3D Motion from Image Derivatives Using the Least Trimmed Square Regression*

Fadi Dornaika and Angel D. Sappa

Computer Vision Center
Edifici O, Campus UAB
08193 Bellaterra, Barcelona, Spain
{dornaika, sappa}@cvc.uab.es

Abstract. This paper presents a new technique to the instantaneous 3D motion estimation. The main contributions are as follows. First, we show that the 3D camera or scene velocity can be retrieved from image derivatives only. Second, we propose a new robust algorithm that simultaneously provides the Least Trimmed Square solution and the percentage of inliers- the non-contaminated data. Experiments on both synthetic and real image sequences demonstrated the effectiveness of the developed method. Those experiments show that the developed robust approach can outperform the classical robust scheme.

1 Introduction

Computing object and camera motions from 2D image sequences has been a central problem in computer vision for many years. More especially, computing the 3D velocity of either the camera or the scene is of particular interest to a wide variety of applications in computer vision and robotics such as calibration, visual servoing, etc. Many algorithms have been proposed for estimating the 3D relative camera motions (discrete case) [1,2] and the 3D velocity (differential case) [3]. While the discrete case requires feature matching and tracking across the images, the differential case requires the computation of the optical flow field (2D velocity field). All these problems are generally ill-conditioned.

This paper has two main contributions. First, we introduce a novel technique to the 3D velocity estimation using image derivatives only, therefore feature extraction and tracking are not required. Second, we propose a robust method that combines the Least Trimmed Square regression and the Golden Section Search algorithm where the number of inliers is not known *a priori*. In our work, we assume that the scene is far from the camera or it contains a dominant planar structure. Using image derivatives has been exploited in [4] to make camera intrinsic calibration. In our study, we deal with the 3D velocity of the camera or the scene. The paper is organized as follows. Section 2 states the problem. Section 3 describes the proposed approach. Experimental results on both synthetic and real image sequences are given in Section 4.

* This work was supported by the MEC project TIN2005-09026 and The Ramón y Cajal Program.

N. Zheng, X. Jiang, and X. Lan (Eds.): IWICPAS 2006, LNCS 4153, pp. 76–84, 2006.

2 Problem Formulation

Throughout this paper we represent the coordinates of a point in the image plane by small letters (x, y) and the object coordinates in the camera coordinate frame by capital letters (X, Y, Z). In our work we use the perspective camera model as our projection model. Thus, the projection is governed by the following equation were the coordinates are expressed in homogeneous form,

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & s & x_c & 0 \\ 0 & r\,f & y_c & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{1}$$

Here, f denotes the focal length in pixels, r and s the aspect ratio and the skew and (x_c, y_c) the principal point. These are called the intrinsic parameters. In this study, we assume that the camera is calibrated, i.e., the intrinsic parameters are known. For the sake of presentation simplicity, we assume that the image coordinates have been corrected for the principal point and the aspect ratio. This means that the camera equation can be written as in (1) with $r = 1$, and $(x_c, y_c) = (0, 0)$. Also, we assume that the skew is zero (s = 0). With these parameters the projection simply becomes

$$x = f\,\frac{X}{Z} \quad \text{and} \quad y = f\,\frac{Y}{Z} \tag{2}$$

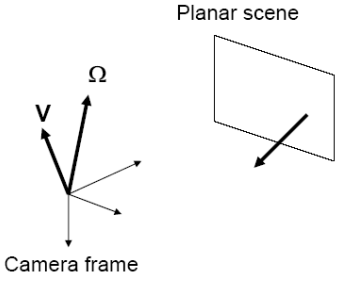

Fig. 1. The goal is to compute the 3D velocity from image derivatives

Let $I(x, y, t)$ be the intensity at pixel (x, y) in the image plane at time t. Let $u(x, y)$ and $v(x, y)$ denote components of the motion field in the x and y directions, respectively. This motion field is caused by the translational and rotational camera velocities $(\mathbf{V}, \boldsymbol{\Omega}) = (V_x, V_y, V_z, \Omega_x, \Omega_y, \Omega_z)$. Using the constraint that the gray-level intensity is locally invariant to the viewing angle and distance we obtain the well-known optical flow constraint equation:

$$I_x\,u + I_y\,v + I_t = 0 \tag{3}$$

where $u = \frac{\partial x}{\partial t}$ and $v = \frac{\partial y}{\partial t}$ denote the motion field. $I_x = \frac{\partial I}{\partial x}$ and $I_y = \frac{\partial I}{\partial y}$ denote the components of the spatial image gradient. They can be computed by convolution with derivatives of a 2D Gaussian kernel. The temporal derivative $I_t = \frac{\partial I}{\partial t}$ can be computed by convolution between the derivative of a 1D Gaussian and the image sequence.

We assume that the perspective camera observes a planar scene[1] described in the camera coordinate system by $Z = \alpha X + \beta Y + \gamma$. One can show that the equations of the motion field are given by these two equations:

$$u(x,y) = a_1 + a_2\, x + a_3\ y + a_7\, x^2 + a_8\, xy \tag{4}$$

$$v(x,y) = a_4 + a_5\, x + a_6\ y + a_7\, xy + a_8\, y^2 \tag{5}$$

where the coefficients are given by:

$$\begin{cases} a_1 = -f\,(\frac{V_x}{\gamma} + \Omega_y) \\ a_2 = (\frac{V_x}{\gamma}\,\alpha + \frac{V_z}{\gamma}) \\ a_3 = \frac{V_x}{\gamma}\,\beta + \Omega_z \\ a_4 = -f\,(\frac{V_y}{\gamma} - \Omega_x) \\ a_5 = (\frac{V_y}{\gamma}\,\alpha - \Omega_z) \\ a_6 = (\frac{V_y}{\gamma}\,\beta + \frac{V_z}{\gamma}) \\ a_7 = \frac{-1}{f}(\frac{V_z}{\gamma}\,\alpha + \Omega_y) \\ a_8 = \frac{-1}{f}(\frac{V_z}{\gamma}\,\beta - \Omega_x) \end{cases} \tag{6}$$

One can notice that the two solutions (V_x, V_y, V_z, γ) and $\lambda\,(V_x, V_y, V_z, \gamma)$ yield the same motion field. This is consistent with the scale ambiguity that occurs in the Structure From Motion problems. The case of a steady camera and a moving planar scene can be obtained by multiplying the right hand side of Eq.(6) by -1. Our goal is to estimate the instantaneous velocity $(\mathbf{V}, \boldsymbol{\Omega})$ as well as the plane orientation from the image derivatives (I_x, I_y, I_t).

In the sequel, we propose a two-step approach. In the first step, the eight coefficients are recovered by solving the system (3) using the Least Trimmed Square (LTS) regression and the Golden Section Search algorithm. In the second step, the 3D velocity as well as the plane orientation are recovered from Eq.(6) using a non-linear technique.

3 Approach

We assume that the image contains N pixels for which the spatio-temporal derivatives (I_x, I_y, I_t) have been computed. In practice, N is very large. In order to reduce this number, one can either drop pixels having small gradient components or adopt a low-resolution representation of the images. In the sequel, we

[1] Our work also addresses the case where the scene contains a dominant planar structure.

do not distinguish between the two cases, i.e., N is either the original size or the reduced one. By inserting Eqs.(4) and (5) into Eq.(3) we get

$$\begin{aligned} &I_x\, a_1 + I_x x\, a_2 + I_x y\, a_3 + I_y\, a_4 + I_y\, x\, a_5 + I_y\, y\, a_6 \\ &+(I_x x^2 + I_y\, x\, y)\, a_7 + (I_x\, x\, y + I_y\, y^2)\, a_8 = -I_t \end{aligned} \tag{7}$$

By concatenating the above equation for all pixels, we get the following over-constrained linear system:

$$\mathbf{G}\,\mathbf{a} = \mathbf{e} \tag{8}$$

where $\mathbf{a}$ denotes the column vector $(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8)^T$.

It is well known that the Maximum Likelihood solution to the above linear system is given by:

$$\mathbf{a} = \mathbf{G}^\dagger\, \mathbf{e} \tag{9}$$

where $\mathbf{G}^\dagger = (\mathbf{G}^T\, \mathbf{G})^{-1}\, \mathbf{G}^T$ is the pseudo-inverse of the $N \times 8$ matrix $\mathbf{G}$. This solution is known as the Least Squares solution (LS). In practice, the system of linear equations may contain outliers. These outliers can be caused by local planar excursions and derivatives errors. Therefore, our idea is to estimate the 8 coefficients using robust statistics [5,6]. Statisticians have developed various kinds of robust estimators such as the Least Median of Squares (LMS) and the RANdom SAmpling Consensus (RANSAC).

3.1 Least Trimmed Square Regression

In this section, we briefly provide the principles of the linear Least Trimmed Square regression. The LTS regression has been proposed by Rousseeuw [6]. Its objective is to compute the unknown parameters (in our case, it is the vector $\mathbf{a}$) by minimizing

$$e = \sum_{i=1}^{h} (r^2)_{i:N} \tag{10}$$

where $(r^2)_{1:N} \leq \cdots (r^2)_{N:N}$ are the ordered squared residuals obtained for the linear system (e.g. (8)) associated with any value for the parameters. This is equivalent to finding the h-subset with the smallest least squares error. The LTS estimate is then the least square solution to this h-subset. The LTS objective function is smoother than that of the LMS. However, the implementation of LTS is less straightforward than the LMS. Notice that h corresponds to the percentage of non-contaminated data, that is, the percentage of inliers. In [7], an efficient implementation of the LTS has been proposed when h is known in advance. The proposed algorithm combines random sampling and an iterative C-step (Condensation step). The basic idea of the C-step is to start from an initial solution and update it iteratively by a Least Square estimator performed on another subset of constraints having the h smallest residuals.

Random sampling: Repeat the following three steps K times

1. Draw a random subsample of p different equations/pixels ($p \geq 8$).
2. For this subsample, indexed by k, compute the eight coefficients, i.e. the vector $\mathbf{a}_k$, from the corresponding p equations using a linear system similar to (8).
3. For this solution $\mathbf{a}_k$, determine the squared residuals with respect to the whole set of N equations. We have N residuals corresponding to the linear system (8). Sort these residuals and compute the trimmed sum $e_k = \sum_{i=1}^{N/2} (r^2)_{i:N}$. Note that this sum can be carried out using another number such as the *a priori* percentage of inliers.

Initial solution: Among the random solutions, keep the best solution, i.e., select the one that provides the smallest error.

Golden section optimization:

1. Select an initial bracketing interval $[\epsilon_a, \epsilon_b]$.
2. Split the bracketing interval into three segments $\epsilon_a, \epsilon_1, \epsilon_2, \epsilon_b$

$$\epsilon_1 = \epsilon_a + w\,(\epsilon_b - \epsilon_a), \text{ and } \epsilon_2 = \epsilon_b - w\,(\epsilon_b - \epsilon_a)$$

   where the fraction $w = (3 - \sqrt{5})/2 = 0.38197$ (see [8]).
3. For each percentage, perform several C-steps starting from the best solution found so far. This provides $\phi(\epsilon_a)$, $\phi(\epsilon_1)$, $\phi(\epsilon_2)$, and $\phi(\epsilon_b)$.
4. Compare $\phi(\epsilon_1)$ and $\phi(\epsilon_2)$, and update accordingly: i) the best solution, and ii) the bracketing interval such the new bracketing interval becomes either $[\epsilon_a, \epsilon_2]$ or $[\epsilon_1, \epsilon_b]$.
5. Generate a new percentage and form a new set of three segments.
6. Evaluate ϕ at the new generated percentage, ϵ. Go to step 4.

Fig. 2. Estimating the 8 coefficients using the LTS regression and the Golden Section Search algorithm

3.2 The Eight Coefficients

The algorithm provided by Rousseeuw assumes that the size of the subset, h, is known. In practice, however, h is not known. We propose an algorithm that simultaneously provides the LTS solution and the percentage of inliers.

Our problem consists in solving the 8-vector $\mathbf{a}$ using the over-constrained linear system (8). When the inlier percentage $\epsilon = \frac{h}{N}$ is unknown, we compute it by minimizing

$$\phi(\epsilon) = \frac{e(\epsilon)}{\epsilon^{\lambda}} \tag{11}$$

where λ is a predefined parameter (in all our tests described in the sequel, we used $\lambda = 6$). The above objective function $\phi(\epsilon)$ minimizes the trimmed error $e(\epsilon)$ while trying to use as many equations/pixels as possible. The minimization procedure is given a search interval $[\epsilon_a, \epsilon_b]$. Its assumes that in this interval the

function has a single minimum and locates the minimum by iterative bracketing with the Golden Section Search algorithm [8]. By default, the minimum of ϕ is searched in the interval [0.5, 1.0] assuming that the inlier percentage is at least 50%. Specifying the interval more strictly improves the computational efficiency of the method. In our case, for an initial bracketing of 10%, about six iterations are sufficient to locate the minimum of $\phi(\epsilon)$ with an acceptable precision of 0.01, i.e. the interval becomes less the 1%. Figure 2 summarizes the proposed approach that estimates the vector $\mathbf{a}$ using the LTS principles and the Golden Section Search algorithm.

3.3 3D Velocity

Once the vector $\mathbf{a} = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8)^T$ is recovered, the 3D velocity and the plane parameters, i.e., $\frac{V_x}{\gamma}$, $\frac{V_y}{\gamma}$, $\frac{V_z}{\gamma}$, Ω_x, Ω_y, Ω_z, α and β, can be recovered by solving the non-linear equations (6). This is carried out using the Levenberg-Marquardt technique [8]. In order to get an initial solution one can adopt assumptions for which Eq.(6) can be solved in a linear fashion. Alternatively, when tracking a video sequence the estimated velocity at the previous frame can be used as an initial solution for the current frame.

4 Experimental Results

Experiments have been carried out on synthetic and real images.

Fig. 3. A computer generated image of a 3D plane that is rotated about 60 degrees about an axis perpendicular to the optical axis

4.1 Synthetic Images

A synthetic planar scene was built whose texture is described by:

$$g(X_o, Y_o) \propto \cos(6\, X_o)\ (\sin(1.5\, X_o) + \sin(1.5\, Y_o))$$

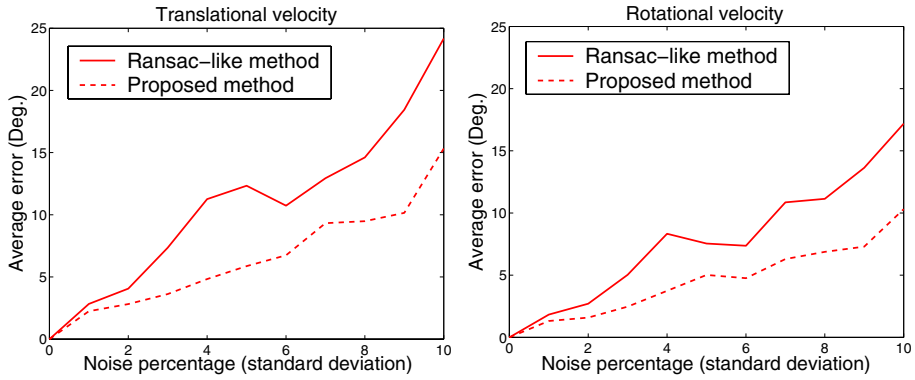

Fig. 4. Average errors obtained with a corrupted system (Gaussian noise and 10 % of outliers)

where X_o and Y_o where the 3D coordinates expressed in the plane coordinate system, see Figure 3. The resolution of the synthesized images was 160×160 pixels. The 3D plane was placed at 100cm from the camera whose focal length is set to 1000 pixels. A synthesized image sequence of the above planar scene was generated according to a nominal camera velocity $(\mathbf{V}_n, \boldsymbol{\Omega}_n)$. A reference image for which we like to compute the camera velocity was then fixed. The associated image derivatives can be computed or set to their theoretical values. Since we use synthetic data, the ground-truth values for the image derivatives as well as for the camera velocity are known. The nominal velocity $(\mathbf{V}_n(cm/s), \boldsymbol{\Omega}_n(rad/s))$ was set to $(10, 10, 1, 0.1, 0.15, 0.1)^T$. The corresponding linear system (8) was then corrupted by adding Gaussian noise and outliers to the spatio-temporal derivatives associated with each pixel. Our approach was then invoked to estimate the camera velocity. The discrepancies between the estimated parameters and their ground truth were then computed. In our case, the camera velocity was given by two vectors: (i) the scaled translational velocity, and (ii) the rotational velocity. Thus, the accuracy of the estimated parameters can be summarized by the angle between the direction of the estimated vector and its ground truth direction.

Figure 4 illustrates the obtained average errors associated with the camera velocity as a function of the Gaussian noise standard deviation. The solid curve corresponds to a RANSAC-like approach adopting a robust threshold (Eq.(8)), and the dashed curve to our proposed robust solution (Section 3). Each average error was computed with 50 random trials. As can be seen, unlike the RANSAC technique, our proposed method has provided more accurate solution. In the above experiment the percentage of outliers was set to 10%.

4.2 Real Images

The experiment was conducted on a 300-frame long video sequence of a moving scene (a newspaper) captured by a steady-camera, see Figure 5. The resolution

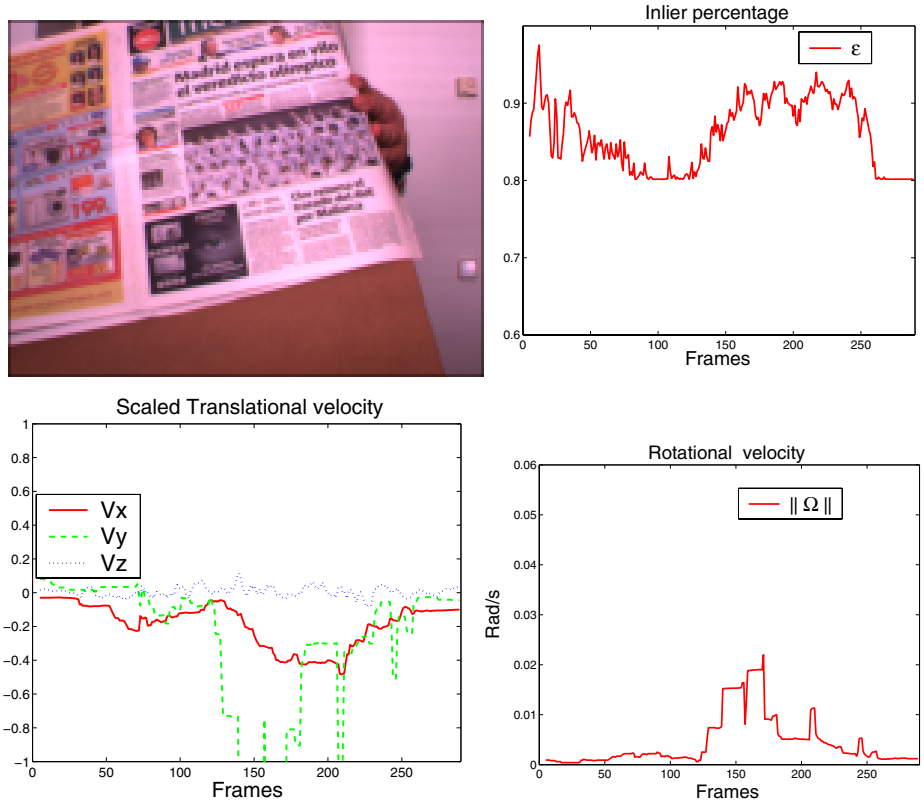

Fig. 5. Top: The used video and the estimated inlier percentage. **Bottom:** The estimated translational and rotational velocities.

is 160×120 pixels. We used 9 consecutive images to compute the temporal derivatives. The top-right shows the estimated inlier percentage. The bottom-left and bottom-right show the estimated 3D translational velocity $(\frac{V_x}{\gamma}, \frac{V_y}{\gamma}, \frac{V_z}{\gamma})$ and the rotational velocity $||\boldsymbol{\Omega}||$, respectively.

Although, the ground-truth is not known, we have found that the estimated 3D motion was consistent with the video.

5 Conclusion

This paper presented an approach to the 3D velocity estimation from spatio-temporal image derivatives. The approach includes a novel robust estimator combining the LTS principles and the Golden Section Search algorithm.

References

1. Alon, J., Sclaroff, S.: Recursive estimation of motion and planar structure. In: IEEE Conference on Computer Vision and Pattern Recognition. (2002)
2. Weng, J., Huang, T.S., Ahuja, N.: Motion and Structure from Image Sequences. Springer-Verlag, Berlin (1993)
3. Brooks, M., Chojnacki, W., Baumela, L.: Determining the egomotion of an uncalibrated camera from instantaneous optical flow. Journal of the Optical Society of America A **14**(10) (1997) 2670–2677
4. Brodsky, T., Fermuller, C.: Self-calibration from image derivatives. International Journal of Computer Vision **48**(2) (2002) 91–114
5. Fischler, M., Bolles, R.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communication ACM **24**(6) (1981) 381–395
6. Rousseeuw, P.J., Leroy, A.: Robust Regression and Outlier Detection. John Wiley & Sons, New York (1987)
7. Rousseeuw, P.J., Driessen, K.V.: Computing LTS regression for large data sets. Estadistica **54** (2002) 163–190
8. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C. Cambridge University Press (1992)