Tectonic SAM: Exact, Out-of-Core, Submap-Based SLAM

Kai Ni*, Drew Steedly†, and Frank Dellaert*
*College of Computing, Georgia Institute of Technology, Atlanta, GA 30332
†Microsoft Live Labs, Redmond, WA 98052
{nikai,dellaert}@cc.gatech.edu, steedly@microsoft.com

***Abstract*— Simultaneous localization and mapping (SLAM) is a method that robots use to explore, navigate, and map an unknown environment. However, this method poses inherent problems with regard to cost and time. To lower computation costs, smoothing and mapping (SAM) approaches have shown some promise, and they also provide more accurate solutions than filtering approaches in realistic scenarios. However, in SAM approaches, updating the linearization is still the most time-consuming step. To mitigate this problem, we propose a submap-based approach, *Tectonic SAM*, in which the original optimization problem is solved by using a divide-and-conquer scheme. Submaps are optimized independently and parameterized relative to a *local coordinate frame*. During the optimization, the global position of the submap may change dramatically, but the positions of the nodes in the submap relative to the local coordinate frame do not change very much. The key contribution of this paper is to show that the linearization of the submaps can be cached and reused when they are combined into a global map. According to the results of both simulation and real experiments, Tectonic SAM drastically speeds up SAM in very large environments while still maintaining its global accuracy.**

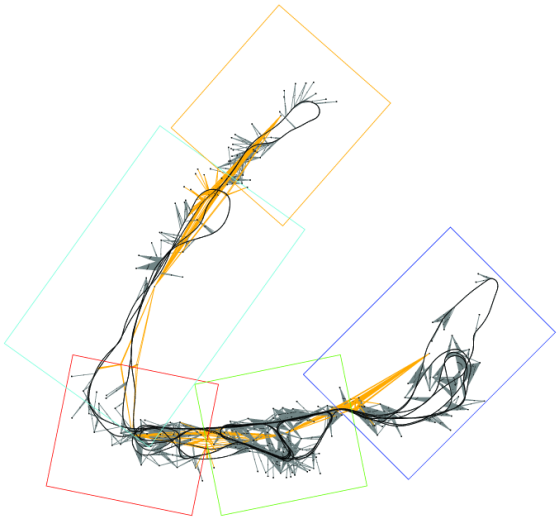


Fig. 1. The optimized Victoria Park data set with 153 landmarks, 6969 robot poses, and 3626 bearing-range measurements.

I. Introduction

Simultaneous localization and mapping (SLAM) is a method to help robots explore, navigate, and map an unknown environment [22], [24]. It is well known that traditional methods for SLAM based on the extended Kalman filter (EKF) suffer computational complexity problems when dealing with large-scale environments, as well as inconsistencies for non-linear SLAM problems [16].

Because of this tremendous computation, growing interest has been noticed in *smoothing* approaches. A typical approach is *simultaneous smoothing and mapping* (SAM) [5], also called *full SLAM* [25]. Instead of considering only robot trajectories [3], [21], [18], SAM attempts to estimate the entire set of robot poses and features in the environment. It has the advantage of providing higher quality solutions often at lower cost [24], [26]. It has already been demonstrated that SAM can be used in real-time [17]. SAM is also the key technology used in structure from motion (SFM), e.g., 3D photo tourism. In this paper, we will propose a new algorithm on the basis of SAM that enhances the performance by solving the large SLAM problems in a divide-and-conquer manner.

In many applications of SLAM, the artifact of interest is the map itself, e.g., for urban reconstruction, search-and-rescue operations and battlefield reconnaissance. In order to map large-scale environments, we can often quite accurately reconstruct parts of the world without necessarily knowing the exact metric relationships with other parts of the world. Although a canonical example for this intuitively appealing idea is offices and meeting rooms in an indoor office environment, the idea applies equally well to large-scale outdoor environments such as cities.

In this paper, we introduce a new approach to map large-scale environments. The basic idea of our algorithm is to maintain a set of *submaps* whose internal structures are relatively well known but whose relative poses are relatively fluid and can move with respect to one another. We name the algorithm *Tectonic SAM* (T-SAM) because of the analogy to the geological phenomenon of plate tectonics.

Many authors have advocated the use of submaps [19], [23], [27], [2], [8], [9], [6], [12], but all make approximations so that the final product does not use all of the available information. An exception is the compressed filter [14], which uses a submap around the current position of the robot but then periodically integrates it into a global map. However,

even the compressed filter yields inconsistent maps as it is still a filtering paradigm [16]. In contrast to those approaches, Tectonic SAM will yield the *exact* solution under mild assumptions. Given that we are *smoothing* rather than filtering, and under the assumption that the variables in the submap are sufficiently constrained by local measurements, our approach will yield the exact maximum likelihood solution for even very large full SLAM problems.

As stated in [5], linearization is the most time-consuming task in the entire smoothing process. Due to the nonlinear nature of typical SLAM problems, smoothing approaches have to relinearize the measurement equations and re-factorize at every iteration. Tectonic SAM solves this problem by introducing *base nodes* to capture the global positions of submaps. The algorithm first solves each submap locally by updating the linearization using the new estimation in each iteration. Then a global alignment follows to compute the relative location of the submaps. Given the fact that the submaps will only slightly change after incorporating all local constraints, the linearization points of the submap variables can be fixed in this stage. As a result, most linearization calculations need to be done only in local submaps while they can be kept constant after that.

The closest related efforts to our approach are the star-node idea in [9] and the tree-map algorithm by Frese [10], [11]. The first graphical SLAM approach in [9] includes a mechanism that compresses the graph in a way very similar to what we will propose below. However, their approach is based on EKF and requires additional tweaking to handle special cases, e.g., loop closing. The second approach [11], [12] uses a binary-tree to perform integration and marginalization, leading to an impressive algorithm that has the ability to deal with very large maps. However, multiple approximations are employed to reduce the complexity, while our approach solves the full and exact problem and therefore allows relinearization of all variables at any time. In [13], once variables are combined in a node, they cannot be separated again. In this way, several different linearization points are propagated in different factors, leading to inconsistency of the final result.

Instead of employing a tree-based data structure, we use a k-way cut as a mechanism to guide computation without making any representational compromises. We made this choice because the error on the linearization points is mostly limited in this case, which will be explained in detail later. In addition, our approach is more generally stated in terms of graphs and their associated matrices: we do not differentiate between landmarks and poses. Unlike most other approaches, our approach maintains local maps explicitly, which is straightforward and crucial in various applications.

II. Smoothing and Mapping

A natural representation for the SLAM problem is a *factor graph*, a bipartite graph containing two types of nodes: unknowns, representing unknown landmarks and poses, and factors corresponding to landmark measurements and odometry. An example of factor graph is shown in Figure 2 and will be used throughout the paper to explain our approach. In this example we assume that, without loss of generality, the landmarks and bearing/range measurements are 2D. The algorithm can equally be applied in a SFM setting to cameras and 3D environments.

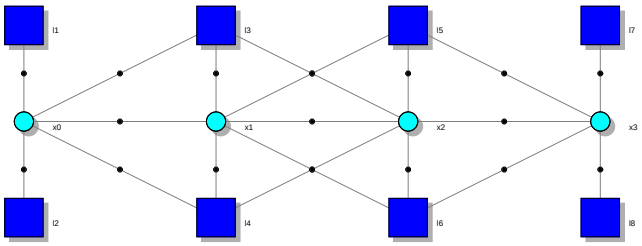


Fig. 2. The factor graph for a typical SLAM problem. The cyan circles are the robot trajectory and the blue squares are the landmarks in the environment. The filled black nodes represent the factors induced by landmark measurements and odometry.

The smoothing approach [7], [15] of optimizing over *all* unknown robot poses in addition to features in the environment is also called SAM in robotics and bundle adjustment in photogrammetry [26]. The advantage of this process is that it produces joint optimal parameters instead of the inexact ones obtained by a non-linear filtering scheme, which inevitably freezes incorrect linearization points. In particular, we seek the *maximum a posteriori* (MAP) estimate for the robot poses $X = \{x_i \mid i \in 0...M\}$ and the map $L = \{l_j \mid j \in 1...N\}$, given the measurements $Z = \{z_k \mid k \in 1...K\}$ and the control inputs $U = \{u_i \mid i \in 1...M\}$ (readers may refer to [5] for more detailed derivations). Under the assumption of independent, zero-mean, normally distributed measurement noise, the MAP estimate is the minimum of a non-linear least-squares criterion:

$$\sum_{i=1}^{M} \|f_i(x_{i-1}, u_i) - x_i\|^2_{\Lambda_i} + \sum_{k=1}^{K} \|h_k(x_{i_k}, l_{j_k}) - z_k\|^2_{\Sigma_k} \quad (1)$$

where $f_i(.)$ is a motion model with associated covariance Λ_i, and $h_k(.)$ a measurement model with associated covariance Σ_k. The notation $\|.\|^2_\Sigma$ stands for the squared Mahalanobis distance with covariance matrix Σ.

The terms in Equation 1 can be linearized as

$$\begin{aligned} & f_i(x_{i-1}, u_i) - x_i \\ \approx \quad & \left\{ f_i(x^0_{i-1}, u_i) + F^{i-1}_i \delta x_{i-1} \right\} - \left\{ x^0_i + \delta x_i \right\} \end{aligned} \quad (2)$$

and

$$\begin{aligned} & h_k(x_{i_k}, l_{j_k}) - z_k \\ \approx \quad & \left\{ h_k(x^0_{i_k}, l^0_{j_k}) + H^{i_k}_k \delta x_{i_k} + J^{j_k}_k \delta l_{j_k} \right\} - z_k \end{aligned} \quad (3)$$

where F^{i-1}_i is the Jacobian of $f_i(.)$ evaluated at x^0_{i-1} and $H^{i_k}_k, J^{j_k}_k$ are the Jacobians of $h_k(.)$ with respect to x_{i_k}, l_{j_k} and evaluated at $(x^0_{i_k}, l^0_{j_k})$.

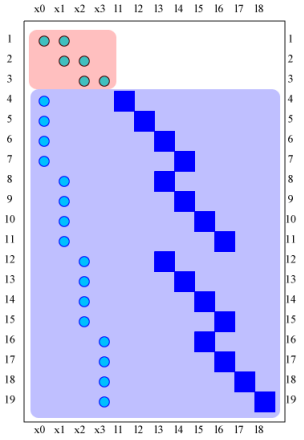


Fig. 3. The corresponding block-structured matrix A' for the factor graph in Figure 2. The red-shaded rows are odometry measurements, and the blue-shaded ones are landmark measurements.

Algorithm 1 Solving the least-squares system in Equation 5 with reordering

- Reorder $A \xrightarrow{\pi} A^{\pi}$
- Repeat
  - Linearize the system and evaluate Jacobian A
  - Cholesky factorization of A: $A^T A \rightarrow R^T R$
  - Solve $R^T y = A^T b$
  - Solve $R\theta = y$
- Until convergence
- Backorder solution $\theta^{\pi} \xrightarrow{\pi} \theta$

By inserting Equations 2 and 3 into Equation 1, we obtain

$$\delta^* = \underset{\delta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{M} \left\| F_i^{i-1} \delta x_{i-1} + G_i^i \delta x_i - a_i \right\|_{\Lambda_i}^2 + \sum_{k=1}^{K} \left\| H_k^{i_k} \delta x_{i_k} + J_k^{j_k} \delta l_{j_k} - e_k \right\|_{\Sigma_k}^2 \right\} \quad (4)$$

where we define $G_i^i \triangleq -I_{d_i}$, $a_i \triangleq x_i^0 - f_i(x_{i-1}^0, u_i)$ and $e_k \triangleq z_k - h_k(x_{i_k}^0, l_{j_k}^0)$.

By combining the Jacobians into matrix A and the vectors a_i and e_k into right-hand side (RHS) vector c, we obtain

$$\delta^* = \underset{\delta}{\operatorname{argmin}} \ \|A\delta - c\|_2^2 \quad (5)$$

We will particularly be interested in the block structure of matrix A, which we denote as A'. As shown in Figure 3, for our small example, the columns of matrix A' correspond to the unknowns $\{x_0; x_1; x_2; x_3; l_1; l_2; l_3; l_4; l_5; l_6; l_7; l_8\}$ and each row represents a measurement. The first three rows correspond to the odometry, while the three groups of four rows below them correspond to landmark measurements at poses x_0, x_1, x_2, and x_3, respectively.

An efficient solution to SAM uses a blend of numerical methods and graph theory. From a linear algebra point of view, solving (5) can be done through Cholesky factorization of the information matrix $\mathcal{I} = A^T A$, which is summarized in Algorithm 1. As the algorithm is based on matrix square roots, this family of approaches will be referred to as *square root SAM* ($\sqrt{\text{SAM}}$) [5].

From a graphic-theoretic point of view, the factorization is equivalent to *variable elimination*, in which we eliminate one node at a time. Eliminating a node p_i introduces dependencies on the adjacent nodes, i.e., all the nodes that are adjacent to p_i have to be fully connected into a clique. In matrix terminology, these extra edges correspond to the non-zero fill-in. The more fill-in, the slower factorization will be, and different variable orderings may yield dramatically more or less fill-in. Since finding an optimal ordering is NP-complete, various algorithms have been developed to find an approximately optimal ordering quickly [20], [1], [4]. The most widely used are known as *minimum degree* algorithms (MD). One of the algorithms in this family is *approximate minimum degree* (AMD) ordering [1], [4], which collects nodes into cliques and eliminates the least constrained ones. We have found empirically that AMD, the ordering used in this paper, is slightly faster than other MD algorithms.

III. Tectonic SAM

In Tectonic SAM, we employ a divide-and-conquer approach and partition the network of nodes into submaps. First, each submap is independently optimized using non-linear least-squares. We then solve the *entire* smoothing and mapping problem by aligning the submaps.

A key element of our approach is that each submap maintains its own *base pose* to define a local coordinate system. Poses and landmarks inside each submap are parameterized relative to this base pose rather than the global coordinate frame. We assume that the local trajectory and landmarks in a submap contain enough information to obtain both a good estimate and a linearization point.

Since we use a local coordinate system, the linearization point of the nodes in the submap remains valid even when the base pose undergoes large transformations. Therefore, the only measurements that are re-linearized are ones that span submaps and hence involve the base poses of at least two submaps.

We align the submaps by iteratively optimizing only the nodes in the *separator*, which are the nodes connected to the measurements spanning submaps. For example, imagine a pair of rooms, joined by a hallway, each room corresponding to a submap, and the nodes in the hallway that see through both rooms are in the separator. By caching the factorization of each submap, the separator optimization is very efficient.

After the separator converges, we finally update the nodes in each submap by back-substitution. Again, the cached factorization of the submaps is employed.

A. Submap Building

We partition the factor graph into P submaps, denoted as $\{M_p \mid p \in 1...P\}$, with each submap containing connected pose nodes and landmark nodes. In Figure 4, we apply a

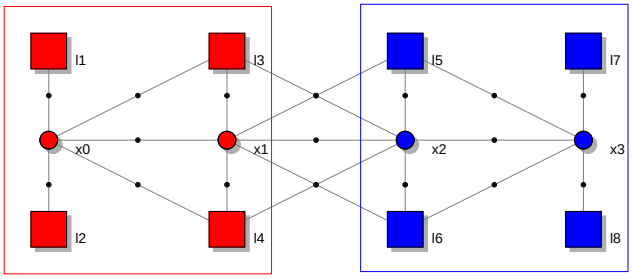


Fig. 4. The factor graph in Figure 2 is partitioned into the two submaps. Each submap is indicated by a colored rectangle boundary.

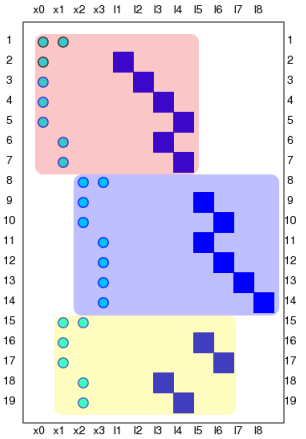


Fig. 5. The block structure of the Jacobian in Figure 3 with rows ordered according to the measurement types and the submap indices. From top to bottom, the shaded areas represent Z_1, Z_2, and $Z_{1,2}$.

vertical cut to our example problem to generate two submaps M_1 and M_2.

We categorize measurements as either *intra-measurements* Z_p or *inter-measurements* $Z_{p,q}$. Intra-measurements Z_p are the set of measurements that involve only the nodes in submap M_p, and the inter-measurements are the set $Z_{p,q}$ that have dependencies on both M_p and M_q. In Figure 4, the inter-measurements are $Z_{1,2} = \{u_{x_1,x_2}, z_{x_1,l_5}, z_{x_1,l_6}, z_{x_2,l_3}, z_{x_2,l_4}\}$, and all other measurements make up the intra-measurements Z_1 and Z_2. The intra-measurements are iteratively relinearized when aligning the submaps locally. The linearization of the intra-measurements is then fixed when optimizing the separator, and only the inter-measurements are linearized during each iteration.

From a matrix point of view, the rows of the block-structure A' can be ordered in a way such that Z_1 and Z_2 (rows 1 to 14 in Figure 5) are placed above $Z_{1,2}$ (rows 15 to 19). As we are going to optimize the submaps one by one, the intra-measurements (rows 1 to 14) are internally ordered with respect to their submap indices p, e.g., first Z_1 (red-shaded area) and then Z_2 (blue-shaded area).

We also define *boundary variables* and *non-boundary variables* with respect to the roles that variables play in the measurements. Variables are boundary if they are involved in at least one inter-measurement and non-boundary otherwise. Thus, each submap M_p is made up of two sets: a non-boundary variable set V_p and a boundary variable set S_p, such that

$$M_p = V_p \cup S_p \tag{6}$$

In Figure 4, we have $V_1 = \{x_0; l_1; l_2\}$, $V_2 = \{x_3; l_7; l_8\}$, $S_1 = \{x_1; l_3; l_4\}$, and $S_2 = \{x_2; l_5; l_6\}$.

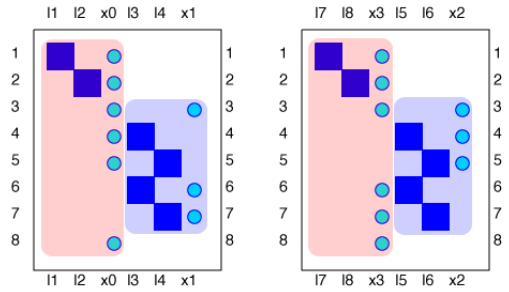


Fig. 6. The subsystems in two submaps. Their columns V_p and S_p are ordered by AMD and covered by red and blue shading.

B. Submap Optimization

Each submap M_p can be optimized locally and independently only using intra-measurements. The optimization of every single submap is simply a small-scale $\sqrt{\text{SAM}}$ problem which can be written as

$$A_p \delta M_p = c_p \tag{7}$$

where A_p and c_p are corresponding parts of A and c in Equation 5 and contain the columns only involved in Z_p.

In order to allow re-use of the linearization point of the intra-measurements, the columns of A_p corresponding to the boundary variables are ordered last, as follows:

$$\begin{bmatrix} A_{V_p} & A_{S_p} \end{bmatrix} \begin{bmatrix} \delta V_p \\ \delta S_p \end{bmatrix} = c_p \tag{8}$$

As always, choosing a good column ordering is important, especially if the submaps contain many variables. As discussed in Section II, we use AMD to obtain a good ordering of both V_p and S_p. The block-structure of the re-ordered matrices A_p in Equation 7 are shown in Figure 6.

Note that after computing the Hessian and its Cholesky factorization, the system of equations being solved is

$$\begin{bmatrix} R_p & T_p \\ 0 & U_p \end{bmatrix} \begin{bmatrix} \delta V_p \\ \delta S_p \end{bmatrix} = \begin{bmatrix} d_p \\ d_{U_p} \end{bmatrix}$$

where $\{d_p; d_{U_p}\}$ is obtained by solving

$$\begin{bmatrix} R_p & T_p \\ 0 & U_p \end{bmatrix}^T \begin{bmatrix} d_p \\ d_{U_p} \end{bmatrix} = c_p$$

Since the separating set variables correspond to the lower right block of the Cholesky factor, the system of equations involving only variables in the boundary variable set can be extracted trivially for later use in the separator optimization:

$$U_p \delta S_p = d_{U_p}$$

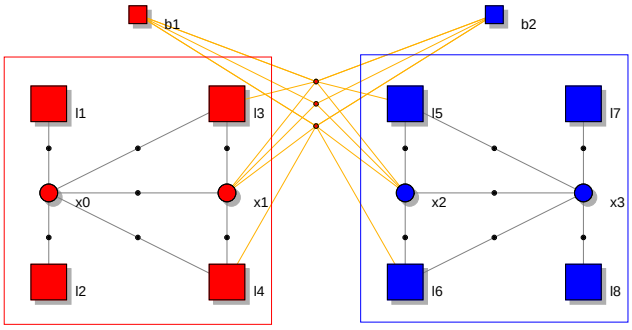


Fig. 7. Two base nodes b_1 and b_2 are added to the partitioned graph in Figure 4. Inter-measurements $Z_{1,2}$ are shown in yellow.

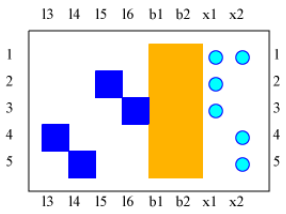


Fig. 8. The subsystem in the separator. Its columns S are also ordered by AMD.

After an individual submap is optimized to convergence, a final linearization and factorization is computed at the optimized submap state and passed on to the separator optimization. In order to ensure that the linearization point remains valid as the submaps are being aligned, we parameterize the submap nodes relative to a local coordinate frame. More specifically, this is accomplished by assigning a *base node* b_p to each submap M_p. All the nodes x in M_p are represented as a relative value x'_i with respect to b_p:

$$x_p = b_p \oplus x'_p$$

Here x can be either a robot pose or a landmark. The set of all base nodes is defined as $B = \{b_i \mid i \in (1, p)\}$.

The result of creating base nodes for both submaps in our example problem is shown in Figure 7. Each inter-measurement connects the two boundary variables and the two base nodes of the corresponding submaps. For instance, in Figure 7, the odometry u_{x_1,x_2} factor connects the four nodes x_1, x_2, b_1, and b_2.

By choosing the parameterization in this way, large global motions of the submaps may significantly affect the linearization point of the inter-measurements, but the linearization point of the intra-measurements will be relatively unaffected.

C. Separator Optimization

Once all the submaps are aligned internally, they are next assembled and optimized. Aligning the submaps and optimizing the separator variables is no longer simply a standard $\sqrt{SAM}$ problem since the linearization of the submaps is cached instead of recomputed at each iteration. The *separator* S is the set of all nodes connected to inter-measurements, indicating that it contains all boundary variables S_p and base nodes b_p:

$$S = S_1 \cup \cdots \cup S_P \cup B$$

As we did with the submaps, we use AMD to order S and we only refer to the reordered S in the remaining paper. In our example problem, the resulting ordered submap variables are $S = \{l_3; l_4; l_5; l_6; b_1; b_2; x_1; x_2\}$.

By linearizing the inter-measurements, we obtain the following system of equations

$$A_S \delta S = c_S \tag{9}$$

which is depicted in Figure 8. These measurements are the only ones relinearized at each iteration of the separator optimization.

The full system of equations is obtained by combining the linearized submap equations with (9). To do that, the update matrices U_p and d_{U_p} have to share the same column ordering as A_S. This is achieved by reordering and inserting all-zero columns, resulting in the update matrices $\tilde{U}_p$ and $\tilde{d}_{U_p}$. The full system of equations can then be expressed as

$$\begin{bmatrix} \tilde{U}_1 \\ \vdots \\ \tilde{U}_P \\ A_S \end{bmatrix} \delta S = \begin{bmatrix} \tilde{d}_{U_1} \\ \vdots \\ \tilde{d}_{U_P} \\ c_S \end{bmatrix} \tag{10}$$

The boundary nodes in the separator keep their values from the submap optimization and the base nodes have to be initialized, for example, by using odometry.

Since the update matrices $\tilde{U} = \left\{\tilde{U}_1; \ldots; \tilde{U}_P\right\}$ do not change from iteration to iteration, the Hessian matrix in the Levenberg-Marquardt optimization is calculated as follows:

$$\begin{bmatrix} \sqrt{\lambda} I \\ \tilde{U} \\ A_S \end{bmatrix}^T \begin{bmatrix} \sqrt{\lambda} I \\ \tilde{U} \\ A_S \end{bmatrix} = A_S^T A_S + \lambda I + \tilde{U}^T \tilde{U}$$

where $\tilde{U}^T \tilde{U}$ is calculated only once. Similarly, for the right-hand-side of Equation 10, we have

$$\begin{bmatrix} \sqrt{\lambda} I \\ \tilde{U} \\ A_S \end{bmatrix}^T \begin{bmatrix} 0 \\ d_U \\ c_S \end{bmatrix} = A_S^T c_S + \tilde{U}^T d_U$$

where $\tilde{U}^T d_U$ is fixed. Although we still need to generate A_S every time, the algorithm still has more than 50% time savings considered that $\tilde{U}$ is bigger than A_S in most test cases.

D. Back-Substitution

Since S_p was modified while optimizing the separator, the final step in the algorithm is to back-substitute and update the submaps. Just as they can be initially optimized independently, the final update of each submap can be done independently. The update step for V_P, given the update step for the separator δS_P, is obtained by solving

$$R_P \delta V_P = d_P - T_P \delta S_P \tag{11}$$

where δS_P is the difference between the linearization point and the final separator value $\delta S_p = S_p^0 - S_p^N$. The final value $\hat{V}_p = V_p^{w_p} - \delta V_p$ for the non-boundary nodes in the submap V_p is then obtained.

E. Summary

In this section, we will summarize the T-SAM algorithm we discussed above, and more importantly, we will show the ultimate matrix system in T-SAM, which may give readers a good intuition.

In T-SAM, a base node is assigned to each partitioned submap. By defining the boundary variables, non-boundary variables, and the separator, we may derive an optimal ordering for the nodes in the factor graph:

$$V_1 @ \cdots @ V_P @ S \tag{12}$$

where @ indicates the concatenating operation. Note that our approach orders small submaps first and the separator last, rather than order the columns of the entire sparse matrix A'. As we discussed above, the ordering obtained here is crucial to speed up the optimization for the submaps and the separator.

In a matrix view, matrix A' is manipulated as follows:

1) Rows are reordered so that intra-measurements $\{Z_p\}$ are placed above inter-measurements $\{Z_{p,q}\}$.
2) Rows for intra-measurements $\{Z_p\}$ are ordered by the submap index p.
3) For Z_p, the measurements involved only with V_p are placed above those involved with S_p.
4) Columns are ordered according to Equation 12.

After these manipulations, the entire system can be restructured as

$$\begin{bmatrix} A_{V_1} & & & A_{S_1} \\ & \ddots & & \vdots \\ & & A_{V_P} & A_{S_P} \\ 0 & \cdots & 0 & A_S \end{bmatrix} \begin{bmatrix} \delta V_1 \\ \vdots \\ \delta V_P \\ \delta S \end{bmatrix} = \begin{bmatrix} c_1 \\ \vdots \\ c_P \\ c_S \end{bmatrix}$$

The example matrix is shown in Figure 9.

One point worth mentioning is the gauge freedom [26]. When we optimize submaps locally, we randomly pick up a node from each submap and use the current estimation of that node as its prior. For the separator, a prior is set on the base node of the first submap.

IV. Numerical Analysis

With the main idea of T-SAM presented mostly in the last section, we will show how the numerical performance of SAM can be enhanced by doing T-SAM. Previous work [5] already shows that a typical SAM iteration consists of three most computationally expensive parts: linearization for the measurements, matrix multiplications from computing Hessian matrices, and factorization.

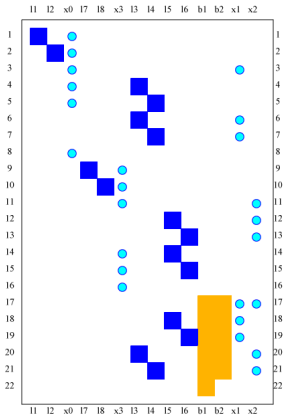


Fig. 9. The full matrix A' after all manipulations.

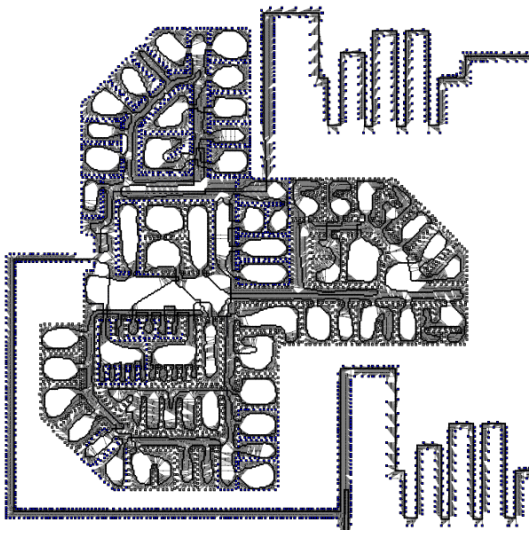


Fig. 11. One floor from the skyscraper simulation data set from [13].

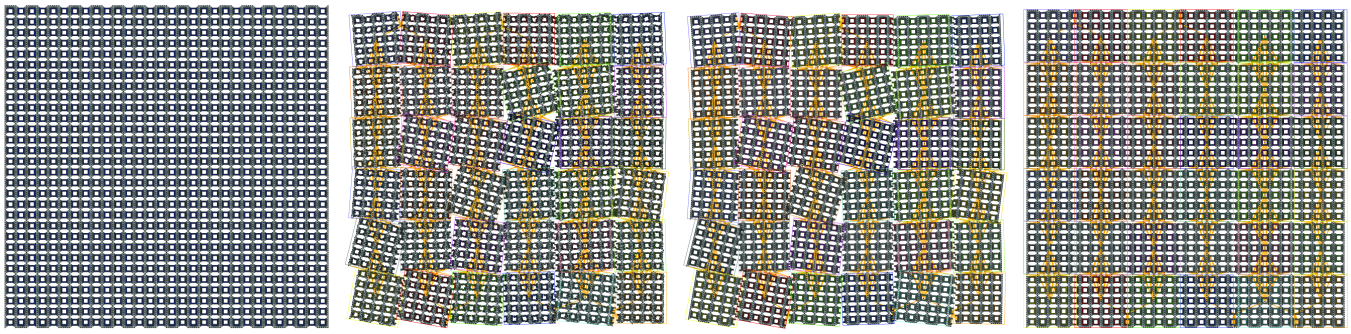
The optimization for the submap is carried out by using normal bundle adjustment, and it does not have much possible tweaking. However, the separator usually contains more variables than any single submap does in practice, and consumes more time. Fortunately considerable computation time can be saved by taking advantage of T-SAM.

Rather than linearize all the measurements involved with the separator variables, we relinearize only the inter-measurements to A_S and keep the factor U from intra-measurements. In the experiments, we found that the size of U is usually slightly larger than A_S, leading to 50% time savings in the linearization work.

V. Experiments

A. Data Set

We have carried out experiments on both synthetic environments and real world data. In the simulation tests, we created block-world environments like the one shown in Figure 10. As we know the exact ground truth, it is ideal to verify the accuracy of T-SAM algorithm. The other simulation data set



(1) (2) (3) (4)

Fig. 10. The block-world data set with noise, which has 18000 landmarks along with a 4856-step walk trajectory, and 55038 measurements. Shown from left to right: 1) the original data set; 2) the system with noise added; 3) the system after submaps are optimized locally; 4) the system after the separator is optimized.

	SAM	T-SAM	Separator
Block World 18000	13.15s	36×0.12s	0.30s
Victoria Park Data	6.28s	5×0.19s	0.10s

TABLE I

RESULTS OF T-SAM APPLIED TO THE SYNTHETIC BLOCK-WORLD DATA AND THE VICTORIA PARK DATA SET. SHOWN FROM LEFT TO RIGHT IS THE TIME PER ITERATION USED IN SAM, THE T-SAM SUBMAP, AND THE SEPARATOR.

is four skyscrapers from [13]. We also test the T-SAM algorithm on the Victoria Park dataset in which a truck equipped with a laser scanner was driven through Victoria Park in Sydney, Australia (available at `http://www.acfr.usyd.edu.au/homepages/academic/enebot/dataset.htm`). Trees within the park serve as point features in the SLAM map. We assume the data association is known and focus on only verifying the SLAM solver. The testbed is a T60 laptop with an Intel T2400 1.83GHz CPU and 1GB memory.

B. Experimental Results

The experimental results are listed in Table I. In the block-world data set, the origin map is partitioned by 6×6 grids, which is the most common way to partition well-structured data. Notice that after the submaps are optimized, the nodes are nicely aligned in Figure 10(2). In the separator, we obtained a Hessian matrix with size 13829×13829 and 411075 non-zero fill-in. The result yields a 65.9% time savings in each iteration compared with normal SAM approach, and the residual is 0.06 compared with SAM's 0.03.

For the Victoria data set, the submaps are created along the trajectory when a new pose or landmark is not in the range of previous submaps. The data set generates a separator with a 779×779 Hessian matrix and 61265 non-zero fill-in. As there are fewer measurements in this data set, the sensor poses do not have many constraints, which make the quality of submap optimization not as good as the block-world one. The residual of the final estimation is 0.81, while SAM achieves a residual of 0.35.

Next, we analyze the performance impact caused by

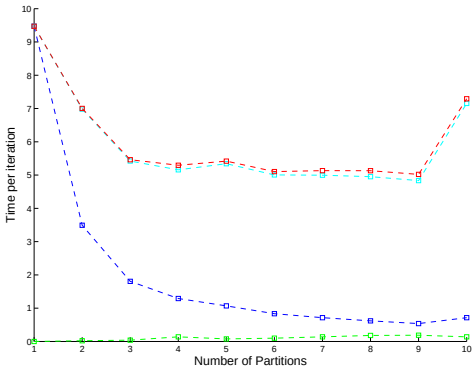


Fig. 12. The time per iteration using different partition number. The blue, cyan, green, and red dashed lines are the time cost for each iteration in one submap, all submaps, the separator, and the overall system respectively.

submap partitions in T-SAM. As the only approximation we made is the linearization point in each submap, the performance is influenced by the partition number. In the extreme case, if we make partitions such that every node in the graph is a submap, then there is no optimization work in the submaps. The separator will involve all the nodes in the original graph and we are actually doing full bundle adjustment when we optimize the separator. In Figure 12, the result of full bundle adjustment using AMD ordering corresponds to the point with partition number 1. We also show the performance with different partition numbers, which are compared with full bundle adjustment. As shown in the figure, the time per iteration in submaps decreases linearly at first and then stays the same later, as does the time for the overall system. With a good partition, the size of the separator does not increase very much and its optimization time stays almost constant.

Not very surprisingly, we also found that decomposed problems converge more easily than the normal ones. In practice, it usually takes 40% to 50% less iterations.

T-SAM can be easily implemented to process data that is too large to fit into a computer's physical memory, i.e., *out-of-*

core, which is essential to tackle large-scale problems such as the one in Figure 11. Given the fact that the submaps can be optimized independently, we implement the out-of-core version of T-SAM, so that the optimized submaps and the update matrices are stored on disks after they are no longer in use, and load them when we start to optimize the separator. By doing this, we only keep a small portion of data in the memory. In theory, our approach can handle any size of problems as long as the memory can hold both single submaps and the separator. We optimized four eight-floor buildings (one of these floors is shown in Figure 11), and each floor takes 2.30 seconds per iteration to optimize locally. The separator is optimized in 0.05 seconds given the fact that each floor is only connected by an elevator.

VI. Conclusion & Discussion

The contributions of this paper can be summarized as follows:

1) We propose a submap algorithm to solve the SLAM problem in a smoothing scheme. Unlike existing EKF-based approaches, our algorithm is *exact* in nature.
2) The optimal orderings are derived from small submaps instead of an entire graph. Our results show that the ordering helps both speeding up the factorization and generating update matrices.
3) Our approach saves time on linearization by fixing linearization points when optimizing the separator.
4) Submaps are explicit, which is ideal for various applications.

Readers may argue that our algorithm can be improved by implementing multiple levels of submaps and separators. We are not doing so because the linearization point is a very crucial factor when solving the non-linear problem. When the system has multiple levels, e.g., [12], factors with imperfect linearizations are passed around from the leaves to the root. The more levels we have, the more error will get accumulated. In some sense, our approach is a partitioned version of SAM, while most previous work uses a bundled version of EKF, which results in more approximation.

For the future work, to yield as few inter-measurements as possible, we are planning to integrate more advanced partition algorithms instead of the current manual partitioning. Our plan also includes a 3D implementation working for cameras instead of laser scanners.

Acknowledgements: This material is based upon work supported by the National Science Foundation under Grant No. IIS - 0448111. Early exploratory work on this topic was done by Alexander Kipp and Peter Krauthausen, without which our task would have been much harder. In addition, we would like to thank Udo Frese as well as Eduardo Nebot and Hugh Durrant-Whyte for sharing their datasets with us.

References

[1] P.R. Amestoy, T. Davis, and I.S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[2] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An Atlas framework for scalable mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2003.

[3] R. Chatila and J.-P. Laumond. Position referencing and consistent world modeling for mobile robots. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 138–145, 1985.

[4] T.A. Davis, J.R. Gilbert, S.I. Larimore, and E.G. Ng. A column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):353–376, 2004.

[5] F. Dellaert. Square Root SAM: Simultaneous location and mapping via square root information smoothing. In *Robotics: Science and Systems (RSS)*, 2005.

[6] C. Estrada, J. Neira, and J.D. Tardos. Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Trans. Robotics*, 21(4):588–596, August 2005.

[7] O.D. Faugeras. *Three-dimensional computer vision: A geometric viewpoint*. The MIT press, Cambridge, MA, 1993.

[8] J. Folkesson and H. I. Christensen. Graphical SLAM - a self-correcting map. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 383 – 390, 2004.

[9] J. Folkesson, P. Jensfelt, and H.I. Christensen. Graphical SLAM using vision and the measurement subspace. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

[10] U. Frese. *An $O(\log n)$ Algorithm for Simultaneous Localization and Mapping of Mobile Robots in Indoor Environments*. PhD thesis, University of Erlangen-Nürnberg, 2004.

[11] U. Frese. Treemap: An $O(\log n)$ algorithm for simultaneous localization and mapping. In *Spatial Cognition IV*, pages 455–476. Springer Verlag, 2005.

[12] U. Frese. Treemap: An $O(\log n)$ algorithm for indoor simultaneous localization and mapping. *Autonomous Robots*, 21(2):103–122, 2006.

[13] U. Frese and L. Schröder. Closing a million-landmarks loop. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5032–5039, Oct 2006.

[14] J. Guivant and E. Nebot. Compressed filter for real time implementation of simultaneous localization and map building. In *FSR 2001 International Conference on Field and Service Robots*, volume 1, pages 309–314, 2001.

[15] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[16] S.J. Julier and J.K. Uhlmann. A counter example to the theory of simultaneous localization and map building. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 4, pages 4238–4243, 2001.

[17] M. Kaess, A. Ranganathan, and F. Dellaert. Fast incremental square root information smoothing. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2129–2134, Hyderabad, India, 2007.

[18] K. Konolige. Large-scale map-making. In *Proc. 21^{th} AAAI National Conference on AI*, San Jose, CA, 2004.

[19] J. J. Leonard and H. J. S. Feder. Decoupled stochastic mapping. *IEEE Journal of Oceanic Engineering*, pages 561–571, October 2001.

[20] R.J. Lipton and R.E. Tarjan. Generalized nested dissection. *SIAM Journal on Applied Mathematics*, 16(2):346–358, 1979.

[21] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, pages 333–349, April 1997.

[22] R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *Intl. J. of Robotics Research*, 5(4):56–68, 1987.

[23] J.D. Tardós, J. Neira, P.M. Newman, and J.J. Leonard. Robust mapping and localization in indoor environments using sonar data. *Intl. J. of Robotics Research*, 21(4):311–330, 2002.

[24] S. Thrun. Robotic mapping: a survey. In *Exploring artificial intelligence in the new millennium*, pages 1–35. Morgan Kaufmann, Inc., 2003.

[25] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.

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

[27] S.B. Williams, G. Dissanayake, and H. Durrant-Whyte. An efficient approach to the simultaneous localisation and mapping problem. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 406–411, 2002.