Gaussian Process Gauss–Newton for non-parametric simultaneous localization and mapping
Chi Hay Tong, Paul Furgale and Timothy D. Barfoot
The International Journal of Robotics Research 2013 32: 507
DOI: 10.1177/0278364913478672

The online version of this article can be found at:
http://ijr.sagepub.com/content/32/5/507
Gaussian Process Gauss–Newton for non-parametric simultaneous localization and mapping

Chi Hay Tong¹, Paul Furgale² and Timothy D. Barfoot¹

Abstract
In this paper, we present Gaussian Process Gauss–Newton (GPGN), an algorithm for non-parametric, continuous-time, nonlinear, batch state estimation. This work adapts the methods of Gaussian process (GP) regression to address the problem of batch simultaneous localization and mapping (SLAM) by using the Gauss–Newton optimization method. In particular, we formulate the estimation problem with a continuous-time state model, along with the more conventional discrete-time measurements. Two derivations are presented in this paper, reflecting both the weight-space and function-space approaches from the GP regression literature. Validation is conducted through simulations and a hardware experiment, which utilizes the well-understood problem of two-dimensional SLAM as an illustrative example. The performance is compared with the traditional discrete-time batch Gauss–Newton approach, and we also show that GPGN can be employed to estimate motion with only range/bearing measurements of landmarks (i.e. no odometry), even when there are not enough measurements to constrain the pose at a given timestep.

Keywords
estimation, SLAM, continuous time, Gaussian processes

1. Introduction
In robotics, the conventional formulation of state estimation algorithms is expressed in discrete time. While this approach has served the robotics community well in the past, we propose an alternative formulation with numerous advantages, that may enable future applications.

Although the methods presented in this paper generalize to other state estimation problems, throughout this paper, we utilize the two-dimensional simultaneous localization and mapping (SLAM) problem as our illustrative example. This problem is well-understood, and provides opportunities to draw parallels to existing techniques. In this problem, a robot is travelling continuously, while obtaining measurements from a nonlinear, (possibly) non-invertible sensor model at discrete times. The state in this scenario is composed of the robot poses, and the landmark locations. Furthermore, we consider the estimation problem over a window of time, where numerous measurements have been obtained. Therefore, the goal of the SLAM problem is to estimate the robot poses at some timesteps of interest, as well as the landmark locations, given the batch of measurements.

This common scenario in mobile robotics is typically addressed by discretizing the robot trajectory, and computing an estimate of the robot pose at each measurement instant. This is illustrated in Figure 1(a), where the robot poses are enumerated in sequence, and the measurement timestamps discarded. However, this approach does not reflect the true physical reality of the underlying system. In reality, the robot motion is continuous, with characteristics defined by the physical and temporal constraints of the mobile platform. Therefore, we propose to model the estimated robot trajectory in continuous time, as depicted in Figure 1(b).

The continuous-time process model offers a number of advantages. The discrete-time approach of computing an estimate at every measurement instant can be restrictive, because without a motion model, non-invertible measurements impose the requirement of multiple sensor readings to produce an estimate for every robot pose. This requirement can be problematic for high-rate sensors such as inertial measurement units, rolling shutter cameras, or...
scanning laser systems, which tend to produce measurements at unique and possibly asynchronous times. This problem is intensified if we desire additional state estimates at instants between measurement timesteps. For example, a laser rangefinder in a pushbroom configuration could be used for accurate terrain modelling, if we were able to provide a smooth estimate of the robot trajectory (using some other sensors for motion estimation). These issues of measurement scarcity are addressed by continuous-time state estimation, because the underlying continuous-time process model provides sufficient information to produce an estimate at any instant of time.

In the past literature, the traditional discrete-time extended Kalman filter (EKF) (Smith et al., 1990) has also been formulated in continuous time (Kalman and Bucy, 1961; Jazwinski, 1970). Unfortunately, these filtering methods require numerical integration for practical implementation, and can produce jumps in the estimates due to the Markov assumption. Similar implementation issues arise for forward/backward smoothing methods (Simon, 2006). A commonly used discrete-time alternative to filtering considers the measurements as a batch, and estimates multiple states simultaneously. This approach maintains estimator consistency, and can produce visually smooth trajectories when utilizing a very large number of discrete poses (Smith et al., 2003). Unfortunately, there are limited scenarios where estimating a large number of poses is computationally efficient, and a process model is required. Instead, we avoid computing a large number of poses by exploiting the underlying smooth trajectory, and utilize a batch optimization formulation in continuous time.

To develop a new method of continuous-time batch state estimation in a framework amenable to robotics, we look to current work from the machine learning community addressing the regression problem. The regression problem considers the case where sample outputs are obtained from an unknown continuous function, and the goal is to predict the value of the function for a new input. Additional complexity arises when the sample outputs have corruptive noise, which requires probabilistic approaches. The state estimation problem in robotics has a similar form, where we start with noisy measurements, and seek to estimate states at given instants of time. However, instead of making further measurement predictions, we seek to determine the latent states that generated the measurements. In the SLAM example, the latent states are the robot poses and landmark locations. This task is complicated by the presence of nonlinear, non-invertible sensor models.

Solutions to the regression problem can be divided into two categories: parametric and non-parametric. In parametric regression, the unknown function is assumed to have a particular form, which is defined by a set of parameters. For example, the unknown function can be modelled as a polynomial, or the weighted sum of a set of known basis functions. With this assumed model, the regression problem reduces to finding the optimal parameter values that produce the best matches to the sample outputs. These optimized parameters can then be used for predicting new outputs. This is a simple and effective method, but its accuracy is highly dependent on the suitability of model selection. For parametric regression, the computational complexity is related to the number of chosen parameters.

The discrete-time batch state estimation formulation shares some similarities with the parametric regression approach, since the robot state is parametrized by discrete poses at instants of time (Figure 1(a)). Determination of the optimal parameter values addresses the state estimation problem. This approach can be extended to the continuous-time domain by utilizing a continuous state representation, such as piecewise splines (Bibby and Reid, 2010; Furgale et al., 2012). The parametric spline approach is a viable method for continuous-time state estimation, and is currently an active research area.

In this paper, we present an alternative approach based on non-parametric regression. Rather than assuming a parametric form, non-parametric techniques allow the unknown function to implicitly lie in a set of functions, which offers significantly more representational power. However, instead of producing a model from the data, the implicit models are resolved by using the data itself to produce predictions. As a result, the computational complexity scales with the number of measurements.

Gaussian process (GP) regression is a non-parametric method where the underlying function is modelled by a mean and covariance function (Rasmussen and Williams,
2006). This probabilistic representation accounts for uncertainty in the observations, and natively suppresses model complexity to avoid overfitting. As a result, it is a suitable candidate for adaptation to the continuous-time batch state estimation problem. Unfortunately, this is not a straightforward task, because we are interested in estimating the latent states, and the measurement models are nonlinear and non-invertible. We accomplish our task by modelling the robot state directly as a GP with the input of time, and utilize the Gauss–Newton algorithm (Gauss, 1855) to recover the latent states from the measurements.

In summary, we present the Gaussian Process Gauss–Newton (GPGN) algorithm in this paper. GPGN is a non-parametric, continuous-time, nonlinear, batch state estimation algorithm, which utilizes concepts from GP regression applied to the robotics domain. This is a novel application of GPs, which differs from the previous uses of GPs in robotics. In particular, we present an algorithm that can be applied to conventional state estimation scenarios. Parallels to the discrete-time approach are discussed, and advantages are highlighted through experimental illustration. GPGN should be considered as an alternative to the parametric approach of Furgale et al. (2012) for solving the continuous-time batch state estimation problem.

This paper presents an extension to the state estimation method presented by Tong et al. (2012), where GPGN has been reformulated to address the SLAM problem by considering a state containing both time-varying and time-invariant quantities. Furthermore, we include two derivations to reflect both the weight-space and function-space approaches from the GP regression literature (Rasmussen and Williams, 2006), and develop an efficient expression for predicting the value of the continuous-time state at additional times of interest. This leads to an approximative method that significantly reduces the computational requirements. Additional experimentation and discussion are also presented, highlighting the tradeoffs between GPGN and the conventional discrete-time Gauss–Newton method. Finally, two appendices are included that detail how to obtain physically motivated covariance functions for a moving point, incorporate prior uncertainty, and utilize derivative-type measurements in the GPGN framework.

The remainder of this paper is organized as follows. We begin with a discussion of related applications of GPs to robotic state estimation in Section 2. This is followed by the derivation of GPGN algorithm in Section 3, and experimental validation in Section 4. Finally, concluding remarks are provided in Section 5.

2. Literature review

The application of GPs to the field of robotics is not a new concept. GPs have been applied successfully to learn measurement models for complex systems such as laser rangefinders (Plagemann et al., 2007), map gas distributions (Stachniss et al., 2009), and perform large-scale terrain modelling (Vasudevan et al., 2009). In fact, the concept of using GPs for modelling terrain can be traced back to the field of geostatistics, where it was termed kriging (Matheron, 1963). These applications utilized GPs in their original formulation, since GPs were employed for modelling sensor outputs. Recent work by Guizilini and Ramos (2012) has adapted this formulation to the state estimation problem by using GPs to learn the mapping from optical flow vectors to metric visual odometry estimates.

GP measurement models have also been utilized for state estimation via particle filtering (Ferris et al., 2006), and recently, generalized into a discrete-time filtering framework with GP Bayes filters (GPBFs) (Ko and Fox, 2009). Our work differs greatly from these approaches. Rather than applying GPs to first model the sensors and then utilize them for discrete-time filtering, we use GPs to perform the state estimation itself. That is, we do not model the sensors with GPs; we model the state itself with a GP, and utilize a batch estimation framework.

In recent literature, the concept of GP latent variable models (GPLVMs) (Lawrence, 2003) has also been applied to the robotics domain. GPLVMs were originally developed as a method for dimensionality reduction, which provided a mapping between high-dimensional GP measurements to a lower-dimensional GP latent state. This concept was applied to problems including Wifi-based SLAM (WiFiSLAM) (Ferris et al., 2007) and human motion tracking (Wang et al., 2008). In addition, GPLVMs were used to train GPBFs without ground truth measurements (Ko and Fox, 2011). However, GPLVMs require a number of assumptions for successful performance in the robotics domain. Since GPLVMs were originally conceived as a dimensionality reduction technique, the assumption that similar sensor measurements are obtained from similar locations is required. This assumption does not hold when utilizing non-invertible measurement models. Furthermore, the lower-dimensional representation produced does not guarantee any relevance to the scenario at hand. To address this issue, additional priors are required to guide the process to produce physically meaningful results for the latent states, such as robot poses.

Although we also model the hidden state as a GP, in our approach, we retain the discrete-time sensor models. This allows GPGN to address conventional state estimation scenarios. By restricting the modification to the estimator’s internal representation of the state, established methods such as observability analysis still apply. Furthermore, by using the traditional sensor models, the latent states represented by the GP retain their physical relevance.

Other related works in the machine learning literature include derivative GP observations (Solak et al., 2003), warped GPs (Snelson et al., 2004), and linear operator measurements (Särkkä, 2011), which share some similarities to how we utilize a discrete-time measurement model, as
opposed to direct observations of the state. A similar problem was considered by Archambeau et al. (2007), where the evolution of a stochastic differential equation (SDE) was approximated with a GP based on discrete observations. However, the presented solution was restricted to simplified GPs that were parametrized by time-invariant coefficients. The inverse approach was also taken by Särkkä and Hartikainen (2012), which addressed the GP regression problem by converting the GP prior into a SDE, and performed Kalman filtering and smoothing for an online learning process. Our work differs from these approaches because we address the batch state estimation problem, utilize nonlinear measurement models, and apply the conventional Gauss–Newton method (Gauss, 1855). In addition, we present our approach in a format more familiar to the robotics community.

3. Gaussian Process Gauss–Newton

In this section, we provide the derivations for GPGN, which addresses the batch nonlinear state estimation problem. For generality, we consider a state composed of both a time-varying quantity, \( x(t) \), and a time-invariant component, \( \ell \). In the SLAM scenario, the time-varying quantity could represent the robot pose, and the time-invariant component could be calibration parameters or landmark locations. Our task is to determine the values of the state given a set of \( N \) measurements obtained over a period of time.

We begin by providing a brief mathematical overview of GPs, and follow with the definition of the system models for the state estimation problem. Two derivations are provided in this section, reflecting both the weight-space and function-space approaches from the GP regression literature (Rasmussen and Williams, 2006). The weight-space formulation provides clear parallels to the discrete-time approach, but the function-space formulation is considerably more concise. In addition to the core derivations, an efficient expression for predicting the value of the continuous-time state at additional times of interest is also developed, which leads to an approximation method that significantly reduces the computational requirements. The section is then concluded with a summary of the GPGN algorithm.

3.1. Background

The most commonly used discrete-time probability distribution in mobile robotics is the Gaussian random variable. A Gaussian random variable, \( x \), is expressed as

\[
x \sim \mathcal{N}(\mathbf{m}, \mathbf{K}),
\]

where \( \mathbf{m} \) is the mean vector, and \( \mathbf{K} \) is the covariance matrix. A GP can be considered as a generalization of a Gaussian random variable to the continuous-time domain (Rasmussen and Williams, 2006). Instead of a mean vector and a covariance matrix, a GP, \( x(t) \), is described by a mean function, \( \mu(t) \), and a covariance function, \( \mathcal{K}(t, t') \):

\[
x(t) \sim \mathcal{G}(\mu(t), \mathcal{K}(t, t')).
\]

These expressions are functions of time, and the covariance function involves two time variables to account for cross-temporal relations. Evaluating these expressions at discrete instants of time results in jointly Gaussian random variables.

3.2. Problem statement

The state space models for the underlying system are defined to be

\[
x(t) \sim \mathcal{G}(\mu(t), \mathcal{K}(t, t'))
\]

\[
\ell \sim \mathcal{N}(\mathbf{d}, \mathbf{L})
\]

\[
z_i = h_i(x(t_i), \ell) + n_i, \quad n_i \sim \mathcal{N}(0, \mathbf{R}_i)
\]

where \( x(t) \) is a GP with mean and covariance functions \( \mu(t) \) and \( \mathcal{K}(t, t') \), respectively, \( \ell \) is a time-invariant discrete Gaussian random variable with prior mean of \( \mathbf{d} \) and covariance \( \mathbf{L} \), and the measurements, \( z_i \), are obtained through a conventional nonlinear, (possibly) non-invertible measurement model, \( h_i(\cdot) \), at \( N \) discrete times, \( t_i \). For simplicity, we have modelled the measurement noise, \( n_i \), as additive, zero-mean, and Gaussian with covariance \( \mathbf{R}_i \).

In contrast to the initial step for GP regression where the additional parameters specifying the GP properties (known as hyperparameters) are first trained using the data, it should be noted that we do not address the specification of hyperparameters in this paper. Rather, we assume that they have already been obtained through some prior system identification process. In practice, these hyperparameters may be related to the physical properties of the underlying system, if an appropriate GP function is chosen. This assumption of prior characterization is typical for state estimation algorithms. For example, the same assumption is applied in the measurement model for \( \mathbf{R}_i \).

To simplify the expressions in the following sections, we begin by combining the two state components by defining

\[
\theta(t) := \begin{bmatrix} x(t) \\ \ell \end{bmatrix}, \quad \eta(t) := \begin{bmatrix} \mu(t) \\ \mathbf{d} \end{bmatrix},
\]

\[
\mathcal{P}(t, t') := \begin{bmatrix} \mathcal{K}(t, t') & 0 \\ 0 & \mathbf{L} \end{bmatrix},
\]

which results in the simplified system equations:

\[
\theta(t) \sim \mathcal{G}(\eta(t), \mathcal{P}(t, t'))
\]

\[
z_i = h_i(\theta(t_i)) + n_i.
\]

It should be noted that we maintain the conventional assumption of independence between the robot poses and landmark position priors in the definition of \( \mathcal{P}(t, t') \). Correlations between the pose and landmark estimates arise when the measurements are incorporated.
3.3. Weight-space derivation

The weight-space derivation begins with a parametric basis function representation for the state. This approach is used as an intermediate step, which allows us to illustrate the parallels to the conventional discrete-time approach by providing a familiar Gaussian random variable formulation. Algebraic manipulations are then conducted to produce a non-parametric form, resulting in the GPGN algorithm.

3.3.1. Basis function representation

In discrete-time batch estimation, the standard approach utilizes a sum-of-squares objective function of the form $J := \frac{1}{2} \sum_{i=1}^{N} e_i^T W_i e_i$, where $e_i$ is a measurement error term, and $W_i$ is its associated weight. If the weight is chosen to be the inverse covariance matrix of $e_i$, it can be shown that this objective function produces the maximum likelihood (ML) solution.

Similarly, we utilize a ML objective function in our derivation. As an intermediate step, we begin by adopting a basis function representation for the state

$$\theta(t) := \Psi(t)b,$$

(9)

where $\Psi(t)$ are the basis functions and $b$ are the coefficients defined by

$$\Psi(t) := \begin{bmatrix} \Phi(t) & 0 \\ 0 & 1 \end{bmatrix}, \quad b := \begin{bmatrix} c \\ \ell \end{bmatrix}. \quad (10)$$

In this representation, the time-varying component of the state is replaced by $\Phi(t)$, a stack of $M$ known temporal basis functions,

$$\Phi(t) := \left[ \phi_1(t) \ldots \phi_M(t) \right], \quad (11)$$

and $c$, a column of coefficients. Although this representation would convert this problem into a parametric estimation problem (Furgale et al., 2012), we assume that the number of basis functions, $M$, is very large, or even infinite. This provides substantial representational power, but with the restriction that we cannot evaluate or store either $\Phi(t)$ or $c$. The non-parametric form that we will obtain in the following section avoids these issues.

With this representation, the measurement model becomes

$$z_i = h_i(\Psi(t_i)b) + n_i,$$

(12)

and, similarly, the mean and covariance functions of $\theta(t)$ become

$$\eta(t) := \Psi(t)u,$$

$$\mathcal{P}(t, t') := \Psi(t)P \Psi(t')^T,$$

(13)\hspace{1cm}(14)

where $u$ and $P$ are defined by

$$b \sim \mathcal{N}(u, P).$$

(15)

For clarity, we keep the time-varying and time-invariant components separate by defining

$$u := \begin{bmatrix} m \\ d \end{bmatrix}, \quad P := \begin{bmatrix} K & 0 \\ 0 & L \end{bmatrix}. \quad (16)$$

These substitutions return us to the familiar Gaussian random variable domain, where (14) was obtained by invoking the ‘kernel trick’ from the machine learning literature (Rasmussen and Williams, 2006) in an inverse manner. This relation will be utilized later in Section 3.3.2 to resolve the issue of computational intractability due to the large number of basis functions.

We can then form a ML objective function using these definitions:

$$J := \frac{1}{2} \sum_{i=1}^{N} (z_i - h_i(\Psi(t_i)b))^T R_i^{-1} (z_i - h_i(\Psi(t_i)b))$$

$$+ \frac{1}{2} (b - u)^T P^{-1} (b - u). \quad (17)$$

To find the minimum of this objective function, we take the Gauss–Newton approach (Gauss, 1855) of linearizing the error terms, minimizing the resulting quadratic function, and iterating until convergence. To linearize the measurement model, we make the assumption that the state is approximated by the value of the current estimate, $\bar{\theta}(t)$, and an additive perturbation, $\delta \bar{\theta}(t)$. Applying the basis function representation (9), we get

$$\theta(t) \approx \bar{\theta}(t) + \delta \bar{\theta}(t)$$

$$= \Psi(t)\bar{b} + \Psi(t) \delta b,$$

(18)

where $\bar{b}$ is our current estimate for the coefficients, and $\delta b$ is the perturbation. At each iteration, we seek the optimal value of the perturbation, $\delta b^*$, which we apply to bring our estimate progressively closer to the optimal value of the state. Under this assumption, our linearized system models are

$$\bar{b} + \delta b \sim \mathcal{N}(u, P),$$

$$z_i \approx h_i(\bar{\theta}(t_i)) + H_i(\bar{\theta}(t_i))\delta b + n_i, \quad H_i := \frac{\partial h_i}{\partial \theta} \bigg|_{\bar{\theta}}. \quad (19)$$

(20)

It should be noted that we require a value for the state at each measurement time to perform linearization. Since each measurement may come at a unique time, updating and storing these values can be prohibitive if there are a large number of measurements. This concern motivates the approximation method developed later in Section 3.6.

Finally, to simplify the objective function, we define

$$z := \begin{bmatrix} z_1 \\ \vdots \\ z_N \end{bmatrix}, \quad h := \begin{bmatrix} h_1(\bar{\theta}(t_1)) \\ \vdots \\ h_N(\bar{\theta}(t_N)) \end{bmatrix}, \quad H := \begin{bmatrix} H_1 \\ \vdots \\ H_N \end{bmatrix},$$

$$W := \begin{bmatrix} \Phi(t_1) \\ \vdots \\ \Phi(t_N) \end{bmatrix}, \quad R := \begin{bmatrix} R_1 \\ \vdots \\ R_N \end{bmatrix}. \quad (21)$$
which results in
\[
J = \frac{1}{2} (z - h - H \Psi \delta b)^T R^{-1} (z - h - H \Psi \delta b) + \frac{1}{2} (b + \delta b - u)^T P^{-1} (b + \delta b - u). \tag{22}
\]
This expression is in the same form as the conventional discrete-time Gauss–Newton objective function. As a result, we proceed in the usual manner. Taking the derivative with respect to \(\delta b\) gives us
\[
\frac{\partial J}{\partial \delta b} = (-H \Psi)^T R^{-1} (z - h - H \Psi \delta b) + P^{-1} (b + \delta b - u), \tag{23}
\]
and setting the value to zero provides us with \(\delta b^*\), the optimal value of the perturbation that minimizes the quadratic expression (at this iteration):
\[
(\Psi^T H^T R^{-1} H \Psi + P^{-1}) \delta b^* = \Psi^T H^T R^{-1} (z - h)
\]
\[
= A^{-1} (z - h - P^{-1} (b - u)). \tag{24}
\]
As can be seen, we now have correlations between the poses and the landmarks due to the landmark measurements. Since this derivation follows the Gauss–Newton approach, we can express the mean, \(\delta b^*\), as
\[
\delta b^* = A^{-1} (\Psi^T H^T R^{-1} (z - h) - P^{-1} (b - u)), \tag{25}
\]
and the covariance as
\[
\text{cov}(\delta b^*, \delta b^*) = A^{-1}. \tag{26}
\]
These expressions provide the solution to the parametric estimation problem.

3.3.2. Returning to a non-parametric form

We return to a non-parametric form by first applying the basis function relation \(\delta \theta^*(t) = \Psi(t) \delta b^*\), which produces the mean function
\[
\delta \theta^*(t) = \Psi(t) A^{-1} \Psi^T H^T R^{-1} (z - h) - \Psi(t) A^{-1} P^{-1} (b - u), \tag{27}
\]
and the associated covariance function
\[
\text{cov}(\delta \theta^*(t), \delta \theta^*(t')) = \Psi(t) A^{-1} \Psi(t')^T. \tag{28}
\]
To manipulate these expressions into a form that we can evaluate, some identities are employed. It can be shown using the Sherman–Morrison–Woodbury (SMW) identity (Sherman and Morrison, 1950; Woodbury, 1950) that
\[
A^{-1} \equiv P - P \Psi^T H^T (R + H \Psi P \Psi^T H^T)^{-1} H \Psi P,
\]
and
\[
A^{-1} \Psi^T H^T \equiv P \Psi^T H^T (R + H \Psi P \Psi^T H^T)^{-1} R.
\]
Substituting these identities into (27)-(28), and reapplying the basis function relations (13)–(14) produces the GPGN update expressions
\[
\delta \theta^*(t) = \Psi(t) P \Psi^T H^T (R + H \Psi P \Psi^T H^T)^{-1} R R^{-1} (z - h) - \Psi(t) (P - P \Psi^T H^T (R + H \Psi P \Psi^T H^T)^{-1} H \Psi P) \times P^{-1} (b - u)
\]
\[
= \Psi(t) (u - \tilde{b}) + \Psi(t) P \Psi^T H^T (R + H \Psi P \Psi^T H^T)^{-1} \times (z - h - H \Psi (u - \tilde{b}))
\]
\[
= (\eta(t) - \tilde{\eta}(t)) + \mathcal{P}(t) H^T (R + H \mathcal{P} H^T)^{-1} \times (z - h - H (\eta - \tilde{\eta})), \tag{31}
\]
and
\[
\text{cov}(\delta \theta^*(t), \delta \theta^*(t'))
\]
\[
= \mathcal{P}(t, t') - \mathcal{P}(t) H^T (R + H \mathcal{P} H^T)^{-1} H \mathcal{P}(t'), \tag{32}
\]
where
\[
\eta := \begin{bmatrix} \mu \\ \ell \end{bmatrix}, \quad \mu := \begin{bmatrix} \mu(t_1) \\ \vdots \\ \mu(t_N) \end{bmatrix}, \quad \bar{\eta} := \begin{bmatrix} \bar{x} \\ \bar{\ell} \end{bmatrix}, \quad \bar{x} := \begin{bmatrix} \bar{x}(t_1) \\ \vdots \\ \bar{x}(t_N) \end{bmatrix},
\]
\[
\mathcal{K}(t) := \Phi(t) K \Phi^T = [\mathcal{K}(t, t_1) \ldots \mathcal{K}(t, t_N)],
\]
\[
\mathcal{K} := \Phi \Phi^T = [\mathcal{K}(t_1, t_1) \ldots \mathcal{K}(t_1, t_N) \\ \vdots \\ \mathcal{K}(t_N, t_1) \ldots \mathcal{K}(t_N, t_N)],
\]
\[
\mathcal{P}(t) := \Psi(t) \Psi^T = \begin{bmatrix} \mathcal{K}(t) 0 \\ 0 \mathcal{L} \end{bmatrix},
\]
\[
\mathcal{P} := \Psi \Psi^T = \begin{bmatrix} \mathcal{K} 0 \\ 0 \mathcal{L} \end{bmatrix}. \tag{33}
\]
The substitution of \(\Psi(t) P \Psi(t')^T\) by \(\mathcal{P}(t, t')\) makes use of the ‘kernel trick’, which recovers computational tractability. The \(\mathcal{K}\) and \(\mathcal{K}(t)\) matrices are constructed by simply evaluating the covariance function, \(\mathcal{K}(t, t')\), at the time of interest, \(t\), and the measurement times \(t_1, \ldots, t_N\).

Further simplifications can be achieved if we consider the perturbations of the state at only the measurement times. That is, if we define \(\delta \theta^* := \begin{bmatrix} \delta x^T(t_1) \\ \delta x^T(t_2) \\ \vdots \\ \delta x^T(t_N) \end{bmatrix}\), where \(\delta x^* := \begin{bmatrix} \delta x^* (t_1) \\ \delta x^* (t_2) \\ \vdots \\ \delta x^* (t_N) \end{bmatrix}\), we get
\[
\delta \theta^* = (\eta - \bar{\theta}) + \mathcal{P}(t, t') H (R + H \mathcal{P} H^T)^{-1} \times (z - h - H \eta), \tag{34}
\]
\[
\text{cov}(\delta \theta^*, \delta \theta^*) = \mathcal{P} - \mathcal{P}(t, t') H (R + H \mathcal{P} H^T)^{-1} H \mathcal{P}, \tag{35}
\]
and applying the SMW identity produces the update equations in information form:

\[
(\mathbf{P}^{-1} + \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}) \mathbf{\delta \theta}^* = \mathbf{P}^{-1} (\mathbf{\eta} - \mathbf{\bar{\eta}}) + \mathbf{H}^T \mathbf{R}^{-1} (\mathbf{z} - \mathbf{h}),
\]

(36)

\[
\text{cov}(\mathbf{\delta \theta}^*, \mathbf{\delta \theta}^*)^{-1} = \mathbf{P}^{-1} + \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}.
\]

(37)

As can be seen, these expressions have similar structure to the conventional discrete-time batch Gauss–Newton iterations. In fact, the estimate update expressions (36)–(37) simply have the additional \(\mathbf{P}^{-1}\) and \(\mathbf{P}^{-1} (\mathbf{\eta} - \mathbf{\bar{\eta}})\) components, which serve as regularization terms (i.e. the GP prior serves as a smoothing factor for the solution).

Furthermore, the overall structure of the linear system of equations is not significantly modified with the inclusion of the regularization terms. This is depicted in Figure 2, where we compare the sparsity pattern of the inverse covariance matrix between the conventional discrete-time batch Gauss–Newton formulation with different choices of covariance functions for GP in the SLAM scenario. The lower-right block of the matrix remains block-diagonal, which allows for efficient solutions by applying the Schur complement. The application of the Schur complement is also known as sparse bundle adjustment (SBA) in the computer vision literature (Brown, 1958), when there are no odometry measurements.

Further computational savings can also be attained in the SLAM problem by considering the structure of the upper-left block of the matrix. As can be seen in Figure 2(a), the discrete-time Gauss–Newton formulation produces a banded-diagonal structure, which can be solved through efficient means. Owing to the additional regularization terms, the computational requirements of inverting the upper-left block in GP in is related to the choice of covariance function in the GP prior. In Section 4, we utilize a physically motivated covariance function, which is derived in Appendix A. Unfortunately, this results in a fully dense covariance matrix, which is depicted in Figure 2(b). However, as depicted in Figure 2(c), the banded-diagonal structure can be recovered through the use of alternative covariance functions that provide compact support (Moreaux, 2008), but the link to physical reality may be lost.

After solving for the optimal perturbation of the state at the measurement times using (36), it is applied as an additive update to the current state estimate by \(\mathbf{\bar{\theta}} \leftarrow \mathbf{\bar{\theta}} + \mathbf{\delta \theta}^*\). The system is then relinearized using the improved state estimate, and the process repeats until convergence.

### 3.4. Function-space derivation

The function-space derivation provides an alternative to the weight-space approach provided in the previous section. Both approaches achieve the same result (31)–(32), but the function-space derivation is considerably more concise.

We begin with our linearized measurement model

\[
\mathbf{z}_t \approx \mathbf{h}_t(\mathbf{\bar{\theta}}(t)) + \mathbf{H}_t \mathbf{\delta \theta}(t) + \mathbf{n}_t,
\]

(38)

and use it to obtain a Gaussian approximation. That is, the mean of the measurement function is

\[
\mathbb{E}[\mathbf{z}] = \mathbb{E}[\mathbf{h}_t(\mathbf{\bar{\theta}}(t)) + \mathbf{H}_t \mathbf{\delta \theta}(t) + \mathbf{n}_t] = \mathbf{h}_t(\mathbf{\bar{\theta}}(t)) + \mathbf{H}_t \mathbb{E}[\mathbf{\delta \theta}(t)] = \mathbf{h}_t(\mathbf{\bar{\theta}}(t)) + \mathbf{H}_t (\mathbf{\eta}(t) - \mathbf{\bar{\theta}}(t)),
\]

(39)

and the covariance is

\[
\mathbb{E}[(\mathbf{z} - \mathbb{E}[\mathbf{z}])(\mathbf{z} - \mathbb{E}[\mathbf{z}])^T] = \mathbb{E}[(\mathbf{n} - \mathbf{H}_t(\mathbf{\eta}(t) - \mathbf{\bar{\theta}}(t)))(\mathbf{n} - \mathbf{H}_t(\mathbf{\eta}(t) - \mathbf{\bar{\theta}}(t)))^T] = \mathbb{E}[\mathbf{n}\mathbf{n}^T] + \mathbf{H}_t \mathbb{E}[(\mathbf{\eta}(t) - \mathbf{\bar{\theta}}(t))(\mathbf{\eta}(t) - \mathbf{\bar{\theta}}(t))^T]\mathbf{H}_t^T = \mathbf{R} + \mathbf{H}_t \mathbb{P}(t, t) \mathbf{H}_t^T.
\]

(40)

These models can also be utilized to obtain the cross-covariance between the state and the measurements:

\[
\mathbb{E}[(\mathbf{\theta}(t) - \mathbb{E}[\mathbf{\theta}(t)])(\mathbf{z} - \mathbb{E}[\mathbf{z}])^T] = \mathbb{E}[(\mathbf{\bar{\theta}}(t) + \mathbf{\delta \theta}(t) - \mathbf{\eta}(t))(\mathbf{n} - \mathbf{H}_t(\mathbf{\eta}(t) - \mathbf{\bar{\theta}}(t)))^T] = \mathbb{E}[\mathbf{\bar{\theta}}(t) - \mathbf{\eta}(t))(\mathbf{\bar{\theta}}(t) - \mathbf{\eta}(t))^T\mathbf{H}_t^T = \mathbb{P}(t, t) \mathbf{H}_t^T.
\]

(41)

This Gaussian approximation is then used with the (21) and (33) definitions to express the joint distribution between the measurements and a single state perturbation as

\[
\begin{bmatrix}
\mathbf{z} \\
\mathbf{\delta \theta}(t)
\end{bmatrix} \sim \mathcal{N}
\begin{bmatrix}
\mathbf{h} + \mathbf{H}_t(\mathbf{\eta} - \mathbf{\bar{\eta}}) \\
\mathbf{\eta} - \mathbf{\bar{\theta}}(t)
\end{bmatrix},
\begin{bmatrix}
\mathbf{R} + \mathbf{H}_t \mathbb{P}(t, t) \mathbf{H}_t^T & \mathbb{P}(t, t) \mathbf{H}_t^T \\
\mathbb{P}(t, t) \mathbf{H}_t^T & \mathbb{P}(t, t)
\end{bmatrix},
\]

(42)

The optimal perturbation is then found by conditioning the measurements onto the state perturbation, which produces

\[
\mathbf{\delta \theta}^*(t) \sim \mathcal{N}((\mathbf{\eta}(t) - \mathbf{\bar{\theta}}(t)) + \mathbb{P}(t, t) \mathbf{H}_t^T (\mathbf{R} + \mathbf{H}_t \mathbb{P}(t, t) \mathbf{H}_t^{-1})^{-1} \times (\mathbf{z} - \mathbf{h} - \mathbf{H}_t(\mathbf{\eta} - \mathbf{\bar{\eta}})),
\]

\[
\mathbb{P}(t, t') - \mathbb{P}(t, t) \mathbf{H}_t^T (\mathbf{R} + \mathbf{H}_t \mathbb{P}(t, t) \mathbf{H}_t^{-1})^{-1} \mathbf{H}_t(t')^T.
\]

(43)

This is the same result as (31)–(32).

### 3.5. Linear prediction equation

After convergence, additional estimates for the continuous-time state component, \(\mathbf{x}(t)\), at other times of interest can be obtained using (31)–(32). We use the term prediction for this task, since it is similar to the conventional use of GPs for regression. However, a more efficient expression for the mean function can be obtained through additional algebraic manipulation.

Starting with (31),

\[
\mathbf{\mathbf{\delta \theta}^*(t) = (\mathbf{\eta}(t) - \mathbf{\bar{\theta}}(t) + \mathbb{P}(t) \mathbf{\delta \theta}(t)) \times (\mathbf{z} - \mathbf{h} - \mathbf{H}_t(\mathbf{\eta} - \mathbf{\bar{\eta}})),
\]

(44)

we insert \(\mathbb{P}^{-1}\) to get

\[
\mathbf{\delta \theta}^*(t) = (\mathbf{\eta}(t) - \mathbf{\bar{\theta}}(t) + \mathbb{P}(t) \mathbb{P}^{-1} \mathbf{H}_t^T (\mathbf{R} + \mathbf{H}_t \mathbb{P}(t, t) \mathbf{H}_t^{-1})^{-1} \times (\mathbf{z} - \mathbf{h} - \mathbf{H}_t(\mathbf{\eta} - \mathbf{\bar{\eta}})),
\]

(45)
and apply the SMW identity to result in
\[
\delta \theta^*(i) = \eta(i) - \tilde{\theta}(i) + \mathcal{P}(i) \mathcal{P}^{-1} (\mathcal{P}^{-1} + \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \times \mathbf{H}^T \mathbf{R}^{-1} (\mathbf{z} - \mathbf{h} - \mathbf{H}(\eta - \tilde{\theta})) .
\] (46)

Next, we add and subtract \( \mathcal{P}^{-1}(\eta - \tilde{\theta}) \), and distribute the terms to obtain
\[
\delta \theta^*(i) = \eta(i) - \tilde{\theta}(i) + \mathcal{P}(i) \mathcal{P}^{-1}(\mathcal{P}^{-1} + \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \times (\mathbf{H}^T \mathbf{R}^{-1} (\mathbf{z} - \mathbf{h}) - \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}(\eta - \tilde{\theta})) + \mathcal{P}^{-1}(\eta - \tilde{\theta}) - \mathcal{P}^{-1}(\eta - \tilde{\theta}) .
\] (47)

and simplify by grouping terms, which results in
\[
\delta \theta^*(i) = \eta(i) - \tilde{\theta}(i) + \mathcal{P}(i) \mathcal{P}^{-1}(-(-\mathcal{P}^{-1} + \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \times (\mathbf{P}^{-1} + \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})(\eta - \tilde{\theta})) + \mathcal{P}(i) \mathcal{P}^{-1}(\delta \theta^*) .
\] (48)

Since we are only interested in predicting values for the time-varying component of the state, we isolate and rearrange the expression to produce
\[
\mathbf{x}(i) + \delta \mathbf{x}^*(i) = \mu(i) - \mathcal{K}(i) \mathcal{K}^{-1} (\mu - \mathbf{x}) + \mathcal{K}(i) \mathcal{K}^{-1} \delta \mathbf{x}^* .
\] (49)

Upon convergence of Gauss–Newton, \( \delta \mathbf{x}^*(i) = \delta \mathbf{x}^* = 0 \), which results in
\[
\mathbf{x}(i) = \mu(i) - \mathcal{K}(i) \mathcal{K}^{-1} (\mu - \mathbf{x}) .
\] (50)

This is an expression for predicting the value of the continuous-time state at other times of interest, which is composed of a linear combination of the converged state estimates at the measurement times. Furthermore, this expression also resembles the linear prediction equations for GP regression (Rasmussen and Williams, 2006).

### 3.6. Measurement interpolation

The final step of the derivation involves utilizing the prediction equations to significantly reduce the computational resource requirements of GPGN. As noted earlier in Section 3.3.1, values for the state at each measurement time are required for linearization. This results in perturbations for the state to be estimated at all of the measurement times during each iteration, which can be prohibitive for large numbers of measurements. While the discrete-time formulation has the same requirements as this basic formulation of GPGN, we demonstrate another significant advantage of the continuous-time formulation by using interpolation to reduce the size of the estimated state.

The linear prediction equations from the previous section can be used to obtain the values of the state at some of the measurement times directly during the Gauss–Newton process. That is, if we subtract (50) from (49) and utilize the result during measurement linearization, we get
\[
\mathbf{z}_i \approx \mathbf{h}_i(\tilde{\mathbf{z}}(t_i)) + \mathbf{H}_i \delta \mathbf{\theta}(t_i) + \mathbf{n}_i = \mathbf{h}_i(\tilde{\mathbf{z}}(t_i)) + \mathbf{H}_i \left[ \begin{array}{c} \mathcal{K}(t_i) \mathcal{K}^{-1} \\ 0 \\ 1 \end{array} \right] \delta \mathbf{\theta} + \mathbf{n}_i ,
\] (51)

where we employ (50) to evaluate the Jacobians.

This expression is of a similar form as (20). As a result, we simply redefine \( \mathbf{H} \) in (21) accordingly, and utilize the same equations as derived before.

We interpret this approach as interpolation because we are utilizing some state estimates to produce values at other times of interest. The mapping from the estimated to the interpolated states is provided by \( \mathcal{K}(t_i) \mathcal{K}^{-1} \), which can be precomputed for each iteration. By estimating fewer states, we gain substantial computational savings. This method of approximation is similar to using fewer basis functions.
in the parametric approach, where we gain computational savings at the cost of representational fidelity.

3.7. Summary
In summary, batch state estimation using the GPGN algorithm is performed as follows:

1. Divide the measurement times into two sets: the times at which the state values will be estimated, and the times at which the state values will be interpolated.
2. Initialize with an initial guess for the state at the estimated times, \( \bar{\theta} \).
3. Interpolate the state values for the remaining measurement times using (50).
4. Linearize and construct the (21) and (33) matrices.
5. Solve for the optimal perturbations at the estimated times, \( \delta \theta^* \), using (36).
6. Apply the additive update to improve the state estimate, \( \bar{\theta} \leftarrow \bar{\theta} + \delta \theta^* \).
7. Repeat steps 3–6 until convergence.
8. After convergence, compute the time-varying state, \( \mathbf{x}(t) \), at other timesteps of interest using (50).

The tradeoffs between this non-parametric approach and an explicit basis function parametrization (Furgale et al., 2012) are related to the time-varying state representation. In the case of parametric estimation, the computational complexity is proportional to the number of basis function coefficients. Similarly, the computational complexity of GPGN is proportional to the number of estimated states. In both cases, limiting the state size for computational savings comes at the cost of representational fidelity.

4. Experimental validation
For experimental validation of the proposed algorithm, we leverage the robotics community’s prior experience with the well-understood problem of two-dimensional landmark-based SLAM. For comparison, GPGN was implemented alongside the conventional discrete-time batch Gauss–Newton approach (GN). Since GN utilizes all of the measurements in a discrete-time ML formulation, it encompasses the traditional single-timestep smoothing approaches. While the GN approach may be modified by conducting velocity estimation, pose interpolation, or other heuristics to obtain additional performance, we compare against the basic formulation for clarity. This comparison is conducted using a combination of simulation results and a hardware experiment.

The intent of this section is not to convince the reader that two-dimensional SLAM problems should be solved using GPGN; these are well-addressed by the conventional methods. Rather, our intent is to illustrate the advantages of using the proposed algorithm without obfuscation by complex experimental configurations, and to demonstrate that GPGN is able to address conventional robotic state estimation scenarios. While we demonstrate improved estimation accuracy with a small increase in computational cost, we remind the reader that the key advantage of the continuous-time formulation is the ability to handle measurement scarcity and asynchronicity. These issues cannot be handled by the discrete-time formulation in a computationally tractable manner.

We begin with a brief overview of the experimental scenario, and then provide some implementation details for the GPGN and GN algorithms. This is followed by the simulation results, and concluded with the hardware experiment.

4.1. Overview
In this scenario, we considered a mobile robot equipped with a laser rangefinder driving in a planar environment. Two measurement types were available: instantaneous linear and angular velocities through wheel odometry, and range and bearing measurements to point landmarks in the scene using the laser rangefinder. Known data association was provided, and the maximum observation range was artificially limited to 2 m. The odometry and landmark measurements were provided to the estimators at a rate of 1 Hz, and after estimator convergence, additional estimates were computed at a rate of 10 Hz to evaluate performance. To emulate the ability of GPGN to produce additional estimates between measurement times, linear interpolation was employed for the GN estimates.

This experimental scenario was carefully designed to allow for easy comparison between the two algorithms. Both odometry and landmark measurements were synchronized, and known data association was provided to allow for unit testing of the estimators themselves. Unless another process model is introduced (Davison et al., 2007), GN is unable to produce estimates using only the limited landmark measurements. We demonstrate the advantage of the continuous-time formulation by also considering the landmark-only scenario for the hardware experiment data.
4.2. GPGN implementation

For GPGN, we defined our time-varying state vector to be

\[
x(t) := \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix},
\]

the robot’s pose at time \( t \), and the time-invariant component to be the landmark locations, \( \ell \). To incorporate physical limitations on the robot platform, we modelled the time-varying state acceleration as a white noise process with

\[
\ddot{x}(t) \sim \mathcal{GP}(0, W \delta(t - t')),
\]

where \( W \) is the power spectral density matrix, and \( \delta() \) is the Dirac delta function. This simple physically motivated model provides a smooth trajectory, but does not address the motion constraints imposed by the hardware configuration. As shown in Appendix A, integrating the mean and covariance functions twice produced the GP for \( x(t) \), where

\[
\begin{align*}
\mu(t) &= x(0), \\
K(t, t') &= W \left( \frac{\min(t, t')^2 \max(t, t')}{2} - \frac{\min(t, t')^3}{6} \right).
\end{align*}
\]

Along with this process model, the instantaneous linear and angular velocity odometry measurements were modelled as

\[
v_k := \begin{bmatrix} \cos \theta(x_k) & \sin \theta(x_k) & 0 \\ -\sin \theta(x_k) & \cos \theta(x_k) & 0 \\ 0 & 0 & 1 \end{bmatrix} \dot{x}(t) + w_k,
\]

\[
w_k \sim \mathcal{N}(0, Q),
\]

where we allowed for a small amount of lateral wheel slip, and \( w_k \) was corruptive zero-mean Gaussian noise with a covariance of \( Q \). This minimal lateral wheel slip model implicitly accounts for the non-holonomic nature of the robot platform. The \( \dot{x}(t) \) values required for linearization were obtained using the linear prediction equation (50), and the odometry measurements were incorporated using measurement interpolation (51). The approach for predicting derivative state values and incorporating derivative-type measurements into GPGN is detailed in Appendix B.

Finally, the laser rangefinder measurements to landmark \( j \) were modelled as

\[
z_i := \begin{bmatrix} (x_j - x(t)) - d \cos \theta(t) \\ (y_j - y(t)) - d \sin \theta(t) \end{bmatrix}^{1/2} + n_{ij},
\]

\[
n_{ij} \sim \mathcal{N}(0, R),
\]

where \((x_j, y_j)\) was the position of landmark \( j \), \( d \) was the offset between the robot and sensor centres, and \( n_{ij} \) was the corruptive Gaussian noise with a covariance of \( R \). An illustrated definition of these state parameters is provided in Figure 3.

4.3. Discrete-time batch Gauss–Newton implementation

Similarly, for GN, we defined our state vector to be the robot pose at timestep \( k \),

\[
x_k := \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix},
\]

and the landmark locations as \( \ell \).

The discrete-time odometry measurement model was obtained by performing an explicit Euler step of the continuous-time model, which resulted in

\[
v_k := \frac{1}{T} \begin{bmatrix} \cos \theta_k & \sin \theta_k & 0 \\ -\sin \theta_k & \cos \theta_k & 0 \\ 0 & 0 & 1 \end{bmatrix} (x_{k+1} - x_k) + w_k,
\]

\[
w_k \sim \mathcal{N}(0, Q),
\]

where \( T \) was the sampling period. The laser measurement model remained the same as the GPGN case (57) with discrete poses, \( x_k \), in place of the \( x(t) \) values. The measurement data and noise parameters were kept identical for both algorithms.

4.4. Simulation results

To facilitate statistical analysis, we conducted 1,000 simulated trials of a two-dimensional SLAM problem. Figure 4 depicts the layout we considered, where the robot travelled two loops in a figure-eight pattern, and observed small clusters of landmarks throughout its traverse. With the 1 Hz
Table 1. Average RMS estimate errors and computation time for the 1,000 simulated trials.

<table>
<thead>
<tr>
<th></th>
<th>Translation error (m)</th>
<th>Orientation error (°)</th>
<th>Landmark position error (m)</th>
<th>Convergence time (s)</th>
<th>Additional estimate interpolation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GN</td>
<td>0.245</td>
<td>3.11</td>
<td>0.251</td>
<td>2.5</td>
<td>0.003</td>
</tr>
<tr>
<td>GPGN</td>
<td>0.213</td>
<td>2.66</td>
<td>0.225</td>
<td>7.3</td>
<td>15</td>
</tr>
</tbody>
</table>

measurement rate, each trial consisted of 300 robot poses and 100 landmarks obtained over a 5 minute time window.

The average root mean squared (RMS) estimate errors over the 1,000 trials were computed for quantitative comparison. We evaluate both mapping and localization performance by computing the errors under the maximally consistent alignment between the estimated and ground truth maps. These results are presented in Table 1. Compared with GN, GPGN achieved improvements in accuracy of 13.1% in translation, 14.5% in orientation, and 10.4% in landmark position.

However, these improvements in estimation accuracy also came with an increase in computation time. Although GPGN required three times the computation time for convergence, the total amount of time is still significantly less than the experiment time window of 5 minutes. Furthermore, the additional estimates would only be needed during post-processing, and can be computed in an offline manner.

As introduced in Section 3.6, we can also reduce the computational requirements by interpolating some of the robot poses, to shrink the size of the estimated state. To give a sense of the tradeoffs in accuracy and computation, we varied the interpolation percentage for one of the simulation trials. The results are depicted in Figure 5, where the RMS translation error and computation times for GPGN are plotted with respect to the percentage of interpolated poses. As can be seen, the number of estimated poses could be moderately reduced without significantly affecting the estimation accuracy. This is likely due to the fact that the figure-eight trajectory can be well-represented by a only subset of the robot poses.

Unfortunately, these computational tradeoffs are highly dependent on the estimation problem. The number of robot poses necessary to model the trajectory depends on the complexity of the traverse, and the computational gains tend to be limited to scenarios where there are a large number of robot poses compared with the number of landmark measurements. Since matrix interpolation introduces additional fill-in, the cost of dense matrix construction can result in increased computation time. For example, a 100 robot pose and 1,000 landmark trial provides significantly different computational tradeoffs than depicted in Figure 5. In this case, 5.6 s was required for GN convergence, 6.3 s for GPGN convergence, and 21 s for GPGN convergence with 50% interpolated poses. The similarity in computation time between GN and GPGN can be attributed to the use of

Fig. 5. Plots depicting the tradeoffs in GPGN estimation accuracy and computation time with respect to the percentage of interpolated poses for a single simulation trial consisting of 300 robot poses and 100 landmarks. In (a), we plot the RMS translation error, and in (b) we show the computation time for convergence and obtaining the additional pose estimates. For comparison, the GN values are also depicted by the horizontal lines. As can be seen, the number of estimated poses could be reduced without significant detriment to the accuracy. In other words, only a subset of the poses were required to represent the robot trajectory.
SBA, which results in the computational complexity growing proportionally to the number of landmarks when there are many more landmarks than robot poses.

4.5. Hardware experiment

The experimental setup consisted of a mobile robot driving in an indoor, planar environment, amongst a forest of plastic tubes, depicted in Figure 6(a). These tubes served as landmarks for our localization problem, and ground truth data was obtained using a Vicon motion capture system that tracked retroreflective markers placed on both the robot and the plastic tubes. Owing to the limited size of the workspace, only 17 landmarks were available in this experiment. As depicted in Figure 6(b), the robot traverse over 5 minutes was sufficiently varied for our evaluation purposes.

To highlight the differences between the two approaches, we begin by focusing on a small section of the traverse. In this section, the robot travelled from left to right, with small corrective turns along the way. The results produced by the two estimators, depicted in Figures 7(a)–7(b), were quite different. The GN estimate in Figure 7(a) contained sharp jumps, which reflects the discrete-time formulation. (b) The estimate for a short section of the traverse produced by GPGN. Owing to the underlying process model, the estimated trajectory and its associated covariance envelope was smooth. (c) The estimate for the whole 5 minutes produced by discrete-time batch Gauss–Newton. Although the performance was quite accurate for the majority of the traverse, sharp jumps in the estimates were still present, resulting in reduced performance at the edges. (d) The estimate for the whole 5 minutes produced by GPGN. The smooth estimated trajectory and its associated covariance envelope accurately matched the ground truth path.
of the traverse, as well as the overall trajectory. In Figure 9(a), we see a much larger lateral covariance envelope, which can be attributed to the reduced number of measurements, and the omission of the minimal lateral wheel slip constraint provided by the odometry measurements. The overall traverse depicted in Figure 9(b) is similar to the plots in Figures 7(c)–7(d), except for the larger covariance envelopes in the regions of measurement scarcity. Since an insufficient amount of information is available at these timesteps, the uncertainty is appropriately higher in these regions. Finally, the quantitative results presented in Table 2 demonstrate that a similar level of accuracy is achieved even without the odometry measurements. The slight improvement in estimator accuracy may be attributed to an inaccurate wheel odometry model, and the large increase in estimation time was due to the omission of the minimal lateral wheel slip constraint. This resulted in a larger search space during optimization, which required many more iterations for convergence.

5. Conclusion

In conclusion, we have introduced GPGN, a novel continuous-time algorithm for non-parametric, nonlinear, batch state estimation. This work adapts concepts from GP regression, and applies it to the robotics domain. GPGN is significantly different from the previous uses of GPs in robotics. In particular, we do not use GPs for output modelling; rather, we use GPs to model the underlying state. As a result, GPGN is able to address conventional robotic state estimation scenarios.

To avoid overwhelming the reader with too many new concepts, we deliberately focused on the theoretical contribution and avoided presenting a complex experimental scenario for validation of GPGN. Instead, we utilized a well-understood state estimation problem to illustrate the tradeoffs of GPGN compared with the conventional discrete-time batch Gauss–Newton approach. These advantages include the production of smooth mean and covariance estimates that better reflect the physical reality of the scenario, and the ability to fall back on the underlying continuous-time process model in cases of measurement scarcity. This was demonstrated by the ability of GPGN to address the landmark-only SLAM scenario. While the improvements in modelling accuracy come with increased computational cost, our experimental results demonstrate that the increase is reasonable for practical problems.

In our experiments, we employed a simple physically motivated GP based on a white noise prior on the state acceleration. While we demonstrated improvements in estimation accuracy, this choice of GP did not capture the limitations of the robot platform, and may not scale easily due to its dense nature. However, our derivation of GPGN does not make any assumptions on the form of the GP. As a result, we are able to utilize any GP in this algorithm, whether dense, sparse, stationary, or heteroscedastic. Furthermore, a large

![Figure 8](image-url)

**Fig. 8.** The number of landmark measurements visible at each robot pose. The green dots indicate poses that observe two or more landmarks, and the red dots indicate poses that do not. Since a minimum of two landmark measurements are required to produce a 2D pose estimate, the discrete-time batch Gauss–Newton formulation is unable to address the landmark-only scenario.
Table 2. RMS estimate errors and computation time for the hardware experiment.

<table>
<thead>
<tr>
<th></th>
<th>Translation error (m)</th>
<th>Orientation error (°)</th>
<th>Landmark position error (m)</th>
<th>Convergence time (s)</th>
<th>Additional estimate interpolation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GN</td>
<td>0.114</td>
<td>4.38</td>
<td>0.177</td>
<td>1.6</td>
<td>0.002</td>
</tr>
<tr>
<td>GPGN</td>
<td>0.087</td>
<td>3.35</td>
<td>0.113</td>
<td>7.4</td>
<td>15</td>
</tr>
<tr>
<td>Landmark-only GPGN</td>
<td>0.077</td>
<td>3.01</td>
<td>0.071</td>
<td>31</td>
<td>15</td>
</tr>
</tbody>
</table>

Fig. 9. Plots depicting the results of landmark-only GPGN estimator. The ground truth trajectory and landmark locations are indicated by the black line and green diamonds, respectively. The estimated robot positions and landmark estimates are indicated in red, and the associated $3\sigma$ covariance envelopes are represented by the blue shading. The lateral covariance envelopes were determined by projecting the estimated $(x, y)$ covariances onto the lateral dimension defined by the mean orientation estimate. (a) The estimate for the short section of the robot traverse depicted in Figures 7(a)–7(b). The smooth growth of the covariance envelope reflects the measurement scarcity, and the omission of the minimal lateral wheel slip model imposed by the odometry measurements. (b) The estimate for the overall traverse. In addition to being more accurate than the plots in Figures 7(c)–7(d), it can be seen that the covariance envelope expands smoothly and accurately in the regions with measurement scarcity, as depicted in Figure 8.

amount of literature is also available on scaling GPs to large systems, including approximation using sparse methods, and compactly supported covariance functions (Rasmussen and Williams, 2006). Online operation may be addressed by utilizing a sliding window formulation (Sibley et al., 2010), or by extending the incremental smoothing and mapping (iSAM) (Kaess et al., 2008, 2012) factorizations to GPGN, which will provide the full batch solution at a fraction of the computational cost. These factorizations have been utilized successfully for online GP regression (Ranganathan et al., 2011), and we expect a similar benefit for continuous-time state estimation. Many of the variations on the basic framework of GP regression should be transferable to the robotics domain.

Possible directions for future work include comparisons with a parametric spline approach, analysis of different GP state models, and investigating methods for adaptive pose placement to minimize the number of estimated poses. In addition, more experimentation can be conducted involving larger datasets, and the presence of inertial measurements. The theoretical contributions of this paper have been applied towards 3D laser-based visual odometry by Tong and Barfoot (2013), where a sparse appearance-based approach was taken, and motion compensation was addressed using a continuous-time state estimation formulation using GPGN. This application is similar to the landmark-only scenario presented in this paper, since the sweeping nature of a 3D laser sensor results in only one landmark observation at any given time.

Notes
1. This is a different odometry model than that used by Tong et al. (2012), as it contains the additional lateral wheel slip term. This resulted in significantly different estimates and convergence rates.
2. Timing information recorded on a MacBook Pro with a 2.66 GHz Core 2 Duo and 4 GB of 1067 MHz DDR3 RAM in Matlab, utilizing both cores.

Funding
This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References


Appendix A: Mean and covariance functions for a moving point

In GPGN, the mean and covariance functions define the properties of the underlying process model that we wish to estimate. As a result, appropriate GP selection can greatly affect the performance of the algorithm. This appendix details the derivation of the physically motivated GP utilized in this paper.
For clarity, we define
\[ a(t) := \ddot{x}(t), \quad (60) \]
\[ v(t) := \dot{x}(t), \quad (61) \]
\[ d(t) := x(t), \quad (62) \]
and for simplicity, we will assume that the integration and initial conditions begin at \( t = 0 \).

A.1. General derivation

We begin by showing how to obtain the stochastic process properties of the velocity and position for a moving point, starting with a GP model on the acceleration. This is followed by a specific example of the white noise prior in the following section.

A.1.1. Acceleration

We define the GP prior on acceleration as
\[ a(t) \sim \mathcal{GP} \left( \mu_a(t), \mathcal{K}_{aa}(t, t') \right), \quad (63) \]
\[ \mu_a(t) := E[a(t)], \quad (64) \]
\[ \mathcal{K}_{aa}(t, t') := E \left[ (a(t) - \mu_a(t)) (a(t') - \mu_a(t'))^T \right]. \quad (65) \]

A.1.2. Velocity

The relation between velocity and acceleration is
\[ \dot{v}(t) = \int_0^t a(\tau) \, d\tau + v(0), \quad (66) \]
where \( v(0) \) is provided as an initial condition for the velocity. Using this expression, the mean function is
\[ \mu_v(t) = E \left[ \int_0^t a(\tau) \, d\tau + v(0) \right] \quad (67) \]
\[ = \int_0^t \mu_a(\tau) \, d\tau + \mu_v(0), \quad (68) \]
and the covariance function is
\[ \mathcal{K}_{vv}(t, t') = E[(\dot{v}(t) - \mu_v(t))(\dot{v}(t') - \mu_v(t'))^T] \]
\[ = \int_0^t \int_0^t E[(a(\tau) - \mu_a(\tau))(a(\tau') - \mu_a(\tau'))^T] \, d\tau \, d\tau' \]
\[ + \int_0^t E[(a(\tau) - \mu_a(\tau))(v(0) - \mu_v(0))^T] \, d\tau \]
\[ + \int_0^t E[(v(0) - \mu_v(0))(a(\tau') - \mu_a(\tau'))^T] \, d\tau' \]
\[ + E[(v(0) - \mu_v(0))(v(0) - \mu_v(0))^T]. \quad (69) \]
We simplify this expression by assuming the acceleration and initial velocity terms are independent, resulting in
\[ \mathcal{K}_{vv}(t, t') = \int_0^t \int_0^t \mathcal{K}_{aa}(\tau, \tau') \, d\tau \, d\tau' + \mathcal{K}_{vv}(0, 0), \quad (70) \]
where \( \mathcal{K}_{aa}(0, 0) \) is typically provided as an initial uncertainty for the estimation problem. In our experiments, we set this initial value to zero.

A.1.3. Position

Similarly, the relation between position and acceleration is
\[ d(t) = \int_0^t \dot{v}(\tau) \, d\tau + d(0) \]
\[ = \int_0^t \int_0^t a(s) \, ds \, d\tau + \int_0^t \dot{v}(0) \, d\tau + d(0), \quad (71) \]

Using this relation, the mean function is
\[ \mu_d(t) = E \left[ \int_0^t \int_0^t a(s) \, ds \, d\tau + \int_0^t \dot{v}(0) \, d\tau + d(0) \right] \]
\[ = \int_0^t \int_0^t \mu_a(s) \, ds \, d\tau + \mu_v(0) t + \mu_d(0), \quad (72) \]
and the covariance function is
\[ \mathcal{K}_{dd}(t, t') = E[(d(t) - \mu_d(t))(d(t') - \mu_d(t'))^T] \]
\[ = \int_0^t \int_0^t \int_0^t \int_0^t E[(a(s) - \mu_a(s))(a(s') - \mu_a(s'))^T] \, ds \, ds' \, d\tau \, d\tau' \]
\[ + \int_0^t \int_0^t \int_0^t E[(a(s) - \mu_a(s))(v(0) - \mu_v(0))^T] \, ds \, d\tau \, d\tau' \]
\[ + \int_0^t \int_0^t \int_0^t E[(v(0) - \mu_v(0))(a(s') - \mu_a(s'))^T] \, ds' \, d\tau \, d\tau' \]
\[ + \int_0^t \int_0^t \int_0^t E[(v(0) - \mu_v(0))(v(0) - \mu_v(0))^T] \, d\tau \, d\tau' \]
\[ = \int_0^t \int_0^t \mathcal{K}_{aa}(\tau, \tau') \, d\tau \, d\tau' + \mathcal{K}_{dd}(0, 0), \quad (73) \]
Once again, we assume the acceleration and initial position and velocity terms are independent, which results in

\[
\mathcal{K}_{dd}(t, t') = \int_0^t \int_0^t \int_0^t \mathcal{K}_{aa}(s, s') \, ds \, ds' \, d\tau' \, d\tau \\
+ \int_0^t \int_0^t \mathcal{K}_{va}(0, 0) \, d\tau \, d\tau' \\
+ \int_0^t \mathcal{K}_{ad}(0, 0) \, d\tau + \int_0^t \mathcal{K}_{dv}(0, 0) \, d\tau' \\
+ \mathcal{K}_{dd}(0, 0),
\]

(74)

where \( \mathcal{K}_{aa}(0, 0) \) and \( \mathcal{K}_{dv}(0, 0) \) are the position–velocity cross-covariance initial uncertainties, and \( \mathcal{K}_{va}(0, 0) \) and \( \mathcal{K}_{dd}(0, 0) \) are the initial velocity and position uncertainties, respectively. These initial values were also set to zero in our experiments.

For completeness, we obtain the cross-covariance functions in a similar manner:

\[
\mathcal{K}_{vd}(t, t') = \int_0^t \int_0^t \int_0^t \mathcal{K}_{ad}(s, s') \, ds \, ds' \, d\tau' \, d\tau \\
+ \mathcal{K}_{va}(0, 0) t' + \mathcal{K}_{vd}(0, 0),
\]

(75)

\[
\mathcal{K}_{dv}(t, t') = \int_0^t \int_0^t \int_0^t \mathcal{K}_{aa}(s, \tau') \, ds \, d\tau' \, d\tau \\
+ \mathcal{K}_{va}(0, 0) t + \mathcal{K}_{dv}(0, 0).
\]

(76)

### A.2. Specific example of a white noise prior on \( \ddot{x}(t) \)

In this section, we develop the specific GP used in this thesis by starting with a white noise prior on the acceleration. That is,

\[
\ddot{x}(t) \sim \mathcal{G}P(0, W \delta(t - t')), \tag{77}
\]

where \( W \) is a power spectral density matrix, and \( \delta(\cdot) \) is the Dirac delta function. If \( v(0) = 0 \) and \( d(0) = 0 \), it is straightforward to show that the mean functions for position, velocity, and acceleration are all zero. As a result, only the covariance functions are derived in the following sections.

#### A.2.1. Acceleration

In the case of acceleration, the covariance function is simply as stated above:

\[
\mathcal{K}_{aa}(t, t') = W \delta(t - t').
\]

(78)

#### A.2.2. Velocity

Next, we integrate the acceleration covariance function to obtain

\[
\int_0^t \int_0^t \mathcal{K}_{aa}(\tau, \tau') \, d\tau \, d\tau' = W \int_0^t \int_0^t \delta(\tau - \tau') \, d\tau \, d\tau'
\]

\[
= W \int_0^t (H(t - \tau') - H(0 - \tau')) \, d\tau'
\]

\[
= W \int_0^t H(t - \tau') \, d\tau', \quad (79)
\]

where \( H(t) \) is the Heaviside step function. We now consider the two cases. For \( t \geq t' \),

\[
W \int_0^t H(t - \tau') \, d\tau' = W \int_0^t d\tau' = Wt', \tag{80}
\]

and for \( t < t' \),

\[
W \int_0^t H(t - \tau') \, d\tau' = W \int_0^t d\tau' = Wt. \tag{81}
\]

Inserting this result into (70) provides the velocity covariance function, which can be summarized by

\[
\mathcal{K}_{va}(t, t') = W \min(t, t') + \mathcal{K}_{va}(0, 0). \tag{82}
\]

#### A.2.3. Position

To obtain the position covariance function, we integrate the expression from the previous section two more times:

\[
\int_0^t \int_0^t \int_0^t \mathcal{K}_{aa}(s, \tau') \, ds \, d\tau' \, d\tau
\]

\[
= \int_0^t \int_0^t W \min(\tau, \tau') \, d\tau \, d\tau'. \tag{83}
\]

We approach this integral in a similar manner by considering the two cases separately. For \( t \geq t' \), we obtain

\[
\int_0^t \int_0^t W \min(\tau, \tau') \, d\tau \, d\tau' \\
= W \int_0^t \left( \int_0^\tau \min(\tau, \tau') \, d\tau + \int_\tau^t \min(\tau, \tau') \, d\tau' \right) \, d\tau' \\
= W \int_0^t \left( \int_0^\tau \tau' \, d\tau + \int_\tau^t \tau' \, d\tau' \right) \, d\tau' \\
= W \int_0^t \left( \frac{\tau'^2}{2} + \tau'(t - \tau') \right) \, d\tau' \\
= W \int_0^t \left( \frac{t'^2}{2} \right) \, d\tau' \\
= W \left( \frac{t'^2}{2} - \frac{t'^3}{6} \right), \tag{84}
\]
and for \( t < t' \),
\[
\int_0^t \int_0^{t'} W \min(\tau, \tau') \, d\tau \, d\tau' = W \left( \int_0^{t'} \min(\tau, \tau') \, d\tau \, d\tau' + \int_0^t \min(\tau, \tau') \, d\tau \, d\tau' \right)
\]
\[
= W \left( \frac{t^3}{2} - \frac{t^3}{6} + \int_0^{t'} \tau \, d\tau \, d\tau' \right)
\]
\[
= W \left( \frac{t^3}{2} + \frac{(t' - t)^2}{2} \right)
\]
\[
= W \left( \frac{t^3}{2} - \frac{t'^3}{6} \right). \tag{85}
\]

This expression can then be inserted into (74) and simplified. As a result, the position covariance function can be expressed as
\[
\mathcal{K}_{dd}(t, t') = W \left( \frac{\min(t, t')^2 \max(t, t')}{2} - \frac{\min(t, t')^3}{6} \right)
\]
\[
+ \mathcal{K}_{vv}(0, 0) t' + \mathcal{K}_{vd}(0, 0) t
\]
\[
+ \mathcal{K}_{dh}(0, 0) t' + \mathcal{K}_{dh}(0, 0). \tag{86}
\]

A similar result is reported by Rasmussen and Williams (2006).

### A.2.4. Summary

In summary, the covariance functions for white noise on acceleration case are
\[
\mathcal{K}_{aa}(t, t') = W \delta(t - t'), \tag{87}
\]
\[
\mathcal{K}_{v0}(t, t') = W \min(t, t'), \tag{88}
\]
\[
\mathcal{K}_{dd}(t, t') = W \left( \frac{\min(t, t')^2 \max(t, t')}{2} - \frac{\min(t, t')^3}{6} \right)
\]
\[
+ \mathcal{K}_{vv}(0, 0) t' + \mathcal{K}_{vd}(0, 0) t
\]
\[
+ \mathcal{K}_{dh}(0, 0) t' + \mathcal{K}_{dh}(0, 0), \tag{89}
\]
and the cross-covariance functions (with the derivations left to the reader) are
\[
\mathcal{K}_{vd}(t, t') = f(t, t') + \mathcal{K}_{vv}(0, 0) t' + \mathcal{K}_{vd}(0, 0), \tag{90}
\]
\[
\mathcal{K}_{dh}(t, t') = f(t, t') + \mathcal{K}_{vv}(0, 0) t + \mathcal{K}_{dh}(0, 0), \tag{91}
\]
\[
f(\tau, \tau') = W \left\{ \begin{array}{ll}
\tau \tau' - \frac{\tau^3}{2} & , \tau < \tau', \\
\frac{\tau^2}{2} & , \tau \geq \tau'.
\end{array} \right. \tag{92}
\]

## Appendix B: Incorporating derivative-type measurements

In this appendix, we detail how to handle derivative-type measurements in GPGN. This is utilized during the experimental validation in Section 4 to incorporate instantaneous linear and angular velocity measurements obtained from wheel odometry. Derivative-type measurements are incorporated in a simple manner using the weight-space approach. Measurements of the form
\[
v_k = g_k(\dot{\theta}(t)) + w_k, \quad w_k \sim \mathcal{N}(0, Q_k), \tag{93}
\]
are linearized by first approximating the state derivative by the current guess, \( \ddot{\theta}(t) \), and an additive perturbation, \( \delta \dot{\theta}(t) \). By applying the basis function representation \( \ddot{\theta}(t) = \Psi(t) \ddot{b} \), we get
\[
\dot{\theta}(t) \approx \ddot{\theta}(t) + \delta \dot{\theta}(t) = \Psi(t) \ddot{b} + \Psi(t) \delta \dot{b}, \tag{94}
\]
which results in
\[
v_k \approx g_k(\ddot{\theta}(t)) + G_k \Psi(t) \delta \dot{b} + w_k, \quad G_k := \frac{\partial g_k}{\partial \dot{\theta}}(\ddot{\theta}(t)). \tag{95}
\]

Given a total of \( K \) derivative measurements, we stack them along with the other measurements by redefining the (21) matrices to be
\[
\begin{bmatrix}
z_1 \\
\vdots \\
v_1 \\
\vdots \\
v_K \\
\end{bmatrix},
\begin{bmatrix}
h_1(\ddot{\theta}(t_1)) \\
\vdots \\
g_1(\ddot{\theta}(t_1)) \\
\vdots \\
g_K(\ddot{\theta}(t_K))
\end{bmatrix}, \tag{96}
\]
\[
\begin{bmatrix}
\Phi_0 \\
\vdots \\
\Phi(t_1) \\
\vdots \\
\Phi(t_K)
\end{bmatrix},
\begin{bmatrix}
R_1 \\
\vdots \\
R_N \\
Q_1 \\
\vdots \\
Q_K \\
\end{bmatrix},
\begin{bmatrix}
H_1 \\
\vdots \\
H_N \\
G_1 \\
\vdots \\
G_K
\end{bmatrix} \tag{97}
\]

and the derivation continues as before.
In the final step of the weight-space derivation, the $\Phi K \Phi^T$ terms are replaced by evaluating the covariance function in (33). A similar substitution occurs when incorporating derivative measurements, but the basis function derivatives, $\dot{\Phi}(t)$, are now involved. The GPGN algorithm can be recovered by redefining the $K$ matrices appropriately using the substitutions

$$\Phi(t) K \Phi(t')^T = K(t, t'),$$

$$\dot{\Phi}(t) K \Phi(t')^T = \frac{\partial K(t, t')}{\partial t},$$

$$\Phi(t) K \dot{\Phi}(t')^T = \frac{\partial K(t, t')}{\partial t'},$$

$$\dot{\Phi}(t) K \dot{\Phi}(t')^T = \frac{\partial^2 K(t, t')}{\partial t \partial t'}. \quad (98a)$$

These expressions can also be utilized in the linear prediction equations (50)–(51), by replacing $K(t)$ appropriately to obtain derivative state values. Instantaneous acceleration measurements and state predictions can be incorporated in a similar manner.

Furthermore, these derivative expressions are related to the covariance functions developed in Appendix A by

$$K_{dd}(t, t') = K(t, t'), \quad (99a)$$

$$K_{vd}(t, t') = \frac{\partial K(t, t')}{\partial t}, \quad (99b)$$

$$K_{dv}(t, t') = \frac{\partial K(t, t')}{\partial t'}, \quad (99c)$$

$$K_{vv}(t, t') = \frac{\partial^2 K(t, t')}{\partial t \partial t'}. \quad (99d)$$

These expressions can also be utilized in the linear prediction equations (50)–(51), by replacing $K(t)$ appropriately to obtain derivative state values. Instantaneous acceleration measurements and state predictions can be incorporated in a similar manner.