Kalman Filter Auto-Tuning With Consistent and Robust Bayesian Optimization

ZHAOZHONG CHEN , Member, IEEE
HAREL BIGGIE , Student Member, IEEE
NISAR AHMED , Member, IEEE
University of Colorado Boulder, Boulder, CO, USA

SIMON JULIER , Member, IEEE
University of College London, London, U.K.

CHRISTOFFER HECKMAN , Senior Member, IEEE
University of Colorado Boulder, Boulder, CO, USA

The nonlinear and stochastic relationship between noise covariance parameter values and state estimator performance makes optimal filter tuning a very challenging problem. Popular optimization-based tuning approaches can easily get trapped in local minima, leading to poor noise parameter identification and suboptimal state estimation. Recently, black box techniques based on Bayesian optimization with Gaussian processes (GPBO) have been shown to overcome many of these issues, using normalized estimation error squared and normalized innovation error statistics to derive cost functions for Kalman filter auto-tuning. While reliable noise parameter estimates are obtained in many cases, GPBO solutions obtained with these conventional cost functions do not always converge to optimal filter noise parameters and lack robustness to parameter ambiguities in time-discretized system models. This article addresses these issues by making two main contributions. First, new cost functions are developed to determine if an estimator has been tuned correctly. It is shown that traditional chi-square tests are inadequate for correct auto-tuning because they do not accurately model the distribution of innovations when the estimator is incorrectly tuned. Second, the new metrics (formulated over multiple time discretization intervals) is combined with a student-t processes Bayesian optimization to achieve robust estimator performance for time discretized state space models. The robustness, accuracy, and reliability of our approach are illustrated on classical state estimation problems.

I. INTRODUCTION

State estimation algorithms are a key component of many autonomous navigation, tracking, and data fusion systems. However, the performance of these estimators is heavily influenced by the choice of the noise parameters within them. Most estimation algorithms have two main steps: state prediction followed by measurement update. The state prediction step uses a process model to predict how the state will evolve over time. The measurement update step uses an observation model to incorporate measured quantities into the estimate of the state. Most models inherently contain uncertainties or imperfections, and the errors in these models are treated as random noise. Since most designs assume these errors are drawn from a white zero mean and uncorrelated distribution, only the noise covariances for the prediction and observation models need to be chosen or tuned.

In this article, we consider the problem of developing an algorithm to automatically tune the noise parameters. We focus specifically on the problem of tuning a Kalman filter [1], although our work should generalize directly to many other types of estimators. Our work makes two main contributions. First, we develop a new measure of statistical consistency. We show that the widely-used normalized estimation error squared/normalized innovation error (NEES/NIS) [2] statistics are *only* χ^2 distributed random variables when the filter is correctly tuned. When the filter is not tuned (e.g., during the tuning process), the NEES/NIS statistics are sampled from a *generalized* χ^2 distribution. This means that optimization metrics based on χ^2 distributions can fail to converge onto correct noise covariance parameters. We use this to derive a new consistency measure and objective functions for tuning algorithms that are robust to this situation. Second, we develop a new tuning algorithm based on student-t process Bayesian optimization (TPBO) [3], [4], which is more flexible and overcomes the limitations of previously developed Gaussian process Bayesian optimization (GPBO) [5] method by providing additional robustness to noisy objective function evaluations in the noise covariance parameter search. We show a simplified overview of contributions in Fig. 1. The rest of this article is organized as follows. Section II presents the problem formulation and preliminaries, where we discuss the background of Kalman filter and the definition of statistical consistency. It also derives and describes the conventional filter consistency measuring approach, where we introduce the NEES and NIS and how to measure the consistency with them. Section III reviews filter consistency measures typically used for optimization-based black box
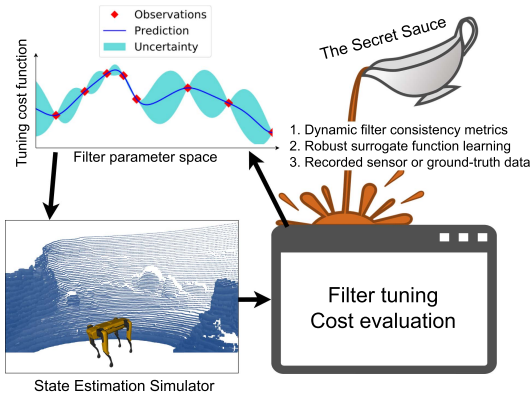
Fig. 1. We utilize nonparameteric BO to optimize the Kalman filter process and measurement noise. Our innovative procedure combines new cost metrics based on Chi-squared tests with T-process-based BO for a better evaluation of the filter consistency.

filter auto-tuning. Section IV describes our new consistency measures for determining whether a filter is tuned correctly, which incorporate information about both the means and variances of the underlying state and measurement error distributions. Section V describes our new TPBO auto-tuning algorithm, which provides additional robustness in the noise covariance parameter search process. Section VI presents comparative results using our new approach and other state-of-the-art algorithms on three different linear filter auto-tuning problems of increasing complexity. Finally, Section VII concludes this article.

II. SYSTEM OVERVIEW AND CONSISTENCY

A. Statistical Consistency of the Kalman Filter

Our approach depends upon adjusting the prediction interval in the Kalman filter. Therefore, it is important to understand the relationship between discrete and continuous time systems. The state of the system at time t is $\mathbf{x}_t$. The system is described by a continuous-time process model and observation models [6], [7], [8]

$$\dot{\mathbf{x}}_t = \mathbf{A}_t \mathbf{x}_t + \mathbf{G}_t \mathbf{u}_t + \mathbf{\Gamma}_t \mathbf{v}_t$$
$$\mathbf{z}_t = \mathbf{H}_t \mathbf{x}_t + \mathbf{w}_t \qquad (1)$$

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 $\mathbf{W}$.

Using time discretization techniques, such as Van Loan's method [9], [10] (Appendix A), the discrete-time process model describes the evolution of the system from timestep $k-1$ to k

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{v}_k \qquad (2)$$

where $\mathbf{u}_k$ is the control input (e.g., under zero-order hold) and $\mathbf{v}_k$ is the process noise, which is assumed to be independent with a zero mean and covariance $\mathbf{Q}_k$. The observation model is

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{w}_k \qquad (3)$$

where $\mathbf{w}_k$ is the observation noise. This is assumed to be zero-mean and independent with covariance $\mathbf{R}_k$. The discretization process shows that the sample time Δt plays a significant role and we utilize it to define our cost function, which is shown in Sections III-C and IV-D.

In this article, we explore the case that a Kalman filter is used to find the optimal estimate [11]. Consider the state at time k conditioned on all measurements up to time j. The estimate from the Kalman filter is the mean $\hat{\mathbf{x}}_{k|j}$ and the associated covariance $\mathbf{P}_{k|j}$. The filter follows the two-stage process of prediction followed by an update. The prediction is

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k$$
$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^\top + \mathbf{Q}_k.$$

The update is

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \mathbf{e}_{\mathbf{z},k}$$
$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{S}_{k|k-1} \mathbf{K}_k^\top$$
$$\mathbf{S}_{k|k-1} = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k$$
$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \mathbf{S}_{k|k-1}^{-1}$$

where $\mathbf{e}_{\mathbf{z},k} = \mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}$ is the *innovation vector*. Many other recursive estimation algorithms, including Gaussian mixture models, particle filters, and grid-based estimators have a similar two-step structure.

When the model implemented in the filter precisely matches the system equations, the filter should behave in a statistically consistent manner. Specifically, the filter should obey the following conditions for dynamical consistency [2].

1) The state estimation errors are unbiased

$$\mathbb{E}\left[\mathbf{e}_{\mathbf{x},k}\right] = \mathbf{0} \quad \forall k. \qquad (4)$$

2) The estimator is efficient

$$\mathbb{E}\left[\mathbf{e}_{\mathbf{x},k} \mathbf{e}_{\mathbf{x},k}^T\right] = \mathbf{P}_{k|k} \quad \forall k. \qquad (5)$$

3) The innovations form a white Gaussian sequence, such that for all times k and j

$$\mathbf{e}_{\mathbf{z},k} \sim \mathcal{N}(\mathbf{0}, \mathbf{S}_{k|k-1})$$
$$\mathbb{E}\left[\mathbf{e}_{\mathbf{z},k}\right] = \mathbf{0}$$
$$\mathbb{E}\left[\mathbf{e}_{\mathbf{z},k} \mathbf{e}_{\mathbf{z},j}^T\right] = \delta_{jk} \cdot \mathbf{S}_{k|k-1} \qquad (6)$$

where $\mathbf{e}_{\mathbf{x},k} = \mathbf{x}_k - \hat{\mathbf{x}}_{k|k}$. In many cases, statistical consistency cannot be achieved due to practical system considerations. For example, lower order models can introduce biases that manifest themselves as time-correlated errors, which violate statistical consistency. Therefore, most practical systems are evaluated on a weaker condition we discuss in Section II-C.

B. Filter Error Statistics

The goal of tuning is to choose values $\mathbf{Q}_k$ and $\mathbf{R}_k$ that meet the consistency conditions. Most tuning approaches use an optimization approach where measures of consistency are introduced, and the noise matrices are

adjusted to maximize a measurement of the system performance.

Two widely used performance measurements are the *NEES* and the *NIS* [2]. The NEES is computed from

$$\epsilon_{\mathbf{x},k} = \mathbf{e}_{\mathbf{x},k}^T \mathbf{P}_{k|k}^{-1} \mathbf{e}_{\mathbf{x},k}. \tag{7}$$

Because this requires knowledge of the true system state ($\mathbf{x}_k$), it can only be computed in simulation or in situations where an external validation system is being used. NIS, on the other hand, only depends on the observation sequence and does not require ground truth (GT) knowledge. It is computed from

$$\epsilon_{\mathbf{z},k} = \mathbf{e}_{\mathbf{z},k}^T \mathbf{S}_{k|k-1}^{-1} \mathbf{e}_{\mathbf{z},k} \tag{8}$$

where $\mathbf{e}_{\mathbf{z},k}$ is the innovation vector.

If the filter is tuned, it can be shown that the NEES and NIS are χ^2-distributed random variables [2]. We illustrate this for the NEES. Define the vector-valued quantity

$$\mathbf{b}_k = \mathbf{P}_{k|k}^{-1/2} \mathbf{e}_{\mathbf{x},\mathbf{k}}. \tag{9}$$

Substituting, (7) can be rewritten as

$$
\begin{aligned}
\epsilon_{\mathbf{x},k} &= \left(\mathbf{b}_k^T \mathbf{P}_{k|k}^{1/2} \right) \mathbf{P}_{k|k}^{-1} \left(\mathbf{P}_{k|k}^{1/2} \mathbf{b}_k \right) \\
&= \mathbf{b}_k^T \mathbf{b}_k \\
&= \sum_{i=1}^{n_{\mathbf{x}}} b_k(i)^2
\end{aligned}
\tag{10}
$$

where $b_k(i)$ is the *i*th component of $\mathbf{b}_k$. Since the filter is tuned, the consistency conditions in (4) and (5) mean that $\epsilon_{\mathbf{x},k}$ is zero-mean with covariance $\mathbf{P}_{k|k}$. Therefore, $\mathbf{b}_k$ is a zero mean, $n_{\mathbf{x}}$-dimensional Gaussian-distributed random variable with covariance equal to the identity matrix. Furthermore, this means that each component $b_k(i)$ is also a zero-mean Gaussian random variable with unit covariance. Therefore, $\epsilon_{\mathbf{x},k}$ is a χ^2-distributed random variables with $n_{\mathbf{x}}$ DOF. Similar logic can be used to show that the NIS is χ^2-distributed random variable with $n_{\mathbf{z}}$ DOF.

Given these insights, the mean values of NEES and NIS are used to evaluate consistency. Their values are given by [2]

$$\mathbb{E}[\epsilon_{\mathbf{x},k}] \approx n_{\mathbf{x}}, \quad \mathbb{E}[\epsilon_{\mathbf{z},k}] \approx n_{\mathbf{z}}. \tag{11}$$

C. Empirically Measuring Consistency

In practice, NEES/NIS χ^2 tests are conducted using multiple offline Monte Carlo "truth model" simulations to obtain GT $\mathbf{x}_k$ values. For a hypothesized set of filter parameters, N Monte Carlo runs are executed: The truth model is run, observations are collected, the filter predicts and updates, and the empirical average of the NEES and NIS are computed. This averaging is often done on a time-step-by-time step basis across all Monte Carlo runs

$$\bar{\epsilon}_{\mathbf{x},k} = \frac{1}{N} \sum_{i=1}^{N} \epsilon_{\mathbf{x},k}^{[i]} \tag{12}$$

$$\bar{\epsilon}_{\mathbf{z},k} = \frac{1}{N} \sum_{i=1}^{N} \epsilon_{\mathbf{z},k}^{[i]} \tag{13}$$

where the superscript [*i*] shows this is the quantity from the *i*th run. The estimated values are stochastic because each run is perturbed by random noise. Therefore, given some desired Type I error [12] rate α, or the level of significance for the hypothesis, the NEES, and NIS χ^2 tests provide lower and upper tail bounds. $[l_{\mathbf{x}}(\alpha, N), u_{\mathbf{x}}(\alpha, N)]$ and $[l_{\mathbf{z}}(\alpha, N), u_{\mathbf{z}}(\alpha, N)]$, such that the Kalman filter tuning is declared to be consistent if, with probability $100(1-\alpha)$ at each time k

$$
\begin{aligned}
\bar{\epsilon}_{\mathbf{x},k} &\in [l_{\mathbf{x}}(\alpha, N), u_{\mathbf{x}}(\alpha, N)] \\
\bar{\epsilon}_{\mathbf{z},k} &\in [l_{\mathbf{z}}(\alpha, N), u_{\mathbf{z}}(\alpha, N)].
\end{aligned}
\tag{14}
$$

If these conditions are not met, the filter is declared to be inconsistent. Specifically, if $\bar{\epsilon}_{\mathbf{x},k} < l_{\mathbf{x}}(\alpha, N)$ or $\bar{\epsilon}_{\mathbf{z},k} < l_{\mathbf{z}}(\alpha, N)$, then the filter tuning is "pessimistic" (underconfident), since the filter-estimated state error/innovation covariances are too large relative to the true values. On the other hand, if $\bar{\epsilon}_{\mathbf{x},k} > u_{\mathbf{x}}(\alpha, N)$ or $\bar{\epsilon}_{\mathbf{z},k} > u_{\mathbf{z}}(\alpha, N)$, then the filter tuning is "optimistic" (overconfident) [13], since the filter-estimated state error/innovation covariance is too small relative to the true values. Note that as the number of Monte Carlo runs N increases, the stochastic variation in $\bar{\epsilon}_{\mathbf{x},k}$ and $\bar{\epsilon}_{\mathbf{z},k}$ decreases, and the bounds become tighter.

We can use the above metric to evaluate the consistency of a filter. However, because this metric is guaranteed to be nonnegative, it is not symmetric—the minimum is zero (if the covariance goes to infinity), and the maximum is unbounded from above (if the covariance matrix becomes singular). One way to overcome this is to use the absolute value of the log of the normalized errors [14], [15], [16], [17]

$$J_{\mathrm{NEES}} = \left| \log\left(\frac{\tilde{\epsilon}_x}{n_{\mathbf{x}}} \right) \right| \tag{15}$$

$$J_{\mathrm{NIS}} = \left| \log\left(\frac{\tilde{\epsilon}_z}{n_{\mathbf{z}}} \right) \right| \tag{16}$$

where

$$\tilde{\epsilon}_x = \frac{1}{T} \sum_{k=1}^{T} \bar{\epsilon}_{\mathbf{x},k}, \quad \tilde{\epsilon}_z = \frac{1}{T} \sum_{k=1}^{T} \bar{\epsilon}_{\mathbf{z},k}.$$

We next review existing work on selecting $\mathbf{Q}_k$ and $\mathbf{R}_k$.

III. PROCESS AND OBSERVATION NOISE SELECTION

A. Two-Stage Approach

There are various existing ways to tune the Kalman filter. Perhaps the simplest approaches use a two-stage divide-and-conquer strategy. The observation noise is first determined using lab-based sensor measurements. The observation noise is held fixed and the process noise is adjusted until the conditions in (14) are satisfied.

However, there are several problems with this two-stage approach. First, with laboratory testing, it is not always possible to model the reaction of sensors in real-world operational environments. For instance, changes in temperature can cause the biases in inertial measurement units to change [18] which can result in the poor

characterization of the observation noise. In addition, classifying the interactions between noise levels and filter performance is not straightforward, even in the case of linear observation models. It has been theoretically shown that even when the process and observation models are linear, the presence of modeling errors leads to state-dependent noise models, which are correlated over time [19], [20].

B. Autotuning Algorithms

A more practical alternative is to use auto-tuning algorithms, which include: Maximum likelihood and Bayesian inference [21], least squares for data processed via Kalman smoothing [22], and auto-/cross-correlation analysis [23]. While all of these methods have their advantages, no single best technique exists for practical use [24], [25]. 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 nonlinear filters, e.g., since the optimal set of noise parameters in linearization-based filters can vary significantly with system state and time [26].

In this work, we focus on posing the tuning as an optimization problem. We try to identify parameters directly using data from actual rollouts of the system and corresponding filters. This effectively eschews the two-step process; instead, we minimize a customized cost attached to some property of deleterious estimates. Researchers have tried various optimizers, such as gradient descent [27], [28], generalized least squares [29], downhill simplex [16], and expectation maximization (EM) algorithm [30]. However, a good initial guess needs to be provided for these methods to prevent them from getting stuck inside of local minima. Such minima are frequent due to inherent nonlinearities in the process and observation models, and because of sampling noise. Therefore, there is considerable interest in exploring more global methods, including genetic algorithms [31], [32], [33] and Bayesian optimization (BO). The approach in [33] finds a globally optimal parameter value set but is sample-inefficient, requiring thousands of cost function evaluations to determine the optimum.

C. Implicit Time Dependence

We discretize the plant transformation using a sample time of Δt. As we have previously shown in [34], the choice of Δt has a significant impact on the performance of the filter. When a single sample time is used, multiple choices of $\mathbf{Q}_k$ and $\mathbf{R}_k$ produce consistent-looking results according to J_{NIS} and J_{NEES}. However, those choices cannot closely match the optimal $\mathbf{Q}_k$, $\mathbf{R}_k$ we set in the GT and leads to an ambiguity estimation. In [34], we demonstrated that the ambiguity could be resolved by evaluating filter performance over several prediction intervals and using the maximum of J_{NIS} as the cost function (evaluating the worst-case behavior). To correctly handle the different prediction intervals, the models were specified

in continuous time and converted to discrete-time using Van Loan's method [35]. In this work, we leverage all the filter performances and sum the cost from each interval prediction as the cost function to the optimization. Specifically, suppose we have n prediction time intervals $\{\Delta t_1, \ldots, \Delta t_n\}$. Let $J_{\mathrm{NIS}}(\Delta t_n)$ be the J_{NIS} value computed when the prediction interval is Δt. In particular, we used the value

$$J_{\mathrm{NIS}} = \sum_{i=1}^{n} J_{\mathrm{NIS}}(\Delta t_n). \tag{17}$$

Theoretically, more prediction intervals can result in better optimization, but it can increase the overall computation time. The n value is decided empirically; e.g., as detailed in Section VI, for a low-dimension system we use $n = 2$, and for a higher dimension system we use $n = 4$. Although this solution was sufficient to eliminate the ambiguity identified in [34], it does not eliminate all ambiguities. In particular, there is a fundamental theoretical issue in the use of NEES and NIS to ensure that a filter has been tuned consistently. We illustrate this in a simple linear example in the following section.

IV. NEW CONSISTENCY MEASURES

In this section, we introduce a new consistency measure. We first present an example that illustrates the limitations of J_{NEES} and J_{NIS}. Then, we analyze the underlying reason for these limitations and propose a new and more robust measure.

A. Illustrative Example

Consider a linear particle tracking problem. The target state is $\mathbf{x} = [x, y, \dot{x}, \dot{y}]^T$, where the motion is corrupted by random accelerations. The particle's position is measured, and the measurements are perturbed by zero-mean, Gaussian-distributed noise. The discrete-time system equations are

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0.5\Delta t^2 \\ 0.5\Delta t^2 \\ \Delta t \\ \Delta t \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^3}{3} v_0 & 0 & \frac{\Delta t^2}{2} v_0 & 0 \\ 0 & \frac{\Delta t^3}{3} v_1 & 0 & \frac{\Delta t^2}{2} v_1 \\ \frac{\Delta t^2}{2} v_0 & 0 & \Delta t v_0 & 0 \\ 0 & \frac{\Delta t^2}{2} v_1 & 0 & \Delta t v_1 \end{bmatrix}. \tag{18}$$

The simulation setup implements this model with noise values $(v_0, v_1, w_0, w_1) = (1, 2, 0.2, 0.1)$ and timestep length $\Delta t = 0.1$.

Next, consider the case where a filter is implemented using the GT values. The mean value of the NIS, computed from (13), should be $n_{\mathbf{z}} = 2$. This is confirmed in Fig. 2, which plots (in red) the average NIS over 100 Monte Carlo runs. However, the figure also shows (in blue) the variance NIS statistics for a filter tuned with the values $(v_0, v_1, w_0, w_1) = (0.855, 3.000, 0.122, 0.294)$. The
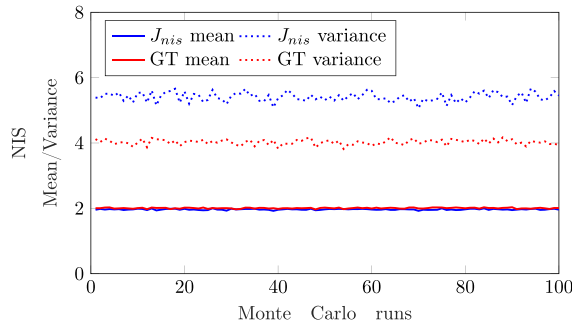
Fig. 2. Linear tracking example, where ground truth (GT, red) and tuned NIS (J_{NIS}, blue) statistics are shown. We observe that the means are the same but the variances are different, which implies the mean is not sufficient for tuning the filter.

NIS variance statistics is around 5.5, whereas the GT is 4. Even though these variance values are very different from the GT parameters, the mean value of the tuned NIS is, once again, 2. This clearly shows that the NIS (and by extension the NEES) values are not sufficient, on their own, to show that a filter is correctly tuned because both mean and variance statistics should be close to the GT value.

We then demonstrate why the J_{NEES} and J_{NIS} are not sufficient to tune the filter theoretically.

B. Moments of a Quadratic Form

Both (7) and (8) are equivalent to computing the expected value of the quadratic form

$$\bar{\epsilon} = \mathbb{E}[\boldsymbol{\epsilon} \boldsymbol{\Lambda} \boldsymbol{\epsilon}^T]$$

where $\boldsymbol{\epsilon}$ is a random vector with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, the $\boldsymbol{\Lambda}$ is a positive definite symmetric matrix. Taking expectations

$$\bar{\epsilon} = \text{tr}[\boldsymbol{\Lambda} \boldsymbol{\Sigma}] + \boldsymbol{\mu}^T \boldsymbol{\Lambda} \boldsymbol{\mu}. \qquad (19)$$

Consider the case we are computing the NEES. When the filter is tuned correctly, the results in Section II-B mean that $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ and $\boldsymbol{\mu} = \boldsymbol{0}$. Substituting into (19)

$$\begin{aligned} \bar{\epsilon} &= \text{tr}[\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}] + \boldsymbol{0}^T \boldsymbol{\Lambda} \boldsymbol{0} \\ &= \text{tr}[\mathbf{I}_n] \\ &= n_{\mathbf{x}}. \end{aligned} \qquad (20)$$

Similarly, for the NIS values, $\boldsymbol{\Lambda} = \mathbf{S}_{k|k-1}^{-1}$, $\boldsymbol{\mu} = \boldsymbol{0}$, and the expected value is $n_{\mathbf{z}}$.

The quadratic form also lets us compute the covariance. If $\boldsymbol{\epsilon}$ is Gaussian-distributed, then it can be shown that

$$\mathbb{E}[(\epsilon - \bar{\epsilon})^2] = 2\text{tr}[\boldsymbol{\Lambda} \boldsymbol{\Sigma} \boldsymbol{\Lambda} \boldsymbol{\Sigma}] + 4\boldsymbol{\mu}^T \boldsymbol{\Lambda} \boldsymbol{\Sigma} \boldsymbol{\Lambda} \boldsymbol{\mu}. \qquad (21)$$

Therefore, when the filter is tuned correctly, the covariance in the NEES is

$$\begin{aligned} \mathbb{E}[(\epsilon - \bar{\epsilon})^2] &= 2\text{tr}[\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}] + 4\boldsymbol{0}^T \boldsymbol{\Lambda} \boldsymbol{\Sigma} \boldsymbol{\Lambda} \boldsymbol{0} \\ &= 2\text{tr}[\mathbf{I}_n^2] \\ &= 2n_{\mathbf{x}} \end{aligned} \qquad (22)$$

and the covariance in the NIS is $2n_{\mathbf{z}}$.

This property demonstrates that a correctly tuned filter should have (11) and

$$\mathbb{E}\left[\bar{\epsilon}_{\mathbf{x},k}, \bar{\epsilon}_{\mathbf{x},k}\right] \approx 2n_{\mathbf{x}}, \quad \mathbb{E}\left[\bar{\epsilon}_{\mathbf{z},k}, \bar{\epsilon}_{\mathbf{z},k}\right] \approx 2n_{\mathbf{z}}. \qquad (23)$$

As a result, a standard χ^2 distribution property $\mathcal{V} = 2\mathbf{m}$ can hold, where $\mathcal{V}$ is the variance, and m is the mean value.

C. Limitations of J_{NEES} and J_{NIS}

We can see the relevance of these values in Fig. 2, which also plots the covariance values of the NIS computed over the Monte Carlo runs. For the correctly tuned filter, the covariance should be around 4, which is $2n_{\mathbf{z}}$. However, for the mistuned filter, the covariance is about 5.75. This tuned NIS value follows a standard χ^2 distribution. The J_{NEES} and J_{NIS} only consider the mean term (20) instead of the covariance term from (22). As a result, they are not sufficient to tune the filter correctly.

J_{NEES} and J_{NIS} both assume that (10) holds true and these values are χ^2-distributed. However, when the filter is not tuned correctly, two effects can arise. First, the state error and innovation might no longer have a zero mean. Second, the filter incorrectly estimates the covariance of the state and the innovation. As a result, (10) now needs to include mean $\mu_k(i)$ and variance $w_k(i)$ and becomes

$$\epsilon_{\mathbf{x},k} = \sum_{i=1}^{n_{\mathbf{x}}} w_k(i) \left(b_k(i) + \mu_k(i)\right)^2. \qquad (24)$$

This random variable, as a sum of squares of independent normal variables, is described by a *generalized* χ^2 distribution. Therefore, we need a consistency measure that will be able to differentiate a mistuned filter with a generalized χ^2 distribution and a correctly tuned filter with a standard χ^2 distribution. Although complicated sophisticated distribution fit tests could be applied, we find that a simple test, based on the moments of quadratic forms, is sufficient. We introduce new consistency measures, which ensure that both the mean and covariance values are correct.

D. New Consistency Measure

Our new NEES measure has the form

$$\begin{aligned} C_{\text{NEES}} &= \left| \log\left(\frac{\tilde{\epsilon}_x}{n_{\mathbf{x}}}\right) \right| + \left| \log\left(\frac{\tilde{S}_x}{2n_{\mathbf{x}}}\right) \right| \\ \tilde{S}_x &= \frac{1}{T(N-1)} \sum_{k=1}^{T} \sum_{i=1}^{N} (\epsilon_{\mathbf{x},k}^i - \bar{\epsilon}_{\mathbf{x},k}^i)^2. \end{aligned} \qquad (25)$$

Similarly, our new NIS measure has the form.

$$\begin{aligned} C_{\text{NIS}} &= \left| \log\left(\frac{\tilde{\epsilon}_z}{n_{\mathbf{z}}}\right) \right| + \left| \log\left(\frac{\tilde{S}_z}{2n_{\mathbf{z}}}\right) \right| \\ \tilde{S}_z &= \frac{1}{T(N-1)} \sum_{k=1}^{T} \sum_{i=1}^{N} \left(\epsilon_{\mathbf{z},k}^i - \bar{\epsilon}_{\mathbf{z},k}^i\right)^2. \end{aligned} \qquad (26)$$

Considering the implicit time dependence property mentioned in Section III-C and (17), and given multiple time discretization intervals Δt_n, our final NEES/NIS metrics

have the form

$$C_{\text{NEES}} = \sum_{i=1}^{n} C_{\text{NEES}}(\Delta t_n)$$

$$C_{\text{NIS}} = \sum_{i=1}^{n} C_{\text{NIS}}(\Delta t_n). \qquad (27)$$

V. BO-BASED TUNING

Now that we have more suitable measures of consistency, the next task is to develop an algorithm that will choose the noise parameters to optimize the desired consistency measure. However, this leads to a very challenging optimization problem. The relationship between noise parameters and C_{NIS} or C_{NEES} can be highly nonlinear with many local minima [14], [34] (visualized in Appendix C). This difficulty is compounded by the fact that these quantities are computed empirically with a finite number of samples. Therefore, each computed value will be stochastic. A principled way to address both multiple local minima and noisy cost function evaluations is to use BO [36], [37], [38]. This poses optimization as a probabilistic search problem in which the goal is to find the global minimum within a bounded region. The system maintains a probabilistic estimate of the cost function within this search region, and the optimizer refines and uses this estimate as it searches for the optimal solution.

A. TPBO Theory

Consider the objective function $y : \mathcal{Q} \to \mathbb{R}$, which maps a point $\mathbf{q}$ from the d-dimensional space $\mathcal{Q} \in \mathbb{R}^d$ to the real value. We assume that the value of each elements of i of $\mathbf{q}$ is bounded in a finite region. Specifically, for the ith component of $\mathbf{q}$, $\mathbf{q}(i) \in [\mathbf{q}(i)_l, \mathbf{q}(i)_u]$. The goal is to find $\mathbf{q}^* \in \mathcal{Q}$, which minimizes y.

There are the two main components in the BO algorithm: 1) The surrogate model $\mathcal{S}$, which encodes statistical beliefs about y; and 2) the acquisition function, which is used to intelligently guide the search for $\mathbf{q}^*$ using $\mathcal{S}$.

1) *Surrogate Model:* This is a probabilistic approximation of the objective function y over the search space. In general, y is expensive to evaluate for filter tuning because data from a candidate filter configuration (i.e., with candidate noise covariance values) must be collected either from GT data for NEES-based evaluation or from recorded sensor data logs for NIS-based evaluation. On the other hand, a surrogate model of y is easier to evaluate and can provide sufficient information to guide a search toward optimum covariance parameters that are informed by data. Moreover, the surrogate model can embed the locations of multiple local minima without trapping the search process.

To this end, we use a nonparameteric regression model based on a student's-t process to construct the surrogate function from sampled data runs using a candidate filter tuning. By definition, a student's-t process is a stochastic process, such that every sample $\mathbf{q}$ from the process has the multivariate student-t joint distribution

$$y(\mathbf{q}) \sim \mathcal{TP}(v, \Phi(\mathbf{q}), k(\mathbf{q}, \mathbf{q}')). \qquad (28)$$

The mean function is $\Phi(\mathbf{q})$, the kernel function is $k(\mathbf{q})$, and the parameter $v > 2$ controls how heavy-tailed the process is. Smaller values of v correspond to heavier tailed distributions, with higher probabilities of extreme values. On the other hand, as $v \to \infty$, the process converges toward a Gaussian process (GP) with light tails.

The student-t process (TP) is attractive because it provides some extra benefits over GP surrogate models that are more commonly used, without incurring more computational cost. For example, the predictive covariance for the TP explicitly depends on observed y data values; this is a useful property, which the GP lacks. Furthermore, distributions over the cost function y may in general be heavy-tailed, so it is better to use TP to "safely" model their behaviors [4]. Similarly, every finite collection of TP samples $\mathbf{q}_{1:n} = (\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n)$ has a multivariate student-t distribution

$$y(\mathbf{q}_{1:n}) \sim \text{MVT}_n(v, \Phi(\mathbf{q}_{1:n}), K) \qquad (29)$$

where K is the covariance matrix consisting of kernel function k [38], [39], [40] evaluations

$$K = \begin{bmatrix} k(\mathbf{q}_1, \mathbf{q}_1) & k(\mathbf{q}_1, \mathbf{q}_2) & \cdots & k(\mathbf{q}_1, \mathbf{q}_n) \\ k(\mathbf{q}_2, \mathbf{q}_1) & k(\mathbf{q}_2, \mathbf{q}_2) & \cdots & k(\mathbf{q}_2, \mathbf{q}_n) \\ \vdots & \vdots & \vdots & \vdots \\ k(\mathbf{q}_n, \mathbf{q}_1) & k(\mathbf{q}_n, \mathbf{q}_2) & \cdots & k(\mathbf{q}_n, \mathbf{q}_n) \end{bmatrix}. \qquad (30)$$

As more and more samples are taken, more and more information becomes available about the objective function. The surrogate model is updated in two ways to exploit this new information. First, the newly sampled $\mathbf{q}$ and y values are added to the vector $\mathbf{q}_{1:n}$ and $y(\mathbf{q}_{1:n})$. Second, the hyperparameters of the kernel function k are re-estimated from the data. The updated surrogate model is then used to compute the acquisition function, which is used to select the next sample $\mathbf{q}$ for evaluation of y.

2) *Acquisition Function:* This queries the surrogate model to predict where the next sample should be taken. Many different acquisition functions have been proposed in the BO literature, varying in the kind probabilistic information they exploit for particular applications. In this work we use *expected improvement* (EI), which selects the point, which has the highest probability of being the optimal solution. To achieve this, we first define the improvement

$$g(\mathbf{q}_{n+1}, \mathbf{q}_{1:n}^*) = \max(0, y_n(\mathbf{q}_{1:n}^*) - y(\mathbf{q}_{n+1})). \qquad (31)$$

This is nonzero only if $\mathbf{q}_{n+1} < \mathbf{q}_{1:n}^*$. Since we only have access to the surrogate function, the improvement is stochastic. Therefore, we use the EI

$$\text{EI}_n(\mathbf{q}) = \mathbb{E}_n[g(\mathbf{q}, \mathbf{q}_{1:n}^*) \mid \mathbf{q}_{1:n}, \mathbf{y}(\mathbf{q}_{1:n})] \qquad (32)$$

where $E_n[\cdot]$ is the expectation based on current posterior distribution, given by the current MVT surrogate model. The sample with the highest EI is found from

$$\mathbf{q}_{n+1} = \arg\max_{\mathbf{q}} \text{EI}_n(\mathbf{q}). \qquad (33)$$

There are two main advantages to the EI measure. First it has a clear intuitive meaning, Second, for the TP surrogate

Algorithm 1: TPBO for Kalman Filter Tuning.

1: Initialize TP seed data $\{\mathbf{q}_s, y_s\}_{s=1}^{N_{seed}}$ and hyperparameters Θ
2: **while** $n < N$ **do**
3:    $\mathbf{q}_j = \arg\max_Q a(\mathbf{q})$
4:    $y(\mathbf{q}_j) \leftarrow$ COMPUTETPBOSAMPLE $\mathbf{q}_j, \Delta T_{1,...,m}$
5:    Add $y(\mathbf{q}_j)$ to $\mathbf{f}(Q)$, $\mathbf{q}_j$ to Q, and update Θ
6: **end while**
7: **return** $\mathbf{q}^* = \arg\min_{\mathbf{q}_j \in Q} \mathbf{f}(\mathbf{q}_j)$

Algorithm 2 Compute TPBO Sample

  procedure COMPUTETPBOSAMPLE $\mathbf{q}, \Delta T_{1,...,m}$
    $Z \leftarrow$ DATAGENERATOR
    for $\Delta T \in \{\Delta T_1, \ldots, \Delta T_m\}$ **do**
      $C_{NIS}^i \leftarrow$ COMPUTECNIS $\mathbf{q}, \Delta T_i, Z$
    end for
    return $\sum(C_{NIS}^i)$
  end procedure

Algorithm 3 Compute C_{NIS} for Each Sample

  procedure COMPUTECNIS $\mathbf{q}, \Delta T, Z$
    Initialize Kalman Filter $\mathbf{K}(\mathbf{q}, Z)$
    for $t \in \Delta T$ **do**
      Iterate $\mathbf{K}$ using timestep t
      $C_{NIS} \leftarrow$ from (26)
    end for
    return C_{NIS}
  end procedure

---

model, a closed form solution exists for (32) [4], [41]

$$\text{EI}_n(\mathbf{q}) = \left(y_n(\mathbf{q}_{1:n}^*) - u\right)\Psi(z)$$
$$+ \frac{v}{v-1}\left(1 + \frac{z^2}{v}\right)\sigma\psi(z). \qquad (34)$$

See Appendix B for more details.

Given this closed form solution, we still need to solve (33). Although this is another optimization problem, it is much easier to solve than the original optimization problem. We use *DIRECT*, which is a derivative free and deterministic nonlinear global optimization algorithm that is widely used for BO via nonparameteric surrogate model regression [42]. Once **q** has been determined, the cost objective function is evaluated and the surrogate model is updated. The acquisition function is used to identify the next sample and this process repeats until the termination criteria are met. We use maximum iteration or minimum observation change between two iterations.

B. Tuning Algorithm Summary

Algorithm 1 summarizes the TPBO procedure for Kalman filter tuning. In addition to the revised consistency measure, it also uses multiple timesteps and the summation scheme from (27). Algorithm 2 details the computation of an individual TBPO sample using a data generator function DATAGENERATOR (GT data for NEES, sensor data for NIS), and calls the metric evaluation function shown in Algorithm 3. Note that while either C_{NEES} or C_{NIS} could be used in the TPBO auto-tuning method, the latter is arguably easier to implement in most real applications. This is because the C_{NIS} only requires single-run or multi-run recorded sensor data logs instead of the multiple GT state data logs required by C_{NEES}, which are generally expensive to acquire either via high-fidelity truth model simulations or high-fidelity sensors. Thus, in Algorithms 2 and 3 and for the rest of this article, C_{NIS} is used to describe and examine properties of the TPBO auto-tuning method.

VI. EXPERIMENTS

In this article, we present filter auto-tuning results on three systems that illustrate different aspects of our approach.

1) *Mass-spring-damper:* Here the state consists of the position and velocity of the mass in the system. The process and measurement noise are both 1-D.

We use the 1-D example to highlight our method's ability to predict the noise parameter precisely. In addition, we optimize over multiple GT pairs to demonstrate that our system is robust to different noise parameters, even in the most challenging situations where these are large.

2) *2-D tracking system:* In the 1-D system, tuned estimates obtained via C_{NIS} and the conventional J_{NIS} cost functions are quite similar and, therefore, we cannot see the benefits of the new C_{NIS} cost function. However, the 2-D tracking system shows that C_{NIS} provides statistically consistent results in higher dimensional problems, whereas again allowing for evaluation against a small set of GT parameters.

3) *Cascade mass-spring-damper system:* We cascade three mass-spring-dampers into six state dimensions to demonstrate consistency in higher dimensions. This problem allows us to show the necessity of the variance term in the new cost function, along with the increased significance of using multiple time discretization intervals.

We compare the results of four auto-tuning strategies in each case. The first one is the proposed TPBO algorithm with the C_{NIS} cost function. To assess the improvements of the new cost function with respect to our previous work, we compare it with TPBO with the J_{NIS} cost function. Next, we evaluate the benefits of the student's-t process surrogate modeling and multitime scale sampling approaches in TPBO by comparing with the previously developed GPBO method developed in [14], using the J_{NIS} cost function and $\Delta t = 0.1$ as the only discretization sample time. Finally, we highlight the benefits of our algorithm by comparing it against the state-of-the-art downhill simplex auto-tuning algorithm developed by Powell [16], using C_{NIS} as the cost function and also taking advantage of different Δt discretization sample times to ensure a fair comparison.

After the optimization converges for each method across 120 Monte Carlo simulation runs, we provide a series of evaluations to compare the filters: The true cost function at the converged solution; filter state estimation error accuracy; filter dynamic consistency, i.e., whether the estimated states satisfy (14); and BO surrogate model visualizations, to demonstrate the solution search process. All experiments were performed using a multithreaded implementation on an Intel CPU 10900 K processor.

A. Mass-Spring-Damping System

In this example, we introduce the mass-spring-damper model and explore the convergence of our method during the optimization process. First, we run multiple optimizations using the same GT and evaluate if the convergence is consistent over multiple runs. Second, we investigate the effectiveness of TPBO on different sets of GT system values.

The classical mass-spring-damping system state is the mass position and velocity $\mathbf{x} = [x, \dot{x}]$. The continuous time state space model is

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}, \quad \mathbf{H} = [1 \quad 0], \quad \boldsymbol{\Gamma} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$\mathbf{V} = v \quad \mathbf{W} = w$$

where $m = 1\,\mathrm{kg}$ is the mass, $k = 1\,\mathrm{N/s}$ is the spring constant, and $b = 0.2\,\mathrm{Ns\,m^{-1}}$ is the damping constant. We assume the a sinusoidal external force as an input, 1-D velocity process noise, and 1-D position measurement noise. We time discretize this model (see Appendix A for details) and, then, use TPBO to optimize the filter noise intensities v and w. We should note that v and w do not change w.r.t. sample time step since they are they continuous time noise parameters but the discrete time noise covariance $\mathbf{Q}$ and $\mathbf{R}$ are different for different sample time Δt.

The control input for the discretized system is $\mathbf{u}_t = 2\cos(0.75t)$ (discretized with zero-order hold). For the two sampling rates required by TPBO, two $\Delta t = [0.1, 0.5]$ are used with a simulation episode time of $T = 200\Delta t$. $N = 120$ Monte Carlo simulations were used in the Bayes optimization methods. Increasing the value of N can make C_{NIS} more robust at the expense of computation time. Several other key parameters had to be set for TPBO and GPBO. For the kernel function, the Matérn Kernel [39], with automatic relevance determination (ARD), was used. ARD uses independent parameters for every dimension of a given problem. As such, we also treat the optimized parameters as independent within BO. The kernel parameter v describes how many times the kernel function is differentiable; in our experiments, we use $v = 3$ according to the convention in [39]. For the remaining parameters such as the kernel mean, kernel hyperparameter relearn iteration number, and the acquisition function optimization number, we found that the default values from the BO library [43] were sufficient and the reader could view the website and understand their comprehensive meaning. For the downhill simplex method,

there are four main parameters: reflection (re), expansion (ex), contraction (co), and full contraction (fc). Here, we used re = 1, ex = 1, co = 0.5, fc = 0.5.

For the fixed GT validation, the GT is $v = 1$, and $w = 0.1$. The BO methods search ranges are $w \in [0.01, 0.5]$ and $v \in [0.1, 5]$. We run each optimization 50 independent times. We record the optimized $[v, w]$, and compute the median, variance, and mean of the 50 runs. These data are presented in Table I. The median and mean of the optimized parameters are expected to be close to the GT with only a minor variance. In Fig. 3, we present the evolution of the TPBO surrogate model during different iterations of the C_{NIS} optimization process for a single optimization run. It can be seen that the proposed method is efficient in predicting and finding the optimized parameters.

For extended validation, multiple GT values were also examined on a 5×4 grid of noise parameters defined as follows: $v = [1, 3, 5, 7, 9]$ and $w = [0.5, 2.5, 4.5, 6.5]$. We increase the BO search range of w and v to $w = [0.1, 10]$ and $v = [1, 20]$ to match the dynamics of the problem. The initial sample is set to 40 and the iteration count is set to 160. For each GT pair on the 5×4 grid, we run the optimization once and record the final optimization result. Fig. 5 displays the GT value grid (circles) and each corresponding optimization result (diamond). From this we can see in most cases the estimated parameters are close to the GT values. Even in test cases where the noise is significantly large (e.g., $v = 9, w = 6.5$) the BO result is still robust. However, we find that when the true noise parameters are large, it is better to increase the TPBO initial sample number and iterations to ensure good results.

B. 2-D Tracking

In this section, we show that TPBO is robust in higher dimensional problems for a system with four noise parameters. We implement a 2-D tracking system where the target state is $\mathbf{x} = [x, y, \dot{x}, \dot{y}]^T$. We assume the same control input for $\dot{x}$ and $\dot{y}$ as in the previous mass-spring-damper system, and add white GP noise to these states. In addition, we add white Gaussian measurement noise to a position sensor on x and y. Thus, our continuous system model is defined by (35) and the closed form discrete time model is (37). In the continuous time system model, we define the GT process and measurement noise parameter $v_0 = 1$, $v_1 = 2$ and $w_0 = 0.2$, $w_1 = 0.1$.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T, \quad (35)$$

$$\boldsymbol{\Gamma} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} v_0 & 0 \\ 0 & v_1 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} w_0 & 0 \\ 0 & w_1 \end{bmatrix}. \quad (36)$$

TABLE I
Mass-Spring-Damping Auto-Tuning Results

	TPBO with C_{NIS}		TPBO with J_{NIS}		GPBO from [14]		Downhill Simplex from [16]		GT
	v	w	v	w	v	w	v	w	
Median	**1.004**	**0.0998**	1.017	0.0995	3.019	0.146	0.29	0.028	
Variance	**0.003**	3.13e-6	0.005	**3.11e-6**	2.362	1.33e-4	0.39	0.003	$v = 1,\ w = 0.1$
Mean	**1.0189**	**0.0997**	1.030	0.0993	3.046	0.0976	0.602	0.018	

The bold entities mean the estimated optimal value is more close to the groundtruth with smaller covariance.
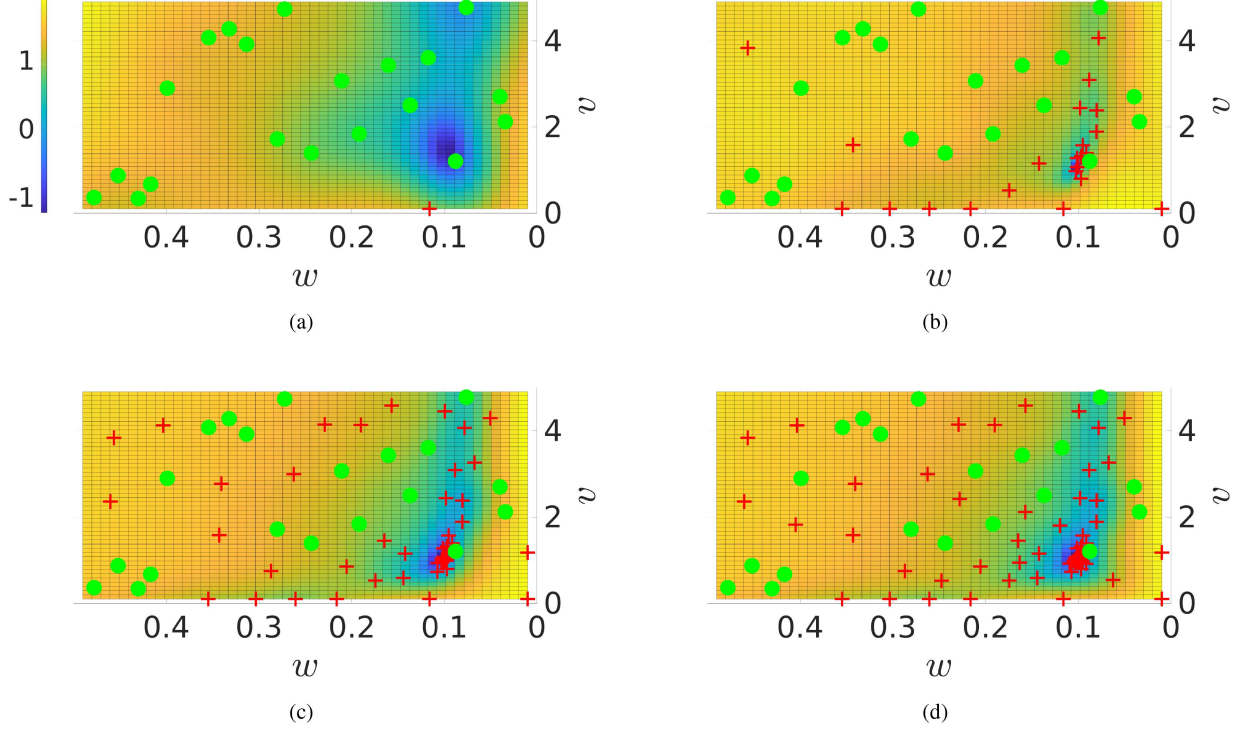

(a)

(b)

(c)

(d)

Fig. 3. TPBO surrogate model for C_{NIS} cost (color map), showing initial random sample points (green dots) and best estimate (red crosses) inferred by TPBO as the number of iterations increase from the start 1 iteration (a) to 30 iterations (b), 70 iterations (c) and 100 iterations (d), where sampled points converge near GT $v = 1$ and $w = 0.1$ after 70 iterations.

TABLE II
2-D Target Tracking Auto-Tuning Results

	TPBO with C_{NIS}				TPBO with J_{NIS}				GPBO from [14]				Downhill Simplex from [16]				GT
	v_0	v_1	w_0	w_1	v_0	v_1	w_0	w_1	v_0	v_1	w_0	w_1	v_0	v_1	w_0	w_1	
Med	**1.43**	**2.14**	**0.20**	**0.096**	1.61	1.41	0.14	0.16	3.377	3.378	0.150	0.152	0.73	0.65	0.06	0.08	$v_0, v_1 = 1, 2$
Var	**0.24**	**0.43**	**0.002**	**3.2e-4**	0.88	0.68	0.005	0.02	1.997	2.510	0.005	0.024	0.33	0.33	0.002	0.004	$w_0, w_1 = 0.2, 0.1$
Mean	**1.49**	2.40	**0.21**	**0.098**	1.73	**1.63**	0.17	0.21	3.32	3.18	0.17	0.23	0.72	0.78	0.06	0.07	

The bold entities mean the estimated optimal value is more close to the groundtruth with smaller covariance.

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0.5\Delta t^2 \\ 0.5\Delta t^2 \\ \Delta t \\ \Delta t \end{bmatrix} \quad (37)$$

$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^3}{3}v_0 & 0 & \frac{\Delta t^2}{2}v_0 & 0 \\ 0 & \frac{\Delta t^3}{3}v_1 & 0 & \frac{\Delta t^2}{2}v_1 \\ \frac{\Delta t^2}{2}v_0 & 0 & v_0\Delta t & 0 \\ 0 & \frac{\Delta t^2}{2}v_1 & 0 & v_1\Delta t \end{bmatrix}. \quad (38)$$

We apply the four optimization methods for 50 independent trials and the results are shown in Table II, where for clarity we just show the median values and variances.

For this problem, the DOF for the measurement innovation and the state error Chi-square distributions are 2 and 4, respectively. Using (11) and (22), we see that a consistent filter must have

$$\mathbb{E}[\bar{\epsilon}_{\mathbf{x},k}] \approx 4, \qquad \mathbb{E}[\bar{\epsilon}_{\mathbf{z},k}] \approx 2,$$
$$\mathbb{E}[\bar{\epsilon}_{\mathbf{x},k}\bar{\epsilon}_{\mathbf{x},k}] \approx 8, \quad \mathbb{E}[\bar{\epsilon}_{\mathbf{z},k}\bar{\epsilon}_{\mathbf{z},k}] \approx 4.$$
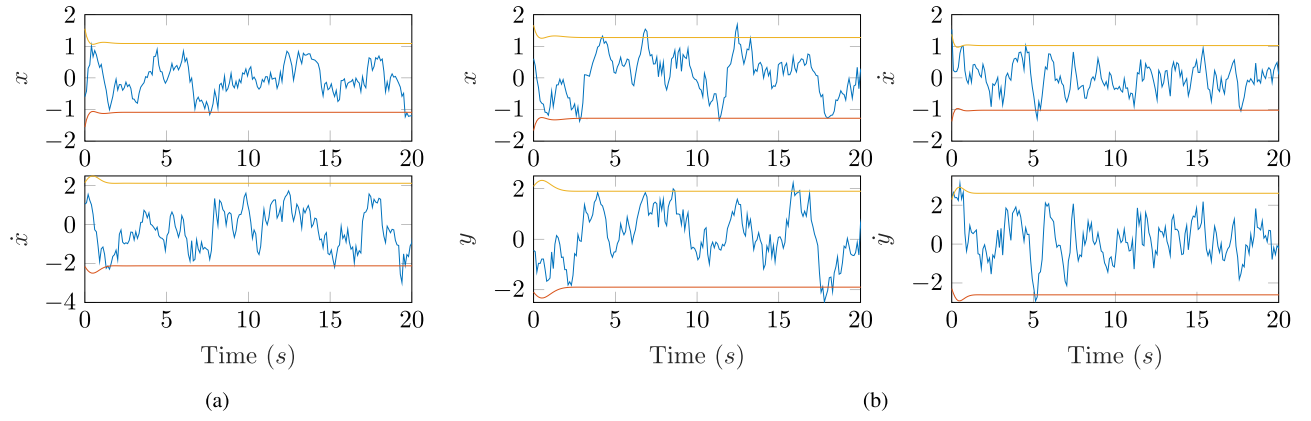
Fig. 4. State errors (vertical axis) versus time (horizontal axis) for (a) mass-spring-damper, and (b) 2-D tracking. Orange lines: 2σ bounds. Blue line: Error between estimated and true states in each step of the Kalman filter. If the system is consistent, around 95% of state errors should be within the 2σ range.
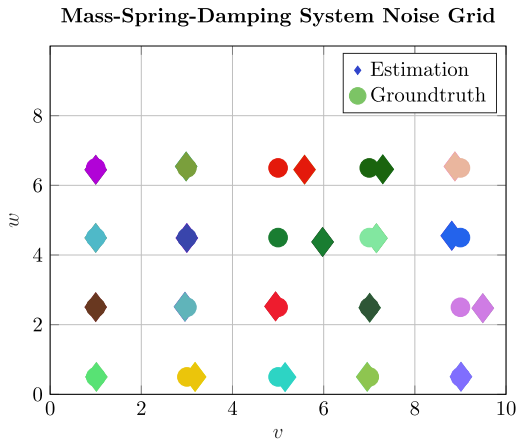

Fig. 5. Various GT pairs (v, w) for the mass-spring-damping system (circles) and TPBO tuning results (diamonds). In this case, w is highly observable, which results in more system sensitivity than v. The results are almost all close to GT; even in tough cases, such as $v = 5$, $w = 4.5$, the TPBO result still passes the χ^2 test.

To validate whether these hold for TPBO, GPBO, and downhill simplex, we apply each method's tuning results to a Kalman filter again with 120 time steps (with sample time $\Delta t = 0.1$ s and GT values $[v_0, v_1] = [1, 2]$, $[w_0, w_1] = [0.2, 0.1]$) and record the statistics for $\mathbb{E}[\bar{\epsilon}_{\mathbf{z},k}]$, $\mathbb{E}[\bar{\epsilon}_{\mathbf{x},k}]$, $\mathbb{E}[\bar{\epsilon}_{\mathbf{z},k}\bar{\epsilon}_{\mathbf{z},k}]$, and $\mathbb{E}[\bar{\epsilon}_{\mathbf{x},k}\bar{\epsilon}_{\mathbf{x},k}]$. The results across 50 such optimization runs are shown in Fig. 6 for TPBO with C_{NIS}, TPBO with J_{NIS}, and GPBO from [14] using J_{NIS} with only one time discretization interval. In addition, since J_{NIS} only uses the mean term in C_{NIS}, we provide results where only the variance term $\tilde{S}_z$ in (26) is used as a new cost function called V_{NIS}. The "GT" label in each plot denotes the distributions for NEES and NIS statistics produced by a Kalman filter across 50 runs. Note that NEES and NIS statistics were not consistent for the downhill simplex method and resulted in large standard deviations; since their scale makes it difficult to visualize the other results, the downhill simplex results are not included on the box plots.

In Fig. 6(a), we can see that TPBO and GPBO with J_{NIS} have the most stable mean NIS value and most closely match the GT mean NIS value, followed by TPBO with C_{NIS}. This is not surprising because minimizing J_{NIS} corresponds to matching the true mean NIS value. Although the mean NIS value from the TBPO C_{NIS} optimization result is larger than results for J_{NIS} from TBPO and GPBO, in reality, the difference is negligible. In contrast, V_{NIS} results in a large mean NIS value because it only considers the variance term in C_{NIS}.

Fig. 6(b) similarly shows the resulting mean NEES values obtained by all methods. The TBPO J_{NIS} method has the best median value but the TBPO C_{NIS} method's NEES value is both stable and close to the expected value of 4. The VNIS and GPBO methods lead to high variance, and the GPBO median is heavily biased.

The resulting NIS and NEES variances are shown in Fig. 6(c) and (d), where we can see that the TBPO C_{NIS} method's NEES and NIS variances are again quite robust and close to the ideal expectation values, which means the χ^2 consistency constraints are better maintained. V_{NIS} gives a stable NIS variance value as expected, but leads to a less stable NEES variance. TBPO and GPBO with J_{NIS} return biased and more volatile results for the NIS and NEES variances, thus, violating the consistency constraints.

The key takeaway from these results is that auto-tuning via TPBO with the C_{NIS} cost stably produces NIS and NEES variances that are close to their ideal expected values, and also stably produces correct expected NIS and NEES means. In this way, TPBO with C_{NIS} maintains the χ^2 consistency constraints and results in a filter with noise parameters that lead to uncertainty estimates accurately describing the actual estimation error statistics. On the other hand, the J_{NIS} or V_{NIS} cost functions for auto-tuning result in local minima that can only maintain either a stable NIS mean
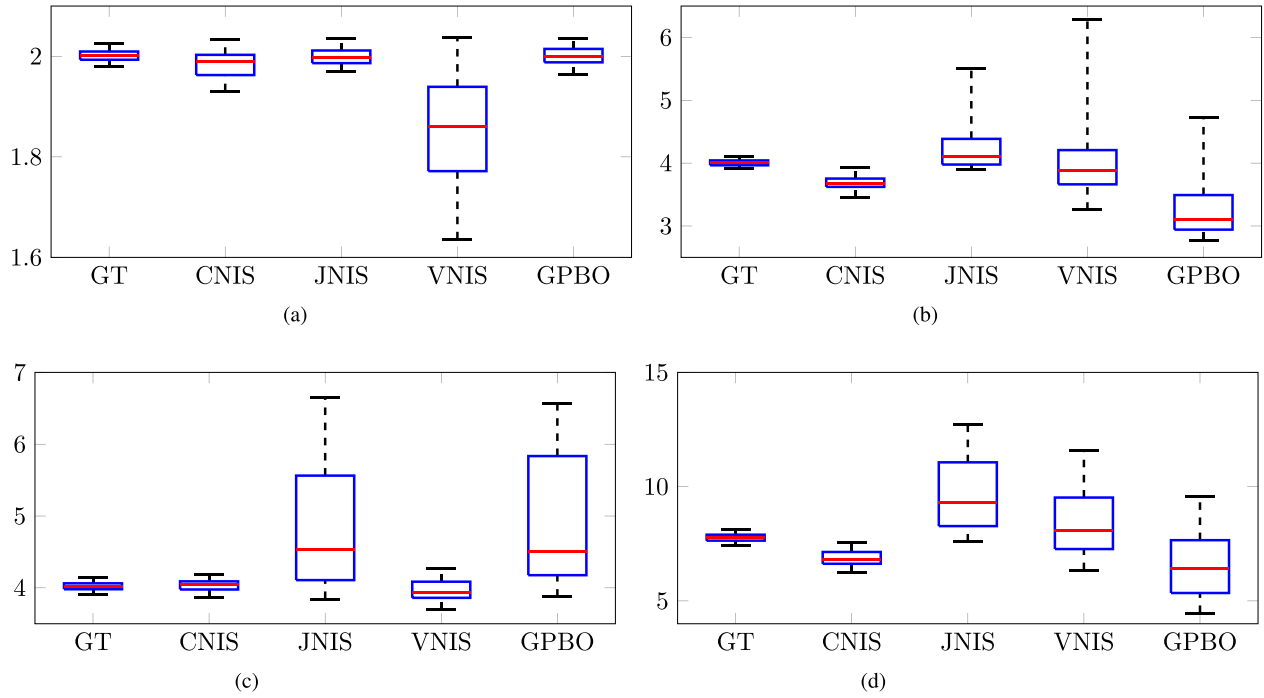
Fig. 6. Statistical summaries over 50 optimization runs for $\mathbb{E}[\bar{\epsilon}_{\mathbf{z},k}]$, $\mathbb{E}[\bar{\epsilon}_{\mathbf{x},k}]$, $\mathbb{E}[\bar{\epsilon}_{\mathbf{z},k}\bar{\epsilon}_{\mathbf{z},k}]$, and $\mathbb{E}[\bar{\epsilon}_{\mathbf{x},k}\bar{\epsilon}_{\mathbf{x},k}]$ when applying results of Kalman filter auto-tuning methods. C_{NIS} has the lowest overall error and variation in the means and variances of the NIS and NEES with respect to GT and, therefore, best maintains the expected χ^2 distributions for consistency. (a) NIS, (b) NEES, (c) NIS variance, (d) NEES variance.
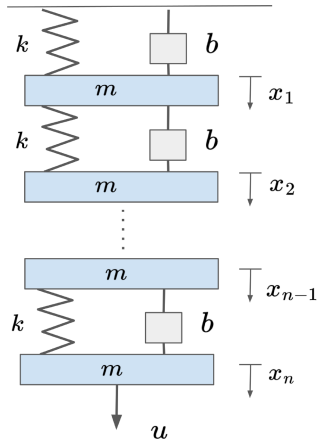

Fig. 7. Cascade mass-spring-damper system with $2n$ states.

can observe whether the parameter optimization meets the χ^2 constraints and, thus, if the resulting filter will remain consistent. The state estimation errors and 2σ bounds are plotted against time for the filter tuned via TPBO with C_{NIS} are shown in Fig. 4(b). Despite the fact that the tuned v_0 is biased from the GT, the resulting estimator performs well and indeed remains consistent.

C. Cascade Mass-Spring-Damper System

A cascade of n mass-spring-dampers is used to produce an even higher dimensional application, shown in Fig. 7. In the following test, $n = 3$ and the state then is $\mathbf{x} = [x_0, \dot{x}_0, x_1, \dot{x}_1, x_2, \dot{x}_2]$. The continuous-time model is

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{2k}{m} & -\frac{2b}{m} & \frac{k}{m} & \frac{b}{m} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{k}{m} & \frac{b}{m} & -\frac{2k}{m} & -\frac{2b}{m} & \frac{k}{m} & \frac{b}{m} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{k}{m} & \frac{b}{m} & -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \quad (39)$$

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T \quad (40)$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (41)$$

or NIS variance, instead of overall χ^2 consistency (this phenomenon can further be seen in Section VI-C).

Table II quantitatively shows that the TBPO C_{NIS} parameter optimization results are quite reasonable, and generally superior to those produced by the other methods. However, when comparing the results with the 1-D system, we can see the resulting v_0 mean and median for all method deviates noticeably from the GT value, resulting in relatively large variances for this parameter. For this 2-D tracking problem, good filtering performance is still achievable even when one of the tuning variables v_0, v_1, w_0, w_1 is not close to the GT. In these cases, we still want to know if the resulting filter satisfies the filter consistency requirement. From Fig. 6, we
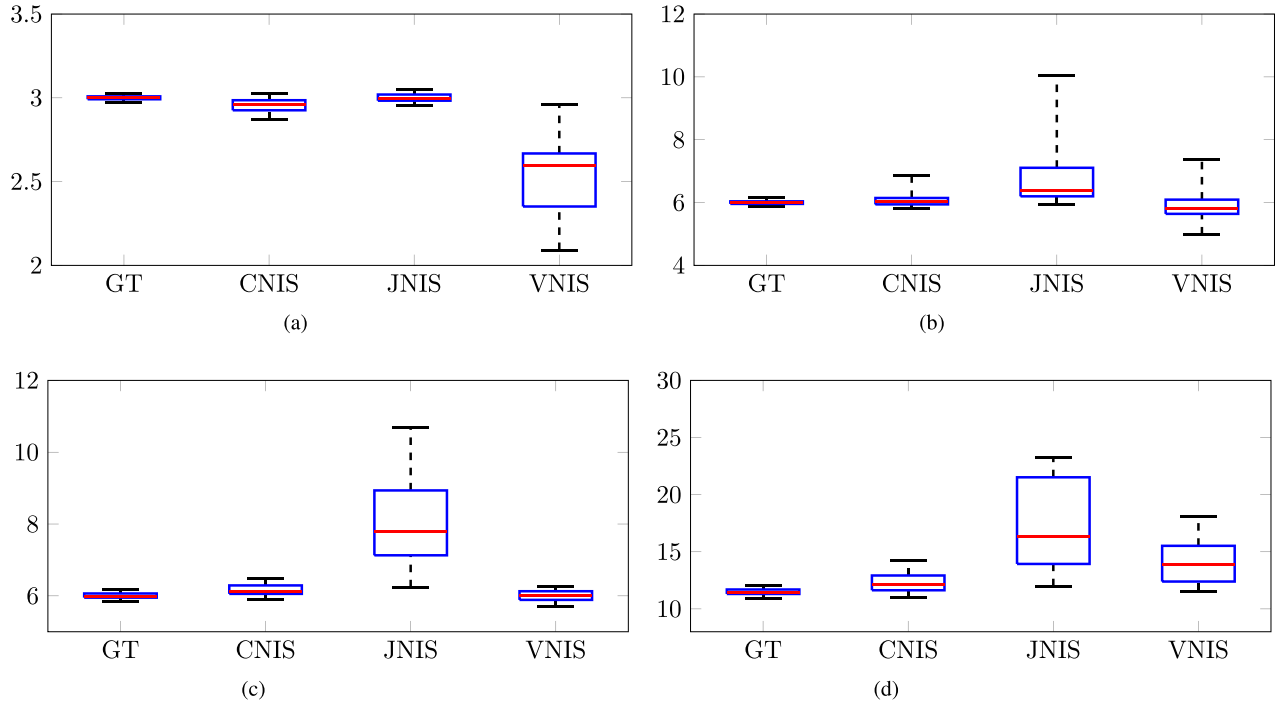
Fig. 8. Similar to Fig. 6, box plots of 50 independent optimizations are shown based on the various methods and then evaluate the resulting filters with optimized noise parameters. The J_{NIS} and V_{NIS} results' deviation from the GT shows the importance of considering both variance and mean term in C_{NIS}. (a) NIS. (b) NEES. (c) NIS variance. (d) NEES variance.
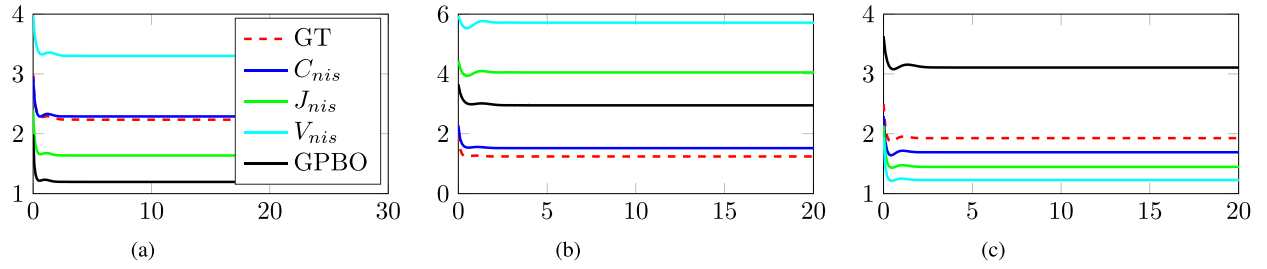

Fig. 9. Innovation covariance matrix diagonal values over time for Kalman filters produced by each auto-tuning method versus GT innovation covariance (GT, red dashed line). (a) $\mathbf{S}_{k|k-1}(0,0)$. (b) $\mathbf{S}_{k|k-1}(1,1)$. (c) $\mathbf{S}_{k|k-1}(2,2)$.

$$\mathbf{\Gamma} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T \quad (42)$$

$$\mathbf{V} = \begin{bmatrix} v_0 & 0 & 0 \\ 0 & v_1 & 0 \\ 0 & 0 & v_2 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} w_0 & 0 & 0 \\ 0 & w_1 & 0 \\ 0 & 0 & w_2 \end{bmatrix}. \quad (43)$$

The same control input as before is applied. We set the GT $[v_0, v_1, v_2] = [1, 2, 3]$ and $[w_0, w_1, w_2] = [0.2, 0.1, 0.15]$. The initial sample number for Bayes optimization is 600 and the iteration count is 1200; the search range for all the process noise parameters is $v \in [0.1, 5]$, and for the measurement noise parameters is $w \in [0.01, 1]$. We run 50 independent optimizations using the C_{NIS}, J_{NIS}, and V_{NIS} cost function for TPBO and GPBO with J_{NIS}. Similar to Fig. 6, we run the filter with the optimized noise parameters and record $\mathbb{E}[\bar{\epsilon}_{\mathbf{z},k}]$, $\mathbb{E}[\bar{\epsilon}_{\mathbf{x},k}]$, $\mathbb{E}[\bar{\epsilon}_{\mathbf{z},k}\bar{\epsilon}_{\mathbf{z},k}]$, and

$\mathbb{E}[\bar{\epsilon}_{\mathbf{x},k}\bar{\epsilon}_{\mathbf{x},k}]$, where a consistent filter should satisfy

$$\mathbb{E}\left[\bar{\epsilon}_{\mathbf{x},k}\right] \approx 6, \qquad \mathbb{E}\left[\bar{\epsilon}_{\mathbf{z},k}\right] \approx 3$$
$$\mathbb{E}\left[\bar{\epsilon}_{\mathbf{x},k}\bar{\epsilon}_{\mathbf{x},k}\right] \approx 12 \quad \mathbb{E}\left[\bar{\epsilon}_{\mathbf{z},k}\bar{\epsilon}_{\mathbf{z},k}\right] \approx 6.$$

From Fig. 8, we observe that the J_{NIS} cost function with TPBO can only maintain the correct NIS mean but not the NIS variance, while the V_{NIS} cost function for TPBO can only maintain the correct NIS variance but not the NIS mean. The C_{NIS} results are the closest to the GT in all four subplots. Also note that for this 6-D optimization example, the deviation from the GT of the NIS variance is even larger for the J_{NIS} results. This implies that there might be more local minima when we implement a higher dimension system; this illustrates the significance of using C_{NIS} to lock down the correct χ^2 error distributions and tune the filter correctly. Note that the GPBO results using the method of [14] are ignored here, since they produce quite large errors in the NIS and NEES variances. This

effect underscores the importance of using multiple time discretization intervals for higher dimension systems.

Finally, Fig. 9 shows the innovation covariance history $\mathbf{S}_{k|k-1}$ for the Kalman filters produced by each auto-tuning method, where the innovation covariance for a correctly tuned filter should be close to the GT. We randomly choose one of the fifty optimization results for the C_{NIS}, J_{NIS}, V_{NIS}, and GPBO methods and run the resulting Kalman filters to record the innovation covariance history. For simplification, only the diagonal values of $\mathbf{S}_{k|k-1}$ are plotted, corresponding to innovation variances of the sensed position states $\mathbf{z}_=[x_0, x_1, x_2]$. These plots show that TPBO using the multitime discretization C_{NIS} cost in (27) produces results closest to the GT.

VII. CONCLUSION

This article derived new cost metrics for BO auto-tuning of Kalman filter process noise and sensor noise covariance parameters, using either simulated or real data. Unlike previously developed cost metrics, the new cost metrics accurately account for both the mean and the variance of underlying state and measurement error statistics during the tuning process. When combined with student's-t process surrogate regression models within BO, the new metrics improve the overall robustness of the parameter search, particularly when multiple time step discretizations are used for state space models that are converted from continuous time to discrete time. Three examples showed that the developed approach can scale to noise parameter searches in nontrivial high-dimensional problems, converges toward optimal parameters faster and more stably than alternative black box optimization methods, and produces statistically consistent estimation results. The ability of our approach to reliably auto-tune Kalman filters using only recorded sensor data is a key capability in practical applications where obtaining GT data from truth model simulations or high-fidelity sensors is usually costly.

This work focused on auto-tuning of Kalman filters for linear systems to establish fundamentally useful insights that can be applied and developed further for a wide range of dynamic state estimation problems. Although linear estimators are still quite useful on their own in applications like target tracking, a logical direction for extending this work includes adaptation of the new metrics and TPBO algorithm to auto-tuning of nonlinear estimators, e.g., based on extended or unscented Kalman filters [2], [44], [45] or batch algorithms, which are important for many real-world applications. For example, the NIS-based version of this work has already been applied to extrinsic parameter calibration in [46], where the pose of a camera relative to other sensors mounted on a mobile robot was estimated using a batch processing algorithm. There are some limitations of the current method. First, in reality, sometimes the electrical digital-to-analog converter allows us to choose different sample time Δt but some of the equipments do not allow us to perform this action. Second, We can consider more practical situations such as missing measurements to extend the work. The effects of nonlinearity and nonGaussian error distributions lead to many challenging opportunities for exploring effective black box estimator auto-tuning strategies. Third, the current time complexity of the TPBO is $O(n^2)$, for higher dimensional system it might be slow due to a large iteration number. The cascade mass-spring-damping system can cost one to two hours to optimize. This is due to the frequent large matrix operation, such as inversion and can be optimized in the future.

APPENDIX A CONTINUOUS TO DISCRETE-TIME TRANSFORMATION

Although the stochastic linear systems we use are discrete time, these are derived from a continuous time formulation. The linear stochastic system, which evolves according to the continuous-time model in (1) can be discretized to timesteps of length Δt via [47]

$$\mathbf{F} = e^{\mathbf{A}\Delta t}, \quad \mathbf{B} = \int_0^{\Delta t} e^{\mathbf{A}\tau}\mathrm{d}\tau, \quad \mathbf{R} = \frac{\mathbf{W}}{\Delta t}$$

$$\mathbf{Q} = \int_0^{\Delta t} e^{\mathbf{A}\tau}\mathbf{\Gamma}\mathbf{V}\mathbf{\Gamma}^T e^{\mathbf{A}^T\tau}\mathrm{d}\tau \tag{44}$$

where Van Loan's method provides a closed-form solution for $\mathbf{Q}$ in (44) [35].

APPENDIX B TP EXPECTED IMPROVEMENT

In this appendix, we expand the terms which appear in (34).

u and σ are the mean and variance of the conditional student's-t distribution of $\mathbf{q}_{n+1}$, which is presented below in (47). $\Psi(\cdot)$ and $\psi(\cdot)$ are the CDF and PDF of the standard student-t distribution $\mathrm{MVT}_1(v, 0, 1)$. The conditional MVT distribution is similar to the conditional multivariate Gaussian distribution: if we have $\mathbf{q}_{1:n+1}$ and $y(\mathbf{q}_{1:n+1})$ described by a multivariate MVT PDF, then

$$\begin{bmatrix} y(\mathbf{q}_{1:n}) \\ y(\mathbf{q}_{n+1}) \end{bmatrix} \sim \mathrm{MVT}_{n+1}\left(v+1, \begin{bmatrix} \Phi(\mathbf{q}_{1:n}) \\ \Phi(\mathbf{q}_{n+1}) \end{bmatrix} \right|$$
$$\left| \begin{bmatrix} K(\mathbf{q}_{1:n}, \mathbf{q}_{1:n}) & K(\mathbf{q}_{1:n}, \mathbf{q}_{n+1}) \\ K(\mathbf{q}_{n+1}, \mathbf{q}_{1:n}) & K(\mathbf{q}_{n+1}, \mathbf{q}_{n+1}) \end{bmatrix} \right) \tag{45}$$

where $K(\mathbf{q}_{1:n}, \mathbf{q}_{1:n})$ is the same as (30), $K(\mathbf{q}_{1:n}, \mathbf{q}_{n+1}) = [k(\mathbf{q}_1, \mathbf{q}_{n+1}), \ldots, k(\mathbf{q}_n, \mathbf{q}_{n+1})]^T$, and $K(\mathbf{q}_{n+1}, \mathbf{q}_{n+1}) = k(\mathbf{q}_{n+1}, \mathbf{q}_{n+1})$. Equation (45) can be written more simply as

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim \mathrm{MVT}_{n+1}\left(v+1, \begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}, \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \right). \tag{46}$$

The conditional student-t distribution of $y(\mathbf{q}_{n+1})$ is then given by [48]

$$y(\mathbf{q}_{n+1}|\mathbf{q}_{1:n}, y(\mathbf{q}_{1:n})) \sim MVT_1(v+n, u, \sigma)$$
$$u = \Phi_2 + K_{21}K_{11}^{-1}(y_1 - \Phi_1)$$
$$\sigma = \frac{v+d}{v+n}K_{22} - K_{21}K_{11}^{-1}K_{12}$$
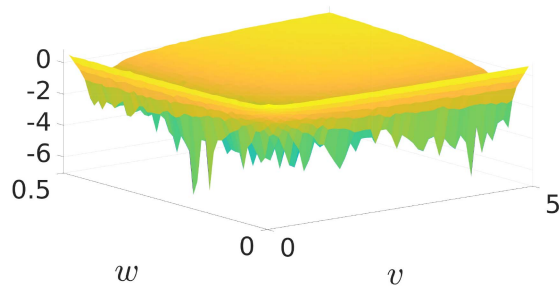$$d = (y_1 - \Phi_1)^T K_{11}^{-1}(y_1 - \Phi_1). \tag{47}$$

Fig. 10. w- and v-axis is the noise parameter we want to optimize. z-axis is the cost function.

Since the value of the prior mean function does not change the final result [49], we simplify the equations by setting $\Phi_1 = \mathbf{0}$ and $\Phi_2 = 0$.

APPENDIX C NEES/NIS-BASED FUNCTION WITH LOCAL MINIMALS

In this appendix, we visualize an example that the NEES/NIS based cost function, which has many local minima. We uniformly sample $w \in [0.01, 0.5]$ and $v \in [0.1, 5]$ fifty times, use each $[w, v]$ pair to run the Kalman filter of the mass-spring-damping model and record the cost. As a result, we generate a 50 by 50 grid plot. In Fig. 10, we show the side view of the 3-D plot, where the w- and v-axis is the noise parameter we want to optimize. The z-axis is the cost function. In this plot, it is easy to visualize that the cost has a various local minimas. This encourages a global search algorithm like TPBO to perform the optimization.

ACKNOWLEDGMENT

REFERENCES
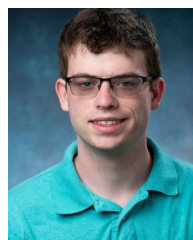
[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960.

[2] Y. B.-Shalom, X. Li, and T. Kirubarajan, *Estimation With Applications to Navigation and Tracking*. New York, NY, USA: Wiley, 2001, pp. 57–59.

[3] J. Srivastava, S. Zacks, and P. Sen, "Journal of statistical planning and inference," *Ency. Stat. Sci.*, vol. 6, pp. 361–362, 2004.

[4] A. Shah, A. G. Wilson, and Z. Ghahramani, "Bayesian optimization using student-t processes," in *Proc. NIPS Workshop Bayesian Optimisation*, 2013.

[5] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2012, pp. 2951–2959.

[6] S. Särkkä and A. Solin, *Applied Stochastic Differential Equations*, vol. 10. Cambridge, U.K.: Cambridge Univ. Press, 2019, pp. 198–200.

[7] C.-T. Chen, *Linear System Theory and Design*. Philadelphia, PA, USA: Saunders, 1984.

[8] H. K. Khalil, *Nonlinear Systems*; 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[9] M. Grewal and A. Andrews, "Van Loan's method for computing Q_k from continuous Q," in *Proc. Kalman Filtering: Theory Pract. MATLAB*, 2015, pp. 150–152.

[10] N. Wahlström, P. Axelsson, and F. Gustafsson, "Discretizing stochastic dynamical systems using Lyapunov equations," *IFAC Proc. Volumes*, vol. 47, no. 3, pp. 3726–3731, 2014.

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

[12] A. Banerjee, U. Chitnis, S. Jadhav, J. Bhawalkar, and S. Chaudhury, "Hypothesis testing, type i and type ii errors," *Ind. Psychiatry J.*, vol. 18, no. 2, 2009, Art. no. 127.

[13] S. Huang and G. Dissanayake, "Convergence analysis for extended Kalman filter based SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 412–417.

[14] Z. Chen, C. Heckman, S. Julier, and N. Ahmed, "Weak in the NEES?: Auto-tuning Kalman filters with Bayesian optimization," in *Proc. 21st Int. Conf. Inf. Fusion*, 2018, pp. 1072–1079.

[15] L. Cai, B. Boyacıoğlu, S. E. Webster, L. V. Uffelen, and K. Morgansen, "Towards auto-tuning of Kalman filters for underwater gliders based on consistency metrics," in *Proc. OCEANS MTS/IEEE SEATTLE*, 2019, pp. 1–6.

[16] T. D. Powell, "Automated tuning of an extended Kalman filter using the downhill simplex algorithm," *J. Guid., Control, Dyn.*, vol. 25, no. 5, pp. 901–908, 2002.

[17] Y. Morales, E. Takeuchi, and T. Tsubouchi, "Vehicle localization in outdoor woodland environments with sensor fault detection," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2008, pp. 449–454.

[18] B. Altinöz and D. Ünsal, "Determining efficient temperature test points for IMU calibration," in *Proc. IEEE/ION Position, Location, Navigation Symp.*, 2018, pp. 552–556.

[19] H. Dette, A. Pepelyshev, and A. Zhigljavsky, "Optimal designs in regression with correlated errors," *Ann. Statist.*, vol. 44, no. 1, 2016, Art. no. 113.

[20] L. Wanninger, "Real-time differential GPS error modelling in regional reference station networks," in *Advances in Positioning and Reference Frames*. Berlin, Germany: Springer, 1998, pp. 86–92.

[21] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[22] S. T. Barratt and S. P. Boyd, "Fitting a Kalman smoother to data," in *Proc. Amer. Control Conf.*, 2020, pp. 1526–1531.

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

[24] L. Zhang, D. Sidoti, A. Bienkowski, K. R. Pattipati, Y. B.-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.

[25] 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," *Int. J. Adaptive Control Signal Process.*, vol. 31, no. 11, pp. 1505–1543, 2017.

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

[27] Q. Xia, M. Rao, Y. Ying, and X. Shen, "Adaptive fading Kalman filter with an application," *Automatica*, vol. 30, no. 8, pp. 1333–1338, 1994.

[28] S. T. Barratt and S. P. Boyd, "Fitting a Kalman smoother to data," in *Proc. Amer. Control Conf.*, 2020, pp. 1526–1531.

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

[30] M. Poncela, P. Poncela, and J. R. Perán, "Automatic tuning of Kalman filters by maximum likelihood methods for wind energy forecasting," *Appl. Energy*, vol. 108, pp. 349–362, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261913002377

[31] T. Ting, K. L. Man, E. G. Lim, and M. Leach, "Tuning of Kalman filter parameters via genetic algorithm for state-of-charge estimation in battery management system," *Sci. World J.*, vol. 2014, 2014, Art. no. 176052.

[32] J. Yan, D. Yuan, X. Xing, and Q. Jia, "Kalman filtering parameter optimization techniques based on genetic algorithm," in *Proc. IEEE Int. Conf. Autom. Logistics*, 2008, pp. 1717–1720.

[33] Y. Oshman and I. Shaviv, "Optimal tuning of a Kalman filter using genetic algorithms," in *Proc. AIAA Guid., Navigation, Control Conf. Exhibit*, 2000, Art. no. 4558.

[34] Z. Chen, C. Heckman, S. Julier, and N. Ahmed, "Time dependence in Kalman filter tuning," in *Proc. IEEE 24th Int. Conf. Inf. Fusion*, 2021, pp. 1–8.

[35] R. G. Brown and P. Y. Hwang, *Introduction to Random Signals and Applied Kalman Filtering: With MATLAB Exercises*. Hoboken, NJ, USA: Wiley, 2012.

[36] M. Pelikan, D. E. Goldberg, and E. C.-Paz, "Boa: The Bayesian optimization algorithm," in *Proc. 1st Annu. Conf. Genet. Evol. Comput.*, 1999, vol. 1, pp. 525–532.

[37] A. Shah, A. Wilson, and Z. Ghahramani, "Student-T processes as alternatives to Gaussian processes," in *Proc. Artif. Intell. Statist.*, 2014, pp. 877–885.

[38] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning, Ser. Adaptive Computation and Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[39] 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.

[40] M. G. Genton, "Classes of kernels for machine learning: A statistics perspective," *J. Mach. Learn. Res.*, vol. 2, pp. 299–312, 2001.

[41] B. D. Tracey and D. Wolpert, "Upgrading from Gaussian processes to student'st processes," in *Proc. AIAA Non-Deterministic Approaches Conf.*, 2018, Art. no. 1659.

[42] D. E. Finkel, "Direct optimization algorithm user guide," *Center Res. Sci. Comput., North Carolina State Univ.*, vol. 2, pp. 1–14, 2003.

[43] R. M.-Cantin, "Bayesopt: A Bayesian optimization library for nonlinear optimization, experimental design and bandits," *J. Mach. Learn. Res.*, vol. 15, pp. 3915–3919, 2014. [Online]. Available: http://jmlr.org/papers/v15/martinezcantin14a.html

[44] E. A. Wan and R. V. D. Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proc. IEEE Adaptive Syst. Signal Process., Commun., Control Symp.*, 2000, pp. 153–158.

[45] Z. Chen, N. Ahmed, S. Julier, and C. Heckman, "Kalman filter tuning with Bayesian optimization," 2019, *arXiv:1912.08601*.

[46] Z. Chen, "Visual-inertial SLAM extrinsic parameter calibration based on Bayesian optimization," Ph.D. dissertation, Univ. of Colorado at Boulder, Boulder, CO, USA, 2018.

[47] D. Simon, *Optimal State Estimation: Kalman, H. Infinity, and Nonlinear Approaches*. Hoboken, NJ, USA: Wiley, 2006.

[48] P. Ding, "On the conditional distribution of the multivariate T distribution," *Amer. Statistician*, vol. 70, no. 3, pp. 293–295, 2016.

[49] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," 2010, *arXiv:1012.2599*.

Harel Biggie (Student Member, IEEE) received the B.S. degree in electrical and computer engineering from the University of Rochester, Rochester, NY, USA, in 2018. He is currently working toward the Ph.D. degree in computer science with the Department of Computer Science, University of Colorado Boulder, Boulder, CO, USA.

His research interests include natural language-based robot navigation, robotic exploration algorithms for unknown environments, and robust localization and mapping algorithms.

Nisar Ahmed (Member, IEEE), received the B.S. degree in engineering from Cooper Union, New York, NY, USA, in 2006, and the Ph.D. degree in mechanical engineering from Cornell University, Ithaca, NY, USA, in 2012.

He is currently an Associate Professor and H.J. Smead Faculty Fellow with the Smead Aerospace Engineering Sciences Department, University of Colorado Boulder, Boulder, CO, USA. He is the Director with the Cooperative Human-Robot Intelligence (COHRINT) Lab. His research interests include probabilistic modeling, estimation and control of autonomous systems, human-robot/machine interaction, sensor fusion, and decision-making under uncertainty.

Simon Julier (Member, IEEE) received the Ph.D. degree in computer science from the University of Oxford, Wellington Square, Oxford, U.K., in 1997.

From 2005 to 2006, he was the Associate Director of the 3DMVEL. For nine years, he was with the 3-D Mixed and Virtual Environments Laboratory, Naval Research Laboratory, Washington, DC, USA, where he was PI of the Battlefield Augmented Reality System (BARS): a research effort to develop man-wearable systems for providing situational awareness information. He is currently a Professor of Computer Science Department, University College London (UCL), London, U.K. His research interests include user interfaces, distributed data fusion, nonlinear estimation, and simultaneous localization and mapping.

Zhaozhong Chen (Member, IEEE) received the B.S. degree in electrical engineering from Tianjin University, Tianjin, China, in 2016, and the Ph.D. degree in electrical engineering from the University of Colorado Boulder, Boulder, CO, USA, in 2021, under the supervision of Prof. Christoffer Heckman.

His research interests include 3-D reconstruction, visual odometry, multi-model navigation, neural rendering, state estimation, and multi-sensor calibration.

Christoffer Heckman (Senior Member, IEEE) received the B.S. in mechanical engineering from the University of California Berkeley, Berkeley, CA, USA, in 2008, and the Ph.D. degree in theoretical and applied mechanics from Cornell University, Ithaca, NY, USA, in 2012.

He was a Postdoctoral Researcher with the Naval Research Laboratory, Washington, DC, USA. He is currently an Assistant Professor with the Department of Computer Science, University of Colorado at Boulder and the Jacques I. Pankove Faculty Fellow with the College of Engineering and Applied Science. His research interests include autonomy, perception, and field robotics.