ABSTRACT: It is known that due to the aperture problem motion cannot be recovered from two frames without additional assumptions. A new method is proposed here, that uses three frames and point or line correspondences to estimate 3-D motion. The algorithm is linearized, efficient and needs no assumptions other than rigidity. Using redundant points and lines the algorithm exhibits stability in the presence of noise. It has been tested with simulated data under a wide variety of conditions.

1. Introduction

Structure from Motion is an area of computer vision that studies the computation of the structure or shape of a scene from successive images. The camera that takes the images can be moving and objects in the scene can be moving as well. There are several approaches to the problem, depending on the assumptions used, on the data available and on the school of thought.

1.1. Aperture Problem

The computation is often divided in two stages. The first stage, which deals with the raw image, computes the image flow, the correspondences of features, or any equivalent transformation of the image between successive frames; the second stage operates on the output of the first and computes the scene structure and motion. Almost always the rigidity assumption is used. This means that either the scene is stationary and the camera moves or the whole scene moves as a rigid body.

The image flow is a two dimensional vector field. The image flow vector at every point in the field represents the apparent movement of the image of a scene point in successive frames. The word apparent means that this motion is not always the same as the projection of the three dimensional motion of the scene point. One reason for this is that the change of brightness of a point can be misleading. The other reason is that along a contour of constant intensity (isophote) the points are indistinguishable. Consider, for example, a straight line. While there are ways to locate a line and match two lines in successive images, under some conditions, it is not possible to trace a point on the line. The line might be moving along its axis in either of the two directions, at any speed, but it looks exactly the same as if it was stationary. Of course one can follow the endpoints, but these might be outside the image, at infinity or occluded; in any case the endpoints are not always reliable and do not give much information about the rest of the points on the line. Since almost any point on an image belongs to an isophote (except for a class of feature points which are distinguishable) this is a major difficulty in obtaining the image flow. In the literature this problem is referred to as the Aperture Problem.

On the other hand, most approaches that are based on feature correspondences avoid the aperture problem by using features of the image that are not affected by it. The cost though of selecting only a small subset of the possible features is high: a large percentage of useful information is unused.

1.2. Approaches to the Aperture problem

There are several suggestions as to how the problem can be tackled. One class of suggestions makes use of some sort of smoothness constraint [2,6,13]. These approaches are based on the fact the images of points on a smooth surface give rise to smooth image flow. Different forms of smoothness constraints have been suggested (minimal second derivatives, locally quadratic surfaces, etc). This introduces another assumption that it is very often violated: that either the flow [2] or the imaged surface [13] are smooth. The corresponding algorithms then, find the smoothest possible solution that fits the data. Of course the scene that is imaged may not be the smoothest even if it is indeed smooth. The main problem here is the lack of information (constraints). The smoothness constraint tries to fill the gap with something that makes sense most of the time. Since there may be many scenes that could give rise to a given pair of images, choosing the smoothest does not always bring us close to the real solution.

Another approach, which is closer to the one presented in this paper, is not to regard the image motion as motion of points on the image surface, but instead to use motion of straight lines as the input to subsequent steps [3,4,8]. Lines have several advantages over points. They are easier to detect and match in successive frames.
and statistical techniques can be used to locate them accurately since they involve many pixels on the image. But most important is the fact that lines do not suffer from the aperture problem (although individual points on a line do suffer), so a major difficulty can be avoided.

The two stages described above (flow computation stage and structure computation stage) change slightly in this context. Instead of computing image flow, one has to detect and trace linear segments on the image. The result is fed to the next step which computes the three dimensional positions of the lines in the image and their motion. This second step we call line based structure from motion. The two stages described above (flow computation stage and structure computation stage) change slightly in this context. Instead of computing image flow, one has to detect and trace linear segments on the image. The result is fed to the next step which computes the three dimensional positions of the lines in the image and their motion. This second step we call line based structure from motion.

A linear solution for it is described in [4.81]. The motion parameters are computed in two steps. The first step solves a linear system of equations to find a set of 27 essential parameters which are nonlinear functions of the motion parameters. In the second step the motion parameters are computed from the essential parameters. If the linear system of equations has a solution then there is a unique solution for the motion parameters if there is a non-zero translation between the three frames. One difference between line based and point based structure from motion [15,12] is that lines need three frames to yield a solution whereas points need two.

1.3. Linear Point and Line Algorithm

The above algorithm gives good results but it is not able to use large part of the information. In any image there are feature points that carry useful information that cannot be used by a line based structure from motion algorithm. Moreover there are small linear segments that cannot be considered either as feature points, because of the aperture problem, or as lines, because their small size degrades the directional information, but they too carry information. In this paper a linear algorithm that can handle both points and lines is presented. The algorithm like all the other linear algorithms for structure from motion, first solves a system of linear equations to obtain the values of a set of essential parameters, and then from them computes the motion parameters. The matrix involved in the linear system is a function of the positions of the points and lines in the image and their motion. If the linear system has a solution then the solution for the motion parameters is unique. The case where one of the translation vectors is zero can be easily resolved.

In the next section the geometric constraints are derived and the need for three frames is discussed. In Section 3 the computation of the motion parameters from the essential parameters is presented. Section 4 deals with the issue of uniqueness and Section 5 contains the experimental results. The paper ends with conclusions and a description of future work.

2. Geometric Analysis

2.1. Camera Geometry We use here a pinhole camera model with a 3-D coordinate system \( OXYZ \) and an image plane parallel to the \( XY \) plane at \( Z=1 \), which sets the focal length equal to unity. An object point \( P_1 = [X_1, Y_1, Z_1]^T \) is projected on the image plane into the point \( p_1 = \left[ \frac{X_1}{Z_1}, \frac{Y_1}{Z_1}, 1 \right]^T \) (Figure 2.1). Often in the literature, point \( p_1 \) is represented as a 2-D vector \( p_1 = \left[ \frac{X_1}{Z_1}, \frac{Y_1}{Z_1} \right]^T \), which is the same thing. We prefer, though, to use 3-D vectors, and not only for uniformity of notation: one can normalize the image vector \( p_1 \) to unit length \( p'_1 = \frac{p_1}{\|p_1\|} \) and thus avoid having the length of the image vector arbitrarily distort the weights in the least squares. Furthermore, when an object point \( P_1 \) rotates to \( R \cdot P_1 \), the corresponding image points are \( p'_1 \) and \( R \cdot p'_1 \) if they are normalized to unity and \( p_1 \) and \( \frac{Z^T \cdot R \cdot p_1}{Z^T \cdot p_1} \) if they are not (\( Z \) is the unit vector along the Z axis). This normalization simplifies things considerably. Note though that we don’t use spherical coordinates but only a notation that is convenient for mathematical manipulations. From now on, we use normalized image point vectors.

2.2. Two Frames, Three Frames. It has been proved [12] that two frames, in the absence of noise, are enough to recover motion and structure. But if noise is present two frames are not enough and the aperture problem just makes things worse. Even if we squeeze every bit of

![Figure 2.1. Basic camera geometry.](image-url)
information out of the data by using an optimal algorithm [9] we can not do many things that we could do using more frames, and it is easy to see why.

Suppose a point \( P_1 \) that belongs to a rigid object undergoes rigid motion with rotation matrix \( R_1 \) and translation vector \( T_1 \) and moves to \( P_2 \). Then

\[
P_2 = R_1 P_1 + T_1
\]

or

\[
\rho_2 \rho_2 = \rho_1 R_1 \rho_1 + T_1
\]

\( \rho_1, \rho_2 \) are the lengths of \( P_1 \) and \( P_2 \). Then the structure (e.g. the length of the vector \( P_1 \)) is

\[
\rho_1 = \frac{(R_1 \rho_1) \times \rho_2}{(R_1 \rho_1) \times \rho_2} ^T \frac{T_1 \times \rho_2}{(R_1 \rho_1) \times \rho_2}
\]

If we consider a third frame then

\[
P_3 = R_2 P_1 + T_2
\]

or

\[
\rho_3 \rho_3 = \rho_1 R_2 \rho_1 + T_2
\]

and the structure is now

\[
\rho_1' = \frac{(R_2 \rho_1) \times \rho_3}{(R_2 \rho_1) \times \rho_3} ^T \frac{T_2 \times \rho_3}{(R_2 \rho_1) \times \rho_3}
\]

In the absence of noise \( \rho_1 = \rho_1' \) should hold. In the presence of noise it doesn’t necessarily hold, if the motion parameters are computed using the frames pairwise. Forcing \( \rho_1 = \rho_1' \) is one more constraint on the motion parameters. So we can get three equations for every point with three frames (Figure 2.2) instead of 2 if we use the frames pairwise. It is important to stress that this is not “yet another” equation. If we needed more equations we could use more points. But this extra equation changes things radically. To say the least the aperture problem is no longer a problem in this three frame formulation and we can easily expand to many frames as in [11].

With the need for three frames established we can go on and describe the theory. The only assumption we need is the rigidity assumption which is expressed by equations (1), (2) above. Indeed these equations are actually equivalent to the rigidity assumption because if a rigid body is undergoing motion then a rotation matrix and a translation vector are enough to describe its motion in the fashion of equations (1), (2). If we now eliminate the structure unknowns \( \rho_1, \rho_2, \rho_3 \) from (1), (2) we are left with a matrix equation in terms of the data (the position of the image point \( p_1 \) in the three frames) of the form (see [7] for the derivation)

\[
\rho_2 [(K, L, M)^* p_1] p_3 = [0]
\]

where if \( p_2 = \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \) then

\[
\begin{bmatrix}
0 & z_2 & -y_2 \\
-z_2 & 0 & x_2 \\
y_2 & -x_2 & 0
\end{bmatrix}
\]

and the same for \( p_3 \). The matrices \( K, L, M \) are

\[
K = T_1 (R_2 \hat{X})^T - (R_1 \hat{X})^T T_2^T
\]

\[
L = T_1 (R_2 \hat{Y})^T - (R_1 \hat{Y})^T T_2^T
\]

\[
M = T_1 (R_2 \hat{Z})^T - (R_1 \hat{Z})^T T_2^T
\]

and the operation \((K, L, M)^* p_1\) is \( x_1 K + y_1 L + z_1 M \)

where \( p_1 = \begin{bmatrix} y_1 \\ x_1 \\ 1 \end{bmatrix} \).

These three matrices do not appear for the first time. In [4, 8] they appear in the solution of the line correspondence problem, which we also solve here as part of the general problem.

Looking at expression (3) one can make some interesting observations. It is a matrix equation that is equivalent to nine scalar equations, only three of which are independent. They are non-linear in terms of the motion parameters, but since we have replaced the non-linear terms with a set of 27 essential parameters (the elements of the three matrices \( K, L, M \)), they are linear with
respect to the essential parameters. For every point we get three independent equations and we need at least nine points to find the essential parameters by solving the system of linear equations that come from applying (3) to several points.

2.3. Lines, Aperture Problem Using equation (3), which is a general constraint, we can derive the constraints for several other problems. We now show how to solve the line problem and the aperture problem. Both are very interesting, the first because we derive the same solution for several other problems. We now show how to solve the line problem and the aperture problem. Both are very famous aperture problems.

Suppose an image point \( p_1 \) is moving and due to the aperture problem we do not know its position in the other two frames but only

\[
p_2 = p_2 + \beta_1 b_1 \\
p_3 = p_3 + \gamma_1 c_1
\]

where \( \beta_1, \gamma_1 \) are undetermined real numbers and \( b_1, c_1 \) are unit vectors parallel to the tangent directions. Since \( \beta_1, \gamma_1 \) are unknowns we eliminate them by pre- and post-multiplying (3) by \( b_1, c_1 \):

\[
0 = b_1^T b_2 \left[ (K, L, M) \ast p_1 \right] b_3 c_1 = (p_2 \ast b_1)^T \left[ (K, L, M) \ast p_1 \right] (p_3 \ast c_1)
\]

or

\[
(p_2 \ast b_1)^T \left[ (K, L, M) \ast p_1 \right] (p_3 \ast c_1) = 0
\]

Now we have one equation with unknowns the elements of \( K, L, M \), and using at least 26 points with tangential motion uncertainty we can find the motion. So as a byproduct we have a 'linear' algorithm that does not suffer from the aperture problem.

The line problem is essentially the same. We represent a line \( e_i \) as a vector normal to both the line and the vector of an image point that belongs to the line. Let \( e_1 = p_1 \ast a_1, e_2 = p_2 \ast b_1, e_3 = p_3 \ast c_1 \) where \( p_1 \) and \( a_1 \) are (as before) a point on the linear element and its direction, respectively. Then (5) implies

\[
e_2^T \left[ (K, L, M) \ast p_1 \right] e_3 = 0
\]

which holds for every image point \( p_1 = p_1' + a_1 a_1 \) where \( a_1 \) is any number. Then

\[
\begin{bmatrix}
e_2^T K e_3 \\
e_2^T L e_3 \\
e_2^T M e_3
\end{bmatrix} (p_1' + a_1 a_1) = 0
\]

or

\[
\begin{bmatrix}
\varepsilon_2 \ast K e_3 \\
\varepsilon_2 \ast L e_3 \\
\varepsilon_2 \ast M e_3
\end{bmatrix} \varepsilon_1 = 0
\]

This is a vector equation that is equivalent to three scalar equations only two of which are independent [4,8].

As we already mentioned equation (3) is linear with respect to the essential parameters \( K, L, M \). So we can compute these parameters. The next step is to find the rotation parameters from the matrices \( K, L, M \). There is a simple and robust way to do all these, that is described in [7]. Due to space limitations it is not presented here.

3. The Issue of Uniqueness

In [10] the issue of uniqueness was discussed in the context of a line based algorithm. Almost anything that applies to lines alone also applies to points and lines. When only lines were involved the solution was proved unique if \( T_1 \neq 0, T_2 \neq 0 \) and \( R_1^T T_2 \neq R_1^2 T_2 \). In this case the solution of the linear system would give two solutions for the translation vector (from the two possible signs of the scale factor): \( K, L, M \) and \( -K, -L, -M \). Only one of them is correct, and it is easy to find out which, because the solution with the wrong sign will give as the 3-D position of the line another line whose image lies in the other side of the vanishing point (both the correct and the spurious solutions have the same vanishing point). It is easy to check which solution is correct by seeing on which side of the vanishing point the line was first detected.

With points the solution is slightly different. It is exactly the same when \( T_1 \neq 0, T_2 \neq 0 \) and \( R_1^T T_2 \neq R_1^2 T_2 \) hold, but even when one of them doesn’t hold there is still a unique solution for points. As in the case of two-frame algorithms the number of solutions, before we go back to the image to decide, is two for the translation and two for the rotation. A total of \( 2 \times 2 = 4 \) solutions. Three of these solutions have either one or both reconstructed images behind the camera [1, 12]. But there is another problem. If one substitutes the values of rotation and translation in \( K, L, M \), when either of the translations is zero the \( K, L, M \) matrices are degenerate (when both are zero there is no solution at all), and no motion parameters can be computed from them. This is an artificial problem, though.

If the motion from frame 1 to frame 2 is \( R_1, T_1 \) and that from frame 1 to frame 3 is \( R_2, T_2 \), and without loss of generality we assume that \( T_1 \) is zero, then we can set up the problem as follows. The motion from 3 to 1 is

\[
R_1' = R_2, T_1' = -R_2^T T_2
\]

and the motion from 3 to 2 is

\[
R_2' = R_2, T_2' = T_1 - R_2^T T_2.
\]

Clearly \( R_1' T_1' = R_2^T T_2 \). But this is not a degenerate form. We can compute the motion parameters. The only thing we cannot do is narrow down the number of solutions from four to two. (Of course in this case any number of lines is not going to do any good in terms of noise sensitivity).
4. Experiments

Several experiments were conducted to determine the sensitivity of the method to noise and the improvement one can get by using both lines and points. The experiments were done using synthetic images with the randomly generated object (a collection of random points and lines) at some distance from the camera, random translation and rotation, and random noise.

The noise is an additive random vector. For points it is added to the image point vector and for lines it is added to the image line vector (a vector that passes through the origin and is normal to the plane defined by the image line and the origin). The same amount of noise was used for both the lines and the points although one would expect the positions of lines to be detectable with higher accuracy than points. In the graphs below the noise figure is given in focal lengths. Thus 1.0e-2 noise for a camera with a 512 x 512 image plane of size one focal length, a slightly wide-angle lens-camera configuration, is about five pixels.

In the following experiments the “object” is 3 units away, 2.5 units wide, and the translation is about one unit. We first compare the sensitivity of the algorithm to noise when points only, lines only and both points and lines are used (Figures 4.1–2).

Figure 4.1: These two graphs show the response of the algorithm to noise when only points or only lines are used. Notice that even with points alone the performance is good. With around two pixels error the translation (which is more sensitive to noise) is off by less than 10%

Figure 4.2: When points and lines are used together the result is better by a factor of about 2.

Figure 4.3: With 160 lines and 240 points, the error improves by a factor of about 3.

The next set of experiments (Figure 4.3) shows the performance of the algorithm for different numbers of points and lines.

The sensitivity to noise increases as the object is positioned further away. This is due to two main factors. The first is that the image of the object becomes smaller. The other is that while the size of the translation remains the same, it becomes smaller compared to the distance to the object. This is illustrated by Figure 4.4.

Of very great interest in a structure from motion method is the improvement one can get from an increase in the number of feature points or lines. Figure 4.5 plots the error in the output vs. the number of lines and points respectively. In all the runs the error is the same. From the results it becomes obvious that there is a tremendous advantage in increasing the number of points and lines, a fact that gives the methods that compute a dense flow field (or correspondence or disparity field) an advantage.

5. Conclusions

Several more things can be done along this path. In [9] mathematical techniques were suggested to achieve
The change of distance has a large effect, especially on the translation. These techniques are applicable here also [11]. The robustness of the method is very promising. Future problems that need to be addressed are how to optimally decipher the \( K, L, M \) matrices, since the method presented here is not optimal in any meaningful sense, although it is very close to optimal.

One more advantage of this approach is that it depends on only one assumption: rigidity. It would be interesting to see how the method can be improved by incorporating a smoothness assumption, when such an assumption holds, or to relax the rigidity assumption and replace it with something weaker.

References