Accurate and Model-Free Pose Estimation of Small Objects for Crash Video Analysis

Stefan K. Gehrig
HPC 050 - G 024
DaimlerChrysler AG
71059 Sindelfingen
Germany
stefan.gehrig@dcx.com

Hernán Badino
HPC 050 - G 024
DaimlerChrysler AG
71059 Sindelfingen
Germany
hernan.badino@dcx.com

Pascal Paysan
Computer Science Department
Basel University
4056 Basel
Switzerland
pascal.paysan@unibas.ch

Abstract

We propose a novel model-free pose estimation algorithm to estimate the relative pose of a rigid object. In most pose estimation algorithms, the object of interest covers a large portion of the image. We focus on pose estimation of small objects covering a field of view of less than 5° by 5° using stereo vision.

With this new algorithm suitable for small objects, we investigate the effect of the object size on the pose accuracy. In addition, we introduce an object tracking technique that is insensitive to partial occlusion.

The main application for this method is the analysis of crash video sequences. In this context, high-speed cameras with high resolution are used. However, the method easily extends to real-time robotics applications with VGA-size images, where relative pose estimation is needed, e.g. for manipulator control.

1 Introduction

Pose estimation is a problem that has undergone much research in Computer Vision. Most pose estimation algorithms rely on models of the observed object (see e.g. [11]). We perform model-free pose estimation, i.e. pose estimation w.r.t. to a reference pose, which is the initial pose in our context. So we obtain a relative pose estimate.

The main application for us is the analysis of stereo crash video sequences. With a stereoscopic sensor setup, it is possible to obtain dense 3D information of the crash scene independent of photogrammetric markers, which are the only source of 3D information in standard crash analysis. In addition, using pose estimation, rotations of rigid parts in the scene can be measured. So for the first time, the maximal torsion of dummy body parts can be obtained. Besides supporting car manufacturers in design of passive safety components, dummy manufacturers can also use this data to put more biological plausibility into their dummies.

A problem closely related to pose estimation is ego-motion estimation. The algorithm proposed here was initially developed for ego-motion estimation by computing the optimal rotation and translation between the tracked static points of multiple frames [2].

The remainder of this paper is organized as follows: In Section 2 we present a short overview of pose estimation using stereo vision. The proposed algorithm is detailed in

BMVC 2006 doi:10.5244/C.20.103

Section 3. Adaptions and input to the algorithm are elaborated in Section 4. Simulation results as well as experimental results on real crash image sequences are presented in Section 5. In the last section we summarize the paper.

2 Related Work

Pose estimation is a central problem in photogrammetry and Computer Vision. The term is used in several related meanings covering calibration, ego-motion, and object pose estimation. We focus on pose estimation of rigid objects viewed from a static, calibrated stereo camera setup.

There exists a vast literature on pose estimation (see e.g. [11]). In this short literature overview, we limit ourselves to work on pose estimation via stereo vision. [4] estimates human poses from stereo images. Here, based on the stereo depth data, human body models are matched. For face tracking, the head pose is estimated using stereo information combined with head models [12]. Investigations on pose estimation of small objects covering only a small field of view are rare in the literature.

Ego-motion and pose estimation are related problems and can be solved using the same mathematical framework (see e.g. [8]). For an overview of robotic camera pose estimation techniques refer to [5]. We adopt an ego-motion estimation technique for use in pose estimation. We start with a review of the ego-motion algorithm, first described in [1] and applied to moving object detection in [2]. In the section following that, we describe adoptions to the algorithm for pose estimation.

3 Robust Ego-Motion Estimation

Computing ego-motion from an image sequence means obtaining the change of position and orientation of the observer with respect to a static scene, i.e. the motion is relative to an environment which is considered static. In most approaches this fact is exploited and ego-motion is computed as the inverse of the scene motion [9], [10], [14], [1]. In the latter a robust approach for the accurate estimation of the six d.o.f. of motion (three components for translation and three for rotation) in traffic situations is presented. In this approach, stereo is computed at different times and clouds of 3D points are obtained. The optical flow establishes the point-to-point correspondence in time. The motion of the camera is computed with a least-squares approach finding the optimal rotation and translation between the clouds. In the next subsections we briefly review the main steps of this approach.

3.1 Obtaining the Absolute Orientation Between Two Frames

Let $X = \{\vec{x}_i\}$ be the set of 3D points of the previous frame and $P = \{\vec{p}_i\}$ the set of 3D points observed at the current frame, where $\vec{x}_i \leftrightarrow \vec{p}_i$, i.e. $\vec{p}_i$ is the version at time t_k, transformed from the point $\vec{x}_i$ at time t_{k-1} (k frame index). In order to obtain the motion of the camera between the current and the previous frame we minimize a function which is expressed as the sum of the weighted residual errors between the rotated and translated

data set X with the data set P, i.e.:

$$\sum_{i=1}^{n} w_i \|\vec{p}_i - R_k \vec{x}_i - \vec{d}_k\|^2 \tag{1}$$

where n is the amount of points in the sets, R_k is a rotation matrix, $\vec{d}_k$ is a translation vector, and w_i are individual weights representing the expected error in the measurement of the points. To solve this least-squares problem we use the method presented by Horn [6], which provides a closed form solution using unit quaternions. In this method the optimal rotation quaternion is obtained as the eigenvector corresponding to the largest positive eigenvalue of a 4×4 matrix. The quaternion is then converted to the rotation matrix. The translation is computed as the difference of the centroid of data set P and the rotated centroid of data set X. The computation of the relative orientation is not constrained to this specific method. Lorusso *et al* [7] shortly describe and compare this method and another three methods for solving the absolute orientation problem in closed form.

3.2 Motion over Several Frames

In order to simplify the notation of the following subsections, we represent the motion in homogeneous coordinates. The computed motion of the camera between two consecutive frames, i.e. from frame $k-1$ to frame k, is represented by the matrix M'_k where:

$$M'_k = \begin{bmatrix} R_k & \vec{d}_k \\ 0 & 1 \end{bmatrix} \tag{2}$$

The rotation matrix $\hat{R}_k$ and translation vector $\vec{\hat{d}}_k$, i.e. the object pose are obtained by just inverting M'_k, i.e.:

$$M'^{-1}_k = \begin{bmatrix} \hat{R}_k & \vec{\hat{d}}_k \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_k^{-1} & -R_k^{-1}\vec{d}_k \\ 0 & 1 \end{bmatrix} \tag{3}$$

The total motion of the camera since initialization can be obtained as the products of the individual motion matrices:

$$M_k = \prod_{i=1}^{k} M'_i \tag{4}$$

A sub-chain of movements from time t_n to time t_m is:

$$M_{n,m} = M_n^{-1} M_m = \prod_{i=n+1}^{m} M'_i \tag{5}$$

Figure 1 shows an example of motion integration with matrices. As we will show later in section 3.4 equation 5 will support the integration of the motion between two non-consecutive frames (multi-step estimation).

3.3 Smoothness Motion Constraint

Optical flow and/or stereo can deliver false information about 3D position or image point correspondence between image frames. Some of the points might also correspond to an independently moving object. A robust method should still be able to give accurate results

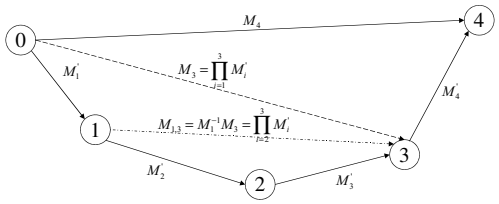

Figure 1: The integration of motion over time can be obtained by just multiplying the individual motion matrices. Every circle denotes the state (position and orientation) of the camera at time t_i. Every arrow indicates the motion between two states in 3D-space.

in such situations. If the frame rate is high enough in order to obtain a smooth motion between consecutive frames, then the current motion is similar to the immediate previous motion. Therefore, before including the pair of points $\vec{p}_i$ and $\vec{x}_i$ into their corresponding data sets P and X, we evaluate if the vector $\vec{v}_i = \overrightarrow{p_i x_i}$ indicates a coherent movement. Let us define $\vec{m} = \begin{bmatrix} \dot{x}_{max} & \dot{y}_{max} & \dot{z}_{max} & 1 \end{bmatrix}$ as the maximal accepted error of the position of a 3D point with respect to a predicted position. Based on our previous ego-motion estimation step we evaluate the motion coherence of the vector $\vec{v}_i$ as:

$$\vec{c}_i = M'_{k-1}\vec{x}_i - \vec{p}_i \tag{6}$$

i.e. the error of our prediction. If the absolute value of any component of c_i is larger than $\vec{m}$ the pair of points are discarded and not included in the data sets for the posterior computation of relative orientation. Otherwise we weight the pair of points as the ratio of change with respect to the last motion:

$$w_i = 1 - \frac{\|\vec{c}_i\|^2}{\|\vec{m}\|^2} \tag{7}$$

which is later used in equation 1. Equations 6 and 7 define the smoothness motion constraint (SMC). One could also include the geometric stereo accuracy as part of the weight, but this makes little difference for points with similar depth.

3.4 Multi-Frame Estimation

Single step estimation, i.e. the estimation of the motion parameters from the current and previous frame is the standard case in most approaches. If we are able to track points over m frames, then we can also compute the motion between the current and the m previous frames and integrate this motion into the single step estimation (see figure 2). The estimation of motion between frame m and the current frame k ($m < k-1$) follows exactly the same procedure as explained above. Only when applying the SMC, a small change takes place, since the prediction of the position for $k-m$ frames is not the same as for a single step. In other words, the matrix M'_{k-1} of equation 6 is not valid any more. If the single step estimation for the current frame was already computed as $\tilde{M}_k$ equation 6 becomes:

$$\vec{c}_i = M_{k-m}^{-1} M_{k-1} \tilde{M}_k \vec{x}_i - \vec{p}_i. \tag{8}$$

Equation 8 represents the estimated motion between times t_{k-m} and t_{k-1} (from equation 5), updated with the current simple step estimation of time t_k. This allows the SMC to be even more precise, since the uncertainty in the movement is now based on an updated prediction. On the contrary in the single step estimation, the uncertainty is based on a position defined by the last motion.

Once the camera motion matrix $\tilde{M}_{m,k}$ between times t_{k-m} and t_k is obtained, it is integrated with the single step estimation. This is performed by an interpolation. The interpolation of motion matrices makes sense if they are estimations of the same motion. This is not the case since the single step motion matrix is referred to as the motion between the last two frames and the multi-step motion matrix as the motion between m frames in the past to the current one. Thus, the matrices to be interpolated are $\tilde{M}_k$ and $M_{m,k-1}^{-1}\tilde{M}_{m,k}$ (see figure 2). The corresponding rotation matrices are converted to quaternions in order to apply a spherical linear interpolation. The interpolated quaternion is converted to the final rotation matrix R_k. Translation vectors are linearly interpolated, obtaining the new translation vector $\vec{t}_k$. The factors of the interpolation are given by the weighted sum of the quadratic deviations obtained when computing the relative motion of equation 1. The

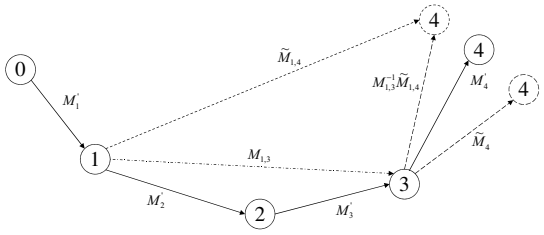

Figure 2: multi-frame approach. Circles represent the position and orientation of the camera. Vectors indicate motion in 3D-space. $\tilde{M}_4$ (single step estimation) and $M_{1,3}^{-1}\tilde{M}_{1,4}$ (multi-step estimation) are interpolated in order to obtain the final estimation M_4'.

multi-frame approach performs better thanks to the integration of more measurements. It also reduces the integration of the errors produced by the single-step estimation between the considered time points. In fact, our experiments have shown that without the multi-frame approach the estimation degenerates quickly and, normally, after a few hundred frames the ego-position diverges dramatically from the true solution. Thus, the multi-frame approach provides additional stability to the estimation process.

4 Model-free pose Estimation

4.1 Adaptions to the Ego-motion Estimation Algorithm

In our pose estimation scenario, the stereo camera system remains static and the observed object moves. All equations shown above remain the same since the relative motion is identical. However, the smoothness motion constraint is not related to the motion of the camera any more but to the observed object. We obtain the initial object position with the initial stereo computation. From there, we transform the motion of the object back into the static reference frame and compare the predicted position via smoothness motion

Figure 3: Smallest and largest dice simulation image.

constraint to the measured position. This way, tracks that are not on the observed object can be eliminated for pose estimation.

4.2 Stereo and Optical Flow Computation

We use a pyramidal correlation-based stereo computation after a planar rectification of the input images [3]. The sum of squared differences (SSD) is used as correlation measure. For tracking, we use the Kanade-Lukas-Tomasi Tracker [13]. Both algorithms can be substituted with other stereo or tracking algorithms. A pure optical flow without tracking cannot be used directly for multi-frame estimation since the flow computation is by definition only performed at integer positions.

4.3 Object Tracking and Point Selection

We select the object of interest manually in the scene. From there on, the object is tracked automatically using the average motion vector of the observed object. To obtain that motion vector, all tracks that originate within the selected region of interest are considered for pose estimation. But only tracks that pass the smoothness motion constraint are used for average motion computation. This way, partial occlusion is handled easily, since these flow vectors are not consistent with the current pose. Therefore, the manual selection does by no means require to be precise as long as the majority of the flow vectors in the selected region corresponds to the object of interest.

5 Results

5.1 Simulation Results

In order to verify the obtainable accuracy of the algorithm, a simulation scene with a dice of variable size is used. Image dimensions are 1600 by 1200 pixels (comparable to high speed/high resolution cameras), and the dice is set at 10m distance (see Figure 3). We varied the size of the dice and let it rotate around the vertical axis to obtain a yaw motion of 2°/frame from the camera perspective. Note, that this motion is much harder to measure than a camera roll motion within the image plane. Stereo and KLT tracks were computed on these images. The obtained errors are listed in the table below and show, that even small objects allow an accurate pose estimation. The maximum errors, including the integrated pose errors over 100 frames, stay within 10°, the average errors stay well below 1°. Multi-frame estimation up to 20 frames was used. Other publications rarely show data on pose estimation algorithms on small objects, so no comparison data can be provided.

Object size (10m distance)	0.25m	0.50m	0.75m	1.00m
Object size [pixel]	70	140	210	280
Number of tracks	2000	8000	11000	12000
Mean Error (std deviation)				
Δ pitch[°/frame]	0.04±0.07	0.00±0.02	0.00±0.01	0.00±0.003
Δ yaw[°/frame]	0.12±0.25	0.01±0.08	0.01±0.02	0.00±0.02
Δ roll[°/frame]	0.01±0.01	0.00±0.003	0.00±0.001	0.00±0.001
integrated pitch[°]	0.52±0.19	0.06±0.09	0.06±0.02	0.01±0.01
slope of integrated yaw[°/frame]	0.08±0.30	0.00±0.07	0.00±0.05	0.00±0.03
integrated roll[°]	0.50±0.18	-0.06±0.09	-0.03±0.02	0.03±0.01
Maximum Error				
Δ pitch[°/frame]	1.9	0.42	0.19	0.11
Δ yaw[°/frame]	4.3	2.9	1.1	0.46
Δ roll[°/frame]	0.3	0.13	0.04	0.04
integrated pitch [°]	6.9	1.6	0.4	0.2
integrated yaw [°]	10.0	2.3	0.8	0.4
integrated roll [°]	6.7	0.7	0.38	0.22

Table 1: Accuracy of pose estimation for varying object sizes. Motions are described in the camera coordinate system, pitch motion around the horizontal axis, yaw around the vertical axis and roll in the image plane.

Figure 4: First and last frame of an A-class offset crash sequence. The middle image shows the moment of deepest airbag penetration.

5.2 Crash Results

We verified the algorithm on several crash video scenes comprising of pedestrian crashes, offset crashes, and crash sequences imaging the bottom of a vehicle. We illustrate the performance showing results of an offset crash, where the driver dummy almost disappears in the airbag at one moment in the sequence. The sequence is recorded at 1000 Hz and consists of 300 images (see Figure 4). Here we used multi-frame estimation up to 100 frames in order to cope with the massive occlusion. To check the plausibility of the measurements, we also tracked the region of the driver door, specifically the letters. Knowing the vehicle dynamics of such offset crashes we expect a yaw motion away from the barrier and a slight camera roll motion due to jumping. Both behaviors can be clearly seen in the plot of the integrated angles of the estimated poses in Figure 8. The pitch motion is expected to remain around 0°and the deviations from that illustrate the noise level of the measurements.

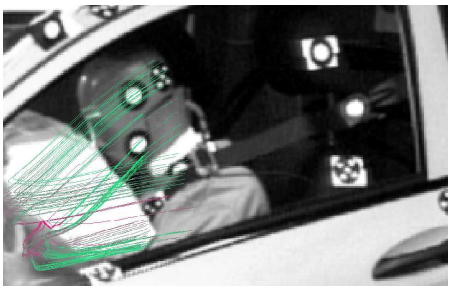

Figure 5: Used tracks for multi-frame estimation in frame 200. Red denotes large deviation to the predicted position, cyan means good agreement. Most good tracks stay at the five-dot-marker at the right part of the head that stays visible throughout the sequence.

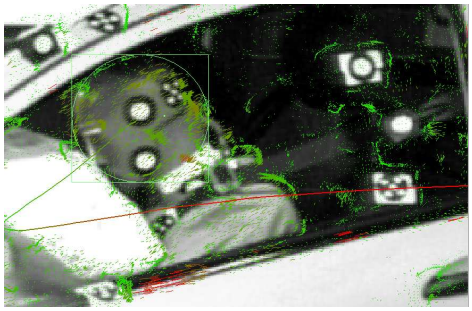

Figure 6: KLT tracks on the same scene. Red denotes large flow vectors, green small flows. the circle shows the tracked region of interest.

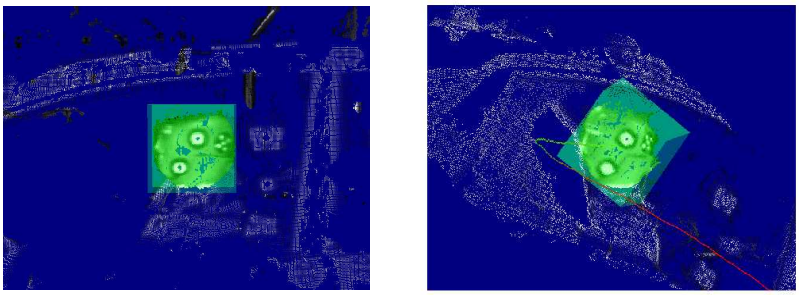

Figure 7: 3D view of the scene. The position of the camera is based on the obtained pose and transforms the head to its initial orientation, viewed from 2.5m distance. The right image depicts the transformed pose of image 200 after reappearing from the airbag. Note the good agreement to the initial pose shown in the left image.

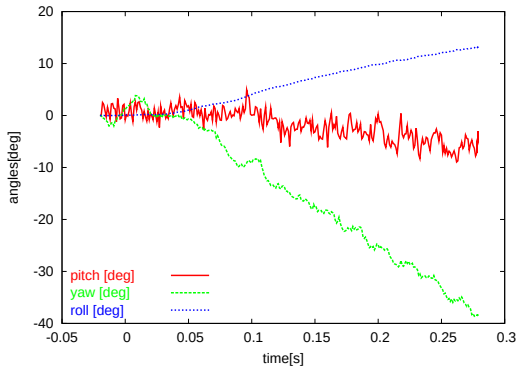

Figure 8: Orientation of the text on the driver door throughout the offset crash. The vehicle pitching up in the air is clearly visible (corresponds to the camera roll angle), and the yaw motion due to the offset barrier is reflected in the yaw angle. The camera pitch angle should remain around 0 in the sequence, but exhibits some noise due to the small size of the object.

For the pedestrian crash video, we tracked the head, the back, and the lower leg of the pedestrian throughout the sequence of 250 frames with little integrated rotation error.

5.3 Computation Time

The computationally relevant parts of the algorithm are rectification, stereo computation, tracking, and pose estimation. For high resolution images and tracking a 3D cube of 280 by 280 pixel image size, we obtain 160ms for rectification of the full stereo pair, stereo computation time 850ms, tracking 750ms, and the actual pose estimation 750ms (with 20 frames multi-frame estimation). 12000 flows were used. Stereo was computed densely and with one pyramid level with a generous margin around the region of interest to make use of stereo post-processing techniques such as dynamic programming and median filter. The same settings applied to the dummy head in the offset crash sequence yields 350ms for stereo, 250ms for tracking and 100ms for pose estimation. 1000 points were tracked. These time measurements were conducted on a Pentium M 1.7GHz laptop.

Real-time capable computation times are obtained for the ego-motion version of the algorithm (see [2]).

6 Summary

We have proposed a novel model-free pose estimation algorithm suitable for small objects. Simulation studies and test on real image sequences validate the algorithm. We have investigated the accuracy that can be obtained with such a technique on small objects. The average error is less than 1°on a 1.5° by 1.5° field-of-view object. In addition, we introduced a simple and efficient tracking technique that is able to handle partial occlusions. In the future, we also want to apply 3D models to the pose estimation process and we want to investigate the accuracy of such a technique.

References

[1] H. Badino. A robust approach for ego-motion estimation using a mobile stereo platform. In 1st *International Workshop on Complex Motion (IWCM'04)*, Günzburg, Germany, October 2004. Springer.

[2] H. Badino, U.Franke, C.Rabe, and S.Gehrig. Stereo-vision based detection of moving objects under strong camera motion. In *International Conference on Computer Vision Theory and Applications*, Setubal, Portugal, February 2006.

[3] U. Franke. Real-time stereo vision for urban traffic scene understanding. In *Proceedings of the Intelligent Vehicles 00 Symposium*, 2000.

[4] D. Grest, J. Woetzel, and R. Koch. Nonlinear body pose estimation from depth images. In *DAGM*, 2005.

[5] B. Grinstead, A. Koschan, and M. A. Abidi. A comparison of pose estimation techniques: Hardware vs. video. In *SPIE Unmanned Ground Vehicle Technology. Orlando, FL*, pages 166–173, 2005.

[6] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. In *Journal of the Optical Society of America A*, volume 4(4), pages 629–642, April 1987.

[7] A. Lorusso, D. Eggert, and R. B. Fisher. A comparison of four algorithms for estimating 3-d rigid transformations. In *Proc. British Machine Vision Conference*, Birmingham, 1995.

[8] C.P. Lu, G. D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Transactions On Pattern Analysis And Machine Intelligence,*, 22(6):610–622, 2000.

[9] L. Matthies and S. A. Shafer. Error modeling in stereo navigation. In *IEEE Journal of Robotics and Automation*, volume RA-3(3), pages 239–248, June 1987.

[10] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone. Rover navigation using stereo ego-motion. In *Robotics and Autonomous Systems*, volume 43(4), pages 215–229, June 2003.

[11] B. Rosenhahn, C. Perwass, and G. Sommer. Pose estimation of 3d free-form contours. *International Journal of Computer Vision*, 62(3):267–289, 2005.

[12] E. Seemann, K. Nickel, and R. Stiefelhagen. Head pose estimation using stereo vision for human-robot interaction. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 626–631, 2004.

[13] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, June 1994.

[14] W. van der M., D. Fontijne, L. Dorst, and F. C. A. Groen. Vehicle ego-motion estimation with geometric algebra. In *Proceedings IEEE Intelligent Vehicle Symposium, Versialles, France, May 18-20 2002*, May 2002.