Sünderhauf, N., Konolige, K., Lacroix, S. & Protzel, P. (2005). Visual Odometry using Sparse Bundle Adjustment on an Autonomous Outdoor Vehicle. In Levi, Schanz, Lafrenz, Avrutin (Hrsg.), Tagungsband Autonome Mobile Systeme 2005, Reihe Informatik aktuell, Springer Verlag, S. 157-163.

Visual Odometry Using Sparse Bundle Adjustment on an Autonomous Outdoor Vehicle

Niko Sünderhauf[1], Kurt Konolige [2], Simon Lacroix [3], Peter Protzel[1]

[1]Technische Universität Chemnitz
Fakulät für Elektrotechnik und Informationstechnik
niko.suenderhauf@informatik.de
peter.protzel@etit.tu-chemnitz.de
[2]SRI International, Menlo Park, USA
konolige@ai.sri.com
[3]LAAS/CNRS, Toulouse, Frankreich
simon.lacroix@laas.fr

Abstract. Visual Odometry is the process of estimating the movement of a (stereo) camera through its environment by matching point features between pairs of consecutive image frames. No prior knowledge of the scene nor the motion is necessary. In this work, we present a visual odometry approach using a specialized method of Sparse Bundle Adjustment. We show experimental results that proof our approach to be a feasible method for estimating motion in unstructured outdoor environments.

1 Introduction

Estimating the motion of a mobile robot is one of the crucial issues in the SLAM problem. Besides odometry, inertia sensors, DGPS, laser range finders and so on, vision based algorithms can contribute a lot of information. Visual Odometry approaches have recently been used in different ways [1] [2]. However, the basic algorithm behind these approaches is always the same: The first step is detecting feature points (or interest points) in the images. This is usually done using the *Harris Corner Detector* [3] which has profen to be a very stable operator in the sense of robustness and invariance against image noise [4]. The second step is to match interest points between the left and right images of a single stereo frame and between two consecutive frames. This matching can be done using a correlation based method as in [1].
After interest point detection and successful matching we can go on and determine the motion the camera undertook between two pairs of images. A variety of methods is available, but in this paper we will concentrate on a method called Sparse Bundle Adjustment.

2 Sparse Bundle Adjustment

Bundle Adjustment provides a solution to the following problem: Consider a set of world points $\mathbf{X}_j$ is seen from a set of cameras with camera matrices P_i. Each

camera projects $\mathbf{X}_j$ to $\mathbf{x}_{ij} = P_i\mathbf{X}_j$, so that $\mathbf{x}_{ij}$ are the image coordinates of the j-th world point in the i-th image.
What are the "optimal" projection matrices P_i and world coordinates $\mathbf{X}_j$ so that the summed squared *reprojection error* is minimal?
Thus we want to solve

$$\min_{P_i,\mathbf{X}_j} \sum_{ij} d(P_i\mathbf{X}_j,\ \mathbf{x}_{ij})^2 \tag{1}$$

where $d(\mathbf{x},\mathbf{y})$ is the Euclidean distance between image points $\mathbf{x}$ and $\mathbf{y}$.
Bundle Adjustment is a non-linear minimization problem which can be solved using iterative non-linear least squares methods such as Levenberg-Marquardt. A very efficient solution to the problem has been proposed by [5] and implemented by [6].
Solving (1) with Levenberg-Marquardt involves iterative solving of normal equations of the form

$$\mathbf{J}^T\mathbf{J}\delta = \mathbf{J}^T\epsilon \tag{2}$$

where $\mathbf{J}$ is the Jacobian of the *reprojection function* $f_{(\mathbf{a},\mathbf{b})} = \tilde{\mathbf{x}}$. f takes $\mathbf{a} = (\mathbf{a}_1^T, \mathbf{a}_2^T, \dots \mathbf{a}_m^T)^T$ and $\mathbf{b} = (\mathbf{b}_1^T, \mathbf{b}_2^T, \dots \mathbf{b}_n^T)^T$ as parameters and returns $\tilde{\mathbf{x}} = (\tilde{\mathbf{x}}_{11}^T, \tilde{\mathbf{x}}_{12}^T, \dots \tilde{\mathbf{x}}_{mn}^T)^T$. Here $\mathbf{a}_i$ is the 6-Vector of the currently estimated parameters of the i-th camera, $\mathbf{b}_j$ is the 3-vector with the parameters of the j-th world point respectively. The projected image coordinates of world point j in the i-th image (according to $\mathbf{a}_i$ and $\mathbf{b}_j$) are given by $\tilde{\mathbf{x}}_{ij}$.

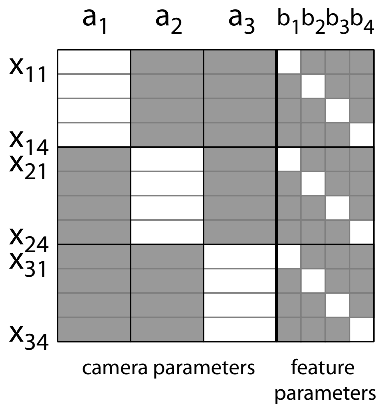

Fig. 1. Structure of a sparse Jacobian matrix for a bundle adjustment problem consisting of 3 cameras and 4 feature points. The gray entries are all zero.

The Jacobian $\mathbf{J}$ of f is made up of entries $\partial\tilde{\mathbf{x}}_{ij}/\partial a_k$ and $\partial\tilde{\mathbf{x}}_{ij}/\partial b_k$. One may notice, that $\partial\tilde{\mathbf{x}}_{ij}/\partial a_k = 0$ unless $i = k$ and similar $\partial\tilde{\mathbf{x}}_{ij}/\partial b_k = 0$ unless $j = k$. This is simply because the projected coordinates of world point j in the i-th image are not dependent on any camera's parameters but the i-th and they neither depend on any other world point but the j-th.

Given this, one verifies that $\mathbf{J}$ contains large blocks with 0-entries (see figure 1). In other words, $\mathbf{J}$ has a sparse structure. The *Sparse Bundle Adjustment* (SBA) implementation as presented by [6] takes advantage of that very structure and thus enables SBA to solve huge minimization problems over many thousands of variables within seconds on a standard PC. The C source code is freely available (under the terms of the GNU General Public License) on the author's website `http://www.ics.forth.gr/~lourakis/sba`.

3 Visual Motion Estimation using Sparse Bundle Adjustment

Sparse Bundle Adjustment (SBA) as implemented in [6] solves the minimization problem as stated in (1), not considering any uncertainty of whatever kind, assuming calibrated cameras and monocular image information.
We extended the original SBA implementation in a way that it can handle stereo data input directly. We therefore had to change some internal algorithms and datastructures, so that the routines take 4-vectors $\mathbf{x}_{ij}$ instead of 2-vectors. The additional two entries to these vectors are the image coordinates in the (right) stereo image. This way, we do not necessarily have to provide proper initial estimates for the world points $\mathbf{X}_j$ or the camera poses P_i. Although this impressively demonstrates the power of Bundle Adjustment, it is of course not an reasonable approach and should not be used in any seriously motivated application. Instead, we should provide SBA with initial estimates for the camera pose acquired by odometry or other sensor systems like inertial navigation systems, compass, or GPS and initialize the 3D coordinates of the feature points by triangulation. Starting with reasonable initial estimates speeds up the optimization process significantly and may detain the optimization from stepping into a false local minimum.
Besides the extension to stereo input, we also implemented a simple iterative outlier rejection method. After SBA finished its minimization loop with the solutions $\mathbf{P}^*$ and $\mathbf{X}^*$, there is still an *residual error* $\epsilon_{ij} = d(P_i^*\mathbf{X}_j^*,\ \mathbf{x}_{ij})^2$ left for every $\mathbf{x}_{ij}$. A simple outlier rejection is to discard all $\mathbf{x}_{ij}$ where $|\epsilon_{ij} - \overline{\epsilon}| > k\sigma_\epsilon$ for any $k > 0$ where $\overline{\epsilon}$ is the mean of all residual errors and σ_ϵ is the corresponding standard deviation. $k = 1.5$ was chosen empirically. After the so determined outliers have been removed from the input data, SBA is restarted using $\mathbf{P}^*$ and $\mathbf{X}^*$ as initial estimates. The procedure is repeated for a fixed number of iterations or until no more outliers are found. This process helps decreasing the mean residual error and is in many cases sufficient to discard outliers arising from false or bad matches. However, as the experiments showed, it is a fairly naive approach, computationally quite expansive, and may fail sometimes. We therefore favor a more robust outlier rejection method, based on RANSAC.

After all these modifications, were able to use SBA for visual motion estimation in two different ways:

1. Sliding Window SBA

2. full SBA

Sliding Window SBA The simplest and fastest estimation method is estimating structure and motion parameters between two consecutive stereo frames only. The overall motion is obtained by simple concatenation or 'chaining' of the single estimates. Intuitively one will expect this to be fairly inaccurate, as possible small errors will accumulate quickly.
To avoid the problems of simple chaining, we implemented a sliding window SBA approach. Instead of optimizing for two consecutive images only, we choose a n-*window*, e.g. a subset of n images which we perform SBA upon. The pose and structure parameters estimated in this way are used as initial estimates in the next run, where we slide the window further one frame. With that basic sliding window approach, we would bundle adjust every consecutive n-window in the sequence of the obtained images, even if the robot has not or only very little moved while the images of the window were taken. Therefore another idea is to only include those images into the window, which pose are more than a certain threshold away from the pose of their respective predecessor in the window. The final algorithm can be summarized as follows:

1. Starting from image I_i find the closest image I_{i+k} so that the motion between I_i and I_{i+k} exceeds a certain threshold. This can be determined by pairwise SBA between I_i and I_{i+k} or, of course, using odometry data.
2. Add I_{i+k} to the window
3. Set $i = i + k$ and repeat from 1. until there are sufficient many (n) images in the window
4. bundle adjust the window using the poses obtained in step 1 as initial estimates

In this way a window size of two corresponds to the simple chaining approach.

Full SBA Full SBA optimizes the whole bundle of obtained images at once. It determines camera poses and structure parameters for all recorded frames in one big optimization loop. Although this should intuitively yield the best results, it is, due to its complexity, an off line (batch) method not usable to continuously update the robot's position as he moves along.

4 Experimental Results

During all experiments we used the triangulated world coordinates as initial estimates for the world points, but did not initialize the camera poses. Instead, we assumed the cameras did not move at all. The outlier rejection was enabled and limited to 10 iterations.
We tested our algorithms on a dataset of 80 images acquired in an outdoor environment using a stereovision bench made of two 640x480 greyscale cameras with

2.8mm lenses and a baseline of approximately 8cm. We would like to acknowledge Max Bajracharya from JPL for providing us with the images and data. Ground truth data was extracted from using a Leica ”total station” surveying instrument. Between 18 and 308 feature points were visible in each image, with 106 on average. Each point was visible in only 4 consecutive images on average.

Sliding Window SBA The dataset was tested with several window sizes and motion thresholds.
The average deviation from the ground truth position for all tested methods was approximately 23 cm (2.3 % of traveled distance). The error tends to increase slightly with increasing motion threshold above 20 cm because fewer feature points can be matched between two consecutive images and the quality of the matches drops with increasing movement between the images. However, the differences between the different parameter settings are not significant. They are far below 1% of the traveled distance. The camera was mounted very close to the ground and was tilted down, so only few feature points were identified (and succesfully tracked) in a feasible distance from the robot to compare the different window sizes and motion thresholds.

motion threshold	min.	max.	mean error
0	22.30	22.61	22.37
10	22.37	23.35	22.67
15	22.03	22.33	22.18
17	22.49	22.97	22.68
20	22.07	23.01	22.53
25	22.48	24.05	23.11
30	23.19	24.37	23.76
35	23.04	24.27	23.29
40	23.57	24.76	24.32

Table 1. Distance from the ground-truth end pose in cm for sliding window SBA

Full SBA The position estimated by full SBA was 20.48 cm away from the ground truth position which is slightly better than the above results, as we expected. The estimation involved 2129 3D points and 8462 image projections. 6947 parameters had to be estimated (3 for each 3D point and 7 for every camera pose). Our implementation took 6 iterations to detect and remove outliers. On a 1GHz P3 machine running Linux, the algorithm finished in 4.5 minutes. This rather long time is caused by the computationally expansive simple outlier rejection method and the stereo-extension forcing us to use SBA in the inefficient

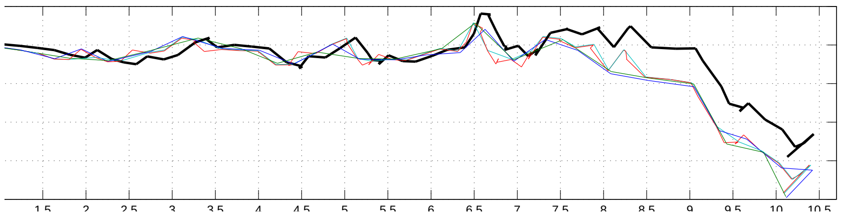

Fig. 2. Several estimated trajectories for the outdoor dataset. The thick black line is the ground truth.

'simple driver' mode. The required runtime may be reduced if more care is taken for runtime efficiency.

We also tested the algorithms with a sequence of indoor images. The results there were even better than for the outdoor sequence, mainly because feature points were visible longer and could be matched more accurately.

5 Conclusions and Further Work

The error, 2.3 % of the traveled distance for the outdoor data, proves SBA to be a feasible method for visual motion estimation. The tested datasets did not show significant differences among the different parameter settings. However, as a rule of thumb one should not use simple chaining (resp. window size 2) as the error tends to be higher than with window sizes 3 or 4. Window sizes above 4 or 5 do in general not help to improve the result, as feature points may in average not be tracked for more than 5 images. This, of course, is highly dependent on your matching and tracking algorithm, image quality and the environment. Larger windows increase computation time drastically, so a tradeoff between computational costs and accuracy has to be considered.

In further work, the methods will be refined to make them more robust against outliers arising from false matches. This will help decreasing the error and, if implemented efficiently, can even decrease overall computation time.
A possible way to achieve this is to use RANSAC [7] to estimate hypotheses for motion between two consecutive frames.
We can use three 3D point correspondences (from before and after the motion) to determine a least squares estimate of the 6D motion parameters as shown in [8] or [9]. A RANSAC approach will then be able to find a consistent estimate of the motion parameters, preventing the negative influence of outliers in the 3D point data. SBA can then be used to refine the solution on the resulting inlier sets of several consecutive frames at once. The techniques we presented here (sliding window, motion threshold) should of course be used further in such an approach.
Visual Odometry should also be fused with other motion-estimating sensors

or methods, such as DGPS or laser-scanmatching. To achieve this (using an Extended Kalman Filter) it is necessary to acquire the covariance matrix of SBA's results. This, in return, can be achieved by propagating the covariance matrix of the initial estimates (structure and motion) through SBA. In its current implementation, SBA uses the identity matrix as covariance matrix for the world points. It should be possible to use any arbitrary covariance matrix instead (see [6]). As pointed out in [5] in Algorithm A6.4, the covariance matrix for the camera parameters is the pseudo-inverse of SBA's internal matrix S. One can use the matrices V and U in the implementation to calculate S as given in [5].

Another interesting idea is to exploit knowledge about the robot's current environment in the feature selection process. For instance feature points on objects that are known to be moving could be discarded. Feature points may also be weighted in the estimation process by the type of terrain they are seen on. Points on concrete or asphalt may be more reliable and useful than points found in high grass (which is likely to move due to the wind). Finally, the feature detection and selection process itself could be modified. For instance the methods presented in [10] could improve feature quality significantly.

References

1. David Nister, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, pages 652–659, 2004.
2. C. Olson, L. Matthies, M. Schoppers, and Maimone Maimone. Robust stereo egomotion for long distance navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-00)*, pages 453–458, Los Alamitos, June 13–15 2000. IEEE.
3. M. Stephens C. Harris. A combined corner and edge detector, 1988.
4. C. Bauckhage C. Schmid, R. Mohr. Evaluation of interest point detectors, 2000.
5. R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
6. M.I.A. Lourakis and A.A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. Available from `http://www.ics.forth.gr/~lourakis/sba`.
7. Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.
8. Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380, 1991.
9. K.S. Arun, T.S. Huang, and S.D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700, 1987.
10. David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. In *International Journal of Computer Vision, 60, 2*, pages 91–110, 2004.