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
Email: {michalb, peleg}@cs.huji.ac.il

August 1993

Abstract

A robust method is introduced for computing the camera motion (the *ego-motion*) in a static scene. The method is based on detecting a single image region and computing its 2D motion in the image plane directly from image intensities. The detected 2D 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 in such registered frames is affected only by the 3D translation of the camera. Canceling the effects of the 3D rotation of the camera by using such 2D image registration simplifies the computation of the translation. The 3D translation of the camera is computed by finding the focus-of-expansion in the translation-only set of registered frames. This step is followed by computing the 3D 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.

*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.

1 Introduction

The motion observed in an image sequence can be caused by camera motion (ego-motion) and by motions of independently moving objects 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. Such constraints include the introduction of a regularization term [HS81], assuming a limited model of the world [Adi85], restricting the range of possible motions [HW88], or assuming some type of temporal motion constancy over a longer sequence [DF90].

3D motion is often estimated from the optical or normal flow field derived between two frames [Adi85, LR83, HJ90, GU91, JH91, Sun91, NL91, HA91, TS91, AD92, DRD93], or from the correspondence of distinguished features (points, lines, contours) previously extracted from successive frames [OFT87, Hor90, COK93]. Both approaches depend on the accuracy of the pre-processing stage, which can not always be assured [WHA89].

Increasing the temporal region to more than two frames improves the accuracy of the computed motion. Methods for estimating local image velocities with large temporal regions have been introduced using a combined spatio-temporal analysis [FJ90, Hee88, SM90]. These methods assume motion constancy in the temporal regions, i.e. motion should remain uniform in the analyzed sequence. In feature based methods, features are tracked over a larger time interval using recursive temporal filtering [DF90]. The issue of extracting good features and overcoming occlusions still remains.

Methods for computing the ego-motion *directly* from image intensities were also suggested [Han91, HW88, Taa92]. Horn and Weldon [HW88] deal only with restricted cases (pure translation, pure rotation, or a general motion with known depth). Hanna [Han91] suggests an iterative method for the most general case, but relies on a reasonably accurate initial guess. Taalebinezhaad [Taa92] has a complete mathematical formulation, but relies on computations at a single image point, which can be highly unstable.

3D rotations and translations induce similar 2D image motions [Adi89, DN93, DMRS88] which cause ambiguities in their interpretation, especially in the presence of noise. At depth discontinuities, however, it is much easier to distinguish between the effects of 3D rotations and translations, as the 2D motion of neighboring pixels at different depths will have very similar rotational components, but very different translational components. Based on this observation [LHP80], methods using motion parallax were constructed to obtain the 3D motion of the camera [LR83, LHP80, Lee88, LC93, COK93]. In [LR83, LHP80] difference vectors are computed from the optical flow to cancel the effects of rotations, and the intersection of these difference vectors is computed to yield the focus of expansion (the FOE, which is the intersection point of the direction of the camera translation with the image plane). The optical flow, however, is most inaccurate near depth discontinuities, which is where motion parallax occurs. These methods therefore suffer from these inaccuracies. [Lee88] shows that given four coplanar points in an object, and two additional points outside that plane, structure and motion of that object can be recovered from two perspective views using the motion parallax at the two additional points. This work relies on prior point extraction and *accurate* correspondence because the data is very sparse. [LC93, COK93] locally compute a 2-D affine motion from three non-colinear points, and use a fourth point which is non-coplanar with the other three points to estimate its local motion parallax. They too require prior point extraction and correspondence

In this work a method for computing the ego-motion directly from image intensities is introduced. It is a coarse-to-fine technique in the sense that at first only a robust 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 [BBH$^+$91, IRP92, IRP93] 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 [LH84]. 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 [MB92], but the use of motion derivatives might 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 [WHA89, Adi89].

To overcome the difficulties and ambiguities in the computation of the ego-motion, we introduce the following 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 3D 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 [LR83, LHP80, Lee88, LC93, COK93], 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 *region*. 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.

There are no severe restrictions on the camera motion or the structure of the environment, except for the requirement that there be a big enough image region with a single 2D parametric motion. An imaged planar surface in the 3D scene produces the required 2D image region (but the scene needs *not* be piecewise linear, as in [Adi85]). Most indoor scenes have a planar surface (e.g., walls, floor, pictures, windows, etc.), and in outdoor scenes the ground or any distant object can serve as a planar surface.

In Section 2 the technique for computing the ego-motion is described, given an image region with its 2D parametric image motion. In Section 3 an application of the ego-motion computation to image stabilization is described. In Section 4 the method for detecting and computing the 2D motion parameters of an image region which corresponds to a 3D planar surface in the scene is described.

2 Ego-Motion from 2D Image Motion

Direct estimation of 3D camera motion from a 2D displacement field can be difficult and ill-conditioned, since the 2D image motion depends on a very large number of variables – the 3D motion parameters of the camera, plus the depth at each scene point. 2D motion estimation of a single image region, on the other hand, is numerically stable, since it has only a small number of parameters (see Section 4.2), and is therefore highly overdetermined when computed from all pixels in the region.

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

2.1 Basic Model and Notations

The notations describing the 3D motion of the camera and the corresponding 2D motion of a planar surface in the image plane are introduced in this section. Let (X, Y, Z) denote the Cartesian coordinates of a scene point with respect to the camera (see Figure 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:

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

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

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

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

When the field of view is not very large and the rotation is relatively small [Adi85], a 3D motion of the camera between two image frames creates a 2D displacement (u, v) of an image point (x, y) in the image plane, which can be expressed by [LH84, BAHH92]:

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

It is important to note the following points from Equation (3):

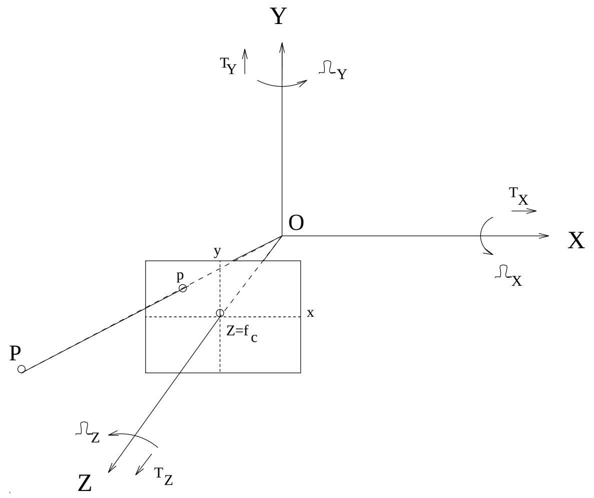

Figure 1: The coordinate system.

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.

- There is a degree of freedom in the magnitude of the translation and depth, as the ratios $\frac{T_X}{Z}$, $\frac{T_Y}{Z}$, and $\frac{T_Z}{Z}$ remain unchanged if both (T_X, T_Y, T_Z) and Z are multiplied by the same scaling factor. Therefore, 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, while the contribution of the camera *translation* to the displacement of an image point *does* depend on the depth Z of the scene point.

All points (X, Y, Z) of a planar surface in the scene satisfy the equation:

$$Z = A + B \cdot X + C \cdot Y$$

for some A, B, and C. Dividing by $A \cdot Z$ and rearranging the terms, we get:

$$\frac{1}{Z} = \frac{1}{A} - \frac{B}{A}\frac{X}{Z} - \frac{C}{A}\frac{Y}{Z}$$

which can be rewritten using Equation (1) as:

$$\frac{1}{Z} = \alpha + \beta \cdot x + \gamma \cdot y \tag{4}$$

where: $\alpha = \frac{1}{A}$, $\beta = -\frac{B}{f_c A}$, $\gamma = -\frac{C}{f_c A}$.

In a similar manipulation to that in [Adi85], substituting (4) in (3) yields:

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

where:

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

Equation (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. Actually, Equation (5) is a good approximation to the 2D projective transformation of that image segment between two frames, as long as the change in angle between the planar surface and the camera in this time interval was negligible.

We call this 2D transformation (Equation (5)) a *pseudo 2D projective* transformation. A method for computing the parameters of the pseudo projective transformation of an image segment is described in Sections 4.2 and 4.3.

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 detected, and its 2D *parametric* image motion is computed (Section 4). The current implementation assumes an image region of a planar surface in the 3D scene, so that its 2D image motion is expressed by a 2D parametric transformation *in the image plane* having eight parameters (The *pseudo 2D projective* transformation, Equation (5)).

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 (Section 2.3), and the translation of the camera can now be computed from the focus-of-expansion between the two *registered* frames (Section 2.4).

3. The 3D rotation of the camera is now computed (Section 2.5) given the 2D motion parameters of the detected image region and the 3D translation of the camera.

2.3 Eliminating Effects of 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 computation of the 2D image motion is described in Section 4 for *planar* 3D surfaces.

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 Section 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 (Equation (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 shape or motion parameters of the planar surface are still unknown. Let f_1^{Reg} 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^{Reg} 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^{Reg}. 3D points that are not located on the 3D surface s (i.e., $Z(x,y) \neq Z_s(x,y)$) will *not* be in registration between f_1^{Reg} and f_2.

We will now show that the residual 2D motion between the registered frames (f_1^{Reg} 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 Equation (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 Equation (2):

$$P_2 = \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = M_{-\Omega} \cdot P_1 - T \tag{7}$$

The warping of f_1 by (u_s, v_s) to form f_1^{Reg} is equivalent to applying the camera motion (Ω, T) to the 3D points as though they are all located on the detected 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:

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

After the image warping, P_s is observed in f_1^{Reg} at $p^{Reg} = (x^{Reg}, y^{Reg})^t$, which corresponds to a 3D scene point P^{Reg}. Therefore, according to Equations (2) and (8):

$$\begin{aligned} P^{Reg} &= M_{-\Omega} \cdot P_s - T \\ &= \frac{Z_s}{Z_1} \cdot M_{-\Omega} \cdot P_1 - T \quad , \end{aligned}$$

and therefore:

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

Using Equation (7) we get:

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

and therefore:

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

Equation (9) shows that the 3D motion between P^{Reg} and P_2 is not affected by the camera rotation Ω, but only by its translation T. Moreover, it shows that P^{Reg} is on the straight line going through P_2 and $-T$. Therefore, the projection of P^{Reg} on the image plane (p^{Reg}) 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^{Reg} is found on the radial line emerging from the FOE towards p_2, or in other words, that the residual 2D motion from f_1^{Reg} to f_2 (i.e., $p^{Reg} - p_2$) at each pixel is directed to, or away from, the FOE, and is therefore induced by the camera translation T .

In Figure 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 FOE (Figure 2.d). Before registration (Figure 2.c) 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 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 [LR83]. Candidates for the FOE are sampled over a half sphere and projected onto the image plane. For each such candidate, an error measure is computed.

Let (α, β) be a candidate for the FOE, and let $(u^{Reg}(x,y), v^{Reg}(x,y))$ denote the optical flow vector at pixel (x,y) after cancelling the camera rotation by image registration (See Section 2.3). Theoretically, in the case of a pure camera translation, all

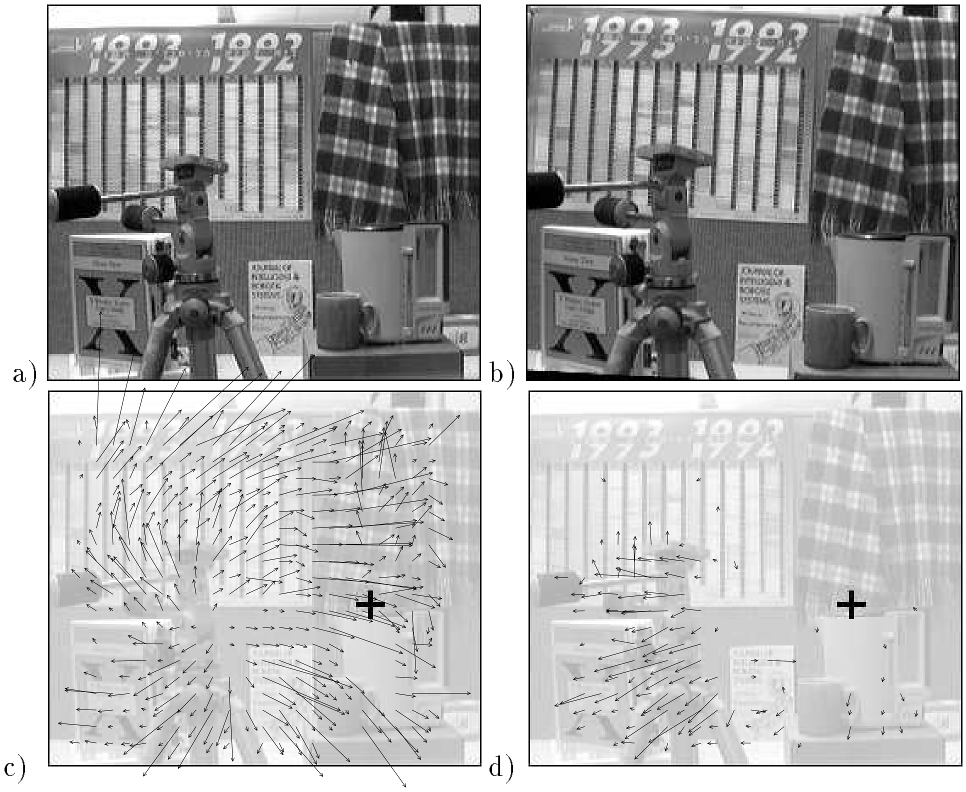

Figure 2: The optical flow before and after registration of the image region corresponding to the the wall. The optical flow is given only for display purposes, and it is *not* used for the registration.
a) The first frame.
b) The second frame, taken after translating the camera by $(T_X, T_Y, T_Z) = (1.7cm, 0.4cm, 12cm)$ and rotating it by $(\Omega_X, \Omega_Y, \Omega_Z) = (0°, -1.8°, -3°)$.
c) The optical flow between Figs. 2.a and 2.b (before registration), overlayed on Figure 2.a . The FOE mistakenly appears to be in the wrong location (in the middle of the frame). This is due to the ambiguity caused by the camera rotation around the Y-axis, which visually appears as a translation along the X-axis. This ambiguity is resolved by the 2D registration.
d) The optical flow after 2D registration of the wall. The flow is induced by pure camera translation (after the camera rotation was cancelled), and points now to the correct FOE (marked by +).

the flow vectors point to, or away from, the FOE. The error measure computed for the candidate FOE (α, β) is therefore defined by:

$$Err(\alpha,\beta) = \frac{\sum_{(x,y)} w(x,y)\cdot((u^{Reg},v^{Reg})_{\perp})^2}{\sum_{(x,y)} w(x,y)} = \frac{\sum_{(x,y)} w(x,y)\cdot \frac{(u^{Reg}\cdot(y-\beta)-v^{Reg}\cdot(x-\alpha))^2}{(x-\alpha)^2+(y-\beta)^2}}{\sum_{(x,y)} w(x,y)} \tag{10}$$

where $(u^{Reg}, v^{Reg})_{\perp} = (u^{Reg}(x,y), v^{Reg}(x,y))_{\perp}$ denotes the flow component *perpendicular* to the radial line emerging from the candidate FOE. $w(x,y)$ are weights determined by the reliability of the optical flow at (x,y):

$$w(x,y) = cond\natural(x,y)\cdot |f_2(x+u^{Reg}, y+v^{Reg}) - f_1^{Reg}(x,y)| \tag{11}$$

where $cond\natural(x,y)$ is the a-priori reliability of the optical flow at (x,y), determined by the condition number of the two well-known optical flow equations [LK81, BAHH92], and $|f_2(x+u^{Reg}, y+v^{Reg}) - f_1^{Reg}(x,y)|$ is the posteriori reliability of the computed optical flow at (x,y), determined by the intensity difference between the source location of (x,y) in image f_1^{Reg} and its computed destination in image f_2.

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. (In [LR83] the error is taken to be the sum of the magnitude of the error *angles*, which we found to give less accurate results).

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.

Difficulties in computing the FOE occur when the flow vectors (u^{Reg}, v^{Reg}) between the two registered frames (after the cancellation of the rotation) are very short. The computation of the FOE therefore becomes sensitive to noise. Observation of Equation (9) shows that this occurs only in the following two cases:

1. $\frac{Z_s(x,y)}{Z_1(x,y)} \approx 1$ for all (x,y), i.e., there are hardly any depth differences between scene points and the detected surface s, or in other words, in the current implementation (where s is a planar surface), the entire scene is approximately planar. This case, however, cannot be resolved by humans either, due to a visual ambiguity [LH84]. This case can be identified by noticing that the image registration according to the 2D motion parameters of the detected planar surface brings the *entire* image into registration.

2. $T = 0$, i.e., the motion of the camera is purely rotational. Once again, as in the previous case, the 2D detection algorithm described in Section 4 will identify the *entire* image as a projected region of a *single* planar surface (even if it is not planar). In this case, however, the ego-motion *can* be recovered. Setting $T_X = T_Y = T_Z = 0$ in Equation (6) yields the following equations for the motion

parameters of the planar surface.

$$\begin{cases} a = -f_c\Omega_Y \\ b = 0 \\ c = \Omega_Z \\ d = f_c\Omega_X \\ e = -\Omega_Z \\ f = 0 \\ g = -\frac{\Omega_Y}{f_c} \\ h = \frac{\Omega_X}{f_c} \end{cases} \tag{12}$$

From Equation (12) it can be noticed that the case of pure rotation can be detected by examining the computed 2D motion parameters of the tracked region for the case when $b \approx f \approx 0$ and $c \approx -e$. The ego-motion in this case is therefore:

$$T_X = T_Y = T_Z = 0 \tag{13}$$

$$\Omega_X = \frac{d}{f_c}, \quad \Omega_Y = -\frac{a}{f_c}, \quad \Omega_Z = \frac{c-e}{2} \tag{14}$$

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 Equation (5). Given these 2D motion parameters and the 3D translation parameters of the camera (T_X, T_Y, T_Z), then 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 the following set of *eight* linear equations in *six unknowns* from Equation (6):

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} -f_cT_X & 0 & 0 & 0 & -f_c & 0 \\ T_Z & -f_cT_X & 0 & 0 & 0 & 0 \\ 0 & 0 & -f_cT_X & 0 & 0 & 1 \\ -f_cT_Y & 0 & 0 & f_c & 0 & 0 \\ 0 & -f_cT_Y & 0 & 0 & 0 & -1 \\ T_Z & 0 & -f_cT_Y & 0 & 0 & 0 \\ 0 & T_Z & 0 & 0 & -\frac{1}{f_c} & 0 \\ 0 & 0 & T_Z & \frac{1}{f_c} & 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \Omega_X \\ \Omega_Y \\ \Omega_Z \end{bmatrix}. \tag{15}$$

From our experience, the parameters g and h in the pseudo 2D projective transformation, computed by the method described in Section 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 Equation (5). Therefore, whenever possible (when the set of equations (15) is numerically overdetermined), we avoid using the last two equations, 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 (15) 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

a)  b) 

Figure 3: The inverse depth map, computed after the camera motion has been determined.
a) First frame, same as Figure 2.a.
b) The obtained inverse depth map. Bright regions correspond to close objects. Dark regions correspond to distant objects. Boundary effects cause some inaccuracies at the image boundaries.

motion in which the first six equations of (15) do not suffice for retrieving the rotation parameters. However, if only the first six equations of (15) 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 between Figure 2.a and Figure 2.b was: $(T_X, T_Y, T_Z) = (1.7cm, 0.4cm, 12cm)$ and $(\Omega_X, \Omega_Y, \Omega_Z) = (0°, -1.8°, -3°)$. The computation of the 3D motion parameters of the camera (after setting $T_Z = 12cm$) yielded: $(T_X, T_Y, T_Z) = (1.68cm, 0.16cm, 12cm)$ and $(\Omega_X, \Omega_Y, \Omega_Z) = (-0.05°, -1.7°, -3.25°)$.

Once the 3D motion parameters of the camera were computed, the 3D scene structure can be reconstructed using a scheme similar to that suggested in [Han91]. Correspondence between small image patches (currently 5×5 pixels) is computed only on the radial line emerging from the FOE. The depth is computed from the magnitude of these displacements. In Figure 3, 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 Figure 4.a3-4.c3 and in Figure 5. Alternatively, the rotations can be filtered by a low-pass filter so that the resulting sequence will appear to have only smooth rotations.

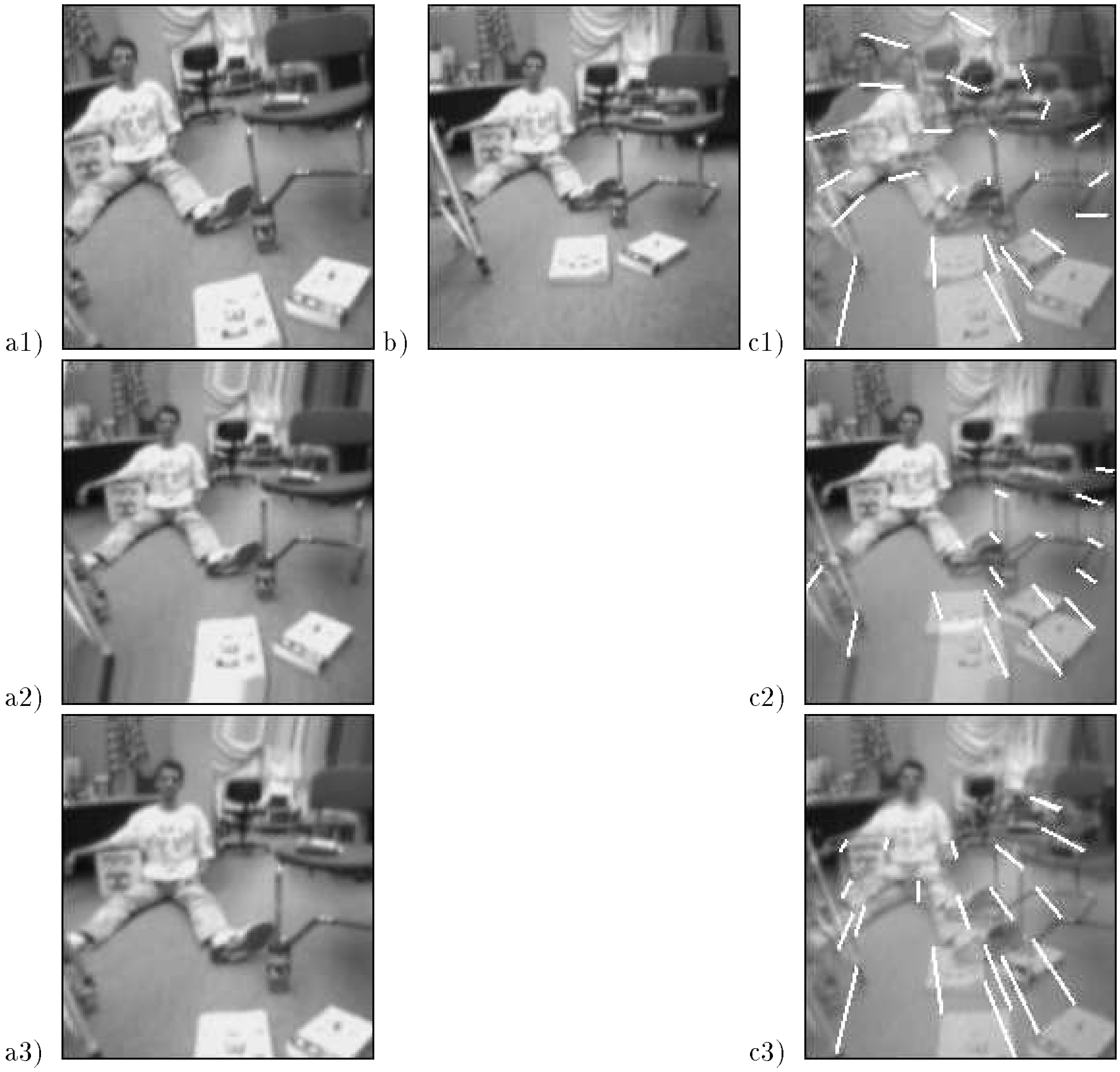

Figure 4: Canceling camera rotation.
a1) The first frame. b) The second frame.
c1) The average image of (a1) and (b), showing the original motion (rotation and translation). The white lines sparsely display the 2D image motion between (a1) and (b).
a2) Frame (a1) warped toward (b) based on the 2D image motion of the detected image region, which is the boy's shirt.
c2) The average image of (a2) and (b), showing the image motion obtained after the 2D warping. This warping brings the boy's shirt into registration. The motion after such a warping is purely translational (the camera translation), but its magnitude at each pixel does not correspond to the real depth of the scene. The camera translation (the FOE) is recovered from such a pair of registered images.
a3) Frame (a1) warped toward (b) based on the computed 3D camera rotation.
c3) The average image of (a3) and (b), showing the image motion obtained when only the camera rotation is cancelled. The motion after such warping is purely translational (the camera translation), this time with magnitude at each pixel corresponding to the *real* depth of the scene. This results in a stabilized pair of images ((a3) and (b)), as if the camera has been mounted on a gyroscopic stabilizer.

a)  b) 

Figure 5: Camera stabilization.
This figure shows an average of three consecutive frames from the examined sequences, before and after processing.
a) Average of the original Sequence. Image motion is highly chaotic.
b) Average of the stabilized sequence. Image motion represents camera translations only. Camera rotations were cancelled (as if the camera has been mounted on a gyroscopic stabilizer).

4 Computing 2D Motion of a Planar Surface

We use previously developed methods [BBH+91, IRP92, IRP93] 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 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.

This section describes very briefly the technique for detecting differently moving planar objects and their 2D motion parameters. For more details refer to [IRP92, IRP93].

4.1 Detecting Multiple Moving Planar Objects

2D motion estimation is difficult when the region of support of an object in the image is not known, which is the common case. It was shown in [BHK91] that the motion parameters of a single *translating* object in the image plane can be recovered accurately by applying the 2D motion computation framework described in Section 4.3 to the *entire* image, using a 2D *translation* motion model (see Section 4.2). This can be done even in the presence of other differently moving objects in the region of analysis, and with no prior knowledge of their regions of support. This object is called the *dominant object*, and its 2D motion the *dominant 2D motion*. Thorough analysis of hierarchical 2D translation estimation is found in [BHK91]. In [IRP92, IRP93] 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 was added to the process, which segments out the region corresponding to the computed dominant 2D motion. This is the region of the dominant planar object in the image.

After detecting and segmenting the dominant planar object between two image frames, the dominant planar object is excluded from the region of analysis, and the detection process is repeated on the remaining parts of the image to find other planar objects and their 2D motions parameters. More details are found in [IRP92].

Temporal integration over longer sequences is used in order to improve the segmentation and the accuracy of the 2D motion parameters [IRP92, IRP93]. A temporally integrated image is obtained by averaging several frames of the sequence after registration with respect to the 2D image motion of the tracked planar surface. After this registration the tracked plane is actually registered, while other parts of the scene that had different image motion are not registered. The temporal average therefore keeps the tracked object sharp, while other image regions blur out. This effect supports continued tracking of the already tracked region.

Figure 6 shows the detection of the first and second dominant planar objects in an image sequence. In this sequence, taken by an infrared camera, the background moves due to camera motion, while the car has another motion. The results are presented after several frames, when the segmentation has improved with temporal integration.

Figure 7 shows the detection and tracking of a single planar surface (the boy's shirt) in an image sequence of a static 3D scene with a moving camera. Despite the fact that all the objects in the scene have the same 3D motion relative to the camera, they create different 2D image motions as they are located at different depths. Figure 7.b displays a temporally integrated image obtained by averaging frames after registration with respect to the 2D image motion of the boy's shirt. The shirt remains sharp, while other regions blur out.

4.2 The 2D Motion Models

We approximate the 2D image motion of projected 3D motions of planar objects by Equation (5). Given two grey level images of an object, $I(x,y,t)$ and $I(x,y,t+1)$, we also use the assumption of grey level constancy:

$$I(x+u(x,y,t), y+v(x,y,t), t+1) = I(x,y,t), \tag{16}$$

where $(u(x,y,t), v(x,y,t))$ is the displacement induced on pixel (x,y) by the motion of the planar object between frames t and $t+1$. Expanding the left hand side of Equation (16) to its first order Taylor expansion around (x,y,t) and neglecting all

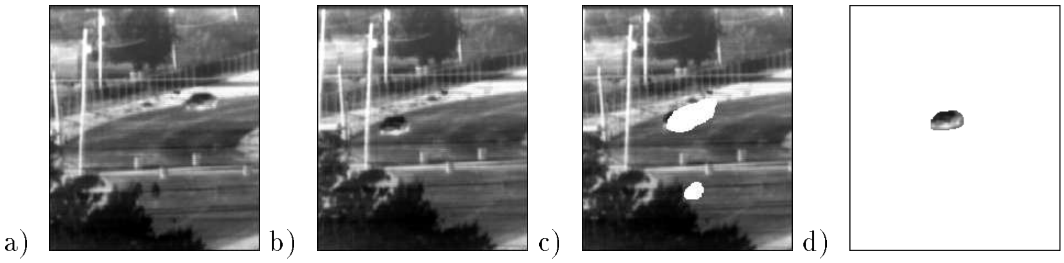

Figure 6: Detecting the first two planar objects in an infrared image sequence with several moving objects. The entire scene moves due to camera rotation, and there are two additional moving objects in the scene: the car and a pedestrian.
a-b) The first and last frames in the sequence. The background and the car move differently.
c) The segmented dominant object (the background) after 5 frames, using a 2D affine motion model. White regions are those *excluded* from the detected region.
d) The segmented second dominant object (the car) after 5 frames, using a 2D affine motion model. White regions are those *excluded* from the detected region.

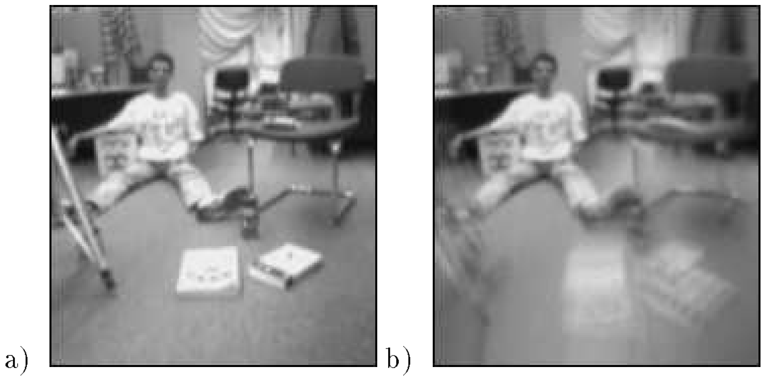

Figure 7: Tracking a planar surface: the camera is moving in a static scene.
a) A single frame from the sequence.
b) A temporally integrated image obtained by averaging several frames after registration with respect to the 2D image motion of the tracked planar surface. The tracked planar surface is the boy's shirt, which is stationary among all the registered frames, and therefore remains sharp. All regions of the scene that deviate from this plane have different 2D image motions and therefore blur out in the average image.

nonlinear terms yields:

$$I(x+u, y+v, t+1) = I(x, y, t) + uI_x + vI_y + I_t, \tag{17}$$

where

$$I_x = \frac{\partial I(x,y,t)}{\partial x}, \quad I_y = \frac{\partial I(x,y,t)}{\partial y}, \quad I_t = \frac{\partial I(x,y,t)}{\partial t}, \quad u = u(x, y, t), \quad v = v(x, y, t).$$

Equations (16) and (17) yield the well-known constraint [HS81]:

$$uI_x + vI_y + I_t = 0. \tag{18}$$

We look for a 2D motion (u, v) which minimize the error function at Frame t in the region of analysis R:

$$Err^{(t)}(u, v) = \sum_{(x,y)\in R} (uI_x + vI_y + I_t)^2. \tag{19}$$

We perform the error minimization over the parameters of one of the following motion models:

1. **Translation:** 2 parameters, $u(x, y, t) = a, \quad v(x, y, t) = d$. In order to minimize $Err^{(t)}(u, v)$, its derivatives with respect to a and d are set to zero. This yields two linear equations in the two unknowns, a and d. Those are the two well-known optical flow equations [LK81, BAHH92], where every small window is assumed to have a single translation in the image plane. In this translation model, the entire *object* is assumed to have a single translation in the image plane.
2. **Affine**: 6 parameters, $u(x, y, t) = a + bx + cy, \quad v(x, y, t) = d + ex + fy$. Deriving $Err^{(t)}(u, v)$ with respect to the motion parameters and setting to zero yields six linear equations in the six unknowns: a, b, c, d, e, f [BBH+91, BAHH92].
3. **A Moving planar surface** (a pseudo 2D projective transformation): 8 parameters [Adi85, BAHH92] (see Section 2.1, Equation (5)), $u(x, y, t) = a + bx + cy + gx^2 + hxy, \quad v(x, y, t) = d + ex + fy + gxy + hy^2$. Deriving $Err^{(t)}(u, v)$ with respect to the motion parameters and setting to zero, yields eight linear equations in the eight unknowns: a, b, c, d, e, f, g, h. Our experience shows that the computed parameters g and h are not as reliable as the other six parameters (a, b, c, d, e, f), as g and h are second order terms.

4.3 The Motion Computation Framework

A multiresolution gradient-based iterative framework [BA87, BBH+91, BBHP90] is used to compute the 2D motion parameters. The basic components of this framework are:

- Construction of a Gaussian pyramid [Ros84], where the images are represented in multiple resolutions.

- Starting at the lowest resolution level:
  1. Motion parameters are estimated by solving the set of linear equations to minimize $Err^{(t)}(u, v)$ (Equation (19)) according to the appropriate 2D motion model (Section 4.2). When the region of support of the planar object is known, minimization is done only over that region. Otherwise, it is performed over the entire image.
  2. The two images are registered by warping according to the computed 2D motion parameters.
     Steps 1 and 2 are iterated at each resolution level for further refinement.
  3. The 2D motion parameters are interpolated to the next resolution level, and are refined by using the higher resolution images.

5 Concluding Remarks

A method is introduced for computing ego-motion in static scenes. 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 a planar scene, and the case of an ego-motion with a translation in the x-y plane only. The first case cannot be 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 of David Sarnoff Research Center for pointing out an error in the first version of this paper.

References

[AD92] Y. Aloimonos and Z. Duric. Active egomotion estimation: A qualitative approach. In *European Conference on Computer Vision*, pages 497–510, Santa Margarita Ligure, May 1992.

[Adi85] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7(4):384–401, July 1985.

[Adi89] G. Adiv. Inherent ambiguities in recovering 3D motion and structure from a noisy flow field. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11:477–489, May 1989.

[BA87] J.R. Bergen and E.H. Adelson. Hierarchical, computationally efficient motion estimation algorithm. *J. Opt. Soc. Am. A.*, 4:35, 1987.

[BAHH92] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *European Conference on Computer Vision*, pages 237–252, Santa Margarita Ligure, May 1992.

[BBH+91] J.R. Bergen, P.J. Burt, K. Hanna, R. Hingorani, P. Jeanne, and S. Peleg. Dynamic multiple-motion computation. In Y.A. Feldman and A. Bruckstein, editors, *Artificial Intelligence and Computer Vision: Proceedings of the Israeli Conference*, pages 147–156. Elsevier, 1991.

[BBHP90] J.R. Bergen, P.J. Burt, R. Hingorani, and S. Peleg. Computing two motions from three frames. In *International Conference on Computer Vision*, pages 27–32, Osaka, Japan, December 1990.

[BHK91] 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, Princeton, New Jersey, October 1991.

[COK93] R. Chipolla, Y. Okamoto, and Y. Kuno. Robust structure from motion using motion paralax. In *International Conference on Computer Vision*, pages 374–382, Berlin, May 1993.

[DF90] R. Deriche and O. Faugeras. Tracking line segments. In *Proc. 1st European Conference on Computer Vision*, pages 259–268, Antibes, April 1990.

[DMRS88] R. Dutta, R. Manmatha, E.M. Riseman, and M.A. Snyder. Issues in extracting motion parameters and depth from approximate translational motion. In *DARPA IU Workshop*, pages 945–960, 1988.

[DN93] K. Daniilidis and H.-H. Nagel. The coupling of rotation and translation in motion estimation of planar surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 188–193, New York, June 1993.

[DRD93] Z. Duric, A. Rosenfeld, and L. Davis. Egomotion analysis based on the Frenet-Serret motion model. In *International Conference on Computer Vision*, pages 703–712, Berlin, May 1993.

[FJ90] D.J. Fleet and A.D. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5(1):77–104, 1990.

[GU91] R. Guissin and S. Ullman. Direct computation of the focus of expansion from velocity field measurements. In *IEEE Workshop on Visual Motion*, pages 146–155, Princeton, NJ, October 1991.

[HA91] L. Huang and Y. Aloimonos. Relative depth from motion using normal flow: An active and purposive solution. In *IEEE Workshop on Visual Motion*, pages 196–204, Princeton, NJ, October 1991.

[Han91] K. Hanna. Direct multi-resolution estimation of ego-motion and structure from motion. In *IEEE Workshop on Visual Motion*, pages 156–162, Princeton, NJ, October 1991.

[Hee88] D.J. Heeger. Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, 1:279–302, 1988.

[HJ90] D.J. Heeger and A. Jepson. Simple method for computing 3d motion and depth. In *International Conference on Computer Vision*, pages 96–100, 1990.

[Hor90] B.K.P. Horn. Relative orientation. *International Journal of Computer Vision*, 4(1):58–78, June 1990.

[HS81] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[HW88] B.K.P. Horn and E.J. Weldon. Direct methods for recovering motion. *International Journal of Computer Vision*, 2(1):51–76, June 1988.

[IRP92] M. Irani, B. Rousso, and S. Peleg. Detecting and tracking multiple moving objects using temporal integration. In *European Conference on Computer Vision*, pages 282–287, Santa Margarita Ligure, May 1992.

[IRP93] M. Irani, B. Rousso, and S. Peleg. Computing occluding and transparent motions. *To appear in International Journal of Computer Vision*, February 1993.

[JH91] A.D. Jepson and D.J. Heeger. A fast subspace algorithm for recovering rigid motion. In *IEEE Workshop on Visual Motion*, pages 124–131, Princeton, NJ, October 1991.

[LC93] J. Lawn and R. Chipolla. Epipole estimation using affine motion parallax. Technical Report CUED/F-INFENG/TR-138, Cambridge, July 1993.

[Lee88] C.H. Lee. Structure and motion from two perspective views via planar patch. In *International Conference on Computer Vision*, pages 158–164, 1988.

[LH84] H.C. Longuet-Higgins. Visual ambiguity of a moving plane. *Proceedings of The Royal Society of London B*, 223:165–175, 1984.

[LHP80] H.C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proceedings of The Royal Society of London B*, 208:385–397, 1980.

[LK81] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Image Understanding Workshop*, pages 121–130, 1981.

[LR83] D.T. Lawton and J.H. Rieger. The use of difference fields in processing sensor motion. In *DARPA IU Workshop*, pages 78–83, June 1983.

[MB92] F. Meyer and P. Bouthemy. Estimation of time-to-collision maps from first order motion models and normal flows. In *International Conference on Pattern Recognition*, pages 78–82, The Hague, 1992.

[NL91] 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.

[OFT87] F. Lustman O.D. Faugeras and G. Toscani. Motion and structure from motion from point and line matching. In *Proc. 1st International Conference on Computer Vision*, pages 25–34, London, 1987.

[Ros84] A. Rosenfeld, editor. *Multiresolution Image Processing and Analysis.* Springer-Verlag, 1984.

[SM90] M. Shizawa and K. Mase. Simultaneous multiple optical flow estimation. In *International Conference on Pattern Recognition*, pages 274–278, Atlantic City, New Jersey, June 1990.

[Sun91] V. Sundareswaran. Egomotion from global flow field data. In *IEEE Workshop on Visual Motion*, pages 140–145, Princeton, NJ, October 1991.

[Taa92] M.A. Taalebinezhaad. Direct recovery of motion and shape in the general case by fixation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14:847–853, August 1992.

[TS91] M. Tistarelli and G. Sandini. Direct estimation of time-to-impact from optical flow. In *IEEE Workshop on Visual Motion*, pages 226–233, Princeton, NJ, October 1991.

[WHA89] J. Weng, T.S. Huang, and N. Ahuja. Motion and structure from two perspective views: Algorithms, error analysis, and error estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(5), 1989.