Time Dependence in Kalman Filter Tuning

Zhaozhong Chen, Christoffer Heckman
Department of Computer Science
University of Colorado Boulder
430 UCB
Boulder, CO 80309
email: Christoffer.Heckman@colorado.edu

Simon Julier
Department of Computer Science
University College London
66–72 Gower Street,
London WC1E 6BT, UK
email: s.julier@ucl.ac.uk

Nisar Ahmed
Smead Aerospace Engineering Sciences
University of Colorado Boulder
429 UCB
Boulder, CO 80309
email: Nisar.Ahmed@colorado.edu

***Abstract*—In this paper, we propose an approach to address the problems with ambiguity in tuning the process and observation noises for a discrete-time linear Kalman filter. Conventional approaches to tuning (e.g. using normalized estimation error squared and covariance minimization) compute empirical measures of filter performance and the parameter are selected manually or selected using some kind of optimization algorithm to maximize these measures of performance. However, there are two challenges with this approach. First, in theory, many of these measures do not guarantee a unique solution due to observability issues. Second, in practice, empirically computed statistical quantities can be very noisy due to a finite number of samples. We propose a method to overcome these limitations. Our method has two main parts to it. The first is to ensure that the tuning problem has a single unique solution. We achieve this by simultaneously tuning the filter over multiple different prediction intervals. Although this yields a unique solution, practical issues (such as sampling noise) mean that it cannot be directly applied. Therefore, we use Bayesian Optimization. This technique handles noisy data and the local minima that it introduces. We demonstrate our results in a reference example and demonstrate that we are able to obtain good results. We share the source code for the benefit of the community[1].**

I. Introduction

State estimation through Kalman filters consist of two main steps: *state prediction* followed by a *measurement update*, both predicated on models of the system. The state prediction step uses a process model to predict how the state evolves over time. The measurement update step uses an observation model to relate a measured quantity to the state estimate. Since both the process and observation models are imperfect, errors in these models are treated as random noise terms that are injected into the system. Most designs assume the noise in these systems is white, zero mean and uncorrelated. As a result, filter tuning consists of choosing the values of the process and observation noise covariances, thereby fully defining the noise distribution.

Given the critical role that tuning plays in the performance of these algorithms, multiple techniques for tuning filters have been developed [1], [2], [3]. Perhaps the simplest approach is to use a two-stage divide-and-conquer strategy. In the first stage, the observation covariance is estimated by operating the system in lab conditions and monitoring the sensor noise characteristics. In the second stage, the observation covariance is held fixed, and the process noise covariance is determined. Since the process noises contain information about the state disturbances and dynamic model uncertainties, which often cannot be reproduced in lab settings, the covariance is often chosen by collecting data from an operational domain and quantifying the quality of the estimates. Typically a performance cost is assigned, and the process noise covariance is adjusted to minimize the value of that cost.

Other approaches include 'black box' auto-tuning methods [4], [5], [6], which construct a cost function to be minimized based on properties of the state or statistical principles regarding estimates produced. We demonstrate that, even in simple examples, these methods do not guarantee convergence to a unique optimum, and frequently converge to the incorrect optimum. We also shed light on the relationship between noise parameter identifiability and use of consistency metrics as fitness measures for auto-tuning methods, particularly to understand how mismatches between the filter-assumed and true system noise parameters impacts search algorithm convergence. Novel solutions to these issues are presented via measurement and process noise perturbation strategies, and demonstrated on reference examples via Bayesian optimization.

II. Prelminaries

A. Discrete and Continuous Time Systems

Our approach depends upon adjusting the prediction interval in the Kalman filter. Therefore, it is important to understand the relationship between the discrete and continuous time systems. The state of the system at time t is $\mathbf{x}_t$. The system is described by continuous time process model and observation models,

$$\begin{aligned} \dot{\mathbf{x}}_t &= \mathbf{A}\mathbf{x}_t + \mathbf{G}\mathbf{u}_t + \mathbf{\Gamma}\mathbf{v}_t, \\ \mathbf{z}_t &= \mathbf{H}\mathbf{x}_t + \mathbf{w}_t, \end{aligned} \tag{1}$$

where $\mathbf{u}_t$ is the control input, the process noise is the additive white process $\mathbf{v}_t$ with intensity $\mathbf{V}$, and the measurement noise is an additive white noise process $\mathbf{w}_t$ with continuous time intensity $\mathbf{W}$. In discrete time, the state at timstep k is $\mathbf{x}_k$. The system evolution from timestep $k-1$ to k is

$$\mathbf{x}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_k + \mathbf{v}_k, \tag{2}$$

[1]https://github.com/arpg/kf_bayesopt

where $\mathbf{u}_k$ is the control input and $\mathbf{v}_k$ is the process noise, which is assumed to be zero mean and independent with covariance $\mathbf{Q}_k$. The observation model is

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{w}_k, \tag{3}$$

where $\mathbf{w}_k$ is the observation noise.

The discrete-time system is derived from the continuous time system using techniques such as Van Loan's method [7],

$$\begin{aligned} \mathbf{F}_k &= e^{\mathbf{A}_t \Delta t}, \quad \mathbf{B}_k = \int_0^{\Delta t} e^{\mathbf{A}_t m} \mathrm{d}m, \\ \mathbf{Q}_k &= \int_0^{\Delta t} e^{\mathbf{A}_t m} \mathbf{\Gamma V \Gamma}^T e^{\mathbf{A}^T m} \mathrm{d}m. \end{aligned} \tag{4}$$

If the observation is from an integrating sensor, the discrete time observation vector is $\mathbf{R}_k = \mathbf{W}/\Delta t$ [7]. For a non-integrating sensor $\mathbf{R}_k = \mathbf{R}_t$, i.e. it is independent of Δt.

B. Kalman Filter

A Kalman filter can be used to find the optimal state estimate [8], via a two stage process of prediction followed by measurement update. The prediction is

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k \tag{5}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^\top + \mathbf{Q}_k \tag{6}$$

while the update is

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{W}_k \mathbf{e}_{\mathbf{z},k}, \tag{7}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{W}_k \mathbf{S}_{k|k-1} \mathbf{W}_k^\top, \tag{8}$$

$$\mathbf{S}_{k|k-1} = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k \tag{9}$$

$$\mathbf{W}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \mathbf{S}_{k|k-1}^{-1} \tag{10}$$

where $\mathbf{e}_{\mathbf{z},k} = \hat{\mathbf{z}}_{k|k-1} - \mathbf{z}_k$ is the innovation vector.

One important issue with this method is tuning: given $\mathbf{F}_k$ and $\mathbf{H}_k$, the process and observation noise processes $\mathbf{V}$ and $\mathbf{W}$ must be determined. This is normally achieved by exploring different values of $\mathbf{V}$ and $\mathbf{W}$ and applying a fitness measure.

C. Parameter Fitness and Tuning

Two widely used measures for fitness are the *normalized estimation error squared (NEES)* and the *normalized innovation error squared (NIS)*. The NEES is computed from

$$\epsilon_{\mathbf{x},k} = \mathbf{e}_{\mathbf{x},k}^T \mathbf{P}_{k|k}^{-1} \mathbf{e}_{\mathbf{x},k}, \tag{11}$$

where $\mathbf{e}_{\mathbf{x},k} = \hat{\mathbf{x}}_{k|k} - \mathbf{x}_k$. Because we need to know $\mathbf{x}_k$, NEES requires a groundtruth measurement of the system state. The NIS, on the other hand, only depends on the observation sequence and does not require knowledge of groundtruth. It is computed from

$$\epsilon_{\mathbf{z},k} = \mathbf{e}_{\mathbf{z},k}^T \mathbf{S}_{k|k-1}^{-1} \mathbf{e}_{\mathbf{z},k}, \tag{12}$$

where $\mathbf{e}_{\mathbf{z},k}$ is the innovation vector. If the filter is statistically consistent, it can be shown that the expected values of the NEES and the NIS are [9]

$$\mathrm{E}\left[\epsilon_{\mathbf{x},k}\right] \approx n_{\mathbf{x}}, \quad \mathrm{E}\left[\epsilon_{\mathbf{z},k}\right] \approx n_{\mathbf{z}}, \tag{13}$$

Although the $\epsilon_{\mathbf{z},k}$ and $\epsilon_{\mathbf{x},k}$ are widely used, they have the property that they are bounded from below (by 0) but not from above. This naturally introduces a bias or asymmetry in the measure. To overcome this, we use a log measure instead:

$$\begin{aligned} J_{NEES} &= \left| \log \left(\frac{\sum_{k=1}^T \bar{\epsilon}_{\mathbf{x},k}/T}{n_{\mathbf{x}}} \right) \right|, \\ \bar{\epsilon}_{\mathbf{x},k} &= \frac{1}{N} \sum_{i=1}^N \epsilon_{\mathbf{x},k}^i. \end{aligned} \tag{14}$$

where N is the number of Monte Carlo runs and T is the period of sampling. J_{NEES} is not bounded. However, when the filter is consistent, $J_{NEES} = 0$.

D. Related Work

Though the problem of Kalman filter tuning has been widely studied, it remains a challenging open problem for which no single best technique exists [10], [11]. These include: maximum likelihood and Bayesian inference [12], least squares for data processed via Kalman smoothing [13], and auto-/cross-correlation analysis [14]. These methods are theoretically advantageous for well-defined linear systems where noise models have known structure, and are useful in online settings. Yet, they can also suffer from numerical stability and implementation issues, making them harder to use. Moreover, they are difficult to generalize for non-linear filters, e.g. since the optimal set of noise parameters in linearization-based filters can vary significantly with system state and time [15].

The family of 'black box' optimization approaches considered here offers a computationally attractive and flexible alternative, whereby filter-assumed noise parameters are adjusted via search algorithms to maximize a set of filter output fitness measures, which are assessed on candidate filter runs against truth model simulations and/or recorded measurement logs. In addition to being highly parallelizable in most cases, black box optimization can readily leverage useful but complex stochastic fitness measures that do not yield tractable 'well-behaved' expressions for objective functions and gradients with respect to unknown noise parameters.

The defining features of black box methods are the choice of filter output fitness measure and search algorithm. Powell [3] proposed using a mean weighted filter state error norm as a fitness measure to be minimized via downhill simplex search. In earlier work, Oshman and Shaviv [16] presented a fitness measure based on chi-square tests for NEES consistency (evaluated using truth model simulations) to tune process noise covariance parameters via genetic algorithms. More recently, [4] developed a technique using Bayesian optimization search and generalized filter output fitness measures based on NIS consistency tests with real/logged data, as well as NEES consistency tests with truth model simulation runs. Other metrics closely related to NIS consistency assessment [17], [18], [19] could also be adapted as fitness measures.

While search methods like genetic algorithms and Bayesian optimization can explore the global parameter space, the observability (i.e. identifiability) of noise parameters relative

to estimation error and consistency-based fitness metrics is not well understood. For instance, [16] noted that their approach generally converged towards an infinite basin of feasible parameters which all satisfy the NEES consistency criterion, without necessarily minimizing the resulting steady state $\mathbf{P}$. As such, [16] also proposed a fitness measure to minimize filter covariance, while ensuring NEES consistency within some tolerance. However, the general conditions for convergence toward unique or multiple/infinite solutions remain unclear. Ref. [10] addresses the observability of $\mathbf{Q}$ and $\mathbf{R}$ in discrete time Gauss-Markov linear systems by deriving a matrix rank test. This is theoretically useful for assessing uniqueness of time invariant $\mathbf{Q}$ and $\mathbf{R}$ parameters, provided the hypothesized matrix structures match the true system behavior. Otherwise, the correctness and sensitivity of the matrix structures and values cannot be readily deduced.

III. The Problem of Observability

The non-uniqueness (non-observability) of noise parameters via consistency-based fitness metrics is a key problem for black box tuning approaches. We illustrate this using the following linear example. We seek to tune the process and observation noise processes for a 1D particle. The particle's state is its position and velocity,

$$\mathbf{x}_t = \begin{bmatrix} x_t & \dot{x}_t \end{bmatrix}^\top.$$

It moves with a constant velocity with noise injected into the acceleration. The particle's position is periodically observed by a non-integrating sensor. Therefore, the continuous time equations are

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad \mathbf{\Gamma} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Van Loan's method yields the familiar discrete-time equations

$$\begin{aligned} \mathbf{F}_k &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \quad \mathbf{B}_k = \begin{bmatrix} \Delta t^2/2 \\ \Delta t \end{bmatrix}, \quad \mathbf{H}_k = \begin{bmatrix} 1 & 0 \end{bmatrix}, \\ \mathbf{Q}_k &= \mathbf{V} \begin{bmatrix} \Delta t^3/3 & \Delta t^2/2 \\ \Delta t^2/2 & \Delta t \end{bmatrix}, \quad \mathbf{R}_k = \mathbf{W}. \end{aligned} \tag{15}$$

Suppose the actual (groundtruth) process and observation noise intensities are $\mathbf{V}^a = 1$ and $\mathbf{W}^a = 0.1$. However, these values are not known, and a black-box tuning algorithm will try candidate values for $\mathbf{V}$ and $\mathbf{W}$. In the appendix, we derive the expressions to compute $J_{NEES}(\mathbf{V}, \mathbf{W}, \mathbf{V}^a, \mathbf{W}^a)$. Fig. 1 plots these values for different choices of $(\mathbf{V}, \mathbf{W})$. When $\mathbf{V} < \mathbf{V}^a$ and $\mathbf{W} < \mathbf{W}^a$ (bottom left), J_{NEES} is high because the filter is inconsistent. When $\mathbf{V} > \mathbf{V}^a$ and $\mathbf{W} > \mathbf{W}^a$ (top right), J_{NEES} is large again because the filter is conservative. The thick curved blue line shows where $J_{NEES} \approx 0$ and shows multiple solutions which appear consistent. The yellow curve is the set of samples of $(\mathbf{V}, \mathbf{W})$ for which $J_{NEES}(\mathbf{V}, \mathbf{W}, \mathbf{V}^a, \mathbf{W}^a) \in [-0.0025, 0.0025]$ ($\epsilon_{\mathbf{x},k}(\mathbf{V}, \mathbf{W}, \mathbf{V}^a, \mathbf{W}^a) \in [1.995, 2.005]$). We refer to this curve as the "NEES line." Fig. 2 plots the log determinants of $\mathbf{P}_{k-1|k-1}(\mathbf{V}, \mathbf{W})$ and $\mathbf{P}^a_{k|k-1}(\mathbf{V}, \mathbf{W}, \mathbf{V}^a, \mathbf{W}^a)$ along this curve. These results largely support Oshman and Shaviv [16]:

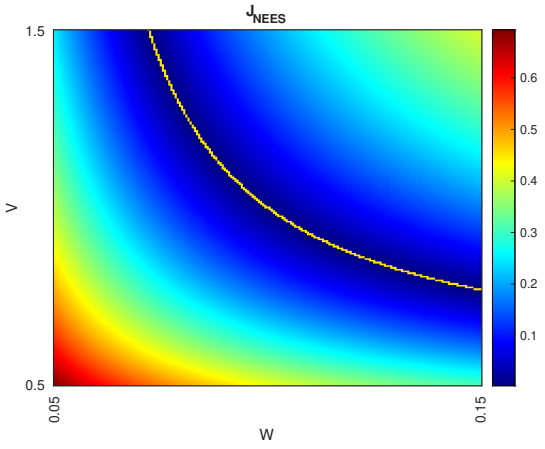

Fig. 1: Theoretically computed $J_{NEES}(\mathbf{V}, \mathbf{W}, \mathbf{V}^a, \mathbf{W}^a)$ for different values of $\mathbf{V}$ and $\mathbf{W}$.

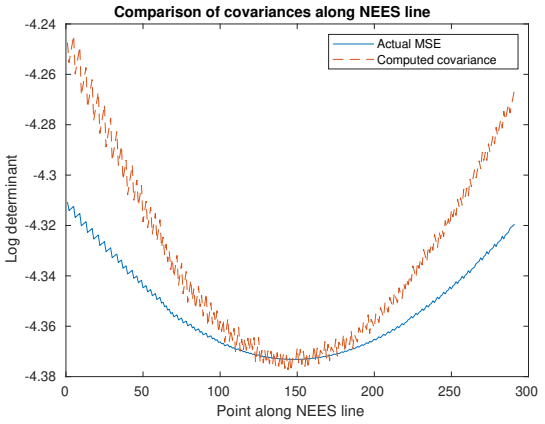

Fig. 2: Log determinant of the actual and computed covariance matrices along the NEES line. The jagged appearance is due to the quantization in the sampling. $\mathbf{V} = \mathbf{V}^a$ and $\mathbf{W} = \mathbf{W}^a$ at point 149.

there are multiple solutions which appear to be consistent with the NEES, and the optimal solution occurs near where the covariance is minimized. However, we see it is possible to choose values which are *slightly* inconsistent.

There are two implications for these results. The first is that, to compute the optimal solution, we had to derive closed form solutions for the NEES. This is possible in linear systems only by knowing the groundtruth noises, which are not available in practice, and for nonlinear systems is generally unachievable in closed form. Therefore, empirical techniques will have to be used. Second, tuning to incorrect noises means that the filter is not robust to changes in the configuration. For example, if the correct values for $\mathbf{V}$ and $\mathbf{W}$ are used, the filter should be consistent given any timestep length. Slight errors in these values no longer means this is true.

For example, consider the filter solution when $\mathbf{V} = 1.045, \mathbf{W} = 0.95$ which is around point 130 on Fig. 2. For $\Delta t = 0.1$ this gives a the value $J_{NEES} = 0.0018$

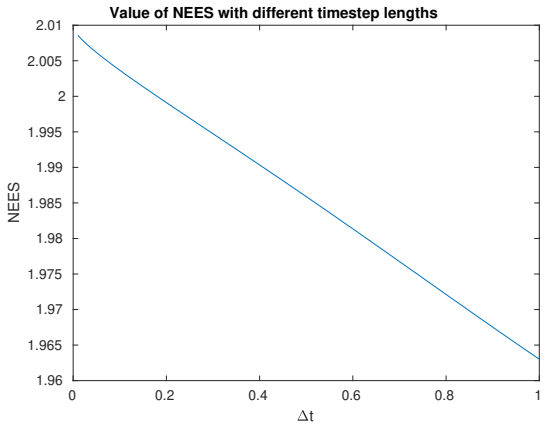

Fig. 3: $\epsilon_{\mathbf{x},k}$ for different values of Δt.

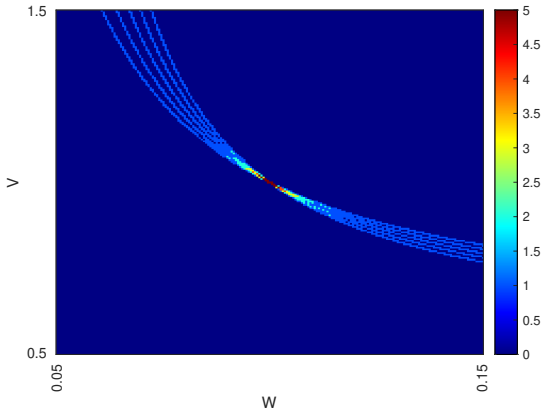

Fig. 4: Overlay of NEES curves with values of $\Delta t = [0.1, 0.2, 0.3, 0.4, 0.5]$. Each cell contains the count of the number of runs within which a NEES value of 2 is obtained.

($\epsilon_{\mathbf{x},k} = 2.0037$). Furthermore, if one computes the values of $\epsilon_{\mathbf{x},k}(\mathbf{V}, \mathbf{W}, \mathbf{V}^a, \mathbf{W}^a)$ using fixed values for noise intensities but varying Δt, there is a clear and significant change in the NEES for these various timestep lengths.

Our motivation is to find a way to expose the errors more clearly, since they can lead to suboptimal solutions in auto-tuning techniques. In Fig. 3, we compute $\epsilon_{\mathbf{x},k}(\mathbf{V}, \mathbf{W}, \mathbf{V}^a, \mathbf{W}^a)$ using fixed values for the noise intensities but varying Δt between $0.1\,\mathrm{s}$ and $1\,\mathrm{s}$. As can be seen, these results suggest that the impact of a tuning error becomes more significant if the filter timestep changes relative to the timestep used when tuning the original filter.

IV. Noise Tuning

A. The Effects of Noise Perturbations

The previous section demonstrated that the J_{NEES} values are ambiguous in supporting correct noise tuning. When coupled with minimising the covariance, the values can be found in theory; however, the differences can be small. The differences become apparent at long prediction intervals, which is computationally costly, and worse converges very slowly over lengthening intervals. However, this can suggest that one strategy is to use different timestep lengths and observe the effect on estimation statistics.

To motivate this, Fig. 4 shows the effect of computing over several different timesteps. For each timestep, van Loan's technique was used to construct the system and the NEES was calculated. As before, only the values close to 2 were kept. For each value of Δt a different NEES curve is generated. All of the curves intersect at the same point which is the groundtruth value of the intensity. This is hardly surprising. If the filter is tuned to the groundtruth values, it should generate the same NEES irrespective of the timestep length. However, it also suggests that the observability of the optimal tuning parameters can be influenced by timestep length.

The foregoing has been conducted purely using a theoretical analysis of NEES calculations. To test the effect of this, we used 200 Monte Carlo runs and computed J_{NEES} using (14). Figs. 5a and 5b plot the J_{NEES} values for $\Delta t = 0.1$ and $\Delta t = 0.5$ respectively. These show that, despite sampling noise, we see a very similar behaviour again with the curve being shifted and values along a ridge being very similar.

B. Cost Function for Optimization

The conclusion of the foregoing argument is that there is implicit dependence of J_{NEES} as a function of Δt. To our knowledge, this is not very well-explored in the literature. In auto-tuning Kalman filter algorithms, the J_{NEES} is typically evaluated conditioned upon a single value of Δt. Of course, the alternative, where Δt is allowed to vary as a parameter to J_{NEES}, results in a computationally expensive parameter search. Yet the extreme value and implicit function theorems imply that such a minimum exists somewhere between $\Delta t = (0, h)$ where h is "small," as is typical for numerical integration and required for local truncation error to be acceptably low, and as long as there are no discontinuities in $\mathbf{F}$ or $\mathbf{S}$.

To avoid the need for an expensive search, we choose a sample of Δt values and a logical operation in our search: for each pair $[\mathbf{V}, \mathbf{W}]$, $\mathbf{W} \in [0.01, 0.5]$, $\mathbf{V} \in [0.1, 5.0]$, groundtruth $\mathbf{V} = 1, \mathbf{W} = 0.1$ we calculate J_{NEES} using $\Delta t = 0.1$ and $\Delta t = 0.5$. Then, we only record the larger J_{NEES} and get another plot.The results are shown in Fig. 5a,5b,5c. Note the plots show $lg(J_{NEES})$ because, in this way, J_{NEES} smaller than 1 will be negative, its color is more clear. In Fig.5a, there is a blue curve shows the small J_{NEES}. The red arrow points out the minimum value, which is not around the groundtruth. in Figure 5b, the minimum is also not at the groundtruth. We find that the global minimum J_{NEES} is quite random when $\Delta t = 0.1$ or $\Delta t = 0.5$ or other single Δt. Thus, when we use an optimization algorithm to search the surface, the possible estimations can be quite random. However, this situation is different in case Figure 5c. The global minimum is always around [0.1,1]. It's obvious now the Δt influences the cost function distribution. It would be interesting to see the mapping between different Δt value and J_{NEES}, which is shown in Figure 6. It shows that when both $\mathbf{V}, \mathbf{W}$ are around the groundtruth value, J_{NEES} is small whatever the

Δt is. These experiments motivates us to tune the KF with different dt and find the solution that can give consistent J_{NEES}. The solution should be the close to the groundtruth. In our experiment, we found that find the solution that gives consistent J_{NEES} with only two different dt are sufficient.

V. Experiments

To investigate the effects of choosing multiple sample times, we apply a Bayesian optimization (BO) auto-tuning algorithm on two linear systems, namely: a 1D tracking problem and a 2D tracking problem. In both examples, the process and measurement noises parameters are optimized together. For the 1D tracking, one process and one measurement noise parameter are jointly optimized. For the 2D tracking problem, two process and two measurement noise parameters are optimized. We run two examples for the following purposes.

A. Bayesian optimization tuning

- **1D tracking**: For the 1D (particle) tracking system we introduced before, we can see the benefits of using multiple sample time during the optimization. We display the numerical optimization result and show the process of BO, from where we can see the exploration ability of the Bayes optimization.
- **2D tracking system**: In the 2D tracking system, we are going to optimize 4D parameters. i.e. 2 process noise parameters and 2 measurement noise parameters. We perform the χ^2 test to show that the filter is consistent.

We use our previous work's optimization process [4]. i.e. GPBO (Gaussian Process Bayesian Optimization). The difference is that for each set of the noise estimation, now we run the Kalman filter (N Monte Carlo simulations) with two sample time (Δt = 0.1 and $\Delta t = 0.5$). We pick the larger cost and feed it into the Bayesian optimization. The motivation is that we want the cost remain small with different sample time.

Results are compared from four auto-tuning strategies. The first one is the proposed GPBO algorithm with the J_{NEES} cost function. To assess the value of the multiple sample time strategy, we compare it to our previous approach, where we use $\Delta t = 0.1$ only. To further extend our previous work, we compare the GPBO with the Downhill Simplex (DS) algorithm. From Figure 5c, we can see that even the groundtruth is at the correct position, the cost along the blue curve is close to each other, which brings a challenge to the optimizer. We show that the GPBO can efficiently explore the cost surface and achieve better results than the Downhill Simplex algorithm. After optimization convergence of each method across 200 Monte Carlo runs, the following are evaluated to compare the resulting filter tuning solutions: the numerical value of the optimized noise parameters; filter dynamic consistency, i.e. the error between the groundtruth state and the estimation should be within a threshold σ; and BO surrogate model visualizations, to demonstrate the solution search process.

B. 1D tracking system

The Bayes Opt searching range for $\mathbf{V}$ is $[0.1, 5]$ and $\mathbf{W}$ is $[0.01, 0.5]$. Two sampling periods ($\Delta t = 0.1s$, $\Delta t = 0.5s$) were used and each Monte Carlo run was carried out for $T = 200\Delta t$ (this means that were either 2000 or 400 timesteps per filter). For the kernel function, the Matérn Kernel [20] with $\nu = 3$ and automatic relevance determination (ARD) was used. For remaining parameters such as the kernel mean, kernel hyperparmeter re-learn iteration number and the acquisition function optimization number, default values from the Bayesian optimization library [21] are used.

GPBO was performed 50 times to optimize $\mathbf{V}$ and $\mathbf{W}$. The results are shown in Table I. From the table we can see our optimization appears robust: the estimation variance is small and the mean is close to the groundtruth value, which is a significant improvement from our previous GPBO method. Note also that the estimation has a large variance owing to the simulations' stochasticity. The downhill simplex algorithm, as expected, can get trapped in different local minima because we initialize the sample at different points. Even with the multiple timestep strategy, the downhill simplex struggles to converge to the groundtruth. An effective optimizer must explore different regions of parameter space to find the global minima, a strength of BO. Figure 7 shows the convergence of the resulting GPBO surrogate function and the set of sampled v and w parameters across 200 iterations. From Figure 7, we can see as the number of iterations increases, GPBO explores increasingly around the local optimum. Finally, the optimal solution is found around $\mathbf{V} = 1$, $\mathbf{W} = 0.1$.

C. 2D tracking system

So far, we have only considered the motion of a 1D particle which required two scalar intensity values. However, our method directly extends to vector-valued intensity values. There, in this section we demonstrate the performance of the approach in a 2D tracking system, where the state is $\mathbf{x} = [x, y, \dot{x}, \dot{y}]^T$. We assume the same control input as in the previous systems, add white Gaussian process noise to $[\dot{x}, \dot{y}]$, and add white Gaussian measurement noise to position $[x, y]$. The discrete time system is

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0.5\Delta t^2 \\ 0.5\Delta t^2 \\ \Delta t \\ \Delta t \end{bmatrix}$$
$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^3}{3} V_0 & 0 & \frac{\Delta t^2}{2} V_0 & 0 \\ 0 & \frac{\Delta t^3}{3} V_1 & 0 & \frac{\Delta t^2}{2} V_1 \\ \frac{\Delta t^2}{2} V_0 & 0 & \Delta t V_0 & 0 \\ 0 & \frac{\Delta t^2}{2} V_1 & 0 & \Delta t V_1 \end{bmatrix} \tag{16}$$

We apply the same optimization methods as in the tracking 1D example for 50 independent trials. We need to increase the GPBO initial sample to 120 and the iteration to 300 since the dimension is higher.

For each optimization result of the algorithm, we apply it to the Kalman Filter again with 200 Monte Carlo runs

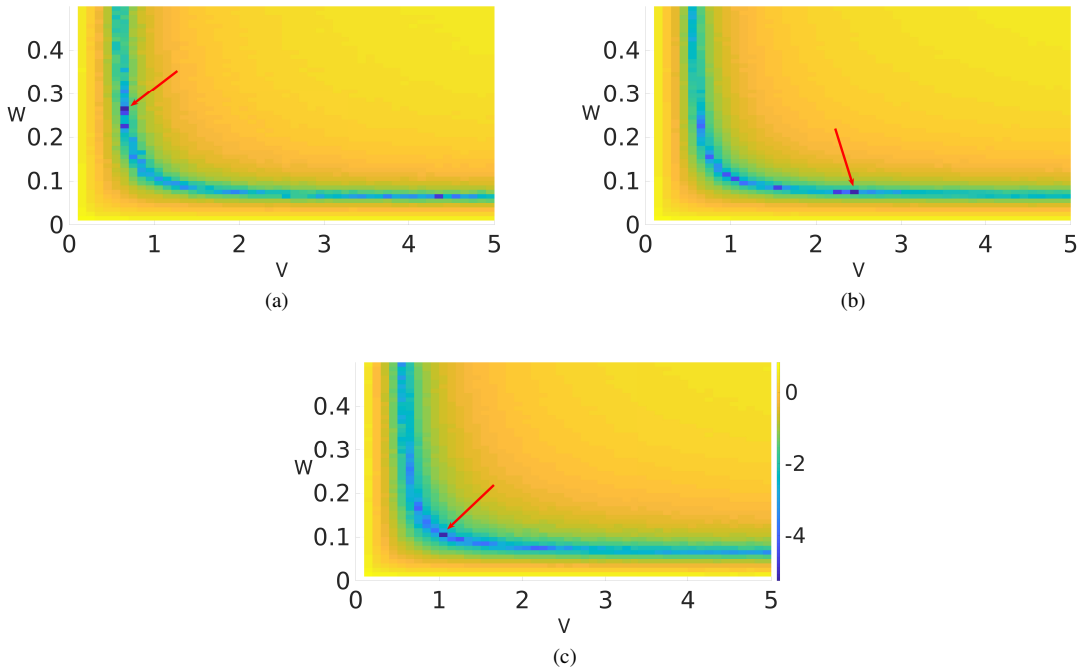

Fig. 5: In Figures (a), (b), (c), the colorbar shows the value of $lg(J_{NEES})$. In (a), $\Delta t = 0.1$; In (b), $\Delta t = 0.5$; In (c), we calculate J_{NEES} at both Δt for each $[\mathbf{V}, \mathbf{W}]$ pair but only pick the larger cost. The red arrow indicates where the J_{NEES} is the smallest in each plot. We can see that only in (c) we have a global minima around the groundtruth value $[\mathbf{V} = 1, \mathbf{W} = 0.1]$

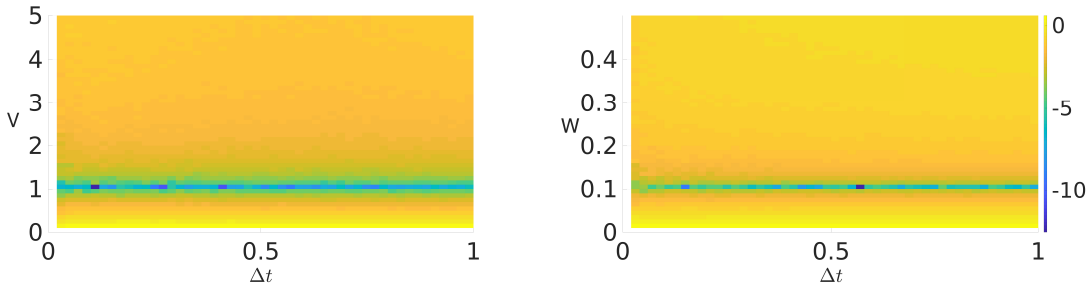

Fig. 6: Figure (a) fixes $\mathbf{W} = 0.1$ and plots $\mathbf{V}, \Delta t$ versus $lg(J_{NEES})$. Figure (a) fixes $\mathbf{V} = 1$ and plots $\mathbf{W}, \Delta t$ versus $lg(J_{NEES})$. We can see only when both $\mathbf{V}, \mathbf{W}$ are around the groundtruth value, J_{NEES} is small whatever the Δt is.

TABLE I: Tracking 1D Optimization result

	GPBO, $\Delta t = 0.1, 0.5$		GPBO, $\Delta t = 0.1$		DS $\Delta t = 0.1, 0.5$		DS $\Delta t = 0.1$		Groundtruth
	v	w	v	w	v	w	v	w	
Mean	**0.958**	**0.152**	1.682	0.296	0.602	0.182	0.317	0.145	w = 0.1
Variance	**0.115**	**0.010**	2.043	0.076	0.094	0.011	0.412	0.012	v = 1

and record the $\mathrm{E}[\bar{\epsilon}_{\mathbf{z},k}], \mathrm{E}[\bar{\epsilon}_{\mathbf{x},k}], \mathrm{E}[\bar{\epsilon}_{\mathbf{z},k}\bar{\epsilon}_{\mathbf{z},k}], \mathrm{E}[\bar{\epsilon}_{\mathbf{x},k}\bar{\epsilon}_{\mathbf{x},k}]$ for validation. We choose sample time $dt = 0.1$ to collect the data. Note that we don't draw the box plot of the downhill sample algorithm with a single sample time since its value is too large. It is hard to visualize other methods' box plots (will be like a line) if we draw it. As we can see, generally the proposed method has better NIS and covariance. However, we still hope them can be closer to the expectations, which brings questions to our future work. 1: If we use NIS based cost function, what the NEES value will be from the optimization result? 2: Is it possible to add the covariance into the cost function constraints so that we can "force" the optimization result has a better χ^2 test?

Finally, we perform the direct consistency check of the

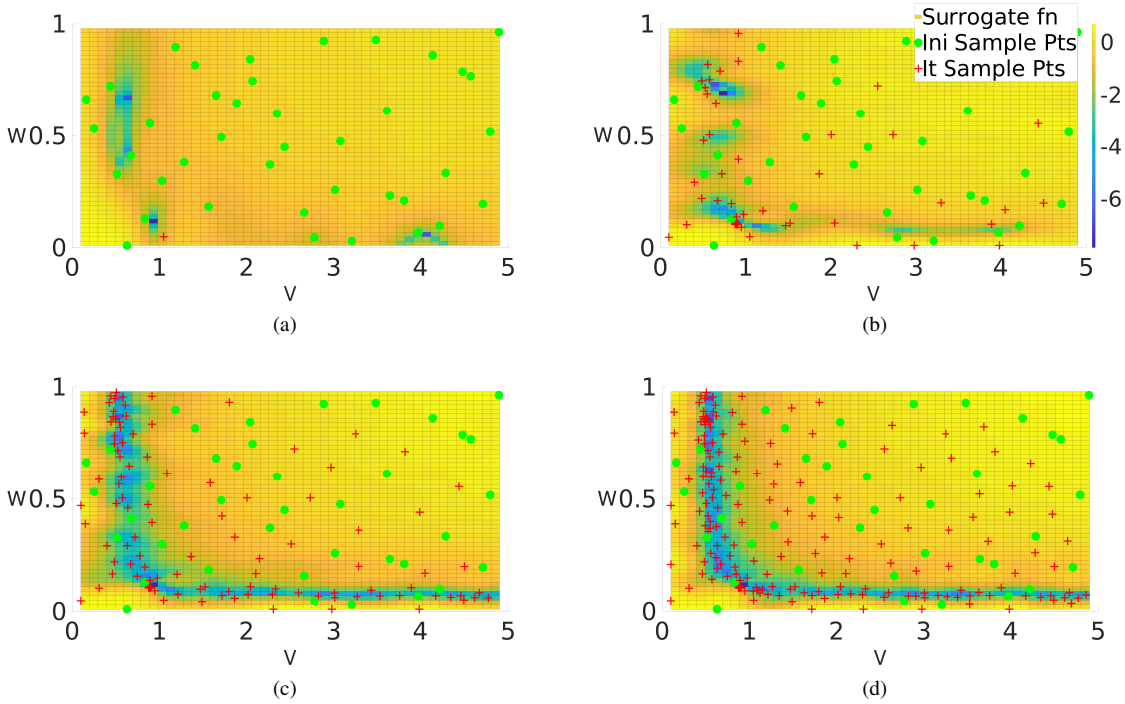

Fig. 7: GPBO surrogate model for J_{NEES} cost, showing initial random sample points (green dots) and best estimate (red crosses) infered by GPBO in different iterations. GPBO successfully explore the blue (possible low cost) area and the final surrogate model is similar to the real cost surface from Figure 5c. Finally, it finds the minimum around $v = 1$ and $w = 0.1$.

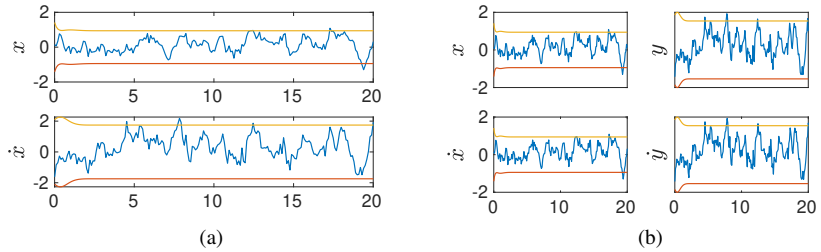

Fig. 8: Orange lines: 2σ bounds; blue line: error between the estimated states and the real states in KF's each step. If the system is consistent, around 95% error should be within 2σ range. (a) is from the estimation result of 1D tracking system. (c) is from 2D tracking system.

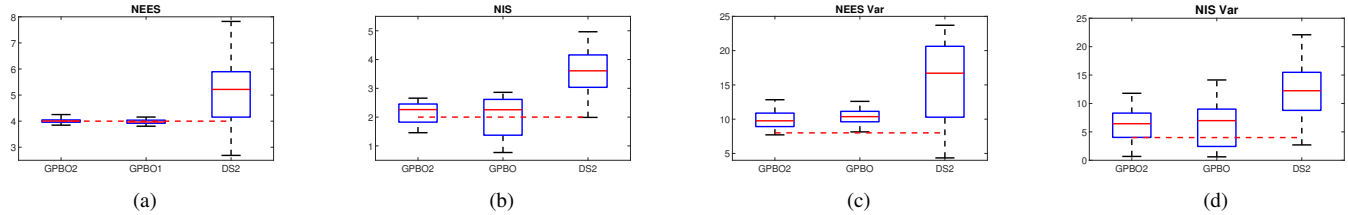

Fig. 9: For each method's 50 runs result, we apply them to the Kalman filter, record the $\mathrm{E}\left[\bar{\epsilon}_{\mathbf{z},k}\right], \mathrm{E}\left[\bar{\epsilon}_{\mathbf{x},k}\right], \mathrm{E}\left[\bar{\epsilon}_{\mathbf{z},k}\bar{\epsilon}_{\mathbf{z},k}\right], \mathrm{E}\left[\bar{\epsilon}_{\mathbf{x},k}\bar{\epsilon}_{\mathbf{x},k}\right]$ and plot the box plot. Red dash line: Expected value. GPBO2: GPBO with two sample time approach; GPBO1: GPBO with sample time at 0.1; DS2: Downhill Simplex with two sample time approach.

proposed method for both 1D and 2D system. We randomly choose one the optimization result and apply it to the Kalman filter. Then we plot each timestep's error and the 2σ boundary, where the $\sigma = \sqrt{\mathbf{P}_{k|k}}$. If the system is consistent, around 95%

error should be within 2σ range.

VI. Conclusion

We have demonstrated that there is implicit dependence of J_{NEES} on Δt, and that as a result, many auto-tuning algorithms face significant challenge short of running a search over multi-dimensional space for optimal noise parameters and their corresponding Δt. While it is true that around the groundtruth noise parameters, J_{NEES} will be small independent of what Δt is, we identify that for other guesses at noise parameters, the J_{NEES} is highly dependent on timestep choice. To address this, we propose a simple sampling procedure that appears to remedy this problem while allaying grievous increases in computational cost. Finally, we demonstrate this new approach on an auto-tuning algorithm for Kalman filter noise parameters. As future work, we believe a proof of this technique would be highly valuable. Furthermore, there exists an open investigation into the effectiveness of various statistical tests for significance in the mean and variance of the auto-tuning algorithms.

Appendix

In this appendix, we compute the expression to derive a closed-form solution for the NEES directly from the system equations. We assume that the system model equations $\mathbf{A}_t$, $\mathbf{G}_t$, $\mathbf{\Gamma}_t$ and $\mathbf{H}_t$ are correct. Only the noise intensities are unknown. For simplicity, we follow the work of Nishimura and Hellner and compute the NEES of the *predicted* covariance.

First consider the filter which has been tuned with the intensities $\mathbf{V}$ and $\mathbf{W}$. Using van Loan's method, we compute the discrete time process model together with the noise covariance matrices $\mathbf{Q}_k(\mathbf{V})$ and $\mathbf{R}_k(\mathbf{W})$, where we have included the intensities to emphasise the functional dependency. The filter will then predict the covariance history according to

$$\begin{aligned}\mathbf{P}_{k|k-1}(\mathbf{V},\mathbf{W}) = \mathbf{X}_k(\mathbf{V},\mathbf{W})\mathbf{P}_{k-1|k-1}(\mathbf{V},\mathbf{W})\mathbf{X}_k^\top(\mathbf{V},\mathbf{W}) \\ + \mathbf{K}_k(\mathbf{V},\mathbf{W})\mathbf{R}_k(\mathbf{W})\mathbf{K}_k^\top(\mathbf{V},\mathbf{W}) \\ + \mathbf{Q}_k(\mathbf{V}),\end{aligned} \tag{17}$$

where

$$\mathbf{K}_k(\mathbf{V},\mathbf{W}) = \mathbf{F}_k\mathbf{W}_k(\mathbf{V},\mathbf{W}) \tag{18}$$

$$\mathbf{X}_k(\mathbf{V},\mathbf{W}) = \mathbf{F}_k - \mathbf{K}_k(\mathbf{V},\mathbf{W})\mathbf{H}_k, \tag{19}$$

and $\mathbf{W}_k(\mathbf{V},\mathbf{W})$ is the usual Kalman filter weight.

However, the real system has noise intensities $\mathbf{V}^a$ and $\mathbf{W}^a$. Given that there are no errors in the system model equations, the expected value of the mean squared error of the filter is actually

$$\begin{aligned}&\mathbf{P}^a_{k|k-1}(\mathbf{V},\mathbf{W},\mathbf{V}^a,\mathbf{W}^a) = \\ &\quad\mathbf{X}_k(\mathbf{V},\mathbf{W})\mathbf{P}^a_{k-1|k-1}(\mathbf{V},\mathbf{W},\mathbf{V}^a,\mathbf{W}^a)\mathbf{X}_k^\top(\mathbf{V},\mathbf{W}) \\ &\quad+\mathbf{K}_k(\mathbf{V},\mathbf{W})\mathbf{R}_k(\mathbf{W}^a)\mathbf{K}_k^\top(\mathbf{V},\mathbf{W}) + \mathbf{Q}_k(\mathbf{V}^a).\end{aligned} \tag{20}$$

Given this, the expected value of the NEES is

$$\begin{aligned}&\epsilon_{\mathbf{x},k}(\mathbf{V},\mathbf{W},\mathbf{V}^a,\mathbf{W}^a) = \\ &\quad\operatorname{trace}\left(\mathbf{P}^{-1}_{k|k-1}(\mathbf{V},\mathbf{W})\mathbf{P}^a_{k|k-1}(\mathbf{V},\mathbf{W},\mathbf{V}^a,\mathbf{W}^a)\right).\end{aligned} \tag{21}$$

The J_{NEES} of this value is

$$J_{NEES}(\mathbf{V},\mathbf{W},\mathbf{V}^a,\mathbf{W}^a) = \left|\log\frac{\epsilon_{\mathbf{x},k}(\mathbf{V},\mathbf{W},\mathbf{V}^a,\mathbf{W}^a)}{n_x}\right|. \tag{22}$$

References

[1] B. M. Åkesson, J. B. Jørgensen, N. K. Poulsen, and S. B. Jørgensen, "A tool for kalman filter tuning," in *Computer Aided Chemical Engineering*. Elsevier, 2007, vol. 24, pp. 859–864.

[2] ——, "A generalized autocovariance least-squares method for kalman filter tuning," *Journal of Process control*, vol. 18, no. 7-8, pp. 769–779, 2008.

[3] T. D. Powell, "Automated tuning of an extended kalman filter using the downhill simplex algorithm," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 5, pp. 901–908, 2002.

[4] Z. Chen, C. Heckman, S. Julier, and N. Ahmed, "Weak in the nees?: Auto-tuning kalman filters with bayesian optimization," in *2018 21st International Conference on Information Fusion (FUSION)*. IEEE, 2018, pp. 1072–1079.

[5] T. Mu and A. K. Nandi, "Automatic tuning of l2-svm parameters employing the extended kalman filter," *Expert Systems*, vol. 26, no. 2, pp. 160–175, 2009.

[6] L. A. Scardua and J. J. Da Cruz, "Automatic tuning of the unscented kalman filter and the blind tricyclist problem: an optimization problem," *IEEE Control Systems Magazine*, vol. 36, no. 3, pp. 70–85, 2016.

[7] Mohinder Grewal and A. Andrews, "Van Loan's Method for Computing Q_k from Continuous Q," in *Kalman Filtering: Theory and Practice with MATLAB*, 2015, pp. 150–152.

[8] 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.

[9] Y. Bar-Shalom, X. Li, and T.Kirubarajan, *Estimation with Applications to Navigation and Tracking*. New York: Wiley, 2001.

[10] L. Zhang, D. Sidoti, A. Bienkowski, K. R. Pattipati, Y. Bar-Shalom, and D. L. Kleinman, "On the Identification of Noise Covariances and Adaptive Kalman Filtering: A New Look at a 50 Year-Old Problem," *IEEE Access*, vol. 8, pp. 59 362–59 388, 2020.

[11] J. Duník, O. Straka, O. Kost, and J. Havlík, "Noise covariance matrices in state-space models: A survey and comparison of estimation methods—Part I," *International Journal of Adaptive Control and Signal Processing*, vol. 31, no. 11, pp. 1505–1543, 2017.

[12] C. M. Bishop, *Pattern recognition and machine learning*. New York: Springer, 2006.

[13] S. T. Barratt and S. P. Boyd, "Fitting a Kalman smoother to data," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 1526–1531.

[14] J. Duník, O. Kost, O. Straka, and E. Blasch, "Covariance estimation and Gaussianity assessment for state and measurement noise," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 1, pp. 132–139, 2020.

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

[16] Y. Oshman and I. Shaviv, "Optimal tuning of a kalman filter using genetic algorithms," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2000, p. 4558.

[17] M. Saha, R. Ghosh, and B. Goswami, "Robustness and sensitivity metrics for tuning the extended Kalman filter," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 4, pp. 964–971, 2013.

[18] R. Piché, "Online tests of Kalman filter consistency," *International Journal of Adaptive Control and Signal Processing*, vol. 30, no. 1, pp. 115–124, 2016.

[19] R. G. Gibbs, "New Kalman filter and smoother consistency tests," *Automatica*, vol. 49, no. 10, pp. 3141–3144, 2013.

[20] B. Minasny and A. B. McBratney, "The matérn function as a general model for soil variograms," *Geoderma*, vol. 128, no. 3-4, pp. 192–207, 2005.

[21] R. Martinez-Cantin, "Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits," *Journal of Machine Learning Research*, vol. 15, pp. 3915–3919, 2014. [Online]. Available: http://jmlr.org/papers/v15/martinezcantin14a.html