Fault-tolerant Cooperative Target Tracking in Distributed UAV Networks

Yoonsoo Kim * **Da-Wei Gu** ** **Ian Postlethwaite** **

* *Department of Mechanical and Mechatronic Engineering, University of Stellenbosch, Private Bag X1, Matieland 7602, South Africa (Tel: +27-21-808-4265, e-mail: ykim@sun.ac.za)*
** *Control & Instrumentation Group, Department of Engineering, University of Leicester, Leicester, LE1 7RH, UK*

Abstract: In this paper, we propose a target tracking scheme for operation in distributed UAV networks in which sensors may read the target position incorrectly. The proposed scheme operates two algorithms concurrently: *semi-decentralized dynamic data fusion* and *fault detection*. The semi-decentralized dynamic data fusion algorithm employs a median-consensus algorithm using *extended* non-faulty neighbours (whose sensor readings for the target position are within a prescribed tolerance level) and subsequently makes local estimates of the target position converge to nearly the actual target position. Meanwhile, the fault detection algorithm first asks each UAV to find the global median of the local target position through extended neighbours, and then diffuses the determined global median to all the UAVs in the network. As a result, the fault detection algorithm allows UAVs to detect and isolate faulty sensors quickly and to carry on target tracking in the semi-decentralized dynamic data fusion mode. As opposed to existing target tracking schemes, the proposed scheme is deterministic and guarantees the complete detection and isolation of faulty sensors on UAVs and thus successful target tracking. Numerical examples are provided to support the developed ideas.

Keywords: Fault-tolerant control; UAV; Multi-agent network; Median-consensus; Extended neighbours

1. INTRODUCTION

In order to meet the increasing demands of autonomy, survivability and cost-effectiveness for civil and military UAV missions, multi-UAV systems have been identified as a promising solution to pursue. For example, suppose that one UAV tracks a moving target, i.e. senses and estimates the position of the target. If the position measurement sensor has *nice* Gaussian noise throughout the mission, this tracking can be easily done with the classical Kalman filter. However, it is clear that there is little hope of finishing the mission successfully, if the *single* sensor does not work as desired. Besides, Gaussian sensor noise models may not be valid in general. In other words, sensor data can be easily corrupted by many unforeseen non-Gaussian factors, e.g. weather, physical obstacles. For these reasons, it is definitely more promising to employ several UAVs, instead of a single UAV, to perform the same tracking task. In this framework, UAVs basically track the same target, and share and update their information for more successful and accurate tracking. Needless to say, the sharing rule then must be judiciously designed such that the sensor failure issues can be fully incorporated, and this is the main focus of this paper.

Multi-UAV target tracking is the same as multi-sensor target tracking in spirit which already has been a popular topic in the literature, e.g. [9]. Among many target tracking schemes, we are particularly interested in *decentralized* target tracking combined with *fault detection* schemes due to their flexibility, cost-effectiveness and robustness. Regarding decentralized target tracking schemes, one can easily notice that the schemes employ so-called *consensus-type* algorithms. These algorithms basically allow each *agent* (sensor, UAV, etc) to gather information from its immediate neighbours and to update its information, so that every agent agrees on the common or very similar information at the end. Depending on the final common information f, these algorithms are called average-consensus, median-consensus, or f-consensus algorithms. Several recent works propose such decentralized consensus algorithms in Gaussian noisy environments, in which they use Kalman-filter type estimation rules, e.g. [7], [11], or low-pass or high-pass filters, e.g. [6], [10].

A large number of fault detection schemes have been proposed in the literature. Most recent applications of these schemes have been for wireless sensor networks, e.g. [1], [2], [3], [5], [8], where maximum likelihood, Bayesian, or voting approaches are taken to detect and isolate faulty sensors. In particular, we note one of the most recent works, [1], in this direction. In [1], the authors claim that under a mild assumption the proposed decentralized scheme is capable of almost detecting faulty sensors, even if half of the neighbouring sensors are faulty. The scheme in [1] uses two measures, $d_{ij}(t)$ and $\Delta d_{ij}^{\Delta t}$, where $d_{ij}(t)$ is the difference between the values of two sensors i and j at time t and $\Delta d_{ij}^{\Delta t}$ is the change of $d_{ij}(t)$ over a fixed time period Δt. If more than half of the

sensors $j \in \mathcal{N}_i$ are such that both measures are less than predefined threshold values, then the sensor i is diagnosed as good and is subsequently used for diagnosing other sensors as good or faulty. Although this scheme is claimed to be probabilistically attractive, we note that the two measures are not sufficient for detecting a group of faulty sensors all together in a faulty zone. For instance, one can easily consider a situation in which a faulty sensor i has its neighbouring sensors j all faulty, and therefore $d_{ij}(t) = \Delta d_{ij}^{\Delta t} = 0$ for a certain period of time and diagnosing the faulty sensor as good.

As a consequence, our current work aims to develop a scheme which is decentralized but guarantees perfect fault detection and isolation if there exist any faulty sensors, so that the central objective of target tracking is successfully achieved. In §2, we clearly state the problem under consideration and in §3 we present two algorithms which combine the ideas from previous decentralized target tracking and fault detection schemes. Numerical examples and concluding remarks are given in §4 and §5, respectively.

2. PROBLEM STATEMENT

Consider a formation of n UAVs which track the position $\mathbf{p}(t)$ of a moving target in an uncertain area $\mathbf{X}$. We assume that $\mathbf{p}(t)$ is smooth, i.e. differentiable, with respect to time t. Each UAV's sensor senses the target position $\mathbf{p}(t)$, and its corresponding reading is denoted by $\mathbf{p}_i(t)$ $(i = 1, 2, \ldots, n)$. When the position sensor is not faulty, its reading follows a Gaussian distribution with a standard deviation of σ with respect to the nominal value $\mathbf{p}(t)$. We assume that sensor readings are uncorrelated with each other over time. The ith UAV shares $\mathbf{p}_i(t)$ with other UAVs within the sensor range (the sphere of radius ρ_i centred at the ith UAV) in order to keep a fixed formation $\mathcal{F}$ even in the presence of faults in its sensor. The operational area $\mathbf{X}$ may contain a faulty (such as bad-weather) zone where some UAVs (sensors) may not correctly measure the target position. Assuming that the communication network of UAVs is connected in a standard graph theoretical sense and there are less than q $(< n/2)$ UAVs incorrectly sensing the target position all the time, our objective is to design a fault-tolerant target tracking scheme such that all the UAVs always keep the initial formation $\mathcal{F}$ within a prescribed fixed tolerance throughout the mission. [1] We assume that once the *processed* (estimated) target location $\mathbf{x}_i(t + \Delta t)$ is known to the ith UAV, the ith UAV is capable of changing its current location $\mathbf{y}_i(t)$ to $\mathbf{y}_i(t+\Delta t)$ with little effort such that $\mathbf{y}_i(t+\Delta t) = \mathbf{x}_i(t+\Delta t)+\mathbf{y}_i(0)-\mathbf{x}_i(0)$, i.e. it keeps the initial relative position with respect to the target location.

Fig. 1 illustrates an example scenario. In this scenario, there are eight UAVs in a ring formation which surrounds a moving target at $\mathbf{p}(t)$. The sensor reading for $\mathbf{p}(t)$ is $\mathbf{p}_i(t)$ for the ith UAV, U_i $(i = 1, 2, \ldots, 8)$. Since U_i can communicate with the UAVs within the sphere of radius ρ_i centred at U_i, each UAV communicates with only two adjacent UAVs in this example. Note that U_1 and U_2 are in

[1] The algorithms to be proposed can allow formation changes during the mission as long as the formations are *connected*. However, for the sake of simplicity, here we assume the initial formation (thus the initial communication topology) is fixed throughout the mission.

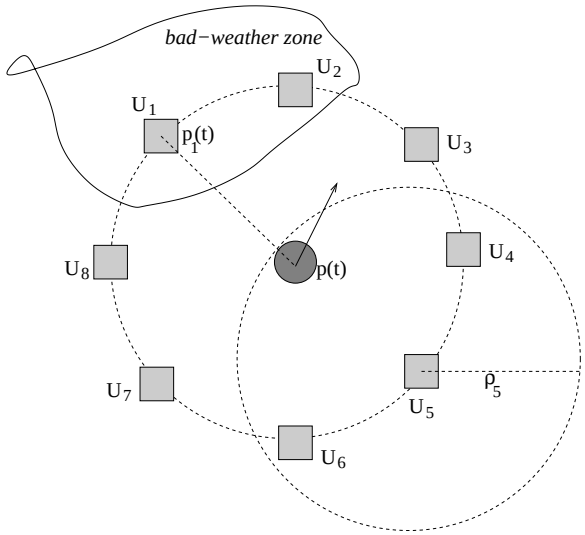

Fig. 1. A circular formation surrounding a moving object

a bad-weather zone, and thus their sensor measurements are probably incorrect.

3. ALGORITHMS

Our fault-tolerant target tracking scheme entails two algorithms: one for semi-decentralized dynamic data fusion and the other for fault detection.

3.1 Semi-decentralized dynamic data fusion

Our semi-decentralized dynamic data fusion algorithm adopts the following updating rule: for each i

$$x_i(k) = a_{ii}(k-1)x_i(k-1) + \sum_{j \in \mathcal{N}_i^{\underline{\kappa}}} b_{ij}(k-\underline{\kappa})p_j(k-\underline{\kappa}), \qquad (1)$$

where $x_i(k)$ is the ith UAV's estimated target position at the kth step $(k = 1, 2, \ldots)$, [2] p_i $(= p + \tilde{p}_i)$ is the ith UAV's position sensor reading, p is the true target position, $\tilde{p}_i$ $(\in [-\sigma, \sigma])$ is a measurement error, $\mathcal{N}_i^{\kappa}$ is the set of the ith UAV itself and the ith UAV's κ-neighbours which can be reached from the ith UAV via κ links, $\underline{\kappa}$ is the smallest integer such that $\mathcal{N}_i^{\underline{\kappa}}$ has at least one non-faulty neighbour of the ith UAV, and finally

$$a_{ii}(k) = 1 - \gamma \quad \text{and}$$
$$b_{ij}(k-\underline{\kappa}) = \frac{\gamma e^{-\alpha|\bar{p}_i(k-\underline{\kappa}) - p_j(k-\underline{\kappa})|}}{\sum_{j \in \mathcal{N}_i^{\underline{\kappa}}} e^{-\alpha|\bar{p}_i(k-\underline{\kappa}) - p_j(k-\underline{\kappa})|}}, \qquad (2)$$

where $\alpha > 0$ and $0 < \gamma < 1$ are constant parameters and $\bar{p}_i$ is the median of the positions of the ith UAV's neighbours and itself. When sensors are not yet diagnosed as faulty or non-faulty, assume that all sensors are non-faulty and thus set $\underline{\kappa} = 1$. For the sake of simplicity, we also assume that the communication delay between the ith UAV and the jth UAV is exactly the sampling time associated with (1) multiplied by the number of communication links between

[2] The rule (1) is for tracking only one of the coordinates of the target position, which is normally two-dimensional. For both coordinates, we need two equations like (1).

the ith UAV and the jth UAV. If the communication delay is larger than the sampling time, each UAV estimates the target location based on its own information (unless its own sensor is faulty) before new information from other UAVs arrives.

The rule (1) may be written as in the following compact form:

$$\begin{aligned}\mathbf{x}(k) &= \mathbf{A}(k-1)\mathbf{x}(k-1)\\ &\quad +\mathbf{B}(k-\underline{\kappa})(p(k-\underline{\kappa})\mathbf{1}+\tilde{\mathbf{p}}(k-\underline{\kappa})),\end{aligned} \tag{3}$$

where $\mathbf{x} = [x_1\ x_2\ \dots\ x_n]^T$, $\tilde{\mathbf{p}} = [\tilde{p}_1\ \tilde{p}_2\ \dots\ \tilde{p}_n]^T$, $\mathbf{A} = [a_{ij}] = (1-\gamma)\mathbf{I}$, $\mathbf{B} = [b_{ij}]$ as in (2) and $\mathbf{1}$ is an all-one vector. Rule (3) is a similar version to those found in, for example, [6] and [11], but with different $\mathbf{A}$ and $\mathbf{B}$. The estimation error $\mathbf{e}(k) = \mathbf{x}(k) - p(k)\mathbf{1}$ is then

$$\begin{aligned}\mathbf{e}(k) &= \mathbf{A}(k-1)\mathbf{e}(k-1) + \mathbf{A}(k-1)p(k-1)\mathbf{1}\\ &\quad +\mathbf{B}(k-\underline{\kappa})(p(k-\underline{\kappa})\mathbf{1}+\tilde{\mathbf{p}}(k-\underline{\kappa})) - p(k)\mathbf{1}.\end{aligned}$$

Taking expected values,

$$\begin{aligned}\mathsf{E}\mathbf{e}(k) &= \mathbf{A}(k-1)\mathsf{E}\mathbf{e}(k-1) + \mathbf{A}(k-1)p(k-1)\mathbf{1}\\ &\quad +\mathbf{B}(k-\underline{\kappa})p(k-\underline{\kappa})\mathbf{1} - p(k)\mathbf{1}.\end{aligned} \tag{4}$$

Therefore, perfect tracking can be achieved if at each time t all the eigenvalues of $\mathbf{A}(t)$ are strictly less than 1 in magnitude and

$$(\mathbf{A}(k-1)p(k-1) + \mathbf{B}(k-\underline{\kappa})p(k-\underline{\kappa}))\mathbf{1} = p(k)\mathbf{1}. \tag{5}$$

However, $p(k)$ is often not known to UAVs at the $(k-1)$th step. Therefore, instead of achieving perfect tracking, we aim to have $\mathsf{E}\mathbf{e}(t)$ asymptotically bounded if $|p(k)-p(k-1)|$ is uniformly bounded for any k.

In the literature, there are several ways of choosing $\mathbf{A}$ and $\mathbf{B}$ while satisfying conditions similar to (5). For example, [6] chooses them so that (3) acts as a low-pass filter, and [11] chooses such that the trace of the corresponding error covariance matrix is minimized at each time step. In this paper, we set $\mathbf{A}$ and $\mathbf{B}$ such that (3) becomes fault-tolerant. However, before we discuss the selection of γ and the fault-tolerant property of (3), we first show that (3), with our choice of $\mathbf{A}$ and $\mathbf{B}$, acts as a low-pass filter, asymptotically converges to the target position for the bounded target speed and guarantees the bounded trace of the corresponding error covariance matrix. The following analysis is motivated by the results in [6] and [11].

Proposition 3.1. The rule (3) is a low-pass filter

Proof: Since $\tilde{\mathbf{p}}$ is Gaussian noise, any linear combination of the elements of $\tilde{\mathbf{p}}$ is Gaussian noise. Therefore, $\mathbf{B}\tilde{\mathbf{p}}$ is Gaussian noise. Since all the eigenvalues of $\mathbf{A}$ are strictly less than 1 in magnitude, (3) must be a low-pass filter for Gaussian noise $\tilde{\mathbf{p}}$. Note that $\mathbf{B}\mathbf{1} = \gamma\mathbf{1}$. ■

Proposition 3.2. The expected value of x_i in (1) asymptotically converges to the set

$$\{x_i \,|\, |x_i - p| \le \bar{\beta}/\gamma\},$$

where $|p(k)-p(k-1)| \le \beta$ and $\bar{\beta} = (1-\gamma+\gamma\underline{\kappa})\beta$.

Proof: First note that if we let $\bar{p} = (1-\gamma)p(k-1)+\gamma p(k-\underline{\kappa}) - p(k)$, then $|\bar{p}| \le \bar{\beta}$. By setting $e_i(k) = x_i(k) - p(k)$ and $V(k) = \mathsf{E}e_i(k)^T\mathsf{E}e_i(k)$, from (4)

$$\begin{aligned}V(k) - V(k-1) &= -(1-(1-\gamma)^2)\{\mathsf{E}e_i(k-1)\}^2\\ &\quad +(1-\gamma)\bar{p}\mathsf{E}e_i(k-1) + |\bar{p}|^2\\ &\le -(1-(1-\gamma)^2)\{\mathsf{E}e_i(k-1)\}^2\\ &\quad +(1-\gamma)\bar{\beta}\mathsf{E}e_i(k-1) + \bar{\beta}^2.\end{aligned}$$

In view of the inequality $\sqrt{-3\gamma^2+6\gamma+1} \le 3-\gamma$ for $0<\gamma<1$, $V(k)-V(k-1)<0$ if

$$\mathsf{E}e(k-1) > \bar{\beta}/\gamma.$$

This proves the claim. ■

Proposition 3.3. The trace of the covariance matrix, $\mathtt{tr}\mathbf{P}(k)$, associated with $\mathbf{e}(k)$ is uniformly bounded. Furthermore,

$$\mathtt{tr}\mathbf{P}(k) \le \left(\frac{\gamma}{2-\gamma}\right)n\sigma^2$$

as $k \to \infty$.

Proof: Defining the error covariance matrix $\mathbf{P}(k)$ as $\mathsf{E}(\mathbf{e}(k) - \mathsf{E}\mathbf{e}(k))(\mathbf{e}(k) - \mathsf{E}\mathbf{e}(k))^T$, we have

$$\mathbf{P}(k) = (1-\gamma)^2\mathbf{P}(k-1) + \sigma^2\mathbf{B}(k-\underline{\kappa})\mathbf{B}(k-\underline{\kappa})^T$$

Taking the trace operator $\mathtt{tr}$ on both sides,

$$\begin{aligned}\mathtt{tr}\mathbf{P}(k) &= (1-\gamma)^2\mathtt{tr}\mathbf{P}(k-1) + \sigma^2\sum_{i,j} b_{ij}(k-\underline{\kappa})^2\\ &= (1-\gamma)^{2k}\mathtt{tr}\mathbf{P}(0)\\ &\qquad +\sigma^2\sum_{ij} b_{i,j}(k-\underline{\kappa})^2\sum_{l=1}^{k}(1-\gamma)^{2(l-1)}\\ &\le (1-\gamma)^{2k}\mathtt{tr}\mathbf{P}(0) + \left(\frac{\gamma^2 n}{1-(1-\gamma)^2}\right)\sigma^2\\ &= (1-\gamma)^{2k}\mathtt{tr}\mathbf{P}(0) + \left(\frac{\gamma}{2-\gamma}\right)n\sigma^2.\end{aligned}$$

The second last inequality is due to the fact that

$$\sum_{ij} b_{ij}(k-\underline{\kappa})^2 \le \gamma^2 n.$$

■

The aforementioned three results clearly imply that the constant parameter γ decides the bandwidth of the low-pass filter, the error covariance and the effect of communication delays (a smaller γ reduces sensitivity to noise and communication delays) as well as the speed and accuracy of $\mathbf{x} \to p\mathbf{1}$ (a larger γ yields faster and more accurate convergence). Thus, one needs to make a judicious choice of γ depending on the application at hand. In our numerical example, for $n = 10$, $\underline{\kappa} = 1$ and $\Delta t = 0.1$ we set $\beta = 1$ ($\approx 10\ m/s$), $\gamma = 0.32$ and thus $\mathtt{tr}\mathbf{P}(k) \le 0.19\sigma^2$ as $k \to \infty$.

Let us now discuss the fault-tolerant property of (3). As noticed in matrix $\mathbf{B}$, (3) calculates the *median* of collected values and then linearly combines the values with different weights inversely proportional to the distances

to the median. The parameter α refines the relationship between the weight and the distance to the median. In our numerical examples, we set $\alpha = 1$. Clearly, this data fusion scheme is fault-tolerant, provided that the number of faulty neighbours (including itself) is less than $|\mathcal{N}_i|/2$ for every i throughout the mission. However, as mentioned before, there could be disturbances occurring in a specific area and subsequently a group of sensors may not provide correct information. In the next section, a fault detection algorithm, which is concurrently run with the present data fusion algorithm, is presented in order to cover these regional-type faults.

3.2 Fault detection and isolation

While the data fusion algorithm operates over the UAV network, we apply another algorithm for detecting any faulty sensors. Our fault detection algorithm entails two steps: (i) finding the *global* median of estimated target positions p_i $(i = 1, 2, \ldots, n)$ within a fixed tolerance; (ii) propagating the determined global median to every UAV for diagnosing faulty or non-faulty sensors.

The first step of the fault detection algorithm makes use of κ-neighbours, as introduced before. Basically, each UAV gathers enough information from (up to $2q$) other UAVs over a time period until it can determine an *almost* global median of p_i $(i = 1, 2, \ldots, n)$.[3] By *almost*, we mean that the distance to the true global median is at most σ. Since there are at most q faulty sensors, the almost global median can be determined if the gathered information contains at least $q+1$ *similar* sensor readings, i.e. $|p_i - p_j| \leq 2\sigma$. For example, consider the example depicted in Fig. 1. Suppose the sensor readings of U_1 and U_2 are corrupted. Then, U_1 and U_2 need to gather more information not only from U_8 and U_3 but from U_7 and U_4, because they need at least three values similar to each other to identify an almost global median. Since there is no direct communication link between U_1 and U_7, it therefore requires two time units for U_1 or U_2 to finish the first step of the fault detection algorithm. In contrast, U_4, U_5, U_6 or U_7 needs only one time unit to determine an almost global median, as the information from its immediate neighbours provides three values similar to each other.

The second step of the fault detection algorithm is the propagation of the determined almost global median to every UAV for detecting and isolating faulty sensors. Once a UAV notices that one of its immediate neighbours knows an almost global median, it compares the almost global median with its target estimation at the time when the fault detection algorithm was initially applied. If the difference is beyond 2σ, the UAV's sensor is diagnosed as faulty. Since the previously proposed data fusion algorithm is now equipped with the fault detection algorithm, $\underline{\kappa}$ and $\mathcal{N}_i^{\underline{\kappa}}$ can be adjusted so that faulty readings cannot enter the data fusion equation (3). Table 1 shows the formal description of the two proposed algorithms which run on the ith UAV. In the table, $|\mathcal{S}|_s$ is used for denoting the number of sensors ($\in \mathcal{S}$) having similar readings.

The idea of using κ-neighbours trivially guarantees that all faulty sensors are found in finite steps as long as the communication network of UAVs is connected. More interestingly, depending on the structure of the communication network of UAVs, the time to detect all faulty sensors may vary. Suppose that the ith UAV takes $\bar{\kappa}$ steps such that $|\mathcal{N}_i^{\bar{\kappa}}|_s \geq q+1$ and subsequently computes the corresponding almost global median. Then, it is clear that it takes at most another $\bar{\kappa}$ steps to propagate the determined almost global median to the sensors (UAVs) belonging to $\mathcal{N}_i^{\bar{\kappa}}$. Since this fact is true for every $i \in \{1, 2, \ldots, n\}$, we have the following result.

Proposition 3.4. If $|\mathcal{N}_i^{\bar{\kappa}}|_s \geq q+1$ for every $i \in \{1, 2, \ldots, n\}$, then all faulty sensors can be found in at most $2\bar{\kappa}$ steps.

Proposition 3.4 implies several facts on the proposed algorithm: (i) the completion time of the proposed fault detection algorithm purely depends on the graph-theoretical property $\bar{\kappa}$; (ii) every UAV can assume that if the fault detection algorithm starts at time t, all faults are completely detected at $t+\bar{t}$, where $\bar{t}$ is equivalent to $2\bar{\kappa}$ steps, at which time the data fusion algorithm starts using newly updated $\mathcal{N}_i^{\underline{\kappa}}$ and the fault detection algorithm restarts; (iii) the fault detection algorithm can operate independently of the data fusion algorithm during $(t, t+\bar{t}]$); (iv) the fault tolerant algorithm is valid under the assumption that for any time period of $\bar{t}$, UAVs with faulty sensors do not deviate too much from other UAVs, so that they can return to the formation after the sensors are correctly diagnosed.

In operations research, there are many problems which involve variants of the property $\bar{\kappa}$, e.g. [12]. The most famous one is the (k, q, r)-centre problem in which for a given graph consisting of vertices and edges, find k groups of vertices in each of which there are at least q vertices and all vertices can be reached from one centre vertex via at

Table 1. The formal description of the proposed algorithms

Part I: Semi-decentralized dynamic data fusion

Input: Δt, $x_i(0)$, $y_i(0)$ and n

Determine $\underline{\kappa}$, $\mathcal{N}_i^{\underline{\kappa}}$ and $b_{ij}(t-(\underline{\kappa}-1)\Delta t)$ for $j \in \mathcal{N}_i^{\underline{\kappa}}$

$x_i(t+\Delta t) \leftarrow (1-\gamma)x_i(t)$
$\quad + \sum_{j \in \mathcal{N}_i^{\underline{\kappa}}} b_{ij}(t-(\underline{\kappa}-1)\Delta t)p_j(t-(\underline{\kappa}-1)\Delta t)$
$y_i(t+\Delta t) \leftarrow x_i(t+\Delta t) + y_i(0) - x_i(0)$

Part II: Fault detection

Input: t, σ, q, n, $\mathcal{N}_i^1$ and $p_j(t)$ where $j \in \mathcal{N}_i^1$

$\mathcal{E}N_i \leftarrow i$ and $\mathcal{V}_i \leftarrow p_i(t)$
$\mathbf{GloMed}_i = \mathbf{NotFound}$

while $|\mathcal{E}N_i|_s < q+1$ and $\mathbf{GloMed}_j = \mathbf{NotFound}$ $\forall j \in \mathcal{E}N_i$
  $\mathcal{E}N_i \leftarrow \mathcal{E}N_i \cup \sum_{j \in \mathcal{N}_i^1} \mathcal{E}N_j$
  $\mathcal{V}_i \leftarrow \{x_j(t) \,|\, j \in \mathcal{E}N_i\}$
  $\mathbf{GloMed}_i \leftarrow \mathbf{GloMed}_j$
    if $\mathbf{GloMed}_j \neq \mathbf{NotFound}$ for some $j \in \mathcal{E}N_i$

if $|\mathcal{E}N_i|_s \geq q+1$
  $\mathbf{GloMed}_i \leftarrow \mathbf{median}(\mathcal{V}_i)$

if $|\mathbf{GloMed}_i - p_i(t)| > 2\sigma$
  the ith UAV's sensor is faulty
else
  the ith UAV's sensor is non-faulty

[3] We assume that each UAV knows n and estimates q before the mission starts.

most r edges. When q and k (or r) are fixed and r (or k) is minimized over all feasible grouping solutions, the problem becomes extremely hard. Fortunately, the problem we are interested in has both k and r free, and thus finding a minimum r $(= \bar{\kappa})$ such that each group of vertices contains at least $q+1$ *similar* (non-faulty) vertices can be easily done by the following algorithm: given the definitions $\mathcal{I}^\kappa := \{i \,|\, |\mathcal{N}_i^\kappa|_s \geq q+1\}$ and $\mathcal{J}^\kappa := \{j \,|\, |\mathcal{N}_j^\kappa|_s < q+1\}$ and $\kappa = 1$, (i) check if $i \in \mathcal{I}^\kappa$ for every $i \in \{1, 2, \ldots, n\}$; (ii) if so, terminate the algorithm and return κ; (iii) otherwise, if for all $j \in \mathcal{J}^\kappa$ $j \in \mathcal{N}_i^\kappa$ for some $i \in \mathcal{I}^\kappa$, then terminate the algorithm and return κ; (iv) go to (i) with $\kappa \leftarrow \kappa + 1$.

In the next section, it will be shown numerically that $\bar{\kappa}$ is reasonably small for randomly-generated connected graphs. This implies that the proposed algorithm detects all faulty sensors within only a few steps on average. We also note that the $\mathcal{N}_i^\kappa$ for $i \in \{1, 2, \ldots, n\}$ are generally not disjoint, and thus one can expect the actual all-fault detection steps to be less than 2κ. This point will also be verified shortly.

4. NUMERICAL TESTS

4.1 $\bar{\kappa}$ of random graphs

We first observe how $\bar{\kappa}$ varies for various graphs, so that one can have an idea of how large $\bar{\kappa}$ can be on average. We consider a 100-by-100 (*meter*) space $\mathbf{X}$ in which 10 UAV ($n = 10$) positions including 3 faulty sensors ($q = 3$) are randomly generated. As defined before, a communication link is added if two UAVs are within a distance of ρ, i.e. $\rho = \rho_i$ $(i = 1, 2, \ldots, n)$. In order to generate different kinds of graphs in terms of *connectedness*, we consider 1000 connected random networks for different ρ. A random generation of UAV positions may not produce a connected network. Thus, we repeat m random generations until 1000 connected random networks are obtained. For each connected random network, we calculate $\bar{\kappa}$ and the actual steps θ needed to finish the fault detection algorithm. We also set $\eta = 1000/m$, so that η roughly represents the probability that the generated random network whose communication links are associated with ρ is connected.

Table 2 summarizes the numerical test results. As shown in the table, $\bar{\kappa}$ is less than 2 on average even for very loosely connected networks with small η. As addressed in the last section, θ is smaller than $2\bar{\kappa}$ and verifies that all faulty sensors in a 10-UAV network can be detected in less than about 3 steps on average. A similar observation can be made for larger $\mathbf{X}$ and n.[4] Figure 2 shows one of 1000 connected random networks with $\mathbf{X} = [0, 1] \times [0, 1]$ (*kilometer*), $n = 100$, $q = 30$, $\rho = 140$ (*meter*) and $\eta = 0.1089$, where circles represent faulty sensor locations. Note that the faulty sensor $\mathbf{s}$ in the figure has 4 faulty but only 3 non-faulty neighbours. The numerical tests show that $\bar{\kappa}$ and θ are still relatively small numbers, 5.78 and 8.85, respectively, on average even for such loosely connected large networks. These tests clearly testify that the proposed fault detection algorithm can be used for large networks, e.g, wireless sensor networks.

[4] Although our main UAV applications in this paper may involve a relatively small number of UAVs, we also consider large sensor networks to see how $\bar{\kappa}$ varies as the connectedness of large networks does.

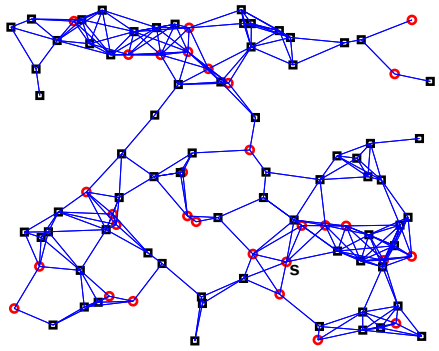

Fig. 2. A sensor network with $n = 100$ and $q = 30$: faulty and non-faulty sensors are denoted by circles and squares, respectively

Table 2. Numerical test results for $n = 10$ and $q = 3$

ρ (m)	η	$\bar{\kappa}$	θ
33	0.0998	1.95	3.17
36	0.2085	1.87	3.02
38	0.3054	1.82	2.93
40	0.3897	1.76	2.85
43	0.5376	1.65	2.70
45	0.6177	1.57	2.61
47	0.7097	1.50	2.51
49	0.7837	1.41	2.39
51	0.8382	1.36	2.32
55	0.8953	1.23	2.16

4.2 Target tracking with circular formation keeping

We consider an example scenario in which 10 UAVs flying in a circular formation track a target on the ground in the middle of them, as shown in Fig. 3. The target is moving at a velocity of $\beta/\Delta t$ $(meter/sec)$, where $\beta = 1$ and the sampling time $\Delta t = 0.1$, with a direction of $[t,\ t^2 \cos p_x]$, where p_x is the x-coordinate of the target position. For every i $(\in \{1, 2, \ldots, 10\})$, the ith UAV is modelled with first-order dynamics and with the same velocity as the target, and can communicate only with its adjacent, $\mathtt{mod}_{10}(i-1)$th and $\mathtt{mod}_{10}(i+1)$th, UAVs using a wireless device, where $\mathtt{mod}_{10}(i) = i$ if $1 \leq i \leq 10$, 1 if $i = 0$ or 11. We set $q = 3$, $\sigma = 0.5$ $(meter)$, $\alpha = 1$ and $\gamma = 0.32$. Based on the imposed communication topology, one can easily figure out that $2\bar{\kappa} = 6$ in this example. This means that the fault detection algorithm is initiated every $2\bar{\kappa}\Delta t$ $(= 0.6)$ seconds. We then assume that the target tracking sensors of the 1st, 2nd and 10th UAVs have incorrect readings during the time interval of $[10, 100]$ (sec). Since the 1st UAV can communicate only with the 2nd and 10th UAVs, the rule (3) with $\underline{\kappa} = 2$ is used for the 1st UAV during this interval.

Figs. 4, 5 and 6 show the simulation results. Fig. 4 is the sensor reading of the 1st UAV for the target location and Fig. 5 is the 1st UAV's estimation of the target location using the proposed algorithms. As expected, the rule (3) acts as a low-pass filter for high-frequency noise. The two

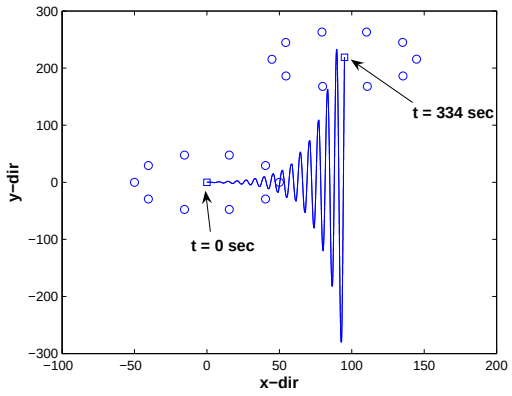

Fig. 3. A circular formation of UAVs (circles): the blue line is the trajectory of the target (square)

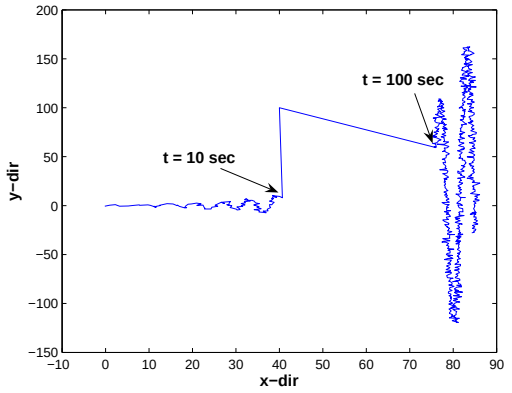

Fig. 4. Sensor readings from the first UAV for the target location

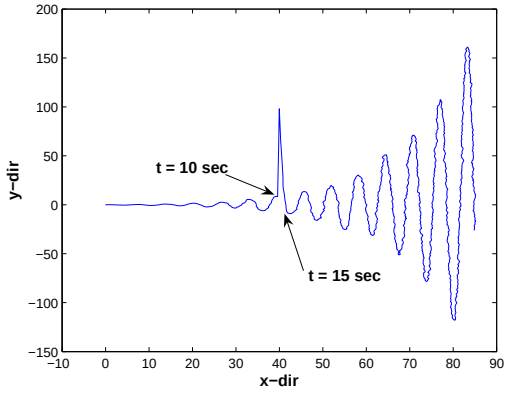

Fig. 5. Target location estimations by the first UAV

figures show that the 1st UAV manages to track the target in 5 seconds after the fault is injected at $t = 10$ (sec). This is also ascertained by Fig. 6 showing the 1st UAV's trajectory (green) versus the true target trajectory shifted by the initial relative position with respect to the first UAV (blue).

5. CONCLUDING REMARKS

We have proposed a fault-tolerant target tracking scheme for operation in distributed UAV networks in which sensors may read the target position incorrectly. The proposed scheme employs two algorithms concurrently: semi-decentralized dynamic data fusion and fault detection. As both the theoretical and numerical studies have shown, the proposed fault detection algorithm allows UAVs to detect and isolate faulty sensors quickly and to carry on target tracking in the semi-decentralized dynamic data fusion mode. In particular, we note that the convergence of the proposed scheme purely depends on the graph theoretical property $\bar{\kappa}$ and this $\bar{\kappa}$ is relatively small (compared to the graph diameter) even for large networks. For this reason, we expect the proposed scheme to be useful in many different contexts.

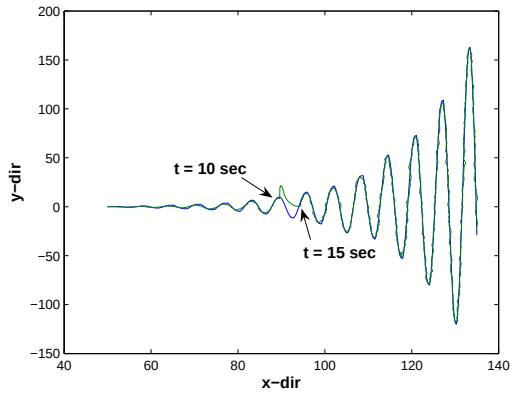

Fig. 6. The first UAV's trajectory (green) versus the true target trajectory shifted by the initial relative position with respect to the first UAV (blue)

ACKNOWLEDGMENT

This research has been supported by the UK Engineering and Physical Sciences Research Council and BAE Systems. The first author has been partially supported by the faculty settlement fund from the University of Stellenbosch in South Africa.

REFERENCES

[1] J. Chen, S. Kher and A. Somani. Distributed fault detection of wireless sensor networks. *In the Proceedings of the 2006 Workshop on Dependability Issue in Wireless Ad Hoc Networks and Sensor Networks*, Los Angeles, Sep. 2006.

[2] M. Ding, D. Chen, K. Xing and X. Cheng. Localized fault-tolerant event boundary detection in sensor networks. *In the Proceedings of IEEE INFOCOM*, Miami, USA, Mar. 2005.

[3] E. Franco, R. Olfati-Sabor, T. Parisini and M. M. Polycarpou. Distributed fault diagnosis using sensor networks and consensus-based filters. *In the Proceedings of the 45th IEEE Conference on Decision & Control*, pp 386–391, Dec. 2006.

[4] Y. Kim and M. Mesbahi. On maximizing the second smallest eigenvalue of a state-dependent Laplacian. *IEEE Transactions on Automatic Control,* 51 (1): 116–120, 2006.

[5] X. Luo, M. Dong and Y. Huang. On distributed fault-tolerant detection in wireless sensor networks. *IEEE Transactions on Computers*, 55 (1): 58–70, Jan. 2006.

[6] R. Olfati-Saber and J. S. Shamma. Consensus Filters for Sensor Networks and Distributed Sensor Fusion. *In the Proceedings of the 44th IEEE Conference on Decision & Control and European Control Conference*, pp6698-6703, Dec. 2005.

[7] W. Ren, R. W. Beard and D. B. Kingston. Multi-agent Kalman Consensus with Relative Uncertainty. *In the Proceedings of the American Control Conference*, pp 1865–1870, Jun. 2005.

[8] L. B. Ruiz, I. G. Siqueira, L. B. Oliveira, H. C. Wong, J. M. S. Nogueira and A. A. F. Loureiro. Fault management in event-driven wireless sensor networks. *In the Proceedings of MSWiM*, Venezia, Italy, Oct. 2004.

[9] D. Smith and S. Singh. Approaches to multisensor data fusion in target tracking: a survey. *IEEE Transactions on Knowledge and Data Enginnering,* 18 (12): 1696–1710, 2006.

[10] D. Spanos, R. Olfati-Saber and R. M. Murray. Dynamic consensus on mobile networks. *In the Proceedings of the 16th IFAC World Congress*, Prague, Czech, 2005.

[11] A. Speranzon, C. Fischione and K. H. Johansson. Distributed and collaborative estimation over wireless sensor networks. *In the Proceedings of the 45th IEEE Conference on Decision & Control*, pp 1025–1030, Dec. 2006.

[12] C. Toregas, R. Swain, C. Revelle and L. Bergman. The location of emergency service facilities. *Operations Research,* 19 (1): 1363–1373, 1971.