Robust Recovery of Ego-Motion*

Michal Irani Benny Rousso Shmuel Peleg

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

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 planar surface in the scene directly from image intensities, and computing its 2D motion in the image plane. The detected 2D motion of the planar surface is used to register the images, so that the planar surface appears stationary. The resulting displacement field for the *entire* scene in such registered frames is affected only by the 3D translation of the camera, which is computed by finding the focus-of-expansion in the registered frames. This step is followed by computing the 3D rotation to complete the computation of the ego-motion.
This 3D motion computation is based on a motion computation scheme which handles the difficult case when multiple image motions are present. This multiple motion analysis is performed together with object segmentation by using a temporal integration approach.

1 Introduction

A method for detecting and tracking multiple moving objects, using both a large spatial region and a large temporal region, is described. When the large spatial region of analysis has multiple moving objects, the motion parameters and the locations of the objects are computed for one object after another. The method has been applied successfully to 2D affine and projective motions in the image plane.

The 2D detection and tracking algorithm is used for estimating the camera motion (*ego-motion*) in general static 3D scenes. Once a single planar surface in a general static scene is detected in the image, and its 2D motion parameters computed, we use this data for estimating the entire 3D scene structure and the 3D motion performed by the camera. The registration of an image region which corresponds to a planar surface in the scene, and examining the motion in the registered sequence, helps to overcome the ambiguities in computing 3D motion only from the image motion of a planar surface [2, 19].

Sect. 2 describes briefly a method for detecting and tracking the differently moving objects in the sequence. Sect. 3 describes the method for computing the 3D motion of the camera (the ego-motion) in a static scene. More details can be found in [13, 15, 14, 16].

* This research was supported by the Israel Science Foundation.
M. Irani and B. Rousso were partially supported by the Leibniz Center.

2 Multiple Motions in Image Sequences

To detect differently moving objects in an image pair, a single motion is first computed, and a single object which corresponds to this motion is identified. We call this motion the *dominant motion*, and the corresponding object the *dominant object*. Once a dominant object has been detected, it is excluded from the region of analysis, and the process is repeated on the remaining image regions to find other objects and their motions. Temporal integration is then used to track detected objects throughout the image sequence. More details can be found in [15].

It is assumed that the projected 3D motions of the objects can be approximated by some 2D parametric transformation in the image plane. This assumption is valid when the differences in depth caused by the motions are small relative to the distances of the objects from the camera. We have chosen to use an iterative, multi-resolution, gradient-based approach for motion computation [4, 6, 7]. The parametric motion models used in our current implementation are: pure 2D translation (2 parameters), 2D affine transformation (6 parameters, [6, 5]) and projective transformation (8 parameters [1, 5]).

Detecting the First Object. The motion parameters of a single object in the image plane can be recovered by applying the iterative detection method to the *entire* region of analysis. 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 [8, 14]. Once a motion has been determined, we would like to identify the region having this motion. To simplify the problem, the two images are registered using the detected motion. The motion of the corresponding region is therefore canceled, and the problem becomes that of identifying the stationary regions. Detection of stationary regions is described in [15].

Tracking by Temporal Integration. Once an object has been detected, it can be tracked throughout the image sequence. This is done by using temporal integration of images registered with respect to the tracked motion. The temporally integrated image serves as a dynamic internal representation image of the tracked object.

Let $\{I(t)\}$ denote the image sequence, and let $M(t)$ denote the segmentation mask of the tracked object computed for frame $I(t)$, using the segmentation method described in [15]. Initially, $M(0)$ is the entire region of analysis. The temporally integrated image is denoted by $Av(t)$, and is constructed as follows:

$$\begin{cases} Av(0) & \stackrel{\text{def}}{=} I(0) \\ Av(t+1) & \stackrel{\text{def}}{=} (1-w) \cdot I(t+1) + w \cdot register(Av(t), I(t+1)) \end{cases}$$

where $register(P, Q)$ denotes the registration of images P and Q by warping P towards Q according to the motion of the tracked object computed between them, and $0 < w < 1$ (currently $w = 0.7$). An example of a temporally integrated image is shown in Fig. 1.

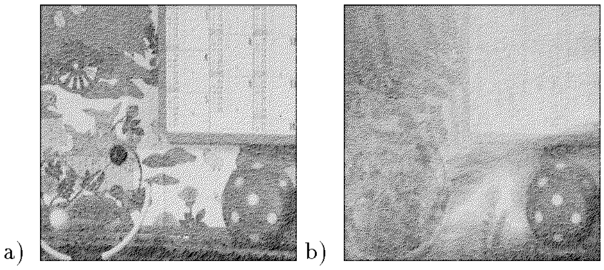


Fig. 1. Temporal integration.
a) A single frame from a sequence which contains four moving objects.
b) The temporally integrated image, where the tracked motion is that of the ball. Other regions blur out.

When the motion model approximates well enough the temporal changes of the tracked object, shape changes relatively slowly over time in registered images. Therefore, temporal integration of registered frames produces a sharp and clean image of the tracked object, while blurring regions having other motions. Fig. 1 shows a temporally integrated image of a tracked rolling ball. Comparing each new frame to the temporally integrated image rather than to the previous frame gives the algorithm a strong bias to keep tracking the same object. Since additive noise is reduced in the the average image of the tracked object, and since image gradients outside the tracked object decrease substantially, both segmentation and motion computation improve significantly.

In the example shown in Fig. 2, temporal integration is used to detect and track the first and second object. In this sequence, taken by an infrared camera, the background moves due to camera motion, while the car moves differently. It is evident that the tracked object in Fig. 2.c is the background, as the background maintains its sharpness, while all other regions in the image are blurred by their motion, and that the tracked object in Fig. 2.e is the car.

3 Ego-Motion in Static Scenes

Direct estimation of 3D motion is a difficult and ill-conditioned problem, due to the very large number of variables – the 3D motion parameters of the camera plus the depth at each point. 2D motion estimation, on the other hand, is a numerically stable problem, because the 2D problem is highly overdetermined (only six unknowns in the affine model, eight unknowns in the projective model).

Previous works on 3D motion estimation use the optical or normal flow field derived between two frames [1, 3, 9, 17, 18, 21, 20], or the correspondence of previously extracted distinguished features (points, lines, contours) [11, 22]. Methods for computing the ego-motion *directly* from image intensities were also suggested [10, 12, 23], but each method has its limitations.

In this section we propose the following scheme in order to use the robustness of the 2D motion computation for computing 3D motion:

1. The 2D image motion of a single planar surface is computed (Sect. 2).

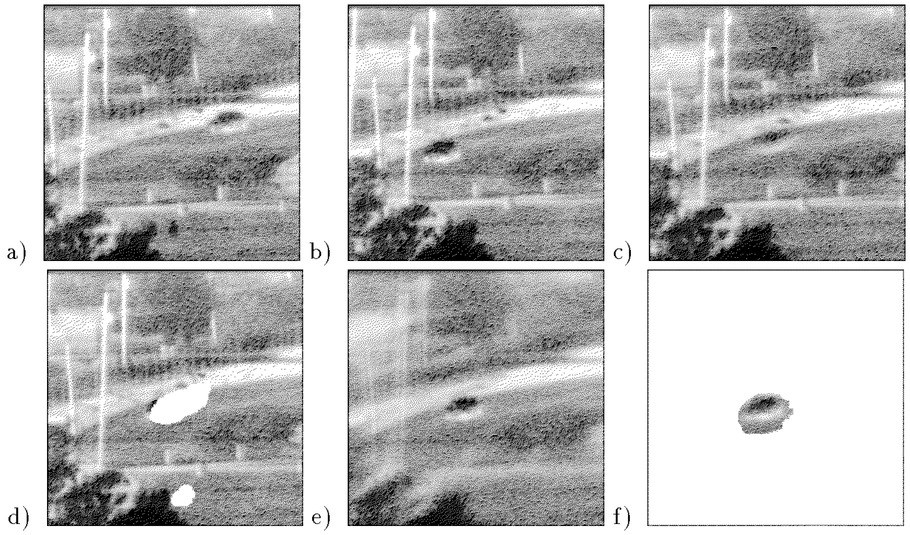


Fig. 2. Detecting and tracking multiple moving objects using temporal integration.
a-b) The first and last frames. Both the background and the car are moving.
c) The temporally integrated image of the first tracked object (background). The car blurs out.
d) Segmentation of the first tracked object (background). White regions are those not belonging to the tracked region.
e) Tracking the second object (the car). The background blurs now.
f) Segmentation of the second tracked object.

2. The two frames are registered according to the computed 2D motion parameters of the detected plane. This cancels the rotational component of the 3D camera motion, and the 3D translation of the camera can be computed from the two registered frames.
3. The 3D rotation of the camera is computed from the previously computed 3D translation of the camera and from the 2D motion parameters of the detected plane.
4. The 3D scene structure can then be reconstructed from the computed 3D motion parameters of the camera (using a scheme similar to that suggested in [10]).

3.1 Projected 2D Motion

When the field of view is not very large and the rotation is relatively small [1], 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 [5]:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f_c(\frac{T_X}{Z} + \Omega_Y) + x\frac{T_Z}{Z} + y\Omega_Z - x^2\frac{\Omega_Y}{f_c} + xy\frac{\Omega_X}{f_c} \\ -f_c(\frac{T_Y}{Z} - \Omega_X) - x\Omega_Z + y\frac{T_Z}{Z} - xy\frac{\Omega_Y}{f_c} + y^2\frac{\Omega_X}{f_c} \end{bmatrix} \tag{1}$$

where, (X, Y, Z) denote the Cartesian coordinates of the scene point projected onto (x, y), (T_X, T_Y, T_Z) and $(\Omega_X, \Omega_Y, \Omega_Z)$ are the 3D motion parameters of the camera (translation and rotation, respectively), and f_c is the focal length of the camera.

3.2 Reducing General Motion to Translation

Let (u, v) denote the 2D displacement field between f_1 and f_2, and let (u_s, v_s) denote the 2D motion parameters of a single 3D plane in the scene. Let f_1^{Reg} denote the frame obtained by warping frame f_1 towards f_2 according to (u_s, v_s). f_1^{Reg} and f_2 will be registered over regions of the projected 3D plane within the image, and unregistered over other image regions. The 2D motion between the registered frames (f_1^{Reg} and f_2) is therefore: $(u^{Reg}, v^{Reg}) = (u - u_s, v - v_s)$. Using Eq. (1) we get:

$$\begin{bmatrix} u^{Reg}(x,y) \\ v^{Reg}(x,y) \end{bmatrix} = \begin{bmatrix} u(x,y) - u_s(x,y) \\ v(x,y) - v_s(x,y) \end{bmatrix} = \begin{bmatrix} -f_c T_X(\frac{1}{Z} - \frac{1}{Z_s}) + xT_Z(\frac{1}{Z} - \frac{1}{Z_s}) \\ -f_c T_Y(\frac{1}{Z} - \frac{1}{Z_s}) + yT_Z(\frac{1}{Z} - \frac{1}{Z_s}) \end{bmatrix} \quad (2)$$

where, $Z_s = Z_s(x, y)$ is the depth function of the 3D plane at pixel (x, y), and $Z = Z(x, y)$ is the real depth at that pixel.

This registration cancels the rotation parameters $(\Omega_X, \Omega_Y, \Omega_Z)$ in Equation (2), leaving the *original* translation parameters (T_X, T_Y, T_Z) between the *registered* images, with *new* scene depths $Z^{Reg}(x, y)$ defined by: $\frac{1}{Z^{Reg}(x,y)} = \frac{1}{Z(x,y)} - \frac{1}{Z_s(x,y)}$. Note that $Z^{Reg}(x, y)$ may also be negative, as opposed to the original scene.

In Fig. 3, the optical flow is displayed before and after registration of two frames according to the computed 2D motion parameters of the wall at the back of the scene. After registration the optical flow points towards the FOE.

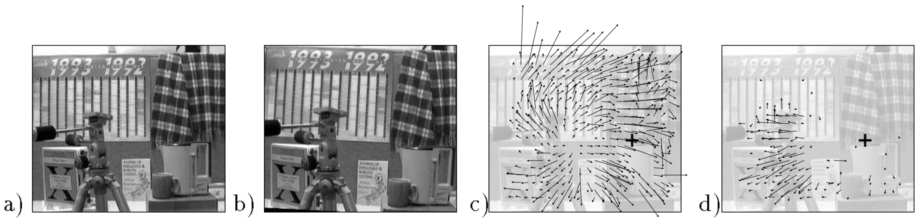


Fig. 3. The optical flow before and after registration of the image region corresponding to the wall. The optical flow is given only for display purposes, and is not used for registration.
a) The first frame.
b) The second frame, taken after translating the camera by $(1.7cm, 0.4cm, 12cm)$ and rotating it by $(0°, -1.8°, -3°)$.
c) Optical flow between Figs. 3.a and 3.b (before registration), overlayed on Fig. 3.a .
d) Optical flow after registration of the wall. It is induced by pure translation, and points to the correct FOE (marked by +).

Once the rotation is cancelled by the registration of the plane, the ambiguity between image motion caused by rotation and that caused by translation no longer exists. When only 3D translation exists, the induced image motion is directed towards the FOE (Focus of Expansion). The computation of the 3D translation therefore becomes a highly overdetermined and a numerically stable problem (as there are only two unknowns to the problem – the location of the FOE in the image plane).

3.3 Computing 3D Rotation

Assuming that the detected parametric surface is planar, (i.e., (X, Y, Z) lies on a planar surface in the 3D scene), it can be described by $Z = A + B \cdot X + C \cdot Y$. By perspective projection, this yields: $\frac{1}{Z} = \alpha + \beta \cdot x + \gamma \cdot y$ where: (x, y) are image coordinates, and $\alpha = \frac{1}{A}, \quad \beta = -\frac{B}{f_c A}, \quad \gamma = -\frac{C}{f_c A}$. Therefore, Eq. (1) can be rewritten as [1, 5]:

$$\begin{bmatrix} u_s \\ v_s \end{bmatrix} = \begin{bmatrix} a + b \cdot x + c \cdot y + g \cdot x^2 + h \cdot xy \\ d + e \cdot x + f \cdot y + g \cdot xy + h \cdot y^2 \end{bmatrix} \tag{3}$$

where:

$$\begin{array}{ll} a = -f_c \alpha T_X - f_c \Omega_Y & e = -\Omega_Z - f_c \beta T_Y \\ b = \alpha T_Z - f_c \beta T_X & f = \alpha T_Z - f_c \gamma T_Y \\ c = \Omega_Z - f_c \gamma T_X & g = -\frac{\Omega_Y}{f_c} + \beta T_Z \\ d = -f_c \alpha T_Y + f_c \Omega_X & h = \frac{\Omega_X}{f_c} + \gamma T_Z \end{array} \tag{4}$$

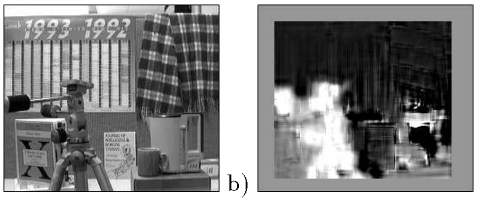
The parameters (a, b, c, d, e, f, g, h) are the 2D motion parameters of the detected 3D plane, computed as described in Sect. 2. Given these 2D motion parameters and the 3D translation parameters of the camera (T_X, T_Y, T_Z), then the 3D rotation parameters of the camera $(\Omega_X, \Omega_Y, \Omega_Z)$ (as well as the surface parameters (α, β, γ)) can be obtained by solving the set (4) of eight linear equations in six unknowns.

Experimental Results. The camera motion between Figure 3.a and Figure 3.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 motion parameters yielded: $(T_X, T_Y, T_Z) = (1.68cm, 0.16cm, 12cm)$ and $(\Omega_X, \Omega_Y, \Omega_Z) = (-0.05°, -1.7°, -3.25°)$. (The translation magnitude cannot be determined, only its direction. T_Z was therefore set to the correct size $12cm$, and the other parameters were then scaled accordingly).

Once the 3D motion parameters of the camera were computed, the 3D scene structure was reconstructed using a scheme similar to that suggested in [10]. In Fig. 4, the computed inverse depth map of the scene $(\frac{1}{Z(x,y)})$ is displayed.

4 Concluding Remarks

A method is introduced for computing ego-motion in static scenes. At first, a planar surface in the scene is detected, and its pseudo 2D projective transformation between successive frames is computed. This plane is detected by temporal



a) b)

Fig. 4. The inverse depth map.
a) First frame.
b) The obtained inverse depth map. Bright regions correspond to close objects. Dark regions correspond to distant objects. The depth was not computed near the the image boundaries.

integration of registered images. The temporal integration proves to be a powerful approach to motion analysis, enabling human-like tracking of moving objects. The tracked object remains sharp while other objects blur out, which enables accurate segmentation and motion computation.

Detection of a single planar surface with its 2D motion parameters is used for computing the 3D motion parameters of a camera (the ego-motion) in a static scene This is done by registering the image sequence using the motion of the detected planar surface. This registration 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.

References

1. 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.
2. 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.
3. 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.
4. J.R. Bergen and E.H. Adelson. Hierarchical, computationally efficient motion estimation algorithm. *J. Opt. Soc. Am. A.*, 4:35, 1987.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. B.K.P. Horn. Relative orientation. *International Journal of Computer Vision*, 4(1):58–78, June 1990.
12. B.K.P. Horn and E.J. Weldon. Direct methods for recovering motion. *International Journal of Computer Vision*, 2(1):51–76, June 1988.
13. M. Irani and S. Peleg. Image sequence enhancement using multiple motions analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, Champaign, June 1992.
14. 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.
15. M. Irani, B. Rousso, and S. Peleg. Computing occluding and transparent motions. *To appear in International Journal of Computer Vision*, 1993.
16. M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using image stabilization. Technical Report 93-11, Institute of Computer Science, The Hebrew University, Jerusalem, Israel, May 1993.
17. 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.
18. D.T. Lawton and J.H. Rieger. The use of difference fields in processing sensor motion. In *DARPA IUWorkshop*, pages 78–83, June 1983.
19. H.C. Longuet-Higgins. Visual ambiguity of a moving plane. *Proceedings of The Royal Society of London B*, 223:165–175, 1984.
20. 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.
21. 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.
22. 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.
23. 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.