Smoothing-Based Submap Merging in Large Area SLAM

Anders Karlsson, Jon Bjärkefur, Joakim Rydell, and Christina Grönwall

Swedish Defence Research Agency
Linköping, Sweden
http://www.foi.se/

Abstract. This paper concerns simultaneous localization and mapping (SLAM) of large areas. In SLAM the map creation is based on identified landmarks in the environment. When mapping large areas a vast number of landmarks have to be treated, which usually is very time consuming. A common way to reduce the computational complexity is to divide the visited area into submaps, each with a limited number of landmarks. This paper presents a novel method for merging conditionally independent submaps (generated using e.g. EKF-SLAM) by the use of smoothing. By this approach it is possible to build large maps in close to linear time. The approach is demonstrated in two indoor scenarios, where data was collected with a trolley-mounted stereo vision camera.

1 Introduction

Simultaneous localization and mapping (SLAM) refers to techniques for estimating the trajectory along which a person or robot moves in an unknown environment, while also creating a map of the surrounding area. Many SLAM methods are based on observing and recognizing landmarks present in the environment, and the created map consists of the estimated positions of these landmarks. Sensors commonly used for SLAM include visual cameras, laser range finders, sonar and radar.

Apart from feature extraction and data association, a successful SLAM implementation needs to tackle three major issues; loop closing, large numbers of landmarks, and consistency. Loop closing shall occur when the sensor revisits a previously seen area and the result should be an updated and improved map. In the loop closure the current state of the sensor is also updated. The SLAM algorithm must be able to handle large maps, i.e., a large number of landmarks. The major problem with many landmarks is that they are usually time consuming to calculate. Executions times proportional to n^2, where n is the number of landmarks, are common. Furthermore, the resulting map needs to be consistent, i.e., correct. Inconsistency is usually related to linearization problems, which cause the uncertainty about landmark positions and sensor position and orientation to be underestimated.

The submap SLAM approach that is presented in this paper handles loop closing, can treat large sets of landmarks in close to linear time, and is consistent. Examples using stereo camera data is shown.

A. Heyden and F. Kahl (Eds.): SCIA 2011, LNCS 6688, pp. 134–145, 2011.

2 Background and Related Work

2.1 SLAM Algorithms

One of the most well-known SLAM methods is EKF-SLAM, which, as the name implies, is based on the extended Kalman filter. The filter state $\mathbf{X}$ contains all information about the sensor pose as well as the position of all landmarks in the map. Assuming that the landmarks are stationary and that a constant velocity and angular velocity model is used for the sensor motion,

$$\mathbf{X} = (\mathbf{t}, \dot{\mathbf{t}}, \mathbf{r}, \dot{\mathbf{r}}, \mathbf{l}_1, \ldots, \mathbf{l}_n)^T,$$

where $\mathbf{t}$ and $\dot{\mathbf{t}}$ denote the sensor position and velocity, respectively, $\mathbf{r}$ and $\dot{\mathbf{r}}$ denote the orientation and angular velocity (using e.g. Euler angles or a quaternion representation), $\mathbf{l}_i$ denotes the ith landmark position and n is the number of landmarks in the map. Obviously, the state dimensionality grows with the number of landmarks. The dimensionality of $\mathbf{X}$ is $12 + 3n$ if the orientation and angular velocity are represented using Euler angles and landmarks are represented using their coordinates in $\mathbb{R}^3$.

Since the EKF explicitly models the covariance between all landmarks, the SLAM algorithm does not need additional logic to handle loop closures. However, due to linearization errors EKF-SLAM tends to underestimate the covariances. This may cause inconsistent estimates of the trajectory, and may preclude successful loop closures. Another disadvantage of EKF-SLAM is that the covariance matrix grows with the square of the number of landmarks in the map. This quickly causes the computational complexity to become prohibitively large.

A number of alternative algorithms, which address one or both of these disadvantages, have been proposed. Information form methods use the information matrix instead of the covariance matrix. The information matrix is "almost sparse", i.e., many elements in this matrix are very close to zero. By removing weak links between landmarks in a controlled manner, the information matrix is made sparse, which improves the computational performance. This sparsification can be performed in several ways. In the Sparse Extended Information Filter (SEIF) [12] the approximation causes the method to produce overconfident estimates (worse than EKF-SLAM), while the Exactly Sparse Extended Information Filter (ESEIF) [13] breaks links in such a way that more conservative estimates are obtained. However, in these methods the landmark covariances are not immediately available, which makes data association more computationally expensive.

Another SLAM method is FastSLAM [7] (or FastSLAM 2.0 [8]), where the sensor pose uncertainty is represented by a particle filter. Since the landmark positions are conditionally independent given the sensor pose, and since each particle represents one (completely certain) pose hypothesis, the covariance between different landmark positions is zero in this formulation. Hence the uncertainty about each landmark can be modeled by a separate covariance matrix of size 3×3, thereby elimininating the quadratic growth rate. On the other hand, a large number of particles may be needed.

2.2 Smoothing and Mapping

The algorithms mentioned above, like most other SLAM algorithms, perform *filtering*, i.e., compute the new sensor pose and updates the map whenever a new measurement is available, but retain the previous sensor pose trajectory without modification. This is obviously suboptimal. In the Smoothing and Mapping (SAM) [3] algorithm, *smoothing* is used instead of filtering. This essentially means that the entire map and sensor pose trajectory are computed using all measurements. Hence, all parameters are recomputed whenever a new measurement becomes available.

SAM solves the least squares problem given by

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}} \left\{ \sum_{i=1}^{N} \|f_i(\mathbf{X}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i\|^2_{\mathbf{\Lambda}_i} + \sum_{k=1}^{K} \|h_k(\mathbf{X}_{i_k}, \mathbf{l}_{j_k}) - \mathbf{z}_k\|^2_{\mathbf{\Sigma}_k} \right\},$$

where $\mathbf{\Theta}^*$ is a parameter vector containing the trajectory and map information, K and N are the number of observations and positions along the trajectory, respectively, and $\mathbf{\Lambda}$ and $\mathbf{\Sigma}$ are the process and measurement noise covariances. f and h are functions defining the the process and measurement models. Further, $\mathbf{z}$ represents actual measurements and $\mathbf{u}$ is the odometry information (the estimated motion in the submap).

By linearizing f and h, $\mathbf{\Theta}^*$ can be obtained by solving a standard linear least squares problem of the form

$$\delta^* = \arg\min_{\delta} \|\mathbf{A}\delta - \mathbf{b}\|^2.$$

Here δ^* is the optimal adjustment of the linearization point $\mathbf{\Theta}$ (the previous estimate). $\mathbf{A}$ contains Jacobians of the process and measurement model, evaluated at the current linearization point, while $\mathbf{b}$ contains the odometry and measurement prediction errors. The details of the SAM algorithm, including how to construct $\mathbf{A}$ and $\mathbf{b}$ are omitted here in order to conserve space; a very nice presentation is available in [3]. Nevertheless, two important points should be noted here: 1) The system of equations rapidly becomes very large. However, $\mathbf{A}$ is sparse since most landmarks are only observed along a short part of the sensor trajectory. By rearranging the columns of $\mathbf{A}$ (and the rows of δ^*) this sparsity can be exploited, greatly reducing the computational complexity of the algorithm. 2) Since the entire trajectory and map are updated and the problem is relinearized in every iteration using the current state estimate, SAM does not suffer from inconsistent solutions due to linearization errors.

2.3 Submaps

One way to cope with rapidly growing computational complexity of SLAM methods is to divide the environment into smaller areas, or *submaps*. In addition to reducing the computational cost, submapping techniques may also improve the

consistency of the globally estimated map [1]. In Conditionally Independent Divide and Conquer SLAM [11] a chain of submaps is created, and the submaps are merged by back-propagating information. Loop closure is performed by adding landmarks common to the current and previous submap at the loop closure location. Conditionally Independent Graph SLAM [10] is similar, but maintains a spanning tree making it possible to transmit information between submaps. Tectonic SAM [9] is based on the smoothing and mapping concept described above, and is similar to the method presented in this paper. However, Tectonic SAM utilizes all observations of landmarks which are shared between different submaps. Also, Tectonic SAM creates the submaps *and* merges them using SAM-like approaches.

3 Method

As stated in the previous section, EKF-SLAM works relatively well in very small environments, but produces inconsistent maps and is computationally expensive when the map grows and linearization errors arise. SAM, on the other hand, does not suffer from inconsistency problems. However, its computational complexity grows rapidly with the number of landmark measurements and with the number of poses in the sensor trajectory. We therefore propose a method where small submaps are created using EKF-SLAM and merged using an approach similar to Tectonic SAM. Compared to Tectonic SAM, our approach uses only one measurement per submap, creates the submaps using EKF-SLAM and merges them using smoothing. In the merging step, each submap is considered just one measurement and one position along the sensor trajectory. Hence, in the SAM step, only a skeleton of the entire trajectory is considered. This submapping approach reduces the computational complexities of both the EKF-SLAM and the SAM steps, while also alleviating the consistency issues of EKF-SLAM.

Section 3.1 presents our approach for creating conditionally independent submaps using EKF-SLAM. Section 3.2 describes the merging process, which we refer to as Submap Joining Smooting and Mapping (SJSAM).

3.1 Conditionally Independent Submaps

Conditionally independent submaps are created with EKF-SLAM in a way very similar to [11]. A new submap is initiated when the sensor has moved a user-specified distance from the start of the current submap. Each submap is locally referenced, i.e., the sensor position is set to the origin and the orientation to a default rotation matrix. The covariance for the new pose is zero. The velocity of the sensor is initiated with the velocity of the sensor in the previous submap, rotated to compensate for resetting the sensor orientation.

The landmarks that were matched in the last step of the previous submap are copied to the new submap and used as its initial landmarks. Their positions are also compensated for the orientation reset. The covariance of the landmark

positions is calculated from the previous submap by marginalization of the previous uncertainty about sensor pose:

$$\mathbf{P}_{l,l} = \mathbf{P}_{l,l} - \mathbf{P}_{l,c}\mathbf{P}_{c,c}^{-1}\mathbf{P}_{c,l},$$

where $\mathbf{P}_{l,l}$ is the covariance matrix for all landmark positions in the submap, $\mathbf{P}_{c,c}$ is the covariance of the sensor pose and $\mathbf{P}_{l,c} = \mathbf{P}_{c,l}^T$ is the cross covariance. The resulting covariance matrix is finally rotated to compensate for the orientation reset within the new submap.

3.2 Submap Merging

In the original SAM approach, the state vector contains all sensor poses along the trajectory. This means that the state vector grows with time, even if no new landmarks are observed. Additionally, all landmark observations are stored in the matrix $\mathbf{A}$, which defines the equation system to be solved in each iteration. If 20 3D landmarks are observed in every frame, a frame rate of 10 Hz corresponds to $20 \times 3 \times 10 = 600$ additional rows in $\mathbf{A}$ every second. While these rows are sparse, this still causes the computational complexity to increase rapidly, particularly since each observation increases the relinearization workload in the algorithm.

In SJSAM, actual landmark observations are not used within the SAM framework. Instead, these observations are used to create submaps using EKF-SLAM. Submaps are then merged using SAM, treating each submap as just *one combined measurement*. Similarly, instead of considering each sensor pose along the trajectory, the entire estimated motion within each submap is treated as *one sample* of odometry data to be used in SAM. This approach results in that only a skeleton of the sensor trajectory is maintained in the SAM state vector. Hence both the dimensionality of the state vector in SAM and the number of equations corresponding to measurements are reduced significantly.

Measurement equation. In SJSAM, the landmarks of a submap are considered relative measurements from the origin of the submap. Each measurement consists of the relative displacement between the landmark and the sensor in three dimensions (Δx, Δy and Δz). The measurement equation is

$$\mathbf{z} = (\Delta x \quad \Delta y \quad \Delta z)^T = h(\mathbf{x}_c, \mathbf{l}) = \mathbf{T}(\mathbf{l} - \mathbf{x}_c)$$

where $\mathbf{z}$ is the measurement, $\mathbf{x}_c$ is the sensor pose (in this context: the global coordinates of the first position in the current submap), $\mathbf{l}$ is the position of the observed landmark (also in global coordinates) and $\mathbf{T}$ is a rotation matrix transforming from global to submap-specific coordinates. The covariances of the landmarks in the submaps are used as measurement noise in SJSAM.

System dynamics. The last position in each submap is considered as odometry information in the prediction part of the SJSAM merging algorithm. This gives an initial estimate of the sensor state and the linearization points needed in

SAM. The equation for predicting a new sensor pose given the previous pose estimate and the odometry information is the following:

$$\mathbf{x}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \\ \phi_i \\ \theta_i \\ \psi_i \end{pmatrix} = f(\mathbf{x}_{i-1}, \mathbf{u}_{i-1}) = \begin{pmatrix} x_{i-1} \\ y_{i-1} \\ z_{i-1} \\ \phi_{i-1} \\ \theta_{i-1} \\ \psi_{i-1} \end{pmatrix} + \begin{pmatrix} \mathbf{T^{-1}} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} \\ \Delta\phi \\ \Delta\theta \\ \Delta\psi \end{pmatrix},$$

where the pose of the robot in the end of a locally referenced submap is Δx Δy Δz $\Delta\phi$ $\Delta\theta$ $\Delta\psi$, i.e., the odometry input given by the final pose in the submap. The covariance of the final robot pose in the submap is used as process noise in SJSAM.

Algorithm. For each submap, a number of steps are carried out, see Algoritm 1. First, the linearization points corresponding to the estimates of the sensor state in sample i and $i-1$ are retrieved. The linearization point in $i-1$ is found in the last rows of the state vector. The linearization point for i is calculated using the process equation using the state in $i-1$ and the odometry information of the previous submap.

The second step is data association. The measurements in a sample correspond to the landmarks of one submap. For reasons of computational complexity, the data association is performed using nearest neighbor calculation with Euclidean distances. For each landmark a SURF descriptor [2], representing the appearance of the surrounding area in the image, is stored. This descriptor is used as a validity check in the data association.

In the third step the measurement matrix $\mathbf{A}$ is augmented with three rows for each landmark in the submap. The odometry data is also appended to $\mathbf{A}$ and the new data about measurement and odometry prediction errors is appended to the vector $\mathbf{b}$. Complete details of how to structure $\mathbf{A}$ and $\mathbf{b}$ are available in [3].

SJSAM will incrementally find better estimates of the linearization points used in the past. Hence $\mathbf{A}$, which depends on the linearization points, needs to be recalculated once in a while. This is called relinearization and is performed whenever data from a new submap (odometry and measurements) is processed by the SAM framework. During relinearization the entire $\mathbf{A}$ matrix and the vector $\mathbf{b}$ are traversed.

SJSAM then solves the least squares problem using Q-less QR-factorization with the sparse $\mathbf{A}$ matrix and the vector $\mathbf{b}$. Solving the resulting QR-factorization gives δ^* with information of how much to change each variable in the current state vector $\mathbf{\Theta}$ (see Section 2.2).

The exact sensor state covariance estimates can be retrieved efficiently from the QR-factor R using back-substitution [6]. Conservative estimates of the landmark covariances can be retrieved efficiently using the sensor covariance and the measurement noise used when the landmark was initiated. Exact estimates are more costly to access.

Algorithm 1. Submap Joining Smoothing and Mapping (SJSAM)

Data: Submaps $\mathbf{s}_i = (\mathbf{x}_i \ \mathcal{L}_i)^T \in \mathcal{S}$ created with e.g. EKF-SLAM. $\mathbf{x}_i$ is the last state of the camera in submap i, and $\mathcal{L}_i$ is the landmarks in submap i
Result: A state vector $\mathbf{\Theta}^* = (\hat{\mathbf{x}} \ \hat{\mathbf{m}})^T$, where $\hat{\mathbf{x}}$ is a skeleton trajectory and $\hat{\mathbf{m}}$ is the map

begin
  forall *submaps* $\mathbf{s}_i \in \mathcal{S}$ **do**
    1. Retrieve the linearization points
    if $i = 1$ **then**
      Initiate $\hat{\mathbf{x}}_\mathbf{1}$ with zeros
    else
      $\hat{\mathbf{x}}_{i-1}$ is the last camera state estimate in $\mathbf{\Theta}$
      $\hat{\mathbf{x}}_i = f(\hat{\mathbf{x}}_{i-1}, \mathbf{u}_{i-1})$ where $\mathbf{u}_{i-1}$ is the odometry information from $\mathbf{s}_{i-1}$
    end
    2. Data association
    Find all $\mathbf{l}_j \in \mathcal{L}_i$ that match a previously seen landmark
    Find all $\mathbf{l}_n \in \mathcal{L}_i$ that have not been seen before
    3. Measurement update
    forall *reobservations* $\mathbf{l}_j$ **do**
      Compute measurement Jacobians with respect to landmark positions and camera state
      Add measurement Jacobians to $\mathbf{A}$ (3 rows)
      Add measurement prediction errors to $\mathbf{b}$ (3 rows)
    end
    forall *new observations* $\mathbf{l}_n$ **do**
      Compute an initial estimate for the landmark using the measurement
      Add initial estimate to $\mathbf{\Theta}$ (3 rows)
      Compute measurement Jacobians with respect to landmark positions and camera state
      Add measurement Jacobians to $\mathbf{A}$ (3 rows, 3 columns)
      Add measurement prediction errors to $\mathbf{b}$ (3 rows)
    end
    4. Time update
    if $i = 1$ **then**
      Add identity matrix to $\mathbf{A}$ (6 rows, 6 columns)
      Add process prediction errors to $\mathbf{b}$ (3 rows)
    else
      Compute the process model Jacobian
      Add Jacobian and identity matrix to $\mathbf{A}$ (6 rows, 6 columns)
      Add process model prediction errors to $\mathbf{b}$ (6 rows)
    end
    Add $\hat{\mathbf{x}}_\mathbf{i}$ to $\mathbf{\Theta}$ (6 rows)
    5. Relinearization
    Recompute all the elements of $\mathbf{A}$ and $\mathbf{b}$ that were created during sample $1 \ldots (i-1)$, using the new linearization points
    6. Solve the least squares problem
    Solve $\mathbf{A}\delta^* = \mathbf{b}$ using Q-less QR-factorization
    $\mathbf{\Theta} = \mathbf{\Theta} + \delta^*$
  end
end

Similar work. The Sparse Local Submap Joining Filter (SLSJF) [4] is similar to SJSAM in that it uses a hierarchical submap approach. SLSJF also stores each start/end pose of the submaps in the state vector. However, SLSJF uses an Extended Information Filter (EIF) representation for the submap merging instead of the SAM representation. SLSJF and I-SLSJF [5] retrieves the state vector from the information matrix by solving a least squares problem. This is similar to SJSAM. However, I-SLSJF only recomputes the measurement matrix and vector somtimes, while SJSAM performs this step in each sample. Additionally, I-SLSJF does not use a specific prediction step.

4 Examples

We have evaluated the idea with SJSAM on several data sets. Two of them will be presented here. Both sets are collected in indoor environments with a stereo camera mounted horizontally on a trolley. The trolley was moved at roughly constant speed throughout the data collections.

4.1 Implementation Details

Both the EKF-SLAM- and SJSAM-algorithm are implemented in MATLAB and are currently not capable of real-time processing. The main reason for this is the feature extraction. We divide the map creation process into four essential steps:

1. Data acquisition.
2. Feature extraction with SURF [2].
3. Creation of submaps using EKF-SLAM.
4. Merging of submaps with SJSAM.

There are currently two different methods for submap creation. The first method is to use the traveled distance as the bounding limit for each submap, e.g., a new submap is initiated when the camera has moved more than one meter since the start of the submap. The other method uses the number of landmarks the submap contains as the condition to start a new submap, e.g., a new submap is created when the current submap contains more than 100 landmarks. This later approach gives a more predictable time complexity but the traveled distance approach gives more evenly spread out submaps. In the experiments below we use a maximum travel distance of one meter for each submap, to test the algorithm's handling of several submaps.

4.2 Small Area Experiment

The first experiment was conducted in a small conference room. During this experiment the trolley moved a total path of approximately 15 m which gives a total of 15 submaps and 358 landmarks. Figure 1 shows the result from the submap generation, colored dots represent landmarks and black dots correspond to the trajectory. The black squares mark the start of each submap and those points will be used as input to the SJSAM algorithm.

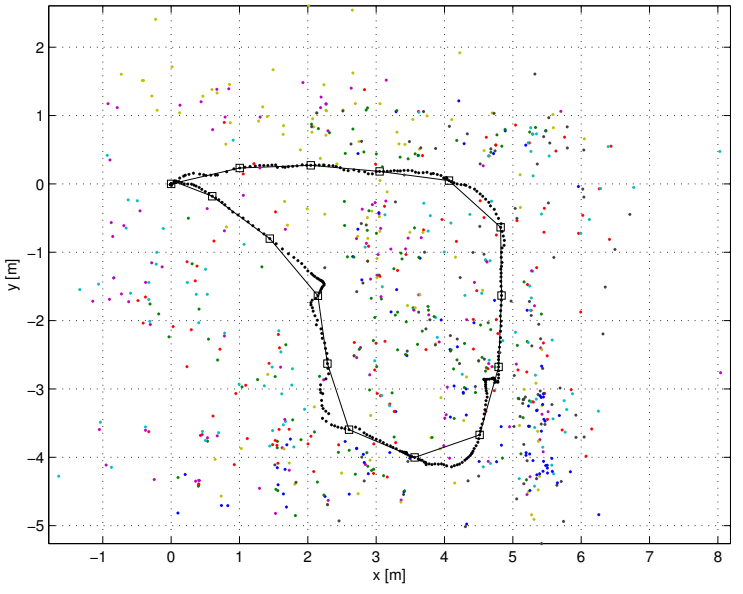

Fig. 1. Submaps from the small area.The start of each submap is marked with a square. The thin lines represent a skeleton map created only from the start of each submap. Axes in meters.

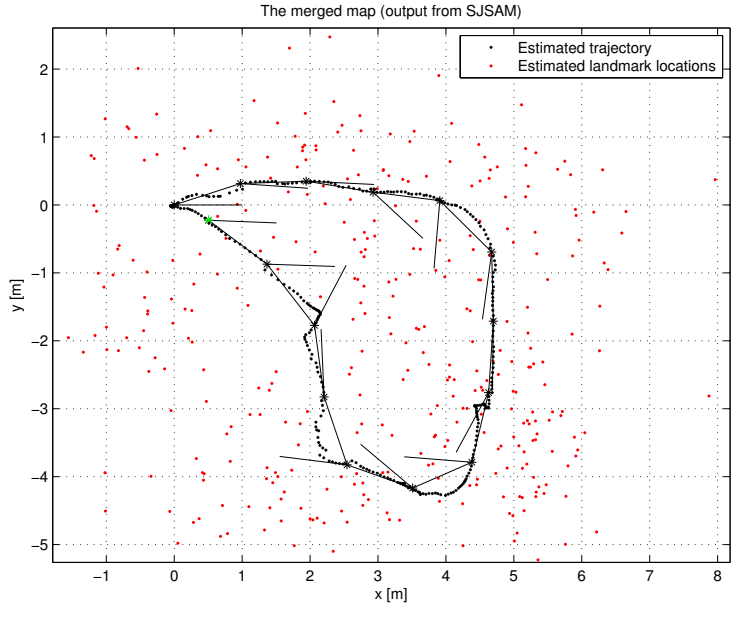

Fig. 2. Submaps from the small area processed with the SJSAM algorithm. The short lines represents the camera's orientation at the start of each submap. In total there are 358 landmarks in the map. Axes in meters.

Since this is a fairly small data set it could also be processed with standard EKF-SLAM for computational comparison. The total processing time for this data set with EKF-SLAM was about 6.5 s and with submaps the same data set can be processed in 3.8 s indicating that the use of submaps significantly reduces the computational complexity.

Processing the submaps with the SJSAM algorithm yields the map depicted in Figure 2. It is hard to see any significant improvement but the figure clearly shows that the algorithm has performed loop-closure and that the trolley has returned almost exactly to its original position.

4.3 Large Area Experiment

The second experiment was conducted in a dining room connected to a conference room by a normal doorway, a total path of approximately 34 meters. The trolley was moved through the dining room, into the conference room, and then back to the initial position. This results in a trajectory that is eight-shaped.

Figure 3 shows the resulting trajectory using EKF-SLAM. The figure shows that the trolley not has returned to its original position. This kind of behavior occurs due to drift and the lack of loop-closure between the submabs. The complete map consists of 34 submaps and the total processing time is about 26 s.

The previously created submaps can be processed by SJSAM algorithm. By sequential merging of the submaps and re-linearization of the whole problem in each sample we get the map presented in Figure 4, left. This map consists of 1275 landmarks and the merging time is 3 seconds. It can be seen from the figure that loop closure has occurred. This can also be seen in the sparse measurement

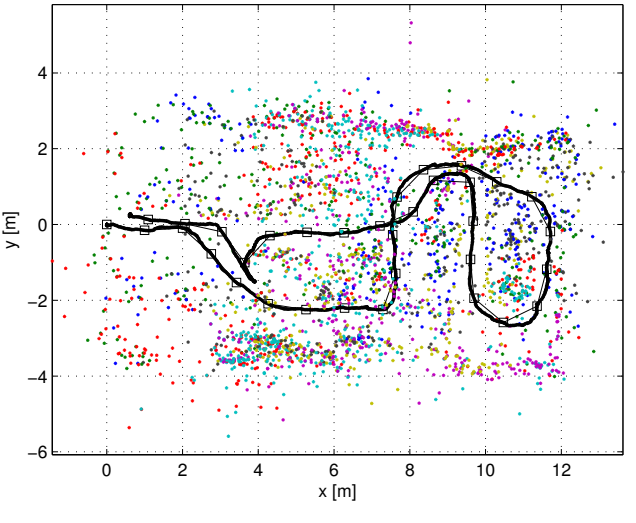

Fig. 3. The large area experiment, the trajectory consisting of 34 submaps. Processed using EKF-SLAM. Axes in meters.

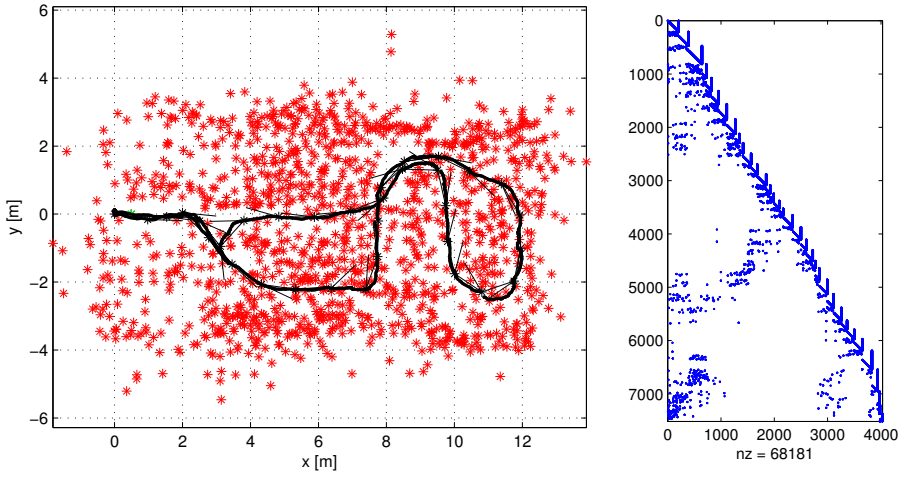

Fig. 4. The large area experiment. Left: the trajectory after merging the 34 submaps using SJSAM. Axes in meters. Right: the sparse measurement matrix **A** illustrating loop closure. The dots illustrates non-zero elements in **A**. Axes represents the position of the elements in the matrix.

matrix **A** in Figure 4, right, by looking at the lower triangular part of the matrix: elements that are non-zero and not close to the diagonal represent landmarks that have been seen before and recognized.

5 Conclusions

We have presented an approach for merging of submaps using smoothing and mapping. The approach is called Submap Joining Smoothing and Mapping.

The approach is illustrated in two examples using data from a stereo vision camera. In both examples we can see that a submap-based approach is successful. The generation of submabs is done in constant time in the number of submaps and merging with SJSAM is done in almost linear time. This is in a relatively low time complexity compared to standard EKF-SLAM, which makes this method suitable for intermediate sized data sets.In the experiment with a larger data set (large area experiment) SJSAM yields a significant improvement. SJSAM is able to perform loop-closure and the resulting map is a large improvement compared to using only submaps.

This paper shows our first results with SJSAM and in the future the performance need to be evaluated. The optimal length of submaps needs to be studied and SJSAM needs to be compared with similar SLAM methods. In the current implementation data association is performed with nearest neighbor. Extending the algorithm with data association that uses e.g. the Mahalanobis distance may lead to improved performance and less sensitivity to drift. It may also be necessary to use picture comparison techniques, such as tree-of-words.

References

1. Bailey, T., Nieto, J., Guivant, J., Stevens, M., Nebot, E.: Consistency of the EKF-SLAM algorithm. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3562–3568 (2006)
2. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
3. Dellaert, F., Kaess, M.l.: Square root sam: Simultaneous localization and mapping via square root information smoothing. Int. J. Rob. Res. 25(12), 1181–1203 (2006)
4. Huang, S., Wang, Z., Dissanayake, G.: Sparse local submap joining filter for building large-scale maps. IEEE Transactions on Robotics 24(5), 1121–1130 (2008)
5. Huang, S., Wang, Z., Dissanayake, G., Frese, U.: Iterated SLSJF: A sparse local submap joining algorithm with improved consistency. In: 2008 Australiasan Conference on Robotics and Automation, Citeseer (2008)
6. Kaess, M., Ranganathan, A., Dellaert, F.: iSAM: Incremental smoothing and mapping. IEEE Transactions on Robotics 24(6), 1365–1378 (2008)
7. Montemerlo, M.: FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (July 2003)
8. Montemerlo, M.l., Thrun, S., Roller, D., Wegbreit, B.: Fastslam 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: Proceedings of the 18th International joint Conference on Artificial Intelligence, San Francisco, CA, USA, pp. 1151–1156 (2003)
9. Ni, K., Steedly, D., Dellaer F.: Tectonic sam: exact, out-of-core, submap-based slam. In: Proc. IEEE International Conference on Robotics and Automation, pp. 1678–1685 (2007)
10. Piniés, P., Paz, L.M., Tardós, J.D.: CI-Graph: An efficient approach for Large Scale SLAM. In: IEEE International Conference on Robotics and Automation (2009)
11. Piniés, P., Tardós, J.D.: Large-scale slam building conditionally independent local maps: Application to monocular vision. IEEE Transactions on Robotics 24(5), 1094–1106 (2008)
12. Thrun, S., Liu, Y., Koller, D., Ng, A.Y., Ghahramani, Z., Durrant-Whyte, H.: Simultaneous Localization and Mapping with Sparse Extended Information Filters. The International Journal of Robotics Research 23(7-8), 693–716 (2004)
13. Walter, M.R., Eustice, R.M., Leonard, J.J.: Exactly sparse extended information filters for feature-based SLAM. The International Journal of Robotics Research 26(4), 335 (2007)