Sparse Gaussian Processes on Matrix Lie Groups: A Unified Framework for Optimizing Continuous-Time Trajectories

Jing Dong, Mustafa Mukadam, Byron Boots, and Frank Dellaert

***Abstract*— Continuous-time trajectories are useful for reasoning about robot motion in a wide range of tasks. Sparse Gaussian processes (GPs) can be used as a non-parametric representation for trajectory distributions that enables fast trajectory optimization by sparse GP regression. However, most previous approaches that utilize sparse GPs for trajectory optimization are limited by the fact that the robot state is represented in vector space. In this paper, we first extend previous works to consider the state on general matrix Lie groups by applying a *constant-velocity* prior and defining *locally* linear GPs. Next, we discuss how sparse GPs on Lie groups provide a unified continuous-time framework for trajectory optimization for solving a number of robotics problems including state estimation and motion planning. Finally, we demonstrate and evaluate our approach on several different estimation and motion planning tasks with both synthetic and real-world experiments.**

I. Introduction

Consider the problem of simultaneous localization and mapping (SLAM) [1], [2]. Solving this problem is a fundamental capability for autonomous mobile robots that must explore and plan in previously unseen environments. While SLAM is a well-studied area of robotics, the majority of existing SLAM algorithms rely on discrete-time representations of robot trajectories. Although discrete-time representations are sufficient for many tasks, they are often difficult to use in several important scenarios, including: (1) when sensors measure the environment continuously, for example spinning LIDAR or rolling-shutter cameras produce measurements that may be distorted by the robot's self-motion; and (2) when sensor measurements arrive asynchronously.

While heuristics are often employed to overcome these challenges, a theoretically sound framework for handling continuous motion and/or measurements can result in better accuracy. Unlike trajectory representations that are densely parameterized with discrete states at frequent, regular time intervals, continuous-time representations are often sparsely parameterized, but contain mechanisms that allow the trajectory to be queried to recover the robot state at *any* time of interest. Several popular continuous-time trajectory representations include linear interpolation [3], [4], [5], splines [6], [7], [9], [10], [11], and hierarchical wavelets [12].

This paper focuses on an alternative probabilistic, non-parametric representation for trajectories based on Gaussian processes (GPs). Tong *et al.* [14], [15] showed that simultaneous trajectory estimation and mapping (STEAM), the continuous-time extension of SLAM, can be reduced to GP regression. By placing various GP priors on robot trajectories, this approach can solve different types of trajectory estimation problems. However, if standard kernels, such as the squared exponential kernel, are used, the method is prohibitively expensive with time and space complexity that scales polynomially with the number of parameters.

Fig. 1: Some experiments we evaluated the proposed GPs on Lie groups, from up to down left to right: 2D planar localization using CMU range-only dataset [13]; 3D monocular visual-inertial SLAM; Full-body planning of PR2 robot in simulation; Full-body planning of Vector robot in real-world.

Maintaining sparsity in SLAM problems has been well-studied [2], [17], and it is the key to scalable optimization in many modern SLAM algorithms. In the context of continuous-time SLAM, Barfoot *et al.* [18] shows that by applying a linear time-varying stochastic differential equation (LTV-SDE) prior on trajectories, the inverse kernel matrices used during optimization are exactly sparse, leading to efficient GP regression that can solve SLAM problems. This approach is further extended to nonlinear SDEs [19] and incremental settings [20].

Other areas of robotics have recently benefited from these techniques as well. For example, recent work in motion planning [21], [22], [23] takes advantage of sparse and incremental GP regression to speed up trajectory optimization, leading to very efficient motion planning algorithms.

A major drawback of all of these GP-based approaches is that they only reason about trajectories where system states evolve in a *vector space*, which may not be a valid

J. Dong, M. Mukadam, B. Boots, and F. Dellaert are with Institute for Robotics & Intelligent Machines, Georgia Institute of Technology, Atlanta, GA, USA. {jdong,mmukadam3}@gatech.edu, {bboots,frank}@cc.gatech.edu. This work was supported by National Institute of Food and Agriculture, USDA award 2014-67021-22556 and NSF NRI award 1637908

assumption for many systems. For example, typical vector-valued representations for rotation of a 3D rigid-body either exhibit singularities (Euler angles) or impose extra nonlinear constraints (quaternions).

Trajectories in many robotics applications can be defined on more general *Lie groups*, rather than simple vector spaces. For example, rotation of a 3D rigid body is the special orthogonal group SO(3), transformation of a 3D rigid body is the special Euclidean group SE(3), and state estimated from a monocular camera with scale drift information is the similarity group Sim(3) [24], [25].

Sparse GP regression for STEAM [18] has been extended to SE(3) by Anderson *et al*. [26]. Anderson *et al*. select *constant body-frame velocity* prior as GP prior, which is physically motivated by inertia, meanwhile generated by local LTV-SDEs. In this paper, we further extend sparse GP regression to general Lie groups, enabling more applications of sparse GPs: rotation estimation on SO(3), monocular visual SLAM on Sim(3), and applications other than state estimation, like motion planning on Lie groups.

Along with our technical contributions, we, for the first time, provide a novel insight that Gaussian processes on Lie groups can be viewed as a unified tool for reasoning about continuous-time trajectories in various robotics applications, including state estimation and motion planning, since they share the same GP continuous-time trajectory representation, the same *factor graph* problem structure, and the same maximum a posteriori probabilistic inference framework.

Our specific contributions include:

- Extending sparse GPs [18], [26] to general matrix Lie groups.
- A new perspective that views sparse GP regression on Lie groups as a unified trajectory optimization tool that can be used to reason about, and sometimes simultaneously solve, a variety of robotics tasks.
- Extensive experimental evaluations on different types of real-world robotics problems, showcasing the effectiveness of reasoning about trajectories with sparse GPs on Lie groups.

II. Preliminaries

We begin by formulating continuous-time trajectory estimation problems as GP regression, review how a sparse GP prior can be defined on vector spaces, and finally give a very brief review of Lie group fundamentals. For a full treatment the readers are encouraged to refer [18] for GP regression as trajectory estimation and [27] for Lie group theory.

A. Problem Definition

We consider the problem of continuous-time trajectory estimation, in which a continuous-time system state $\boldsymbol{x}(t)$ is estimated from observations [14]. The system model is described as

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

$$\mathbf{z}_i = \mathbf{h}_i(\boldsymbol{x}(t_i)) + \mathbf{n}_i, \mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_i), \tag{2}$$

where $\boldsymbol{x}(t)$ is represented by a GP with mean function $\boldsymbol{\mu}(t)$ and covariance function $\boldsymbol{\mathcal{K}}(t,t')$. Measurements $\mathbf{z}_i$ at time t_i are obtained by the (generally nonlinear) discrete-time measurement function $\mathbf{h}_i$ in Eq. (2) and assumed to be corrupted by zero-mean Gaussian noise with covariance $\boldsymbol{\Sigma}_i$.

B. Maximum a Posteriori Estimation

GP regression is performed by *maximum a posteriori* (MAP) inference, where the most likely trajectory is the mode (and also the mean) of the posterior distribution i.e. the GP in Eq. (1) conditioned on the measurements. The MAP estimate of the trajectory can be computed through Gaussian process Gauss-Newton (GPGN) [14]. To define the objective function, we assume that there are M observations and

$$\boldsymbol{x} \doteq \begin{bmatrix} \boldsymbol{x}(t_1) \\ \vdots \\ \boldsymbol{x}(t_M) \end{bmatrix}, \boldsymbol{\mu} \doteq \begin{bmatrix} \boldsymbol{\mu}(t_1) \\ \vdots \\ \boldsymbol{\mu}(t_M) \end{bmatrix}, \boldsymbol{\mathcal{K}} \doteq [\boldsymbol{\mathcal{K}}(t_i,t_j)]\Big|_{ij,1\le i,j\le M},$$

$$\mathbf{z} \doteq \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_M \end{bmatrix}, \mathbf{h}(\boldsymbol{x}) \doteq \begin{bmatrix} \mathbf{h}_1(\boldsymbol{x}_1) \\ \vdots \\ \mathbf{h}_M(\boldsymbol{x}_M) \end{bmatrix}, \boldsymbol{\Sigma} \doteq \begin{bmatrix} \boldsymbol{\Sigma}_1 & & \\ & \ddots & \\ & & \boldsymbol{\Sigma}_M \end{bmatrix}.$$

The MAP estimation objective can then be written as

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}} \left\{ \frac{1}{2} \parallel \boldsymbol{x} - \boldsymbol{\mu} \parallel^2_{\boldsymbol{\mathcal{K}}} + \frac{1}{2} \parallel \mathbf{h}(\boldsymbol{x}) - \mathbf{z} \parallel^2_{\boldsymbol{\Sigma}} \right\}, \tag{3}$$

where $\|\|_{\boldsymbol{\Sigma}}$ is Mahalanobis distance defined as $\parallel \mathbf{x} \parallel^2_{\boldsymbol{\Sigma}} \doteq \mathbf{x}^\top \boldsymbol{\Sigma}^{-1} \mathbf{x}$. MAP estimation is thus formulated as a nonlinear least squares optimization problem.

We use a Gauss-Newton approach to solve the nonlinear least squares problem. By linearizing the measurement function $\mathbf{h}_i$ around a linearization point $\overline{\boldsymbol{x}}_i$, we obtain

$$\mathbf{h}_i(\overline{\boldsymbol{x}}_i + \delta\boldsymbol{x}_i) \approx \mathbf{h}_i(\overline{\boldsymbol{x}}_i) + \mathbf{H}_i \delta\boldsymbol{x}_i, \mathbf{H}_i \doteq \frac{\partial \mathbf{h}_i}{\partial \boldsymbol{x}}\Big|_{\overline{\boldsymbol{x}}_i}, \tag{4}$$

in which $\mathbf{H}_i$ is the Jacobian matrix of the measurement function (2) at linearization point $\overline{\boldsymbol{x}}_i$. By defining $\mathbf{H} \doteq \operatorname{diag}(\mathbf{H}_1, \ldots, \mathbf{H}_M)$, we can generate a linearized least squares problem around the linearization points $\overline{\boldsymbol{x}}$

$$\delta\boldsymbol{x}^* = \underset{\delta\boldsymbol{x}}{\operatorname{argmax}} \left\{ \frac{1}{2} \parallel \overline{\boldsymbol{x}} + \delta\boldsymbol{x} - \boldsymbol{\mu} \parallel^2_{\boldsymbol{\mathcal{K}}} + \frac{1}{2} \parallel \mathbf{h}(\overline{\boldsymbol{x}}) + \mathbf{H}\delta\boldsymbol{x} - \mathbf{z} \parallel^2_{\boldsymbol{\Sigma}} \right\}. \tag{5}$$

The GPGN algorithm starts from some initial guess of $\overline{\boldsymbol{x}}$, and then at each iteration, the optimal perturbation $\delta\boldsymbol{x}^*$ is found by solving the linear system

$$(\boldsymbol{\mathcal{K}}^{-1} + \mathbf{H}^\top \boldsymbol{\Sigma}^{-1} \mathbf{H}) \delta\boldsymbol{x}^* = \boldsymbol{\mathcal{K}}^{-1}(\boldsymbol{\mu} - \overline{\boldsymbol{x}}) + \mathbf{H}^\top \boldsymbol{\Sigma}^{-1}(\mathbf{z} - \mathbf{h}) \tag{6}$$

and updating the solution $\overline{\boldsymbol{x}} \leftarrow \overline{\boldsymbol{x}} + \delta\boldsymbol{x}^*$ until convergence.

The information matrix $\boldsymbol{\mathcal{K}}^{-1}$ in Eq. (6) encodes the GP prior information, and $\mathbf{H}^\top \boldsymbol{\Sigma}^{-1} \mathbf{H}$ represents information from measurements. $\mathbf{H}^\top \boldsymbol{\Sigma}^{-1} \mathbf{H}$ is block-wise sparse in most SLAM problems [2], but $\boldsymbol{\mathcal{K}}^{-1}$ is not usually sparse for most commonly used kernels. Next, we define GP priors with sparse structure that can be exploited to efficiently solve the nonlinear least squares problem above.

C. Sparse GP Priors on Vector Space

We now describe a class of exactly sparse GP priors on vector space for trajectory estimation proposed in Barfoot *et al*. [18]. Here GP priors are considered on vector-valued system states $\boldsymbol{x}(t) \in \mathbb{R}^N$ generated by linear time-varying stochastic differential equations (LTV-SDEs)

$$\dot{\boldsymbol{x}}(t) = \mathbf{A}(t)\boldsymbol{x}(t) + \mathbf{u}(t) + \mathbf{F}(t)\mathbf{w}(t), \tag{7}$$

where $\mathbf{u}(t)$ is the known system control input, $\mathbf{w}(t)$ is white process noise, and both $\mathbf{A}(t)$ and $\mathbf{F}(t)$ are time-varying system matrices. The white process noise is represented by

$$\mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_C\delta(t-t')), \tag{8}$$

where $\mathbf{Q}_C$ is the power-spectral density matrix, which is a hyperparameter [19], and $\delta(t-t')$ is the Dirac delta function. The mean and covariance of the LTV-SDE generated GP are

$$\boldsymbol{\mu}(t) = \boldsymbol{\Phi}(t,t_0)\boldsymbol{\mu}_0 + \int_{t_0}^{t} \boldsymbol{\Phi}(t,s)\mathbf{u}(s)ds \tag{9}$$

$$\boldsymbol{\mathcal{K}}(t,t') = \boldsymbol{\Phi}(t,t_0)\boldsymbol{\mathcal{K}}_0\boldsymbol{\Phi}(t',t_0)^\top + \int_{t_0}^{\min(t,t')} \boldsymbol{\Phi}(t,s)\mathbf{F}(s)\mathbf{Q}_C\mathbf{F}(s)^\top\boldsymbol{\Phi}(t',s)^\top ds \tag{10}$$

where $\boldsymbol{\mu}_0$ is the initial mean value of first state, $\boldsymbol{\mathcal{K}}_0$ is the covariance of first state, and $\boldsymbol{\Phi}(t,s)$ is transition matrix. If the system is generated by the LTV-SDE in Eq. (7), the inverse covariance matrix $\boldsymbol{\mathcal{K}}^{-1}$ is block-tridiagonal [18].

A commonly used GP prior is the *constant-velocity* prior, which is derived from our understanding of the physical properties of robotic systems. In most robots, large acceleration implies extreme force, which may be harmful if directly applied to the system. Therefore, acceleration should be kept to a minimum, to the extent possible.

The constant-velocity GP prior is generated by a LTV-SDE with white noise on the acceleration and has been used in trajectory estimation [14], [18]

$$\ddot{\mathbf{p}}(t) = \mathbf{w}(t), \tag{11}$$

where $\mathbf{p}(t)$ is the N-dimensional vector-valued position (or pose) variable of trajectory. To convert this prior into the LTV-SDE form of Eq. (7), a Markov system state variable is declared

$$\boldsymbol{x}(t) \doteq \begin{bmatrix} \mathbf{p}(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix}. \tag{12}$$

The prior in Eq. (11) then can easily be converted into a LTV-SDE in Eq. (7) by defining

$$\mathbf{A}(t) = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{u}(t) = \mathbf{0}, \quad \mathbf{F}(t) = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}. \tag{13}$$

D. Lie Group Basics

A group is an algebraic structure $\{G, \circ\}$ consisting of a set of elements G and an operator $\circ$. We limit our discussion on matrix Lie groups, whose elements are square, invertible matrices, and $\circ$ is matrix multiplication. Every N-dimensional matrix Lie group G has an associated *Lie algebra* $\mathfrak{g}$ [27, p.16]. The Lie algebra $\mathfrak{g}$ coincides with the local tangent space to the manifold of G. The *exponential map* $\exp : \mathfrak{g} \to G$ and *logarithm map* $\log : G \to \mathfrak{g}$ define the mapping between the Lie group and Lie algebra respectively [27, p.18]. G also has an associated *hat operator* $\wedge : \mathbb{R}^N \to \mathfrak{g}$ and *vee operator* $\vee : \mathfrak{g} \to \mathbb{R}^N$ that converts elements in local coordinates $\mathbb{R}^N$ to the Lie algebra $\mathfrak{g}$ and vice versa [27, p.20]. The vector space $\mathbb{R}^N$ is just a trivial Lie group, where $\exp$, $\log$, $\wedge$ and $\vee$ are all identity.

III. Sparse GP Priors on Lie Groups

This section summarizes how GP priors are defined on Lie groups. For more details, readers are encouraged to read the author's technical report [28].

A. Constant Body-Frame Velocity Prior

We use $T \in G$ to represent an object in Lie group G, so the continuous-time trajectory is written as $T(t)$, and trajectory measurements are observed at times $t_1, \ldots, t_M$, the associated states are $T_1, \ldots, T_M$. To estimate $T(t)$, we first define the Markov system state by appending the state with velocity

$$\boldsymbol{x}(t) \doteq \{T(t), \boldsymbol{\varpi}(t)\}, \tag{14}$$

where $\boldsymbol{\varpi}(t)$ is the 'body-frame velocity' variable[1] defined

$$\boldsymbol{\varpi}(t) \doteq (T(t)^{-1}\dot{T}(t))^\vee. \tag{15}$$

The $\vee$ operator can be always applied to $T(t)^{-1}\dot{T}(t)$ since $\forall T \in G,\ T^{-1}\dot{T} \in \mathfrak{g}$ [27, p.20].

Similar to Eq. (11), we can define the constant 'body-frame velocity' prior on Lie groups as

$$\dot{\boldsymbol{\varpi}}(t) = \mathbf{w}(t), \quad \mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_C\delta(t-t')), \tag{16}$$

but this is a nonlinear SDE, which does not match the LTV-SDE defined by Eq. (7).

B. Locally Linear Constant-Velocity GP Priors

To define a LTV-SDE which can leverage the constant-velocity GP prior, we linearize the Lie group manifold around each T_i, and define both a *local* GP and LTV-SDE on the linear tangent space. We first define a local GP for any time t on trajectory which meets $t_i \le t \le t_{i+1}$,

$$T(t) = T_i \exp(\boldsymbol{\xi}_i(t)^\wedge), \quad \boldsymbol{\xi}_i(t) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\mathcal{K}}(t_i,t)). \tag{17}$$

the *local* pose variable $\boldsymbol{\xi}_i(t) \in \mathbb{R}^N$ around T_i is defined by

$$\boldsymbol{\xi}_i(t) \doteq \log(T_i^{-1}T(t))^\vee. \tag{18}$$

The local LTV-SDE that represents constant-velocity information is

$$\ddot{\boldsymbol{\xi}}_i(t) = \mathbf{w}(t), \quad \mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_C\delta(t-t')). \tag{19}$$

If we define the local Markov system state

$$\boldsymbol{\gamma}_i(t) \doteq \begin{bmatrix} \boldsymbol{\xi}_i(i) \\ \dot{\boldsymbol{\xi}}_i(t) \end{bmatrix}, \tag{20}$$

[1] In SO(3) and SE(3), $\boldsymbol{\varpi}(t)$ is the body-frame velocity (see [29, p.52] and [29, p.55]), so we just define and call $\boldsymbol{\varpi}(t)$ the 'body-frame velocity' for general Lie groups.

the local LTV-SDE is rewritten as

$$\dot{\boldsymbol{\gamma}}_i(t) = \frac{d}{dt}\begin{bmatrix}\boldsymbol{\xi}_i(t)\\ \dot{\boldsymbol{\xi}}_i(t)\end{bmatrix} = \begin{bmatrix}\dot{\boldsymbol{\xi}}_i(t)\\ \mathbf{w}(t)\end{bmatrix}. \tag{21}$$

To prove the equivalence between the nonlinear SDE in Eq. (16) and the local LTV-SDE in Eq. (19), we first look at equation [27, p.26]

$$T(t)^{-1}\dot{T}(t) = \big(\boldsymbol{\mathcal{J}}_r(\boldsymbol{\xi}_i(t))\dot{\boldsymbol{\xi}}_i(t)\big)^{\wedge}, \tag{22}$$

where $\boldsymbol{\mathcal{J}}_r$ is the *right Jacobian* of G. With Eq. (15) we have

$$\dot{\boldsymbol{\xi}}_i(t) = \boldsymbol{\mathcal{J}}_r(\boldsymbol{\xi}_i(t))^{-1}\boldsymbol{\varpi}(t). \tag{23}$$

If the small time interval assumption between any t_i and t_{i+1} is satisfied, we have small enough $\boldsymbol{\xi}_i(t)$ with a good approximation of right Jacobian $\boldsymbol{\mathcal{J}}_r(\boldsymbol{\xi}_i(t)) \approx \mathbf{I}$ and

$$\dot{\boldsymbol{\xi}}_i(t) \approx \boldsymbol{\varpi}(t). \tag{24}$$

So we have proved that the LTV-SDE in Eq. (19) and (21) is a good approximation of constant 'body-frame velocity' prior defined by Eq. (16).

Both the local GP and LTV-SDE are defined on the tangent space, so they should only be applied around the current linearization point T_i. But if all t_i and t_{i+1} pairs have a small enough interval, the GP and LTV-SDE can be defined in a piecewise manner, and every point on the trajectory can be converted to a local variable $\boldsymbol{\xi}_i(t)$ based on its nearby estimated state T_i.

Note that, although we assume that time intervals between all t_i and t_{i+1} are small to prove LTV-SDE in Eq. (19) is a good approximation of constant 'body-frame velocity' prior by Eq. (16), this assumption does not need to be satisfied to make the above LTV-SDE a *valid* GP prior. So we can still use the proposed sparse GP when trajectory time intervals are large, although the actual GP prior applied is less accurate compared to the true constant 'body-frame velocity' prior.

Note that a special case of the above theory to SE(3), was proposed in previous work [26], from which our extension to general matrix Lie groups is inspired.

C. A Factor Graph Perspective

Once the local GP and constant-velocity LTV-SDE are defined, we can write down the cost function J_{gp} used to incorporate information about the GP prior into the nonlinear least squares optimization in Eq. (3). As discussed, the GP prior cost function has the generic form

$$J_{gp} = \frac{1}{2}\parallel \boldsymbol{\mu} - \boldsymbol{x} \parallel^2_{\boldsymbol{\mathcal{K}}}, \tag{25}$$

but if the trajectory is generated by a constant-velocity LTV-SDE in Eq. (21), the GP prior cost can be specified as [18]

$$J_{gp} = \sum_i \frac{1}{2}\mathbf{e}_i^{\top}\mathbf{Q}_i^{-1}\mathbf{e}_i, \tag{26}$$

$$\mathbf{e}_i = \boldsymbol{\Phi}(t_{i+1}, t_i)\boldsymbol{\gamma}_i(t_i) - \boldsymbol{\gamma}_i(t_{i+1}), \tag{27}$$

where $\mathbf{Q}_i$ is the covariance matrix [18]

$$\boldsymbol{\Phi}(t,s) = \begin{bmatrix}\mathbf{1} & (t-s)\mathbf{1}\\ \mathbf{0} & \mathbf{1}\end{bmatrix}, \mathbf{Q}_i = \begin{bmatrix}\frac{1}{3}\Delta t_i^3\mathbf{Q}_C & \frac{1}{2}\Delta t_i^2\mathbf{Q}_C\\ \frac{1}{2}\Delta t_i^2\mathbf{Q}_C & \Delta t_i\mathbf{Q}_C\end{bmatrix}, \tag{28}$$

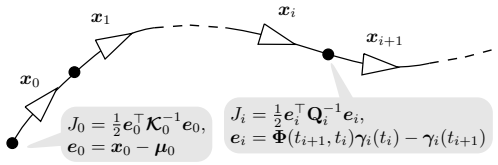


Fig. 2: An example factor graph, showing states (triangles) and factors (black boxes). GP prior factors connect consecutive states, and define the prior information on first state.

where $\Delta t_i = t_{i+1} - t_i$.

Since the GP prior cost J_{gp} has been written as a sum of squared cost terms, and each cost term is only related to nearby (local) Markov states, we can represent the least squares problem by *factor graph* models. In factor graphs the system states are represented by variable factors, and the cost terms are represented by cost factors. An example factor graph is shown in Fig. 2. By converting nonlinear least squares problems into factor graphs, we can take advantage of factor graph inference tools to solve the problems efficiently. Additional information about the relationship between factor graphs and sparse GP and SLAM problems can be found in [2], [18], [19], [20]

D. Querying the Trajectory

One of the advantages of representing the continuous-time trajectory as a GP is that we have the ability to query the state of the robot at any time along the trajectory. For constant-velocity GP priors, the system state $\boldsymbol{x}(\tau), t_i \le \tau \le t_{i+1}$ can be estimated by two nearby states $\boldsymbol{x}(t_i)$ and $\boldsymbol{x}(t_{i+1})$ [18], which allows efficient $O(1)$ interpolation. We first calculate the mean value of local state $\hat{\boldsymbol{\gamma}}_i(\tau)$

$$\hat{\boldsymbol{\gamma}}_i(\tau) = \boldsymbol{\Lambda}(\tau)\hat{\boldsymbol{\gamma}}_i(t_i) + \boldsymbol{\Psi}(\tau)\hat{\boldsymbol{\gamma}}_i(t_{i+1}), \tag{29}$$

where

$$\boldsymbol{\Lambda}(\tau) = \boldsymbol{\Phi}(\tau, t_i) - \mathbf{Q}_{\tau}\boldsymbol{\Phi}(\tau, t_i)^{\top}\mathbf{Q}_{i+1}^{-1}\boldsymbol{\Phi}(t_{i+1}, t_i), \tag{30}$$

$$\boldsymbol{\Psi}(\tau) = \mathbf{Q}_{\tau}\boldsymbol{\Phi}(\tau, t_i)^{\top}\mathbf{Q}_{i+1}^{-1}. \tag{31}$$

Once we have the mean value of local state $\hat{\boldsymbol{\gamma}}_i(\tau)$, the mean value of the full state $\hat{\boldsymbol{x}}(\tau) = \{\hat{T}(\tau), \hat{\boldsymbol{\varpi}}(\tau)\}$ is

$$\hat{T}(\tau) = \hat{T}_i \exp\Big(\big(\boldsymbol{\Lambda}_1(\tau)\hat{\boldsymbol{\gamma}}_i(t_i) + \boldsymbol{\Psi}_1(\tau)\hat{\boldsymbol{\gamma}}_i(t_{i+1})\big)^{\wedge}\Big), \tag{32}$$

$$\hat{\boldsymbol{\varpi}}(\tau) = \boldsymbol{\mathcal{J}}_r\big(\hat{\boldsymbol{\xi}}_i(\tau)\big)^{-1}\big(\boldsymbol{\Lambda}_2(\tau)\hat{\boldsymbol{\gamma}}_i(t_i) + \boldsymbol{\Psi}_2(\tau)\hat{\boldsymbol{\gamma}}_i(t_{i+1})\big), \tag{33}$$

where

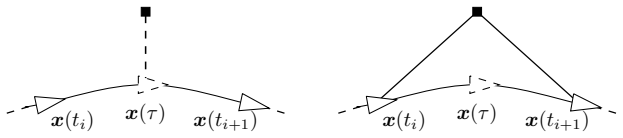
$$\boldsymbol{\Lambda}(\tau) = \begin{bmatrix}\boldsymbol{\Lambda}_1(\tau)\\ \boldsymbol{\Lambda}_2(\tau)\end{bmatrix},\ \boldsymbol{\Psi}(\tau) = \begin{bmatrix}\boldsymbol{\Psi}_1(\tau)\\ \boldsymbol{\Psi}_2(\tau)\end{bmatrix},$$

$$\hat{\boldsymbol{\gamma}}_i(t_i) = \begin{bmatrix}\mathbf{0}\\ \hat{\boldsymbol{\varpi}}(t_i)\end{bmatrix},\ \hat{\boldsymbol{\gamma}}_i(t_{i+1}) = \begin{bmatrix}\hat{\boldsymbol{\xi}}_i(t_{i+1})\\ \boldsymbol{\mathcal{J}}_r(\hat{\boldsymbol{\xi}}_i(t_{i+1}))^{-1}\hat{\boldsymbol{\varpi}}(t_{i+1})\end{bmatrix},$$

$$\hat{\boldsymbol{\xi}}_i(\tau) = \log(\hat{T}_i^{-1}\hat{T}(\tau))^{\vee},\quad \hat{\boldsymbol{\xi}}_i(t_{i+1}) = \log(\hat{T}_i^{-1}\hat{T}_{i+1})^{\vee},$$

E. Fusion of Asynchronous Measurements

Continuous-time trajectory interpolation affords GP-based trajectory estimation methods several advantages over discrete-time localization algorithms. In addition to providing a method for querying the trajectory at any time of interest, GP interpolation can be used to reduce the number of states needed to represent the robot's trajectory, and elegantly handle asynchronous measurements.



(a) Meausurement (b) Interpolated Factor

Fig. 3: (a) Measurement at time τ, dashed line indicates it's not an actual factor. (b) The interpolated factor encodes measurement at time τ.

Assume there is a measurement $\mathbf{z}_\tau$ of state $\boldsymbol{x}(\tau)$ available at an arbitrary time $\tau, t_i \leq \tau \leq t_{i+1}$, with measurement function $\mathbf{h}_\tau$ and corresponding covariance $\boldsymbol{\Sigma}_\tau$. The measurement cost can be incorporated as a 'virtual unary factor' in the graph and is given by

$$J_\tau(\boldsymbol{x}(\tau)) = \frac{1}{2} \parallel \mathbf{z}_\tau - \mathbf{h}_\tau(\boldsymbol{x}(\tau)) \parallel^2_{\boldsymbol{\Sigma}_\tau} . \tag{34}$$

Since system state $\boldsymbol{x}(\tau)$ is not explicitly available for optimization, we perform trajectory interpolation between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$ by Eq. (32) – (33), and rewrite the cost in terms of the interpolated mean value $\hat{\boldsymbol{x}}(\tau)$

$$J_\tau(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) = \frac{1}{2} \parallel \mathbf{z}_\tau - \mathbf{h}_\tau(\hat{\boldsymbol{x}}(\tau)) \parallel^2_{\boldsymbol{\Sigma}_\tau} . \tag{35}$$

Because the measurement cost is represented by $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$, a binary 'interpolated' factor can be added to the factor graph, and $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$ are optimized without explicitly adding an additional state. The factor graph illustration is shown in Fig. 3.

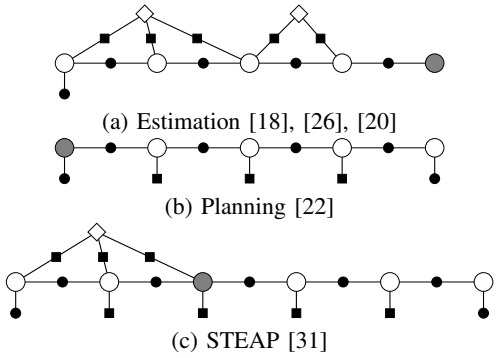
IV. GPs as a Unified Trajectory Representation

Estimation: As noted in the introduction, the SLAM community has been using continuous-time trajectory representations for many years to naturally handle asynchronous measurements and continuous-time sensors like rolling shutter camera or laser scanner. Compared to standard continuous-time trajectory representations like splines, using sparse GPs in SLAM [18], [26], [20] has several special advantages:

- A GP represents a trajectory distribution, so the mean trajectory of interest naturally has a notion of associated uncertainty.
- The sparse GP prior does not break the inherent sparsity underlying SLAM problems, and aligns well with their factor graph structure (shown in Fig. 4(a)).
- The sparse factor graph structure allows for the use of efficient incremental inference tools, like iSAM2 [30], to enable online incremental GP regression [20].
- The constant velocity prior [26] has physical meaning.

Planning: Recent work in motion planning has used sparse GPs for fast trajectory interpolation [21] and then later, GP regression to solve trajectory optimization problems [22]. For motion planning, sparse GPs provide the following benefits:

- Structure exploiting inference can be performed on factor graphs (shown in Fig. 4(b)) as a direct result of using a sparse GP prior, yielding efficient motion planning algorithms.
- The number of states to optimize can be significantly reduced by parameterizing the trajectory with a few support states and using O(1)-time GP interpolation, further improving efficiency.
- The factor graph structure also allows for incremental inference tools [30] to speed up the solution to incremental problems including online replanning.



(a) Estimation [18], [26], [20]

(b) Planning [22]

(c) STEAP [31]

Fig. 4: Factor graphs comparing GP-based approaches for reasoning about continuous-time trajectories. White circles are states of the trajectories, diamonds are landmarks; black circles are GP prior factors, and squares are measurement factors. Gray circles indicate the states at the current time.

It is evident from Fig. 4(a) and (b) that both estimation and planning share a similar factor graph structure in which the GP prior is defined as in Fig. 2. Both problems can be solved via MAP estimation on factor graphs. The fundamental difference is that the estimation graph is backward-looking while the planning graph is forward-looking. When an autonomous robot needs to perform these two tasks at the same time, a logical idea is to combine these two factor graph as a single one, and performing MAP estimation on the combined graph. This removes redundancy and allows for solving the two problems simultaneously. The factor graph for this technique, called "simultaneous trajectory estimation and planning" (STEAP) [31], is shown in Fig. 4(c). The information flow between the estimation and planning subgraphs enhances the overall quality of the solution and the factor graph structure allows the use of incremental inference tools, like iSAM2 [30]. Thus, STEAP can naturally solve closed-loop online replanning and yield a fast solution to trajectory estimation.

Estimation, planning, or STEAP can be derived from a common framework by simply changing the factors or the 'current state' designation. A unifying theme is the use of a sparse GP prior to represent trajectory distributions and MAP inference, i.e. GP regression, to find the optimal trajectory. Sparse GPs not only provide efficiency and model uncertainty, but also help bridge the gap between various robotics problems, allowing them to be combined and solved simultaneously. In the next section, we provide experimental results using our extension to sparse GPs on Lie groups for estimation and planning tasks that cannot be solved well by using states defined in vector space.

V. Experimental Results

We evaluated our framework on three different trajectory estimation tasks on SE(2), SO(3) and Sim(3), and one

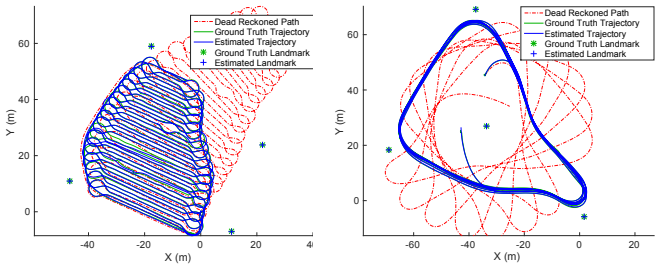


Fig. 5: Planar SLAM results on the Plaza1/2 datasets [13], with odometry-only and ground truth trajectories.

TABLE I: Planar SLAM comparison, Linear and SE(2).

	Plaza1		**Plaza2**	
	Linear	SE(2)	Linear	SE(2)
Position RMS (m)	0.252	0.238	0.523	0.152
Rotation RMS (deg)	2.822	2.508	1.952	0.981
Landmark RMS (m)	0.053	0.026	0.479	0.029
Optimization Time (s)	1.998	2.107	0.832	0.888

planning task on $\mathrm{SE}(2)\times\mathbb{R}^N$. We have open sourced the C+ code for our library that implements GPs on Lie group building on the GTSAM library[2] for both estimation[3] ar planning[4]. Trajectory estimation on SE(3) (a special case o our general Lie group formulation) has been reported in [26 and planning on vector space (a trivial Lie group) has bee reported in [21], [22], so we do not repeat those evaluation

A. SE(2): 2D Planar Range-Only SLAM

We conducted experiments on two range-only 2D SLAI problems to evaluate the proposed GP approach on SE(2). The datasets are *Plaza1* and *Plaza2* from Djugash *et al.* [13]. Both datasets were collected by a lawn mower equipped with a gyroscope and wheel encoders to measure odometry, and a radio node measuring ranges of four fixed beacons. Ground truth beacon positions and robot trajectories were measured by RTK-GPS. Fig. 5 shows the results, including ground-truth trajectories and landmark positions, and odometry-only dead reckoning trajectories. We plot trajectory error and trajectory distribution (shown by 3σ variance) estimated for Plaza1 dataset in Fig. 6. In the middle of the Plaza1 dataset, the robot loses range measurements to all beacons. From the estimated trajectory distribution we see the uncertainty increases in response to this missing data.

The performance of our approach on SE(2) is compared with the naïve sparse linear GP prior [18], which uses canonical coordinates $\boldsymbol{x}(t) = [x(t), y(t), \theta(t)]^\top \in \mathbb{R}^3$ as system state. The accuracy and efficiency of both approaches are evaluated by root mean square (RMS) error and optimization time respectively. Table I shows the comparison results. Both trajectory and landmark estimation accuracies are improved by our Lie group approach. Since both of approaches share the same sparse factor graph representation, there is no significant difference of efficiency between these two approaches.

[2]https://bitbucket.org/gtborg/gtsam
[3]https://github.com/gtrll/gpslam
[4]https://github.com/gtrll/gpmp2

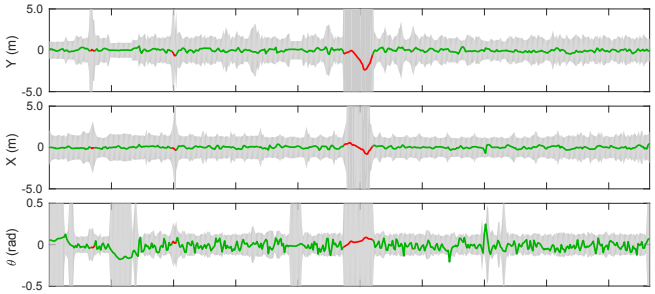


Fig. 6: Trajectory error and 3σ variance estimated of Plaza1 dataset. Green lines are states with range measurements, and

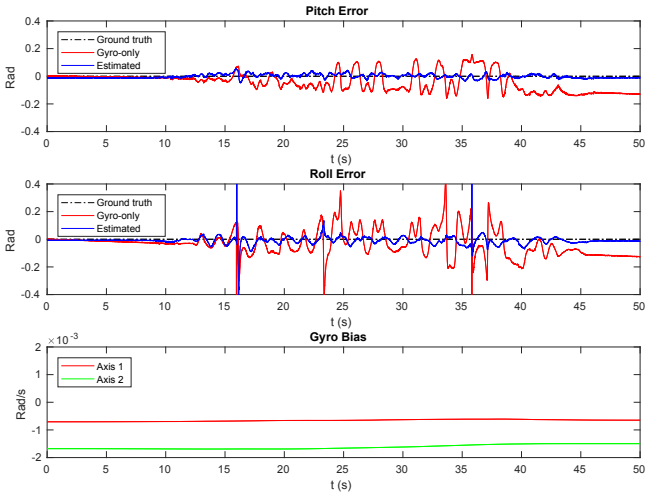


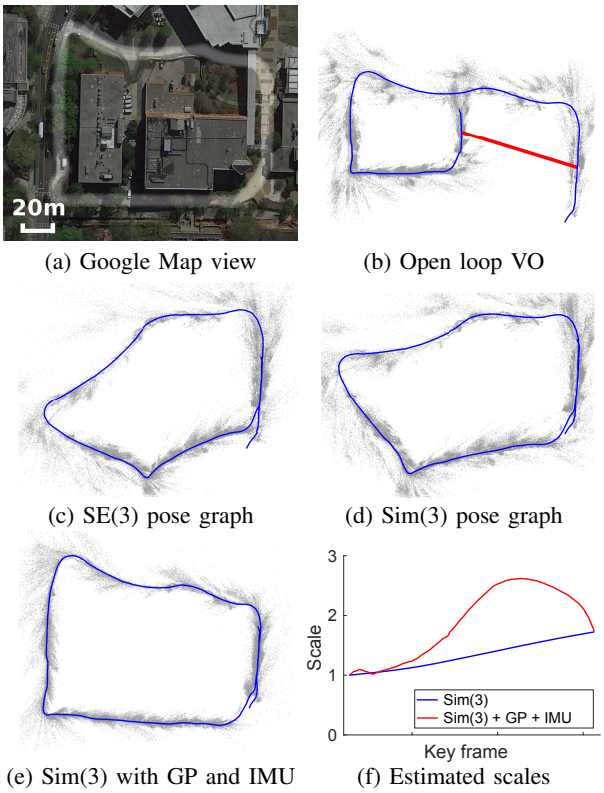
Fig. 7: Attitude estimation errors by proposed approach of IMU dataset, compared with gyroscope-only results, and estimated gyroscope bias by proposed approach.

B. SO(3): 3D Attitude Estimation of IMU

To evaluate our GP approach on SO(3), we designed an attitude estimation experiment using an inertial measurement unit (IMU). We collected an IMU dataset using a low-cost Pixhawk autopilot, which has a 3-axis accelerometer and a 3-axis gyroscope. Both acceleration and angular rates were collected asynchronously, as angular rate was available at 166Hz but acceleration was available at 40Hz. Ground truth attitude was collected by an Optitrack motion capture system.

A major advantage of using continuous-time trajectory representations is that estimation with asynchronous sensors can be accomplished simply. We implemented a batch attitude estimation approach with pre-integrated IMU factors [32] containing gyroscope measurements, and GP interpolated acceleration factors which compare acceleration measurements against gravity. After optimization, attitudes are estimated at the gyroscope time stamps. Since the accelerometer does not provide yaw angle information with respect to the world frame, only pitch and roll angles are estimated and compared with ground truth.

Fig. 7 shows the estimated pitch and roll angle errors from the IMU dataset, compared with gyroscope-only estimation, and compared with motion capture ground truth. The results show that both estimated pitch and roll angles are better aligned with ground truth, since the gyroscope drift from



(a) Google Map view (b) Open loop VO

(c) SE(3) pose graph (d) Sim(3) pose graph

(e) Sim(3) with GP and IMU (f) Estimated scales

Fig. 8: Scale drift aware monocular SLAM results. (a) Google Map© view of the map with path highlighted; (b) shows open loop VO results, with the loop closure marked in red; (c)-(e) are SE(3) and Sim(3) results, and (f) shows scale estimations of Sim(3) approaches.

bias is compensated for by fusing asynchronous acceleration measurements and sensor bias is also estimated. The existence of the few peaks in roll angle error plot are due to singularity caused by the Euler angle representation.

C. Sim(3): Scale Drift Aware Monocular SLAM

The transformation of a 3D rigid body is represented by SE(3), and most 3D SLAM approaches consider trajectories on SE(3). In monocular visual SLAM, camera motion and scene structure are recovered up to scale, due to the projective nature of a single camera. Although the scale of monocular visual SLAM is locally consistent, the scale estimate suffers from drift due to the lack of an anchor. Several approaches [24], [25] have been proposed to solve this issue by estimating the trajectory on Similarity group Sim(3) [24], which is defined by

$$\mathbf{S} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \mathbf{R} \in \mathrm{SO}(3), \tag{36}$$

where s is the estimated local scale.

We implemented sparse GPs on Sim(3) and conducted monocular visual SLAM experiments to show how GPs help with estimation on Sim(3). We built a hand-held monocular camera with an IMU, and collected a forward looking outdoor dataset as shown in Fig. 8(a). The camera and IMU readings are also collected asynchronously, similar to the IMU dataset. We used a visual odometry (VO) algorithm, similar to [33], but without IMU and GPS measurements. The VO result is illustrated in Fig. 8(b). We can clearly see the scale drift since the path is no longer a closed loop. We received a loop closure measurement marked in red (in Fig. 8(b)) which includes relative scale information [24].

We implemented and tested three methods for comparison: a 6-DOF SE(3) pose graph, a 7-DOF Sim(3) pose graph [24], and a 7-DOF Sim(3) pose graph with GP and IMU factors. All results are shown in Fig. 8. We can see that, compared to the ground truth map, the estimated 6-DOF pose graph is distorted due to scale drift; the 7-DOF Sim(3) pose graph is better but still distorted; and our 7-DOF Sim(3) on GP with IMU approach achieves the best result. Although a "ground truth" GPS dataset was collected, we don't report a quantitative evaluation using it for two reasons: (1) results are up to scale, so there is no direct way to compare these results with a GPS path with an associated metric scale, (2) the GPS noise is larger than the noise in our estimated trajectory.

To understand why the 7-DOF approach with GPs beats the one without GPs, we plot the estimated local scale for both approaches in Fig. 8(f). Although the loop closure gives relative scale information at the start and end of the estimation, there is no other information given in the middle of the trajectory, so the 7-DOF Sim(3) pose graph assumes the scale drift changes at a near-constant rate during the whole length of the trajectory. But that's not the case in the dataset. With the help from GP interpolation, asynchronous IMU measurements provide more information about how relative scale drifts in the middle of the trajectory, and we can see in Fig. 8(f) that our Sim(3) approach, which uses GPs and IMU, gives scale estimation of non-constant changing rate, leading to better SLAM results.

D. $SE(2)\times\mathbb{R}^N$: Mobile Manipulator Planning

To evaluate our framework on motion planning applications, we extended the Gaussian process motion planner 2 (GPMP2) [22] to general Lie groups from vector space. We set up a planning benchmark for mobile manipulators (a robot body mounted on a mobile base) with a state space $\mathrm{SE}(2)\times\mathbb{R}^N$, where the state space of the robot arm or upper body is $\mathbb{R}^N$ and the mobile base is SE(2) (the base has planar motion). We used TrajOpt [34] PR2 full-body planning dataset for our benchmark, and set up 36 full body planning problems for the PR2 robot. The state space of the PR2 is $\mathrm{SE}(2)\times\mathbb{R}^{15}$, 18 degrees of freedom (DOF), where 3 DOF is for the SE(2) 2D mobile base, 1 DOF for the torso vertical linear actuator, and 7 DOF each for the two arms.

We compared Lie group enabled GPMP2 with TrajOpt [34], a state-of-the-art trajectory optimizer. We ran both GPMP2 and TrajOpt with straight-line initializations in configuration space. As discussed in Sec. IV, a major benefit of using GPs in trajectory optimization is that the number of states optimized is reduced through GP interpolation, so we setup two GPMP2 benchmarks for comparison: one where GPMP2 is run with all 61 states to be optimized and no GP interpolation, called **GPMP2_noint**; and one where GPMP2

TABLE II: Full body planning benchmark results.

	GPMP2_int	GPMP2_noint	TrajOpt
Success rate (%)	72.22	63.89	50
Average runtime (s)	0.24	0.28	0.92
Maximum runtime (s)	0.48	0.55	1.45

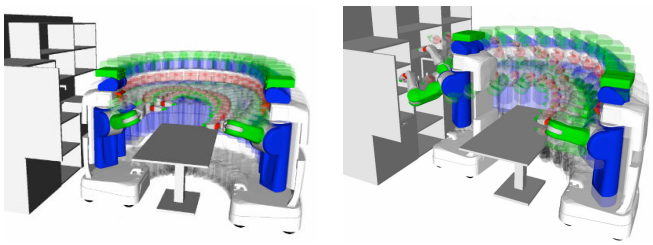


Fig. 9: Two example PR2 trajectories planned by GPMP2.

is run with 11 states to be optimized and collision cost is evaluated with 5 interpolated states between each optimized state pair (equaling 61 overall states), called **GPMP2_int**.

Benchmark results are shown in Table II, and two example trajectories planned by GPMP2 are shown in Fig. 9. Analogous to the results reported in [22] using vector space (on robots with the base fixed), GPMP2 on Lie group $\mathrm{SE}(2)\times\mathbb{R}^{15}$ is faster compared to TrajOpt. The sparsity of the Lie group-based GP prior maintains GPMP2's superior runtime performance. We can see how GP interpolation increases GPMP2's performance by comparing results of GPMP2_int and GPMP2_noint: with GP interpolation both success rate and speed improve.

VI. Conclusion

We present sparse GPs on Lie groups as a framework for reasoning about continuous-time trajectory distributions. We show that this representation is a general tool that can be utilized for various robotics tasks, including estimation and planning. Finally, we perform extensive experimental evaluations to show that our proposed framework works well in various estimation and planning tasks, particularly when a vector space representation of the robot state is inappropriate.

References

[1] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[2] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research*, vol. 25, pp. 1181–1203, Dec 2006.

[3] M. Bosse and R. Zlot, "Continuous 3D scan-matching with a spinning 2D laser," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 4312–4319, 2009.

[4] M. Li, B. H. Kim, and A. I. Mourikis, "Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2013.

[5] H. Dong and T. D. Barfoot, "Lighting-invariant visual odometry using lidar intensity imagery and pose interpolation," in *Field and Service Robotics (FSR)*, pp. 327–342, 2014.

[6] C. Bibby and I. Reid, "A hybrid SLAM representation for dynamic marine environments," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 257–264, 2010.

[7] S. Anderson and T. D. Barfoot, "Towards relative continuous-time SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2013.

[8] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *Intl. J. of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[9] A. Patron-Perez, S. Lovegrove, and G. Sibley, "A spline-based trajectory representation for sensor fusion and rolling shutter cameras," *Intl. J. of Computer Vision*, vol. 113, no. 3, pp. 208–219, 2015.

[10] P. Furgale, C. H. Tong, T. D. Barfoot, and G. Sibley, "Continuous-time batch trajectory estimation using temporal basis functions," *Intl. J. of Robotics Research*, vol. 34, no. 14, pp. 1688–1710, 2015.

[11] S. Anderson, F. Dellaert, and T. Barfoot, "A hierarchical wavelet decomposition for continuous-time SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014.

[12] J. Djugash, B. Hamner, and S. Roth, "Navigating with ranging radios: Five data sets with ground truth," *J. of Field Robotics*, vol. 26, no. 9, pp. 689–695, 2009.

[13] C. H. Tong, P. Furgale, and T. D. Barfoot, "Gaussian process gauss–newton for non-parametric simultaneous localization and mapping," *Intl. J. of Robotics Research*, vol. 32, no. 5, pp. 507–525, 2013.

[14] C. H. Tong, S. Anderson, H. Dong, and T. D Barfoot, "Pose interpolation for laser-based visual odometry," *J. of Field Robotics*, vol. 31, no. 5, pp. 731–757, 2014.

[15] R. Eustice, H. Singh, and J. Leonard, "Exactly sparse delayed-state filters for view-based SLAM," *IEEE Trans. Robotics*, vol. 22, pp. 1100–1114, Dec 2006.

[16] T. Barfoot, C. H. Tong, and S. Sarkka, "Batch continuous-time trajectory estimation as exactly sparse gaussian process regression," *Robotics: Science and Systems (RSS)*, 2014.

[17] S. Anderson, T. D. Barfoot, C. H. Tong, and S. Särkkä, "Batch nonlinear continuous-time trajectory estimation as exactly sparse gaussian process regression," *Autonomous Robots*, vol. 39, no. 3, pp. 221–238, 2015.

[18] X. Yan, V. Indelman, and B. Boots, "Incremental sparse GP regression for continuous-time trajectory estimation & mapping," *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, 2015.

[19] M. Mukadam, X. Yan, and B. Boots, "Gaussian process motion planning," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2016.

[20] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using Gaussian processes and factor graphs," in *Robotics: Science and Systems (RSS)*, 2016.

[21] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots., "Towards robust skill generalization: Unifying learning from demonstration and motion planning," in *Proceedings of the 2017 Conference on Robot Learning (CoRL)*, 2017.

[22] H. Strasdat, J. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM," *Robotics: Science and Systems (RSS)*, 2010.

[23] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conf. on Computer Vision (ECCV)*, 2014.

[24] S. Anderson and T. D. Barfoot, "Full STEAM ahead: Exactly sparse gaussian process regression for batch continuous-time trajectory estimation on SE (3)," *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[25] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*, vol. 2. Springer Science & Business Media, 2011.

[26] J. Dong, B. Boots, and F. Dellaert, "Sparse Gaussian processes for continuous-time trajectory estimation on matrix Lie groups," *arXiv preprint arXiv:1705.06020*, 2017.

[27] R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

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

[29] M. Mukadam, J. Dong, F. Dellaert, and B. Boots, "Simultaneous trajectory estimation and planning via probabilistic inference," in *Robotics: Science and Systems (RSS)*, 2017.

[30] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," *Robotics: Science and Systems (RSS)*, 2015.

[31] J. Dong, J. G. Burnham, B. Boots, G. Rains, and F. Dellaert, "4D crop monitoring: Spatio-temporal reconstruction for agriculture," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017.

[32] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *Intl. J. of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.