Batch Continuous-Time Trajectory Estimation as Exactly Sparse Gaussian Process Regression

Timothy D. Barfoot
University of Toronto, Canada
<tim.barfoot@utoronto.ca>

Chi Hay Tong
University of Oxford, UK
<chi@robots.ox.ac.uk>

Simo Särkkä
Aalto University, Finland
<simo.sarkka@aalto.fi>

***Abstract*—In this paper, we revisit batch state estimation through the lens of *Gaussian process* (GP) regression. We consider continuous-discrete estimation problems wherein a trajectory is viewed as a one-dimensional GP, with time as the independent variable. Our continuous-time prior can be defined by any linear, time-varying stochastic differential equation driven by white noise; this allows the possibility of smoothing our trajectory estimates using a variety of vehicle dynamics models (e.g., 'constant-velocity'). We show that this class of prior results in an inverse kernel matrix (i.e., covariance matrix between all pairs of measurement times) that is exactly sparse (block-tridiagonal) and that this can be exploited to carry out GP regression (and interpolation) very efficiently. Though the prior is continuous, we consider measurements to occur at discrete times. When the measurement model is also linear, this GP approach is equivalent to classical, discrete-time smoothing (at the measurement times). When the measurement model is nonlinear, we iterate over the whole trajectory (as is common in vision and robotics) to maximize accuracy. We test the approach experimentally on a simultaneous trajectory estimation and mapping problem using a mobile robot dataset.**

I. Introduction

Probabilistic state estimation has been a core topic in mobile robotics since the 1980s [11, 39, 40], often as part of the *simultaneous localization and mapping* (SLAM) problem [2, 10]. Early work in estimation theory focused on recursive (as opposed to batch) formulations [23], and this was mirrored in the formulation of SLAM as a filtering problem [40]. However, despite the fact that continuous-time estimation techniques have been available since the 1960s [20, 24], trajectory estimation for mobile robots has been formulated almost exclusively in discrete time.

Lu and Milios [30] showed how to formulate SLAM as a batch estimation problem incorporating both odometry measurements (to smooth solutions) as well as landmark measurements. This can be viewed as a generalization of *bundle adjustment* [5, 38], which did not incorporate odometry. Today, batch approaches in mobile robotics are commonplace (e.g., GraphSLAM by Thrun and Montemerlo [44]). Kaess et al. [21] show how batch solutions can be efficiently updated as new measurements are gathered and Strasdat et al. [43] show that batch methods are able to achieve higher accuracy than their filtering counterparts, for the same computational cost. Most of these results are formulated in discrete time.

Discrete-time representations of robot trajectories are sufficient in many situations, but they do not work well when estimating motion from certain types of sensors (e.g., rolling-shutter cameras and scanning laser-rangefinders) and sensor combinations (e.g., high datarate, asynchronous). In these cases, a smooth, continuous-time representation of the trajectory is more suitable. For example, in the case of estimating motion from a scanning-while-moving sensor, a discrete-time approach (with no motion prior) can fail to find a unique solution; something is needed to tie together the observations acquired at many unique timestamps. Additional sensors (e.g., odometry or inertial measurements) could be introduced to serve in this role, but this may not always be possible. In these cases, a motion prior can be used instead (or as well), which is most naturally expressed in continuous time.

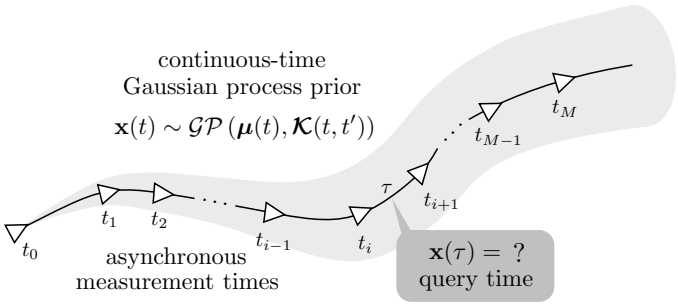


Fig. 1: To carry out batch trajectory estimation, we use GP regression with a smooth, continuous-time prior and discrete-time measurements. This allows us to query the trajectory at any time of interest, τ.

One approach to continuous-time trajectory representation is to use interpolation (e.g., linear, spline) directly between nearby discrete poses [3, 4, 9, 15, 19, 29]. Instead, we choose to represent the trajectory nonparametrically as a one-dimensional Gaussian process (GP) [33], with time as the independent variable (see Figure 1). Tong et al. [45, 46] show that querying the state of the robot at a time of interest can be viewed as a nonlinear, GP regression problem. While their approach is very general, allowing a variety of GP priors over robot trajectories, it is also quite expensive due to the need to invert a large, dense kernel matrix.

While GPs have also been used in robotic state estimation to accomplish dimensionality reduction [13, 14, 27] and to represent the measurement and motion models [7, 25, 26], these uses are quite different than representing the latent robot trajectory as a GP [45, 46].

In this paper, we consider a particular class of GPs (generated by linear, time-varying (LTV) stochastic differential

equations (SDE) driven by white noise) whereupon the inverse kernel matrix is *exactly sparse* (block-tridiagonal) and can be derived in closed form. Concentrating on this class of covariance functions results in only a minor loss of generality, because many commonly used covariance functions such the Matérn class and the squared exponential covariance function can be exactly or approximately represented as linear SDEs [17, 37, 41]. We provide an example of this relationship at the end of this paper. The resulting sparsity allows the approach of Tong et al. [45, 46] to be implemented very efficiently. The intuition behind why this is possible is that the state we are estimating is *Markovian* for this class of GPs, which implies that the corresponding precision matrices are sparse [28].

This sparsity property has been exploited in estimation theory to allow recursive methods (both filtering and smoothing) since the 1960s [23, 24]. The tracking literature, in particular, has made heavy use of motion priors (in both continuous and discrete time) and has exploited the Markov property for efficient solutions [31]. In vision and robotics, discrete-time batch methods commonly exploit this sparsity property as well [47]. In this paper, we make the (retrospectively obvious) observation that this sparsity can also be exploited in a batch, continuous-time context. The result is that we derive a principled method to construct trajectory-smoothing terms for batch optimization (or factors in a factor-graph representation) based on a class of useful motion models; this paves the way to incorporate vehicle dynamics models, including exogenous inputs, to help with trajectory estimation.

Therefore, our main contribution is to emphasize the strong connection between classical estimation theory and machine learning via GP regression. We use the fact that the inverse kernel matrix is sparse for a class of useful GP priors [28, 37] in a new way to efficiently implement nonlinear, GP regression for batch, continuous-time trajectory estimation. We also show that this naturally leads to a subtle generalization of SLAM that we call *simultaneous trajectory estimation and mapping* (STEAM), with the difference being that chains of discrete *poses* are replaced with *Markovian trajectories* in order to incorporate continuous-time motion priors in an efficient way. Finally, by using this GP paradigm, we are able to exploit the classic GP interpolation approach to query the trajectory at any time of interest in an efficient manner.

This ability to query the trajectory at any time of interest in a principled way could be useful in a variety of situations. For example, Newman et al. [32] mapped a large urban area using a combination of stereo vision and laser rangefinders; the motion was estimated using the camera and the laser data were subsequently placed into a three-dimensional map based on this estimated motion. Our method could provide a seamless means to (i) estimate the camera trajectory and then (ii) query this trajectory at every laser acquisition time.

The paper is organized as follows. Section II summarizes the general approach to batch state estimation via GP regression. Section III describes the particular class of GPs we use and elaborates on our main result concerning sparsity. Section IV demonstrates this main result on a mobile robot example using a 'constant-velocity' prior and compares the computational cost to methods that do not exploit the sparsity. Section V provides some discussion and Section VI concludes the paper.

II. Gaussian Process Regression

We take a *Gaussian-process-regression* approach to state estimation. This allows us to (i) represent trajectories in continuous time (and therefore query the solution at any time of interest), and (ii) optimize our solution by iterating over the entire trajectory (recursive methods typically iterate at a single timestep).

We will consider systems with a continuous-time, GP process model and a discrete-time, nonlinear measurement model:

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

$$\mathbf{y}_i = \mathbf{g}(\mathbf{x}(t_i)) + \mathbf{n}_i, \qquad t_1 < \cdots < t_M, \tag{2}$$

where $\mathbf{x}(t)$ is the state, $\boldsymbol{\mu}(t)$ is the mean function, $\boldsymbol{\mathcal{K}}(t,t')$ is the covariance function, $\mathbf{y}_i$ are measurements, $\mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i)$ is Gaussian measurement noise, $\mathbf{g}(\cdot)$ is a nonlinear measurement model, and $t_1 < \ldots < t_M$ is a sequence of measurement times. For the moment, we do not consider the STEAM problem (i.e., the state does not include landmarks), but we will return to this case in our example later.

We follow the approach of Tong et al. [46] to set up our batch, GP state estimation problem. We will first assume that we want to query the state at the measurement times, and will return to querying at other times later on. We start with an initial guess, $\bar{\mathbf{x}}$, for the trajectory that will be improved iteratively. At each iteration, we solve for the optimal perturbation, $\delta\mathbf{x}^\star$, to our guess using GP regression, with our measurement model linearized about the current best guess.

If we let $\mathbf{x} = \bar{\mathbf{x}} + \delta\mathbf{x}$, the joint likelihood between the state perturbation and the measurements (both at the measurement times) is

$$p\left(\begin{bmatrix}\delta\mathbf{x}\\ \mathbf{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}-\bar{\mathbf{x}}\\ \mathbf{g}+\mathbf{C}(\boldsymbol{\mu}-\bar{\mathbf{x}})\end{bmatrix}, \begin{bmatrix}\boldsymbol{\mathcal{K}} & \boldsymbol{\mathcal{K}}\mathbf{C}^T\\ \mathbf{C}\boldsymbol{\mathcal{K}}^T & \mathbf{C}\boldsymbol{\mathcal{K}}\mathbf{C}^T+\mathbf{R}\end{bmatrix}\right), \tag{3}$$

where

$$\delta\mathbf{x} = \begin{bmatrix}\delta\mathbf{x}(t_0)\\ \vdots\\ \delta\mathbf{x}(t_M)\end{bmatrix}, \quad \bar{\mathbf{x}} = \begin{bmatrix}\bar{\mathbf{x}}(t_0)\\ \vdots\\ \bar{\mathbf{x}}(t_M)\end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix}\boldsymbol{\mu}(t_0)\\ \vdots\\ \boldsymbol{\mu}(t_M)\end{bmatrix},$$

$$\mathbf{y} = \begin{bmatrix}\mathbf{y}_1\\ \vdots\\ \mathbf{y}_M\end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix}\mathbf{g}(\bar{\mathbf{x}}(t_1))\\ \vdots\\ \mathbf{g}(\bar{\mathbf{x}}(t_M))\end{bmatrix}, \quad \mathbf{C} = \left.\frac{\partial\mathbf{g}}{\partial\mathbf{x}}\right|_{\bar{\mathbf{x}}},$$

$$\mathbf{R} = \operatorname{diag}(\mathbf{R}_1, \ldots, \mathbf{R}_M), \quad \boldsymbol{\mathcal{K}} = \left[\boldsymbol{\mathcal{K}}(t_i,t_j)\right]_{ij}.$$

Note, we have linearized the measurement model about our best guess so far. We then have that

$$p(\delta\mathbf{x}|\mathbf{y}) = \mathcal{N}\Big(\left(\boldsymbol{\mathcal{K}}^{-1}+\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\right)^{-1}\left(\boldsymbol{\mathcal{K}}^{-1}(\boldsymbol{\mu}-\bar{\mathbf{x}})\right.$$
$$\left.+\mathbf{C}^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{g})\right), \left(\boldsymbol{\mathcal{K}}^{-1}+\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\right)^{-1}\Big), \tag{4}$$

or, by rearranging the mean expression using the Sherman-Morrison-Woodbury identity, we have a linear system for $\delta\mathbf{x}^\star$

(the mean of the perturbation):

$$\left(\boldsymbol{\mathcal{K}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\right)\,\delta\mathbf{x}^\star = \boldsymbol{\mathcal{K}}^{-1}(\boldsymbol{\mu}-\bar{\mathbf{x}}) + \mathbf{C}^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{g}), \quad (5)$$

which can be viewed as the solution to the associated *maximum a posteriori* (MAP) problem. We know that the $\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}$ term in (5) is block-diagonal (assuming each measurement depends on the state at a single time), but in general $\boldsymbol{\mathcal{K}}^{-1}$ could be dense, depending on the choice of GP prior. At each iteration, we solve for $\delta\mathbf{x}^\star$ and then update the guess according to $\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} + \delta\mathbf{x}^\star$. This is effectively Gauss-Newton optimization over the whole trajectory.

We may want to also query the state at some other time(s) of interest (in addition to the measurement times). Though we could jointly estimate the trajectory at the measurement and query times, a better idea is to use GP interpolation after the solution at the measurement times converges [33, 46] (see Section III-D for more details). GP interpolation automatically picks the correct interpolation scheme for a given prior; it arrives at the same answer as the joint approach (in the linear case), but at lower computational cost.

In general, this GP approach has complexity $O(M^3 + M^2N)$, where M is the number of measurement times and N is the number of query times (the initial solve is $O(M^3)$ and the query is $O(M^2N)$). This is quite expensive, and therefore we will seek to improve the cost by exploiting the structure of the matrices involved under a particular class of GP priors.

III. A Class of Exactly Sparse GP Priors

A. Linear, Time-Varying Stochastic Differential Equations

We now show that the inverse kernel matrix is exactly sparse for a particular class of useful GP priors. We consider GPs generated by linear, time-varying (LTV) stochastic differential equations (SDE) of the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{v}(t) + \mathbf{F}(t)\mathbf{w}(t), \quad (6)$$

where $\mathbf{x}(t)$ is the state, $\mathbf{v}(t)$ is a (known) exogenous input, $\mathbf{w}(t)$ is white process noise, and $\mathbf{A}(t)$, $\mathbf{F}(t)$ are time-varying system matrices. The process noise is given by

$$\mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_C\,\delta(t-t')), \quad (7)$$

a (stationary) zero-mean *Gaussian process* (GP) with (symmetric, positive-definite) *power-spectral density matrix*, $\mathbf{Q}_C$, and $\delta(\cdot)$ is the *Dirac delta function*.

The general solution to this LTV SDE [31, 42] is

$$\mathbf{x}(t) = \boldsymbol{\Phi}(t,t_0)\mathbf{x}(t_0) + \int_{t_0}^{t} \boldsymbol{\Phi}(t,s)\left(\mathbf{v}(s) + \mathbf{F}(s)\mathbf{w}(s)\right)\,ds, \quad (8)$$

where $\boldsymbol{\Phi}(t,s)$ is known as the *transition matrix*. From this model, we seek the mean and covariance functions for $\mathbf{x}(t)$.

B. Mean Function

For the mean function, we take the expected value of (8):

$$\boldsymbol{\mu}(t) = E[\mathbf{x}(t)] = \boldsymbol{\Phi}(t,t_0)\boldsymbol{\mu}_0 + \int_{t_0}^{t} \boldsymbol{\Phi}(t,s)\mathbf{v}(s)\,ds, \quad (9)$$

where $\boldsymbol{\mu}_0 = \boldsymbol{\mu}(t_0)$ is the initial value of the mean. If we now have a sequence of measurement times, $t_0 < t_1 < t_2 < \cdots < t_M$, then we can write the mean at these times in *lifted form* as

$$\boldsymbol{\mu} = \mathbf{A}\mathbf{v}, \quad (10)$$

where

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}(t_0) \\ \boldsymbol{\mu}(t_1) \\ \vdots \\ \boldsymbol{\mu}(t_M) \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \boldsymbol{\mu}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_M \end{bmatrix}, \quad \mathbf{v}_i = \int_{t_{i-1}}^{t_i} \boldsymbol{\Phi}(t_i,s)\mathbf{v}(s)\,ds, \quad (11)$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\Phi}(t_1,t_0) & \mathbf{1} & \cdots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\Phi}(t_2,t_0) & \boldsymbol{\Phi}(t_2,t_1) & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\Phi}(t_{M-1},t_0) & \boldsymbol{\Phi}(t_{M-1},t_1) & \cdots & \mathbf{1} & \mathbf{0} \\ \boldsymbol{\Phi}(t_M,t_0) & \boldsymbol{\Phi}(t_M,t_1) & \cdots & \boldsymbol{\Phi}(t_M,t_{M-1}) & \mathbf{1} \end{bmatrix}.$$

Note that $\mathbf{A}$, the *lifted transition matrix*, is lower-triangular. We arrive at this form by simply splitting up (9) into a sum of integrals between each pair of measurement times.

C. Covariance Function

For the covariance function, we take the second moment of (8) to arrive at

$$\begin{aligned} \boldsymbol{\mathcal{K}}(t,t') &= E\left[(\mathbf{x}(t)-\boldsymbol{\mu}(t))(\mathbf{x}(t')-\boldsymbol{\mu}(t'))^T\right] \\ &= \boldsymbol{\Phi}(t,t_0)\boldsymbol{\mathcal{K}}_0\boldsymbol{\Phi}(t',t_0)^T \\ &\quad + \int_{t_0}^{\min(t,t')} \boldsymbol{\Phi}(t,s)\mathbf{F}(s)\mathbf{Q}_C\mathbf{F}(s)^T\boldsymbol{\Phi}(t',s)^T\,ds, \end{aligned} \quad (12)$$

where $\boldsymbol{\mathcal{K}}_0$ is the initial covariance at t_0 and we have assumed $E[\mathbf{x}(t_0)\mathbf{w}(t)^T] = \mathbf{0}$. Using a sequence of measurement times, $t_0 < t_1 < t_2 < \cdots < t_M$, we can write the covariance between two times as

$$\boldsymbol{\mathcal{K}}(t_i,t_j) = \begin{cases} \boldsymbol{\Phi}(t_i,t_j)\left(\sum_{n=0}^{j}\boldsymbol{\Phi}(t_j,t_n)\mathbf{Q}_n\boldsymbol{\Phi}(t_j,t_n)^T\right) & t_j < t_i \\ \sum_{n=0}^{i}\boldsymbol{\Phi}(t_i,t_n)\mathbf{Q}_n\boldsymbol{\Phi}(t_i,t_n)^T & t_i = t_j \\ \left(\sum_{n=0}^{i}\boldsymbol{\Phi}(t_i,t_n)\mathbf{Q}_n\boldsymbol{\Phi}(t_i,t_n)^T\right)\boldsymbol{\Phi}(t_j,t_i)^T & t_i < t_j \end{cases} \quad (13)$$

where

$$\mathbf{Q}_i = \int_{t_{i-1}}^{t_i} \boldsymbol{\Phi}(t_i,s)\mathbf{F}(s)\mathbf{Q}_C\mathbf{F}(s)^T\boldsymbol{\Phi}(t_i,s)^T\,ds, \quad (14)$$

for $i = 1 \ldots M$ and $\mathbf{Q}_0 = \boldsymbol{\mathcal{K}}_0$ (to keep the notation simple). Given this preparation, we are now ready to state the main sparsity result that we will exploit in the rest of the paper.

Lemma 1. *Let $t_0 < t_1 < t_2 < \cdots < t_M$ be a monotonically increasing sequence of time values. Using* (13)*, we define the $(M+1)\times(M+1)$ kernel matrix (i.e., the prior covariance matrix between all pairs of times), $\boldsymbol{\mathcal{K}} = [\boldsymbol{\mathcal{K}}(t_i,t_j)]_{ij}$. Then, we can factor $\boldsymbol{\mathcal{K}}$ according to a lower-diagonal-upper decomposition,*

$$\boldsymbol{\mathcal{K}} = \mathbf{A}\mathbf{Q}\mathbf{A}^T, \quad (15)$$

where $\mathbf{A}$ is the lower-triangular matrix given in (11) *and $\mathbf{Q} = \operatorname{diag}(\boldsymbol{\mathcal{K}}_0, \mathbf{Q}_1, \ldots, \mathbf{Q}_M)$ with $\mathbf{Q}_i$ given in* (14)*.*

Proof: Straightforward to verify by substitution. ■

Theorem 1. *The inverse of the kernel matrix constructed in Lemma 1, $\boldsymbol{\mathcal{K}}^{-1}$, is exactly sparse (block-tridiagonal).*

Proof: The decomposition of $\boldsymbol{\mathcal{K}}$ in Lemma 1 provides

$$\boldsymbol{\mathcal{K}}^{-1} = (\mathbf{A}\mathbf{Q}\mathbf{A}^T)^{-1} = \mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1}. \tag{16}$$

where the inverse of the lifted transition matrix is

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ -\boldsymbol{\Phi}(t_1,t_0) & \mathbf{1} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{\Phi}(t_2,t_1) & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \cdots & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & -\boldsymbol{\Phi}(t_M,t_{M-1}) & \mathbf{1} \end{bmatrix}, \tag{17}$$

and $\mathbf{Q}^{-1}$ is block-diagonal. The block-tridiagonal property of $\boldsymbol{\mathcal{K}}^{-1}$ follows by substitution and multiplication. ■

While the block-tridiagonal property stated in Theorem 1 has been exploited in vision and robotics for a long time [30, 47, 44], the usual route to this point is to begin by converting the continuous-time motion model to discrete time and then to directly formulate a maximum a posteriori optimization problem; this bypasses writing out the full expression for $\boldsymbol{\mathcal{K}}$ and jumps to an expression for $\boldsymbol{\mathcal{K}}^{-1}$. However, we require expressions for both $\boldsymbol{\mathcal{K}}$ and $\boldsymbol{\mathcal{K}}^{-1}$ to carry out our GP reinterpretation and facilitate querying the trajectory at an arbitrary time (through interpolation). That said, it is also worth noting we have not needed to convert the motion model to discrete time and have made no approximations thus far.

Given the above results, the prior over the state (at the measurement times) can be written as

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\mathcal{K}}) = \mathcal{N}\left(\mathbf{A}\mathbf{v}, \mathbf{A}\mathbf{Q}\mathbf{A}^T\right). \tag{18}$$

More importantly, using the result of Theorem 1 in (5) gives

$$\overbrace{\left(\mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\right)}^{\text{block-tridiagonal}} \delta\mathbf{x}^\star = \mathbf{A}^{-T}\mathbf{Q}^{-1}(\mathbf{v} - \mathbf{A}^{-1}\bar{\mathbf{x}}) + \mathbf{C}^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{g}). \tag{19}$$

which can be solved in $O(M)$ time (at each iteration), using a sparse solver (e.g., sparse Cholesky decomposition then forward-backward passes). In fact, although we do not have space to show it, in the case of a linear measurement model, one such solver is the classical, forward-backward smoother (i.e., Kalman or Rauch–Tung–Striebel smoother). Put another way, the forward-backward smoother is possible *because* of the sparse structure of (19). For nonlinear measurement models, our scheme iterates over the whole trajectory; it is therefore related to, but not the same as, the 'extended' version of the forward-backward smoother [35, 36, 37].

Perhaps the most interesting outcome of Theorem 1 is that, although we are using a continuous-time prior to smooth our trajectory, at implementation we require only $M + 1$ smoothing terms in the associated MAP optimization problem: M between consecutive pairs of measurement times plus 1 at the initial time (unless we are also estimating a map). As mentioned before, this is the same form that we would have arrived at had we started by converting our motion model to discrete time at the beginning [30, 47, 44]. This equivalence has been noticed before for recursive solutions to estimation problems with a continuous-time state and discrete-time measurements [34], but not in the batch scenario. Figure 2 depicts the $M+1$ smoothing terms in a factor-graph representation of the prior [8, 22].

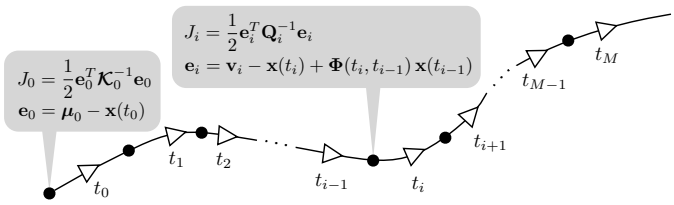


Fig. 2: Although we began with a continuous-time prior to smooth our trajectory, the class of exactly sparse GPs results in only $M+1$ smoothing terms, J_i, in the associated optimization problem, the solution of which is (19). We can depict these graphically as factors (black dots) in a factor-graph representation of the prior [8]. The triangles are *trajectory states*, the nature of which depends on the choice of prior.

However, while the form of the smoothing terms/factors is similar to the original discrete-time form introduced by Lu and Milios [30], our approach provides a principled method for their construction, starting from the continuous-time motion model. Critically, we stress that the state being estimated must be *Markovian* in order to obtain the desirable sparse structure. In the experiment section, we will investigate a common GP prior, namely the 'constant-velocity' or white-noise-on-acceleration model: $\ddot{\mathbf{p}}(t) = \mathbf{w}(t)$, where $\mathbf{p}(t)$ represents position. For this model, $\mathbf{p}(t)$ is not Markovian, but

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{p}(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix}, \tag{20}$$

is. This implies that, if we want to use the 'constant-velocity' prior and enjoy the sparse structure without approximation, we must estimate a stacked state with both position and velocity. Marginalizing out the velocity variables fills in the inverse kernel matrix, thereby destroying the sparsity.

If all we cared about was estimating the value of the state at the measurement times, our GP paradigm arguably offers little beyond a reinterpretation of the usual discrete-time approach to batch estimation. However, by taking the time to set up the problem in this manner, we can now query the trajectory at *any* time of interest using the classic interpolation scheme that is inherent to GP regression [46].

D. Querying the Trajectory

As discussed in Section II, after we solve for the trajectory at the measurement times, we may want to query it at other times of interest. This operation also benefits greatly from the sparse structure. To keep things simple, we consider a single query time, $t_i \leq \tau < t_{i+1}$ (see Figure 1). The standard linear

GP interpolation formula [33, 46] is

$$\bar{\mathbf{x}}(\tau) = \boldsymbol{\mu}(\tau) + \boldsymbol{\mathcal{K}}(\tau)\boldsymbol{\mathcal{K}}^{-1}(\bar{\mathbf{x}} - \boldsymbol{\mu}), \tag{21}$$

where $\boldsymbol{\mathcal{K}}(\tau) = \begin{bmatrix}\boldsymbol{\mathcal{K}}(\tau, t_0) & \cdots & \boldsymbol{\mathcal{K}}(\tau, t_M)\end{bmatrix}$.

For the mean function at the query time, we simply have

$$\boldsymbol{\mu}(\tau) = \boldsymbol{\Phi}(\tau, t_i)\boldsymbol{\mu}_i + \int_{t_i}^{\tau} \boldsymbol{\Phi}(\tau, s)\mathbf{v}(s)\, ds, \tag{22}$$

which can be evaluated in $O(1)$ time.

The computational savings come from the sparsity of the product $\boldsymbol{\mathcal{K}}(\tau)\boldsymbol{\mathcal{K}}^{-1}$, which represents the burden of the cost in the interpolation formula. After some effort, it turns out we can write $\boldsymbol{\mathcal{K}}(\tau)$ as

$$\boldsymbol{\mathcal{K}}(\tau) = \mathbf{V}(\tau)\mathbf{A}^T, \tag{23}$$

where $\mathbf{A}$ was defined before,

$$\begin{aligned}\mathbf{V}(\tau) = \Big[&\boldsymbol{\Phi}(\tau, t_i)\boldsymbol{\Phi}(t_i, t_0)\boldsymbol{\mathcal{K}}_0 \quad \boldsymbol{\Phi}(\tau, t_i)\boldsymbol{\Phi}(t_i, t_1)\mathbf{Q}_1 \quad \cdots \\ &\cdots \quad \boldsymbol{\Phi}(\tau, t_i)\boldsymbol{\Phi}(t_i, t_{i-1})\mathbf{Q}_{i-1} \quad \boldsymbol{\Phi}(\tau, t_i)\mathbf{Q}_i \quad \cdots \\ &\cdots \quad \mathbf{Q}_\tau\boldsymbol{\Phi}(t_{i+1}, \tau)^T \quad \mathbf{0} \quad \cdots \quad \mathbf{0}\Big],\end{aligned} \tag{24}$$

and

$$\mathbf{Q}_\tau = \int_{t_i}^{\tau} \boldsymbol{\Phi}(\tau, s)\mathbf{F}(s)\mathbf{Q}_C\mathbf{F}(s)^T\boldsymbol{\Phi}(\tau, s)^T\, ds. \tag{25}$$

Returning to the desired product, we have

$$\boldsymbol{\mathcal{K}}(\tau)\boldsymbol{\mathcal{K}}^{-1} = \mathbf{V}(\tau)\underbrace{\mathbf{A}^T\mathbf{A}^{-T}}_{\mathbf{1}}\mathbf{Q}^{-1}\mathbf{A}^{-1} = \mathbf{V}(\tau)\mathbf{Q}^{-1}\mathbf{A}^{-1}. \tag{26}$$

Since $\mathbf{Q}^{-1}$ is block-diagonal, and $\mathbf{A}^{-1}$ has only the main diagonal and the one below it non-zero, we can evaluate the product very efficiently. The result can be seen in Equation (27) at the bottom of the page, which has exactly two non-zero block-columns. Inserting this into (21), we have

$$\bar{\mathbf{x}}(\tau) = \boldsymbol{\mu}(\tau) + \boldsymbol{\Lambda}(\tau)(\bar{\mathbf{x}}_i - \boldsymbol{\mu}_i) + \boldsymbol{\Psi}(\tau)(\bar{\mathbf{x}}_{i+1} - \boldsymbol{\mu}_{i+1}), \tag{28}$$

which is a linear combination of just the two terms from t_i and t_{i+1}. If the query time is beyond the last measurement time, $t_M < \tau$, the expression will involve only the term at t_M and represents extrapolation/prediction rather than interpolation/smoothing. In summary, to query the trajectory at a single time of interest is $O(1)$ complexity.

E. Training the Hyperparameters

As with any GP regression, we have *hyperparameters* associated with our covariance function, namely $\mathbf{Q}_C$, which affect the smoothness and length scale of the class of functions we are considering as motion priors. The covariances of the measurement noises can also be unknown or uncertain. The standard approach to selecting these parameters is to use a training dataset (with groundtruth), and perform optimization using the log marginal likelihood (log-evidence) or its approximation as the objective function [33]. Fortunately, in the present case, the computation of the log marginal likelihood can also be done efficiently due to the sparseness of the inverse kernel matrix (not shown).

Alternatively, since these parameters have physical meaning, they can be computed directly from the training data. In our experiments, we obtained $\mathbf{Q}_C$ by modelling it as a diagonal matrix, and fitting Gaussians to the state accelerations. The measurement noise properties were determined from the training data in a similar manner.

F. Complexity

We conclude this section with a brief discussion of the time complexity of the overall algorithm when exploiting the sparse structure. If we have M measurement times and want to query the trajectory at N additional times of interest, the complexity of the resulting algorithm using GP regression with *any* linear, time-varying process model driven by white noise will be $O(M + N)$. This is broken into the two major steps as follows. The initial solution to find $\bar{\mathbf{x}}$ (at the measurement times) can be done in $O(M)$ time (per iteration) owing to the block-tridiagonal structure discussed earlier. Then, the queries at N other times of interest is $O(N)$ since each individual query is $O(1)$. Clearly, $O(M+N)$ is a big improvement over the $O(M^3 + M^2N)$ cost when we did not exploit the sparse structure of the problem.

We could consider lumping all the measurement and query times together into a larger set during the initial solve to avoid querying the trajectory after the fact; this would also be $O(M + N)$. Tong et al. [46] also discuss a scheme to remove some of the measurement times from the initial solve, which could further reduce cost but with some loss of accuracy. Some further experimentation is necessary to better understand which approach is best in which situation.

IV. Mobile Robot Example

A. Constant-Velocity GP Prior

We will demonstrate the advantages of the sparse structure through an example employing the 'constant-velocity' prior, $\ddot{\mathbf{p}}(t) = \mathbf{w}(t)$. This can be expressed as a linear, time-invariant SDE of the form in (6) with

$$\mathbf{x}(t) = \begin{bmatrix}\mathbf{p}(t) \\ \dot{\mathbf{p}}(t)\end{bmatrix},\ \mathbf{A}(t) = \begin{bmatrix}\mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0}\end{bmatrix},\ \mathbf{v}(t) = \mathbf{0},\ \mathbf{F}(t) = \begin{bmatrix}\mathbf{0} \\ \mathbf{1}\end{bmatrix}, \tag{29}$$

where $\mathbf{p}(t) = \begin{bmatrix}x(t) & y(t) & \theta(t)\end{bmatrix}^T$ is the pose and $\dot{\mathbf{p}}(t)$ is the pose rate. In this case, the transition function is

$$\boldsymbol{\Phi}(t, s) = \begin{bmatrix}\mathbf{1} & (t-s)\mathbf{1} \\ \mathbf{0} & \mathbf{1}\end{bmatrix}, \tag{30}$$

$$\boldsymbol{\mathcal{K}}(\tau)\boldsymbol{\mathcal{K}}^{-1} = \Big[\mathbf{0} \quad \cdots \quad \mathbf{0} \quad \underbrace{\boldsymbol{\Phi}(\tau, t_i) - \mathbf{Q}_\tau\boldsymbol{\Phi}(t_{i+1}, \tau)^T\mathbf{Q}_{i+1}^{-1}\boldsymbol{\Phi}(t_{i+1}, t_i)}_{\boldsymbol{\Lambda}(\tau),\ \text{block column } i} \quad \underbrace{\mathbf{Q}_\tau\boldsymbol{\Phi}(t_{i+1}, \tau)^T\mathbf{Q}_{i+1}^{-1}}_{\boldsymbol{\Psi}(\tau),\ \text{block column } i+1} \quad \mathbf{0} \quad \cdots \quad \mathbf{0}\Big], \tag{27}$$

which can be used to construct $\mathbf{A}$ (or $\mathbf{A}^{-1}$ directly). As we will be doing a STEAM example, we will constrain the first trajectory state to be $\mathbf{x}(t_0) = \mathbf{0}$ and so will have no need for $\boldsymbol{\mu}_0$ and $\boldsymbol{\mathcal{K}}_0$. For $i = 1 \dots M$, we have $\mathbf{v}_i = \mathbf{0}$ and

$$\mathbf{Q}_i = \begin{bmatrix} \frac{1}{3}\Delta t_i^3 \mathbf{Q}_C & \frac{1}{2}\Delta t_i^2 \mathbf{Q}_C \\ \frac{1}{2}\Delta t_i^2 \mathbf{Q}_C & \Delta t_i \mathbf{Q}_C \end{bmatrix}, \tag{31}$$

with $\Delta t_i = t_i - t_{i-1}$. The inverse blocks are

$$\mathbf{Q}_i^{-1} = \begin{bmatrix} 12\Delta t_i^{-3} \mathbf{Q}_C^{-1} & -6\Delta t_i^{-2} \mathbf{Q}_C^{-1} \\ -6\Delta t_i^{-2} \mathbf{Q}_C^{-1} & 4\Delta t_i^{-1} \mathbf{Q}_C^{-1} \end{bmatrix}, \tag{32}$$

so we can build $\mathbf{Q}^{-1}$ directly. We now have everything we need to represent the prior: $\mathbf{A}^{-1}$, $\mathbf{Q}^{-1}$, and $\mathbf{v} = \mathbf{0}$, which can be used to construct $\boldsymbol{\mathcal{K}}^{-1}$.

We will also augment the trajectory with a set of L landmarks, $\boldsymbol{\ell}$, into a combined state, $\mathbf{z}$, in order to consider the STEAM problem:

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\ell} \end{bmatrix}, \quad \boldsymbol{\ell} = \begin{bmatrix} \boldsymbol{\ell}_1 \\ \vdots \\ \boldsymbol{\ell}_L \end{bmatrix}, \quad \boldsymbol{\ell}_j = \begin{bmatrix} x_j \\ y_j \end{bmatrix}. \tag{33}$$

While others have folded velocity estimation into discrete-time, filter-based SLAM [6] and even discrete-time, batch SLAM [16], we are actually proposing something more general than this: the choice of prior tells us what to use for the trajectory states. And, although we solve for the state at a discrete number of measurement times, our setup is based on an underlying continuous-time prior, meaning that we can query it at any time of interest in a principled way.

B. Measurement Models

We will use two types of measurements: range/bearing to landmarks (using a laser rangefinder) and wheel odometry (in the form of robot-oriented velocity). The range/bearing measurement model takes the form

$$\begin{aligned} \mathbf{y}_{ij} &= \mathbf{g}_{\mathrm{rb}}(\mathbf{x}(t_i), \boldsymbol{\ell}_j) + \mathbf{n}_{ij} \\ &= \begin{bmatrix} \sqrt{(x_j - x(t_i))^2 + (y_j - y(t_i))} \\ \operatorname{atan2}(y_j - y(t_i), x_j - x(t_i)) \end{bmatrix} + \mathbf{n}_{ij}, \end{aligned} \tag{34}$$

and the wheel odometry measurement model takes the form

$$\mathbf{y}_i = \mathbf{g}_{\mathrm{wo}}(\mathbf{x}(t_i)) + \mathbf{n}_i = \begin{bmatrix} \cos\theta(t_i) & \sin\theta(t_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \dot{\mathbf{p}}(t_i) + \mathbf{n}_i, \tag{35}$$

which gives the longitudinal and rotational speeds of the robot. Note that the velocity information is extracted easily from the state since we are estimating it directly. In the interest of space, we omit the associated Jacobians.

C. Exploiting Sparsity

Figure 3 shows an illustration of the STEAM problem we are considering. In terms of linear algebra, at each iteration we need to solve a linear system of the form

$$\underbrace{\begin{bmatrix} \mathbf{W}_{xx} & \mathbf{W}_{\ell x}^T \\ \mathbf{W}_{\ell x} & \mathbf{W}_{\ell\ell} \end{bmatrix}}_{\mathbf{W}} \underbrace{\begin{bmatrix} \delta\mathbf{x}^\star \\ \delta\boldsymbol{\ell}^\star \end{bmatrix}}_{\delta\mathbf{z}^\star} = \underbrace{\begin{bmatrix} \mathbf{b}_x \\ \mathbf{b}_\ell \end{bmatrix}}_{\mathbf{b}}, \tag{36}$$

which retains exploitable structure despite introducing the landmarks. In particular, $\mathbf{W}_{xx}$ is block-tridiagonal (due to our GP prior) and $\mathbf{W}_{\ell\ell}$ is block-diagonal [5]; the sparsity of the off-diagonal block, $\mathbf{W}_{\ell x}$, depends on the specific landmark observations. We reiterate the fact that if we marginalize out the $\dot{\mathbf{p}}(t_i)$ variables and keep only the $\mathbf{p}(t_i)$ variables to represent the trajectory, the $\mathbf{W}_{xx}$ block becomes dense (for this prior); this is precisely the approach of Tong et al. [46].

To solve (36) efficiently, we can begin by either exploiting the sparsity of $\mathbf{W}_{xx}$ or of $\mathbf{W}_{\ell\ell}$. Since each trajectory variable represents a unique measurement time (range/bearing or odometry), there are potentially a lot more trajectory variables than landmark variables, $L \ll M$, so we will exploit $\mathbf{W}_{xx}$.

We use a sparse (lower-upper) Cholesky decomposition:

$$\underbrace{\begin{bmatrix} \mathbf{V}_{xx} & \mathbf{0} \\ \mathbf{V}_{\ell x} & \mathbf{V}_{\ell\ell} \end{bmatrix}}_{\mathbf{V}} \underbrace{\begin{bmatrix} \mathbf{V}_{xx}^T & \mathbf{V}_{\ell x}^T \\ \mathbf{0} & \mathbf{V}_{\ell\ell}^T \end{bmatrix}}_{\mathbf{V}^T} = \underbrace{\begin{bmatrix} \mathbf{W}_{xx} & \mathbf{W}_{\ell x}^T \\ \mathbf{W}_{\ell x} & \mathbf{W}_{\ell\ell} \end{bmatrix}}_{\mathbf{W}} \tag{37}$$

We first decompose $\mathbf{V}_{xx}\mathbf{V}_{xx}^T = \mathbf{W}_{xx}$, which can be done in $O(M)$ time owing to the block-tridiagonal sparsity. The resulting $\mathbf{V}_{xx}$ will have only the main block-diagonal and the one below it non-zero. This means we can solve $\mathbf{V}_{\ell x}\mathbf{V}_{xx}^T = \mathbf{W}_{\ell x}$ for $\mathbf{V}_{\ell x}$ in $O(LM)$ time. Finally, we decompose $\mathbf{V}_{\ell\ell}\mathbf{V}_{\ell\ell}^T = \mathbf{W}_{\ell\ell} - \mathbf{V}_{\ell x}\mathbf{V}_{\ell x}^T$, which we can do in $O(L^3 + L^2 M)$ time. This completes the decomposition in $O(L^3 + L^2 M)$ time. We then perform the standard forward-backward passes, ensuring to exploit the sparsity: first solve $\mathbf{V}\mathbf{d} = \mathbf{b}$ for $\mathbf{d}$, then $\mathbf{V}^T \delta\mathbf{z}^\star = \mathbf{d}$ for $\delta\mathbf{z}^\star$, both in $O(L^2 + LM)$ time. Note, this approach does not marginalize out any variables during the solve, as this can ruin the sparsity (i.e., we avoid inverting $\mathbf{W}_{xx}$). The whole solve is $O(L^3 + L^2 M)$.

At each iteration, we update the state according to $\bar{\mathbf{z}} \leftarrow \bar{\mathbf{z}} + \delta\mathbf{z}^\star$ and iterate to convergence. Finally, we query the trajectory at N other times of interest using the GP interpolation discussed earlier. The whole procedure is then $O(L^3 + L^2 M + N)$, including the extra queries.

D. Experiment

For experimental validation, we employed the same mobile robot dataset as used by Tong et al. [46]. This dataset consists

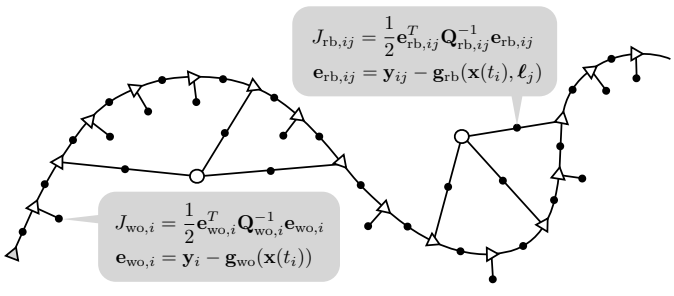


Fig. 3: Factor-graph representation of our STEAM problem. There are factors (black dots) for (i) the prior (binary), (ii) the landmark measurements (binary), and (iii) the wheel odometry measurements (unary). Triangles are *trajectory states* (position and velocity, for this prior); the first trajectory state is locked. Hollow circles are landmarks.

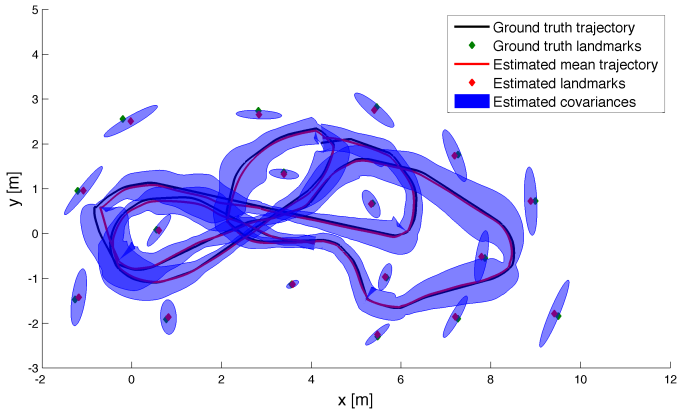


Fig. 4: The smooth and continuous trajectory and 3σ covariance envelope estimates produced by the *GP-Traj-Sparse* estimator for a short segment of the dataset.

of a mobile robot equipped with a laser rangefinder driving in an indoor, planar environment amongst a forest of 17 plastic-tube landmarks. The odometry and landmark measurements are provided at a rate of 1Hz, and additional trajectory queries are computed at a rate of 10Hz after estimator convergence. Groundtruth for the robot trajectory and landmark positions is provided by a Vicon motion capture system.

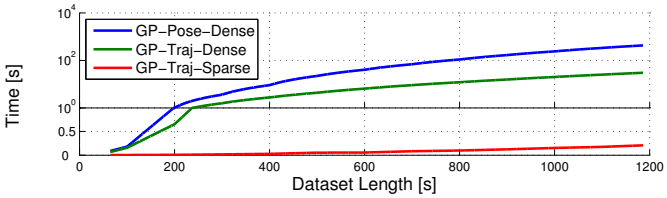
We implemented three estimators for comparison. The first was the algorithm described by Tong et al. [46], *GP-Pose-Dense*, the second was a naive version of our estimator, *GP-Traj-Dense*, that simply estimated a stacked state but did not exploit sparsity, and the third was a full implementation of our estimator, *GP-Traj-Sparse*, that exploited the sparsity structure as described in this paper.

Though the focus of this section is to demonstrate the significant reductions in computational cost, we provide Figure 4 to illustrate the smooth trajectory estimates we obtained from the continuous-time formulation. While the three algorithms differed in the number of degrees of freedom of their estimated states, their overall accuracies were similar for this dataset.

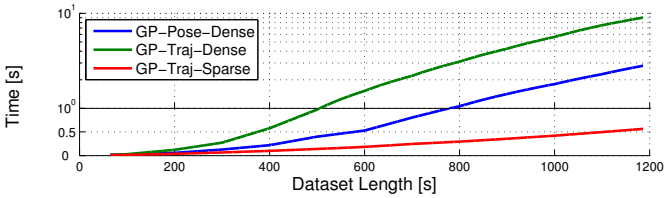
To evaluate the computational savings, we implemented all three algorithms in Matlab on a MacBook Pro with a 2.7GHz i7 processor and 16GB of 1600MHz DDR3 RAM, and timed the computation for segments of the dataset of varying lengths. These results are shown in Figure 5, where we provide the computation time for the individual operations that benefit most from the sparse structure, as well as the overall processing time.

We see that the *GP-Traj-Dense* algorithm is much slower than the original *GP-Pose-Dense* algorithm of Tong et al. [46]. This is because we have reintroduced the velocity part of the state, thereby doubling the number of variables associated with the trajectory. However, once we start exploiting the sparsity with *GP-Traj-Sparse*, the increase in number of variables pays off.

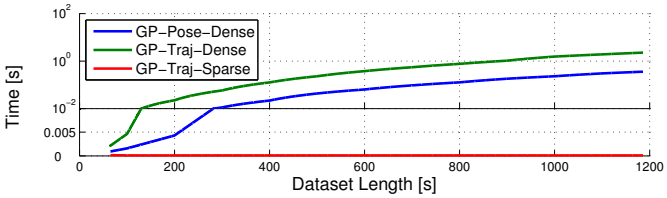
For *GP-Traj-Sparse*, we see in Figure 5(a) that the kernel matrix construction was linear in the number of estimated states. This can be attributed to the fact that we constructed the sparse $\mathcal{K}^{-1}$ directly. As predicted, the optimization time per



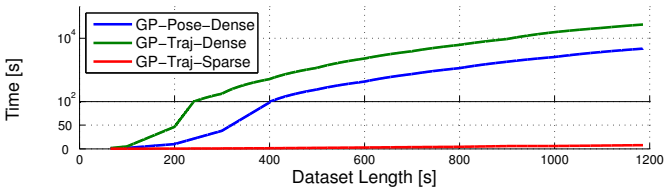
(a) Kernel matrix construction time.



(b) Optimization time per iteration.



(c) Interpolation time per additional query time.



(d) Total computation time.

Fig. 5: Plots comparing the compute time (as a function of trajectory length) for the *GP-Pose-Dense* algorithm described by Tong et al. [46] and two versions of our approach: *GP-Traj-Dense* (does not exploit sparsity) and *GP-Traj-Sparse* (exploits sparsity). The plots confirm the predicted computational complexities of the various methods; notably, *GP-Traj-Sparse* has linear cost in trajectory length. Please note the change from a linear to a log scale in the upper part of each plot.

iteration was also linear in Figure 5(b), and the interpolation time per additional query was constant regardless of state size in Figure 5(c). Finally, Figure 5(d) shows that the total compute time was also linear.

We also note that the number of iterations for optimization convergence varied for each algorithm. In particular, we found that the *GP-Traj-Sparse* implementation converged in fewer iterations than the other implementations due to the fact that we constructed the inverse kernel matrix directly, which resulted in greater numerical stability. The *GP-Traj-Sparse* approach clearly outperforms the other algorithms in terms of computational cost.

V. Discussion and Future Work

It is worth elaborating on a few issues. The main reason that the $\mathbf{W}_{xx}$ block is sparse in our approach, as compared to Tong et al. [46], is that we reintroduced velocity variables that had effectively been marginalized out. This idea of reintroducing variables to regain exact sparsity has been used before by Eustice et al. [12] in the delayed state filter and by Walter et al. [48] in the extended information filter. This is a good lesson to heed: the underlying structure of a problem may be exactly sparse, but by marginalizing out variables it appears dense. For us this means we need to use a Markovian trajectory state that is appropriate to our prior.

In much of mobile robotics, odometry measurements are treated more like inputs to the mean of the prior than pure measurements. We believe this is a confusing thing to do as it conflates two sources of uncertainty: the prior over trajectories and the odometry measurement noise. In our framework, we have deliberately separated these two functions and believe this is easier to work with and understand. We can see these two functions directly in Figure 3, where the prior is made up of binary factors joining consecutive trajectory states, and odometry measurements are unary factors attached to some of the trajectory states (we could have used binary odometry factors but chose to set things up this way due to the fact that we were explicitly estimating velocity).

While our analysis appears to be restricted to a small class of covariance functions, we have only framed our discussions in the context of robotics. Recent developments from machine learning [17] and signal processing [37] have shown that it is possible to generate other well-known covariance functions using a LTV SDE (some exactly and some approximately). This means they can be used with our framework. One case is the Matérn covariance family [33],

$$\boldsymbol{\mathcal{K}}_{\mathrm{m}}(t,t') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\ell} |t-t'| \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{\ell} |t-t'| \right) \mathbf{1} \tag{38}$$

where σ, ν, $\ell > 0$ are magnitude, smoothness, and length-scale parameters, $\Gamma(\cdot)$ is the gamma function, and $K_\nu(\cdot)$ is the modified Bessel function. For example, if we let

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{p}(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix}, \tag{39}$$

with $\nu = p + \frac{1}{2}$ with $p = 1$ and use the following SDE:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ -\lambda^2 \mathbf{1} & -2\lambda \mathbf{1} \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix} \mathbf{w}(t), \tag{40}$$

where $\lambda = \sqrt{2\nu}/\ell$ and $\mathbf{w}(t) \sim \mathcal{GP}\left(\mathbf{0}, \mathbf{Q}_C\, \delta(t-t')\right)$ (our usual white noise) with power spectral density matrix,

$$\mathbf{Q}_C = \frac{2\sigma^2 \pi^{\frac{1}{2}} \lambda^{2p+1} \Gamma(p+1)}{\Gamma(p+\frac{1}{2})} \mathbf{1}, \tag{41}$$

then we have that $\mathbf{p}(t)$ is distributed according to the Matérn covariance family: $\mathbf{p}(t) \sim \mathcal{GP}(\mathbf{0}, \boldsymbol{\mathcal{K}}_{\mathrm{m}}(t,t'))$ with $p = 1$. Another way to look at this is that passing white noise through LTV SDEs produces particular coloured-noise priors (i.e., not flat across all frequencies).

In terms of future work, we are currently concentrating on extending our results to GP priors generated by *nonlinear* stochastic differential equations. We believe this will be fruitful in terms of incorporating the dynamics (i.e., kinematics plus Newtonian mechanics) of a robot platform into the trajectory prior. Real sensors do not move arbitrarily through the world as they are usually attached to massive robots and this serves to constrain the motion. Another idea is to incorporate *latent force models* into our GP priors (e.g., see Alvarez et al. [1] or Hartikainen et al. [18]). We also plan to look further at the sparsity of STEAM and integrate our work with modern solvers to tackle large-scale problems; this should allow us to exploit more than just the primary sparsity of the problem and do so in an online manner.

VI. Conclusion

We have considered continuous-discrete estimation problems where a trajectory is viewed as a one-dimensional *Gaussian process* (GP), with time as the independent variable and measurements acquired at discrete times. Querying the trajectory can be viewed as nonlinear, GP regression. Our main contribution in this paper is to show that this querying can be accomplished very efficiently. To do this, we exploited the Markov property of our GP priors (generated by linear, time-varying stochastic differential equations driven by white noise) to construct an inverse kernel matrix that is sparse. This makes it fast to solve for the state at the measurement times (as is commonly done in vision and robotics) but also at any other time(s) of interest through GP interpolation. We also considered a slight generalization of the SLAM problem, *simultaneous trajectory estimation and mapping* (STEAM), which makes use of a continuous-time trajectory prior and allows us to query the state at any time of interest in an efficient manner. We hope this paper serves to deepen the connection between classical state estimation theory and recent machine learning methods by viewing batch estimation through the lens of Gaussian process regression.

Acknowledgments

Thanks to Dr. Alastair Harrison at Oxford who asked the all-important question: *how can the GP estimation approach [46] be related to factor graphs?* This work was supported by the Canada Research Chair Program, the Natural Sciences and Engineering Research Council of Canada, and the Academy of Finland.

References

[1] M Alvarez, D Luengo, and N Lawrence. Latent force models. In *Proceedings of the Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2009.

[2] T Bailey and H Durrant-Whyte. SLAM: Part II State of the art. *IEEE RAM*, 13(3):108–117, 2006.

[3] C Bibby and I D Reid. A hybrid SLAM representation for dynamic marine environments. In *Proc. ICRA*, 2010.

[4] M Bosse and R Zlot. Continuous 3D scan-matching with a spinning 2D laser. In *Proc. ICRA*, 2009.

[5] D C Brown. A solution to the general problem of multiple station analytical stereotriangulation. RCA-MTP data reduction tech. report no. 43, Patrick Airforce Base, 1958.

[6] A J Davison, I D Reid, N D Molton, and O Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE T. PAMI*, 29(6):1052–1067, 2007.

[7] M P Deisenroth, R Turner, M Huber, U D Hanebeck, and C E Rasmussen. Robust filtering and smoothing with Gaussian processes. *IEEE T. Automatic Control*, 57:1865–1871, 2012.

[8] F Dellaert and M Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *IJRR*, 25(12):1181–1204, 2006.

[9] H J Dong and T D Barfoot. Lighting-invariant visual odometry using lidar intensity imagery and pose interpolation. In *Proc. Field and Service Robotics*, 2012.

[10] H Durrant-Whyte and T Bailey. SLAM: Part I Essential algorithms. *IEEE RAM*, 11(3):99–110, 2006.

[11] H F Durrant-Whyte. Uncertain geometry in robotics. *IEEE Journal of Robotics and Automation*, 4(1):23–31, 1988.

[12] R M Eustice, H Singh, and J J Leonard. Exactly sparse delayed-state filters for view-based SLAM. *IEEE TRO*, 22(6):1100–1114, 2006.

[13] B Ferris, D Hähnel, and D Fox. Gaussian processes for signal strength-based localization. In *Proc. RSS*, 2006.

[14] B Ferris, D Fox, and N Lawrence. Wifi-SLAM using Gaussian process latent variable models. In *Proc. IJCAI*, 2007.

[15] P T Furgale, T D Barfoot, and G Sibley. Continuous-time batch estimation using temporal basis functions. In *Proc. ICRA*, 2012.

[16] O Grau and J Pansiot. Motion and velocity estimation of rolling shutter cameras. In *Proceedings of the 9th European Conference on Visual Media Production*, pages 94–98, 2012.

[17] J Hartikainen and S Särkkä. Kalman filtering and smoothing solutions to temporal Gaussian process regression models. In *Proc. of the IEEE Int. Work. on Machine Learning for Signal Processing*, 2010.

[18] J Hartikainen, M Seppänen, and S Särkkä. State-space inference for non-linear latent force models with application to satellite orbit prediction. In *Proc. ICML*, 2012.

[19] J Hedborg, P Forssén, M Felsberg, and E Ringaby. Rolling shutter bundle adjustment. In *Proc. CVPR*, 2012.

[20] A H Jazwinski. *Stochastic Processes and Filtering Theory*. Academic, New York, 1970.

[21] M Kaess, A Ranganathan, and R Dellaert. iSAM: Incremental smoothing and mapping. *IEEE TRO*, 24(6):1365–1378, 2008.

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

[23] R E Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[24] R E Kalman and R S Bucy. New results in linear filtering and prediction theory. *Transactions of the ASME–Journal of Basic Engineering*, 83(3):95–108, 1961.

[25] J Ko and D Fox. GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90, July 2009.

[26] J Ko and D Fox. Learning GP-BayesFilters via Gaussian process latent variable models. *Auton. Robots*, 30(1):3–23, 2011.

[27] N Lawrence. Gaussian process latent variable models for visualization of high dimensional data. In *Proc. NIPS*, 2003.

[28] F Lindgren, H Rue, and J Lindström. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *J. of the Royal Stat. Society: Series B*, 73(4):423–498, 2011.

[29] S Lovegrove, A Patron-Perez, and G Sibley. Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In *Proc. BMVC*, 2013.

[30] F Lu and E Milios. Globally consistent range scan alignment for environment mapping. *Auton. Robots*, 4(4):333–349, 1997.

[31] P S Maybeck. *Stochastic Models, Estimation, and Control*, volume 141 of *Mathematics in Science and Engineering*. Academic Press Inc., 1979.

[32] P Newman, G Sibley, M Smith, M Cummins, A Harrison, C Mei, I Posner, R Shade, D Schroeter, L Murphy, W Churchill, D Cole, and I Reid. Navigating, recognising and describing urban spaces with vision and laser. *IJRR*, 28(11-12):1406–1433, 2009.

[33] C E Rasmussen and C K I Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[34] S Särkkä. *Recursive Bayesian Inference on Stochastic Differential Equations*. PhD thesis, Helsinki Uni. of Technology, 2006.

[35] S Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[36] S Särkkä and J Sarmavuori. Gaussian filtering and smoothing for continuous-discrete dynamic systems. *Signal Processing*, 93 (2):500–510, 2013.

[37] S Särkkä, A Solin, and J Hartikainen. Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through Kalman filtering. *IEEE Signal Processing Magazine*, 30(4):51–61, 2013.

[38] G Sibley, L Matthies, and G Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 27(5):587–608, 2010.

[39] R C Smith and P Cheeseman. On the representation and estimation of spatial uncertainty. *IJRR*, 5(4):56–68, 1986.

[40] R C Smith, M Self, and P Cheeseman. Estimating uncertain spatial relationships in robotics. In Ingemar J. Cox and Gordon T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, New York, 1990.

[41] A Solin and S Särkkä. Explicit link between periodic covariance functions and state space models. In *Proceedings of the Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2014.

[42] R F Stengel. *Optimal Control and Estimation*. Dover Publications Inc., 1994.

[43] H Strasdat, J M M Montiel, and A J Davison. Real-time monocular SLAM: Why filter? In *Proc. ICRA*, 2010.

[44] S Thrun and M Montemerlo. The graph SLAM algorithm with applications to large-scale mapping of urban structures. *IJRR*, 25(5-6):403–429, 2006.

[45] C H Tong, P Furgale, and T D Barfoot. Gaussian process Gauss-Newton: Non-parametric state estimation. In *Proc. of the 9th Conf. on Computer and Robot Vision*, pages 206–213, 2012.

[46] C H Tong, P T Furgale, and T D Barfoot. Gaussian process Gauss-Newton for non-parametric simultaneous localization and mapping. *IJRR*, 32(5):507–525, 2013.

[47] B Triggs, P McLauchlan, R Hartley, and A Fitzgibbon. Bundle adjustment — A modern synthesis. In B Triggs, A Zisserman, and R Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer Berlin Heidelberg, 2000.

[48] M R Walter, R M Eustice, and J J Leonard. Exactly sparse extended information filters for feature-based SLAM. *IJRR*, 26 (4):335–359, 2007.