A Fast Covariance Union Algorithm for Inconsistent Sensor Data Fusion

XUEDONG WANG, SHUDONG SUN, YAQIONG LIU, AND ZHAN LIU
School of Mechanical Engineering, Northwestern Polytechnical University, Xi’an 710072, China
Key Laboratory of Industrial Engineering and Intelligent Manufacturing, Ministry of Industry and Information Technology, Xi’an 710072, China
Corresponding author: Shudong Sun (sdsun@nwpu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 51975482, and in part by the Shaanxi Provincial Key Research and Development Program of China Grant 2019ZDLGY14-10.

ABSTRACT We consider a challenging scenario in this research, where the sensors may receive spurious sensor data, potentially causing inconsistent state estimates. Covariance union (CU) is a fault-tolerant algorithm that can deal with inconsistent state estimation fusion. However, existing CU algorithms suffer from high computational costs due to optimizing nonlinear cost functions when generating fusion weights. To overcome this deficiency, an efficient CU algorithm named fast covariance union (FCU) is developed. We have proved that the fusion weight of FCU can be optimally generated by a closed-form algorithm without optimizing any nonlinear cost function, leading to better fusion efficiency. In addition, the FCU algorithm ensures the fused estimate be consistent as long as one of the estimates is consistent. Finally, the Monte Carlo simulation results show that the FCU algorithm has higher computational efficiency than the existing CU algorithms and handles the spurious sensor data fusion effectively.

INDEX TERMS Multisensor fusion, covariance union, fault tolerant.

I. INTRODUCTION
Multisensor fusion is an effective way to improve the reliability, robustness, and accuracy of the estimate system by combining information from different individual sensors. Multisensor systems are widely used in many areas, such as robotics [1], [2], and sensor networks [3], [4]. Generally, depending on whether the raw data are processed, the architecture of existing multisensor systems can be classified into two groups: centralized and distributed. In the former, raw data are directly sent to a central node and then are fused for state estimates, leading to optimal global estimates. In the latter, each sensor only handles local raw data and communicates with neighbors or a fusion center. Compared with the centralized fusion architecture, the distributed fusion architecture has less computation and communication costs, and it is robust to a single point failure and enjoys good scalability.

Research on distributed data fusion has increasingly gained attention in recent years. It can be classified into two main types [5]: 1) fusion for deciding on a hypothesis, such as detecting the presence of targets [6], [7] or classifying a signal [8]; 2) fusion for target tracking [9]–[12].

The associate editor coordinating the review of this manuscript and approving it for publication was Dusmanta Kumar Mohanta.

In the distributed target tracking fusion problem, consistency is one of the most important criteria for evaluating the reliability of an estimator [13], [14]. However, getting a consistent estimate in a distributed manner is challenging, owing to unknown inter-estimate correlations and inconsistent estimates. Covariance intersection (CI) [15] is a well-known consistent fusion approach, which can obtain consistent estimates even facing an unknown degree of inter-estimate correlation. It has been widely used in many fields [12], [16]–[18]. Practically, sensors produce spurious data because the sensor measurements are corrupted by unexpected uncertainties, such as permanent failures, spike faults, slowly developing failures, and sensor glitches. In these cases, the estimates provided by the sensor may be inconsistent due to unmodeled faults [10], [19]. However, fusing inconsistent estimates with the consistent estimates in the CI algorithm may leads to severely inaccurate results [11].

Compared with CI, covariance union (CU) is more fault-tolerant. The idea of the CU algorithm was first proposed by Uhlmann [11] in 2003. Although the spurious estimates are unidentified, the consistency can be guaranteed by extending the fused covariance to cover all local means and covariances. The specific applications of the CU algorithm were proposed in [20] and [21]. A generalized CU algorithm was proposed in [22], which is an amalgamation of standard mixture
reduction and CU. The generalized CU gives a tighter covariance matrix than that in CU. Bochardt and Uhlmann [23] proposed a general CU algorithm by applying a covariance addition to the CU algorithm. The solution of this general CU is equivalent to the minimum enclosing ellipsoid. However, these CU algorithms suffer from high computational costs due to optimizing nonlinear cost functions when generating fusion weights [10].

In this paper, a fast covariance union (FCU) algorithm is proposed to solve the multisensor fusion problem with spurious sensor data. The main innovations of this work are outlined as follows:

1) A computationally efficient CU algorithm called FCU is proposed.
2) We have proved that the fusion weight of FCU can be optimally generated by a closed-form algorithm without optimizing any nonlinear cost function.
3) The FCU algorithm ensures the fused estimate be consistent as long as one of the estimates is consistent.

II. SYSTEM MODEL AND PROBLEM STATEMENT

Consider a sensor network composed of M nodes, where each node can communicate with neighboring nodes. At time k + 1, node i takes measurement of the target which the sensor network interests. A state-space model associated with the target and the measurement is given as follows:

\[
\begin{align*}
\dot{X}_{k+1} &= f (\dot{X}_k) + w_k \\
Z_{k+1}^i &= g (\dot{X}_{k+1}) + v_{k+1}^i
\end{align*}
\]

where \(\dot{X}_{k+1}\) is the target state to be estimated, \(Z_{k+1}^i\) the measurement of the target taken by node \(i\), \(f\) and \(g\) are nonlinear functions, \(w_k\) and \(v_{k+1}^i\) are zero-mean Gaussian noise with covariance \(Q_k\) and \(R_k\), respectively.

We denote the estimate of target obtained by node \(i\) based on local measurement \(Z_{k+1}^i\) as \((\hat{X}_{k+1}^i, P_{k+1}^i)\), where \(\hat{X}_{k+1}^i\) denotes the estimated state vector and \(P_{k+1}^i\) denotes the estimated error covariance matrix. When \(P_{k+1}^i \geq E[(\hat{X}_{k+1}^i - \hat{X}_{k}^i)(\hat{X}_{k+1}^i - \hat{X}_{k}^i)^T]\), the \((\hat{X}_{k+1}^i, P_{k+1}^i)\) is a consistent estimate [24].

In many practical situations, sensor noise may be corrupted by unexpected uncertainties. In these cases, the sensor noise is biased [10], [26]. Since the unexpected uncertainties are not attributable to the inherent noise, they are difficult to model. Subsequently, the target state estimated by node \(i\) may be inconsistent.

Given two mean and covariance estimates \((X_a, P_a)\) and \((X_b, P_b)\), with \(X_a = [a_1, \ldots, a_n]^T\), \(X_b = [b_1, \ldots, b_n]^T\),

\[
\begin{align*}
P_a &= \begin{bmatrix}
\sigma_{11} & \cdots & \sigma_{1n} \\
\vdots & \ddots & \vdots \\
\sigma_{n1} & \cdots & \sigma_{nn}
\end{bmatrix} \\
P_b &= \begin{bmatrix}
\sigma_{11} & \cdots & \sigma_{1n} \\
\vdots & \ddots & \vdots \\
\sigma_{n1} & \cdots & \sigma_{nn}
\end{bmatrix}
\end{align*}
\]

where \(n\) is the dimension of \(X\).

When \((X_a, P_a)\) and \((X_b, P_b)\) are all consistent estimates, Julier and Uhlmann [15] introduced CI algorithm, which can yield a consistent estimate even fusing estimates are with unknown correlations. However, when either \((X_a, P_a)\) or \((X_b, P_b)\) is an inconsistent estimate of the target, the fusion of CI cannot guarantee consistency [25]. To resolve this problem, Uhlmann [11] proposed a CU algorithm. No matter which estimate is spurious, the CU combines these estimates to form a mean and covariance estimate \((u, U)\), which is guaranteed to be consistent.

III. FAST COVARIANCE UNION ALGORITHM

Matzka and Altdorfer [21] proposed a CU algorithm, referred to as MCU in this paper. The algorithm is given as follows:

\[
\begin{align*}
u &= (1 - \omega)X_a + \omega X_b \\
U^a &= P_a + (u - X_a)(u - X_a)^T \\
U^b &= P_b + (u - X_b)(u - X_b)^T \\
U &= \max(U^a, U^b) \\
\omega &= \min \text{tr} (U)
\end{align*}
\]

where \(\max(U^a, U^b)\) is the element-wise maximum value of the matrices \(U^a\) and \(U^b\).

In the MCU algorithm, solving for weight \(\omega\) is a nonlinear convex optimization problem, suffering from high computational cost. Instead, we use the minimization trace optimization instead of (7),

\[
\omega = \min \text{tr} (U)
\]

Substituting (3) into (4) and (5) yields,

\[
\begin{align*}
U^a &= \begin{bmatrix}
\omega^2(b_1 - a_1)^2 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \omega^2(b_n - a_n)^2
\end{bmatrix} \\
U^b &= \begin{bmatrix}
(1 - \omega)^2(b_1 - a_1)^2 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & (1 - \omega)^2(b_n - a_n)^2
\end{bmatrix}
\end{align*}
\]

Set

\[
y_i = \max (U_{i_1}^a, U_{i_1}^b), i = 1, \ldots, n
\]

\[
Y_\omega = \sum_{i=1}^{n} y_i
\]

Then, from (6), (8)-(10) we can get \(\min \text{tr} (U) = \min Y_\omega\). When \(U_{i_1}^a = U_{i_1}^b, i = 1, \ldots, n\), we can get weight \(\hat{\omega}\) by (11),

\[
\hat{\omega}_i = \left(1 + \frac{(\sigma_{a}^b - \sigma_{a}^a)}{(b_i - a_i)^2}\right) / 2
\]
Due to $0 \leq \omega \leq 1$, then
\[
\hat{\omega}_i = \begin{cases} 0, & \hat{\omega}_i < 0 \\ \hat{\omega}_i, & 0 \leq \hat{\omega}_i \leq 1 \\ 1, & \hat{\omega}_i > 1 \end{cases}
\] (12)

From (9), we can get
\[
y_i = \begin{cases} U_{i,j}, & \omega < \hat{\omega}_i \\ U_{i,j}, & \omega \geq \hat{\omega}_i \end{cases}
\]

We denote the non-decreasing order of $\hat{\omega}_1, \ldots, \hat{\omega}_n$, by $\hat{\omega}_{\rho(k)}, k = 1, \ldots, n$, where $\rho(k)$ is the index of the $k$th largest number of $\hat{\omega}_1, \ldots, \hat{\omega}_n$. From (10), we can get (13), as shown at the bottom of the page.

**Theorem 1:** The minimum value of $Y_\omega$ exists at $\omega \in \hat{\omega}_{\rho(1)}, \hat{\omega}_{\rho(n)}$.

**Proof:** When $\omega \in [0, \hat{\omega}_{\rho(1)}]$, if $\hat{\omega}_{\rho(1)} = 0$, then $\omega \in [0, \hat{\omega}_{\rho(1)}]$ is $\emptyset$. In $\omega \in [\hat{\omega}_{\rho(1)}, \hat{\omega}_{\rho(n)}]$, we can get $Y_0 = Y_{\hat{\omega}_{\rho(1)}}$, where $Y_0$ and $Y_{\hat{\omega}_{\rho(1)}}$ are the value of $Y_\omega$, when $\omega = 0$ and $\omega = \hat{\omega}_{\rho(1)}$, respectively.

If $0 < \hat{\omega}_{\rho(1)} < 1$, the first derivative of $\omega$ satisfies
\[
\frac{\partial Y_\omega}{\partial \omega} = \frac{\partial}{\partial \omega} \left( \sum_{k=1}^{n} U_{\rho(k), \rho(k)}^a + \sum_{k=1+1}^{n} U_{\rho(k), \rho(k)}^b \right)
\]
\[
= (2\omega - 2) \sum_{k=1}^{n} (b_{\rho(k)} - a_{\rho(k)})^2 
< 0
\]

therefore, $Y_0 > Y_{\hat{\omega}_{\rho(1)}}$. Similarly, we can get $Y_1 \geq Y_{\hat{\omega}_{\rho(n)}}$. \hfill $\square$

**Theorem 2:** If $Y_\omega$ has an extreme value where $\omega \in \hat{\omega}_{\rho(j), \rho(j+1)}$, $j = 1, \ldots, n-2$, or $\omega \in \hat{\omega}_{\rho(n-1), \hat{\omega}_{\rho(n)}}$, it is an unique local minimum value.

**Proof:** If $Y_\omega$ has an extreme value when $\omega \in \hat{\omega}_{\rho(j), \rho(j+1)}$, $j = 1, \ldots, n-2$, or $\omega \in \hat{\omega}_{\rho(n-1), \hat{\omega}_{\rho(n)}}$.

\[
Y_\omega = \begin{cases} \sum_{k=1}^{n} U_{\rho(k), \rho(k)}^a, & \omega \in [0, \hat{\omega}_{\rho(1)}] \\ \sum_{k=1}^{n} U_{\rho(k), \rho(k)}^a + \sum_{k=1+1}^{n} U_{\rho(k), \rho(k)}^b, & \omega \in [\hat{\omega}_{\rho(1)}, \hat{\omega}_{\rho(2)}] \\ \vdots & \vdots \\ \sum_{k=n-j}^{n} U_{\rho(k), \rho(k)}^a + \sum_{k=n+1}^{n} U_{\rho(k), \rho(k)}^b, & \omega \in [\hat{\omega}_{\rho(j)}, \hat{\omega}_{\rho(j+1)}] \\ \vdots & \vdots \\ \sum_{k=1}^{n} U_{\rho(k), \rho(k)}^a, & \omega \in [\hat{\omega}_{\rho(n-1)}, \hat{\omega}_{\rho(n)}] \\ \sum_{k=1}^{n} U_{\rho(k), \rho(k)}^a, & \omega \in (\hat{\omega}_{\rho(n)}, 1] \end{cases}
\]

i.e., $\frac{\partial Y_\omega}{\partial \omega} = 0$, that is
\[
\frac{\partial Y_\omega}{\partial \omega} = \frac{\partial}{\partial \omega} \left( \sum_{k=1}^{l} U_{\rho(k), \rho(k)}^a + \sum_{k=l+1}^{n} U_{\rho(k), \rho(k)}^b \right)
\]
\[
= 2\omega \sum_{k=1}^{n} (b_{\rho(k)} - a_{\rho(k)})^2 - 2 \sum_{k=l+1}^{n} (b_{\rho(k)} - a_{\rho(k)})^2
\]
\[
= 0
\]

where $l = 1, \ldots, n-1$.

The second derivative of $\omega$ is
\[
\frac{\partial^2 Y_\omega}{\partial \omega^2} = \frac{\partial^2}{\partial \omega^2} \left( \sum_{k=1}^{l} U_{\rho(k), \rho(k)}^a + \sum_{k=l+1}^{n} U_{\rho(k), \rho(k)}^b \right)
\]
\[
= 2 \sum_{k=1}^{n} (b_{\rho(k)} - a_{\rho(k)})^2
\]
\[
> 0
\]

Thus, $Y_\omega$ is convex in $\omega \in [\hat{\omega}_{\rho(j)}, \hat{\omega}_{\rho(j+1)}]$, $j = 1, \ldots, n-2$, or $\omega \in [\hat{\omega}_{\rho(n-1)}, \hat{\omega}_{\rho(n)}]$. Furthermore, the positive second derivative indicates that the extreme is a local minimum value.

We now prove the uniqueness. From (14) we can get
\[
\frac{\partial^2 Y_\omega}{\partial \omega^2} = \frac{\partial^2}{\partial \omega^2} \left( \sum_{k=1}^{l} U_{\rho(k), \rho(k)}^a + \sum_{k=l+1}^{n} U_{\rho(k), \rho(k)}^b \right)
\]
\[
= 2 \sum_{k=1}^{n} (b_{\rho(k)} - a_{\rho(k)})^2
\]
\[
= \frac{n}{n+1} \sum_{k=1}^{n} (b_{\rho(k)} - a_{\rho(k)})^2
\]

$\hat{\omega}_{\rho(l)}$ increases with the increase of $l$, while $\hat{\omega}_{\rho(l)}$ decreases with the increase of $l$ from (15). Thus, we cannot get $\frac{\partial Y_\omega}{\partial \omega} = 0$ when $\omega \in [\hat{\omega}_{\rho(l+1)}, \hat{\omega}_{\rho(n)}]$. Similarly, we cannot get $\frac{\partial Y_\omega}{\partial \omega} = 0$ when $\omega \in [\hat{\omega}_{\rho(2)}, \hat{\omega}_{\rho(n)}]$.

Therefore, we cannot get another local minimum value when $\omega \in [\hat{\omega}_{\rho(1)}, \hat{\omega}_{\rho(l)}]$ or $\omega \in [\hat{\omega}_{\rho(l+1)}, \hat{\omega}_{\rho(n)}]$.

\hfill $\square$
Theorem 3: If \( Y_\omega \) has a local minimum value \( Y_{\min} \) when \( \omega \in \left[ \hat{\omega}_{p(j)}, \hat{\omega}_{p(j+1)} \right], j = 1, \ldots, n - 2, \) or \( \omega \in \left[ \hat{\omega}_{p(n-1)}, \hat{\omega}_{p(n)} \right] \), \( Y_{\min} \) is the minimum value of \( Y_\omega \) when \( \omega \in \left[ \hat{\omega}_{p(1)}, \hat{\omega}_{p(n)} \right] \).

Proof: From Theorem 2, we know that \( Y_\omega \) has only one local minimum value in \( \omega \in \left[ \hat{\omega}_{p(1)}, \hat{\omega}_{p(n)} \right] \). Therefore, we only need to prove \( Y_{\min} \leq Y_{\omega_{\hat{p}(k)}}, \ k = 1, \ldots, n. \) Due to \( Y_{\min} \) is a local minimum value at \( \omega \in \left[ \hat{\omega}_{p(j)}, \hat{\omega}_{p(j+1)} \right], \) thus, \( Y_{\min} < Y_{\hat{\omega}_{p(j+1)}} \) and \( Y_{\min} \leq Y_{\hat{\omega}_{p(n)}} \). Let us consider the two cases: \( \omega \in \left[ \hat{\omega}_{p(j+1)}, \hat{\omega}_{p(n)} \right] \) and \( \omega \in \left[ \hat{\omega}_{p(1)}, \hat{\omega}_{p(j)} \right] \).

When \( \omega \in \left[ \hat{\omega}_{p(j+1)}, \hat{\omega}_{p(n)} \right], \) then \( Y_\omega = \sum_{k=1}^{n} U_{\rho(k),\rho(k)} + \sum_{k=j+2}^{n} U_{\rho(k),\rho(k)} \). The first derivative of \( \omega \) is

\[
\frac{\partial Y_\omega}{\partial \omega} = \frac{\partial}{\partial \omega} \left( \sum_{k=1}^{j+1} U_{\rho(k),\rho(k)} + \sum_{k=j+2}^{n} U_{\rho(k),\rho(k)} \right) = 2\omega \sum_{k=1}^{n} (b_{\rho(k)} - a_{\rho(k)})^2 - 2 \sum_{k=j+2}^{n} (b_{\rho(k)} - a_{\rho(k)})^2 = 0.
\]

From (14) we know that

\[
2\omega \sum_{k=1}^{n} (b_{\rho(k)} - a_{\rho(k)})^2 - 2 \sum_{k=j+2}^{n} (b_{\rho(k)} - a_{\rho(k)})^2 = 0.
\]

Then

\[
\frac{\partial Y_\omega}{\partial \omega} = 2\omega \sum_{k=1}^{n} (b_{\rho(k)} - a_{\rho(k)})^2 - 2 \sum_{k=j+2}^{n} (b_{\rho(k)} - a_{\rho(k)})^2 > 0 \quad (16)
\]

Therefore, \( Y_{\min} < Y_{\hat{\omega}_{p(j+1)}} < Y_{\hat{\omega}_{p(n)}} \). Assume \( Y_{\min} < Y_{\hat{\omega}_{p(j+1)}} < Y_{\hat{\omega}_{p(n)}} \). Assume \( \omega \in \left[ \hat{\omega}_{p(j+1)}, \hat{\omega}_{p(n)} \right], \) \( z = 2, \ldots, n - (j + 2) \). When \( \omega \in \left[ \hat{\omega}_{p(j+2)}, \hat{\omega}_{p(j+3)} \right], \) \( Y_\omega \) is monotonically increasing when \( \omega \in \left[ \hat{\omega}_{p(j+1)}, \hat{\omega}_{p(j+2)} \right], \) the minimum value of \( Y_\omega \) when \( \omega \in \left[ \hat{\omega}_{p(j+1)}, \hat{\omega}_{p(j+2)} \right]. \)

Corollary 1: If \( Y_\omega \) has no extreme value when \( \omega \in \left[ \hat{\omega}_{p(j)}, \hat{\omega}_{p(j+1)} \right], j = 1, \ldots, n - 2, \) or \( \omega \in \left[ \hat{\omega}_{p(n-1)}, \hat{\omega}_{p(n)} \right], \) the minimum value of \( Y_\omega \) in \( \omega \in \left[ \hat{\omega}_{p(1)}, \hat{\omega}_{p(n)} \right] \) is \( Y_{\min} \).

Proof: If \( Y_\omega \) has no extreme value, from (16), \( Y_\omega \) is monotonically increasing or decreasing when \( \omega \in \left[ \hat{\omega}_{p(j)}, \hat{\omega}_{p(j+1)} \right], j = 1, \ldots, n - 2, \) or \( \omega \in \left[ \hat{\omega}_{p(n-1)}, \hat{\omega}_{p(n)} \right] \).

Corollary 2: If \( Y_\omega \) is monotonically decreasing when \( \omega \in \left[ \hat{\omega}_{p(j+1)}, \hat{\omega}_{p(j+2)} \right], j = 1, \ldots, n - 2, \) and \( Y_\omega \) is monotonically increasing when \( \omega \in \left[ \hat{\omega}_{p(j)}, \hat{\omega}_{p(n)} \right], \) the minimum value of \( Y_\omega \) when \( \omega \in \left[ \hat{\omega}_{p(j)}, \hat{\omega}_{p(n)} \right] \) is \( Y_{\min} \).

Proof: If \( Y_\omega \) is monotonically decreasing when \( \omega \in \left[ \hat{\omega}_{p(j+1)}, \hat{\omega}_{p(j+2)} \right], j = 1, \ldots, n - 2, \) and \( Y_\omega \) is monotonically increasing when \( \omega \in \left[ \hat{\omega}_{p(j)}, \hat{\omega}_{p(n)} \right], \) from (16), \( Y_\omega \) is monotonically increasing when \( \omega \in \left[ \hat{\omega}_{p(j+1)}, \hat{\omega}_{p(n)} \right]. \)

Corollary 3: If \( Y_\omega \) is monotonically increasing when \( \omega \in \left[ \hat{\omega}_{p(j+1)}, \hat{\omega}_{p(j+2)} \right], j = 1, \ldots, n - 2, \) the minimum value of \( Y_\omega \) when \( \omega \in \left[ \hat{\omega}_{p(j+1)}, \hat{\omega}_{p(n)} \right] \) is \( Y_{\min} \).

Proof: If \( Y_\omega \) is monotonically increasing when \( \omega \in \left[ \hat{\omega}_{p(j+1)}, \hat{\omega}_{p(j+2)} \right], j = 1, \ldots, n - 2, \) from (16), \( Y_\omega \) is monotonically increasing when \( \omega \in \left[ \hat{\omega}_{p(j+1)}, \hat{\omega}_{p(n)} \right]. \)

From Theorems 1-3 and Corollaries 1-3, we can get the following conclusions: Firstly, the minimum value of \( Y_\omega \) exists in \( \omega \in \left[ \hat{\omega}_{p(1)}, \hat{\omega}_{p(n)} \right]. \) Secondly, if \( Y_\omega \) has an extreme value when \( \omega \in \left[ \hat{\omega}_{p(1)}, \hat{\omega}_{p(n)} \right], \) it is the minimum value; Thirdly, if \( Y_\omega \) is monotonically decreasing when \( \omega \in \left[ \hat{\omega}_{p(j+1)}, \hat{\omega}_{p(j+2)} \right], j = 1, \ldots, n - 2, \) and \( Y_\omega \) is monotonically increasing when \( \omega \in \left[ \hat{\omega}_{p(j)}, \hat{\omega}_{p(n)} \right], \) the minimum value of \( Y_\omega \) when \( \omega \in \left[ \hat{\omega}_{p(j)}, \hat{\omega}_{p(n)} \right] \) is \( Y_{\min} \).

Based on this algorithm, we can get an exact value of the nonlinear-convex optimization problem. Moreover, since the nonlinear-convex optimization problem has been transformed.

Algorithm 1 FCU Algorithm

Input: \((X_{a}, P_{a}), (X_{b}, P_{b})\)
Output: \((u, U)\)

1. Calculate \( \hat{\omega}_{p(i)}, i = 1, \ldots, n, \) using (11) and (12), sort them in non- decreasing order get \( \hat{\omega}_{p(k)}, k = 1, \ldots, n \)
2. Using (14) check whether \( Y_\omega \) has extreme value when \( \omega \in \left[ \hat{\omega}_{p(1)}, \hat{\omega}_{p(n)} \right] \)
3. If \( Y_\omega \) has extreme value when \( \omega \in \left[ \hat{\omega}_{p(1)}, \hat{\omega}_{p(n)} \right], \) then calculate the extreme point of \( \omega \) using (15)
5. else if \( Y_\omega \) is monotonically decreasing when \( \omega \in \left[ \hat{\omega}_{p(1)}, \hat{\omega}_{p(n)} \right], j = 1, \ldots, n - 2, \) \( Y_\omega \) is monotonically increasing when \( \omega \in \left[ \hat{\omega}_{p(1)}, \hat{\omega}_{p(n)} \right], \) then
7. else if \( Y_\omega \) is monotonically decreasing when \( \omega \in \left[ \hat{\omega}_{p(1)}, \hat{\omega}_{p(n)} \right], \) then
9. end
12. Calculate \((u, U)\) using \( \omega \) and (3)-(6)
into a linear function, the computation time has been greatly reduced.

**IV. SIMULATIONS**

The performance of the FCU algorithm was evaluated by numerical simulation under two conditions: 1) normal sensor data, and 2) abnormal sensor data. We compared the performance of FCU with the following four algorithms: 1) MCU [21]; 2) fast fault-tolerant convex combination (FFCC) algorithm, which is an extension of the CI algorithm and can handle inconsistent sensor data fusion [26]; 3) the CU algorithm proposed by Julier (JCU) [20], and 4) CI [15].

In this simulation, two sensors $S_1$ and $S_2$ were used to track a moving target with constant-velocity. We refer to the moving target as $O$, the state of the target at time $k$ is denoted as $\hat{X}_k^O = [\hat{x}_k^O; \hat{v}_k^O; \hat{a}_k^O]^T$, where $[\hat{x}_k^O; \hat{v}_k^O]^T$ and $[\hat{v}_k^O; \hat{a}_k^O]^T$ denote target’s position and velocity, respectively. The state transition model of the moving target is given by (17).

$$\tilde{X}_{k+1}^O = \Phi \tilde{X}_k^O + \Gamma \xi_k$$

with

$$\Phi = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} \Delta t^2/2 & 0 & 0 & 0 \\ \Delta t & 0 & 0 & 0 \\ 0 & \Delta t^2/2 & 0 & 0 \\ 0 & 0 & \Delta t & 0 \end{bmatrix}$$

where $\xi_k$ is a zero-mean white Gaussian process noise with covariance $\mathbf{E}[\xi_k]$. The measurements of the two sensors at time $k+1$ are modeled as:

$$\tilde{Z}_{k+1}^i = \mathbf{H} \tilde{X}_{k+1}^O + \mathbf{y}_{k+1}^i, \ i = 1, 2$$

with

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

where $\mathbf{y}_{k+1}^i$ is a zero-mean white Gaussian process noise with covariance $\mathbf{E}[\mathbf{y}_{k+1}^i]$. At time $k$, the state and covariance of the target estimated by $S_i$ are denoted as $X_{k}^{S_i,O}, P_{k}^{S_i,O}$. When sensor $S_i$ obtains its measurement $\tilde{Z}_{k+1}^i$ at time $k+1$, we first use the state transition model which is given in (17) to predict the state of target as follows:

$$X_{k+1|k}^{S_i,O} = \Phi X_{k|k}^{S_i,O}$$

The covariance term of the target estimated by $S_i$ is propagated as follows:

$$P_{k+1|k}^{S_i,O} = \Phi P_{k|k}^{S_i,O} \Phi^T + \Gamma \mathbf{E}[\xi_k] \Gamma^T$$

Using the measurement $\tilde{Z}_{k+1}^{S_i}$, the state and covariance of the target are updated by the Kalman filter as follows:

$$X_{k+1}^{S_i,O^-} = X_{k+1|k}^{S_i,O} - K_{k+1}^{S_i} \left( \tilde{Z}_{k+1}^{S_i} - \mathbf{H} X_{k+1|k}^{S_i,O} \right)$$

$$P_{k+1}^{S_i,O^-} = \left( I - K_{k+1}^{S_i} \mathbf{H} \right) P_{k+1|k}^{S_i,O}$$

with

$$K_{k+1}^{S_i} = P_{k+1|k}^{S_i,O} \mathbf{H}^T (\mathbf{V}_{k+1}^{S_i})^{-1}$$

$$\mathbf{V}_{k+1}^{S_i} = \mathbf{H} P_{k+1|k}^{S_i,O} \mathbf{H}^T + \sigma_{k+1}^i.$$  

Finally, $S_i$’s local estimate $\left( X_{k+1}^{S_i,O^-}, P_{k+1}^{S_i,O^-} \right)$ are fused with another sensor $S_j$’s local estimate $\left( X_{k+1}^{S_j,O^-}, P_{k+1}^{S_j,O^-} \right)$ by FCU as follows:

$$\tilde{X}_{k+1}^O = (1 - \omega_{k+1}) X_{k+1}^{S_i,O^-} + \omega_{k+1} X_{k+1}^{S_j,O^-}$$

$$\tilde{P}_{k+1} = \max \left( T_i, T_j \right)$$

$$\omega_{k+1} = \min \left( \frac{1}{2}, \frac{T_i}{\max(T_i, T_j)} \right)$$

with

$$T_i = P_{k+1}^{S_i,O^-} + \left( \tilde{X}_{k+1}^O - X_{k+1}^{S_i,O^-} \right) \left( \tilde{X}_{k+1}^O - X_{k+1}^{S_i,O^-} \right)^T$$

$$T_j = P_{k+1}^{S_j,O^-} + \left( \tilde{X}_{k+1}^O - X_{k+1}^{S_j,O^-} \right) \left( \tilde{X}_{k+1}^O - X_{k+1}^{S_j,O^-} \right)^T$$

where $j \in \{ 1, 2 \} \setminus \max(T_i, T_j)$ is the element-wise maximum value of the matrices $T_i$ and $T_j$.

The fused results $\left( \tilde{X}_{k+1}^O, \tilde{P}_{k+1} \right)$ are the final state and covariance of the target estimated by the two sensors at time $k+1$. Then the fused results stored in $S_1$ and $S_2$, respectively, i.e., $X_{k+1}^{S_i} = \tilde{X}_{k+1}^O, P_{k+1}^{S_i} = \tilde{P}_{k+1}$.

In this simulation, we use a computer with i7 CPU 3.6GHz, RAM 8GB, and Matlab 2008a. TABLE 1 lists the simulation parameters.

**TABLE 1. Simulation parameters.**

| Parameter description | Value |
|-----------------------|-------|
| Sample step $\Delta t$ | 1s |
| Initial state of target $X_1^O$ | $[0 \ 5 \ 0 \ 5]^T$ |
| Initial state covariance of target $P_{1}^{S_i,O}$ | $I$ |
| Monte Carlo runs $S_1$’s sensing noise $\mathbf{y}_k^{S_1}$ | $N \left( (0,0), \left( (2m)^2, (2m)^2 \right) \right)$ |
| $S_2$’s sensing noise $\mathbf{y}_k^{S_2}$ | $N \left( (0,0), \left( (3m)^2, (3m)^2 \right) \right)$ |
| Object’s motion noise $\xi_k$ | $N \left( (0,0), \left( (1m/s)^2, (1m/s)^2 \right) \right)$ |

We chose root mean square error (RMSE), normalized estimation error squared (NEES) [27] and time to measure the accuracy, consistency and computational efficiency of the five algorithms, respectively.
A. SIMULATION WITH NORMAL SENSOR DATA

In this simulation, the measurements of the two sensors are affected by normal sensor noise, which is given in (20). Figs. 1 and 2 show the RMSE and NEES over time of the position estimation of the target, respectively. TABLE 2 shows the average results over the whole simulation time. It is noteworthy that the accuracy and consistency of the MCU and FCU algorithms are very close. This implies no significant difference between the results obtained by the minimization trace optimization and minimization determinant optimization in weight \( \omega \) calculation. From Fig. 2, we can see that the NEES values are around the ideal value of 2 in the CI algorithm. For MCU, JCU, FCU, and FFCC algorithms, the NEES values are much smaller than 2. This is because that using these fusion algorithms, the covariance is enlarged to cover all local means and covariances.

The average computational time of the five fusion algorithms in each Monte Carlo run is given in TABLE 2. The weights \( \omega \) in MCU, CI, and JCU algorithms are computed by the “fminbnd” optimization function in the Matlab. From TABLE 2, it can be seen that the FCU algorithm has the shortest calculation time, while the JCU algorithm has the largest calculation time among the five algorithms.

| Algorithm     | MCU | FFCC | JCU | CI  | FCU |
|---------------|-----|------|-----|-----|-----|
| Average RMSE (m) | 1.9353 | 2.0252 | 1.9273 | 1.9659 | 1.9177 |
| Average NEES  | 0.8310 | 0.6601 | 0.7175 | 1.9756 | 0.8141 |
| Time (s)      | 0.2090 | 0.0045 | 1.2981 | 0.4650 | 0.0042 |

B. SIMULATION WITH ABNORMAL SENSOR DATA

In TABLE 1, it can be seen that \( S_1 \) and \( S_2 \) are heterogeneous sensors, and the sensor noise of \( S_1 \) is smaller than \( S_2 \). To verify the influence of inconsistent estimates on the fusion results, three conditions are considered: 1) the sensor data of \( S_1 \) is abnormal; 2) the sensor data of \( S_2 \) is abnormal, and 3) \( S_1 \) and \( S_2 \) have the same processing sensor noise, and the sensor data of \( S_2 \) is abnormal.

1) THE SENSOR DATA OF \( S_1 \) IS ABNORMAL

We assume that the measurements of \( S_1 \) are corrupted by a biased Gaussian noise \( \eta_{k+1} \), i.e.,

\[
Z_{k+1}^{S_1} = H\hat{X}_{k+1}^O + \eta_{k+1}
\]  

(33)

where \( \eta_{k+1} = N[Rv \times (2, 2), (2m)^2, (2m)^2] \), \( Rv \) is the coefficient, while \( \sigma_{k+1}^{S_1} \) is still used to compute \( V_{k+1}^{S_1} \) in (27). Then, the model mismatch of the Kalman filter in \( S_1 \) leads to an inconsistent state estimation of the target.

When \( Rv = 1 \), the simulation results are shown in Figs. 3, 4, and TABLE 3. We can see that the CI algorithm gets disappointing results in both RMSE and NEES.

FIGURE 1. RMSE of target position estimated under normal sensor data.

FIGURE 2. NEES of target position estimated under normal sensor data.

FIGURE 3. RMSE of target position estimated when sensor \( S_1 \)'s data is abnormal.

FIGURE 4. NEES of target position estimated when sensor \( S_1 \)'s data is abnormal.
compared with the MCU, JCU, FCU, and FFCC algorithms. The NEES values for the CI algorithm are around 5, which indicates that the state estimation of the target is inconsistent. In MCU, JCU, and FCU algorithms, consistent state estimates are obtained. This is because in the CU algorithm, even if \( S_1 \)'s data is abnormal, the consistent estimate of the target can be achieved with the help of \( S_2 \). Besides, the MCU, JCU, and FCU algorithms have similar performance regarding RMSE and NEES. Additionally, the average computational time of the five fusion algorithms is similar to the results in TABLE 2. The FCU algorithm has the highest computational efficiency.

| TABLE 3. The comparison of five algorithms when sensor \( S_1 \)'s data is abnormal. |
|---------------------------------------------------------------|
|                  MCU  | FPCC | JCU | CI | FCU   |
| Average RMSE (m)  | 2.2388 | 2.2460 | 2.3043 | 3.2924 | 2.2434 |
| Average NEES      | 1.1281 | 0.7909 | 1.0294 | 5.1347 | 1.1001 |
| Time (s)          | 0.2085 | 0.0081 | 1.3021 | 0.4472 | 0.0040 |

Fig. 5 displays the influence of different \( R_v \) values on different algorithms when \( S_1 \)'s data is abnormal. With the increase in \( R_v \), the average NEES values are around the ideal value of 2 in the MCU, JCU and FCU algorithms. In the FFCC algorithm, as \( R_v \) increases, the average NEES value gradually exceeds the ideal value of 2. This is due to the fact that when \( R_v \) is much larger, the belief \( \delta \) will not decrease proportionally [26].

2) THE SENSOR DATA OF \( S_2 \) IS ABNORMAL

We assume that the measurements of \( S_2 \) are corrupted by a biased Gaussian noise \( \tau_{k+1} \), i.e.,

\[
Z_{S_2}^{k+1} = H\bar{X}_{k+1} + \tau_{k+1}
\]

(34)

where \( \tau_{k+1} = N [R_v * (2, 2), (3m)^2, (3m)^2] \). The \( S_1 \) is affected by the normal sensor noise which is given in (20).

Fig. 6 displays the influence of different \( R_v \) values on different algorithms when \( S_2 \)'s data is abnormal. It is noteworthy that the average RMSE and NEES values in the CI algorithm do not change with an increase in \( R_v \). This is because when the determinant or trace of the covariance matrix is smaller, the weight \( \omega \) in the CI algorithm is larger [28]. In this simulation, no matter what \( R_v \) is, the weight \( \omega \) in the CI algorithm
is always $\omega = 0.9999$, which means that $S_1$ will play a dominating role in the fusion.

3) \textit{S}_1 \text{ AND S}_2 \text{ HAVE THE SAME PROCESSING SENSOR NOISE, AND THE SENSOR DATA OF S}_2 \text{ IS ABNORMAL} \\
In this simulation, we study the two sensors have the same processing noise, i.e., $\sigma^1_{k+1} = \sigma^2_{k+1}$. The sensor noise of $S_1$ is given in TABLE 1 and $S_2$ is affected by an abnormal sensor data as follows:

$$Z^{S_2}_{k+1} = HX^O_{k+1} + \eta_{k+1} \quad (35)$$

The results are shown in Fig. 7 and are similar to those in Fig. 5. The CI algorithm gets disappointing results in RMSE and NEES compared with the MCU, JCU, FCU, and FFCC algorithms. Besides, the MCU and FCU get the same estimated results. This is because when $\sigma^1_{k+1} = \sigma^2_{k+1}$, we can get $P^{S_1,O-}_{k+1} = P^{S_2,O-}_{k+1}$, then whether using the minimization trace optimization or minimization determinant optimization in weight $\omega$ calculation, the weight $\omega$ is always 0.5.

V. CONCLUSION
This paper has considered the estimation fusion problem under spurious sensor data. We have presented an FCU algorithm to solve this problem. The FCU algorithm ensures the fused estimate be consistent as long as one of the estimates is consistent. In addition, we have proved that the fusion weight of FCU can be optimally generated by a closed-form algorithm without optimizing any nonlinear cost function. Thus, the computational complexity of the FCU algorithm is reduced significantly compared with the CU algorithm under the determinant minimization criterion. The performance of the proposed FCU algorithm has been verified by comparing it with state-of-the-art algorithms in simulation. The results show that the FCU algorithm can efficiently handle the spurious sensor data fusion and has higher computational efficiency than the existing CU algorithms.

REFERENCES
[1] R. A. Romeo and L. Zollo, “Methods and sensors for slip detection in robotics: A survey,” IEEE Access, vol. 8, pp. 73027–73050, 2020.
[2] X. Wang, S. Sun, T. Li, and Y. Liu, “Fault tolerant multi-robot cooperative localization based on covariance union,” IEEE Robot. Autom. Lett., vol. 6, no. 4, pp. 7799–7806, Oct. 2021.
[3] T. Li, V. Elvira, H. Fan, and J. M. Corchado, “Local-diffusion-based distributed SMC-PHD filtering using sensors with limited sensing range,” IEEE Sensors J., vol. 19, no. 4, pp. 1580–1589, Feb. 2018.
[4] Y. Wu, Y. Hao, H. Liu, L. Zhao, T. Jiang, D. Deng, and Z. Wei, “Performance improvement for wireless sensors networks by adopting hybrid subcarrier intensity modulation over exponentiated Weibull turbulence channels,” IEEE Access, vol. 8, pp. 118612–118622, 2020.
[5] M. E. Liggins, C.-Y. Chong, I. Kadar, M. G. Alford, V. Vannicola, and S. Thomopoulos, “Distributed fusion architectures and algorithms for target tracking,” Proc. IEEE, vol. 85, no. 1, pp. 95–107, Jan. 1997.
[6] M. A. Al-Jarrah, A. Dweik, M. Kalil, and S. S. Ikki, “Decision fusion in distributed cooperative wireless sensor networks,” IEEE Trans. Veh. Technol., vol. 68, no. 1, pp. 797–811, Jan. 2019.
[7] A. Sharma and S. Chauhan, “Sensor fusion for distributed detection of mobile intruders in surveillance wireless sensor networks,” IEEE Sensors J., vol. 20, no. 24, pp. 15224–15231, Dec. 2020.
[8] H. Li, A. Mehul, J. Le Kernec, S. Z. Gurbuz, and F. Fioranelli, “Sequential human gait classification with distributed radar sensor fusion,” IEEE Sensors J., vol. 21, no. 6, pp. 7590–7603, Mar. 2021.
[9] T. Li, H. Fan, J. Garcia, and J. M. Corchado, “Second-order statistics analysis and comparison between arithmetic and geometric average fusion: Application to multi-sensor target tracking,” Inf. Fusion, vol. 51, pp. 233–243, Nov. 2019.
[10] M. A. Bakre and S. Lee, “Distributed multisensor data fusion under unknown correlation and data inconsistency,” Sensors, vol. 17, no. 11, p. 2472, Oct. 2017.
[11] J. K. Uhlmann, “Covariance consistency methods for fault-tolerant distributed data fusion,” Inf. Fusion, vol. 4, no. 3, pp. 201–215, Sep. 2003.
[12] L. Chen, P. O. Arambel, and R. K. Mehra, “Estimation under unknown correlation: Covariance intersection revisited,” IEEE Trans. Autom. Control, vol. 47, no. 11, pp. 1879–1882, Nov. 2002.
[13] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, Estimation With Applications to Tracking and Navigation: Theory Algorithms and Software. Hoboken, NJ, USA: Wiley, 2004.
[14] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, “Observability-based rules for designing consistent EKF SLAM estimators,” Int. J. Robot. Res., vol. 29, no. 5, pp. 502–528, 2010.
[15] S. Julier and J. K. Uhlmann, “General decentralized data fusion with covariance intersection,” in Handbook Multisensor Data Fusion: Theory Practices. Boca Raton, FL, USA: CRC Press, 2009, pp. 319–344.
[16] S. J. Julier and J. K. Uhlmann, “Using covariance intersection for SLAM,” Robot. Auton. Syst., vol. 55, no. 1, pp. 3–20, Jan. 2007.
[17] J. Ajgl and O. Straka, “Covariance intersection in track-to-track fusion: Comparison of fusion configurations,” *IEEE Trans. Ind. Inform.*, vol. 14, no. 3, pp. 1127–1136, Mar. 2018.

[18] C. H. Kang, S. Y. Kim, and J. W. Song, “Data fusion with inverse covariance intersection for prior covariance estimation of the particle flow filter,” *IEEE Access*, vol. 8, pp. 221203–221213, 2020.

[19] M. Kumar, D. P. Garg, and R. A. Zachery, “A method for judicious fusion of inconsistent multiple sensor data,” *IEEE Sensors J.*, vol. 7, no. 5, pp. 723–733, May 2007.

[20] S. J. Julier, J. K. Uhlmann, and D. Nicholson, “A method for dealing with assignment ambiguity,” in *Proc. Amer. Control Conf.*, 2004, pp. 4102–4107.

[21] S. Matzka and R. Altdorfer, “A comparison of track-to-track fusion algorithms for automotive sensor fusion,” in *Multisensor Fusion and Integration for Intelligent Systems*. Berlin, Germany: Springer, 2009, pp. 69–81.

[22] S. Reece and S. Roberts, “Generalised covariance union: A unified approach to hypothesis merging in tracking,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 46, no. 1, pp. 207–221, Jan. 2010.

[23] O. Bochardt and J. Uhlmann, “On the equivalence of the general covariance union (GCU) and minimum enclosing ellipsoid (MEE) problems,” 2010, arXiv:1012.4795. [Online]. Available: http://arxiv.org/abs/1012.4795

[24] S. J. Julier and J. K. Uhlmann, “New extension of the Kalman filter to nonlinear systems,” *Proc. SPIE*, vol. 3068, pp. 182–193, Apr. 1997.

[25] T. Li, J. M. Corchado, and S. Sun, “Partial consensus and conservative fusion of Gaussian mixtures for distributed PHD fusion,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 5, pp. 2150–2163, Oct. 2019.

[26] Y. Wang and X. R. Li, “Distributed estimation fusion with unavailable cross-correlation,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 48, no. 1, pp. 259–278, Jan. 2012.

[27] X. R. Li, Z. Zhao, and V. P. Jilkov, “Practical measures and test for credibility of an estimator,” in *Proc. Workshop Estimation, Tracking, Fusion: A Tribute to Yaakov Bar-Shalom*, 2001, pp. 481–495.

[28] W. Niehsen, “Information fusion based on fast covariance intersection filtering,” in *Proc. Int. Conf. Inf. Fusion*, vol. 2, 2002, pp. 901–904.

**XUEDONG WANG** received the M.S. degree in mechanical design and theory from the Lanzhou University of Technology, in 2016. He is currently pursuing the Ph.D. degree in mechanical engineering from Northwestern Polytechnical University, China. His research interests include multi-sensor fusion and multi-robot cooperative localization.

**SHUDONG SUN** received the Ph.D. degree in mechanical engineering from the Nanjing University of Aeronautics and Astronautics, China, in 1989. He has been a Professor with the School of Mechanical Engineering, Northwestern Polytechnical University (NWPU), China, since 1993. He is the Director of the Key Laboratory of Industrial Engineering and Intelligent Manufacturing. His research interests include multi-robot systems, system modeling and optimization, and industrial engineering.

**YAQIONG LIU** received the M.S. degree in industrial engineering from the South China University of Technology, in 2016. She is currently pursuing the Ph.D. degree in mechanical engineering from Northwestern Polytechnical University, China. Her research interests include multi-agent scheduling optimization and industrial engineering.

**ZHAN LIU** received the B.S. degree in mechatronics engineering from Chang’an University, in 2019. He is currently pursuing the M.S. degree with the School of Mechanical Engineering, Northwestern Polytechnical University, China. His research interest includes multi-robot slam.