2017 IEEE 56th Annual Conference on Decision and Control (CDC)
December 12-15, 2017, Melbourne, Australia

Particle-Filter-Enabled Real-Time Sensor Fault Detection Without a Model of Faults

Matthew A. Wright and Roberto Horowitz

Abstract— **We are experiencing an explosion in the amount of sensors measuring our activities and the world around us. These sensors are spread throughout the built environment and can help us perform state estimation and control of related systems, but they are often built and/or maintained by third parties or system users. As a result, by outsourcing system measurement to third parties, the controller must accept their measurements without being able to directly verify the sensors' correct operation. Instead, detection and rejection of measurements from faulty sensors must be done with the raw data only. Towards this goal, we present a method of detecting possibly faulty behavior of sensors. The method does not require that the control designer have any model of faulty sensor behavior. As we discuss, it turns out that the widely-used particle filter state estimation algorithm provides the ingredients necessary for a hypothesis test against all ranges of correct operating behavior, obviating the need for a fault model to compare measurements. We demonstrate the applicability of our method by showing its ability to reject faulty measurements and accuracy in state estimation of a nonlinear vehicle traffic model, without information of generated faulty measurements' characteristics. In our test, we correctly identify nearly 90% of measurements as faulty or non-faulty without having any fault model. This leads to only a 3% increase in state estimation error over a theoretical 100%-accurate fault detector.**

I. Introduction

Much ado has been made about the contemporary explosion of ubiquitous sensors and actuators in engineered artifacts and the built environment. "Big data," the "Internet of Things" - these and other recently-coined, oft-heard phrases call back to the idea that, in the past few years, a sea change has occurred in our ability to measure the world. For control and systems engineers, these new data increase the ability to measure and precisely control built-environment systems, with applications like localized responsive building HVAC regulation, connected vehicles for traffic management and accident reduction, and power systems, up to and including automated coordination between power producers, consumers, and storers in a "smart" grid. When third-party measurements are collected and submitted by end users of Internet-of-Things devices and the like, fusion of these data can enhance the estimation and control of these sorts of highly-sensed and/or highly-actuated systems.

However, these new data can present their own problems. Like any sensor, these third-party sensors cannot always behave perfectly, and data may be collected and submitted by a sensor that is faulty, overly noisy, tampered-with, or otherwise misreporting. This problem of sensor fault detection is a problem as widespread and highly-studied across industries as the use of sensors themselves. Some fault detection methods attempt general applicability to large classes of systems, while others take advantage of application-specific knowledge to detect failures along known failure modes. See [1] for a recent survey that covers both general and specific methods in the literature. For the example application considered in this work, that of vehicle speed measurements for traffic control, we show a general algorithm. Since third-party sensors in this example vary widely, it was necessary to focus on a generally-applicable scheme.

In particular, we make use of the particle filter, a widely-used state estimator for general stochastic nonlinear systems [2]. We are not the first authors to use a particle filter state estimator for sensor fault detection purposes. The particle filter is a flexible and widely-applicable state estimator, and it has been used for fault detection by providing "analytical redundancy" [1]. This means its estimate is used to produce a predicted sensor reading that is compared to received measurements, with any difference being potential evidence of faults. Often, however, it is assumed that the engineer has models for sensor faults, and the particle filter estimate is used to select a maximum-likelihood estimate among the known types of sensor modes (cf. interacting-multiple-model estimators, and e.g., [3]–[6]). Some recent works have considered a more probabilistic handling of potential faults, in keeping with the particle filter's nature as a probabilistic state estimator. In [6], for example, when a sensor measurement diverges from the particle filter estimate, the errors between the observed and predicted measurements over the next few timesteps are collected. After a while, a hypothesis test is performed to determine whether the errors are large and consistent enough to confirm a sensor fault, and which (if any) known fault models are active.

Two common threads among these particle-filter-based methods are assumptions of knowledge of fault models and that the sensors will repeatedly report so that potential faults can be examined over time. However, when dealing with third-party sensors, we cannot guarantee these assumptions: sensors may start and stop reporting uncontrollably, and the engineer may not have the opportunity to examine the sensors and see how they may fail before having to use their readings.

In contrast to these previous works, the present paper presents a method for sensor fault detection where no model of faulty behavior is available. As we shall see, the construction of the particle filter gives us the data to perform a

This work was supported by the California Department of Transportation and the Anselmo Macchi Fellowship.

Mechanical Engineering Department and Partners for Advanced Transportation Technologies, University of California, Berkeley, CA 94720, USA
{mwright, horowitz}@berkeley.edu

particular hypothesis test for sensor fault detection. This test forms the basis of our fault detection method. This particular form of hypothesis test is nonparametric, as opposed to [6]'s parametric hypothesis test, and only needs a model of non-faulty sensor behavior. This method can be used in real-time to detect faults in third-party sensors that may not have a fault model, and may sporadically or unpredictably report.

Section II gives a mathematical statement of the fault detection problem, and Section III discusses the extension of filtering and state estimation to include our fault detection formulation. Section IV introduces the constraint of a lack of a fault model, and outlines where, mathematically, this prevents us from both state estimation and fault detection. Section V introduces the particle filter, and Section VI discusses how its structure allows us to naturally overcome the lack-of-fault-model problem. Section VII describes an example application, where faulty speed measurements from third-party global navigational satellite system (GNSS) devices are detected in a particle filter for estimating vehicle density on a freeway.

II. Problem Statement

Let $x_k \in \mathbb{R}^N$ denote the state vector of our system at time k, and $y_k \in \mathbb{R}^{M_k}$ denote the measurement vector at time k (note that y_k may be of different size at different times k). The state and observation vectors evolve over time through discrete-time stochastic state and output equations, denoted $\mathcal{F}_\theta(\cdot)$ and $\mathcal{G}_\theta(\cdot)$ respectively,

$$\begin{aligned} x_k &= \mathcal{F}_\theta\left(x_{k-1}\right) \\ y_k &= \mathcal{G}_\theta\left(x_k\right), \end{aligned} \tag{1}$$

with θ a parameter vector describing the randomness or process/measurement noise of $\mathcal{F}$ and $\mathcal{G}$. An alternative probabilistic notation may rewrite (1) as

$$X_k|\left(X_{k-1}=x_{k-1}\right) \sim f\left(x_k|x_{k-1},\theta\right) \tag{2a}$$

$$Y_k|\left(X_k=x_k\right) \sim g\left(y_k|x_k,\theta\right), \tag{2b}$$

where X_k (Y_k) denotes a random vector and x_k (y_k) the value of a particular realization. The functions $f(\cdot)$ and $g(\cdot)$ are the probability density functions (PDFs) induced by $\mathcal{F}_\theta(\cdot)$ and $\mathcal{G}_\theta(\cdot)$, respectively. The initial condition of the system, x_0, is assumed fixed or distributed with some known density $p(x_0|\theta)$. More precisely, $f(x_k|x_{k-1},\theta)$ is a Markov transition kernel with a distribution on the random vector $X_k|(X_{k-1}=x_{k-1})$, and $g(y_k|x_k,\theta)$ is a typical observation likelihood. To reduce notational clutter, we omit the symbol θ from now on.

In (2), the observation PDF is a joint PDF over all elements of the observation vector Y_k. This is the most general formulation of this PDF, and allows for all elements of Y_k to be statistically dependent. This would be appropriate, if, for example, the entire vector Y_k was reported by a single sensor, and the noise in measuring one element of Y_k was correlated with the noise in measuring the other elements.

In our setting, however, we are dealing with many different sensors and reporting devices. Say that Y_k contains measurements from m_k different sensors (like M_k (the length of Y_k), m_k may be different at different timesteps k), and let $j \in \{1,\dots,m_k\}$ index the sensors that report at time k (we will not have $m_k = M_k$ if any sensor reports more than one measurement). Say that Y_{k_j} is the random vector that contains the element(s) of Y_k from sensor j at time k. Now, if we assume that individual sensors j have independent measurement noises, we may rewrite (2b) as

$$Y_k|\left(X_k=x_k\right) \sim g\left(y_k|x_k\right) = \prod_{j=1}^{m_k} g_{k_j}\left(y_{k_j}|x_k\right), \tag{3}$$

where y_{k_j} is the measurement(s) received from sensor j at time k (i.e., a realization of the random vector Y_{k_j}), and $g_{k_j}(\cdot)$ is the PDF for Y_{k_j}. Equation (3) is an assumption on the form of the joint sensing PDF and is a common assumption in multisensor filtering and sensor fusion (e.g., [7, (6)], [8, (9)], [9, (35.3)], [10, Section 2.2])). We have factored $g(\cdot)$ into its component per-sensor PDFs.

As written in (3), $g_{k_j}(\cdot)$ is in a general form, with no explicit model of faulty vs. non-faulty measurements. For our purposes, "faulty" means any measurement that does not provide information about x_k and that we would like to discard. We now describe how we handle faulty sensors by explicitly indicating whether sensor j at time k is reporting in either a valid (i.e., as-intended) mode, or a faulty mode. To this end, we introduce a Bernoulli random variable Z_{k_j}, which takes value 1 in the event that a measurement from sensor j is not faulty, and 0 in the event that a measurement is faulty. Written as a probability mass function (PMF),

$$\begin{gathered} p(z_{k_j}|x_k) = \left(\phi_{k_j}(x_k)\right)^{z_{k_j}} \left(1-\phi_{k_j}(x_k)\right)^{\left(1-z_{k_j}\right)} \\ \text{for } z_{k_j} \in \{0,1\}, \end{gathered} \tag{4}$$

where ϕ_{k_j} is a weight equal to the prior probability that sensor j reports a measurement in a valid, as-intended manner. It is a *prior* probability, in that it is the probability that any given measurement is non-faulty before we actually see the measurement. It can be based on prior knowledge, or estimated in real time.

The parameter ϕ_{k_j} being a function of x_k allows for the possibility that the Z_{k_j} are dependent on X_k; that is, that certain areas of the state space for x_k could lead to a greater likelihood of faulty measurements than others. This is the most general formulation, and we may simplify the problem by assuming independence of Z_{k_j} from X_k later.

Now, we can condition $g_{k_j}(\cdot)$ from (3) on Z_{k_j},

$$g_{k_j}(y_{k_j}|z_{k_j},x_k) \triangleq g_{k_j}^{z_{k_j}}(y_{k_j}|x_k), \tag{5}$$

where $g_{k_j}^1(\cdot)$ is the PDF for the valid sensor behavior, and $g_{k_j}^0(\cdot)$ is the PDF for faulty measurements. In practice, we should have a model for $g_{k_j}^1(\cdot)$, the expected behavior of the sensor, but $g_{k_j}^0(\cdot)$ may be unknown if we cannot predict every way in which the sensor may fail and/or the probabilities of each failure mode.

III. Extending filtering to include Z_k

In this Section, we reintroduce recursive, or Bayesian, filtering in a form that includes the variable Z_k we have just

defined. At this point we do not consider the unavailability of $g_{k_j}^0(\cdot)$: that will be considered beginning in Section IV.

A. Reformulation of Bayesian Filtering

In this formulation, detecting a fault for a particular sensor j at time k means determining whether z_{k_j} is equal to 0 or 1. Of course, z_{k_j} is not directly observable, and its value must be estimated alongside x_k from the observed y_{k_j}.

This can be written as a slight extension of traditional recursive Bayesian filtering. This extension is very similar to filtering for a hybrid system as used in, e.g., [4]. Let Z_k represent the collection of random variables Z_{k_j} at time k, and let $y_K = \{y_k, y_{k-1}, \dots, y_1\}$, i.e. the set of all measurements up to time k. Assume that at time k we have an estimate of the PDF $p(z_{k-1}, x_{k-1}|y_{K-1})$. Then, we can predict Z_k and X_k before observing measurements Y_k,

$$\begin{aligned} p(z_k, x_k|y_{K-1}) =& p(z_k|x_k)\, p(x_k|y_{K-1}) \\ =& p(z_k|x_k)\Bigg(\sum_{z_{k-1}} \int p(z_{k-1}, x_{k-1}|y_{K-1}) \\ & \times f(x_k|x_{k-1}) dx_{k-1}\Bigg), \end{aligned} \quad (6)$$

where we have used the Markov property of our system to have $p(z_k|x_k) = p(z_k|x_k, y_{K-1})$, and marginalized out the variables X_{k-1} and Z_{k-1}. Often, when $f(\cdot)$ is nonlinear, computing the integral in closed form in (6) is difficult, and an approximation is used. One such approximation, the particle filter approximation, is discussed in Section V.

After computing (6), we update our predictions once measurements Y_k have been received,

$$\begin{aligned} p(z_k, x_k|y_K) &= \frac{p(z_k, x_k, y_k|y_{K-1})}{p(y_k|y_{K-1})} \\ &= \frac{p(z_k, x_k|y_{K-1}) p(y_k|x_k, z_k)}{p(y_k|y_{K-1})} \end{aligned} \quad (7)$$

where we again use the Markov property in writing $p(y_k|x_k, z_k) = p(y_k|x_k, z_k, y_{K-1})$. Equations (6) and (7) are the prediction and filtering steps, respectively, of recursive Bayesian filtering [2, Section 2.2], with a slight modification in that Z_k is added. We will outline how these equations and PDFs are different from the standard formulation.

The function $p(z_k|x_k)$ is the prior PMF of fault/non-fault sensor behavior (prior, in that it is the PMF of Z_k before the measurement vector Y_k is seen). The PDF $p(y_k|x_k, z_k)$ is the joint likelihood of this measurement vector Y_k from our unobserved system variables X_k and Z_k. Since, in (3), we assumed that the sensors j at time k had independent measurement noises, and could factor the observation model across sensors, we will do the same thing with these two functions,

$$\begin{aligned} p(z_k|x_k) &= \prod_{j=1}^{m_k} p(z_{k_j}|x_k) \\ p(y_k|x_k, z_k) &= \prod_{j=1}^{m_k} g_{k_j}^{z_{k_j}}(y_{k_j}|x_k), \end{aligned}$$

with $p(z_{k_j}|x_k)$ and $g_{k_j}^{z_{k_j}}(y_{k_j}|x_k)$ given by (4) and (5), respectively.

The marginal likelihood,

$$\begin{aligned} p(y_k|y_{K-1}) = \sum_{z_{k-1}} \sum_{z_k} \iint & p(z_{k-1}, x_{k-1}|y_{K-1}) f(x_k|x_{k-1}) \\ & \times p(z_k|x_k) p(y_k|x_k, z_k) dx_{k-1} dx_k, \end{aligned} \quad (8)$$

plays the role of a normalizing constant.

B. Relevant conditional and marginal PDFs

The PDFs (6) and (7) are the joint PDFs of the state X_k and sensor fault probabilities Z_k conditioned on measurements. These joint PDFs, however, are high dimensional, which presents two important drawbacks.

First, the PDFs can become very large. If the size of an estimate of $p(x_k|y_k)$ is proportional to N (the dimensionality of X_k), the size of an estimate of the joint PDF $p(z_k, x_k|y_K)$ will be proportional to $N \cdot 2^{M_k}$ (recall M_k is the length of Y_k), since a different estimate of X_k will exist for every combination of faulty/not faulty of all sensors (cf. [3]). For all but trivial problems, this may be infeasible.

Second, they are difficult to interpret: these large joint probabilities do not immediately answer the questions "what is the probability that sensor j at time k is faulty" or "what is the best estimate of the state X_k after accounting for potential faults."

Both concerns are answered with marginal or conditional PDFs for Z_{k_j} and X_k, respectively. We discuss how to compute these marginal and conditional PDFs in this Section.

One of these values, the posterior probability that a particular sensor j at time k is faulty for a particular value of X_k has the PMF

$$\begin{aligned} p(z_{k_j}|x_k, y_K) = p(z_{k_j}|x_k, y_{k_j}) &= \frac{p(z_{k_j}|x_k) p(y_{k_j}|x_k, z_{k_j})}{p(y_{k_j}|x_k)} \\ &= \frac{p(z_{k_j}|x_k) g_{k_j}^{z_{k_j}}(y_{k_j}|x_k)}{p(y_{k_j}|x_k)} \\ &= \frac{\left(\phi_{k_j}(x_k)\right)^{z_{k_j}} \left(1 - \phi_{k_j}(x_k)\right)^{(1-z_{k_j})} g_{k_j}^{z_{k_j}}(y_{k_j}|x_k)}{p(y_{k_j}|x_k)} \\ & \text{for } z_{k_j} \in \{0, 1\}, \end{aligned} \quad (9)$$

where in the first equality we have again used the assumption that the sensors j at time k have independent measurement noises and the Markov property of $f(\cdot)$, and the final two equalities use (4) and (5).

From (9), we can see that determining the probability that a sensor is faulty is itself a Bayesian inference problem, with $p(z_{k_j}|x_k)$ being a prior distribution and $g_{k_j}^{z_{k_j}}(y_{k_j}|x_k)$ a likelihood. The marginal likelihood in the denominator,

$$\begin{aligned} p(y_{k_j}|x_k) &= \sum_{z_{k_j}} p(z_{k_j}|x_k) p(y_{k_j}|x_k, z_{k_j}) \\ &= \sum_{z_{k_j}} p(z_{k_j}|x_k) g_{k_j}^{z_{k_j}}(y_{k_j}|x_k), \end{aligned} \quad (10)$$

again acts as a normalizing constant.

The PMF (9) is a conditional probability on X_k: it gives the probability that sensor j at time k is faulty for a *particular* value of X_k. The marginal PMF $p(z_{k_j}|y_K)$, which averages the fault probability over all possible values of X_k; and PDF $p(x_k|y_K)$, which weights our state estimate by the estimated probability of fault for each sensor; are easy to calculate from (7),

$$p(z_{k_j}|y_K) = \int \sum_{z_k \setminus z_{k_j}} p(z_k, x_k|y_K) dx_k \tag{11a}$$

$$p(x_k|y_K) = \sum_{z_k} p(z_k, x_k|y_K). \tag{11b}$$

IV. Problem statement without a model of faults

Recall from Section II that we may only have a model of $g_{k_j}^{z_{k_j}}(y_{k_j}|x_k)$ of the case $Z_{k_j} = 1$ (the measurement PDF of a sensor operating correctly), if we cannot predict every way a measurement might be faulty. This Section discusses how not having a model of faulty behavior stymies Section III's straightforward attempts at filtering.

Note that $g_{k_j}^0(\cdot)$ appears in all of the PDFs of interest in Section III: explicitly as in (9) or within the summations over z_k that appear in the marginal likelihoods (8) and (10). Not having a model of $g_{k_j}^0(\cdot)$ means that these equations cannot actually be computed. In the case of $p(z_{k_j}|x_k, y_{k_j})$, it is simple enough to restrict ourselves to $Z_{k_j} = 1$ and compute (9) in that special case,

$$P(Z_{k_j}=1|X_k=x_k, Y_{k_j}=y_{k_j}) = \frac{\phi_{k_j}(x_k) g_{k_j}^1(y_{k_j}|x_k)}{p(y_{k_j}|x_k)}.$$

However, the denominator $p(y_{k_j}|x_k)$, the marginal likelihood and normalizing constant, still requires $g_{k_j}^0(\cdot)$ to compute it from (10). With what we have so far, the best we can do is to emphasize that the marginal likelihood is a constant since it does not vary with Z_{k_j},

$$\begin{aligned} P(Z_{k_j} = 1|X_k = x_k, Y_{k_j} = y_{k_j}) \\ = C_{k_j} \phi_{k_j}(x_k) g_{k_j}^1(y_{k_j}|x_k), \end{aligned} \tag{12}$$

for some scalar constant C_{k_j}.

Parallel analyses are possible for the joint PDF $p(z_k, x_k|y_K)$ and the marginals in (11), with the unsolvable marginal likelihood in these cases being $p(y_k|y_{K-1})$ in (8).

In (12), we have replaced the normalizing denominator containing $g_{k_j}^0(\cdot)$ with a constant that must be estimated. This can be thought of as a general, and difficult, parameter estimation or adaptive control problem. The best parameter estimation method for C_{k_j} will vary on a case-by-case basis with the particular system considered and available data.

As it turns out, though, the inability to calculate normalizing denominators is a particularly common issue in Bayesian filtering: marginal likelihoods like (8) and (10) often cannot be computed when the PDF $f(x_k|x_{k-1})$ is not available in closed form or not easily integrable. In these situations, the *particle filtering* algorithm is appropriate, as it can approximate the prediction and filtering steps without computing $p(y_k|y_{K-1})$. The next Section introduces the particle filtering algorithm, and Section VI discusses how to extend it to deal with the inability to directly compute (8) or (10) due to a lack a model of faults.

V. Particle Filter Implementation

The methods discussed so far make use of many PDFs (and integrals thereof). These operations may be performed in closed form when the integrals are relatively simple, but this is often not the case when the system (1) is nonlinear.

For these situations, state estimation is often performed using Monte Carlo methods, with perhaps the most widely-used method being the particle filter [2]. A particle filter may be used when no closed-form model for $f(\cdot)$ exists (but the PDF may be sampled from by, e.g., running a stochastic simulation many times), or when numerically computing the integrals in (6) and (8) is computationally expensive.

A particle filter is constructed by replacing PDFs for X_k and Z_k with approximate PDFs (denoted with a hat) made up of many discrete samples (also called particles) from the continuous PDF. We extend a traditional particle filter by including the random variable Z_k from (6) (again, this is similar to previous works, e.g., [4]). Starting with an approximate PDF from the previous timestep, $\hat{p}(z_{k-1}, x_{k-1}|y_{K-1})$,

$$\begin{aligned} p(z_k, x_k|y_{K-1}) &= p(z_k|x_k)\Bigg(\sum_{z_{k-1}} \int p(z_{k-1}, x_{k-1}|y_{K-1}) \\ &\qquad \times f(x_k|x_{k-1}) dx_{k-1} \Bigg) \\ &\approx p(z_k|x_k) \sum_{p=1}^{P} \Bigg(\sum_{z_{k-1}} p(z_{k-1}, x_{k-1}^p|y_{K-1}) \delta_f(x_k^p|x_{k-1}^p) \Bigg) \\ &= p(z_k|x_k) \hat{p}(x_k|y_{K-1}) \\ &= \hat{p}(z_k, x_k|y_{K-1}), \end{aligned} \tag{13}$$

where P is an integer denoting the total number of particles drawn from $f(\cdot)$, $p \in \{1, \ldots, P\}$ indexes individual particles (or atoms of the probability distribution), and $\delta_f\left(x_k^p|x_{k-1}^p\right)$ is a Dirac delta, which places a unit mass on the point $x_k^p|x_{k-1}^p$, which is in turn equal to $\mathcal{F}(x_{k-1}^p)$ for the pth particle. The term $p(z_{k-1}, x_{k-1}^p|y_{K-1})$ refers to the probability for this particle p from our approximate PDF from the prior timestep, $\hat{p}(z_{k-1}, x_{k-1}|y_{K-1})$. The final two equalities indicate that the empirical PDF $\hat{p}(z_k, x_k|y_{K-1})$ consists of a weighted sum of P points $(z_k|x_k^p,\ x_k^p|x_{k-1}^p)$, with individual weights $p(x_k^p|y_{K-1})$, where the weights sum to one.

The corresponding particle filter update equation comes from plugging (13) into (7),

$$\begin{aligned} p(z_k, x_k|y_K) &= \frac{p(z_k, x_k|y_{K-1}) p(y_k|x_k, z_k)}{p(y_k|y_{K-1})} \\ &\approx \frac{\hat{p}(z_k, x_k|y_{K-1}) p(y_k|x_k, z_k)}{p(y_k|y_{K-1})} \\ &= \frac{1}{p(y_k|y_{K-1})} \Bigg(\sum_{p=1}^{P} p(z_k|x_k^p) p(y_k|x_k^p, z_k^p) \delta_f(x_k^p|x_{k-1}^p) \\ &\qquad \times \sum_{z_{k-1}} p(z_{k-1}^p, x_{k-1}^p|y_{K-1}) \Bigg) \end{aligned}$$

$$= \frac{\sum_{p=1}^{P} p(z_k^p, x_k^p|y_k)\delta_f(x_k^p|x_{k-1}^p)}{p(y_k|y_{K-1})} \quad (14)$$
$$= \hat{p}(z_k, x_k|y_K).$$

This posterior empirical PDF $\hat{p}(z_k, x_k|y_K)$ is thus made of the same collection of Dirac deltas as in $\hat{p}(z_k, x_k|y_{K-1})$, but with updated weights to reflect each point's updated probability (i.e., the prior probability times the likelihood).

As mentioned above, use of a particle filter also allows us to avoid having to calculate the marginal likelihood $p(y_k|y_{K-1})$ using (8). Instead, after $p(z_k^p, x_k^p|y_K)$ is calculated for every particle p in (14), we normalize these probabilities so that they sum to one,

$$p(y_k|y_{K-1}) \approx \sum_{p=1}^{P} p(z_k^p, x_k^p|y_K). \quad (15)$$

Equations (14)-(15) make up the update computation that is used in practice. However, for fault detection we will use a different update step, detailed in the next Section.

For brevity, we omit discussion of the particle filter's post-update resampling step. See, e.g., [2], for details.

VI. Particle-filter-enabled nonparametric hypothesis testing

A. Estimating the PDF of our null hypothesis

We now discuss the use of particle filter for sensor fault detection without a model of faults, $g_{k_j}^0(\cdot)$. To do so, we propose a modification described in this Section. Once we have computed a prediction step (13), instead of the traditional update step (14), we examine individually whether each sensor j at time k can be determined to be faulty. For every sensor, we can find

$$\begin{aligned} p(y_{k_j}|Z_{k_j}=1, y_{K-1}) &= \int p(y_{k_j}, x_k|Z_{k_j}=1, y_{K-1})dx_k \\ &= \int p(y_{k_j}|Z_{k_j}=1, x_k)p(x_k|Z_{k_j}=1, y_{K-1})dx_k \\ &= \int g_{k_j}^1(y_{k_j}|x_k)\frac{p(x_k, Z_{k_j}=1|y_{K-1})}{P(Z_{k_j}=1|y_{K-1})}dx_k \\ &= \int g_{k_j}^1(y_{k_j}|x_k)\frac{P(Z_{k_j}=1|x_k)p(x_k|y_{K-1})}{P(Z_{k_j}=1|y_{K-1})}dx_k \\ &\approx \frac{\sum_{p=1}^{P} g_{k_j}^1(y_{k_j}|x_k^p)\phi_{k_j}(x_k^p)p(x_k^p|y_{K-1})}{P(Z_{k_j}=1|y_{K-1})} \\ &= \hat{p}(y_{k_j}|Z_{k_j}=1, y_{K-1}), \end{aligned} \quad (16)$$

where, now, with y_{k_j} *not* being conditioned on, $g_{k_j}^1(y_{k_j}|x_k^p)$ means the actual PDF for $Y_{k_j}|(X_k^p = x_k^p, Z_{k_j} = 1)$. Therefore, the PDF $\hat{p}(y_{k_j}|Z_{k_j}=1, y_{K-1})$ is now a sum of non-delta PDFs; it can be compared to a kernel density estimate [11, Ch. 6]. The term $p(x_k^p|y_{K-1})$ in (16) is just the particle p's prior probability from $\hat{p}(z_k, x_k|y_{K-1})$.

The sum over $p(x_k^p|y_{K-1})$ for particles p weighs this PDF by the probabilities of particular values of x_k. Note that, since we have conditioned on $Z_{k_j} = 1$, the constant C_{k_j} from (12) does not appear. The denominator $P(Z_{k_j}=1|y_{K-1})$ is unknown, and computing it in closed from requires integrals over x_k. However, taking advantage of the particle filter structure, we can approximate this term similarly to (15),

$$P(Z_{k_j}=1|y_{K-1}) \approx \sum_{p=1}^{P} \phi_{k_j}(x_k^p)p(x_k^p|y_{K-1}). \quad (17)$$

The empirical PDF in (16)-(17) is an estimate of $p(y_{k_j}|Z_{k_j}=1, y_{K-1})$, the PDF of sensor j at time k's measurements under non-faulty sensor behavior. We will use this PDF to evaluate whether specific sensors might be faulty.

B. Performing the hypothesis test

Plugging a particular value of y_{k_j} into $p(y_{k_j}|Z_{k_j}=1, y_{K-1})$, assuming Y_{k_j} is a continuous random variable or vector, gives (loosely speaking) $P(Y_{k_j} \in B_\epsilon(y_{k_j})|Z_{k_j}=1, y_{K-1})$, where $B_\epsilon(y_{k_j})$ is an ϵ-ball centered at y_{k_j}. Finding the value of our estimate $\hat{p}(y_{k_j}|Z_{k_j}=1, y_{K-1})$ at an observed y_{k_j}, then, gives $\hat{P}(Y_{k_j} \in B_\epsilon(y_{k_j})|Z_{k_j}=1, y_{K-1})$.

At this point, we have established the necessary components to conduct a hypothesis test for a null hypothesis $H_0 : Z_{k_j} = 1$. For some $\alpha \in (0, 1)$, we can reject H_0 and conclude that $Z_{k_j} = 0$ (that is, that the sensor is faulty and should be excluded) if

$$\hat{P}(Y_{k_j} \in B_\epsilon(y_{k_j})|Z_{k_j}=1, y_{K-1}) < \alpha \quad (18)$$

for the observed value of y_{k_j}.

The α here plays the role of a *significance level* for the hypothesis test, which is the maximum acceptable probability of declaring a sensor faulty when it is actually non-faulty. Here, the test in (18) is a *nonparametric* hypothesis test, as we use a nonparametric model of $p(y_{k_j}|Z_{k_j}=1, y_{K-1})$ as given by (16)-(17)[1].

Once we have rejected sensors that meet (18), the remaining sensors (those for which we could not reject H_0) can be concluded to be non-faulty and their measurements can be used in a particle filter update equation that takes $Z_{k_j} = 1$ for these remaining sensors:

$$\begin{aligned} \hat{p}(x_k|y_K) &= \frac{1}{p(y_k|y_{K-1})} \\ &\times \sum_{p=1}^{P}\Big(\prod_{k_j \in \mathcal{H}_0} g_{k_j}^1(y_{k_j}|x_k^p)\Big)\delta_f(x_k^p|x_{k-1}^p)p(x_{k-1}^p|y_{K-1}) \\ &= \frac{\sum_{p=1}^{P} p(x_k^p|y_K)\delta_f(x_k^p|x_{k-1}^p)}{p(y_k|y_{K-1})}, \end{aligned} \quad (19)$$

where $\mathcal{H}_0$ is the set of sensors that did not meet (18), and the denominator $p(y_k|y_{K-1})$ is estimated via (15).

A higher value of α leads to more aggressive rejection of measurements. Also note that in general we do *not* have $\hat{P}(Y_{k_j} \in B_\epsilon(y_{k_j})|Z_{k_j}=0, y_{K-1}) = 1 - \hat{P}(Y_{k_j} \in B_\epsilon(y_{k_j})|Z_{k_j}=1, y_{K-1})$. The RHS, the true probability that a sensor j at time k is taken to be non-faulty when it is actually faulty, needs knowledge or estimation of $g_{k_j}^0(\cdot)$.

[1] Note that, strictly speaking, (18)'s hypothesis test is not tied to a particle filter as we have developed it in this paper. The particle filter, though, gives us a generally-applicable nonparametric hypothesis test. If we have a system where we know a parametric filter (such as a Kalman or Extended Kalman Filter) is appropriate (i.e., if we have a closed-form, easily-integrable $f(\cdot)$), we could derive (16) and (18) for the alternative filter's parametric PDFs.

C. Algorithm

Our particle filter method with real-time fault detection without fault models can be summarized as:

1) Perform a prediction step as normal, using (13).
2) For each sensor j at time k, calculate $\hat{p}(y_{k_j} \mid Z_{k_j} = 1, y_{K-1})$ using (16).
3) For each sensor, determine whether to reject it as faulty using (18) for the observed y_{k_j}.
4) Perform an update step with the non-rejected measurements using (19) (and, if desired, a resampling step).
5) Advance in time, $k \leftarrow k+1$, return to step 1, repeat.

In summary, repurposing of the particles we already have when we use a particle filter allows us to perform hypothesis tests, and detect faults of unknown form, for free.

VII. A CASE STUDY: GNSS SENSOR FUSION FOR HIGHWAY TRAFFIC STATE ESTIMATION

Our case study concerns a traffic control system that takes possibly-faulty third-party data from connected vehicles' global navigational satellite system (GNSS) devices to estimate the state of a road network. Accurate knowledge of traffic systems' operations is needed for reactive traffic control, and gaining this knowledge from third-party data is a goal of modern intelligent transportation system "smart city" applications [12], [13].

A. System model equations for our case study

In [14, Section IV-B], we presented results obtained using a particle filter that used the techniques detailed in Section V. We demonstrated the particle filter's performance on a 19-mile portion of I-210 West in southern California. As our system model $f(\cdot)$, we make use of the macroscopic Cell Transmission Model (CTM) [15], which approximates traffic as compressible fluid flows. This type of model can capture important nonlinear emergent features in traffic flows like traffic jams and congestion waves.

In the CTM, the freeway is discretized into a sequence of finite-volume cells, also called links. The state vector x_k is the vector of link densities $\rho_{\ell,k}$. The state update equation for link ℓ is

$$\rho_{\ell,k+1} = \rho_{\ell,k} + \frac{1}{L_\ell}(q_{\ell-1,k} - q_{\ell,k} + r_{\ell,k} - s_{\ell,k}), \qquad (20)$$

where L_ℓ is the length of link ℓ, $q_{\ell,k}$ denotes the vehicle flow leaving link ℓ to link $\ell+1$ at time k, $r_{\ell,k}$ is the flow entering link ℓ from an onramp (if any) at time k, and $s_{\ell,k}$ is the flow leaving link ℓ to an offramp (if any) at time k. When there is no onramp entering link $\ell+1$, the inter-link flows in (20) are given by

$$\begin{aligned} q_{\ell,k} = \min(&v_{f,\ell} \cdot \rho_{\ell,k} \cdot L_\ell,\, Q_{max,\ell}, \\ &w_{\ell+1} \cdot L_{\ell+1} \cdot (\rho_{J,\ell+1} - \rho_{\ell+1,k})), \end{aligned} \qquad (21)$$

where $v_{f,\ell}$ is the freeflow speed of link ℓ, $Q_{max,\ell}$ is the capacity, or maximum possible flow over a time period, of link ℓ, $w_{\ell+1}$ is the speed at which congestion waves propagate upstream in link $\ell+1$, and $\rho_{J,\ell+1}$ is the jam density, or maximum possible density, of link $\ell+1$. The third term in the $\min(\cdot)$ function in (21) lets the downstream link $\ell+1$ refuse to accept flow from link ℓ if $\ell+1$ is too full.

When there is an onramp entering link $\ell+1$, its available supply (the third argument to (21)'s $\min(\cdot)$ function) is distributed among link ℓ and the onramp according to the junction model of [16]. The ramp flows themselves, $s_{\ell,k}$ and $r_{\ell,k}$ in (20), are random variables. See [14] for full implementation details of these last two points.

A common type of first-party sensor for freeway traffic are inductive loop detectors buried in the pavement. These detectors can noisily measure density. A third-party source of data are vehicle-carried GNSS devices that report the speed of individual vehicles. In the CTM, the speed of traffic in link ℓ at time k is $v_{\ell,k} = L_l \cdot \rho_{\ell,k}/q_{\ell,k}$. A high vehicle density leads to congestion, and hence low speeds. We can use speed measurements to estimate density using this relationship in a Rao-Blackwellized particle filter, as discussed in [14].

To test our fault detection method, we simulated a realization of our freeway model, with randomness introduced by the random onramp, offramp, and upstream boundary flows. In addition to noisy density measurements from 41 loop detectors, we simulated GNSS speed measurements with a simulated penetration rate of 2%. To generate the faulty third-party measurements, we used $\phi_{k_j}(x_k) = 0.7$ for all j, k, and x_k, i.e., each speed measurement had a 30% probability of being faulty. We used two fault models: a faulty measurement had a 1/3 probability of reporting zero (i.e., a stopped car misreporting its location), and a 2/3 probability of drawing from a Gaussian distribution with mean 30 m/s and standard deviation 10 m/s. The non-fault model for velocity measurements, $g^1_{k_j}(\cdot)$, was Gaussian with a mean of the true link velocity and standard deviation of 10% of the mean (similar to [12]). Fig. 1 shows the true state and velocity measurements used. Note that the particle filter state estimator/fault detector has no information of which measurements might be faulty or what faults may look like.

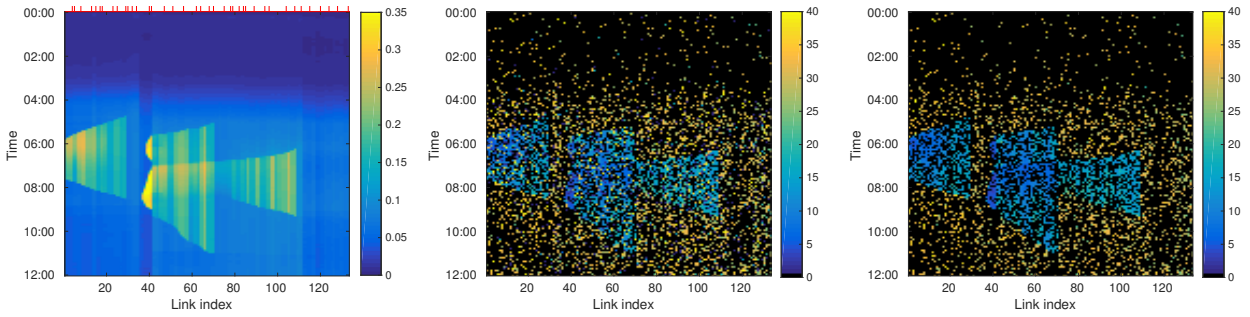
TABLE I
SIMULATED GNSS MEASUREMENT FAULT DETECTION RESULTS

	$\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.1$	No Faults
True Positives	1092	1340	1994	-
False Positives	88	95	2868	-
True Negatives	4506	4499	1726	-
False Negatives	914	666	12	-
Labeling Error	15.18%	11.53%	43.64%	-
Density MAPE	3.81%	3.51%	4.63%	3.43%

In these results, an incorrect $\phi_{k_j} = 0.3$ is used (see text). "Positives" refer to sensors for which we accepted H_1, i.e., sensors that our fault detection hypothesis test (18) concluded were faulty. "True" and "False" refer to correct and incorrect decisions, respectively, of whether a sensor is faulty. MAPE = mean absolute percentage error.

B. Results

Table I presents the results of estimation with our fault detection method for several levels of α, as well as a "best case" state estimator where the faulty measurements are not present. In these fault detection tests with our previously-generated measurements, we used $\phi_{k_j} = 0.3$ in the estimation



(a) True density state (veh/m) (b) Speed measurements (m/s) (c) Non-faulty speed subset (m/s)

Fig. 1. Simulated true density state trajectory (a), speed measurements (b), and non-faulty subset of speed measurements (c) used in simulation. Traffic moves to the right, and the time period considered is midnight to noon (as marked on the vertical axis). In (a), the links instrumented with loop detectors that noisily measure density are marked with red ticks. At peak morning demand, bottlenecks near links 30, 70, and 110 lead to traffic jams that propagate upstream (i.e., they extend to the left as time advances), leading to increased density and lower speed. The jams later dissipate as demand falls.

algorithm. This ϕ_{k_j} value was chosen as an arbitrary example of an incorrect ϕ_{k_j} value, since the true ϕ_{k_j} for this example problem was 0.7 (as set in the problem specification). By running our algorithm with an incorrect ϕ_{k_j} value, we demonstrate that the hypothesis test (18) is robust to very large uncertainty in this prior probability in this application. Each state estimator used 1000 particles.

We see the influence of α that we would expect: higher values lead to more aggressive flagging of measurements as faults, with both true and false positives increasing as α increases. The state estimation error is represented by the mean absolute percentage error (MAPE), the average of $|\hat{\rho}_{\ell,k} - \rho_{\ell,k}|/\rho_{\ell,k}$ for all ℓ and k, with $\hat{\rho}_{\ell,k}$ the ℓth entry of $\sum_{p=1}^{P} x_k^p \cdot p(x_k^p|y_k)$, i.e., the mean of the posterior particle filter PDF. No state estimators that saw faulty data were able to match the MAPE of the faultless estimator. However, the estimation error decreases with fault labeling error, and the α with the lowest fault labeling error (11.53% for $\alpha = 0.01$) has only a 3% increase in MAPE over the fault-free case.

VIII. Conclusion

This paper considered a problem where state estimation is desired, but some measurements may be faulty in unknown ways. Our method for handling this problem takes advantage of a particle filter's structure to perform real-time nonparametric hypothesis tests against the known non-faulty mode. Now that we have a tool that can pick out faulty sensors, next steps include convergence analysis or inclusion of adaptive control techniques for estimation of unknown values, including the faulty sensor PDF $g_{k_j}^0(\cdot)$ and the optimum α in (18). For example, to estimate the Bernoulli probability of a sensor fault, ϕ_{k_j}, one could model it as being drawn from a beta distribution (the conjugate prior of the Bernoulli distribution). Then, as sensors are marked as faulty or not in (18), ϕ_{k_j}'s distribution can be updated based on standard Bayesian inference [17].

Our discussion was in the context of sensors that are unintentionally faulty, but these methods should also be applicable for robustness to falsified sensors. If a third party has control of sensors used by a control system, a malicious actor may feed spoofed measurements to the system to purposely manipulate its operation. Our fault-model-free method may complement existing defenses that actively search for patterns indicative of spoofed data: perhaps the lack of an explicit model for faulty or spoofed data may make it harder for attacks to be tailored to evade such a detector.

References

[1] Z. Gao, C. Cecati, and S. X. Ding, "A Survey of Fault Diagnosis and Fault-Tolerant Techniques– Part I: Fault Diagnosis With Model-Based and Signal-Based Approaches," *IEEE Trans. on Industrial Electronics*, vol. 62, no. 6, pp. 3757–3767, Jun. 2015.

[2] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," in *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011, pp. 656–704.

[3] T. Wei, Y. Huang, and C. L. P. Chen, "Adaptive Sensor Fault Detection and Identification Using Particle Filter Algorithms," *IEEE Trans. on Sys., Man, and Cybernetics, C*, vol. 39, no. 2, pp. 201–213, Mar. 2009.

[4] N. De Freitas, "Rao-Blackwellised particle filtering for fault diagnosis," in *IEEE Aerosp. Conf. Proc.*, vol. 4, 2002, pp. 1767–1772.

[5] F. Caron, M. Davy, E. Duflos, and P. Vanheeghe, "Particle Filtering for Multisensor Data Fusion With Switching Observation Models: Application to Land Vehicle Positioning," *IEEE Trans. on Signal Processing*, vol. 55, no. 6, pp. 2703–2719, Jun. 2007.

[6] P. Tadić and Z. Durović, "Particle filtering for sensor fault diagnosis and identification in nonlinear plants," *Journal of Process Control*, vol. 24, no. 4, pp. 401–409, Apr. 2014.

[7] Y. Chen and Y. Rui, "Real-time speaker tracking using particle filter sensor fusion," *Proc. of IEEE*, vol. 92, no. 3, pp. 485–494, Mar 2004.

[8] J. F. Chamberland and V. V. Veeravalli, "Decentralized detection in sensor networks," *IEEE Trans. on Signal Processing*, vol. 51, no. 2, pp. 407–416, Feb 2003.

[9] H. Durrant-White and T. C. Henderson, "Multisensor data fusion," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2016, ch. 35, pp. 867–896.

[10] L. Mihaylova, R. Boel, and A. Hegyi, "Freeway traffic estimation within particle filtering framework," *Automatica*, vol. 43, no. 2, pp. 290–300, 2007.

[11] D. W. Scott, *Multivariate Density Estimation*. Wiley, 2015.

[12] D. Work, S. Blandin, O.-P. Tossavainen, B. Piccoli, and A. Bayen, "A traffic model for velocity data assimilation," *Applied Mathematics Research eXpress*, vol. 2010, no. 1, pp. 1–35, 2010.

[13] A. A. Kurzhanskiy and P. Varaiya, "Traffic management: An outlook," *Economics of Transportation*, vol. 4, no. 3, pp. 135–146, Sep. 2015.

[14] M. Wright and R. Horowitz, "Fusing Loop and GPS Probe Measurements to Estimate Freeway Density," *IEEE Trans. on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3577–3590, Dec 2016.

[15] C. Daganzo, "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory," *Transportation Research, Part B*, vol. 28, no. 4, pp. 269–287, 1994.

[16] A. Muralidharan, G. Dervisoglu, and R. Horowitz, "Freeway traffic flow simulation using the Link Node Cell transmission model," in *American Control Conference*, June 2009, pp. 2916–2921.

[17] C. Robert, *The Bayesian Choice*. Springer, 2007.