to an overdetermined 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 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 $Q_1 = [1, X, Y, Z], Q_2 = [0, X + \delta X, Y + \delta Y, Z + \delta Z]$, 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:

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

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 instance i.



(a-1) (a-2)

(b-1) (b-2)

Figure 4. The top tow (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 middle row (b-1,b-2) shows the recovered 3D line L as projected on the extreme two frames. The red dot on Meggie's nose 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 less than 1.5 pixels for all the frames in the sequence.

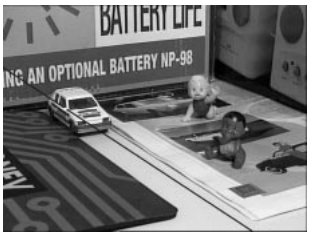


Figure 5. Projecting the 3D line defined by the moving car on one of the images of the sequence.

2.4 Extension to Conic Triangulation

We have also extended the basic approach to deal with curved trajectories by allowing the 3D point to move along a conic (i.e., an elliptic, parabolic or hyperbolic trajectories in space). We show that the 3D conic can be recovered uniquely from 8 views [5], and the algorithm for recovering the conic uses numerical optimization (Gauss-Newton iterations). This problem is reminiscent of the *orbit determination* problem in astrodynamics (cf. [1]), where a set of measurements of a moving object (a planet) from a moving observer (Earth) are used to determine the orbit of the planet which is known to be elliptic. The difference between our algorithm and that of orbit determination lies in the fact that in astrodynamics one assumes motion in a gravitational field, i.e., motion satisfies Keplerian law (equal areas swept by equal times). We do not make any assumption on the motion along the trajectory — the type of conic and its position and orientation in space are recovered regardless of the manner in which the object is moving along the trajectory. Our generalization of the classic work on orbit determination opens up new applications in Computer Vision where dynamic scenes are to be reconstructed.

3 Experiments

We show the results of two of the experiments conducted. In the first experiment Fig. 5, a sequence of 7 images of the moving car was taken with a moving camera (Fig. 1). 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. 4 we used a video sequence of 30 frames. Again, the camera was moving while little Meggie 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 Meggie was manually selected and automatically tracked along the sequence. However, in this experiment we have used just the first 10 even-numbered frames $(2, 4..., 20)$ 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 Meggie will appear in the last 10 frames. The average distance between the tracked point and the projected line was about 1 pixel. In addition, we have used the chess-board in the scene to obtain a Euclidean reconstruction and, since we recovered the position of Meggie at each time instance, we were able to produce a virtual movie where the camera moves freely in 3D space, capturing Meggie from different viewpoints at different time instances.

4 Summary

We have introduced a new approach for handling scenes with dynamically moving objects viewed by a monocular moving camera. In a general situation in which both the camera and the target are moving without any constraint the problem is not solvable, i.e., one cannot recover the 3D motion of the target even when the camera ego-motion is known. We show that by assuming the target is moving along a straight 3D line the problem of recovering the target's trajectory is uniquely solved given at least five views of the moving target.

Acknowledgments

This research was partially funded by DARPA through the U.S. Army Research Labs under grant DAAL01-97-R-9291.

[1] P.R. Escobal. *Methods of Orbit Determination*. Krieger Publishing Co., 1976.

[2] O.D. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between N images. In *Proceedings of the International Conference on Computer Vision*, Cambridge, MA, June 1995.

[3] P. Meer and Y. Leedan. Estimation with bilinear constraints in computer vision. In *Proceedings of the International Conference on Computer Vision*, pages 733–738, Bombay, India, January 1998.

[4] Torr P.H.S., Zisserman A., and Murray D. Motion clustering using the trilinear constraint over three views. In *Workshop on Geometrical Modeling and Invariants for Computer Vision*. Xidian University Press., 1995.

[5] A. Shashua, S. Avidan, and M. Werman. Trajectory triangulation over conic sections. Technical report, Inst. of Computer Science, Hebrew University of Jerusalem, March 1999.

[6] S. Teller. Computing the antipenumbra cast by an area light source. *SIGGRAPH*, 26(2):139–148, 1992.