2012 Ninth Conference on Computer and Robot Vision

Gaussian Process Gauss-Newton: Non-Parametric State Estimation

Chi Hay Tong
Autonomous Space Robotics Lab
University of Toronto
Toronto, Canada
chihay.tong@utoronto.ca

Paul Furgale†
Autonomous Systems Lab
ETH Zürich
Zürich, Switzerland
paul.furgale@mavt.ethz.ch

Timothy D. Barfoot
Autonomous Space Robotics Lab
University of Toronto
Toronto, Canada
tim.barfoot@utoronto.ca

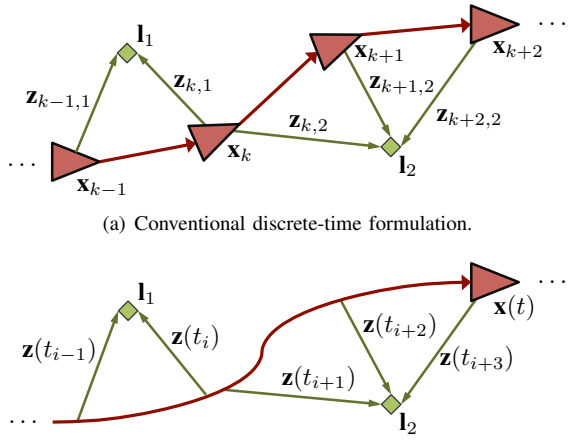
***Abstract*—In this paper, we present Gaussian Process Gauss-Newton (GPGN), an algorithm for non-parametric, continuous-time, nonlinear, batch state estimation. This work adapts the methods of Gaussian Process regression to the problem of batch state estimation by using the Gauss-Newton method. In particular, we formulate the estimation problem with a continuous-time state model, along with the more conventional discrete-time measurements. Our derivation utilizes a basis function approach, but through algebraic manipulations, returns to a non-parametric form by replacing the basis functions with covariance functions (i.e., the kernel trick). The algorithm is validated through hardware-based experiments utilizing the well-understood problem of 2D rover localization using a known map as an illustrative example, and is compared to the traditional discrete-time batch Gauss-Newton approach.**

I. 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.

Though the methods presented in this paper generalize to other state estimation problems, throughout this paper, we utilize the problem of rover localization using a known map as our illustrative example. This problem is well-understood, and provides opportunities to draw parallels to existing techniques. In this problem, a rover is travelling continuously, while obtaining measurements from a nonlinear, non-invertible sensor model at discrete times. We define the state in this scenario to be the rover poses, and consider the estimation problem over a window of time, where numerous measurements have been obtained. Therefore, the goal of the state estimation problem is to estimate the rover poses at some timesteps of interest given the batch of measurements.

This common scenario in mobile robotics is typically addressed by discretizing the rover trajectory, and computing an estimate of the rover pose at each measurement instant. This is illustrated in Figure 1(a), where the rover 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 rover motion is continuous, with characteristics defined by the physical and temporal constraints of the mobile platform. Therefore, we propose to model the estimated rover trajectory in continuous time, as depicted in Figure 1(b).



(a) Conventional discrete-time formulation.

(b) Proposed continuous-time formulation.

Figure 1. Illustrations comparing the two state estimation formulations, with the rover poses depicted by the red triangles, landmarks by green diamonds, and green arrows for measurements. In (a), we depict the conventional discrete-time formulation, which discretizes and enumerates the rover poses in sequence, and discards the measurement timestamps. In (b), we depict the more realistic continuous-time formulation, where the rover pose is modelled as a smooth function of time, and the discrete measurements are stored along with their timestamps.

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 a non-invertible measurement model imposes the requirement of multiple sensor readings to produce an estimate for every rover pose. This requirement can be problematic for high-rate sensors such as inertial measurement units or scanning laser systems, and 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 rover trajectory (using some other sensors for motion estimation). These issues of measurement scarcity are addressed by

†Work carried out while in the Autonomous Space Robotics Lab, University of Toronto Institute for Aerospace Studies.

978-0-7695-4683-4/12 $26.00 © 2012 IEEE
DOI 10.1109/CRV.2012.35

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) [1] has also been formulated in continuous-time [2], [3]. Unfortunately, these filtering methods require numerical integration for practical implementation, and can produce jagged estimates due to the Markov assumption. Similar implementation issues arise for forward/backward smoothing methods [4]. A commonly-used discrete-time alternative to filtering considers the measurements as a batch, which provides smoother results because all of the states are estimated together. We wish to take this approach, but formulate it 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. 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 our example, the latent states are the rover poses. 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. With this assumed model, the regression problem reduces to finding the optimal parameter values that produce the best matches to the sample outputs. The 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.

The discrete-time batch state estimation formulation shares some similarities with the parametric regression approach, since the state is parametrized by discrete rover poses at instants of time (Figure 1(a)). Estimation 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 [5], [6]. However, the performance will also suffer from the same issues of modelling accuracy.

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.

Gaussian Process (GP) regression is a non-parametric method where the underlying function is modelled by a mean and covariance function [7]. 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 rover state directly as a GP with the input of time, and utilize the Gauss-Newton algorithm [8] 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 is significantly different 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 drawn where possible, and the advantages are highlighted through experimental illustration. GPGN should be considered as an alternative to the parametric approach of [6] for solving the continuous-time batch state estimation problem.

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 II. This is followed by a brief mathematical overview of GPs in Section III, and a derivation of the proposed GPGN algorithm in Section IV. The algorithm is validated through experimentation in Section V, and concluding remarks are made in Section VI.

II. 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 [9], map gas distributions [10], and perform large-scale terrain modelling [11].These applications utilized GPs in their original formulation, since GPs were employed for modelling sensor outputs.

GP measurement models have also been utilized for state estimation via particle filtering [12], and recently, generalized into a discrete-time filtering framework with GP-BayesFilters (GPBFs) [13]. 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 with a GP, and utilize a batch estimation framework.

In recent literature, the concept of GP Latent Variable Models (GPLVMs) [14] has also been applied to the robotics domain. GPLVMs were originally developed as a method for

dimensionality reduction, which mapped high-dimensional GP measurements to a lower-dimensional GP latent state. This concept was applied to problems including Wifi-based Simultaneous Localization and Mapping (WifiSLAM) [15] and human motion tracking [16]. In addition, GPLVMs were used to train GPBFs without ground truth measurements [17]. 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. Furthermore, the lower-dimensional representation produced does not guarantee any relevance to the scenario at hand. To address this issue, additional weak labels are required to guide the process to produce physically meaningful results for the latent states, such as rover poses.

Though 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 warped GPs [18] and derivative GP observations [19], which share some similarity to how we handle the discrete-time measurements. A similar problem was considered in [20], 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 which were parametrized by time-invariant coefficients. Our work utilizes the conventional Gauss-Newton method [8], and is presented in a format more familiar to the robotics community.

III. Background

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

$$\mathbf{x} \sim \mathcal{N}(\mathbf{m}, \mathbf{K}), \tag{1}$$

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 [7]. Instead of a mean vector and a covariance matrix, a GP, $\mathbf{x}(t)$, is described by a mean *function*, $\boldsymbol{\mu}(t)$, and a covariance *function*, $\boldsymbol{\mathcal{K}}(t, t')$:

$$\mathbf{x}(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \boldsymbol{\mathcal{K}}(t, t')). \tag{2}$$

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.

IV. Gaussian Process Gauss-Newton

In this section, we provide the formulation of GPGN, which addresses the batch nonlinear state estimation problem. That is, we are tasked with determining the value of the state, $\mathbf{x}(t)$, given a set of N measurements obtained over a period of time.

We proceed by first defining the system models for the state estimation problem, and follow with a parametric basis function representation of the state. This approach is used as an intermediate step for derivation, and 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.

A. Problem Statement

We begin by defining the form of the state space models for the underlying system. The system is modelled as

$$\mathbf{x}(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \boldsymbol{\mathcal{K}}(t, t')), \tag{3}$$

$$\mathbf{z}_i = \mathbf{h}_i(\mathbf{x}(t_i)) + \mathbf{n}_i, \;\; \mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i), \tag{4}$$

where $\mathbf{x}(t)$ is a GP with mean and covariance functions $\boldsymbol{\mu}(t)$ and $\boldsymbol{\mathcal{K}}(t, t')$, respectively, and the measurements, $\mathbf{z}_i$, are obtained through a conventional nonlinear, non-invertible measurement model, $\mathbf{h}_i(\cdot)$, at N discrete times, t_i. For simplicity, we have modelled the measurement noise, $\mathbf{n}_i$, as additive, zero-mean, and Gaussian with covariance $\mathbf{R}_i$. While it may appear that we have assumed the state to be noise-free in (3), the use of a GP implicitly encompasses process noise.

B. Basis Function Derivation

In discrete-time batch estimation, the standard approach utilizes a sum-of-squares objective function of the form $J := \frac{1}{2}\sum_{i=1}^{N} \mathbf{e}_i^T \mathbf{W}_i \mathbf{e}_i$, where $\mathbf{e}_i$ is a measurement error term, and $\mathbf{W}_i$ is its associated weight. If the weight is chosen to be the inverse covariance matrix of $\mathbf{e}_i$, it can be shown that minimizing 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

$$\mathbf{x}(t) := \boldsymbol{\Phi}(t)\, \mathbf{c}, \tag{5}$$

where $\boldsymbol{\Phi}(t)$ is a stack of M known temporal basis functions,

$$\boldsymbol{\Phi}(t) := \begin{bmatrix} \boldsymbol{\phi}_1(t) & \dots & \boldsymbol{\phi}_M(t) \end{bmatrix}, \tag{6}$$

and $\mathbf{c}$ is a column of coefficients. Though this representation would convert this problem into a parametric estimation problem [6], 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 $\boldsymbol{\Phi}(t)$ or $\mathbf{c}$. The non-parametric form that we will obtain avoids this issue.

With this representation, the measurement model becomes

$$\mathbf{z}_i = \mathbf{h}_i(\mathbf{\Phi}(t_i)\,\mathbf{c}) + \mathbf{n}_i, \tag{7}$$

and similarly, the mean and covariance functions become

$$\boldsymbol{\mu}(t) \;=:\; \mathbf{\Phi}(t)\,\mathbf{m}, \tag{8}$$

$$\boldsymbol{\mathcal{K}}(t,t') \;=:\; \mathbf{\Phi}(t)\,\mathbf{K}\mathbf{\Phi}(t')^T, \tag{9}$$

where $\mathbf{m}$ and $\mathbf{K}$ are defined by

$$\mathbf{c} \sim \mathcal{N}(\mathbf{m}, \mathbf{K}), \tag{10}$$

returning us to the familiar Gaussian random variable domain. Equation (9), also known as the *kernel trick* in the machine learning literature [7], will be utilized in Section IV-C 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:

$$J := \frac{1}{2}\sum_{i=1}^{N} \left(\mathbf{z}_i - \mathbf{h}_i(\mathbf{\Phi}(t_i)\,\mathbf{c})\right)^T \mathbf{R}_i^{-1} \left(\mathbf{z}_i - \mathbf{h}_i(\mathbf{\Phi}(t_i)\,\mathbf{c})\right) + \frac{1}{2}\left(\mathbf{c}-\mathbf{m}\right)^T \mathbf{K}^{-1}\left(\mathbf{c}-\mathbf{m}\right). \tag{11}$$

To find the minimum of this objective function, we take the Gauss-Newton approach [8] 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, $\bar{\mathbf{x}}(t)$, and an additive perturbation, $\delta\mathbf{x}(t)$. Applying the basis function representation (5), we get

$$\begin{aligned} \mathbf{x}(t) &\approx \bar{\mathbf{x}}(t) + \delta\mathbf{x}(t) \\ &= \mathbf{\Phi}(t)\,\bar{\mathbf{c}} + \mathbf{\Phi}(t)\,\delta\mathbf{c}, \end{aligned} \tag{12}$$

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

$$\bar{\mathbf{c}} + \delta\mathbf{c} \;\sim\; \mathcal{N}\left(\mathbf{m}, \mathbf{K}\right), \tag{13}$$

$$\mathbf{z}_i \;\approx\; \mathbf{h}_i(\bar{\mathbf{x}}(t_i)) + \underbrace{\left.\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}}(t_i)}}_{=:\mathbf{H}_i} \mathbf{\Phi}(t_i)\,\delta\mathbf{c} + \mathbf{n}_i. \tag{14}$$

It should be noted that we require a value for the state at each measurement time to perform linearization. In light of this requirement, these values are stored after each iteration.

Finally, to simplify the objective function, we define

$$\mathbf{z} := \begin{bmatrix}\mathbf{z}_1^T & \dots & \mathbf{z}_N^T\end{bmatrix}^T, \tag{15a}$$

$$\mathbf{h} := \begin{bmatrix}\mathbf{h}_1(\bar{\mathbf{x}}(t_1))^T & \dots & \mathbf{h}_N(\bar{\mathbf{x}}(t_N))^T\end{bmatrix}^T, \tag{15b}$$

$$\mathbf{H} := \operatorname{diag}\left\{\mathbf{H}_1, \dots, \mathbf{H}_N\right\}, \tag{15c}$$

$$\mathbf{R} := \operatorname{diag}\left\{\mathbf{R}_1, \dots, \mathbf{R}_N\right\}, \tag{15d}$$

$$\mathbf{\Phi} := \begin{bmatrix}\mathbf{\Phi}(t_1)^T & \dots & \mathbf{\Phi}(t_N)^T\end{bmatrix}^T, \tag{15e}$$

which results in

$$J = \frac{1}{2}\left(\mathbf{z}-\mathbf{h}-\mathbf{H}\,\mathbf{\Phi}\,\delta\mathbf{c}\right)^T \mathbf{R}^{-1}\left(\mathbf{z}-\mathbf{h}-\mathbf{H}\,\mathbf{\Phi}\,\delta\mathbf{c}\right) + \frac{1}{2}\left(\bar{\mathbf{c}}+\delta\mathbf{c}-\mathbf{m}\right)^T \mathbf{K}^{-1}\left(\bar{\mathbf{c}}+\delta\mathbf{c}-\mathbf{m}\right). \tag{16}$$

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 $\delta\mathbf{c}$ gives us

$$\frac{\partial J}{\partial\,\delta\mathbf{c}}^T = \left(-\mathbf{H}\mathbf{\Phi}\right)^T \mathbf{R}^{-1}\left(\mathbf{z}-\mathbf{h}-\mathbf{H}\,\mathbf{\Phi}\,\delta\mathbf{c}\right) + \mathbf{K}^{-1}\left(\bar{\mathbf{c}}+\delta\mathbf{c}-\mathbf{m}\right), \tag{17}$$

and setting the value to zero provides us with $\delta\mathbf{c}^*$, the optimal value of the perturbation that minimizes the quadratic expression (at this iteration):

$$\begin{aligned} \underbrace{\left(\mathbf{\Phi}^T\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{\Phi} + \mathbf{K}^{-1}\right)}_{=:\mathbf{D}} \delta\mathbf{c}^* \\ = \mathbf{\Phi}^T\mathbf{H}^T\mathbf{R}^{-1}(\mathbf{z}-\mathbf{h}) - \mathbf{K}^{-1}\left(\bar{\mathbf{c}}-\mathbf{m}\right). \end{aligned} \tag{18}$$

Since this derivation follows the Gauss-Newton approach, we can express the mean, $\delta\mathbf{c}^*$, as

$$\delta\mathbf{c}^* = \mathbf{D}^{-1}\left(\mathbf{\Phi}^T\mathbf{H}^T\mathbf{R}^{-1}(\mathbf{z}-\mathbf{h}) - \mathbf{K}^{-1}(\bar{\mathbf{c}}-\mathbf{m})\right), \tag{19}$$

and the covariance as

$$\operatorname{cov}(\delta\mathbf{c}^*, \delta\mathbf{c}^*) = \mathbf{D}^{-1}. \tag{20}$$

These expressions provide the solution to the parametric estimation problem.

C. Returning to a Non-Parametric Form

We return to a non-parametric form by first applying the basis function relation (5) to obtain the mean function, $\delta\mathbf{x}^*(t)$, which can be expressed as

$$\delta\mathbf{x}^*(t) = \mathbf{\Phi}(t)\,\mathbf{D}^{-1}\mathbf{\Phi}^T\mathbf{H}^T\mathbf{R}^{-1}(\mathbf{z}-\mathbf{h}) - \mathbf{\Phi}(t)\,\mathbf{D}^{-1}\mathbf{K}^{-1}(\bar{\mathbf{c}}-\mathbf{m})), \tag{21}$$

and with a covariance function of

$$\operatorname{cov}(\delta\mathbf{x}^*(t), \delta\mathbf{x}^*(t')) = \mathbf{\Phi}(t)\,\mathbf{D}^{-1}\,\mathbf{\Phi}(t')^T. \tag{22}$$

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 [21], that

$$\mathbf{D}^{-1} \equiv \mathbf{K} - \mathbf{K}\mathbf{\Phi}^T\mathbf{H}^T\left(\mathbf{R}+\mathbf{H}\mathbf{\Phi}\mathbf{K}\mathbf{\Phi}^T\mathbf{H}^T\right)^{-1}\mathbf{H}\mathbf{\Phi}\mathbf{K}, \tag{23}$$

and

$$\mathbf{D}^{-1}\mathbf{\Phi}^T\mathbf{H}^T \equiv \mathbf{K}\mathbf{\Phi}^T\mathbf{H}^T\left(\mathbf{R}+\mathbf{H}\mathbf{\Phi}\mathbf{K}\mathbf{\Phi}^T\mathbf{H}^T\right)^{-1}\mathbf{R}. \tag{24}$$

We complete our derivation by substituting these identities into (21)-(22), and reapplying the basis function relations (8)-(9), to produce the GPGN update expressions (25)-(26),

$$
\begin{aligned}
\delta\mathbf{x}^*(t) &= \boldsymbol{\Phi}(t)\,\mathbf{K}\boldsymbol{\Phi}^T\mathbf{H}^T\left(\mathbf{R}+\mathbf{H}\boldsymbol{\Phi}\mathbf{K}\boldsymbol{\Phi}^T\mathbf{H}^T\right)^{-1}\mathbf{R}\mathbf{R}^{-1}(\mathbf{z}-\mathbf{h}) \\
&\quad -\boldsymbol{\Phi}(t)\left(\mathbf{K}-\mathbf{K}\boldsymbol{\Phi}^T\mathbf{H}^T\left(\mathbf{R}+\mathbf{H}\boldsymbol{\Phi}\mathbf{K}\boldsymbol{\Phi}^T\mathbf{H}^T\right)^{-1}\mathbf{H}\boldsymbol{\Phi}\mathbf{K}\right)\mathbf{K}^{-1}(\bar{\mathbf{c}}-\mathbf{m})) \\
&= \boldsymbol{\Phi}(t)\,(\mathbf{m}-\bar{\mathbf{c}})+\boldsymbol{\Phi}(t)\,\mathbf{K}\boldsymbol{\Phi}^T\mathbf{H}^T\left(\mathbf{R}+\mathbf{H}\boldsymbol{\Phi}\mathbf{K}\boldsymbol{\Phi}^T\mathbf{H}^T\right)^{-1}(\mathbf{z}-\mathbf{h}-\mathbf{H}\boldsymbol{\Phi}\,(\mathbf{m}-\bar{\mathbf{c}})) \\
&= (\boldsymbol{\mu}(t)-\bar{\mathbf{x}}(t))+\boldsymbol{\mathcal{K}}(t)\,\mathbf{H}^T\left(\mathbf{R}+\mathbf{H}\boldsymbol{\mathcal{K}}\mathbf{H}^T\right)^{-1}(\mathbf{z}-\mathbf{h}-\mathbf{H}\,(\boldsymbol{\mu}-\bar{\mathbf{x}}))\,, \qquad (25) \\
\mathrm{cov}(\delta\mathbf{x}^*(t),\delta\mathbf{x}^*(t')) &= \boldsymbol{\Phi}(t)\left(\mathbf{K}-\mathbf{K}\boldsymbol{\Phi}^T\mathbf{H}^T\left(\mathbf{R}+\mathbf{H}\boldsymbol{\Phi}\mathbf{K}\boldsymbol{\Phi}^T\mathbf{H}^T\right)^{-1}\mathbf{H}\boldsymbol{\Phi}\mathbf{K}\right)\boldsymbol{\Phi}(t')^T \\
&= \boldsymbol{\Phi}(t)\,\mathbf{K}\boldsymbol{\Phi}(t')^T-\boldsymbol{\Phi}(t)\,\mathbf{K}\boldsymbol{\Phi}^T\mathbf{H}^T\left(\mathbf{R}+\mathbf{H}\boldsymbol{\Phi}\mathbf{K}\boldsymbol{\Phi}^T\mathbf{H}^T\right)^{-1}\mathbf{H}\boldsymbol{\Phi}\mathbf{K}\boldsymbol{\Phi}(t')^T \\
&= \boldsymbol{\mathcal{K}}(t,t')-\boldsymbol{\mathcal{K}}(t)\,\mathbf{H}^T\left(\mathbf{R}+\mathbf{H}\boldsymbol{\mathcal{K}}\mathbf{H}^T\right)^{-1}\mathbf{H}\boldsymbol{\mathcal{K}}(t')^T, \qquad (26)
\end{aligned}
$$

where

$$
\begin{aligned}
\boldsymbol{\mu} &:= \begin{bmatrix}\boldsymbol{\mu}(t_1)^T & \dots & \boldsymbol{\mu}(t_N)^T\end{bmatrix}^T, && (27a)\\
\bar{\mathbf{x}} &:= \begin{bmatrix}\bar{\mathbf{x}}(t_1)^T & \dots & \bar{\mathbf{x}}(t_N)^T\end{bmatrix}^T, && (27b)\\
\boldsymbol{\mathcal{K}}(t) &:= \boldsymbol{\Phi}(t)\,\mathbf{K}\boldsymbol{\Phi}^T = \begin{bmatrix}\boldsymbol{\mathcal{K}}(t,t_1) & \dots & \boldsymbol{\mathcal{K}}(t,t_N)\end{bmatrix}, && (27c)\\
\boldsymbol{\mathcal{K}} &:= \boldsymbol{\Phi}\mathbf{K}\boldsymbol{\Phi}^T = \begin{bmatrix}\boldsymbol{\mathcal{K}}(t_1,t_1) & \dots & \boldsymbol{\mathcal{K}}(t_1,t_N)\\ \vdots & \ddots & \vdots \\ \boldsymbol{\mathcal{K}}(t_N,t_1) & \dots & \boldsymbol{\mathcal{K}}(t_N,t_N)\end{bmatrix}. && (27d)
\end{aligned}
$$

The substitution of $\boldsymbol{\Phi}(t)\,\mathbf{K}\boldsymbol{\Phi}(t')^T$ by $\boldsymbol{\mathcal{K}}(t,t')$ (known as the *kernel trick*) recovers computational tractability. Further simplifications can be achieved if we are interested in the perturbations of the state at only the measurement times. If we define $\delta\mathbf{x}^* := \begin{bmatrix}\delta\mathbf{x}^*(t_1)^T & \dots & \delta\mathbf{x}^*(t_N)^T\end{bmatrix}^T$, we get

$$
\begin{aligned}
\delta\mathbf{x}^* &= (\boldsymbol{\mu}-\bar{\mathbf{x}})+\boldsymbol{\mathcal{K}}\mathbf{H}^T\left(\mathbf{R}+\mathbf{H}\boldsymbol{\mathcal{K}}\mathbf{H}^T\right)^{-1} \\
&\quad \times(\mathbf{z}-\mathbf{h}-\mathbf{H}\,(\boldsymbol{\mu}-\bar{\mathbf{x}}))\,, \qquad (28)\\
\mathrm{cov}(\delta\mathbf{x}^*,\delta\mathbf{x}^*) &= \boldsymbol{\mathcal{K}}-\boldsymbol{\mathcal{K}}\mathbf{H}^T\left(\mathbf{R}+\mathbf{H}\boldsymbol{\mathcal{K}}\mathbf{H}^T\right)^{-1}\mathbf{H}\boldsymbol{\mathcal{K}}^T, \qquad (29)
\end{aligned}
$$

which, by application of the SMW identity once more, produces the equivalent expressions in information form:

$$
\begin{aligned}
\left(\boldsymbol{\mathcal{K}}^{-1}+\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)\delta\mathbf{x}^* &= \boldsymbol{\mathcal{K}}^{-1}(\boldsymbol{\mu}-\bar{\mathbf{x}}) \\
&\quad +\mathbf{H}^T\mathbf{R}^{-1}(\mathbf{z}-\mathbf{h}), \qquad (30)\\
\mathrm{cov}(\delta\mathbf{x}^*,\delta\mathbf{x}^*)^{-1} &= \boldsymbol{\mathcal{K}}^{-1}+\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}. \qquad (31)
\end{aligned}
$$

In summary, state estimation using the GPGN algorithm is performed as follows:

1) Start with an initial guess for the state at the measurement times, $\bar{\mathbf{x}}$, defined by (27b).
2) Linearize the system models, and construct the (15) and (27) matrices.
3) Solve for the optimal value of the perturbations at the measurement times, $\delta\mathbf{x}^*$, using (30).
4) Apply the additive update, $\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}}+\delta\mathbf{x}^*$.
5) Repeat steps 2-4 until convergence.
6) After convergence, compute the state and covariance of $\mathbf{x}(t)$ at other timesteps using (25) and (26).

As can be seen, this algorithm has a similar structure to the conventional discrete-time batch Gauss-Newton method. In fact, the estimate update expressions (30)-(31) simply have the additional $\boldsymbol{\mathcal{K}}^{-1}$ and $\boldsymbol{\mathcal{K}}^{-1}(\boldsymbol{\mu}-\bar{\mathbf{x}})$ components, which serve as regularization terms. The storage and computational requirements scale with the number of unique measurement times in both approaches, but conventional discrete-time Gauss-Newton implementations can utilize less computational resources by exploiting the matrix sparsity structure. Unfortunately, the $\boldsymbol{\mathcal{K}}^{-1}$ matrix in GPGN may be fully dense.

V. Experimental Validation

For experimental validation of the proposed algorithm, we leverage the robotics community's prior experience with the well-understood problem of 2D landmark-based localization using a known map. For comparison, GPGN was implemented alongside the conventional discrete-time batch Gauss-Newton approach. Discrete-time batch Gauss-Newton encompasses the traditional smoothing approaches, as it utilizes all of the measurements in a discrete-time ML formulation. To emulate the ability of GPGN to produce additional estimates between measurement times, linear interpolation was employed for the discrete-time estimates.

The intent of this section is not to convince the reader that 2D localization problems should be solved using GPGN; these are better solved using 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. Due to space limitations, we are unable to present derivations for the mathematical models in this section.

A. Overview

The experimental setup consisted of a mobile rover driving in an indoor, planar environment, amongst a forest of plastic tubes, depicted in Figure 2. These tubes served as landmarks for our localization problem. The rover provided two measurement types: instantaneous linear and angular

Figure 2. The hardware setup for the rover localization experiment. This consisted of a mobile rover mounted with a laser rangefinder driving in a forest of plastic tubes, and a motion tracking system for ground truth data.

velocities through wheel odometry, and range and bearing measurements to the landmarks using a laser rangefinder. Known data association was provided, and the maximum observation range was artificially limited to 3m. The odometry and landmark measurements were collected at a rate of 1Hz during the 100m traverse, and after estimator convergence, additional estimates were computed at a rate of 10Hz. For ground truth data, a Vicon motion capture system was used to track retroreflective markers placed on both the robot and the plastic tubes.

B. GPGN Implementation

For the continuous-time formulation, we defined our state vector to be

$$\mathbf{x}(t) := \begin{bmatrix} x(t) & y(t) & \theta(t) & \dot{x}(t) & \dot{y}(t) & \dot{\theta}(t) \end{bmatrix}^T, \quad (32)$$

the rover's pose and velocity at time t. The velocity estimates were required to linearize the odometry measurements. The state acceleration was modelled as a white noise process

$$\ddot{\mathbf{x}}(t) \sim \mathcal{GP}\left(\mathbf{0}, \mathbf{P}\,\delta(t-t')\right), \quad (33)$$

where $\mathbf{P}$ is the state covariance at an instant of time, and $\delta(\cdot)$ is the Dirac delta function. Integrating the mean and covariance functions twice produced the GP for $\mathbf{x}(t)$, where

$$\boldsymbol{\mu}(t) = \mathbf{x}(0), \quad (34)$$

$$\mathcal{K}(t,t') = \mathbf{P}\left(\frac{\min(t,t')^2 \max(t,t')}{2} - \frac{\min(t,t')^3}{6}\right). \quad (35)$$

Along with this process model, the odometry measurements of instantaneous linear and angular velocities were defined as

$$\mathbf{u}_k := \begin{bmatrix} \sqrt{\dot{x}(t_k)^2 + \dot{y}(t_k)^2} \\ \dot{\theta}(t_k) \end{bmatrix} + \mathbf{w}_k, \;\; \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k), \quad (36)$$

where $\mathbf{w}_k$ was corruptive zero-mean Gaussian noise with a covariance of $\mathbf{Q}_k$. Derivative-type measurements are incorporated into GPGN by simply defining $\dot{\mathbf{x}}(t) := \dot{\mathbf{\Phi}}(t)\,\mathbf{c}$, and redefining the associated matrices accordingly.

Finally, the laser rangefinder measurements to landmark l were modelled as

$$\begin{aligned} \mathbf{z}_i &:= \begin{bmatrix} \Big((x_l - x(t_i) - d\cos\theta(t_i))^2 \\ + (y_l - y(t_i) - d\sin\theta(t_i))^2\Big)^{1/2} \\ \text{atan2}\Big(y_l - y(t_i) - d\sin\theta(t_i), \\ x_l - x(t_i) - d\cos\theta(t_i)\Big) - \theta(t_i) \end{bmatrix} + \mathbf{n}_{i,l}, \\ \mathbf{n}_{i,l} &\sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{i,l}), \end{aligned} \quad (37)$$

where (x_l, y_l) was the known position of landmark l, d was the offset between the robot and sensor centres, and $\mathbf{n}_{i,l}$ the corruptive Gaussian noise with a covariance of $\mathbf{R}_{i,l}$.

For all three system models, the noise parameters were determined using the ground truth data by fitting Gaussian distributions to the computed errors.

C. Discrete-Time Gauss-Newton Implementation

Similarly, for the discrete-time formulation, we defined our state vector to be the rover pose at timestep k:

$$\mathbf{x}_k := \begin{bmatrix} x_k & y_k & \theta_k \end{bmatrix}^T. \quad (38)$$

Velocities were not required because we utilized a discrete-time unicycle model to produce the odometry measurements

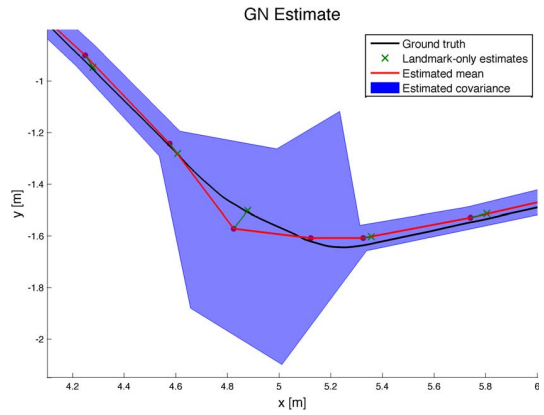
$$\begin{aligned} \mathbf{u}_k &:= \frac{1}{T}\begin{bmatrix} \cos\theta_{k-1} & \sin\theta_{k-1} & 0 \\ 0 & 0 & 1 \end{bmatrix}(\mathbf{x}_k - \mathbf{x}_{k-1}) + \mathbf{w}_k, \\ \mathbf{w}_k &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k), \end{aligned} \quad (39)$$

where T was the sampling period. The laser measurement model remained the same as the GPGN case (37).

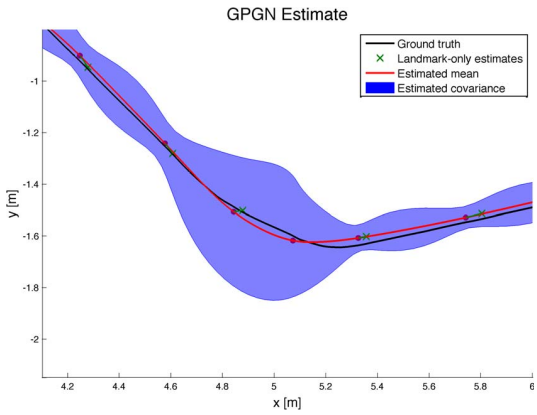
D. Results

To highlight the differences between the two approaches, we begin by focusing on a small section of the traverse. In this section, the rover conducted a short turn, and landmarks were not observed at one timestep during the maneuver. This caused the estimators to rely only on the odometry measurement for that period of time. The results produced by the two estimators, depicted in Figure 3, were quite different.

For discrete-time Gauss-Newton, Figure 3(a) shows that the one timestep without landmark measurements led to an inaccurate estimate, and a large jump in the uncertainty envelope. Since sufficient landmark measurements were available on either side of the turn, a jagged estimate in the middle was produced to connect the two sides. In comparison, Figure 3(b) shows that the underlying process model provided a much smoother estimate for GPGN, including the section with a momentary loss of landmark measurements. This continuous-time probabilistic process model also produced covariance envelopes that grew and shrank smoothly between measurement timesteps. Since no measurement information is available between timesteps, it is appropriate that the uncertainty is higher in these regions.

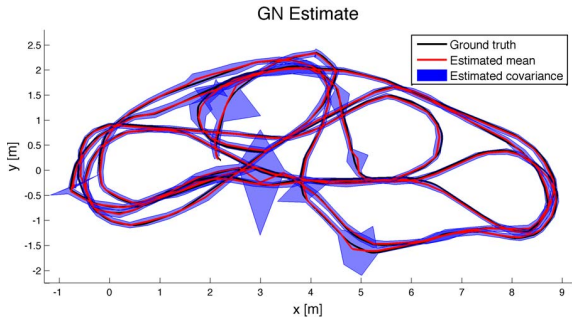


(a) The estimate produced by discrete-time Gauss-Newton. As can be seen, the one timestep without landmark measurements led to an inaccurate angular estimate, and a large jump in the uncertainty envelope. This resulted in a jagged mean estimate during the momentary loss of landmark observations.

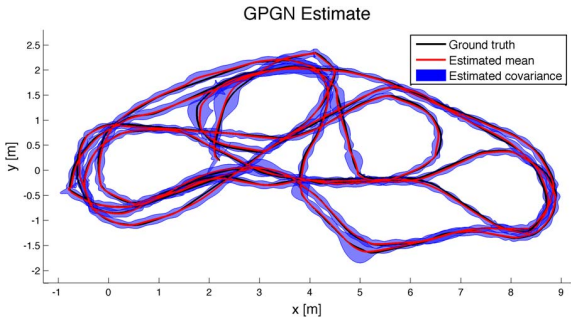


(b) The estimate produced by GPGN. The estimated trajectory and its associated covariance envelope was smooth, including the section with a momentary loss of landmark measurements. In addition, the covariance envelope grew and shrank appropriately between measurement timesteps.

Figure 3. Plots depicting the results of the two estimators for a short section of the rover traverse. The ground truth trajectory is indicated by the black line, and the estimated positions, along with their associated 3σ lateral covariance envelopes, are indicated by the red line and the blue shading, respectively. Furthermore, measurement timesteps are indicated by the red dots. The green crosses indicate estimates produced at each measurement timestep using the landmark measurements only, and the green lines connect the landmark-only measurements with their associated timesteps. The lateral covariance envelopes were determined by marginalizing the estimated (x, y) covariances onto the lateral dimension defined by the mean orientation estimate.



(a) The estimate produced by discrete-time Gauss-Newton. Though the performance was quite accurate for the majority of the traverse, jagged estimates were present. Similar inconsistencies can be seen for the covariance envelope in the curved sections.



(b) The estimate produced by GPGN. The estimated trajectory and its associated covariance envelope was smooth, accurately matching the ground truth path. Once again, the covariance envelope grew and shrank appropriately in intervals of insufficient data.

Figure 4. Plots depicting the results of the two estimators for the rover traverse. The ground truth trajectory is indicated by the black line, and the estimated positions, along with their associated 3σ lateral covariance envelopes, are indicated by the red line and the blue shading, respectively. The lateral covariance envelopes were determined by marginalizing the estimated (x, y) covariances onto the lateral dimension defined by the mean orientation estimate.

These differences between the two approaches can also be seen in the estimates for the whole traverse, depicted in Figure 4. The estimate produced by discrete-time Gauss-Newton contained jagged sections, and while the covariance envelope appears to capture the ground truth path accurately, inconsistencies can be seen in the curved sections. Once again, a smooth estimate was produced by GPGN, and the covariance envelope grew and shrank appropriately, resulting in the humps between measurement timesteps.

Again, while the intent was not to convince the reader that 2D localization problems should be solved using GPGN, we computed the root mean squared (RMS) estimate errors for quantitative comparison. Discrete-time Gauss-Newton achieved RMS errors of 0.0399m in translation, and 2.17° in rotation. In comparison, GPGN achieved RMS errors of 0.0349m in translation, and 1.84° in rotation, which was a 12.5% improvement in translational accuracy, and 15.0% in rotational accuracy.

Unfortunately, these smooth estimates came with increased computation time. For discrete-time Gauss-Newton, the estimation algorithm converged in 1.3s, with an extra 0.1s for the linear interpolation of the additional estimates. This was significantly less than the 47s utilized by GPGN for convergence, and the extra 135s for the additional estimates.

Timing information recorded on a MacBook Pro with a 2.66GHz Core 2 Duo and 4GB of 1067MHz DDR3 RAM in Matlab, utilizing both cores.

VI. Conclusion

In conclusion, we have presented GPGN, a novel algorithm for non-parametric, continuous-time, nonlinear, batch state estimation. This work adapts concepts from GP regression, and applies it to the robotics domain. This application of GPs is significantly different from the previous uses of GPs in robotics. In particular, GPGN is able to address conventional robotic state estimation scenarios. To avoid overwhelming the reader with too many new concepts, we deliberately avoided presenting a complex experimental scenario for validation of GPGN. Rather, we utilized a well-understood state estimation problem to illustrate the tradeoffs of GPGN compared with the conventional discrete-time 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 handle issues related to measurement scarcity.

While the improvements in modelling accuracy also come with increased computational cost, we have simply presented the full GPGN solution without any approximations. Future work will involve addressing these concerns. A large amount of literature is available on GPs, including approximation using sparse methods, and alternative covariance functions [7]. Many of these variations on the basic framework of GP regression can be transferred to the robotics domain. In fact, the form of the mean and covariance functions were kept general in the GPGN derivation for this purpose.

Possible directions for future work include the incorporation of prior uncertainty to handle the sliding-window filtering scenario, and the formulation of GPGN with both continuous and discrete-time state components to address the canonical mobile robotic estimation problem, SLAM.

References

[1] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," *Autonomous Robot Vehicles*, pp. 167–193, 1990.

[2] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Journal of Basic Engineering*, vol. 83, no. 1, pp. 95–108, 1961.

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

[4] D. Simon, *Optimal State Estimation: Kalman, H_∞, and Nonlinear Approaches*. New York: John Wiley & Sons, Inc., 2006.

[5] C. Bibby and I. D. Reid, "A hybrid SLAM representation for dynamic marine environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, USA, 3-7 May 2010, pp. 257–264.

[6] P. Furgale, T. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *Proceedings of the 2012 International Conference on Robotics and Automation (ICRA), to appear.*, St. Paul, MN, USA, 14-18 May 2012.

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

[8] C. F. Gauss, *Méthode des Moindres Carrés*. Quai des Augustins no. 55, Paris: Mallet-Bachelier, Impreimeur-Libraire de L'École Polytechnique, 1855.

[9] C. Plagemann, K. Kersting, P. Pfaff, and W. Burgard, "Gaussian beam processes: A nonparametric Bayesian measurement model for range finders," in *Proceedings of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.

[10] C. Stachniss, C. Plagemann, and A. Lilienthal, "Learning gas distribution models using sparse gaussian process mixtures," *Autonomous Robots*, vol. 26, no. 2-3, April 2009.

[11] S. Vasudevan, F. T. Ramos, E. Nettleton, and H. Durrant-Whyte, "Gaussian process modeling of large-scale terrain," *Journal of Field Robotics*, vol. 26, no. 10, pp. 812–840, 2009.

[12] B. Ferris, D. Hähnel, and D. Fox, "Gaussian processes for signal strength-based localization," in *Proceedings of Robotics: Science and Systems (RSS)*, Philadelphia, USA, 2006.

[13] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using gaussian process prediction and observation models," *Autonomous Robots*, vol. 27, no. 1, pp. 75–90, July 2009.

[14] N. Lawrence, "Gaussian process latent variable models for visualization of high dimensional data," in *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[15] B. Ferris, D. Fox, and N. Lawrence, "Wifi-SLAM using gaussian process latent variable models," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2007, pp. 2480–2485.

[16] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 30, no. 2, pp. 283–298, February 2008.

[17] J. Ko and D. Fox, "Learning GP-BayesFilters via gaussian process latent variable models," *Autonomous Robots*, vol. 30, no. 1, pp. 3–23, January 2011.

[18] E. Snelson, C. E. Rasmussen, and Z. Ghahramani, "Warped gaussian processes," in *Advances in Neural Information Processing Systems (NIPS)*, 2004.

[19] E. Solak, R. Murray-Smith, W. E. Leithead, D. J. Leith, and C. E. Rasmussen, "Derivative observations in gaussian process models of dynamical systems," in *Advances in Neural Information Processing Systems (NIPS)*, 2003, pp. 1033–1040.

[20] C. Archambeau, D. Cornford, M. Opper, and J. Shawe-Taler, "Gaussian process approximations of stochastic differential equations," in *Journal of Machine Learning Research Workshop and Conference Proceedings*, vol. 1, 2007, pp. 1–16.

[21] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *Annals of Mathematics and Statistics*, vol. 21, no. 1, pp. 124–127, 1950.