Iterative computation of 3D plane parameters

H. Baltzakis\textsuperscript{a}, P.E. Trahanias\textsuperscript{b,∗}

\textsuperscript{a}Institute of Computer Science, Foundation for Research and Technology — Hellas, P.O. Box 1385, 71110 Heraklion, Crete, Greece
\textsuperscript{b}Department of Computer Science, University of Crete, P.O. Box 2208, 71409 Heraklion, Crete, Greece

Received 28 February 2000; revised 25 July 2000; accepted 27 July 2000

Abstract

Knowledge of the position and orientation of 3D planes that exist in a scene is very important for many machine vision-based tasks, such as navigation, self-localization and docking. In this paper we propose two iterative methods for robustly computing the plane parameters out of given image sequences. Both methods include the (limited) ability to determine unknown parameters of the 3D-camera motion.

Keywords: 3D plane estimation; Iterative methods; Camera motion; Point correspondences

1. Introduction

Most vision-based navigation tasks usually rely on certain scene (environmental) features, which provide adequate information for the task at hand. Planar regions are important aspects in indoor workspaces and are frequently employed in machine vision problems. They constitute very common structures in man-made environments and play a crucial role when dealing with certain tasks, such as indoor navigation, self-localization and docking. Applications aiming to derive and use plane parameters in order to achieve such tasks usually involve two distinct phases: (a) scene segmentation to extract planar areas, and (b) estimation of plane parameters for the areas of interest. Projective invariants \[7,14\] provide a means to tackle the former phase, while, for the latter phase, depending on the actual task at hand, various approaches have been proposed to obtain analytical solutions of the plane unknowns \[4,5,11\].

The estimation of 3D plane parameters is a specific case of the reconstruction problem. This is due to the fact that knowledge of the plane position and orientation provides enough information to determine the 3D coordinates of each point on it. On the other hand, a solution to the reconstruction problem (estimation of 3D point coordinates) provides also the exact plane parameters through the application of best-fit algorithms.

The pursuit of any task regarding recovery of 3D information usually involves, as a first, unavoidable step, the establishment of point correspondences. Even in cases where, theoretically, only a small number of point correspondences is actually needed (e.g. three points are enough to completely determine all parameters of a plane), it is usual practice to overconstrain the problem by employing a larger set of low-confidence correspondences and extracting the results by various kinds of error minimization algorithms. The establishment of large sets of low-confidence correspondences is usually pursued by employing image displacement measures, such as optic flow \[1,3,13\]. For utilizing established point correspondences in order to compute 3D space coordinates, camera motion must also be known. The problem of extracting camera motion has often been addressed in the past \[2,6,9,10,15\] either as a stand-alone problem, or, as an inevitable step in approaches aiming at various aspects of the reconstruction problem.

An inherent constraint in methods trying to resolve both camera motion and plane parameters is that translation magnitude and plane distance cannot be simultaneously determined \[8\]. This, stems from the fact that optical stimuli received from planes lying far from a fast-moving observer (camera) are equivalent to stimuli received from planes lying closer to a slower-moving observer. A priori knowledge about either the plane distance or the camera translation magnitude is necessary in order to overcome this ambiguity and obtain unique solutions. In practical applications, such knowledge may be inferred from other sources, such as motion encoders of a mobile platform or given 3D measurements of the workspace.

In this paper we present two novel, iterative methods for...
the estimation of 3D plane parameters, based on the minimization of a cost function. Assuming known translational magnitude, both the plane distance and plane normal direction are computed. When the magnitude of the translational motion is not available, plane distance is only inferred relatively to this unknown magnitude. The iterative nature of the respective algorithms provides the ability to determine simultaneously both the plane parameters and the camera motion (rotation parameters and direction of translation). This is facilitated by including two terms in the cost function: the cost arising from false estimation of the plane parameters and the cost arising from false estimation of the camera motion.

The proposed methods are dual ones, in the sense that the first starts from (assumed) known point correspondences and iteratively computes the plane parameters, whereas the second starts from an initial plane estimation, which is refined through the derivation and evaluation of point correspondences. According to the first method (forward approach), 3D point coordinates are computed out of known point correspondences and presumed camera motion parameters. The resulting 3D point coordinates are evaluated according to their distance from a plane with presumed parameters and the error is calculated as the percentage of points that have a distance greater than a (variable) threshold. The presumed motion and plane parameters change in each iteration using an error reduction scheme much like the well-known simulated annealing scheme. In addition to these parameters, as the error becomes smaller, the variable distance threshold becomes smaller, too.

According to the second method (backward approach), 3D point correspondences are analytically computed from presumed motion and plane parameters. The computed point correspondences are evaluated by mapping them on the original images. The resulting error percentage constitutes the error function that is utilized by the iterative algorithm. This method eliminates the need for estimation of point correspondences, relying only on initial (rough) estimates of plane and motion parameters.

Both methods rely on known equations for the computation of 3D coordinates and camera motion parameters. However, their innovative formulation as iterative approaches furnishes them with enhanced performance and robustness, even in cases of limited a priori information. Moreover, this formulation relaxes the need for establishing accurate point correspondences, whereas in the case of the backward approach it completely eliminates the need for computing point correspondences out of the image data. A significant advantage of both methods is the ability to utilize more than one pair of frames for the sake of robustness. This is extremely meaningful in cases where the additional frames introduce no extra unknown parameters (e.g. when the camera motion is constant). Both methods were tested extensively and yielded very promising results for real and artificial indoor sequences. In order to evaluate the results, a comparative test against the well-known method of the flow matrix [8] is also presented.

2. 3D plane estimation

2.1. Mathematical preliminaries

Let \( c_0 \) be a camera and let \( O^0, x^0, y^0, z^0 \) be its orthogonal coordinate system defined by the optical center \( O^0 = (O^0_x, O^0_y, O^0_z) \) and the unit vectors \( \hat{x}^0, \hat{y}^0, \hat{z}^0 \). Let \( c_1 \) be a second camera with optical center and unit vectors, defined with respect to the coordinate system of the first camera, \( O^1 = (O^1_x, O^1_y, O^1_z) \) and \( \hat{x}^1, \hat{y}^1, \hat{z}^1 \), respectively. Also, let \( P \) be a space point lying on a plane, with \( (X^0, Y^0, Z^0) \), \( (X^1, Y^1, Z^1) \) its coordinates with respect to the two cameras, \( c_0 \) and \( c_1 \).

If \( R^{0\rightarrow1} \) is the rotation matrix (the matrix having as columns the unit vectors \( \hat{x}^1, \hat{y}^1, \hat{z}^1 \)), it can be shown [8] that:

\[
\begin{bmatrix}
X^0 \\
Y^0 \\
Z^0
\end{bmatrix} = -O^1 + R^{0\rightarrow1} \begin{bmatrix}
X^0 \\
Y^0 \\
Z^0
\end{bmatrix}
\]

which expands to,

\[
\begin{bmatrix}
X^0 \\
Y^0 \\
Z^0
\end{bmatrix} = -\begin{bmatrix} O^1_x \ O^1_y \ O^1_z \end{bmatrix} + \begin{bmatrix} \hat{x}^1 \ \hat{y}^1 \ \hat{z}^1 \end{bmatrix} \begin{bmatrix}
X^0 \\
Y^0 \\
Z^0
\end{bmatrix}
\]

(2)

Eq. (2) associates the 3D coordinates of point \( P \) with respect to \( c_1 \) with the coordinates of the same point as seen from \( c_0 \). The projections of point \( P \) on the image planes of the two cameras are given as

\[
(x^0, y^0) = \left( f \frac{X^0}{Z^0}, f \frac{Y^0}{Z^0} \right)
\]

(3)

\[
(x^1, y^1) = \left( f \frac{X^1}{Z^1}, f \frac{Y^1}{Z^1} \right)
\]

(4)

where \( f \) is the camera focal length.

2.2. The forward approach

If point correspondences are known, that is, for each point \( (x^0, y^0) \) on the image plane of the first camera, its corresponding point \( (x^1, y^1) \) on the second camera is known, the assumption of known camera motion (known position and orientation of \( c_1 \) with respect to \( c_0 \)) may be employed to
become, if we compute the product on the right side of the above equation, we obtain an expanded form of the above equation as:

\[
\begin{bmatrix}
\frac{x_i}{f} \\
\frac{y_i}{f} \\
\frac{Z_i}{f}
\end{bmatrix}
= -\begin{bmatrix}
O_{x_i}^1 \\
O_{y_i}^1 \\
O_{z_i}^1
\end{bmatrix}
+ Z_i
\begin{bmatrix}
\frac{x_0}{f} \\
\frac{y_0}{f} \\
\frac{Z_0}{f}
\end{bmatrix}
\]

(5)

If we compute the product on the right side of the above equation, we obtain an expanded form of the above equation as:

\[
\begin{bmatrix}
\frac{x_i}{f} \\
\frac{y_i}{f} \\
\frac{Z_i}{f}
\end{bmatrix}
= -\begin{bmatrix}
O_{x_i}^1 \\
O_{y_i}^1 \\
O_{z_i}^1
\end{bmatrix}
+ \begin{bmatrix}
\frac{x_0}{f} + \frac{x_0}{f} + \frac{y_0}{f} + \frac{z_0}{f}
\\
\frac{x_0}{f} + \frac{y_0}{f} + \frac{z_0}{f}
\\
\frac{x_0}{f} + \frac{y_0}{f} + \frac{z_0}{f}
\end{bmatrix}
\]

(6)

By splitting the above matrix equation to the three imposed element equations, and by substituting the value of \(Z_i\), as it is eliminated from the third equation, the first two equations become,

\[
\begin{align*}
x_i^1 \frac{1}{f} \left( -O_{x_i}^1 + Z_i \left( \frac{x_0}{f} + \frac{y_0}{f} + \frac{z_0}{f} \right) \right) \\
y_i^1 \frac{1}{f} \left( -O_{y_i}^1 + Z_i \left( \frac{x_0}{f} + \frac{y_0}{f} + \frac{z_0}{f} \right) \right)
\end{align*}
\]

(7)

\[
\begin{align*}
x_i^1 \frac{1}{f} \left( -O_{x_i}^1 + Z_i \left( \frac{x_0}{f} + \frac{y_0}{f} + \frac{z_0}{f} \right) \right) \\
y_i^1 \frac{1}{f} \left( -O_{y_i}^1 + Z_i \left( \frac{x_0}{f} + \frac{y_0}{f} + \frac{z_0}{f} \right) \right)
\end{align*}
\]

(8)

Each of the above equations can be individually utilized in order to compute the value of \(Z_i\) when the camera motion and the point correspondences are known (or assumed). Actually, each of the above equations involves only one component of the motion projection and yields a solution for \(Z_i\), according to that component only. More specifically, Eq. (7) involves only the horizontal component, while Eq. (8) involves only the vertical component. Ideally, if the assumed camera motion were correct, solutions coming from both equations would be identical, corresponding to correct \(Z_i\) values. In practice, the two solutions obtained differ owing to lack of knowledge of the true camera motion. This is taken into account in our implementation by considering the average of the two solutions.

Finally, by substituting the resulting computed value for \(Z_i\) in Eq. (3), we obtain the full 3D coordinate vector for point P. Repetition of the above procedure for many coplanar points provides a set of 3D coordinates that, ideally, satisfy a plane equation (with respect to the coordinate system \(c_0\) of the form:

\[
aX^{\text{bo}} + bY^{\text{bo}} + cZ^{\text{bo}} + d = 0
\]

(9)

In practice, even in the case where the assumed camera motion is correct, because of inaccuracies in the point correspondences, the above equation will not be satisfied. Rather, an equivalent inequality will hold true, which can be formulated as

\[
|aX^{\text{bo}} + bY^{\text{bo}} + cZ^{\text{bo}} + d| \leq \frac{D}{2}
\]

(10)

The geometric interpretation of the above expression is that each computed point lies between two parallel planes a distance \(D\) apart; this is illustrated in Fig. 1. Whatever assumptions of the plane parameters and the camera motion we make, there will always be a value of \(D\) such that, all computed 3D points will satisfy Eq. (10).

Based on the above inference, the following iterative algorithm implements the first proposed (forward) method for computing both the plane and motion parameters:

**Forward iterative algorithm**

i. Initialization of the (unknown) plane parameters \(a, b, c, d\), the camera motion parameters, and the threshold \(D\).

ii. For each of the available image pairs repeat:

  o For each of the unknown parameters repeat:

    – Compute the 3D coordinates of all the plane points as described in Section 2.1.

    – Compute the error as the percentage of points that do not lie between the planes \((a, b, c, d - D/2)\) and \((a, b, c, d + D/2)\), i.e. the error is the percentage of points that do not satisfy Eq. (6).

    – If the error is less than a threshold, then decrease parameter \(D; D \leftarrow pD\), \(p < 1\).

    – If the error is greater than a threshold, then increase parameter \(D; D \leftarrow qD\), \(q > 1\).
we obtain:

\[ ax^b + bx^c + cZ^d + d = 0 \] (11)

where \( a, b, c, \) and \( d \) are the parameters of the plane with respect to the coordinate system of \( c_0 \). By solving Eq. (11) for \( Z^d \), we obtain

\[ Z^d = \frac{df}{ax^b + bx^c + cf} \] (12)

The above value for \( Z^d \) can be substituted in Eq. (1) resulting in:

\[
\begin{bmatrix}
X^i \\
Y^i \\
Z^i
\end{bmatrix} = \begin{bmatrix}
O^i_1 & \frac{1}{ax^0 + by^0 + cf} R^{10m-1i} \\
O^i_2 \\
O^i_3
\end{bmatrix} \begin{bmatrix}
x^0 \\
y^0 \\
d
\end{bmatrix}
\] (13)

Moreover, expanding the rotation matrix in elements and combining the left part of Eq. (13) with the equations for the projective transformation that takes place in \( c_1 \) (Eq. (4)), we obtain:

\[
x^i = \left( O^i_1 + \frac{x^0 x^i + y^0 y^i + z^i}{ax^0 + by^0 + cf} \right) / O^i_3
\]

\[
\times \left( O^i_2 + \frac{x^0 x^i + y^0 y^i + d^i}{ax^0 + by^0 + cf} \right)
\] (14)

and

\[
y^i = \left( O^i_1 + \frac{x^0 x^i + y^0 y^i + z^i}{ax^0 + by^0 + cf} \right) / O^i_3
\]

\[
\times \left( O^i_2 + \frac{x^0 x^i + y^0 y^i + d^i}{ax^0 + by^0 + cf} \right)
\] (15)

Eqs. (14) and (15) indicate that knowledge of the relative positions of the two cameras and the plane parameters provides enough information to analytically compute point correspondences for each plane point projected on the image planes.

It is a straightforward task to test the accuracy of calculated point correspondences, just by mapping each couple of implied image coordinates on the original images and checking whether their respective intensity values match. Since the accuracy of the calculated image points is heavily dependent on the accuracy of the assumptions made about the camera motion and the plane equation, the above technique provides a means for evaluating the correctness of all parameters involved. Assumptions far from the correct ones give rise to point correspondences that are false, and, hence, lead to a very small number of verified point matches. As the assumptions become better, the number of correctly matched points will eventually increase.

This inference forms the basis for devising an iterative algorithm for computing the plane parameters and the camera motion:

**Backward iterative algorithm**

i. Initialization of the (unknown) plane parameters \( a, b, c, \) and the camera motion parameters.

ii. For each of the available image pairs repeat:

- Calculate point correspondences using Eqs. (14) and (15).
- Evaluate the point correspondences by backprojecting them on the image plane and matching them with the original image data. Extract the error as the percentage of unmatched points.
- Change the current parameter towards the direction that minimizes the error.

iii. If no parameters have changed in step ii then exit. Otherwise, goto step ii.

**2.4. Implementation notes**

For implementing both iterative algorithms, simulated annealing-like schemes have been employed. While eliminating the need for computation of error-space gradients, their probabilistic nature ensures convergence even in cases where local error minima are reached during the iterative process. This has also been verified through extensive experimentation; in all cases, both algorithms were able to rapidly converge to the correct set of parameters (see also the next section).

The values of the parameters appearing in both algorithms have been experimentally set; their operational values correspond to our intuition of the respective quantities. Parameters intrinsic to the simulated annealing algorithm (such as temperatures and step values) are set according to standard techniques in the field.

Threshold \( D \), needed for the forward algorithm, is initially set to a large value and, during the iterative process, increases or decreases so that the error stays within acceptable limits. In our implementation, threshold \( D \) decreases by a factor \( p = 0.9 \) whenever the error percentage drops below 5%. On the other hand, when the error percentage becomes larger than 60%, \( D \) increases by a factor \( q = 1.5 \).

In the case of the backward algorithm, step ii involves the evaluation of point correspondences. This is performed by backprojecting them on the initial image plane and...
correlating the corresponding pixel intensity values in a small neighborhood. The presence of intensity variations (e.g. texture) on the imaged scene is assumed in order for this matching procedure to work effectively.

When more than two image frames are available, slight modifications to the above algorithms can be made in order to utilize the extra information. This is extremely meaningful in cases where the additional frames introduce no extra, unknown parameters (e.g. when the camera motion is constant).

3. Experimental results

Both methods presented were tested extensively with artificial and real image sequences. Sample results from these experiments are presented here that demonstrate the effectiveness of the proposed methods.

As mentioned earlier, there is always an inherent ambiguity between the camera translation magnitude and the distance of the plane. To cope with that, available ground truth for the camera translation magnitude was employed in our experiments.

3.1. Artificial data

Fig. 2 shows two consecutive frames of an artificial sequence of a textured wall that has been employed in our experiments; ground truth is available for this sequence and hence it can be used for experimental verification. The camera is initially placed at a distance of 250 units from the wall and faces it with an angle of 45°. The camera motion consists of a translation of 10 units towards the direction of its optical axis (z-axis) and a rotation of 1° around its vertical axis (y-axis).

Fig. 3 shows the estimation of the plane and motion parameters during the iterative process of the backward algorithm. For the sake of clarity of presentation, camera rotation is shown as the Euler angles \( \alpha, \beta, \) and \( \gamma \) that correspond to the successive rotation operations around the unit vectors \( \hat{x}, \hat{y}, \hat{z} \) giving rise to the rotation matrix \( R_{b_0}^{b_1} \) (see Eq. (1)) [12].

As can be observed, the algorithm succeeded in reaching the correct values in less than twenty iterations.

For applying the forward algorithm on the same sequence, we calculated the optical flow field analytically. The results obtained are illustrated in Fig. 4. The threshold distance \( D \) was initially set to a very large value causing the error to be very small and the unknown parameters to vary in an almost random way. As the threshold \( D \) was becoming smaller, the unknown parameters gradually converged to the desired values.

3.2. Real data

For evaluating the performance of the methods with real data, we used the well-known coca-cola sequence (Fig. 5). In this sequence the camera is translating with a constant velocity in the direction of its optical axis. Ground truth concerning both the motion and the camera parameters is available, but not for the imaged scene (depth of image points).

Both methods were applied in order to compute the
parameters of two planes: (a) the plane defined by the background wall, and (b) the plane defined by the table. These planes were modeled in our experiments by constraining the algorithms to rectangular areas, manually placed on the imaged scene. The plane of the background wall was modeled by two rectangles, as shown in the left image (case a) of Fig. 5. The plane of the table was modeled by one rectangle, as shown in the right image (case b) of Fig. 5. Both methods quickly converged to values consistent with the ground truth provided (in case of the camera motion) and our visual perception (in case of the plane parameters). The results are summarized in Table 1.

In addition to the above sequence, results from a real sequence (Fig. 6), acquired in a corridor outside our laboratory, are provided. In this experiment the camera axis was forming an angle of 45° with the wall while translating in a direction parallel to it. Despite the lack of accurate knowledge of the camera parameters, the results were quite accurate and are illustrated in Table 2.

Finally, a comparative test with the well-known method of the flow matrix [8] was performed. The framesets used were artificial, depicting the wall shown in Fig. 2 from various positions and directions and with varying camera motion. The results are summarized in Table 3.

The results in Table 3 are only indicative of the ability of the methods to converge to correct parameters. The accuracy of the algorithms, especially of the iterative ones, can, in most cases, be tuned to be more accurate by choosing appropriate thresholds, or just by letting the algorithm perform more iterations. As can be observed in the above table, all methods performed well in the cases where the magnitude of the camera motion was relatively small. As
the camera motion, and especially rotation, grew larger, the flow matrix method failed to give accurate results. This failure of the flow matrix method may be explained by the fact that it employs the simplified equations giving instant motion estimations, instead of the full rotation matrix expressions employed by both iterative methods. Moreover, the iterative nature of the proposed methods furnishes them with the ability to converge to the actual parameter values, even in cases where the initial estimations are rather inaccurate.

4. Discussion

In this paper two new, iterative methods for plane parameter estimation have been proposed. The methods start from known relations describing the camera motion and induced motion on the image plane and present novel formulations as iterative procedures that result in effective computational implementations. Both methods were tested extensively and yielded very good results for both artificial and real image sequences.

According to the first (forward) method, the solution of the problem of plane and motion parameters is seen as a simple “count points that match the assumptions” problem. This eliminates the influence of distorted point correspondences and facilitates convergence even if the initial assumptions are far from the correct values.

According to the second (backward) method, point correspondences are analytically computed from estimated plane and motion parameters. This approach is completely different from existing ones, in that the problem is assumed...
completely reversed, starting from what, usually, is what we seek for. Hence, no a priori point correspondences are required and the method can be applied directly to raw image data.

The iterative — simulated annealing-like — nature of the methods furnishes them with the ability to relax the requirement for accurate knowledge of point correspondences and, at the same time, to provide robustness not easily seen in straightforward, equation-reversion methods. The experimental results obtained have demonstrated that iterative approaches are applicable for plane parameter estimation. At the expense of simplicity, more sophisticated iterative algorithms could be designed in order to achieve even faster convergence.

References