Recovery of Ego-Motion Using Image Stabilization *

Michal Irani† Benny Rousso Shmuel Peleg

Institute of Computer Science
The Hebrew University of Jerusalem
91904 Jerusalem, ISRAEL

Abstract

A method for computing the 3D camera motion (the *ego-motion*) in a static scene is introduced, which is based on computing the 2D image motion of a single image region directly from image intensities. The computed image motion of this image region is used to register the images so that the detected image region appears stationary. The resulting displacement field for the *entire* scene between the registered frames is affected only by the 3D translation of the camera. After canceling the effects of the camera rotation by using such 2D image registration, the 3D camera translation is computed by finding the focus-of-expansion in the translation-only set of registered frames. This step is followed by computing the camera rotation to complete the computation of the ego-motion.

The presented method *avoids* the inherent problems in the computation of optical flow and of feature matching, and does not assume any prior feature detection or feature correspondence.

1 Introduction

The motion observed in an image sequence can be caused by camera motion (ego-motion) and by motions of objects moving in the scene. In this paper we address the case of a camera moving in a static scene. Complete 3D motion estimation is difficult since the image motion at every pixel depends, in addition to the six parameters of the camera motion, on the depth at the corresponding scene point. To overcome this difficulty, additional constraints are usually added to the motion model or to the environment model.

3D motion is often estimated from the optical or normal flow derived between two frames [1, 12, 22], or from the correspondence of distinguished features (points, lines, contours) extracted from successive frames [10, 13, 7]. Both approaches depend on the accuracy of the feature detection, which can not always be assured. Methods for computing the ego-motion *directly* from image intensities were also suggested [11, 14].

Camera rotations and translations can induce similar image motions [2, 8] causing ambiguities in their interpretation. At depth discontinuities, however, it is much easier to distinguish between the effects of camera rotations and camera translations, as the image motion of neighboring pixels at different depths will have similar rotational components, but different translational components. Motion parallax methods use this effect to obtain the 3D camera motion. [18, 17, 7]. Other methods use motion parallax for shape representation and analysis [23, 6, 9].

In this paper a method for computing the ego-motion directly from image intensities is introduced. At first only 2D image motion is extracted, and later this 2D motion is used to simplify the computation of the 3D ego-motion.

We use previously developed methods [15, 16] to detect and track a single image region and to compute its 2D parametric image motion. It is important to emphasize that the 3D camera motion *cannot* be recovered solely from the 2D parametric image motion of a single image region, as there are a couple of such 3D interpretations [20]. It was shown that 3D motion of a planar surface can be computed from its 2D affine motion in the image and from the motion derivatives [21], but motion derivatives introduce sensitivity to noise. Moreover, the problem of recovering the 3D camera motion directly from the image motion field is an ill-conditioned problem, since small errors in the 2D flow field usually result in large perturbations in the 3D motion [2].

To overcome the difficulties and ambiguities in the computation of the ego-motion, we introduce the fol-

*This research has been sponsored by the U.S. Office of Naval Research under Grant N00014-93-1-1202, R&T Project Code 4424341—01.

†M. Irani is now with David Sarnoff Research Center.

lowing scheme: The first frame is warped towards the second frame using the computed 2D image motion at the detected image region. This registration cancels the effects of the camera rotation for the *entire* scene, and the resulting image displacements between the two *registered* frames are due only to the 3D translation of the camera. This translation is computed by locating the FOE (focus-of-expansion) between the two *registered* frames. Once the 3D translation is known it can be used, together with the 2D motion parameters of the detected image region, to compute the 3D rotation of the camera by solving a set of *linear* equations.

The 2D image region registration technique used in this work allows easy decoupling of the translational and rotational motions, as only motion parallax information remains after the registration. As opposed to other methods using motion parallax [18, 19, 17, 7], our method does not rely on 2D motion information computed near depth discontinuities, where it is inaccurate, but on motion computed over an entire image. The effect of motion parallax is obtained at all scene points that are not located on the *extension* of the 3D surface which corresponds to the registered image region. This gives *dense* parallax data, as these scene points need *not* be adjacent to the registered 3D surface.

The advantage of this technique is in its simplicity and in its robustness. No prior detection and matching are assumed, it requires solving only small sets of linear equations, and each computational step is stated as an overdetermined problem which is numerically stable.

2 Ego-Motion from 2D Image Motion

In this section we describe the technique for computing the 3D ego-motion given the 2D parametric motion of a single image region. The method for automatically computing the 2D motion of a single image region is briefly described in Sec. 4

2.1 Basic Model and Notations

Let (X, Y, Z) denote the Cartesian coordinates of a scene point with respect to the camera (see Fig. 1), and let (x, y) denote the corresponding coordinates in the image plane. The image plane is located at the focal length: $Z = f_c$. The perspective projection of a scene point $P = (X, Y, Z)^t$ on the image plane at a point $p = (x, y)^t$ is expressed by:

$$p = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{X}{Z} f_c \\ \frac{Y}{Z} f_c \end{bmatrix} \qquad (1)$$

The camera motion has two components: a translation $T = (T_X, T_Y, T_Z)^t$ and a rotation $\Omega = (\Omega_X, \Omega_Y, \Omega_Z)^t$.

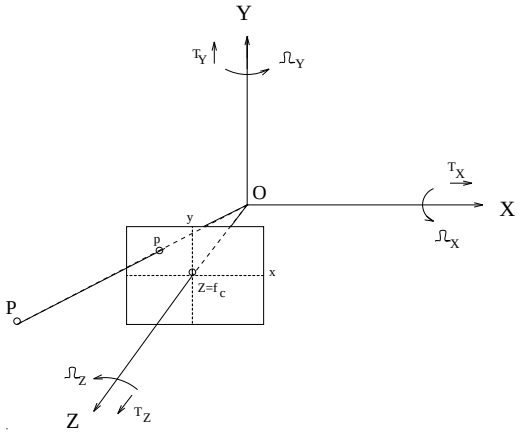

Figure 1: The coordinate system.
The coordinate system (X, Y, Z) is attached to the camera, and the corresponding image coordinates (x, y) on the image plane are located at $Z = f_c$. A point $P = (X, Y, Z)^t$ in the world is projected onto an image point $p = (x, y)^t$. $T = (T_X, T_Y, T_Z)^t$ and $\Omega = (\Omega_X, \Omega_Y, \Omega_Z)^t$ represent the relative translation and rotation of the camera in the scene.

Due to the camera motion the scene point $P = (X, Y, Z)^t$ appears to be moving relative to the camera with rotation $-\Omega$ and translation $-T$, and is therefore observed at new world coordinates $P' = (X', Y', Z')^t$, expressed by:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = M_{-\Omega} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - T \qquad (2)$$

where $M_{-\Omega}$ is the matrix corresponding to a rotation by $-\Omega$.

When the field of view is not very large and the camera motion has a relatively small rotation [1], the 2D displacement (u, v) of an image point (x, y) in the image plane can be expressed by [20, 3]:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f_c(\frac{T_X}{Z} + \Omega_Y) + x\frac{T_Z}{Z} + y\Omega_Z - x^2\frac{\Omega_Y}{f_c} + xy\frac{\Omega_X}{f_c} \\ -f_c(\frac{T_Y}{Z} - \Omega_X) - x\Omega_Z + y\frac{T_Z}{Z} - xy\frac{\Omega_Y}{f_c} + y^2\frac{\Omega_X}{f_c} \end{bmatrix} \qquad (3)$$

The following is noted from Eq. (3):

- Since all translations are divided by the unknown depth Z, only the direction of the translation can be recovered, but not its magnitude.
- The contribution of the camera *rotation* to the displacement of an image point is *independent* of the depth Z of the corresponding scene point.

All points (X, Y, Z) of a planar surface in the 3D scene satisfy a plane equation $Z = A + B \cdot X + C \cdot Y$, which can

be expressed in terms of image coordinates by using Eq. (1) as:

$$\frac{1}{Z} = \alpha + \beta \cdot x + \gamma \cdot y. \quad (4)$$

In a similar manipulation to that in [1], substituting Eq. (4) in Eq. (3) yields:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a + b \cdot x + c \cdot y + g \cdot x^2 + h \cdot xy \\ d + e \cdot x + f \cdot y + g \cdot xy + h \cdot y^2 \end{bmatrix} \quad (5)$$

where:

$$\begin{array}{ll} a = -f_c \alpha T_X - f_c \Omega_Y & e = -\Omega_Z - f_c \beta T_Y \\ b = \alpha T_Z - f_c \beta T_X & f = \alpha T_Z - f_c \gamma T_Y \\ c = \Omega_Z - f_c \gamma T_X & g = -\frac{\Omega_Y}{f_c} + \beta T_Z \\ d = -f_c \alpha T_Y + f_c \Omega_X & h = \frac{\Omega_X}{f_c} + \gamma T_Z \end{array} \quad (6)$$

Eq. (5) describes the 2D parametric motion in the image plane, expressed by eight parameters (a, b, c, d, e, f, g, h), which corresponds to a general 3D motion of a planar surface in the scene, assuming a small field of view and a small rotation. We call Eq. (5) a *pseudo 2D projective* transformation, since under these assumptions, it is a good approximation to the 2D projective transformation.

2.2 General Framework of the Algorithm

In this section we present a scheme which utilizes the robustness of the 2D motion computation for computing 3D motion between two consecutive frames:

1. A single image region is automatically detected, and its 2D *parametric* image motion is computed (Sec. 4).
2. The two frames are registered according to the computed 2D parametric motion of the detected image region. This image region stabilization cancels the rotational component of the camera motion for the *entire* scene (Sec. 2.3), and the camera translation can now be computed from the focus-of-expansion between the two *registered* frames (Sec. 2.4).
3. The 3D rotation of the camera is now computed (Sec. 2.5) given the 2D motion parameters of the detected image region and the 3D translation of the camera.

2.3 Eliminating Camera Rotation

At this stage we assume that a single image region with a parametric 2D image motion has been detected, and that the 2D image motion of that region has been computed. The automatic detection and computation of the 2D image motion for *planar* 3D surfaces is described in Sec. 4.

Let $(u(x,y), v(x,y))$ denote the 2D image motion of the entire scene from frame f_1 to frame f_2, and let $(u_s(x,y), v_s(x,y))$ denote the 2D image motion of a single image region (the *detected* image region) between the two frames. It was mentioned in Sec. 2.1 that (u_s, v_s) can be expressed by a 2D parametric transformation in the image plane if the image region is an image of a *planar* surface in the 3D scene (Eq. (5)). Let s denote the 3D surface of the detected image region, with depths $Z_s(x,y)$. Note that only the 2D motion parameters $(u_s(x,y), v_s(x,y))$ of the planar surface are known. The 3D position or motion parameters of the planar surface are still unknown. Let f_1^R denote the frame obtained by warping the entire frame f_1 towards frame f_2 according to the 2D parametric transformation (u_s, v_s) *extended* to the entire frame. This warping will cause the image region of the detected planar surface, as well as scene parts which are coplanar with it, to be stationary between f_1^R and f_2. In the warping process, each pixel (x,y) in f_1 is displaced by $(u_s(x,y), v_s(x,y))$ to form f_1^R. 3D points that are not located on the surface s (i.e., $Z(x,y) \neq Z_s(x,y)$) will *not* be in registration between f_1^R and f_2.

We will now show that the 2D image motion between the registered frames, (f_1^R and f_2) is affected only by the camera translation T.

Let $P_1 = (X_1, Y_1, Z_1)^t$ denote the 3D scene point projected onto $p_1 = (x_1, y_1)^t$ in f_1. According to Eq. (1): $P_1 = (x_1 \frac{Z_1}{f_c}, y_1 \frac{Z_1}{f_c}, Z_1)^t$. Due to the camera motion (Ω,T) from frame f_1 to frame f_2, the point P_1 will be observed in frame f_2 at $p_2 = (x_2, y_2)^t$, which corresponds to the 3D scene point $P_2 = (X_2, Y_2, Z_2)^t$. According to Eq. (2):

$$P_2 = \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = M_{-\Omega} \cdot P_1 - T. \quad (7)$$

The warping of f_1 by (u_s, v_s) to form f_1^R is equivalent to applying the camera motion (Ω,T) to the 3D points as though they are all located on the surface s (i.e., with depths $Z_s(x,y)$). Let P_s denote the 3D point on the surface s which corresponds to the pixel (x,y) with depth $Z_s(x,y)$. Then:

$$P_s = \begin{bmatrix} x_1 \frac{Z_s}{f_c} \\ y_1 \frac{Z_s}{f_c} \\ Z_s \end{bmatrix} = \frac{Z_s}{Z_1} \cdot \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \frac{Z_s}{Z_1} \cdot P_1 \quad . \quad (8)$$

After the image warping, P_s is observed in f_1^R at $p^R = (x^R, y^R)^t$, which corresponds to a 3D scene point P^R.

Therefore, according to Eq. (2) and Eq. (8):

$$P^R = M_{-\Omega} \cdot P_s - T = \frac{Z_s}{Z_1} \cdot M_{-\Omega} \cdot P_1 - T \quad ,$$

and therefore:

$$P_1 = \frac{Z_1}{Z_s} \cdot {M_{-\Omega}}^{-1} \cdot (P^R + T) \quad .$$

Using Eq. (7) we get:

$$\begin{aligned} P_2 &= M_{-\Omega} \cdot P_1 - T \\ &= \frac{Z_1}{Z_s} \cdot M_{-\Omega} \cdot {M_{-\Omega}}^{-1} \cdot (P^R + T) - T \\ &= \frac{Z_1}{Z_s} \cdot P^R + (1 - \frac{Z_1}{Z_s}) \cdot (-T) \quad , \end{aligned}$$

and therefore:

$$P^R = \frac{Z_s}{Z_1} \cdot P_2 + (1 - \frac{Z_s}{Z_1}) \cdot (-T) \quad . \qquad (9)$$

Eq. (9) shows that the 3D motion between P^R and P_2 is not affected by the camera rotation Ω, but only by its translation T. Moreover, it shows that P^R is on the straight line going through P_2 and $-T$. Therefore, the projection of P^R on the image plane (p^R) is on the straight line going through the projection of P_2 on the image plane (i.e., p_2) and the projection of $-T$ on the image plane (which is the FOE). This means that p^R is found on the radial line emerging from the FOE towards p_2. In other words, the motion between the registered frames f_1^R and f_2 (i.e., $p^R - p_2$) is directed towards, or away from, the FOE, and is therefore induced by the camera translation T .

In Fig. 2, the optical flow is displayed before and after registration of two frames according to the computed 2D motion parameters of the image region which corresponds to the wall at the back of the scene. The optical flow is given for display purposes only, and was *not* used in the registration. After registration, the rotational component of the optical flow was canceled for the *entire* scene, and all flow vectors point towards the real FOE (Fig. 2.c). Before registration (Fig. 2.b) the FOE mistakenly appears to be located elsewhere (in the middle of the frame). This is due to the ambiguity caused by the rotation around the Y-axis, which visually appears as a translation along the X-axis. This ambiguity is resolved by the 2D registration.

2.4 Computing Camera Translation

Once the rotation is canceled by the registration of the detected image region, the ambiguity between

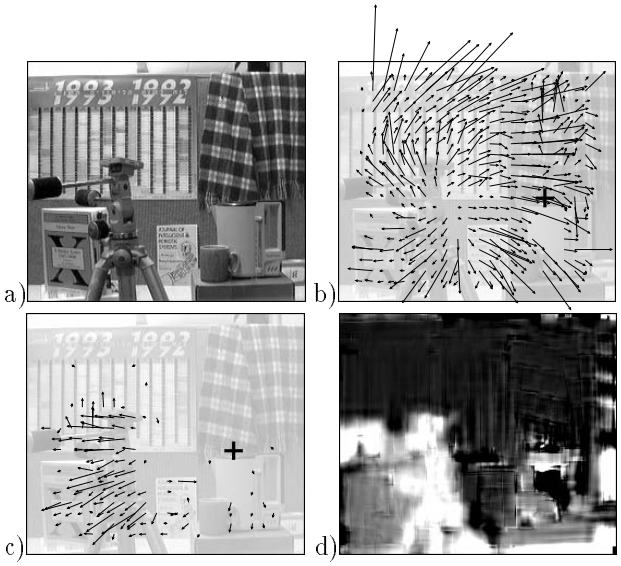

Figure 2: The effect of region registration. The real FOE is marked by +.
a) One of the frames.
b) The optical flow between two adjacent frames (before registration), overlayed on Fig. 2.a.
c) The optical flow after 2D registration of the wall. The flow is induced by pure camera translation (after the camera rotation was canceled), and points now to the correct FOE.
d) The computed depth map. Bright regions correspond to close objects.

image motion caused by 3D rotation and that caused by 3D translation no longer exists. Having only camera translation, the flow field is directed to, or away from, the FOE. The computation of the 3D translation therefore becomes overdetermined and numerically stable, as the only two unknowns indicate the location of the FOE in the image plane.

To locate the FOE, the optical flow between the *registered* frames is computed, and the FOE is located using a search method similar to that described in [18]. Candidates for the FOE are sampled over a half sphere and projected onto the image plane. For each such candidate, a global error measure is computed from local deviations of the flow field from the radial lines emerging from the candidate FOE. The search process is repeated by refining the sampling (on the sphere) around good FOE candidates. After a few refinement iterations, the FOE is taken to be the candidate with the smallest error.

Since the problem of locating the FOE in a *purely translational* flow field is a highly overdetermined problem, the computed flow field need *not* be accurate. This is opposed to most methods which try to compute the ego-motion from the flow field, and require

an *accurate* flow field in order to resolve the rotation-translation ambiguity.

2.5 Computing Camera Rotation

Let (a,b,c,d,e,f,g,h) be the 2D motion parameters of the 3D planar surface corresponding to the detected image region, as expressed by Eq. (5). Given these 2D motion parameters and the 3D translation parameters of the camera (T_X,T_Y,T_Z), the 3D rotation parameters of the camera $(\Omega_X,\Omega_Y,\Omega_Z)$ (as well as the planar surface parameters (α,β,γ)) can be obtained by solving Eq. (6), which is a set of *eight* linear equations in *six unknowns*.

¿From our experience, the parameters g and h in the pseudo 2D projective transformation, computed by the method described in Sec. 4, are not as reliable as the other six parameters (a,b,c,d,e,f), as g and h are second order terms in Eq. (5). Therefore, whenever possible (when the set of Eq. (6) is numerically overdetermined), we avoid using the last two equations (for g and h), and use only the first six. This yields more accurate results.

As a matter of fact, the only case in which all eight equations of (6) must be used to recover the camera rotation is the case when the camera translation is parallel to the image plane (i.e., $\vec{T} \neq 0$ and $T_Z = 0$). This is the only configuration of camera motion in which the first six equations of (6) do not suffice for retrieving the rotation parameters. However, if only the first six equations of (6) are used (i.e., using only the reliable parameters a,b,c,d,e,f, and disregarding the unreliable ones, g and h), then only Ω_Z can be recovered in this case. In order to recover the two other rotation parameters, Ω_X and Ω_Y, the second order terms g and h must be used. This means that for the case of an existing translation with $T_Z = 0$, only the translation parameters (T_X,T_Y,T_Z) and one rotation parameter, Ω_Z (the rotation around the optical axis), can be recovered accurately. The other two rotation parameters, Ω_X and Ω_Y, can only be approximated.

In all other configurations of camera motion the camera rotation can be reliably recovered.

2.6 Experimental Results

The camera motion (In cm) between the two frames in Fig. 2 was: $(T_X,T_Y,T_Z) = (1.7, 0.4, 12)$ and $(\Omega_X,\Omega_Y,\Omega_Z) = (0°, -1.8°, -3°)$. The computation of the 3D motion parameters of the camera (after setting $T_Z = 12$) yielded: $(T_X,T_Y,T_Z) = (1.68, 0.16, 12)$ and $(\Omega_X,\Omega_Y,\Omega_Z) = (-0.05°, -1.7°, -3.25°)$.

Once the 3D motion parameters of the camera are computed, the 3D scene structure can be reconstructed using a scheme similar to that suggested in [11]. Correspondences between small image patches (currently 5×5 pixels) are computed only along the radial lines emerging from the FOE (taking the rotations into account). The depth map is computed from the magnitude of these displacements. In Fig. 2.d, the computed inverse depth map of the scene $(\frac{1}{Z(x,y)})$ is displayed.

3 Camera Stabilization

Once the ego-motion of the camera is determined, this information can be used for post-imaging stabilization of the sequence, as if the camera has been mounted on a gyroscopic stabilizer.

For example, to make perfect stabilization, the images can be warped back to the original position of the first image to cancel the computed 3D rotations. Since rotation is depth-independent, such image warping is easy to perform, resulting in a new sequence which contains only 3D translations, and looks as if taken from a stabilized platform. An example of such stabilization is shown in Fig. 3.d . Alternatively, the rotations can be filtered by a low-pass filter so that the resulting sequence will appear to have only smooth rotations, but no jitter.

4 Computing 2D Motion of a Planar Surface

We use previously developed methods [15, 16] in order to detect an image region corresponding to a planar surface in the scene with its pseudo 2D projective transformation. These methods treated dynamic scenes, in which there were assumed to be multiple moving *planar* objects. The image plane was segmented into the differently moving objects, and their 2D image motion parameters were computed.

In this work we use the 2D detection algorithm in order to detect a single planar surface and its 2D image motion parameters. Due to camera translation, planes at different depths or orientations will have different 2D motions in the image plane, and will therefore be identified as differently moving planar objects. When the scene is not piecewise planar, but contains planar surfaces, the 2D detection algorithm still detects the image motion of its planar regions.

In this section we describe very briefly how the technique for detecting multiple moving planar objects locks onto the one planar object and its 2D motion parameters. More details appear in [15, 16].

The projected 2D image motion $(u(x,y),v(x,y)$ of a planar moving object in the scene can be approximated by the 2D parametric transformation of Eq. (5). If the support R of this planar object were known in

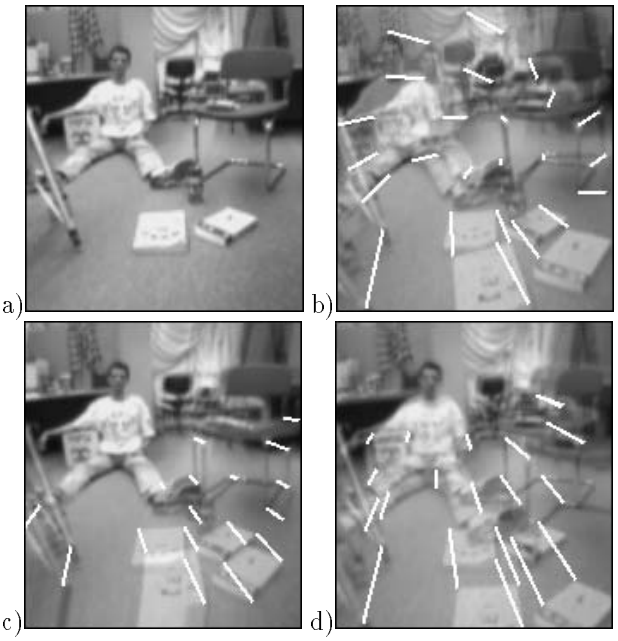

Figure 3: Camera Stabilization.
a) One of the frames in the sequence.
b) The average of two frames, having both rotation and translation. The white lines display the image motion.
c) The average of the two frames after registration of the shirt. Only effects of camera translation remain.
d) The average of the two frames after recovering the ego-motion, and canceling the camera rotation. This results in a stabilized pair of images.

the image plane, then it would be simple to estimate its 2D parametric image motion (u, v) between two successive frames, $I(x, y, t)$ and $I(x, y, t+1)$. This could be done by computing the eight parameters (a, b, c, d, e, f, g, h) of the transformation (u, v) (see Eq. (5)) which minimize the following error function over the region of support R [16]:

$$Err^{(t)}(a, b, c, d, e, f, g, h) = \sum_{(x,y)\in R} (uI_x + vI_y + I_t)^2. \tag{10}$$

The error minimization is performed iteratively using a Gaussian pyramid [4, 15, 16].

Unfortunately, the region of support R of a planar object is not known in advance. Applying the error minimization technique to the *entire* image would usually yield a meaningless result.

This, however, is not true for simple 2D *translations*, where the 2D motion can be expressed by $(u(x, y), v(x, y)) = (a, d)$. It was shown in [5] that the motion parameters of a single *translating* image region can be recovered accurately by minimizing the error function $Err^{(t)}(a, d) = \sum_{(x,y)}(uI_x + vI_y + I_t)^2$ with respect to a and d over the *entire* image (again, using iterations on a multiresolution data structure). This can be done even in the presence of other moving objects in the region of analysis, and with no prior knowledge of their regions of support. This object is called the *dominant translating object*, and its 2D translation the *dominant 2D translation*.

In [15, 16] this method was extended to compute *higher order* 2D motions (2D affine, 2D projective) of a single planar object among differently moving objects. A segmentation step, which marks the region corresponding to the computed dominant 2D motion, was added. This is the region of the dominant planar object in the image.

The scheme for locking onto a single planar object and its 2D image motion is gradual, where the complexity of the 2D motion model is increased in each computation step, and the segmentation of the planar object is refined accordingly. More details can be found in [16] The 2D motion models used in the gradual locking on a planar object are listed below in increasing complexity:

1. **Translation:** 2 parameters, $u(x, y) = a$, $v(x, y) = d$. This model is applied to the *entire* image to get an initial motion estimation. This computation is followed by segmentation to obtain a rough estimate of the object's location.
2. **Affine**: 6 parameters, $u(x, y) = a + bx + cy$, $v(x, y) = d + ex + fy$. This model is applied only to the segmented region obtained in the translation computation step, to get an affine approximation of the object's motion. The previous segmentation is refined accordingly.
3. **A Moving planar surface** (a pseudo 2D projective transformation): 8 parameters [1, 3] (see Eq. (5)), $u(x, y) = a + bx + cy + gx^2 + hxy$, $v(x, y) = d + ex + fy + gxy + hy^2$. This model is applied to the previously segmented region to further refine the 2D motion estimation of the planar object, and its segmentation.

5 Concluding Remarks

A method is for computing ego-motion in static scenes was introduced. At first, an image region corresponding to a planar surface in the scene is detected, and its 2D motion parameters between successive frames are computed. The 2D transformation is then used for image warping, which cancels the rotational component of the 3D camera motion for the *entire* scene, and reduces the problem to pure 3D translation. The 3D translation (the FOE) is computed

from the registered frames, and then the 3D rotation is computed by solving a small set of linear equations.

It was shown that the ego-motion can be recovered reliably in all cases, except for two: The case of an entirely planar scene, and the case of an ego-motion with a translation in the x-y plane only. The first case cannot be uniquely resolved by humans either, due to a visual ambiguity. In the second case it was shown that only the translation parameters of the camera and the rotation around its optical axis can be recovered accurately. The panning parameters (rotation around the x and y axes) can only be roughly estimated in this special case. In all other configurations of camera motion the ego-motion can be reliably recovered.

The advantage of the presented technique is in its simplicity, and in the robustness and stability of each computational step. The choice of an initial 2D motion model enables efficient motion computation and numerical stability. There are no severe restrictions on the ego-motion or on the structure of the environment. Most steps use only image intensities, and the optical flow is used only for extracting the FOE in the case of pure 3D translation, which does not require accurate optical flow. The inherent problems of optical flow and of feature matching are therefore avoided.

Acknowledgment

The authors wish to thank R. Kumar for pointing out an error in the first version of this paper.

References

[1] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Trans. on PAMI*, 7(4):384–401, July 1985.

[2] G. Adiv. Inherent ambiguities in recovering 3D motion and structure from a noisy flow field. *IEEE Trans. on PAMI*, 11:477–489, May 1989.

[3] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *ECCV*, pages 237–252, 1992.

[4] J.R. Bergen, P.J. Burt, R. Hingorani, and S. Peleg. A three-frame algorithm for estimating two-component image motion. *IEEE Trans. on PAMI*, 14:886–895, September 1992.

[5] P.J. Burt, R. Hingorani, and R.J. Kolczynski. Mechanisms for isolating component patterns in the sequential analysis of multiple motion. In *IEEE Workshop on Visual Motion*, pages 187–193, 1991.

[6] S. Carlsson and J.O. Eklundh. Object detection using model based prediction and motion parallax. In *ECCV*, pages 297–306, April 1990.

[7] R. Chipolla, Y. Okamoto, and Y. Kuno. Robust structure from motion using motion paralax. In *ICCV*, pages 374–382, Berlin, May 1993.

[8] K. Daniilidis and H.-H. Nagel. The coupling of rotation and translation in motion estimation of planar surfaces. In *CVPR*, pages 188–193, June 1993.

[9] W. Enkelmann. Obstacle detection by evaluation of optical flow fields from image sequences. In O. Faugeras, editor, *ECCV*, pages 134–138, 1990.

[10] O.D. Faugeras, F. Lustman, and G. Toscani. Motion and structure from motion from point and line matching. In *ICCV*, pages 25–34, 1987.

[11] K. Hanna. Direct multi-resolution estimation of ego-motion and structure from motion. In *IEEE Workshop on Visual Motion*, pages 156–162, Princeton, Oct. 1991.

[12] D.J. Heeger and A. Jepson. Simple method for computing 3d motion and depth. In *ICCV*, pages 96–100, 1990.

[13] B.K.P. Horn. Relative orientation. *Int. J. of Computer Vision*, 4(1):58–78, June 1990.

[14] B.K.P. Horn and E.J. Weldon. Direct methods for recovering motion. *Int. J. of Computer Vision*, 2(1):51–76, June 1988.

[15] M. Irani, B. Rousso, and S. Peleg. Detecting and tracking multiple moving objects using temporal integration. In *ECCV*, pages 282–287, 1992.

[16] M. Irani, B. Rousso, and S. Peleg. Computing occluding and transparent motions. *Int. J. of Computer Vision*, 12(1):5–16, January 1994.

[17] J. Lawn and R. Chipolla. Epipole estimation using affine motion parallax. Technical Report CUED/F-INFENG/TR-138, Cambridge, July 1993.

[18] D.T. Lawton and J.H. Rieger. The use of difference fields in processing sensor motion. In *ARPA IU Workshop*, pages 78–83, June 1983.

[19] C.H. Lee. Structure and motion from two perspective views via planar patch. In *ICCV*, pages 158–164, 1988.

[20] H.C. Longuet-Higgins. Visual ambiguity of a moving plane. *Proceedings of The Royal Society of London B*, 223:165–175, 1984.

[21] F. Meyer and P. Bouthemy. Estimation of time-to-collision maps from first order motion models and normal flows. In *ICPR*, pages 78–82, 1992.

[22] S. Negahdaripour and S. Lee. Motion recovery from image sequences using first-order optical flow information. In *IEEE Workshop on Visual Motion*, pages 132–139, Princeton, NJ, October 1991.

[23] A. Shashua. Projective depth: a geometric invariant for 3d reconstruction from two perspective/orthographic views and for visual recognition. In *ICCV*, pages 583–590, Berlin, May 1993.