Multi-Sensor Fault Recovery in the Presence of Known and Unknown Fault Types

Steven Reece and Stephen Roberts
Robotics Research Group
Dept. Engineering Science
Oxford University, UK.
reece@robots.ox.ac.uk
sjrob@robots.ox.ac.uk

Christopher Claxton and David Nicholson
Advanced Technology Centre
BAE Systems
Bristol, UK.
christopher.claxton@baesystems.com
david.nicholson2@baesystems.com

Abstract – *This paper proposes an efficient online, hybrid, Bayesian multi-sensor fusion algorithm for target tracking in the presence of modelled and unmodelled faults. The algorithm comprises two stages. The first stage attempts to remove modelled faults from each individual sensor estimate. The second stage de-emphasises estimates which have been subject to unanticipated faults and are still faulty despite undergoing the Stage 1 fault recovery process. The algorithm is a computationally efficient and decentralisable hybrid of two standard approaches to fault detection, namely model-based fault detection and majority voting. The algorithm is tested on two distinct simulated scenarios (1) when the target process model does not match reality and (2) in the presence of simultaneous modelled and unanticipated faults.*

Keywords: multi-sensor data fusion, Kalman filter, fault detection and recovery.

1 Introduction

The deployment of sensor networks to monitor and track events, objects, and situations, in a variety of environments, is becoming widespread. This paper is concerned with how to filter and fuse the uncertain sensor data to ensure it produces statistically consistent state estimates. Specifically, three main types of uncertainty are considered:

- Uncertainty associated with noisy sensor measurements.
- Systematic uncertainty, associated with faulty sensor measurements where the fault parameters are known.
- Epistemic uncertainty, associated with faulty sensor measurements where the fault parameters are not completely known.

Uncertainty due to random noise is endemic in sensor measurements, but its effect can be reduced by standard filtering and fusion processes, such as Kalman filtering. Sensors can generate faulty measurements for a number of reasons, such as power failure, physical damage, and miscalibration, to name but a few. If parameterised models for the fault types are available, they can be exploited by a fault recognition algorithm. However, often an explicit fault model is not known, in which case the algorithm must be able to deal with model incompleteness. This paper describes a general data fusion framework that accommodates all three types of uncertainty.

Recent work has seen an increasing trend in sensor networks which decentralise the filtering and fusion processes to reduce the computational and communication bottlenecks associated with a centralised process. However, this can exacerbate the problem of a faulty sensor because its measurement may be now fused at multiple sites and then the fused data is re-propagated to further fusion nodes so its impact could spread through the network. This means there is a particularly strong imperative to recognise, remove, or mitigate faulty sensor data in a networked sensor fusion system.

The framework presented here incorporates two data processing stages. The first stage attempts fault recognition and removal by hypothesising a number of models to describe the faults. This enables the fault to be removed from the data. For each model a recursive estimator is applied to the data and a Bayesian approach is used to learn the probability that the model is correct as well as the probability density function over the model parameters. The state estimates are processed by a mixture reduction algorithm to provide a single summary estimate encompassing all fault types from each source.

The second stage applies a further data fusion algorithm to combine state estimates from multiple sources. Robustness is required because the estimates may not be entirely reliable, due to unmodelled or imprecisely modelled fault types. A learning stage determines the reliability of each estimate and a fusion algorithm combines them. The Stage 2 algorithm may be replicated on all fusion nodes within a decentralised sensor system.

In our framework, if we know what the fault is (i.e. the known known), we remove it at Stage 1 of our algorithm. Alternatively, if we assume a closed set of fault types but we don't know which one is operating at a given time (i.e. known unknowns) then we place a distribution over the

faults in Stage 1 of our algorithm. In this case, the distribution is known but the fault is unknown. Finally, if our world is not closed and there are faults which we haven't modelled (i.e. the unknown unknowns) then Stage 2 utilises a Bayesian voting scheme to remove unreliable sensors.

Two decentralised system architectures are possible. The architectures differ in the way they perform Stage 1 of the fault recovery algorithm. One architecture performs Stage 1 at each sensor node before communicating the results to a fusion centre (see Figure 1). Alternatively, the sensors can communicate their raw observations to the fusion centre with the requirement that the fusion centre performs Stage 1 on each sensor input stream. The former architecture is communication unintensive but places a large computational burden on the fusion centre. The latter architecture is communication intensive but requires less computation at the fusion centre. We adopt the latter architecture in this paper although our two stage fault recovery algorithm can be applied to both approaches.

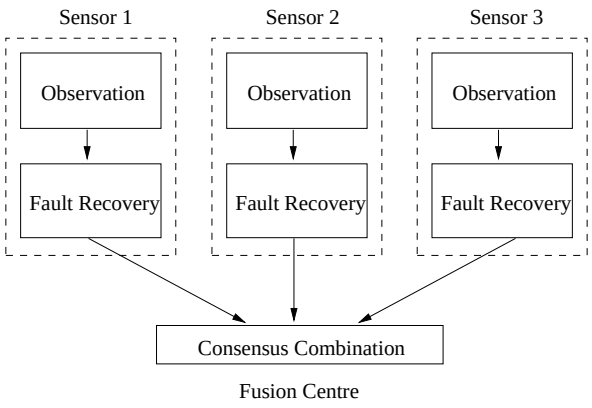

Figure 1: System architecture showing (1) Stage 1 fault recovery operating within each sensor and (2) unanticipated fault processing at Stage 2 performed by the consensus combination algorithm.

Venkatasubramanian *et al.* [14] classify fault recognition algorithms into three broad categories: quantitative model-based methods, qualitative methods and process history based methods. Particularly related to our work are the quantitative methods that employ recursive state estimators. For example, the Kalman filter is commonly used to monitor innovation processes and prediction error [16, 2]. Banks of Kalman filters have also been applied to fault recognition, where each filter typically corresponds to a specific fault mode [8, 1]. In addition, pattern recognition methods, such as recursive least-squares have been described for online fault parameter estimation, e.g. drifts [6]. For complex faults, with nonlinear dynamics and discrete/continuous parameters, a Dynamic Bayesian Network framework may be more appropriate for fault detection and diagnosis [9]. Voting techniques are frequently used to fuse multi-sensor data in the presence of faults [16]. Another approach, is to search, given error bounds on different sensors, for subsets of sensor measurements with different degrees of consistency [4].

Most previous approaches to fault recognition and fusion in the presence of faults assume the fault modes can be perfectly described by a parameterised model. In practice, there is likely to be some deviation between what the actual faults look like and what their models predict. This residual may or may not be important given the sensor system and its application. However, as indicated earlier, in a decentralised network even small residuals can have a significant impact as they propagate through the fusion network. The framework described in this paper contains the required level of robustness to ensure that residual errors in fault recognition/removal do not prevent statistically consistent estimates from being formed at the sensor fusion nodes.

Our approach uses both model-based and voting-based approaches to perform fault identification, isolation and fault removal and presents the user with a consistent estimate of the target's location. Model-based and voting approaches to fault identification and isolation are not new. However, by casting the solution in a Bayesian framework we are able to use the soft decision making properties of Bayes to formulate a hybrid model-based/voting approach for target tracking in the presence of modelled and unmodelled faults. Unlike [10], whose Bayesian approach to fault detection invokes the power set of sensors, we are able to cast a solution in terms of individual sensor nodes. As a result, our approach is computationally efficient and also decentralisable.

The paper is organised as follows. Section 2 presents a specification of the multi-sensor estimation problem in the presence of faults. Section 3 then describes fault models for an array of faults (drift, shocks, spikes and echos) that sensors can experience. Then, Section 4 describes a Bayesian approach to inferring fault free estimates given observations from sensors which are subject to modelled faults. Section 5 describes a Bayesian multi-sensor approach to fault recovery when sensors are subjected to unmodelled faults. These two approaches can be combined into a two stage fault removal algorithm. The efficacy of our algorithm is illustrated via examples in Section 6 and then via statistical analysis in Section 7. Finally, we conclude and offer future research directions in Section 8.

2 Problem Description

We aim to track a process x over time t where:

$$x(t) = G(t)x(t-1) + \omega_t \ ,$$

$G(t)$ is the linear, time dependent plant model and ω_t is zero-mean, Gaussian distributed with covariance Q_t. The process is observed by a set S of potentially faulty sensors. For each $s \in S$:

$$z_s(t) = x(t) + e_s(t) + \nu_{s,t}$$

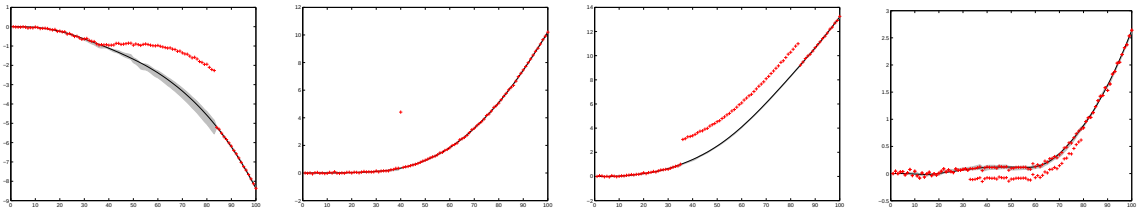

Figure 2: Fault types: Figures show the true target trajectory (solid line) and the crosses show the, sometime faulty, target observations. Also shown are target trajectory estimates obtained using the approach outlined in Section 4 with the faulty observations.

where $\nu_{s,t}$ is zero-mean, Gaussian distributed with covariance $W_{s,t}$. We assume uncorrelated noise throughout, $E[\omega_{t_1}, \omega_{t_2}] = 0$ and $E[\nu_{s_1,t_1}, \nu_{s_2,t_2}] = 0$ for $s_1 \neq s_2$ or $t_1 \neq t_2$ and, also, $E[\omega_{t_1}, \nu_{s_2,t_2}] = 0$ for all t_1, t_2 and s_2.

The error process $e(t)$ is a function of the *fault type (ft)*. In this paper we consider drift, spike, shock and echo fault types. These fault types are described in Section 3. Each fault has a *fault start time (fst)* and *fault end time (fet)* which bound the period over which the fault occurs. Our aim is to estimate the target's location, x, using potentially faulty sensor readings from our network of sensors. Each sensor tracks the target over time and attempts to remove any faults from its observations before fusing them with its estimate of the target's location. This is the first stage of our algorithm. A second stage fuses together the fault rectified estimates supplied by each sensor. This second stage accommodates inadequacies in each sensor's fault recovery process when the sensor's estimate still encodes some residual fault. This stage de-emphasises sensor estimates for which the faults have not been successfully removed.

3 Known knowns: Fault Models

Four fault types are considered (see Figure 2). However, our approach is not limited to these fault types alone. These fault types are:

1. *Drift*: The error gradually increases linearly from zero over time until there is an abrupt cut-off at which the sensor bias disappears.

$$e(t) = \begin{cases} \Delta(t - \text{fst}) & \text{if fst} \leq t \leq \text{fet} \\ 0 & \text{otherwise} \end{cases}$$

   The parameter Δ is the drift rate.

2. *Spike*: A single observation is offset.

$$e(t) = \begin{cases} \Delta & \text{if fst} \leq t \leq \text{fet} \\ 0 & \text{otherwise} \end{cases}$$

   and fst $=$ fet. The parameter Δ is the spike magnitude.

3. *Shock*: A constant offset is sustained over a long period. For example, a sensor is knocked out of line and then reset.

$$e(t) = \begin{cases} \Delta & \text{if fst} \leq t \leq \text{fet} \\ 0 & \text{otherwise} \end{cases}$$

   and fst $<$ fet. The parameter Δ is the observation bias due to the shock.

4. *Echo*: Two observations are received, one is the true observation and another is the target image formed by the signal reflecting from a wall say. The distance between the observations remains constant. We model the echo by conflating the observations into a single value, $z = 0.5(z_1 + z_2)$. Thus:

$$e(t) = \begin{cases} \Delta & \text{if fst} \leq t \leq \text{fet} \\ 0 & \text{otherwise} \end{cases}$$

   The parameter Δ now describes the offset of z from the truth. Note, an echo is distinguishable from a shock as two observation streams are observed for an echo and only one for a shock.

Note that, in each case, $\Delta = 0$ describes a fault free sensor.

4 Stage One: Known Unknowns

We shall use a Bayesian approach to remove the faults. In essence, we learn the current fault characteristics (if a fault has occurred) online. Then we can remove the fault once its effect is known. Since there will always be some uncertainty as to the nature of the fault, we integrate over the fault probabilities and generate a single estimate which captures the uncertainties in the fault.

Each sensor is dealt with individually and, for ease of representation, we omit the sensor index in this section. Let x_t denote the target's state at time t, z_1^t denote the sensor's observations up to time t and let z_t denote the sensor's observation at time t. Let $\theta = \{\text{ft}, \text{fst}, \text{fet}, \Delta\}$ denote the failure hypothesis [3] with parameters which are, respectively, the fault type, fault start and end times and the magnitude of the fault. Our aim is not only to identify which type of fault has occurred but also when it occurred. We also would like to remove the fault from the data so that the sensor can supply

an accurate estimate of the target's location. The target state given (possibly) faulty observations z_1^t is distributed thus:

$$p(x_t \mid z_1^t) = \int d\theta p(x_t \mid z_1^t, \theta) p(\theta \mid z_1^t) \ .$$

The posterior $p(x_t \mid z_1^t, \theta)$ is Gaussian and can be obtained from $p(x_{t-1} \mid z_1^{t-1}, \theta)$ using the Kalman filter. The mean and covariance for $p(x_t \mid z_1^t, \theta)$ are:

$$\hat{x}_\theta(t \mid t) = \hat{x}_\theta(t \mid t-1) + K_\theta[z(t) - e_\theta(t) - \hat{x}_\theta(t \mid t-1)]$$
$$P_\theta(t \mid t) = [I - K_\theta] P_\theta(t \mid t-1)$$

where K_θ is the Kalman gain, $K_\theta = P_\theta(t \mid t-1)[P_\theta(t \mid t-1) + W(t)]^{-1}$. Values for $\hat{x}_\theta(t \mid t-1)$ and $P_\theta(t \mid t-1)$ can be obtained in the usual way:

$$\hat{x}_\theta(t \mid t-1) = G(t)\hat{x}_\theta(t-1 \mid t-1) \ ,$$
$$P_\theta(t \mid t-1) = G(t)P_\theta(t-1 \mid t-1)G(t)^T + Q(t) \ .$$

Thus:

$$\hat{x}(t \mid t') = \int d\theta \ p(\theta \mid z_1^{t'}) \ \hat{x}_\theta(t \mid t') \ ,$$
$$P(t \mid t') = \int d\theta \ p(\theta \mid z_1^{t'})[P_\theta(t \mid t') + \Delta\hat{x}_\theta(t \mid t')\Delta\hat{x}_\theta(t \mid t')^T]$$

where $\Delta\hat{x}_\theta(t \mid t') \triangleq \hat{x}_\theta(t \mid t') - \hat{x}(t \mid t')$ and $t' = t$ or $t' = t-1$, the latter corresponding to state prediction. Similarly, the parameter probability $p(\theta \mid z_1^t)$ can be obtained thus:

$$\begin{aligned} p(\theta \mid z_1^t) &= \int dx_t p(\theta, x_t \mid z_1^t) \\ &= \int dx_t \frac{p(z_t \mid \theta, x_t, z_1^{t-1})}{p(z_t \mid z_1^{t-1})} p(\theta \mid z_1^{t-1}) p(x_t \mid z_1^{t-1}, \theta) \\ &= \int dx_t \frac{p(z_t \mid \theta, x_t)}{p(z_t \mid z_1^{t-1})} p(\theta \mid z_1^{t-1}) p(x_t \mid z_1^{t-1}, \theta) \\ &= c \, p(\theta \mid z_1^{t-1}) \times \\ &\qquad \mathbb{N}[z_t - e_\theta(t); \hat{x}_\theta(t \mid t-1), P_\theta(t \mid t-1) + W(t)] \end{aligned}$$

where c is a normalisation constant such that $\int d\theta p(\theta \mid z_1^t) = 1$. Thus $p(\theta \mid \cdot)$ can be calculated recursively. The prior $p(\theta)$ should be uninformative (a flat prior is often adequate).

Our Bayesian approach can be implemented in numerous ways. We chose to implement this approach using a multi-hypothesis dual Kalman filter. Each dual Kalman filter simultaneously estimates the state x and the parameter Δ given the fault type, ft, and the fault start and end times, fst and fet. A multi-hypothesis KF scheme is used to incorporate uncertainty in ft, fst and fet. Each KF encodes a distinct $\{\text{ft}, \text{fst}, \text{fet}\}$ triplet for discrete values for fst and fet. To obtain state estimates and state covariances, the ft, fst and fet parameters are integrated out by merging the hypotheses using standard mixture reduction [13]. Probability distributions over ft and distributions for the fst and fet can be obtained directly from $p(\theta \mid z_1^t)$ via marginalisation. A Gibbs sampler implementation would be more sophisticated than the multi-hypothesis implementation. However, details of how our fault recovery approach is implemented is irrelevant to this paper.

The output of our algorithm is called a *report* and comprises the current state estimate, state covariance matrix, predictions for the current state which exclude current observations, distributions over fault start and end times and a probability for each possible fault type.

$$R(t) = \{\hat{x}(t \mid t), P(t \mid t), \hat{x}(t \mid t-1), P(t \mid t-1), \\ p_{\text{fst}}(t), p_{\text{fet}}(t), p_{\text{ft}}(t)\} \ .$$

Figure 2 shows the efficacy of our approach on each of the fault types described in Section 3. Note, the fault recovery algorithm is online and does not make use of smoothing operations. We may obtain an output from the Stage 1 algorithm at any time, even before the fault has ended. The figures show the actual output from Stage 1 at each time instant.

5 Stage Two: Unknown Unknowns

We use redundancy within the multi-sensor system to remove errors caused by unmodelled faults. Like [10] we compare the estimates from each sensor and remove outlier sensors. Unlike [10] we assign reliability to individual sensors and not sensor groups. The advantage of considering sensors individually is clear for modular systems where a new sensor can be added at any time and also for computational reasons within large multi-sensor systems.

A sensor is *unreliable* (or *untrustworthy*) when its output is still inconsistent even after undergoing fault recovery during Stage 1 of the algorithm. The residual faults that may remain there can be caused by simultaneous multiple faults within a sensor or can arise when the target is not moving according to the KF target process model or even when the particular fault type is not modelled. If a sensor $s \in S$ is unreliable at time t, we write $w_s(t) = 0$. If it is reliable then $w_s(t) = 1$. The *reliability* of a sensor at time t is a measure between 0 and 1 and is defined to be the probability that the sensor is reliable: $p(w_s(t) = 1 | \cdot)$.

We assume that sensors become unreliable independently of each other. This assumption is valid when, for example, individual sensors are knocked out of line. However, the assumption may fail, for example, when sensors share a common, and yet inappropriate model for the target dynamics. Fortunately, as will be demonstrated in Section 5, even when the failing component is shared between sensors, the sensor suite can still recover from the fault. We assume, like [10], at any time the number of reliable sensors exceeds the number of unreliable sensors. Consequently, the probability of simultaneous failure of all sensors is zero:

$$\prod_s p(w_s(t) = 0 \mid \cdot) = 0 \tag{2}$$

and therefore, $\max_s\{p(w_s(t) = 1 \mid \cdot)\} = 1$.

$$
\begin{aligned}
P_{1 \uplus 2}(t \mid t') = {} & p(w_1(t) = 1 \mid R_1^{t'}) p(w_2(t) = 1 \mid R_1^{t'}) [P_{1 \oplus 2}(t \mid t') + (\hat{x}_{1 \oplus 2}(t \mid t') - \hat{x}_{1 \uplus 2}(t \mid t'))(\hat{x}_{1 \oplus 2}(t \mid t') - \hat{x}_{1 \uplus 2}(t \mid t'))^T] \\
& + p(w_1(t) = 1 \mid R_1^{t'}) [1 - p(w_2(t) = 1 \mid R_1^{t'})] [P_1(t \mid t') + (\hat{x}_1(t \mid t') - \hat{x}_{1 \uplus 2}(t \mid t'))(\hat{x}_1(t \mid t') - \hat{x}_{1 \uplus 2}(t \mid t'))^T] \\
& + [1 - p(w_1(t) = 1 \mid R_1^{t'})] p(w_2(t) = 1 \mid R_1^{t'}) [P_2(t \mid t') + (\hat{x}_2(t \mid t') - \hat{x}_{1 \uplus 2}(t \mid t'))(\hat{x}_2(t \mid t') - \hat{x}_{1 \uplus 2}(t \mid t'))^T]
\end{aligned} \tag{1}
$$

5.1 Combining Estimates from Unreliable Sensors

Before describing how reliability is calculated we describe our *consensus* algorithm which combines unreliable estimates from multiple sources. The consensus algorithm uses a combination of Kalman filter fusion and mixture reduction [13]. Consider combining two estimates, each is assigned a reliability $p(w_s(t) = 1 \mid \cdot)$. Four scenarios are possible.

1. Both sensors are reliable in which case we fuse both estimates.
2. Sensor 1 is unreliable and Sensor 2 is reliable in which case we keep Sensor 2's estimate and discard Sensor 1's estimate.
3. Sensor 1 is reliable and Sensor 2 is unreliable in which case we keep Sensor 1's estimate and discard Sensor 2's estimate.
4. Both sensors are unreliable in which case we discard both estimates.

Estimates are fused using the KF or even covariance intersection [7] when the estimates are strongly correlated but their correlation is unknown. Whichever fusion algorithm is used we will denote the fused estimate as $\hat{x}_{1 \oplus 2}$ and the fused estimate covariance as $P_{1 \oplus 2}$. We shall denote the estimate and covariance obtained by using the consensus algorithm as $\hat{x}_{1 \uplus 2}$ and $P_{1 \uplus 2}$, respectively. Taking the expectation over the above scenarios:

$$
\begin{aligned}
& \hat{x}_{1 \uplus 2}(t \mid t') \\
& \quad = p(w_1(t) = 1 \mid R_1^{t'}) p(w_2(t) = 1 \mid R_1^{t'}) \hat{x}_{1 \oplus 2}(t \mid t') \\
& \quad + p(w_1(t) = 1 \mid R_1^{t'}) [1 - p(w_2(t) = 1 \mid R_1^{t'})] \hat{x}_1(t \mid t') \\
& \quad + [1 - p(w_1(t) = 1 \mid R_1^{t'})] p(w_2(t) = 1 \mid R_1^{t'}) \hat{x}_2(t \mid t')
\end{aligned}
$$

where either $t' = t$ or $t' = t - 1$, the latter corresponding to a prediction. Similarly for the covariance (see equation (1) at the top of this page). The reliability $p(w_{1 \uplus 2} = 1 \mid \cdot)$ assigned to the combined estimates is simply $p(w_{1 \uplus 2} = 1 \mid \cdot) = p(w_1 = 1 \mid \cdot) + p(w_2 = 1 \mid \cdot) - p(w_1 = 1 \mid \cdot) p(w_2 = 1 \mid \cdot)$.

To combine more than two estimates we can iterate over the entire set of estimates using the scheme described above. Iteration i combines estimate $\hat{x}_{i+1}(\cdot)$ with the consensus estimate $\hat{x}_{1 \uplus 2 \uplus \ldots \uplus i}(\cdot)$, using the consensus procedure described above. Note, that the probability that the combined estimates from all the sensors is reliable is unity, $p(w_{1 \uplus \ldots \uplus s} \mid \cdot) = 1$.

5.2 Calculating Sensor Reliability

It remains to describe how a sensor's reliability is calculated. Let $\hat{x}(t \mid t)$ denote the fault recovered estimates from all sensors supplying reports up to and including time t and let R_1^t denote all reports over all times 1 to t, from all sensors. We assume that the sensor's reliability process is Markovian and that the reports at time t depend only on the sensor's status at time t and not at previous times. The expression for the reliability, $p(w_s(t) = 1 \mid R_1^t)$, is shown at the top of the next page.

Since we assume that the sensors are independently reliable then:

$$
\begin{aligned}
& p(R(t) \mid w_s(t) = 1, R_1^{t-1}) \\
& \qquad = \prod_{i \in S} p(R_i(t) \mid w_s(t) = 1, R_1^{t-1}) \\
& \qquad = p(R_s(t) \mid w_s(t) = 1, R_1^{t-1}) \prod_{i \neq s} p(R_i(t) \mid R_1^{t-1}) \\
& \qquad = \frac{p(R_s(t) \mid w_s(t) = 1, R_1^{t-1})}{p(R_s(t) \mid R_1^{t-1})} \prod_{i \in S} p(R_i(t) \mid R_1^{t-1}) \\
& \qquad = \frac{p(R_s(t) \mid w_s(t) = 1, R_1^{t-1})}{p(R_s(t) \mid R_1^{t-1})} p(R(t) \mid R_1^{t-1})
\end{aligned}
$$

where $R_s(t)$ is the report from sensor s at time t. A value for $\frac{p(R(t))}{p(R(t) \mid R_1^{t-1})}$ can be calculated using $\max\{p(w_s(t) = 1 \mid \cdot)\} = 1$ from (2).

Thus, we can recursively calculate the reliability $p(w_s(t) = 1 \mid \cdot)$ of each sensor given a model of how its reliability varies over time, $p(w_s(t) = 1 \mid w_s(t-1), R_1^{t-1})$, and a model of the sensor's likelihood given estimate reports from the multi-sensor suite:

$$
L_s[R_1^t] \triangleq \frac{p(R_s(t) \mid w_s(t) = 1, R_1^{t-1})}{p(R_s(t) \mid R_1^{t-1})} \, .
$$

A sensor's reliability is assumed not to change unless there is evidence to indicate otherwise. Thus, the transition model is:

$$
\begin{aligned}
p(w_s(t) = 1 \mid w_s(t-1) = 1, R_1^{t-1}) &= 1 \, , \\
p(w_s(t) = 1 \mid w_s(t-1) = 0, R_1^{t-1}) &= 0 \, .
\end{aligned}
$$

We are free to choose the form of L_s subject to the constraint that a greater likelihood is assigned to the sensor failing the further its estimate is away from the truth. Thus, $L_s[R_1^t]$ must be a monotonically decreasing function of the *distance* between the sensor's estimate $\hat{x}_s(t \mid \cdot)$ and the target's state. An approximation for the target's state

$$p(w_s(t)=1 \mid R_1^t) = \frac{p(R(t) \mid w_s(t)=1, R_1^{t-1}) p(w_s(t)=1 \mid R_1^{t-1})}{p(R(t))}$$
$$= p(R(t) \mid w_s(t)=1, R_1^{t-1}) \sum_{w_s(t-1)\in\{0,1\}} \frac{p(w_s(t)=1 \mid w_s(t-1), R_1^{t-1}) p(w_s(t-1) \mid R_1^{t-1})}{p(R(t))}$$

is the *consensus* estimate $\bar{x}$ obtained from the sensor suite $\{\hat{x}_s(t \mid t-1)\}$. The consensus estimate is formed by combining predictions for the current target state from the sensor reports R_1^{t-1} using the procedure outline in Section 5.1. If $\bar{x}$ and P are the predicted consensus estimate and its covariance then:

$$\bar{x}(t) = \hat{x}_{1 \uplus 2 \uplus \ldots \uplus s}(t \mid t-1) \,,$$
$$P(t) = P_{1 \uplus 2 \uplus \ldots \uplus s}(t \mid t-1) \,.$$

We base L_s on the uniformly best constant power test [15] which is often used in fault detection [12, 5, 11]. A sensor is deemed to be faulty if the Mahalanobis distance between its output and its predicted output is greater than a threshold value β. We soften this rule so that outliers do not immediately lead to a sensor being designated unreliable.

$$L_s[R_1^t] = \begin{cases} c & \text{if } M_s \leq \beta \,, \\ c \exp(-(M_s - \beta)^2) & \text{otherwise.} \end{cases}$$

where:

$$M_s^2 = (\hat{x}_s(t \mid t) - \bar{x}(t))^T [P_s(t \mid t) + P(t)]^{-1} (\hat{x}_s(t \mid t) - \bar{x}(t)) \,.$$

Note, the constant c does not contribute to the posterior reliability distribution as it vanishes after normalisation. The β parameter is chosen according to the problem. For a 1D track we feel that $\beta = 3$ is a sensible value since, with this choice of β, a report only counts towards a sensor being unreliable if its estimate is more than 3 standard deviations from the consensus estimate.

We are free to choose the prior reliability $p(w_s = 1)$ for each sensor provided that at least one sensor is initially trustworthy (i.e. $p(w_s = 1) = 1$ for some sensor s). This sensor may not remain reliable but, in this case, another sensor must be either totally reliable or become reliable.

6 Illustrative Examples

We demonstrate the end-to-end efficacy of our two stage algorithm on a simple single target, 1D tracking problem for which the target is initially tracked by 10 sensors. These sensors are subject to various faults outlined in Section 3.

Our target process model is the near constant acceleration model (NCAM). However, we investigate the impact that this assumption has on the accuracy of our filter by simulating targets which do not move exactly according to an NCAM but still follow a smooth trajectory. With this mismatch between the process model and reality, we find that

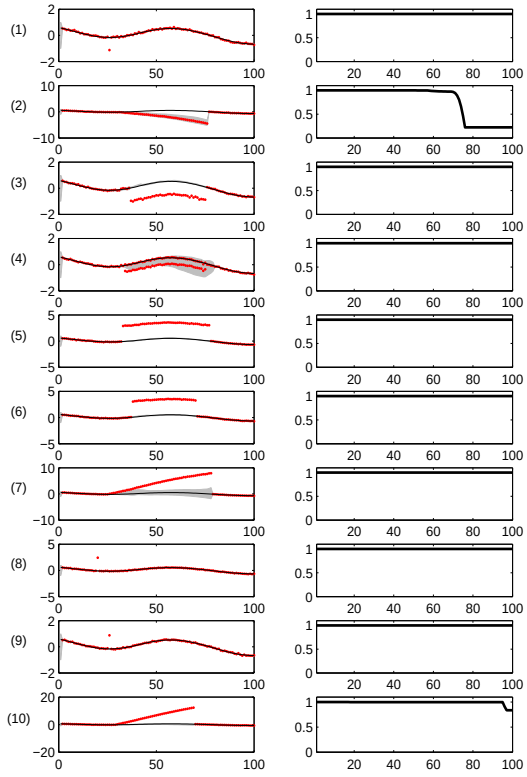

Figure 3: Ten sensors exhibiting various faults. The left column shows that faulty data ('.'), truth (solid line) and the fault recovered estimate first standard deviation confidence intervals (grey region). The right column shows the sensors' trustworthiness.

the fault recovery algorithm sometimes fails when sensors are subject to drift. [1]

Figure 3 shows 10 sensors exhibiting various faults. The fault recovered estimates are also shown along with the reliability $p(w_s = 1 \mid \cdot)$ over time for each sensor. A drifting sensor, Sensor 2, has failed to correct its fault. However, Stage 2 of our algorithm identifies this failing sensor and decreases its trustworthiness accordingly. Figure 4 shows the result of combining the outputs from the sensors using Stage 2.

In Figures 5 and 6 the above scenario is repeated but with a reduced set of sensors. We have chosen to use sensors $2, 3$ and 7. In this experiment it is these sensors alone which

[1] The NCAM acceleration process noise covariance is learned and the acceleration model parameter is represented within θ.

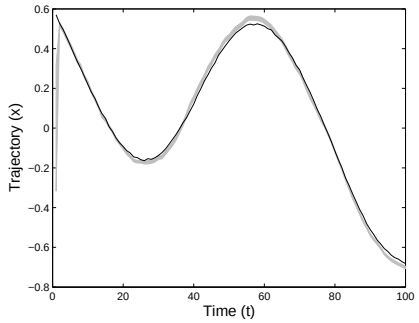

Figure 4: The result of combining the outputs from the sensors in Figure 3, using the algorithm described at the end of Section 5.

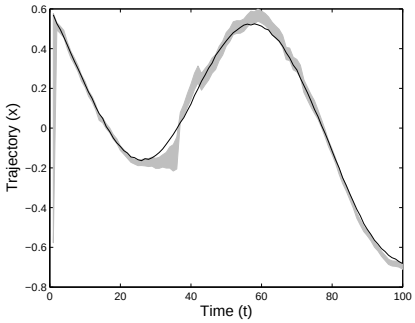

Figure 6: The result of combining the outputs from the sensors in Figure 5, using the algorithm described at the end of Section 5.

are used to determine the reliability shown in Figure 4 and ultimately the combined estimates in Figure 6. Again, the Stage 2 of our algorithm is able to correct for the deficiencies of Stage 1.

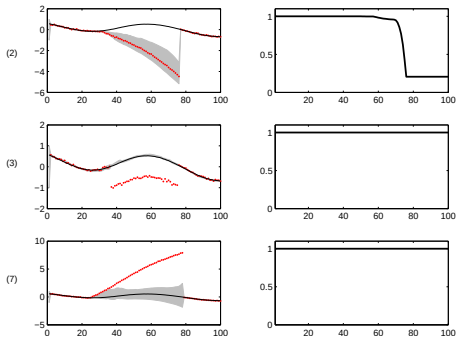

Figure 5: Three sensors exhibiting various faults. The left column shows that faulty data ('.'), truth (solid line) and the fault recovered estimate first standard deviation confidence intervals (grey region). The right column shows the sensors' trustworthiness.

Figures 7 and 8 show the results for a different failure recovery scenario. In this case, unlike the previous case, the sensor process models accurately describe the target trajectory. However, part way through monitoring the target, sensors 2 and 3 are knocked and as a result their observations incur a constant offset bias on top of any modelled fault that they are experiencing. Stage 1 fails to correct the bias induced by the unanticipated sensor knocks. However, Stage 2 successfully detects the inconsistent output of these sensors and reduces their trustworthiness accordingly. The fused estimate is shown in Figure 8.

7 Experiments

We examine the statistical characteristics of our algorithm and investigate the relative contributions that each stage makes to the fault recovery process. We examine four instances of our algorithm:

1. Neither fault recovery stages are employed. The fault sensor estimates are simply fused together.

2. Only stage 1 of the fault recovery process is used. The outputs from each sensor, which have been processed by the Stage 1 algorithm, are fused together.

3. Only state 2 of the fault recover process is used. Sensor estimates are assumed by each sensor to be fault free and are passed directly to Stage 2 without attempting to remove any faults from the estimates.

4. The end-to-end algorithm uses both stages to remove faults and unreliable sensors.

The system comprises five sensors and two randomly chosen sensors from the five undergo an unmodelled and unanticipated knock at time $t = 50$. The knock is simulated by adding a fixed value to the sensor's observations. This offset persists to the end of each run unlike the modelled faults which have all ended by $t = 80$. The modelled fault types and their parameters are chosen randomly.

The fault start times are restrained to lie within the interval $[20, 40]$ and the fault end times are constrained to lie within the interval $[70, 80]$ except for the spike fault which lasts for only one time step. Constraining the start and end times this way gives three distinct temporal regions in which we can compare our algorithms. Within the first region, up until time 20, no fault has occurred. Sensors, which have been fault corrected using the Stage 1 fault recovery algorithm, are reliable until $t = 50$ when two randomly chosen sensors are subject to a knock and become misaligned. This misalignment persists until the end of each run.

We gather 500 runs and calculate the normalised standard error (NSE) over time:

$$S(t) = E\left[(\hat{x}(t) - x(t))^T P^{-1}(t)(\hat{x}(t) - x(t))\right] .$$

For an estimator to be consistent the normalised standard error should be no greater than the cardinality of the state

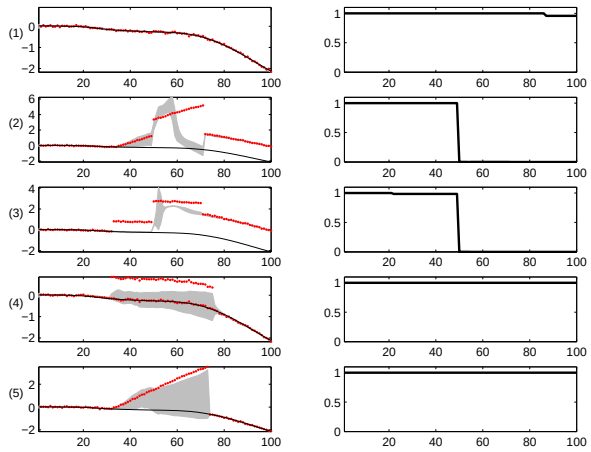

Figure 7: Five sensors exhibiting various faults. The left column shows that faulty data ('.'), truth (solid line) and the fault recovered estimate first standard deviation confidence intervals (grey region). The right column shows the sensors' trustworthiness.

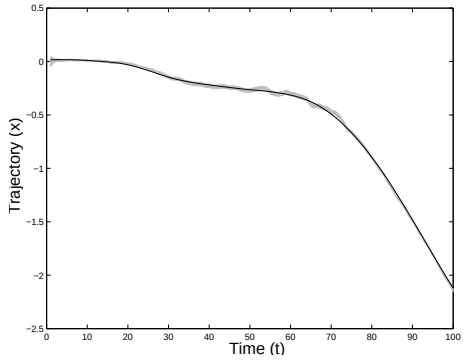

Figure 8: The result of combining the outputs from the sensors in Figure 7, using the algorithm described at the end of Section 5.

vector (i.e. 1 in this case). Ideally the value of S should be close to the cardinality of the state vector indicating that the estimate covariance is not too conservative. We also determine the accuracy of the filter. This is obtained from the the RMS error:

$$R(t) = \sqrt{E\left[(\hat{x}(t) - x(t))^T(\hat{x}(t) - x(t))\right]}\,.$$

Ideally, the RMS error is small indicating that the estimate is close to the truth. The results for our experiments are shown in Figures 9 and 10.

Figure 9 demonstrates that up until $t = 20$, when the faults set in, the algorithms have similar NSE and RMS errors. Any differences is caused by Stage 2 reducing the reliability of the sensors marginally. After $t = 20$, when the modelled faults set in, the algorithms which use Stage 1 remain consistent and the others diverge rapidly. The RMS value, especially in the range $[20, 50]$ is sensitive to the

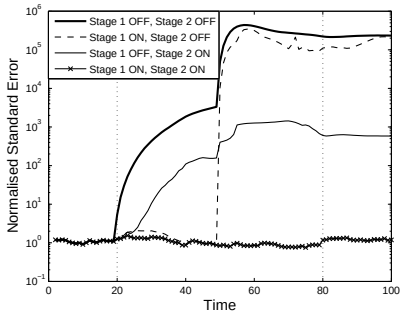

Figure 9: Result of Monte-Carlo Simulation showing the normalised standard error for the four fault recovery algorithms.

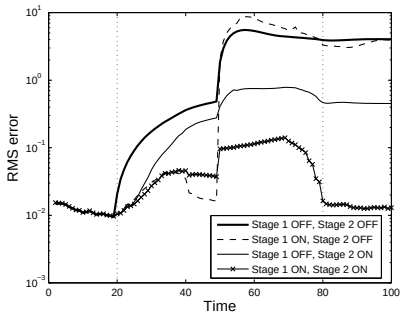

Figure 10: Result of Monte-Carlo Simulation showing the RMS error for the four fault recovery algorithms.

threshold value, β, for those algorithms which use Stage 2. As β decreases the RMS value increases. The reason for this is that Stage 2 of the algorithm is operating throughout the run. Fault recovered sensor estimates which are very informative with values very close to the truth can be discounted by the Stage 2 algorithm when they happen to be statistical outliers. This happens especially when offset faults have occurred. Algorithms which do not use Stage 2 would simply fuse these estimates leading to a smaller RMS error. If we wish to guarantee a consistent estimate throughout the run, however, then it is necessary to deploy both stages of the algorithm. After $t = 50$ the algorithm which uses both Stage 1 and Stage 2 is the only one which remains consistent.

8 Conclusions

We have presented a two stage Bayesian fault detection, identification and removal procedure for robust multi-sensor tracking in the presence of sensors which experience anticipated (i.e. modelled) and unanticipated fault types. Our approach is modular and decentralisable and computationally efficient as it performs fault recovery on each sensor individually during Stage 1. Stage 2 then de-emphasises sensor estimates which still contain (unanticipated) faults. Our approach was tested on two simulated problems, each exhibited a range of modelled fault types and also unanticipated faults such as mismatched target motion models and sensor

misalignment.

Future work will focus on developing efficient implementations of our approach and we will also investigate its extension to sequences of faults and simultaneous faults. Also, we intend to compare our approach with that of [10] in which sensor reliability is assigned to groups of sensors.

9 Acknowledgments

This research was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Networks) project and is jointly funded by a BAE Systems and EPSRC strategic partnership (EP/C548051/1).

References

[1] V. Aggarwal, K. Nagarajan, and K. Slatton, "Estimating failure modes using a multiple-model kalman filter," ASPL, Tech. Rep. no. Rep_2004-03-001, 2004, available at http://www.aspl.ece.ufl.edu/reports/Rep_2004-03-001.pdf.

[2] M. Basseville, "Detecting changes in signals and systems – a survey," *Automatica*, vol. 24, no. 3, pp. 309–326, 1988.

[3] E. Y. Chow and A. S. Willsky, "Bayesian design of decision rules for failure detection," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 20, no. 6, pp. 761–774, 1984.

[4] M. Desai and A. Ray, "A fault detection and isolation methodology – theory and application," in *Proceedings of American Control Conference, San Diego, USA*, 1984, pp. 262–270.

[5] G. Hang and Y. Min, "Data fusion in distributed multi-sensor system," *Geo-Spatial Information Science (quarterly)*, vol. 7, no. 3, pp. 426–482, 2004.

[6] R. Isermann, "Process fault detection based on modelling and estimation methods – a survey," *Automatica*, vol. 20, pp. 387–404, 1984.

[7] S. J. Julier and J. K. Uhlmann, "A non-divergent estimation algorithm in the presence of unknown correlations," in *Proceedings of the 1997 American Control Conference (ACC'97), Albuquerque, New Mexico*, 1997, pp. 2369–2373.

[8] T. Kobayashi and D. Simon, "Application of a bank of kalman filters for aircraft engine fault diagnosis," in *Proceedings of ASME Turbo Expo 2003, Power for Land, Sea, and Air, June 16-19 2003, Atlanta, Georgia, USA*, 2003.

[9] U. Lerner, R. Parr, D. Koller, and G. Biswas, "Bayesian fault detection and diagnosis in dynamic systems," in *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI)*, 2000, pp. 531–537.

[10] K. S. Ni and G. J.Pottie, "Sensor network data fault detection using bayesian maximum a posterior sensor selection and hierarchical bayesian space-time models," Center for Embedded Network Sensing, Tech. Rep. no. 68, 2009, available at http://repositories.cdlib.org/cens/techrep/68.

[11] I. Nikiforov, "Non-bayesian fault detection/isolation with nuisance parameters and constraints," in *Proceedings of the American Control Conference (ACC '07), Atlanta, Georgia, USA*, 2007, pp. 214–234.

[12] J. A. Ramagnoli and G. Stephanopoulos, "Rectification of process measurement data in the presence of gross errors," *Chemical Engineering Science*, vol. 36, no. 11, pp. 1849–1863, 1981.

[13] D. J. Salmond, "Mixture reduction algorithms for target tracking in clutter," in *Signal and Data Processing of Small Targets, SPIE 1305, O. E. Drummond, Ed.*, 1990, pp. 434–445.

[14] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. Kavuri, "A review of process fault detection and diagonsis. part 1: Quantitative model-based methods," *Computers and Chemical Engineering*, vol. 27, pp. 293–311, 2003.

[15] A. Wald, "Tests of statistical hypotheses concerning several parameters when the number of observations is large," *Transactions of the American Mathematical Society*, vol. 54, pp. 426–482, 1943.

[16] A. Willsky, "A survey of design methods for failure detection in dynamic systems," *Automatica*, vol. 12, pp. 601–611, 1976.