Batch Continuous-Time Trajectory Estimation

Mollie Bianchi and Timothy D. Barfoot

***Abstract*— In this project, we conduct batch state estimation on two mobile robot datasets using Gaussian process (GP) regression. The measurements occur at discrete times but by introducing a continuous time prior, we are able to represent the trajectory in continuous time. To do so, a two step procedure is employed where first the state is solved for at the measurement times and then interpolation is used to query the trajectory at any additional times of interest. This project also explains how using a special class of prior allows for efficient implementation by exploiting the sparsity of the matrices involved. This is demonstrated by interpolating the state at a constant rate in a linear, one-dimensional problem and a non-linear, two dimensional problem for cases with asynchronous measurement times or prolonged measurement dropout.**

I. Introduction

Often robot trajectories are represented in discrete-time and in many cases this representation is sufficient; however, there are situations when the state of the robot is required between the discrete times at which it is estimated. For example, a vehicle receives pose estimates from a GPS unit while driving with a scanning lidar. Since the vehicle is moving while the lidar is rotating, the positions of the vehicle during the scan are needed to account for the motion distortion. These position estimates are required at a higher frequency than what is provided from GPS. A similar situation arises when using rolling-shutter cameras. In these cases, a continuous time representation of the trajectory would be more suitable. Another example would be when a robot's controller expects state estimates at a high, synchronous rate but receives measurements at asynchronous times. Or if the controller needs to send a command some time after the last measurement, and needs a more current estimate of the robot's pose.

One option to accomplish this continuous time representation is to just interpolate between the discrete poses using a chosen scheme (e.g. linear, cubic, etc.) [1]. This project instead represents the trajectory as a one-dimensional Gaussian process (GP) [2] with time as the independent variable. The choice of GP prior then automatically defines the interpolation scheme for the query times, removing the need to choose an interpolation scheme on an ad hoc basis. By choosing a particular class of GP priors based on linear, time-varying stochastic differential equations, the inverse kernel matrix is block-tridiagonal which allows for efficient implementation of GP regression. The GP regression can be separated into two steps: first solving for the trajectory at all of the measurement times, and then interpolating at the additional query times.

II. Related Work

Treating the query of a robot's state at an additional time of interest as a nonlinear, GP regression problem was shown in Tong et al. [3]. They take a very general approach and do not restrict the GP prior to a specific class. This does not allow them to exploit any sparse structure in the matrices and as a result, this approach lacks efficiency due to the inversion of a large, dense kernel matrix.

In [4], Barfot et al. introduce the idea of using priors generated by linear, time-varying stochastic differential equations. They showed that this results in an inverse kernel matrix that is block-tridiagonal allowing for efficient implementation. Restricting to this class of prior only results in a limited loss of generality. They demonstrate this approach on a simultaneous trajectory estimation and mapping problem using a mobile robot dataset.

This approach has been furthered in subsequent papers. [6] shows how a prior based on a nonlinear, time-varying stochastic differential equation also leads to a sparse inverse kernel matrix. [5] defines a prior using a first-order stochastic differential equation model on SE(3). They are then able to use the approach of [4] to conduct batch continuous-time trajectory estimation for bodies rotating and translating in three-dimensional space.

This project follows the approach presented in Chapter 3.4 of State Estimation for Robotics [7] for the case with a linear measurement model and follows [4] for the case with a nonlinear measurement model.

III. Methodology

A. Gaussian Process Regression

We will first consider systems with a continuous-time Gaussian Process (GP) model prior and a discrete-time, linear measurement model:

$$\mathbf{x}(t) \sim \mathcal{GP}(\check{\mathbf{x}}(t), \check{\mathbf{P}}(t,t')), \qquad t_0 < t, t' \tag{1}$$

$$\mathbf{y}_k = \mathbf{C}_k \mathbf{x}(t_k) + \mathbf{n}_k, \qquad t_1 < ... < t_K \tag{2}$$

where $\mathbf{x}(t)$ is the state, $\check{\mathbf{x}}(t)$ is the mean function, $\check{\mathbf{P}}(t,t')$ is the covariance or kernel function, $\mathbf{y}_k$ are the measurements, $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$ is Gaussian measurement noise, and $\mathbf{C}_k$ is the measurement model coefficient matrix. We want to query the state at times $(\tau_0 < \tau_1 < ... < \tau_J)$ which may be different from the measurement times $(t_0 < t_1 < ... < t_K)$. We can write the joint density between the state at the query times and the measurements at the measurement times as:

$$p\left(\begin{bmatrix}\mathbf{x}_\tau \\ \mathbf{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\check{\mathbf{x}}_\tau \\ \mathbf{C}\check{\mathbf{x}}\end{bmatrix}, \begin{bmatrix}\check{\mathbf{P}}_{\tau\tau} & \check{\mathbf{P}}_\tau \mathbf{C}^T \\ \mathbf{C}\check{\mathbf{P}}_\tau & \mathbf{R} + \mathbf{C}\check{\mathbf{P}}\mathbf{C}^T\end{bmatrix}\right) \tag{3}$$

where

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}(t_0) \\ \vdots \\ \mathbf{x}(t_K) \end{bmatrix}, \check{\mathbf{x}} = \begin{bmatrix} \check{\mathbf{x}}(t_0) \\ \vdots \\ \check{\mathbf{x}}(t_K) \end{bmatrix}, \mathbf{x}_\tau = \begin{bmatrix} \mathbf{x}(\tau_0) \\ \vdots \\ \mathbf{x}(\tau_J) \end{bmatrix},$$

$$\check{\mathbf{x}}_\tau = \begin{bmatrix} \check{\mathbf{x}}(\tau_0) \\ \vdots \\ \check{\mathbf{x}}(\tau_J) \end{bmatrix}, \mathbf{y} = \begin{bmatrix} \mathbf{y}_0 \\ \vdots \\ \mathbf{y}_K \end{bmatrix},$$

$$\mathbf{C} = \text{diag}(\mathbf{C}_0, ..., \mathbf{C}_K), \mathbf{R} = \text{diag}(\mathbf{R}_0, ..., \mathbf{R}_k),$$

$$\check{\mathbf{P}} = [\check{\mathbf{P}}(t_i, t_j)]_{ij}, \check{\mathbf{P}}_\tau = [\check{\mathbf{P}}(\tau_i, t_j)]_{ij}, \check{\mathbf{P}}_{\tau\tau} = [\check{\mathbf{P}}(\tau_i, \tau_j)]_{ij}$$

We then have:

$$p(\mathbf{x}_\tau|\mathbf{y}) = \mathcal{N}\Big(\check{\mathbf{x}}_\tau + \check{\mathbf{P}}_\tau \mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{C}\check{\mathbf{x}}), \\ \check{\mathbf{P}}_{\tau\tau} - \check{\mathbf{P}}_\tau \mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1} \mathbf{C}\check{\mathbf{P}}_\tau^T\Big). \quad (4)$$

If we take the query times to be at the measurement times, the above expression simplifies to:

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}\Big(\check{\mathbf{x}} + \check{\mathbf{P}}\mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{C}\check{\mathbf{x}}), \\ \check{\mathbf{P}} - \check{\mathbf{P}}\mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1} \mathbf{C}\check{\mathbf{P}}^T\Big). \quad (5)$$

If we apply the Sherman-Morrison-Woodbury (SMW) identity to (5), we get:

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}\Big((\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C})^{-1}(\check{\mathbf{P}}^{-1}\check{\mathbf{x}} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{y}), \\ (\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C})^{-1}\Big) \quad (6)$$

which we can rearrange to form the following linear system for $\hat{\mathbf{x}}$:

$$\underbrace{(\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C})^{-1}}_{\hat{\mathbf{P}}} \hat{\mathbf{x}} = \check{\mathbf{P}}^{-1}\check{\mathbf{x}} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{y}. \quad (7)$$

After solving for $\hat{\mathbf{x}}$ and $\hat{\mathbf{P}}$, the state and covariance at the measurement times, if we want to solve for $\hat{\mathbf{x}}_\tau$ and $\hat{\mathbf{P}}_{\tau\tau}$, the state and covariance at additional query times, we can use GP interpolation to do so. To arrive at these equations, we start with the mean and the covariance expressions from (5):

$$\hat{\mathbf{x}} = \check{\mathbf{x}} + \check{\mathbf{P}}\mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{C}\check{\mathbf{x}}) \quad (8)$$

$$\hat{\mathbf{P}} = \check{\mathbf{P}} - \check{\mathbf{P}}\mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1} \mathbf{C}\check{\mathbf{P}}^T \quad (9)$$

and rearrange to get:

$$\check{\mathbf{P}}^{-1}(\hat{\mathbf{x}} - \check{\mathbf{x}}) = \mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{C}\check{\mathbf{x}}) \quad (10)$$

$$\check{\mathbf{P}}^{-1}(\hat{\mathbf{P}} - \check{\mathbf{P}})\check{\mathbf{P}}^{-T} = -\mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}\mathbf{C}. \quad (11)$$

We can substitute these equations back into the mean and covariance equations from (4):

$$\hat{\mathbf{x}}_\tau = \check{\mathbf{x}}_\tau + \check{\mathbf{P}}_\tau \underbrace{\mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{C}\check{\mathbf{x}})}_{\check{\mathbf{P}}^{-1}(\hat{\mathbf{x}} - \check{\mathbf{x}})} \quad (12)$$

$$\hat{\mathbf{P}}_{\tau\tau} = \check{\mathbf{P}}_{\tau\tau} - \check{\mathbf{P}}_\tau \underbrace{\mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}\mathbf{C}}_{\check{\mathbf{P}}^{-1}(\hat{\mathbf{P}} - \check{\mathbf{P}})\check{\mathbf{P}}^{-T}} \check{\mathbf{P}}_\tau^T. \quad (13)$$

This gives the GP interpolation equations:

$$\hat{\mathbf{x}}_\tau = \check{\mathbf{x}}_\tau + (\check{\mathbf{P}}_\tau \check{\mathbf{P}}^{-1})(\hat{\mathbf{x}} - \check{\mathbf{x}}) \quad (14)$$

$$\hat{\mathbf{P}}_{\tau\tau} = \check{\mathbf{P}}_{\tau\tau} + (\check{\mathbf{P}}_\tau \check{\mathbf{P}}^{-1})(\hat{\mathbf{P}} - \check{\mathbf{P}})(\check{\mathbf{P}}_\tau \check{\mathbf{P}}^{-1})^T \quad (15)$$

.

In summary, Gaussian Process Regression can be seperated into a 2 step process. First $\hat{\mathbf{x}}$ and $\hat{\mathbf{P}}$, the state and covariance at the measurement times, is solved for using (7). Then $\hat{\mathbf{x}}_\tau$ and $\hat{\mathbf{P}}_{\tau\tau}$, the state and covariance at additional query times, is solved using the interpolation equations in (14) and (15). Assuming we know nothing about the structure of these matrices, the first step would have complexity $O(K^3)$ and the second step would have complexity $O(K^2 J)$ where K is the number of measurements and J is the number of additional query times. The following section presents a special class of GP priors that allow the sparsity of the matrices to be exploited.

B. Linear Time-Varying Stochastic Differential Equations

This section will show that using a prior based on linear time-varying (LTV) stochastic differential equations (SDEs) will result in efficient implementation of GP regression by exploiting the sparsity of the matrices. These priors have the form of:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{v}(t) + \mathbf{L}(t)\mathbf{w}(t) \quad (16)$$

with

$$\mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_c \delta(t - t')) \quad (17)$$

where $\mathbf{x}(t)$ is the state, $\mathbf{v}(t)$ is a known input, $\mathbf{A}(t), \mathbf{L}(t)$ are time-varying system matrices, and $\mathbf{w}(t)$ is the process noise given by a zero-mean GP with power spectral density matrix $\mathbf{Q}_c$. The general solution to this LTE SDE is:

$$\mathbf{x}(t) = \mathbf{\Phi}(t, t_0)\mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{\Phi}(t, s)\left(\mathbf{v}(s) + \mathbf{L}(s)\mathbf{w}(s)\right) ds \quad (18)$$

where $\mathbf{\Phi}(t, s)$ is the transition function and must satisfy the following properties:

$$\mathbf{\Phi}(t, t) = \mathbf{1} \quad (19)$$

$$\dot{\mathbf{\Phi}}(t, s) = \mathbf{A}(t)\mathbf{\Phi}(t, s) \quad (20)$$

$$\mathbf{\Phi}(t, s) = \mathbf{\Phi}(t, r)\mathbf{\Phi}(r, s). \quad (21)$$

$$(22)$$

To find the mean function, we take the expected value of the solution in (18):

$$\underbrace{\text{E}[\mathbf{x}(t)]}_{\bar{\mathbf{x}}(t)} = \mathbf{\Phi}(t, t_0) \underbrace{\text{E}[\mathbf{x}(t_0)]}_{\bar{\mathbf{x}}_0} + \int_{t_0}^{t} \mathbf{\Phi}(t, s)\big(\mathbf{v}(s) \\ + \mathbf{L}(s) \underbrace{\text{E}[\mathbf{w}(s)]}_{\mathbf{0}}\big) ds \quad (23)$$

where $\check{\mathbf{x}}_0$ is the initial value of the mean at t_0. The mean function is then:

$$\check{\mathbf{x}}(t) = \mathbf{\Phi}(t, t_0)\check{\mathbf{x}}_0 + \int_{t_0}^{t} \mathbf{\Phi}(t, s)\mathbf{v}(s) ds. \quad (24)$$

This can be written for a discrete sequence of measurement times as:

$$\check{\mathbf{x}}(t_k) = \mathbf{\Phi}(t_k, t_0)\check{\mathbf{x}}_0 + \sum_{n=1}^{k} \mathbf{\Phi}(t_k, t_n)\mathbf{v}_n \tag{25}$$

where

$$\mathbf{v}_k = \int_{t_{k-1}}^{t_k} \mathbf{\Phi}(t_k, s)\mathbf{v}(s)ds, \quad k = 1...K \tag{26}$$

or in lifted form as:

$$\check{\mathbf{x}} = \mathbf{A}\mathbf{v} \tag{27}$$

where

$$\check{\mathbf{x}} = \begin{bmatrix} \check{\mathbf{x}}_0 \\ \check{\mathbf{x}}_1 \\ \vdots \\ \check{\mathbf{x}}_K \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} \check{\mathbf{x}}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_K \end{bmatrix}, \tag{28}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{\Phi}(t_1, t_0) & \mathbf{1} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{\Phi}(t_2, t_0) & \mathbf{\Phi}(t_2, t_1) & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{\Phi}(t_{K-1}, t_0) & \mathbf{\Phi}(t_{K-1}, t_1) & \dots & \mathbf{1} & \\ \mathbf{\Phi}(t_K, t_0) & \mathbf{\Phi}(t_K, t_1) & \dots & \mathbf{\Phi}(t_K, t_{K-1}) & \mathbf{1} \end{bmatrix} \tag{29}$$

It is important to note that $\mathbf{A}$ is lower triangular.

To find the covariance function, we take the second probability moment of 18:

$$\begin{aligned} &\underbrace{\mathrm{E}[(\mathbf{x}(t) - \mathrm{E}[\mathbf{x}(t)])(\mathbf{x}(t') - \mathrm{E}[\mathbf{x}(t')])^T}_{\check{\mathbf{P}}(t,t')} \\ &= \mathbf{\Phi}(t, t_0) \underbrace{\mathrm{E}[(\mathbf{x}(t_0) - \mathrm{E}[\mathbf{x}(t_0)])(\mathbf{x}(t_0) - \mathrm{E}[\mathbf{x}(t_0)])^T}_{\check{\mathbf{P}}_0} \mathbf{\Phi}(t', t_0)^T \\ &+ \int_{t_0}^{t}\int_{t_0}^{t'} \mathbf{\Phi}(t, s)\mathbf{L}(s) \underbrace{E[\mathbf{w}(s)\mathbf{w}(s')^T]}_{\mathbf{Q}_c\delta(s-s')} \mathbf{L}(s')^T\mathbf{\Phi}(t', s')^T ds'ds \end{aligned} \tag{30}$$

where $\check{\mathbf{P}}_0$ is the initial covariance at t_0 and we assume $E[\mathbf{w}(t_0)\mathbf{w}(t)^T] = \mathbf{0}$. This then gives:

$$\begin{aligned} &\check{\mathbf{P}}(t, t') = \mathbf{\Phi}(t, t_0)\check{\mathbf{P}}_0\mathbf{\Phi}(t', t_0) \\ &+ \int_{t_0}^{t}\int_{t_0}^{t'} \mathbf{\Phi}(t, s)\mathbf{L}(s)\mathbf{Q}_c\mathbf{L}(s')^T\mathbf{\Phi}(t', s')^T\delta(s - s')ds'ds. \end{aligned} \tag{31}$$

If we integrate the second term once, we get:

$$\int_{t_0}^{t} \mathbf{\Phi}(t, s)\mathbf{L}(s)\mathbf{Q}_c\mathbf{L}(s')^T\mathbf{\Phi}(t', s)^T H(t' - s)ds \tag{32}$$

where $H(\cdot)$ is the Heaviside step function meaning there are three cases to consider: $t < t', t = t'$ and $t > t'$. We can write the second term then as:

$$\begin{aligned} &\int_{t_0}^{\min(t,t')} \mathbf{\Phi}(t, s)\mathbf{L}(s)\mathbf{Q}_c\mathbf{L}(s')^T\mathbf{\Phi}(t', s')^T ds = \\ &\begin{cases} \mathbf{\Phi}(t, t')\Big(\int_{t_0}^{t'} \mathbf{\Phi}(t', s)\mathbf{L}(s)\mathbf{Q}_c\mathbf{L}(s)^T\mathbf{\Phi}(t', s)^T ds\Big), & t' < t \\ \int_{t_0}^{t'} \mathbf{\Phi}(t, s)\mathbf{L}(s)\mathbf{Q}_c\boldsymbol{L}(s)^T\mathbf{\Phi}(t, s)^T ds, & t' = t \\ \Big(\int_{t_0}^{t'} \mathbf{\Phi}(t, s)\mathbf{L}(s)\mathbf{Q}_c\mathbf{L}(s)^T\mathbf{\Phi}(t, s)^T ds\Big)\mathbf{\Phi}(t, t')^T, & t < t' \end{cases} \end{aligned} \tag{33}$$

This can be written for a discrete sequence of measurement times as:

$$\begin{aligned} &\check{\mathbf{P}}(t_i, t_j) = \\ &\begin{cases} \mathbf{\Phi}(t_i, t_j)\Big(\sum_{n=0}^{j} \mathbf{\Phi}(t_j, t_n)\mathbf{Q}_n\mathbf{\Phi}(t_j, t_n)^T\Big), & t_j < t_i \\ \sum_{n=0}^{i} \mathbf{\Phi}(t_i, t_n)\mathbf{Q}_n\mathbf{\Phi}(t_i, t_n)^T, & t_i = t_j \\ \Big(\sum_{n=0}^{i} \mathbf{\Phi}(t_i, t_n)\mathbf{Q}_n\mathbf{\Phi}(t_i, t_n)^T\Big)\mathbf{\Phi}(t_j, t_i)^T, & t_i < t_j \end{cases} \end{aligned} \tag{34}$$

where

$$\mathbf{Q}_k = \int_{t_{k-1}}^{t_k} \mathbf{\Phi}(t_k, s)\mathbf{L}(s)\mathbf{Q}_c\mathbf{L}(s)^T\mathbf{\Phi}(t_k, s)^T ds, \quad k = 1...K \tag{35}$$

and $\mathbf{Q}_0 = \check{\mathbf{P}}_0$. We can then construct the full $\check{\mathbf{P}}$ matrix by evaluating (34) for each block in $\check{\mathbf{P}} = [\check{\mathbf{P}}(t_i, t_j)]_{ij}$. We can then factor $\check{\mathbf{P}}$ according to a block-lower-diagonal-upper decomposition:

$$\check{\mathbf{P}} = \mathbf{A}\mathbf{Q}\mathbf{A}^T \tag{36}$$

where $\mathbf{A}$ is the lower triangular matrix defined in (29) and $\mathbf{Q} = \mathrm{diag}(\check{\mathbf{P}}_0, \mathbf{Q}_1, \mathbf{Q}_2, ..., \mathbf{Q}_K)$ where the individual $\mathbf{Q}_k$'s are defined in (35). The inverse can be computed by:

$$\check{\mathbf{P}}^{-1} = (\mathbf{A}\mathbf{Q}\mathbf{A}^T)^{-1} = \mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1} \tag{37}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ -\mathbf{\Phi}(t_1, t_0) & \mathbf{1} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{\Phi}(t_2, t_1) & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{1} & \\ \mathbf{0} & \mathbf{0} & \dots & -\mathbf{\Phi}(t_K, t_{K-1}) & \mathbf{1} \end{bmatrix}. \tag{38}$$

Since the only non-zero elements of $\mathbf{A}^{-1}$ are on the main diagonal and the one below it and $\mathbf{Q}^{-1}$ is block-diagonal, $\check{\mathbf{P}}^{-1}$ can easily be computed and we see that $\check{\mathbf{P}}^{-1}$ has a block tridiagonal structure. The prior at the measurement times can then be summarized:

$$\mathbf{x} \sim \mathcal{N}(\check{\mathbf{x}}, \check{\mathbf{P}}) = \mathcal{N}(\mathbf{A}\mathbf{v}, \mathbf{A}\mathbf{Q}\mathbf{A}^T) \tag{39}$$

C. Solving at the Measurement Times

We can plug in our values for the prior mean and covariance into (7):

$$\underbrace{(\mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C})}_{\text{block tridiagonal}}\hat{\mathbf{x}} = \mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{v} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{y} \tag{40}$$

Since the left hand side of this system is block tridiagonal, it can be solved in $O(K)$ time using a sparse solver (e.g. sparse Cholesky decomposition then a forward backwards pass or the Rauch-Tung-Striebel smoother).

D. Querying the Gaussian Process

If we consider a single query time, $t_k \leq \tau \leq t_{k+1}$, we can write expressions for the mean and covariance at τ using (24) and (31):

$$\check{\mathbf{x}}(\tau) = \mathbf{\Phi}(\tau, t_k)\check{\mathbf{x}}(t_k) + \int_{t_k}^{\tau} \mathbf{\Phi}(\tau, s)\mathbf{v}(s)ds \tag{41}$$

$$\begin{aligned}\check{\mathbf{P}}(\tau,\tau) = \mathbf{\Phi}(\tau, t_k)\check{\mathbf{P}}(t_k, t_k)\mathbf{\Phi}(\tau, tk)^T \\ + \int_{t_k}^{\tau} \mathbf{\Phi}(\tau, s)\mathbf{L}(s)\mathbf{Q}_c\mathbf{L}(s)^T\mathbf{\Phi}(\tau, s)^T ds\end{aligned} \tag{42}$$

Looking at the interpolation equations in (14) and (15), we still need to compute the $\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}$ term. The matrix $\check{\mathbf{P}}(\tau)$ can be written as:

$$\check{\mathbf{P}}(\tau) = [\check{\mathbf{P}}(\tau, t_0)\ \check{\mathbf{P}}(\tau, t_1)\ ...\ \check{\mathbf{P}}(\tau, t_K)] \tag{43}$$

where the blocks are given by:

$$\check{\mathbf{P}}(\tau, t_j) = \begin{cases} \mathbf{\Phi}(\tau, t_k)\mathbf{\Phi}(\tau, t_k)\big(\sum_{n=0}^{j} \mathbf{\Phi}(t_j, t_n)\mathbf{Q}_n\mathbf{\Phi}(t_j, t_n)^T\big), & t_j < t_k \\ \mathbf{\Phi}(\tau, t_k)\big(\sum_{n=0}^{k} \mathbf{\Phi}(t_k, t_n)\mathbf{Q}_n\mathbf{\Phi}(t_k, t_n)^T\big), & t_j = t_k \\ \mathbf{\Phi}(\tau, t_k)\big(\sum_{n=0}^{k} \mathbf{\Phi}(t_k, t_n)\mathbf{Q}_n\mathbf{\Phi}(t_k, t_n)^T\big)\mathbf{\Phi}(t_{k+1}, t_k)^T & \\ \quad + \mathbf{Q}_\tau\mathbf{\Phi}(t_{k+1}, \tau)^T, & t_{k+1} = t_j \\ \mathbf{\Phi}(\tau, t_k)\big(\sum_{n=0}^{k} \mathbf{\Phi}(t_k, t_n)\mathbf{Q}_n\mathbf{\Phi}(t_k, t_n)^T\big)\mathbf{\Phi}(t_j, t_k)^T & \\ \quad \mathbf{Q}_\tau\mathbf{\Phi}(t_{k+1}, \tau)^T\mathbf{\Phi}(t_j, t_{k+1})^T, & t_{k+1} < t_j \end{cases}$$

where

$$\mathbf{Q}_\tau = \int_{t_k}^{\tau} \mathbf{\Phi}(\tau, s)\mathbf{L}(s)\mathbf{Q}_c\mathbf{L}(s)^T\mathbf{\Phi}(\tau, s)^T ds. \tag{44}$$

Based on this, $\check{\mathbf{P}}(\tau)$ can be factored according to:

$$\check{\mathbf{P}}(\tau) = \mathbf{V}(\tau)\mathbf{A}^T \tag{45}$$

where

$$\begin{aligned}\mathbf{V}(\tau) = [\mathbf{\Phi}(\tau, t_k)\mathbf{\Phi}(t_k, t_0)\check{\mathbf{P}}_0 \quad \mathbf{\Phi}(\tau, t_k)\mathbf{\Phi}(t_k, t_1)\mathbf{Q}_1\ ... \\ ...\ \mathbf{\Phi}(\tau, t_k)\mathbf{\Phi}(t_k, t_{k-1})\mathbf{Q}_{k-1}\ ... \\ ...\ \mathbf{\Phi}(\tau, t_k)\mathbf{Q}_k \quad \mathbf{Q}_\tau\mathbf{\Phi}(t_{k+1}, \tau)^T \quad 0\ ...\ 0].\end{aligned} \tag{46}$$

Plugging this factorization into the term of interest gives:

$$\begin{aligned}\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1} &= \mathbf{V}(\tau)\underbrace{\mathbf{A}^T\mathbf{A}^{-T}}_{\mathbf{1}}\mathbf{Q}^{-1}\mathbf{A}^{-1} \qquad (47)\\ &= \mathbf{V}(\tau)\mathbf{Q}^{-1}\mathbf{A}^{-1} \qquad (48)\end{aligned}$$

This is now straightforward to evaluate as $\mathbf{Q}^{-1}$ is block-diagonal and $\mathbf{A}^{-1}$ has only the main diagonal and one below it non zero. The product is then:

$$\check{\mathbf{P}}_\tau\check{\mathbf{P}}^{-1} = [\mathbf{0} \quad ... \quad \mathbf{0} \quad \underbrace{\mathbf{\Lambda}(\tau)}_{\text{column k}} \quad \underbrace{\mathbf{\Psi}(\tau)}_{\text{column k+1}} \quad \mathbf{0} \quad ... \quad \mathbf{0}] \tag{49}$$

where

$$\mathbf{\Psi}(\tau) = \mathbf{Q}_\tau\mathbf{\Phi}(t_{k+1}, \tau)^T\mathbf{Q}_{k+1}^{-1} \tag{50}$$

$$\mathbf{\Lambda}(\tau) = \mathbf{\Phi}(\tau, t_k) - \mathbf{\Psi}(\tau)\mathbf{\Phi}(t_{k+1}, t_k). \tag{51}$$

The product only has two non-zero columns at k and $k+1$. This can then be plugged back into the original interpolation equations from (14) and (15) giving:

$$\hat{\mathbf{x}}(\tau) = \check{\mathbf{x}}(\tau) + [\mathbf{\Lambda}(\tau) \quad \mathbf{\Psi}(\tau)]\left(\begin{bmatrix}\hat{\mathbf{x}}_k \\ \hat{\mathbf{x}}_{k+1}\end{bmatrix} - \begin{bmatrix}\check{\mathbf{x}}(t_k) \\ \check{\mathbf{x}}(t_{k+1})\end{bmatrix}\right) \tag{52}$$

$$\begin{aligned}\hat{\mathbf{P}}(\tau,\tau) = \check{\mathbf{P}}(\tau,\tau) + [\mathbf{\Lambda}(\tau) \quad \mathbf{\Psi}(\tau)]\left(\begin{bmatrix}\hat{\mathbf{P}}_{k,k} & \hat{\mathbf{P}}_{k,k+1} \\ \hat{\mathbf{P}}_{k+1,k} & \hat{\mathbf{P}}_{k+1,k+1}\end{bmatrix}\right. \\ \left. - \begin{bmatrix}\check{\mathbf{P}}(t_k, t_k) & \check{\mathbf{P}}(t_k, t_{k+1}) \\ \check{\mathbf{P}}(t_{k+1}, t_k) & \check{\mathbf{P}}(t_{k+1}, t_{k+1})\end{bmatrix}\right)\begin{bmatrix}\mathbf{\Lambda}(\tau)^T \\ \mathbf{\Psi}(\tau)^T\end{bmatrix}.\end{aligned} \tag{53}$$

Now instead of using all the terms of the prior during the interpolation step, only the terms at time k and time $k+1$ are used. It should also be noted that the interpolation scheme itself is implicit in the choice of prior, it is not determined in an ad hoc manner. Each of (52) and (53) can be evaluated in constant time. Therefore solving for all the query times is $O(J)$ which is a significant improvement over $O(K^2J)$.

E. Non-linear Measurement Model

The case with a non-linear measurement follows a similar procedure. We start with an initial guess for the trajectory at the measurement times, $\bar{\mathbf{x}}$, and then solve for the optimal perturbation, $\delta\mathbf{x}$ using the measurement model linearized about the current best guess. If we let $\mathbf{x} = \bar{\mathbf{x}} + \delta\mathbf{x}$, we can write the joint likelihood between the state perturbation and the measurements at the measurement times as:

$$p\left(\begin{bmatrix}\delta\mathbf{x} \\ \mathbf{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\check{\mathbf{x}} - \bar{\mathbf{x}} \\ \mathbf{g} + \mathbf{C}(\check{\mathbf{x}} - \bar{\mathbf{x}})\end{bmatrix}, \begin{bmatrix}\check{\mathbf{P}} & \check{\mathbf{P}}\mathbf{C}^T \\ \mathbf{C}\check{\mathbf{P}}^T & \mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R}\end{bmatrix}\right) \tag{54}$$

where

$$\delta\mathbf{x} = \begin{bmatrix}\delta\mathbf{x}(t_0) \\ \vdots \\ \delta\mathbf{x}(t_K)\end{bmatrix}, \bar{\mathbf{x}} = \begin{bmatrix}\bar{\mathbf{x}}(t_0) \\ \vdots \\ \bar{\mathbf{x}}(t_K)\end{bmatrix}, \check{\mathbf{x}} = \begin{bmatrix}\check{\mathbf{x}}(t_0) \\ \vdots \\ \check{\mathbf{x}}(t_K)\end{bmatrix},$$

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

$$\mathbf{R} = \text{diag}(\mathbf{R}_1, ..., \mathbf{R}_K),\ \check{\mathbf{P}} = [\check{\mathbf{P}}(t_i, t_j)]_{ij}.$$

We can then write:

$$p(\delta\mathbf{x}|\mathbf{y}) = \mathcal{N}\Bigg((\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C})^{-1}\big(\check{\mathbf{P}}^{-1}(\check{\mathbf{x}} - \bar{\mathbf{x}}) + \mathbf{C}^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{g})\big), (\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C})^{-1}\Bigg). \quad (55)$$

Applying the SMW identity, we can write the following linear system for $\delta\mathbf{x}$:

$$(\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C})\delta\mathbf{x}^\star = \check{\mathbf{P}}^{-1}(\check{\mathbf{x}} - \bar{\mathbf{x}}) + \mathbf{C}^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{g}). \quad (56)$$

We can plug our definitions for the priors defined in the previous section into this equation:

$$\overbrace{(\mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C})}^{\text{block tridiagonal}}\delta\mathbf{x}^\star = \mathbf{A}^{-T}\mathbf{Q}^{-1}(\mathbf{v} - \mathbf{A}^{-1}\bar{\mathbf{x}}) + \mathbf{C}^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{g}). \quad (57)$$

Because of our choice of prior the left hand side is block tridiagonal and the entire system can be sovled in $O(K)$ time as is the case with a linear measurement model. We solve (57) for $\delta\mathbf{x}^\star$ at each iteration, and update the guess according to $\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} + \delta\mathbf{x}^\star$ and repeat until convergence. Once the state at the measurement times has been solved for in this manner, the state and covariance at additional query times can be interpolated using same equations as in the linear measurement case, (52) and (53).

F. White Noise on Acceleration Prior

We can now consider a specific example of a prior that will be used in the mobile robot examples. If we take:

$$\ddot{\mathbf{p}}(t) = \mathbf{w}(t) \quad (58)$$

where $\mathbf{p}(t)$ corresponds to position and

$$\mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_c\delta(t - t')). \quad (59)$$

This can be written in the form presented above:

$$\dot{\mathbf{x}}(t) = \boldsymbol{A}\mathbf{x}(t) + \mathbf{v}(t) + \boldsymbol{L}\mathbf{w}(t) \quad (60)$$

with

$$\mathbf{x}(t) = \begin{bmatrix}\mathbf{p}(t)\\ \dot{\mathbf{p}}(t)\end{bmatrix},\ \boldsymbol{A} = \begin{bmatrix}\mathbf{0} & \mathbf{1}\\ \mathbf{0} & \mathbf{0}\end{bmatrix},\ \mathbf{v}(t) = \mathbf{0},\ \boldsymbol{L} = \begin{bmatrix}\mathbf{0}\\ \mathbf{1}\end{bmatrix} \quad (61)$$

We can then compute the transition function:

$$\exp(\boldsymbol{A}\Delta t) = \mathbf{1} + \boldsymbol{A}\Delta t + \frac{1}{2}\boldsymbol{A}^2\Delta t^2 + ... = \begin{bmatrix}\mathbf{1} & \Delta t\mathbf{1}\\ \mathbf{0} & \mathbf{1}\end{bmatrix} \quad (62)$$

so

$$\boldsymbol{\Phi}(t_k, t_{k-1}) = \begin{bmatrix}\mathbf{1} & \Delta t_{k:k-1}\mathbf{1}\\ \mathbf{0} & \mathbf{1}\end{bmatrix}. \quad (63)$$

As well, we can compute $\mathbf{Q}_k$ and $\mathbf{Q}_k^{-1}$:

$$\mathbf{Q}_k = \int_{t_{k-1}}^{t_k} \boldsymbol{\Phi}(t_k, s)\boldsymbol{L}\mathbf{Q}_c\boldsymbol{L}^T\boldsymbol{\Phi}(t_k, s)^T ds \quad (64)$$

$$= \int_{t_{k-1}}^{t_k} \begin{bmatrix}\mathbf{1} & \Delta t_{k:s}\mathbf{1}\\ \mathbf{0} & \mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{0}\\ \mathbf{1}\end{bmatrix}\boldsymbol{Q}_c\begin{bmatrix}\mathbf{0} & \mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{1} & \mathbf{0}\\ \Delta t_{k:s} & \mathbf{1}\end{bmatrix} ds \quad (65)$$

$$= \begin{bmatrix}\frac{1}{3}\Delta t_{k:k-1}^3\mathbf{Q}_c & \frac{1}{2}\Delta t_{k:k-1}^2\mathbf{Q}_c\\ \frac{1}{2}\Delta t_{k:k-1}^2\mathbf{Q}_c & \Delta t_{k:k-1}\mathbf{Q}_c\end{bmatrix} \quad (66)$$

$$\mathbf{Q}_k^{-1} = \begin{bmatrix}12\Delta t_{k:k-1}^{-3}\mathbf{Q}_c^{-1} & -6\Delta t_{k:k-1}^{-2}\mathbf{Q}^{-1}\\ -6\Delta t_{k:k-1}^{-2}\mathbf{Q}_c^{-1} & 4\Delta t_{k:k-1}^{-1}\mathbf{Q}^{-1}\end{bmatrix} \quad (67)$$

IV. Mobile Robot Examples

A. Dataset 1

This first example is a linear, one-dimensional problem. The dataset was generated by a mobile robot equipped with a laser rangefinder driving back and forth between two fixed rails. A large cylinder was placed at the end of the rails. Range measurements from the robot to the cylinder at a rate of 10 Hz are included in the dataset. Ground truth position data was provided by a Vicon motion capture system also at 10 Hz. We will use the prior presented in Section III-F, where $\mathbf{p}(t) = x(t)$ and thus $\dot{\mathbf{p}}(t) = \dot{x}(t)$. The measurement model is:

$$y_k := x_c - r_k = \begin{bmatrix}1 & 0\end{bmatrix}\mathbf{x}_k + n_k \quad (68)$$

where x_c is the position of the cylinder's centre and r_k is the range measurement.

Figure 1 shows the case when measurements are received at asynchronous times as indicated by the vertical grey lines. The position in the x direction is plotted in the upper graph and the velocity in the x direction is plotted in the lower graph. The state was initially solved for at these times and is shown by the red markers, with a hollow circle indicating the mean and the smaller, filled circles indicating the $\pm 3\sigma$ uncertainty envelope. Interpolation was then conducted at a rate of 10Hz. The state and uncertainty envelope at the interpolated times is shown with the blue markers. In the position graph, the ground truth is also shown with a green line. We can see that the interpolated states smoothly connect the states at the measurement times using the interpolation scheme inherent in the choice of prior.

Figure 2 shows the case where there is a 20 second measurement dropout between 70s to 90s. The upper plot shows position in the x direction versus time and the lower plot shows velocity in the x direction versus time. Up until 70s and after 90s, measurements are received at a rate of 10Hz. The state was first solved at these measurement times shown in red. Interpolation was then used to solve for the state at a rate of 10Hz during the measurement dropout as shown in blue. Due to the choice of prior, only the states at time 70s and time 90s are used for all the interpolations. None of the earlier states before 70s or later states after 90s are used. This is the result of the Markovian property of the prior. We can also see that the uncertainty envelope around

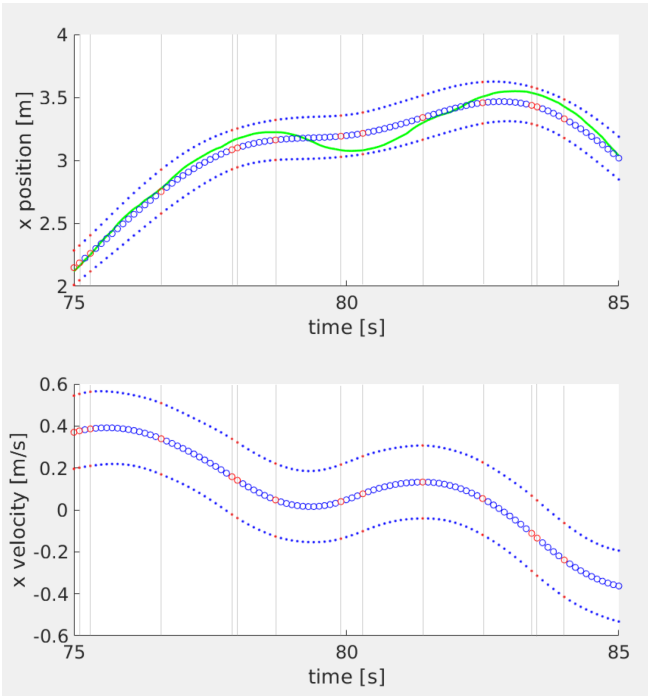

Fig. 1: Measurements are received at asynchronous times indicated by the vertical grey lines. Interpolation occurs at a rate of 10Hz as shown by the hollow blue dots for the mean and the smaller filled dots for thee uncertainty envelope. The green line show the ground truth position.

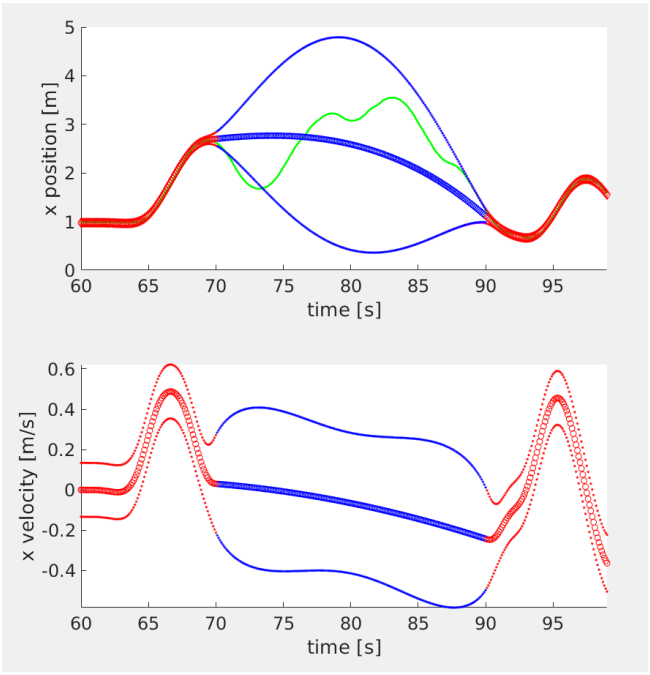

Fig. 2: Measurement dropout between 70s and 90s. Interpolation occurs at a rate of 10Hz during the dropout as shown in blue. The green line shows the ground truth position.

the position increases the further the time is from the two endpoints. The ground truth position is shown in green.

B. Dataset 2

The second example is a non-linear 2 dimensional problem. This dataset was generated by a robot travelling through a forest of 17 plastic tubes and is the same dataset as used in [4]. We want to estimate the robot's position in terms of x and y and its orientation, θ. Using a laser range finder mounted on the robot, range and bearing measurements to each of the cylindrical landmarks were recorded at a rate of 10Hz. Ground truth position and orientation data was provided by a Vicon motion capture system at 10Hz. Again we can use the prior presented in Section III-F, but for this example $\mathbf{p}(t) = [x(t)\ y(t)\ \theta(t)]^T$ and $\dot{\mathbf{p}}(t) = [\dot{x}(t)\ \dot{y}(t)\ \dot{\theta(t)}]^T$. The measurement model is:

$$\mathbf{y}_k^l = \begin{bmatrix} r_k^l \\ \psi_k^l \end{bmatrix} = \begin{bmatrix} R \\ B \end{bmatrix} + \mathbf{n}_k^l \tag{69}$$

$$R = \sqrt{(x_l - x_k - d\cos\theta_k)^2 + (y_l - y_k - d\sin\theta_k)^2} \tag{70}$$

$$B = \operatorname{atan2}(y_l - y_k - d\sin\theta_k, x_l - x_k - d\cos\theta_k) - \theta_k \tag{71}$$

where (r_k^l, ϕ_k^l) is the range/bearing to landmark l, (x_l, y_l) is the position of the center of landmark l, and d is the distance between the robot center and where the laser rangefinder was mounted.

The process outlined in Section III-E for non-linear measurement models was used. That is, the states at the measurement times were solved for by iterating (57) until convergence, and then the standard interpolation equations (52) and (53) were used to solve for the state at additional interest times.

As in the first example, Figure 3 shows the case where measurements are received at asynchronous times indicated by the vertical grey lines and Figure 4 shows the case with a 20 second measurement dropout from 105s to 125s. In both cases the state was interpolated at a rate of 10Hz and is shown in blue. Since the state is now six dimensional, $\mathbf{x} = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}]^T$, there are six plots instead of two. As with the first example, the interpolated states result in a smooth trajectory between measurement times and the uncertainty envelope for the position variables increases the further the time is from the closest measurements.

V. Conclusions

This project has demonstrated how to perform batch continuous time trajectory estimation through the lens of GP regression. This is a two step procedure. First, the state and covariance at the measurement times is solved for and then interpolation is used to solve for the state and covariance at additional query times. Selecting a GP prior generated from a LTV SDE means the inverse kernel matrix is block tridiagonal and the entire trajectory can be solved for in $O(K + J)$ were K is the number of measurement times and J is the number of additional query times. This was then performed on two mobile robot datasets. The first being a linear, one-dimensional problem, and the second being a non-linear, two-dimensional problem. For each dataset

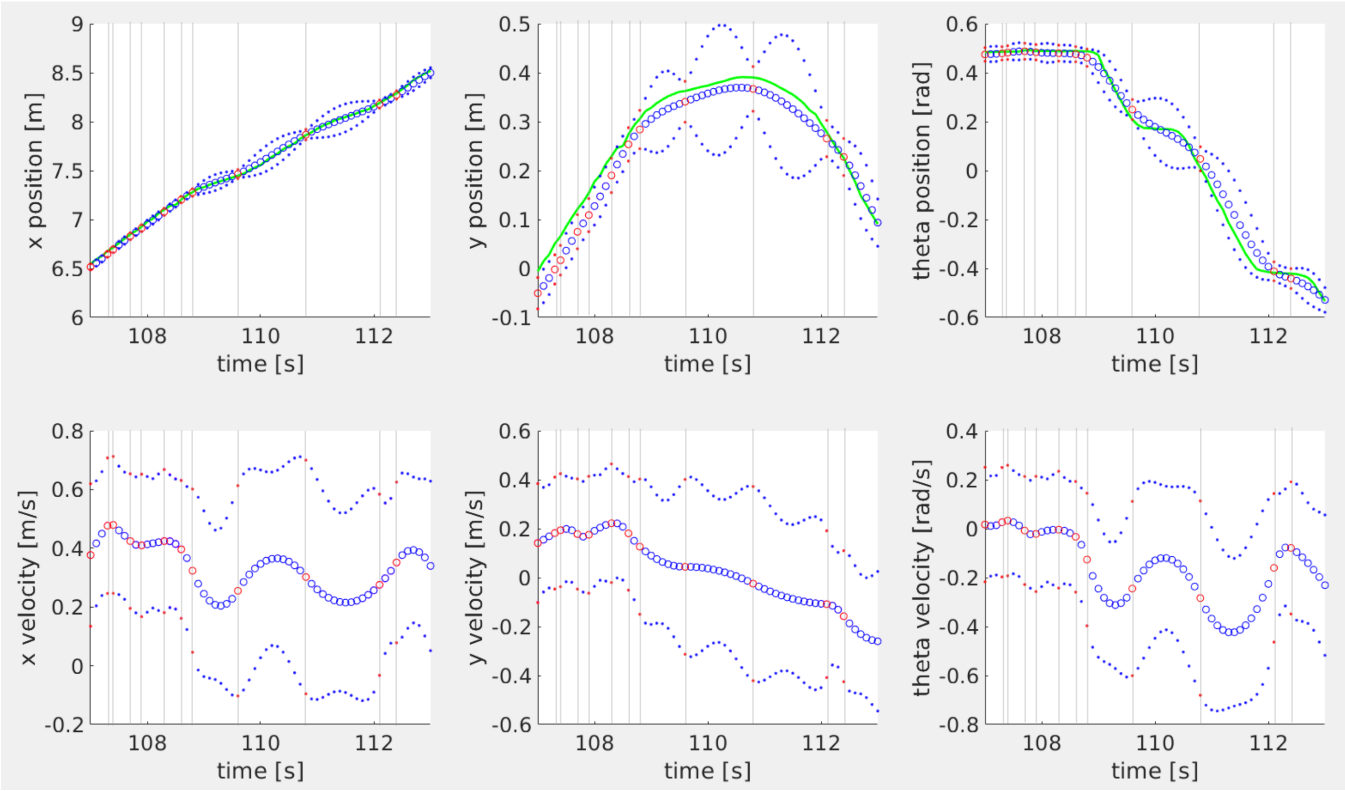

Fig. 3: Measurements are received at asynchronous times indicated by the vertical grey lines. THe state was interpolated at 10Hz with the hollow blue markers indicating the mean and the smaller, solid markers showing the $\pm 3\sigma$ uncertainty envelope. The green line shows the ground truth position data.

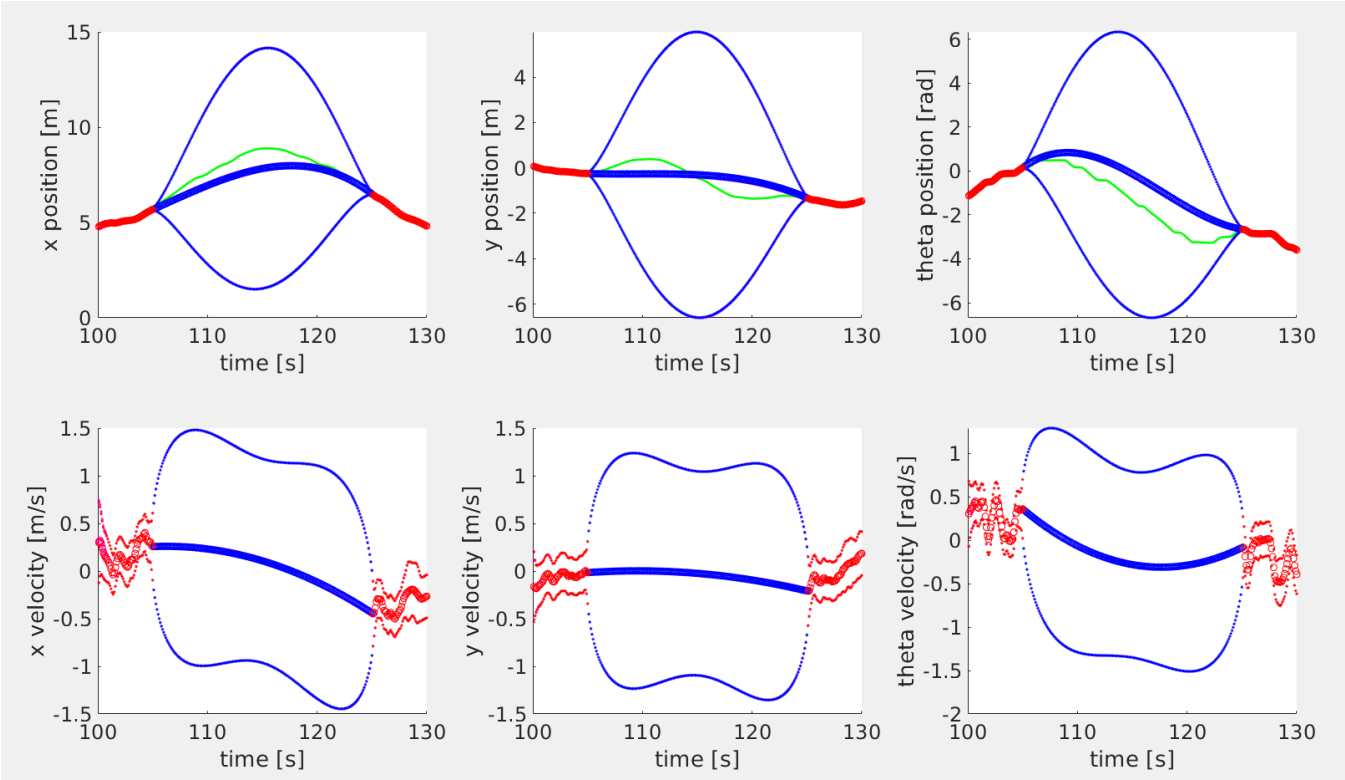

Fig. 4: Measurement dropout between 70s and 90s. Interpolation occurs at a rate of 10Hz during the dropout as shown in blue. The green line shows the ground truth position.

interpolation was conducted at a rate of 10Hz for a case with asynchronous measurements and a case of prolonged measurement dropout. GP regression allowed both the state and the covariance to be queried efficiently resulting in a smooth continuous representation.

REFERENCES

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

[2] C E Rasmussen and C K I Williams. Gaussian Processes for Machine Learning. MIT PRess, Cambridge, MS, 2006.

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

[4] T D Barfoot, C H Tong, and S Sarkka, "Batch Continuous-Time Trajectory Estimation as Exactly Sparse Gaussian Process Regression", RSS, 2014.

[5] S.Anderson and T D Barfoot , "Full STEAM Ahead: Exactly Sparse Gaussian Process Regression for Batch Continuous-Time Trajectory Estimation on SE(3)", IROS, 2015.

[6] S Anderson, T D Barfoot, C H Tong, and S Sarkka, "Batch Nonlinear Continuous-Time Trajectory Estimation as Exactly Sparse Gaussian Process Regression", RSS, 2015.

[7] Chapter 3.4 "Batch Continuous-Time Estimation" in Barfoot T D. State Estimation for Robotics. Cambridge University Press, 2017