Incremental Sparse GP Regression for Continuous-time Trajectory Estimation & Mapping

Xinyan Yan
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA
xinyan.yan@cc.gatech.edu

Vadim Indelman
Faculty of Aerospace Engineering
Technion - Israel Institute of Technology
Haifa 32000, Israel
vadim.indelman@technion.ac.il

Byron Boots
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA
bboots@cc.gatech.edu

Abstract

Recent work has investigated the problem of continuous-time trajectory estimation and mapping for mobile robots by formulating the problem as sparse Gaussian process regression. Gaussian processes provide a continuous-time representation of the robot trajectory, which elegantly handles asynchronous and sparse measurements, and allows the robot to query the trajectory to recover it's estimated position at any time of interest. One of the major drawbacks of this approach is that Gaussian process regression formulates continuous-time trajectory estimation as a *batch* estimation problem. In this work, we provide the critical extensions necessary to transform this existing batch approach into an extremely efficient incremental approach. In particular, we are able to vastly speed up the solution time through efficient variable reordering and incremental sparse updates, which we believe will greatly increase the practicality of Gaussian process methods for robot mapping and localization. Finally, we demonstrate the approach and its advantages on both synthetic and real datasets.

1 Introduction & Related Work

The problem of simultaneously recovering the location of a robot and a map of its environment from sensor readings is a fundamental challenge in robotics, persisting as a core research topic for several decades. Well-known approaches to this problem, such as smoothing and mapping (SAM) [1], have focused on regression-based methods that exploit the sparse structure of the problem to efficiently compute a solution. The main weakness of the original SAM algorithm was that it was a *batch* method: all of the data must be collected before a solution can be found. For a robot traversing an environment, the inability to update an estimate of its trajectory online is a significant drawback. In response to this weakness, Kaess et al. [2, 3] developed a critical extension to the batch SAM algorithm, called incremental smoothing and mapping (iSAM 2.0), that overcomes this problem by *incrementally* computing a solution. The approach employs an efficient data structure called a *Bayes tree* [4] to perform incremental variable reordering and just-in-time relinearization. iSAM 2.0 and its extensions are widely considered to be state-of-the-art in robot trajectory estimation and mapping.

The majority of previous approaches to trajectory estimation and mapping have formulated the problem in discrete time. However, in the real world, the trajectory of a robot is continuous and sensor measurements are often sampled asynchronously. A continuous-time formulation of the SAM problem where the robot trajectory is a *function* $\boldsymbol{x}(t)$ that maps any time t to a robot pose is more appropriate for this problem. The problem of estimating this function along with landmark loca-

tions has been dubbed *simultaneous trajectory estimation and mapping* (STEAM). Tong et al. [5, 6] proposed a Gaussian process (GP) regression approach to solving the STEAM problem. While their approach was able to accurately model and interpolate asynchronous data to recover a trajectory and landmark estimate, it suffered from significant computational challenges: naive Gaussian process approaches to regression have notoriously high space and time complexity. Additionally, Tong et al.'s approach is a *batch* method, so updating the solution necessitates saving all of the data and completely resolving the problem.

In this work, we provide the critical extensions necessary to transform the existing Gaussian process-based approach to solving the STEAM problem into an extremely efficient incremental approach. Our algorithm elegantly combines the benefits of Gaussian processes and iSAM 2.0. Like the GP regression approaches to STEAM, our approach can model continuous trajectories and handle asynchronous measurements, and, like iSAM 2.0, our approach uses a Bayes tree to efficiently calculate a *maximum a posteriori* (MAP) estimate of the GP trajectory while performing incremental factorization, variable reordering, and just-in-time relinearization. The result is an online GP-based solution to the STEAM problem that remains computationally efficient while scaling up to extremely large datasets.

2 Batch Trajectory Estimation & Mapping as Gaussian Process Regression

Following Tong et al. [5] and Barfoot et al. [6], we take a Gaussian process regression approach to state estimation, where we represent robot trajectories $\boldsymbol{x}$ as functions of time t.

$$\boldsymbol{x}(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \boldsymbol{\mathcal{K}}(t, t')) \quad t_0 < t, t' \tag{1}$$

$$\boldsymbol{y}_i = \boldsymbol{h}_i(\boldsymbol{\theta}_i) + \boldsymbol{n}_i, \quad \boldsymbol{n}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_i), \quad i = 1, 2, ..., N \tag{2}$$

Here, $\boldsymbol{x}(t)$ is the continuous-time trajectory of the robot through state space, represented by a Gaussian process with mean $\boldsymbol{\mu}(t)$ and covariance $\boldsymbol{\mathcal{K}}(t, t')$. Measurements $\boldsymbol{y}_i$ are nonlinear functions of the set of related variables $\boldsymbol{\theta}_i$ plus some Gaussian noise $\boldsymbol{n}_i$. The related variables for a range measurement are the robot state $\boldsymbol{x}(t)$ at the measurement time and the associated landmark location $\boldsymbol{\ell}_j$. The total number of measurements is N. The landmarks $\boldsymbol{\ell} = \begin{bmatrix} \boldsymbol{\ell}_1 & \boldsymbol{\ell}_2 & \dots & \boldsymbol{\ell}_M \end{bmatrix}$ are assumed to be sampled from a joint Gaussian distribution $\boldsymbol{\ell} \sim \mathcal{N}(\boldsymbol{d}, \boldsymbol{W})$ with mean $\boldsymbol{d}$ and covariance $\boldsymbol{W}$. The prior distribution of the combined state $\boldsymbol{\theta}$ that consists of robot poses at measurement times and landmarks is, therefore, a joint Gaussian distribution:

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\eta}, \boldsymbol{\mathcal{P}}), \quad \boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{\mu} & \boldsymbol{d} \end{bmatrix}^\mathsf{T}, \quad \boldsymbol{\mathcal{P}} = \begin{bmatrix} \boldsymbol{\mathcal{K}} & \\ & \boldsymbol{W} \end{bmatrix} \tag{3}$$

To solve the STEAM problem, we compute the *maximum a posteriori* (MAP) estimate of the combined state *conditioned* on measurements:

$$\begin{aligned} \boldsymbol{\theta}^* &= \operatorname*{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\boldsymbol{y}) = \operatorname*{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \boldsymbol{y}) \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta}) p(\boldsymbol{y}|\boldsymbol{\theta}) = \operatorname*{argmin}_{\boldsymbol{\theta}} \left(-\log p(\boldsymbol{\theta}) - \log p(\boldsymbol{y}|\boldsymbol{\theta})\right) \\ &= \operatorname*{argmin}_{\boldsymbol{\theta}} \left((\boldsymbol{\theta} - \boldsymbol{\eta})^\mathsf{T} \boldsymbol{\mathcal{P}}^{-1} (\boldsymbol{\theta} - \boldsymbol{\eta}) + (\boldsymbol{y} - \boldsymbol{h}(\boldsymbol{\theta}))^\mathsf{T} \boldsymbol{R}^{-1} (\boldsymbol{y} - \boldsymbol{h}(\boldsymbol{\theta}))\right) \end{aligned} \tag{4}$$

where $\boldsymbol{h}(\boldsymbol{\theta})$ and $\boldsymbol{R}$ are the mean and covariance of the measurements, respectively:

$$\boldsymbol{h}(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{h}_1(\boldsymbol{\theta}_1) & \boldsymbol{h}_2(\boldsymbol{\theta}_2) & \dots & \boldsymbol{h}_N(\boldsymbol{\theta}_N) \end{bmatrix}^\mathsf{T}, \quad \boldsymbol{R} = \operatorname{diag}(\boldsymbol{R}_1, \boldsymbol{R}_2, \dots, \boldsymbol{R}_N)$$

If measurement functions $\boldsymbol{h}_i(\cdot)$ are nonlinear, nonlinear optimization methods are required. Examples include Gauss-Newton or Levenberg-Marquardt [7], which solve a sequence of approximated linearized problems to approach the minimum iteratively. A linearization of a measurement function at current state estimate $\bar{\boldsymbol{\theta}}_i$ can be accomplished by a first-order Taylor expansion:

$$\boldsymbol{h}_i\left(\bar{\boldsymbol{\theta}}_i + \delta\boldsymbol{\theta}_i\right) \approx \boldsymbol{h}_i(\bar{\boldsymbol{\theta}}_i) + \left.\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{\theta}_i}\right|_{\bar{\boldsymbol{\theta}}_i} \delta\boldsymbol{\theta}_i \tag{5}$$

Combining with Eq. 4, the optimal increment $\delta\boldsymbol{\theta}^*$ at the current combine state estimate $\bar{\boldsymbol{\theta}}$ is

$$\delta\boldsymbol{\theta}^* = \operatorname*{argmin}_{\delta\boldsymbol{\theta}} \left((\bar{\boldsymbol{\theta}} + \delta\boldsymbol{\theta} - \boldsymbol{\eta})^\mathsf{T} \boldsymbol{\mathcal{P}}^{-1} (\bar{\boldsymbol{\theta}} + \delta\boldsymbol{\theta} - \boldsymbol{\eta}) + (\boldsymbol{y} - \boldsymbol{h}(\bar{\boldsymbol{\theta}}) - \boldsymbol{H}\delta\boldsymbol{\theta})^\mathsf{T} \boldsymbol{R}^{-1} (\boldsymbol{y} - \boldsymbol{h}(\bar{\boldsymbol{\theta}}) - \boldsymbol{H}\delta\boldsymbol{\theta})\right) \tag{6}$$

Where $\boldsymbol{H}$ is the measurement Jacobian:

$$\boldsymbol{H} = \text{diag}(\boldsymbol{H}_1, \boldsymbol{H}_2, \ldots, \boldsymbol{H}_N), \quad \boldsymbol{H}_i = \left.\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{\theta}_i}\right|_{\bar{\boldsymbol{\theta}}_i}$$

To solve Eq. 6, we take the derivative with respect to $\delta\boldsymbol{\theta}$, and set it to zero, which gives us $\delta\boldsymbol{\theta}^*$ embedded in a set of linear equations

$$\underbrace{(\boldsymbol{\mathcal{P}}^{-1} + \boldsymbol{H}^\intercal \boldsymbol{R}^{-1} \boldsymbol{H})}_{\boldsymbol{\mathcal{I}}} \delta\boldsymbol{\theta}^* = \underbrace{\boldsymbol{H}^\intercal \boldsymbol{R}^{-1}(\boldsymbol{y} - \bar{\boldsymbol{h}}) - \boldsymbol{\mathcal{P}}^{-1}(\bar{\boldsymbol{\theta}} - \boldsymbol{\eta})}_{\boldsymbol{b}} \tag{7}$$

The positive definite matrix $\boldsymbol{\mathcal{P}}^{-1} + \boldsymbol{H}^\intercal \boldsymbol{R}^{-1} \boldsymbol{H}$ is the *a posteriori* information matrix, which we label $\boldsymbol{\mathcal{I}}$. To solve this set of linear equations for $\delta\boldsymbol{\theta}^*$, we don't actually have to calculate the inverse $\boldsymbol{\mathcal{I}}^{-1}$. Instead, factorization-based methods can be leveraged for a fast, numerically stable solution. For example, $\delta\boldsymbol{\theta}^*$ can be found by first performing a Cholesky factorization $\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}^\intercal = \boldsymbol{\mathcal{I}}$, and then solving $\boldsymbol{\mathcal{L}}\boldsymbol{d} = \boldsymbol{b}$ and $\boldsymbol{\mathcal{L}}^\intercal \delta\boldsymbol{\theta}^* = \boldsymbol{d}$ by back substitution. If $\boldsymbol{\mathcal{I}}$ is dense, the time complexity of a Cholesky factorization and back substitution are $O(N^3)$ and $O(N^2)$ respectively. However, if $\boldsymbol{\mathcal{I}}$ has sparse structure, then the solution can be found much faster. For example, for a banded matrix, the computation time is $O(N)$ instead of $O(N^3)$. Fortunately, we can guarantee sparsity for the STEAM problem (see Section 2.1 below). At each iteration we update the state $\bar{\boldsymbol{\theta}} \leftarrow \bar{\boldsymbol{\theta}} + \delta\boldsymbol{\theta}^*$ and repeat the process until convergence.

2.1 Sparse Gaussian Process Regression

The efficiency of the Gaussian Process Gauss-Newton algorithm presented in Section 2 is heavily dependent on the choice of kernel. It is well-known that if the information matrix $\boldsymbol{\mathcal{I}}$ is sparse, then it is possible to very efficiently compute the solution to Eq. 7 [2]. For the batch GP trajectory estimation and mapping problem, Barfoot et al. [6] suggest a kernel matrix with a sparse inverse that is well-suited to the simultaneous trajectory estimation and mapping problem. In particular, Barfoot et al. show that $\boldsymbol{\mathcal{K}}^{-1}$ is exactly block-tridiagonal when the GP is assumed to be generated by linear, time-varying (LTV) stochastic differential equation (SDE) which we describe here:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{v}(t) + \boldsymbol{F}(t)\boldsymbol{w}(t), \qquad \boldsymbol{w}(t) \sim \mathcal{GP}(\boldsymbol{0},\, \boldsymbol{Q}_c\delta(t - t')) \quad t_0 < t, t' \tag{8}$$

where $\boldsymbol{x}(t)$ is state, $\boldsymbol{v}(t)$ is exogenous input, $\boldsymbol{w}(t)$ is process noise, and $\boldsymbol{F}(t)$ is time-varying system matrix. The process noise $\boldsymbol{w}(t)$ is modeled by a Gaussian process, and $\delta(\cdot)$ is the *Dirac delta function*. (See [6] for details). We consider a specific case of this model in the experimental results in Section 4.1. Assuming the GP is generated by Eq. 8, and the measurements are range and odometry measurements, the sparse information matrix becomes

$$\boldsymbol{\mathcal{I}} = \begin{bmatrix} \boldsymbol{\mathcal{I}}_{xx} & \boldsymbol{\mathcal{I}}_{x\ell} \\ \boldsymbol{\mathcal{I}}_{x\ell}^\intercal & \boldsymbol{\mathcal{I}}_{\ell\ell} \end{bmatrix} \tag{9}$$

where $\boldsymbol{\mathcal{I}}_{xx}$ is block-tridiagonal and $\boldsymbol{\mathcal{I}}_{\ell\ell}$ is block-diagonal. $\boldsymbol{\mathcal{I}}_{x\ell}$'s density depends on the frequency of landmark measurements and how they are carried out.

3 Bayes Tree for Fast Incremental Updates to Sparse GP Regression

The iSAM 2.0 algorithm proposed by Kaess et al. [3] was designed to efficiently solve a nonlinear discrete-time trajectory estimation problem. It is an incremental and real-time approach that works by directly operating on the factor graph representation of the SAM problem. The core technology behind this approach is the *Bayes tree* data structure which allows for incremental variable reordering and fluid relinearization [4]. We apply the same data structure to sparse Gaussian process regression in the context of the STEAM problem, thereby eliminating the need for periodic batch computation.

To understand how the Bayes tree can be incorporated into the problem, it is helpful to understand the how the GP estimation problem can be represented as a factor graph. A factor graph is a bipartite graph $G = (\mathcal{F}, \boldsymbol{\theta}, \mathcal{E})$, where $\mathcal{F}$ is the set of factor nodes that encodes all probabilistic constraints on variables, including range measurements, odometry measurements or smoothing priors, $\boldsymbol{\theta}$ is the set of variable nodes to estimate, and $\mathcal{E}$ is the set of edges in the graph. Based on the variable dependence captured in the factor graph, the joint probability of variables to estimate is factored as

$$f(\boldsymbol{\theta}) = \prod_i f_i(\boldsymbol{\theta}_i) \tag{10}$$

where $f_i \in \mathcal{F}$ is one of the factors, and $\boldsymbol{\theta}_i$ is the set of variables directly connected to f_i. Then the estimation problem is to find $\boldsymbol{\theta}^*$ that maximizes Eq. 10. As stated in Section 2.1, Barfoot et al. prove that when GPs are generated by LTV SDE, $\boldsymbol{\mathcal{K}}^{-1}$ is block-tridiagonal [6]. This leads to an interesting interpretation of the states as Markovian, even though we are using a continuous-time prior. In other words, the Gaussian process smoothing factors only connect consecutive pairs of states, and they result from the underlying continuous-time process model.

The factor graph can be converted to a Bayes net by a *bipartite elimination game* [8]. This procedure is equivalent to converting Eq. 6 to least-squares form and computing the square-root information matrix by Householder reflections or Gram-Schmidt orthogonalization. In order to facilitate marginalization and optimization, a Bayes-tree is constructed from the Bayes net [4]. The Bayes tree groups several variables together based on their dependence, with each node corresponding to a clique in the net. From a linear algebra perspective, the Bayes tree captures the structure of the Cholesky factor $\boldsymbol{\mathcal{L}}$ of $\boldsymbol{\mathcal{I}}$, and the sequence of back substitutions that can be performed. When we add a new measurement, add a prior for new variables, or relinearize a previous measurement, $\boldsymbol{\mathcal{L}}$ will change accordingly. Importantly, all these modifications to the factor graph only have *local* effects on $\boldsymbol{\mathcal{L}}$. Exploiting this observation is the foundation for efficient incremental updates.

Since the nodes of Bayes tree encode conditional probability distributions which directly correspond to rows in $\boldsymbol{\mathcal{L}}$, the structure of the tree can be leveraged to efficiently update the factor $\boldsymbol{\mathcal{L}}$. The nodes containing the variables that are involved in new factors or whose linear step is larger than a predetermined threshold are identified.[1] Only these nodes and their ascendants in the Bayes tree are then updated. Note that when a sub-tree is updated, variables in the sub-tree are reordered by constrained COLAMD [9] to heuristically maximize the locality of future updates. Last but not least, $\delta\boldsymbol{\theta}^*$ is computed from tree root to leaves because of the information about the structure of $\boldsymbol{\mathcal{L}}$ encoded in the Bayes tree. This propagation stops when updates to the conditioning variables are below a predetermined threshold. Algorithm 1 summarizes the incremental Gaussian process regression leveraging the Bayes tree data structure in detail.

Algorithm 1 Updating sparse GP Regression by Bayes tree

while collecting data **do**

1. Get new measurement results, and identify new factors $\mathcal{F}_{new}$ and related variables $\boldsymbol{\theta}_{rel} = \bigcup \boldsymbol{\theta}_i$, $f_i \in \mathcal{F}_{new}$
2. For each affected variable in $\boldsymbol{\theta}_{aff} = \boldsymbol{\theta}_{lin} \cup \boldsymbol{\theta}_{rel}$, remove the corresponding clique and ascendants up to the root of Bayes tree
3. Relinearize all factors required to recreate the removed part of the tree
4. Add cached marginal factors from orphaned sub-trees of removed cliques and create a factor graph
5. Eliminate the factor graph by a new variable ordering, create a Bayes tree, and attach back orphaned sub-trees
6. Partially update estimate from root to leaves and stop walking down a branch when the updates to variables that the child clique is conditioned on are not significant enough
7. Collect variables involved in the measurement factors $\mathcal{F}_{lin}$ where previous linearization point is far from current estimate, $\boldsymbol{\theta}_{lin} = \bigcup \boldsymbol{\theta}_i$, $f_i \in \mathcal{F}_{lin}$

end while

4 Experimental Results

4.1 Synthetic SLAM Exploration Task

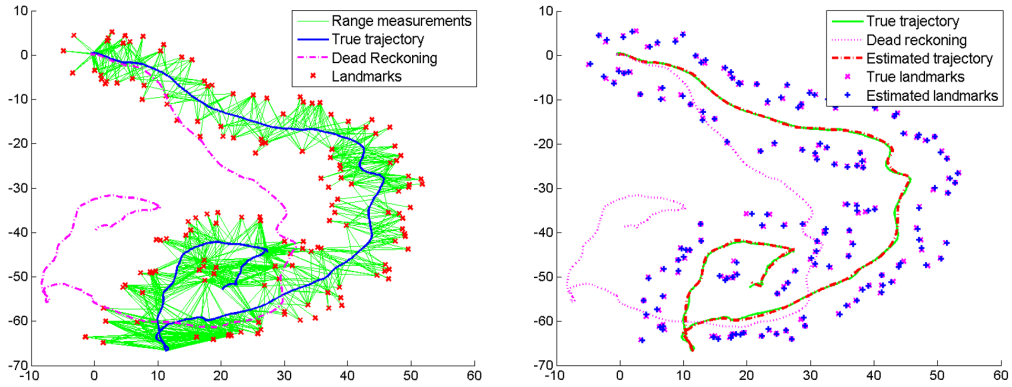
The data consists of an exploration task with 1,500 process states $\boldsymbol{x}(t_i) = [\ \boldsymbol{p}(t_i) \quad \dot{\boldsymbol{p}}(t_i)\ ]^\intercal$, $\boldsymbol{p}(t_i) = [\ x(t_i) \quad y(t_i) \quad \theta(t_i)\ ]^\intercal$, with a contant time interval, 151 landmarks, and 1,500 range and odometry measurements (Fig 1a). Each process state has one odometry and one range measurement. The trajectory is generated from a white noise acceleration prior $\ddot{\boldsymbol{p}}(t) = \boldsymbol{w}(t)$, i.e. constant velocity prior.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{F}\boldsymbol{w}(t) \tag{11}$$

where

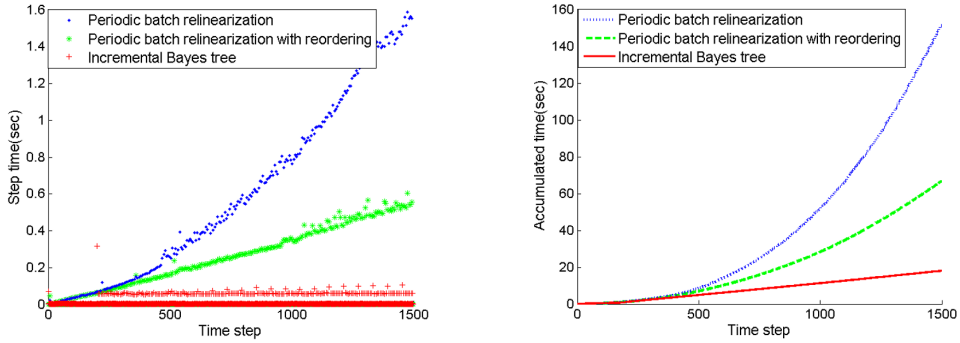
$$\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{p}(t) \\ \dot{\boldsymbol{p}}(t) \end{bmatrix}, \quad \boldsymbol{p}(t) = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{w}(t) \sim \mathcal{GP}(\boldsymbol{0}, \boldsymbol{Q}_c\delta(t-t'))$$

[1] The reader is referred to [3] for additional details regarding this just-in-time relinearization.



(a) True trajectory and landmarks (b) Estimated trajectory and landmarks

Figure 1: Solving the STEAM problem on a synthetic dataset. Ground truth (a) and combined state estimate (b). In (a), the lines between trajectory points and landmarks indicate range measurements. At each time step, the robot receives a noisy measurement of the distance between itself and a random nearby landmark. (b) shows that the estimate is very close to the ground truth.



(a) Comparison of time spent on each time step (b) Comparison of accumulated time

Figure 2: Comparison of time spent on each time step and accumulated time for each of the three approaches – periodic batch relinearization, periodic batch relinearization with variable reordering, and the incremental Bayes tree-based approach. Because the number of landmarks, which is 151, is not negligible compared to the number of process states, variable reordering dramatically improves the performance. In all cases, a new iteration of the algorithm, i.e. an update to the combined state estimate, is carried out when 5 new range measurements have accumulated. So in (a), the computation time in many time steps is close to zero. Notice that the Bayes tree approach has almost no increase in computation time when the size of problem grows.

Note that velocity $\dot{\boldsymbol{p}}(t)$ has to be included in process state to represent the motion in LTV SDE form, which leads to Markovian states and block-tridiagonal inverse kernel matrix $\boldsymbol{\mathcal{K}}^{-1}$. The odometry and range measurements are specified in Eq. 12 and Eq. 13 respectively.

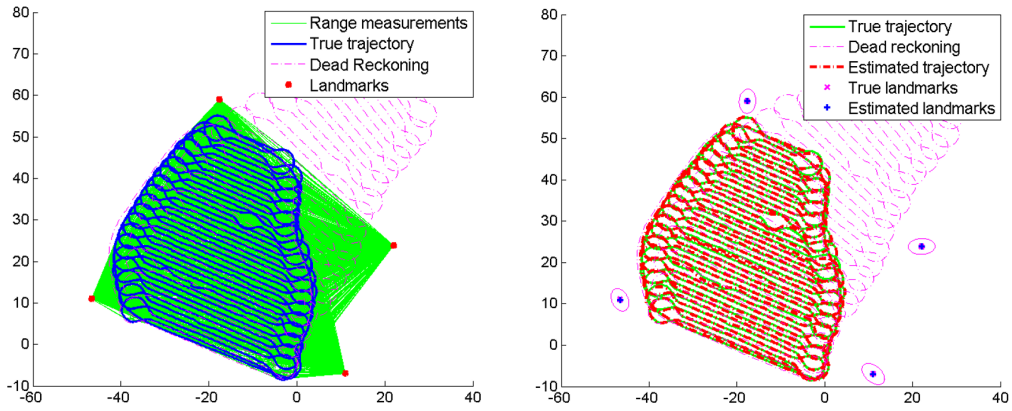
$$\boldsymbol{y}_{io} = \begin{bmatrix} \cos\theta(t_{i-1}) \cdot (x(t_i) - x(t_{i-1})) + \sin\theta(t_{i-1}) \cdot (y(t_i) - y(t_{i-1})) \\ \theta(t_i) - \theta(t_{i-1}) \end{bmatrix} + \boldsymbol{n}_o \tag{12}$$

$$y_{ir} = \left\| \begin{bmatrix} x(t_i) \\ y(t_i) \end{bmatrix} - \boldsymbol{\ell}_{ij} \right\|_2 + n_r \tag{13}$$

where $\boldsymbol{y}_{io}$ consists of robot-oriented distance and heading difference between t_i and t_{i-1}, and y_{ir} is the distance between the robot and a specific landmark $\boldsymbol{\ell}_{ij}$ at t_i. The results are shown in Fig. 1b. In Fig. 2, we compare the performance of the periodic relinearization approach, a naive periodic relinearization with variable reordering approach, and the Bayes tree approach from Section 3.

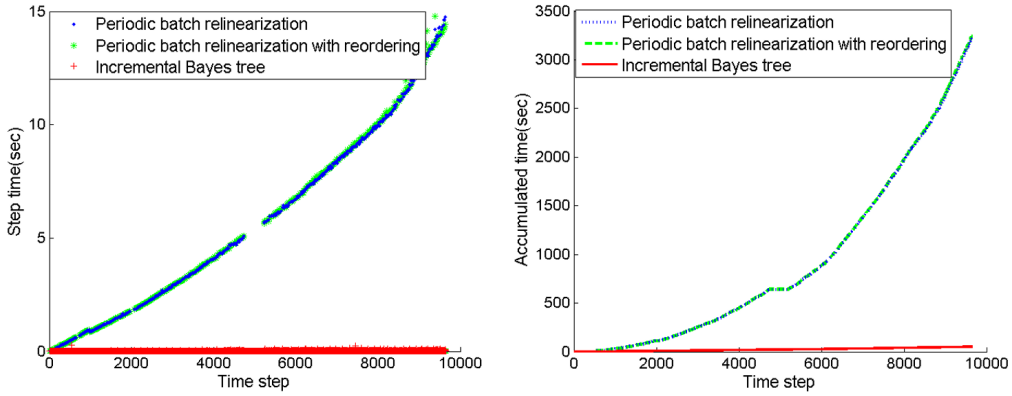
4.2 Autonomous Lawnmower

In our second experiment, we applied all three approaches to a freely available range-only SLAM dataset collected from an autonomous lawn-mowing robot [10]. The environment, including the



(a) True trajectory and landmarks (b) Estimated trajectory and landmarks

Figure 3: Solving the STEAM problem on a real dataset. Ground truth (a) and estimate (b) of "Plaza" dataset. In (a), the lines between trajectory points and landmarks represent range measurements. The range measurements are sparse, with approximately 11 time steps (and up to 500 steps) between range readings for the worst landmark. (b) shows that ground truth and estimate are very close.



(a) Comparison of time spent on each time step (b) Comparison of accumulated time

Figure 4: Comparison of time spent on each time step and accumulated time. There are 4 landmarks, which is extremely small relative to the number of process states, so no performance gain is observed from variable reordering. However, in the Bayes tree-based approach, which is fully incremental, the update of each iteration has very limited local effects and the computation time almost remains the same even with a large number of process states. The gap around time step 5,000 in (a) is caused by the vacancy of range measurements in that time period.

locations of the landmarks and the ground truth paths, are shown in Fig. 3a. While collecting the dataset the robot travelled 1.9km, occupied 9,658 poses, and received 3,529 range measurements. The path is a commonly used path pattern in lawn mowing applications. This dataset is very sparse, with approximately 11 time steps (and up to 500 steps) between range readings for the worst landmark. Besides range measurements, the dataset also contains odometry measurements at each time step. And all the measurements are considered in the same manner as in Section 4.1. The results of our algorithm are shown in Fig. 3b and demonstrate that we are able to accurately estimate the robot's trajectory and map.

Performance of three approaches – periodic batch relinearization, periodic batch relinearization with variable reordering, and incremental Bayes tree are compared in Fig. 4. In all cases, as in Section 4.1, a new iteration, i.e. an update to the combined state estimate, is carried out when 5 new range measurements have accumulated. In this dataset, the number of landmarks are 4, which is extremely small relative to the number of process states, so there is no performance gain from reordering. However, the Bayes tree-based approach dramatically outperforms the other two approaches. As the problem size increases, there is negligible increase in computation time, even for close to 10,000 process states.

References

[1] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Reasearch*, 25:2006, 2006.

[2] M. Kaess, A. Ranganathan, and F. Dellaert. isam: Incremental smoothing and mapping. *Robotics, IEEE Transactions on*, 24(6):1365–1378, Dec 2008.

[3] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research, IJRR*, 31(2):217–236, Feb 2012.

[4] Michael Kaess, Viorela Ila, Richard Roberts, and Frank Dellaert. The bayes tree: An algorithmic foundation for probabilistic robot mapping. In *Algorithmic Foundations of Robotics IX*, pages 157–173. Springer, 2011.

[5] Chi Hay Tong, Paul Furgale, and Timothy D Barfoot. Gaussian process gauss–newton for non-parametric simultaneous localization and mapping. *The International Journal of Robotics Research*, 32(5):507–525, 2013.

[6] Tim Barfoot, Chi Hay Tong, and Simo Sarkka. Batch continuous-time trajectory estimation as exactly sparse gaussian process regression. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[7] J. E. Dennis, Jr. and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations (Classics in Applied Mathematics, 16)*. Soc for Industrial & Applied Math, 1996.

[8] Pinar Heggernes and Pontus Matstoms. Finding good column orderings for sparse qr factorization. Technical report, In Second SIAM Conference on Sparse Matrices, 1996.

[9] Timothy A. Davis, John R. Gilbert, Stefan I. Larimore, and Esmond G. Ng. Algorithm 836: Colamd, a column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):377–380, September 2004.

[10] Joseph Djugash. *Geolocation with Range: Robustness, Efficiency and Scalability*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, November 2010.