Sünderhauf, N. & Protzel, P. (2006). Towards Using Sparse Bundle Adjustment for Robust Stereo Odometry in Outdoor Terrain. In Proc. of Towards Autonomous Robotic Systems TAROS06, Guildford, UK, pp. 206–213.

Towards Using Sparse Bundle Adjustment for Robust Stereo Odometry in Outdoor Terrain

Niko Sünderhauf, Peter Protzel
Department of Electrical Engineering and Information Technology
Chemnitz University of Technology
09111 Chemnitz
Germany
{niko.suenderhauf, peter.protzel}@etit.tu-chemnitz.de

Abstract

Stereo 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 stereo odometry approach using a specialized method of Sparse Bundle Adjustment. We show experimental results with simulated and real-world data that prove 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. In general, vision based motion estimation algorithms track certain feature points (sometimes called interest points or simply landmarks) through a series of consecutive images and estimate the motion of the camera by evaluating these tracks, making use of the known camera parameters.
With monocular vision, a single observation does not give enough information to calculate the full state of the mentioned features or landmarks in the world, because we can not determine the depth or distance of a landmark from a monocular image. Bearing-Only SLAM approaches developed methods that allow the use of monocular vision for motion estimation nonetheless. (Lemaire and Lacroix, 2006) and (Solà et al., 2005) describe how to initialize the state of the landmarks using a set of Gaussians and pruning these Gaussians after several observations until a single Gaussian which estimates the true state of the landmark remains.
Stereo vision in contrast is able to determine the full state of each visible landmark with a single observation, making things easier in this sense. Stereo vision based approaches for visual odometry have recently been used in different ways, even on Mars (Cheng et al., 2006). However, the basic algorithm behind these approaches is always the same: The first step is detecting the landmarks or feature points in the images. This is usually done using the *Harris Corner Detector* (C. Harris, 1988) which has proven to be a very stable operator in the sense of robustness and invariance against image noise (Schmid et al., 2000). 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 simple correlation based method as in (Nister et al., 2004).
We will not go into the details of interest point detection and matching here. In our implementation Harris is used to find the interest points in the images. A fairly simple Sum-Of-Absolute-Differences approach is used during the matching. After interest point detection and successful matching we have two sets of corresponding 3D-points: One set with the coordinates before the motion and another set with coordinates after the motion. Now we can go on and determine the movement of the camera between the two sets of observations.

2. The Motion Between Two Frames

The problem of determining the relative motion that transforms a set of 3D-points into another is well known in the computer vision community and has been solved in several ways.
Suppose we are given two sets of rigid 3D-points $X = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n\}$ and $Y = \{\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n\}$ where $\mathbf{X}_i$ and $\mathbf{Y}_i$ are the 3D-coordinates of the i-th interest point *before* and *after* the motion. What is the translation t and rotation R that transforms X into Y so that the mean squared error ϵ^2 becomes minimal? The mean squared error is then given by

$$\epsilon^2_{(R,t)} = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{Y}_i - (R\mathbf{X}_i + t)\|^2 \qquad (1)$$

This problem is sometimes called the *absolute orientation problem.* The literature knows several solutions, like (Horn, 1987a), (Arun et al., 1987), (Horn, 1987b) (using quaternions) or (Huang et al., 1986) (iterative solution). The method described in this section was published in (Umeyama, 1991) and bases on (Horn, 1987a) and (Arun et al., 1987) but corrects a mistake which led to wrong results in some degenerated cases.
The idea behind the algorithm is to decouple translation and rotation. The coordinates of the points $\mathbf{X}_i$ and $\mathbf{Y}_i$ relative to their centroid μ_x and μ_y will be equal before and after the transformation. This is simply because the transformation by R and t is an euclidean transformation and does not affect the relative position of the points to each other. They are moved like one *rigid* body. Given this information. we can split the original problem into two parts:

1. Find R to minimize

$$\epsilon^2 = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{Y}_i - R\mathbf{X}_i\|^2 \tag{2}$$

2. Then the translation t is given by $t = \mu_y - R\mu_x$.

The minimization problem in (2) can be solved using the *singular value decomposition* SVD:

1. Calculate the centroids

$$\mu_x = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}_i \tag{3}$$

$$\mu_y = \frac{1}{n}\sum_{i=1}^{n} \mathbf{Y}_i \tag{4}$$

2.

$$\Sigma_{xy} = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{Y}_i - \mu_y)(\mathbf{X}_i - \mu_x)^T \tag{5}$$

3. Let UDV^T be the singular value decomposition of Σ_{xy}, SVD(Σ_{xy}).

4.

$$S = \begin{cases} I, & \text{if } \det(U)\cdot\det(V) = 1 \\ \text{diag}(1,1,\ldots,-1), & \text{if } \det(U)\cdot\det(V) = -1 \end{cases} \tag{6}$$

5.

$$R = USV^T \tag{7}$$

6.

$$t = \mu_y - R\mu_x \tag{8}$$

The problem stated above is clearly a least squares optimization problem, as we want to minimize a squared error measure. Least squares problems are prone to outliers which will lead to wrong results. What are outliers in the terms of our motion estimation problem? Remember that we work with pairs of matched image points (x_i, y_j). If all matches are correct, or only affected by Gaussian noise, the above algorithm will work well. But if some of the matches are wrong, so that x_i and y_j are not the projections of the same world point, the algorithm will fail. We can call these false matches outliers. However, there are robust methods that can cope with outliers in the data. So obviously, the above algorithm can not be used alone, without any sort of improvement, as long as we have to expect outliers in our matching data.

3. Robust Motion Estimation Using RANSAC

As we have seen in the last section, the SVD-based algorithm can not be used without any further improvement to make it robust against outliers in the data. RANSAC (Fischler and Bolles, 1981) is a very well known method for robust estimation. Inspired by (Nister, 2003), we decided to combine the SVD-based algorithm with RANSAC and use the above algorithm as hypothesis generator during the RANSAC-loop.
It is worth to take a look at the scoring function: The SVD-algorithm gives a certain hypothesis, which is a transformation H defined by R and t. Now we can transform every $\mathbf{X}_i \in X$ to $\hat{\mathbf{X}}_i = H\mathbf{X}_i$. If H was right, every $\hat{\mathbf{X}}_i$ should equal $\mathbf{Y}_i$ if $(\mathbf{X}_i, \mathbf{Y}_i)$ was a matched point pair. However due to small errors in the matching or estimation process, there will always be a small error, a difference between $\hat{\mathbf{X}}_i$ and $\mathbf{Y}_i$. We could now define an error measure for our scoring function to be

$$\epsilon_i^2 = \|\hat{\mathbf{X}}_i - \mathbf{Y}_i\|^2 \tag{9}$$

However, we know that the coordinates of both $\mathbf{X}_i$ and $\mathbf{Y}_i$ are affected by the triangulation error (remember that we retrieved both coordinates by triangulation during the stereo process). So we better do not use the 3D-coordinates for determining the error of our motion hypothesis H, as the triangulation error and the error from the hypothesis would accumulate and make the resulting error measure less significant. A better error measure can be achieved when we go back to the image space by projecting $\hat{\mathbf{X}}_i$ back into image coordinates and comparing it to the expected image coordinates given by y_i. Then we can define the error measure to be

$$\epsilon_i^2 = \|P\hat{\mathbf{X}}_i - y_i\|^2 \tag{10}$$

$$\epsilon_i^2 = \|PH\mathbf{X}_i - y_i\|^2 \tag{11}$$

where P is the camera projection matrix. ϵ will be an error in pixel units, which makes the evaluation easy:

The point pair $(\mathbf{X}_i, \mathbf{Y}_i)$ is a supporter of hypothesis H if and only if the error ϵ is below a certain threshold, e.g. 1 pixel. The algorithm now iterates as long as enough supporters for one hypothesis are found or as long as enough hypotheses have been tested.
With this implementation, we are able to estimate the movement of the stereo camera between two image pairs (stereo frames). We are able to cope with outliers that result from false matches during the interest point detection and matching process. Furthermore, due to the least squares optimization the algorithm can cope with the normal image noise that arises from errors in the triangulation during stereo processing and from the small instabilities from the interest point detection algorithms. We will see how we can improve the results of this algorithm and make it more robust during the rest of this paper.

4. Bundle Adjustment

The motion estimation method introduced so far estimated the motion between *two* consecutive frames. The upcoming section shows another approach that can estimate motion between more than two frames. How does that make sense? Remember that we estimated the motion from a set of matched point pairs $\mathbf{X}_i$ and $\mathbf{Y}_i$. Now consider that we move the camera again, so that $\mathbf{Y}_i$ can be matched with a point $\mathbf{Z}_i$ after the second motion. The approach we know so far would at first estimate motion M_1 from the point sets X and Y and then the second motion M_2 from Y and Z. Each of the two motions will be prone to small errors. Now, if we know that $\mathbf{X}_i$ and $\mathbf{Y}_i$ and $\mathbf{Z}_i$ are the coordinates of one and the same world point, why shouldn't we use that information? The algorithms we have seen so far are not able to make use of this extra knowledge. But *bundle adjustment* can. Bundle adjustment is able to estimate the motion between an arbitrary large number of frames. In addition, it does not only optimize the motion estimate, but can also reduce the errors that were introduced into the 3D-coordinates of our interest points during the stereo matching step. Figure 1 illustrates how the different estimation approaches use the provided images.

So in bundle adjustment, we have a set of world points $\mathbf{X}_j$ that is seen from a set of different cameras with camera matrices P_i. Of course, in reality we have only one camera that is moved through the environment, so that P_i is the camera matrix of our camera after the i-th motion. However, in bundle adjustment we usually treat the P_i as if they were different cameras. In fact, it does not make any difference if we have one moving camera or different rigid ones. The position and orientation of each of these cameras is contained in the camera projection matrix 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 (interest point) in the i-th image.

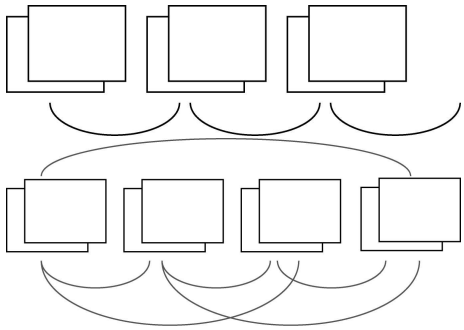

Figure 1: Top: Two-frame motion estimation methods estimate the motion between two consecutive frames or positions only. Bottom: Bundle adjustment can process the information from several successive stereo images. The links between the images can be seen as constraints that have to be regarded in the estimation step. The problem of bundle adjustment is much more constrained than two-frame problems.

The problem we want so solve is this: 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

$$\min_{P_i,\mathbf{X}_j} \sum_{ij} d(P_i\mathbf{X}_j,\ \mathbf{x}_{ij})^2 \tag{12}$$

where $d(\mathbf{x}, \mathbf{y})$ is the Euclidean distance between image points $\mathbf{x}$ and $\mathbf{y}$.
Equation (12) can be extended to a weighted least-squares formulation, such as

$$\min_{P_i,\mathbf{X}_j} \sum_{ij} w_{ij}\ d(P_i\mathbf{X}_j,\ \mathbf{x}_{ij})^2 \tag{13}$$

where the weights w_{ij} may be chosen according to the variances σ_{ij}^2 of the measured image coordinates $\mathbf{x}_{ij}$.

As we see from the equations above, bundle adjustment is a non-linear minimization problem providing a maximum likelihood estimate for both camera and structure parameters if the measurement noise is considered to be zero-mean Gaussian. It can be solved using iterative non-linear least squares methods such as Levenberg-Marquardt. A very comprehensive introduction to bundle adjustment in general can be found in (Triggs et al., 2000). This paper covers the whole spectrum of topics concerning BA, from basic optimization to efficient implementations.
Bundle adjustment (BA) can be used as a final optimization step after good initial estimates for P_i and $\mathbf{X}_j$ have been obtained by other methods, e.g. the approaches described earlier in this paper. Like other optimization methods, bundle adjustment is dependent

of a good initial estimate, as a starting point for the optimization routine. Therefore it is necessary to do a kind of pre-estimation step in order to acquire these initial estimates. As bundle adjustment is a nonlinear least squares method, it is prone to errors arising from outliers in the data to be optimized. So we also have to remove outliers from the data before we start BA in order to get valid results. In our implementation, both is done using the combined RANSAC and SVD algorithm approach already described.

A short reflection about the complexity and computational costs of bundle adjustment instantly reveals a distinct problem of the approach: Each of the world points $\mathbf{X}_j$ has 3 degrees of freedom that have to be estimated. Each of the projection matrices P_i has 11 degrees of freedom in general and still 6 DOF when the camera calibration is known. Thus, a reconstruction over n world points and m cameras requires a minimization over $3n + 6m$ parameters, which can become practically intractable very quickly with growing n and m.

5. Sparse Bundle Adjustment

However, an efficient solution to the problem has been proposed by (Hartley and Zisserman, 2004) and implemented by (Lourakis and Argyros, 2004).

Solving (12) with Levenberg-Marquardt involves iterative solving of normal equations of the form

$$(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})\delta = \mathbf{J}^T\epsilon \tag{14}$$

where $\mathbf{J}$ is the Jacobian of the *reprojection function* $f_{(\mathbf{a},\mathbf{b})} = \hat{\mathbf{x}}$. 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 $\hat{\mathbf{x}} = (\hat{\mathbf{x}}_{11}^T, \hat{\mathbf{x}}_{12}^T, \dots \hat{\mathbf{x}}_{mn}^T)^T$. Here $\mathbf{a}_i$ is the 7-vector of the currently estimated parameters of the i-th camera. We use a quaternion notation, so we have 4 parameters for the rotation plus 3 for translation. $\mathbf{b}_j$ 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 $\hat{\mathbf{x}}_{ij}$.

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 2). In other words, $\mathbf{J}$ has a sparse structure. The *Sparse Bundle Adjustment* (SBA) implementation as presented by (Lourakis and Argyros, 2004) 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`.

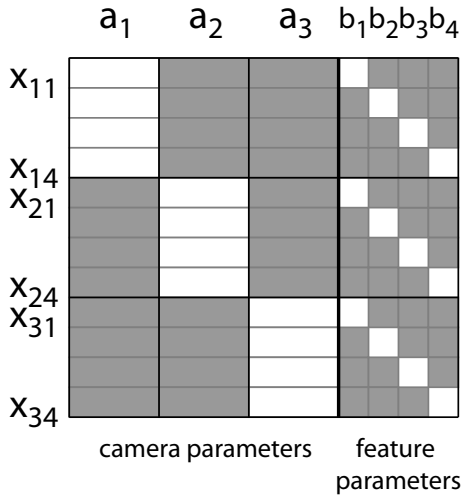

Figure 2: Structure of a sparse Jacobian matrix for a bundle adjustment problem consisting of 3 cameras and 4 feature points. The grey entries are all zero.

5.1 The Algorithm

Given the vectors $\mathbf{a}$ and $\mathbf{b}$ as defined above, we can start solving the sparse bundle adjustment problem with Levenberg-Marquardt. The procedure follows the standard LM-algorithm (see (Madsen et al., 2004) for a comprehensive introduction), until we reach the point where we have to solve the equation $(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})\delta = \mathbf{J}^T\epsilon$. This is the point where we can make use of the sparseness of $\mathbf{J}$. First we compute the matrices $\mathbf{A}_{ij} = [\partial\hat{\mathbf{x}}_{ij}/\mathbf{a}_j]$ and $\mathbf{B}_{ij} = [\partial\hat{\mathbf{x}}_{ij}/\mathbf{b}_i]$. The error vectors ϵ_{ij} are computed by $\epsilon_{ij} = \mathbf{x}_{ij} - \hat{\mathbf{x}}_{ij}$. Now we can compute some auxiliary variables:

$$U_j = \sum_i A_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} A_{ij} \quad V_i = \textstyle\sum_j B_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} B_{ij}$$

$$W_{ij} = \sum_j A_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} B_{ij} \qquad Y_{ij} = W_{ij} V_i^{*-1}$$

$$\epsilon_{\mathbf{a}_j} = \sum_i A_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} \epsilon_{ij} \quad \epsilon_{\mathbf{b}_i} = \textstyle\sum_j B_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} \epsilon_{ij}$$

Augment U_j and V_i by adding μ to their diagonal entries to yield U_j^* and V_i^*. μ is the dampening term from Levenberg-Marquardt.

Compute

$$Y_{ij} = W_{ij} V_i^{*-1}. \tag{15}$$

Let S be a $m \times m$ block matrix with blocks S_{jk} defined by

$$S_{jj} = -\sum_i Y_{ij} W_{ij}^T + U_j^* \tag{16}$$

and

$$S_{jk} = -\sum_i Y_{ij} W_{ik}^T \forall j \neq k \tag{17}$$

Define

$$\mathbf{e}_j = \epsilon_{\mathbf{a}_j} - \sum_i Y_{ij} \epsilon_{\mathbf{b}_i} \tag{18}$$

Now we can compute $\delta_{\mathbf{a}} = (\delta_{\mathbf{a}_1}^T, \ldots, \delta_{\mathbf{a}_m}^T)^T$ from

$$S\delta_{\mathbf{a}} = (\mathbf{e}_1^T, \ldots, \mathbf{e}_m^T)^T \tag{19}$$

Given $\delta_{\mathbf{a}}$, we can compute $\delta_{\mathbf{b}}$ from

$$\delta_{\mathbf{b}_i} = V_i^{*-1}(\epsilon_{\mathbf{b}_i} - \sum_j W_{ij}^T \delta_{\mathbf{a}_j}) \tag{20}$$

The sought solution δ of $(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})\delta = \mathbf{J}^T\epsilon$ is now formed by simply stacking $\delta = (\delta_{\mathbf{a}}^T, \delta_{\mathbf{b}}^T)^T$. This δ is the Levenberg-Marquardt step in the optimization routine and is added to the parameter vector p we want to optimize. Notice that δ contains the updates for both the camera and the 3D-point structure parameters. The above algorithm is the core part of the Levenberg-Marquardt optimization process. It is executed iteratively as long as $\|\delta\|$ is above a threshold (so the changes to the parameter vector are significant) or until a certain number of iterations have been reached. The efficiency of this approach lies in using the sparse structure: We do not have a measurement for every $\mathbf{x}_{ij}$, so some $\mathbf{x}_{ij}$ simply do not exist. These missing terms are now simply omitted from the relevant summations above. This includes all $A_{ij}, B_{ij}, \Sigma_{x_{ij}}, W_{ij}$, and Y_{ij}. We do not waste computation time by calculating meaningless (zero) values in the variables. At the same time, this strategy helps to reduce memory consumption, as we can keep the data structures containing the matrices small if we use techniques like the Compressed Row Storage format described in (Barrett et al., 1994). This technique has been used to store the sparse matrices in the SBA implementation.

5.2 Error-Modelling - Retrieving a Covariance Matrix

In all the equations above, the term $\Sigma_{\mathbf{x}_{ij}}$ is the covariance matrix of the measurement vector $\mathbf{x}_{ij}$ (which contains the pre-estimated 3D-point coordinates and the camera parameters). In case we do not know the covariance, we simply assume $\Sigma_{\mathbf{x}_{ij}}$ to be the identity matrix. However, how can we retrieve a covariance matrix for the estimated camera and structure parameters from the sparse bundle adjustment algorithm? To do so, we have to adapt the algorithm slightly, as proposed in (Hartley and Zisserman, 2004). Above, we defined $Y_{ij} = W_{ij}V_i^{*-1}$. If we want to calculate the covariance of SBA's results, we have to redefine $\hat{Y}_{ij} = W_{ij}V_i^{-1}$. Notice that we replaced V_i^{*-1} by V_i^{-1}. So we use the unaugmented version of V. Now we define $\hat{S}$ using the same formula as above, but using $\hat{Y}$ instead of Y and adding U_j instead of U_j^*. So again we use the unaugmented U (Remember that we augmented U and V by adding μ to the diagonal entries.).
Now the covariance for the estimated camera parameters can be defined as $\Sigma_{\mathbf{a}} = \hat{S}^+$ where $\hat{S}^+$ is the pseudo-inverse of $\hat{S}$.

6. Integrating Sparse Bundle Adjustment

During the past sections of this paper we described several algorithms that are used in our stereo odometry implementation. We saw how we can use the SVD-based algorithm of (Umeyama, 1991) for estimating the motion between two frames and how this algorithm can be used as a hypothesis generator in a robust estimation framework like RANSAC. Bundle Adjustment and its efficient implementation from (Lourakis and Argyros, 2004) (SBA) is then used to refine the robust motion estimates. We did not, however, explain how we integrate SBA into the rest of the algorithms. In general, we can use SBA in two different ways:

6.1 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 an image into the window, if its pose is more than a certain threshold away from the pose of its 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.

6.2 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 it moves along.

7. Our Algorithm at a Glance

Now that we laid out details of the several steps of our stereo odometry algorithm, we want to summarize things before we look at experimental results in the next section.

1. Acquire stereo images and 3D information from the stereo camera.
2. Find interest points in the left image by using the Harris operator.
3. Move.
4. Again, get a stereo image and find interest points in it.
5. Match the interest points that were found in steps 2 and 4.
6. Enter the RANSAC loop, using the SVD-based algorithm for generating hyoptheses about the motion. RANSAC will return a motion estimate and a set of inliers. Outliers (mainly false matches) will be removed.
7. Use sparse Bundle Adjustment over the last n frames to refine the motion estimated by RANSAC
8. repeat

8. Experimental Results

We already showed that sparse bundle adjustment is a feasible method for motion estimation on mobile robots in outdoor terrain in earlier work (Sünderhauf et al., 2005), but want to repeat the results here for the sake of completeness.

8.1 Real-World Data

In this early work, we did not use RANSAC for removing outliers in the data, but a simple and naive iterative method. However, only a little number of false matches were contained in the data, because a very sophisticated matching algorithm was used (Jung, 2004).
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. Max Bajracharya from JPL provided 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 results for different parameters settings are shown in figure 8.1. The average deviation from the ground truth position for all tested methods was approximately 23 cm (2.3 % of travelled 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 successfully tracked) in a feasible distance from the robot to compare the different window sizes and motion thresholds.

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.

8.2 Experimental Results from Simulation

The algorithm proposed in this work has not been tested with real image data yet, so we have to rely on simulated results.
The simulation maintains a world consisting of interest points only. A freely parameterizable stereo camera is moved through this point world following arbitrary paths. The simulated images (i.e. the projections of the world-points) can be disturbed by noise and then saved to a file and used for processing in the stereo odometry framework.
So by using simulated data, we skip the steps of finding and matching interest points. We can still use the simulated data to evaluate the functionality of motion

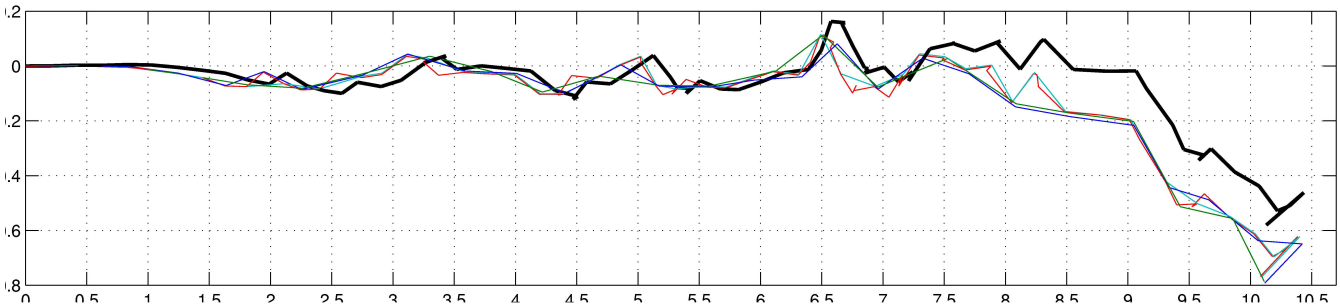

Figure 3: Several estimated trajectories for the outdoor dataset. The thick black line is the ground truth.

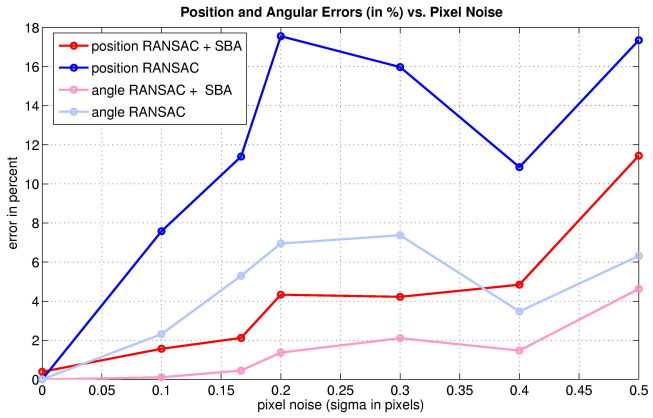

Figure 4: Simulated position and orientation error (in percent) of RANSAC and SBA vs. increasing image noise.

estimation by the RANSAC-SVD-algorithm scheme and sparse bundle adjustment. As these were the mainly focused parts during this work, using simulated data is not such a big drawback at all. Of course, the whole process will be tested and evaluated with real-world data during future work.

The diagram in fig. 4 shows the relative errors in the estimated position and orientation with rising noise in the simulated images. What can be read from this? The refinement by SBA clearly pays off, as the error of the combined RANSAC + SBA approach is well below the RANSAC-only error. So once there is some noise present in the image data (which will always be the case in practice), SBA is a clear improvement compared to the RANSAC-only algorithm.

As usual, we have to consider how good reality can be represented by the simulation. The standard deviation of the simulated noise rises from 0 to 0.5 pixels. What value can we expect in practice? This is mainly dependent on the interest point detection and matching scheme we use. We expect the standard deviation sigma of the interest point detection process to be between 1/6 and 1/3 pixel. The relative errors in percent at this noise level are approximately 4% for the position and 2% for the orientation. These results are good and in the order of magnitude of the results of other approaches and results cited before. Still, these values have to be confirmed with real data.

Our goal is to implement the stereo odometry process on the robots and use it online, in real time. Most time is spent with interest point matching and outlier removal (RANSAC). The SBA estimation then runs relatively quickly in approximately 100ms. So the biggest potential for optimization is clearly the interest point matching step. If we can improve this part of the software, we do not only accelerate the matching process itself, better matches lead to better initial estimates and a quicker outlier removal.

9. Conclusions and Further Work

We presented our approach towards a robust, efficient and yet accurate stereo odometry algorithm that can be used online on autonomous outdoor robots. Results from earlier work already proved sparse bundle adjustment to be a feasible method for visual motion estimation. The robust estimation scheme using RANSAC allows the use of a fast and simple interest point matching algorithm.

In future work our approach will be tested with real word data to further prove its use. Ideas for further improvements include the use of other robust estimation schemes like MSAC or MLESAC that may be superior to RANSAC. A quantitative discussion about the tradeoff between quick (but dirty) interest points matching and computationally rather expensive robust estimation methods is for sure worth doing in the future.

Stereo odometry should also be fused with other motion-estimating methods using an Extended Kalman Filter. It is therefore necessary to acquire the covariance matrix of SBA's results. We showed how to do this and will implement this approach as well.

Finally, the feature detection and selection process itself could be modified. For instance the methods presented in (Lowe, 2004) could improve feature quality significantly. Again, the tradeoff between this more

expensive feature point finding and matching strategy has to be discussed.

References

Arun, K., Huang, T., and Blostein, S. (1987). Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700.

Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and der Vorst, H. V. (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA.

C. Harris, M. S. (1988). A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference 1988*, pages 147–151.

Cheng, Y., Maimone, M. W., and Matthies, L. (2006). Visual odometry on the mars exploration rovers. *IEEE Robotics and Automation Magazine*, 13(2):54–62.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

Horn, B. K. P. (1987a). Closed–form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America*, 5(7):1127–1135.

Horn, B. K. P. (1987b). Closed–form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4):629–642.

Huang, T. S., Blostein, S. D., and Margerum, E. A. (1986). Least-squares estimation of motion parameters from 3–d point correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, Florida, 1986*, pages 24–26. IEEE Computer Society Press.

Jung, I.-K. (2004). *SLAM in 3D Environments with Stereovision*. PhD thesis, LAAS, Toulouse.

Lemaire, T. and Lacroix, S. (2006). Vision-based slam: Stereo and monocular approaches. Technical report, LAAS-CNRS. submitted to IJCV/IJRR special joint issue.

Lourakis, M. and Argyros, A. (2004). 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. Available from `http://www.ics.forth.gr/~lourakis/sba`.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. In *International Journal of Computer Vision, 60, 2*, pages 91–110.

Madsen, K., Nielsen, H. B., and Tingleff, O. (2004). Methods for non-linear least squares problems, 2nd edition.

Nister, D. (2003). Preemptive ransac for live structure and motion estimation. In *Proceedings of the 9th International Conference on Computer Vision, Nice*.

Nister, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, pages 652–659.

Schmid, C., Mohr, R., and Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172.

Solà, J., Devy, M., Monin, A., and Lemaire, T. (2005). Undelayed initialization in bearing only slam. In *IEEE International Conference on Intelligent Robots and Systems*.

Sünderhauf, N., Konolige, K., Lacroix, S., and Protzel, P. (2005). Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle. In Levi, Schanz, Lafrenz, and Avrutin, (Eds.), *Tagungsband Autonome Mobile Systeme 2005*, Reihe Informatik aktuell, pages 157–163. Springer-Verlag.

Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (2000). Bundle adjustment – A modern synthesis. In Triggs, W., Zisserman, A., and Szeliski, R., (Eds.), *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag.

Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380.