sensors

MDPI

Article

# A Novel Loss Recovery and Tracking Scheme for Maneuvering Target in Hybrid WSNs

**Hanwang Qian [1,2], Pengcheng Fu [1,2], Baoqing Li [1], Jianpo Liu [1] and Xiaobing Yuan [1,*]**

[1] Science and Technology on Microsystem Laboratory, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 201800, China; qianhw@mail.sim.ac.cn (H.Q.); fupc@mail.sim.ac.cn (P.F.); sinoiot@mail.sim.ac.cn (B.L.); liujp@mail.sim.ac.cn (J.L.)

[2] University of Chinese Academy of Sciences, Beijing 100049, China

* Correspondence: sinowsn@mail.sim.ac.cn; Tel.: +86-021-6907-5861

Received: 26 December 2017; Accepted: 23 January 2018; Published: 25 January 2018

**Abstract:** Tracking a mobile target, which aims to timely monitor the invasion of specific target, is one of the most prominent applications in wireless sensor networks (WSNs). Traditional tracking methods in WSNs only based on static sensor nodes (SNs) have several critical problems. For example, to void the loss of mobile target, many SNs must be active to track the target in all possible directions, resulting in excessive energy consumption. Additionally, when entering coverage holes in the monitoring area, the mobile target may be missing and then its state is unknown during this period. To tackle these problems, in this paper, a few mobile sensor nodes (MNs) are introduced to cooperate with SNs to form a hybrid WSN due to their stronger abilities and less constrained energy. Then, we propose a valid target tracking scheme for hybrid WSNs to dynamically schedule the MNs and SNs. Moreover, a novel loss recovery mechanism is proposed to find the lost target and recover the tracking with fewer SNs awakened. Furthermore, to improve the robustness and accuracy of the recovery mechanism, an adaptive unscented Kalman filter (AUKF) algorithm is raised to dynamically adjust the process noise covariance. Simulation results demonstrate that our tracking scheme for maneuvering target in hybrid WSNs can not only track the target effectively even if the target is lost but also maintain an excellent accuracy and robustness with fewer activated nodes.

**Keywords:** target tracking; hybrid wireless sensor networks; target recovery; data fusion; dynamic cluster scheduling

## 1. Introduction

Wireless sensor networks (WSNs) have recently emerged as an increasingly significant area of research owing to their wide range of applications, such as environmental monitoring, security surveillance, industry control, and intrusion detection [1–4]. In addition, they can be used in military applications. Examples include monitoring a battlefield, assessing battle damage, biological and chemical attack detection, and monitoring of water quality control [5]. Among these applications, tracking a moving target is a prominent application that can be realized by deploying a great deal of sensor nodes in the interested area to timely monitor the invasion of specific targets, such as enemy vehicles, enemy soldiers, and wild animals [6].

Generally, WSNs consist of many static sensor nodes (SNs) which are tiny low-cost, energy-limited, and sensing range-limited for cost saving. Hence, it is imperative to efficiently manage the sensors' resources to prolong the lifetime of tracking networks without sacrificing performance. Much research effort has been dedicated to resolve the issue from different perspectives, for example, energy-efficient tracking scheme [7–10] and energy-balanced tracking scheme [11–13]. However, as long as the sensor nodes are static, this issue cannot be fully tackled. In recent years, empowered by embedded computing and wireless communication techniques, some sensor nodes can move around when they are installed

on mobile equipments. In general, mobile sensor nodes (MNs) are resource-rich devices with more energy, higher communication power, and more powerful sensing and computational capabilities than SNs [14]. In the hybrid WSNs, SNs are responsible for sensing environment variables, while the MNs, also called mobile sinks, move to designated positions for gathering data or results sent by SNs and then forward them to the remote end. Typically, to support mobility of the MNs, a source node (e.g., a cluster head) can report the target state and other related data to the MNs, and the MNs could move itself to some position according to the target position and then broadcasts its arrival.

However, some aforementioned limitations of SNs in hybrid WSNs remain, raising the need for some specialized measures, such as dynamic network structure, position computation of target, future-position prediction of target. These measures will decrease the number of SNs participating in tracking as small as possible, which may result in the loss of mobile target [15]. Thus, in practical scenarios of target tracking in WSNs, the problem of losing target may frequently crop up. Many researchers have focused on this issue and put forward some efficient solutions. Hsu et al. [16], proposed two recovery algorithms namely CORS and TORS. The CORS searches for the lost target sequentially based on the probability of being located in certain faces. While the TROS wakes up all sensor nodes within a circular area that is centered on the position where the target is lost, and the radius of the circular area is the distance that the target may travel with its maximum speed. Patil et al. [15] proposed an energy efficient recovery mechanism which considers two types of network scenarios. The first type is wireless boundaries are known by the network (WSHAN), and the anther is wireless sensor hole unaware (WSHUN) where the hole boundary nodes are unknown. To decrease the energy consumption in tracking, Samarah et al. [17] introduced a prediction-based tracking technique using sequential patterns (PTSPs). Since PTSP approach uses a prediction technique, the tracking may experience some target missing. To overcome the problem, three recovery mechanisms have been implemented: source recovery mechanism, destination recovery mechanism, and all neighbors recovery mechanism. After comparing the experiment results, the source recovery mechanism is deemed the best one among the three mechanisms.

However, most of recovery mechanisms (including the above methods) are put forward for static sensor networks rather than hybrid sensor networks. In hybrid WSNs, there are also many reasons resulting in the loss of target, such as communication failures, node death, sudden change in target speed or direction, localization errors, and coverage holes in the deployment monitoring area [18]. In real environments, the mobility of target and the distribution of the sensor nodes are usually the two most difficult factors that users of the tracking networks could control, especially in the battlefront or hostile environment. Hence, the tracking network often misses the mobile target because the target sudden changes its speed/direction or enters coverage holes in the deployment monitoring area.

In this paper, we focus on tracking the maneuvering target in hybrid WSNs and put forward a novel loss recovery mechanism aiming at the situations that the target moves with time-varying speed and enters coverage holes in the deployment monitoring area. More specifically, considering characteristics and constraints of target tracking and recovery in hybrid WSNs, we utilise the following mechanisms to efficiently carry out tracking tasks: (1) a cluster-based structure to cooperation tracking the mobile target, which consist of a few static sensor nodes and will vary with the moving of the target; (2) a prediction-based method to dynamically select appropriate task cluster nodes according to their current energy and distance to the predicted position of target; (3) the cluster head (CH) will fuse different detection results from other cluster members with its own by using unscented Kalman filter (UKF) algorithm; (4) the MNs, which are assumed with unlimited energy, higher communication and sensing capabilities will also cooperate with the task cluster to implement the tracking under normal conditions; (5) once the target is declared lost, the MNs will continue performing the tracking and activate the related static nodes to form recovery task cluster; and (6) an adaptive unscented Kalman filter (AUKF) which adaptively adjusts the prior process noise covariance matrix is proposed for the MNs to improve the accuracy and robustness of recovery mechanism. Our main contributions are:

- Propose an effective target tracking scheme in hybrid WSNs where the MNs and the dynamic activated cluster nodes are integrated for cooperation tracking.
- Design a novel loss recovery mechanism for mobile target in hybrid WSNs, which aims to recover the mobile target with fewer active nodes in the cases that the target suddenly changes its speed or direction and the target enters coverage holes in the deployment monitoring area.
- Propose an adaptive UKF (AUKF) algorithm which adaptively adjusts the process noise covariance matrix based on the weighting combination of its current theoretical estimation value and previous data.

The organization of the paper is as follows. In Section 2, we formulate the basic problems and system models involved in target tracking in hybrid WSNs. Section 3 briefly introduces UKF algorithm and presents the proposed AUKF algorithm. The mechanism of dynamically selecting cluster members and cluster head is discussed in Section 4. Section 5 describes the tracking process in hybrid WSNs. Section 6 illustrates the proposed target recovery mechanism. Simulation experiments are reported in Section 7. Finally, Section 8 concludes the paper.

## 2. Problem Formulation and System Models

This section presents the basic problems and system models involved in target tracking. Based on the realistic models, the definition of tracking probability is introduced at the end. Table 1 has summarized some key symbols in this paper.

### *2.1. Problem Formulation and System Overview*

As shown in Figure 1, a lot of static sensor nodes are deployed randomly and unevenly in a area of interest, and some MNs, also called mobile sinks, which could move anywhere in a random way locate initially in the area boundary [19]. The network consists of $N_s$ cheap and low-power SNs $S = \{s_1, s_2, ..., s_{N_s}\}$ and a few MNs $M = \{m_1, m_2, ..., m_i\}$. Each of SNs is equipped with an ultrasonic distance sensor as well as a low-cost passive infrared (PIR) sensor and the MNs are fitted with an angular sensor besides the above two sensors. Both the sensing and communication models of nodes are the unit-disk graph model. In order that all sensor nodes that sense the same target can communicate with each other, their communication radii $R$ are set twice of their sensing radii $r$. In this paper, we assume that the sensing radius of the MNs is much greater than that of SN and the energy consumption of MNs is less constrained, as they can replenish their energy due to the mobility [20]. Meanwhile, the location of each sensor node which can be obtained by on-board GPS receiver is known by itself after the initialization of network. Without loss of generality, the target and all sensor nodes are assumed to locate in a 2-D area in this paper. Thus we formulate the target tracking problem with a 2-D model.

In this paper, all static sensor nodes work in two modes: sleep (inactive) and wake up (active). When nodes are in the sleep mode, they stay in the sleep state and wake up for a relatively short time periodically, during which time they can detect whether the target appears in their sensing area [21]. When a maneuvering target moves along a curve path in the surveillance area, only some of sensor nodes along this path will be woken up to form a task cluster which includes the cluster head (CH) and the cluster member (CM). They measure the distances between target and themselves, and report the measurements to the CH which serves as the local fusion center. Meanwhile, the MN closest to the target will follow closely behind the target. It acquires the target position via inquiring the current CH.

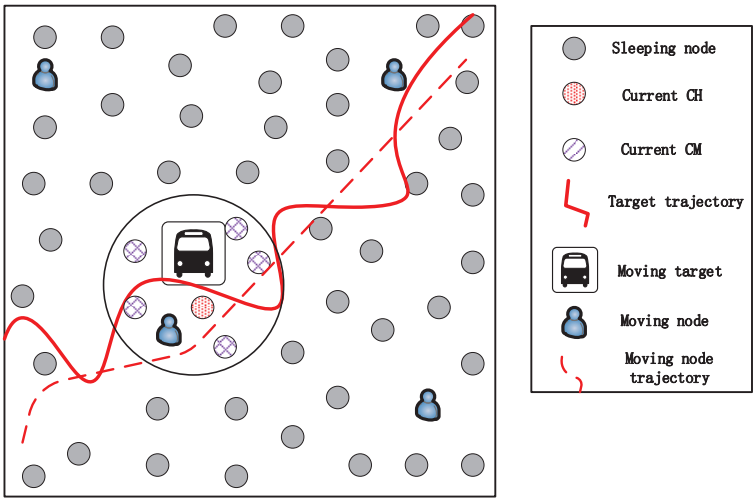


**Figure 1.** A moving target tracking scene in a wireless sensor network.

**Table 1.** Key symbols and their notations.

| Symbol | Notation | Symbol | Notation | Symbol | Notation |
|---|---|---|---|---|---|
| MN | Motion node | $\mathbf{c}$ | Position vector of $\Im$ | $\mathbf{m}$ | Position vector of MN |
| r | Sensing radius | $r_t$ | Uncertainty distance | $\mathbf{R}$ | Covariance matrix of $\mathbf{v}$ |
| SN | Static node | $\mathbf{x}$ | Target state vector | $p_d(s_i)$ | Probability of $\Im$ sensed by $s_i$ |
| $\Im$ | The target | $\mathbf{l}_i$ | Position vector of $s_i$ | $d(s_i, \Im)$ | Distance between $s_i$ and $\Im$ |
| $\Theta$ | Tracking cluster | $s_i$ | The $i^{th}$ sensor node | $P_D$ | Probability of $\Im$ sensed by $\Theta$ |
| $\mathbf{w}$ | Process noise | $\mathbf{z}$ | Measurement vector | $\mathbf{Q}$ | Covariance matrix of $\mathbf{w}$ |
| $Y_k$ | Cluster node set | $\mathbf{v}$ | Measurement noise | $\Delta$ | Sampling time interval |
| $N_k$ | Number of $Y_k$ | $\mathbf{v}_k$ | Velocity vector of $\Im$ | $E_{sp}$ | Sensing and processing cost |
| $E_r$ | Receiving cost | $E_t$ | Transmission cost | $E_{con}$ | Total energy cost of a node |
| $\hat{\mathbf{x}}$ | Estimation of $\mathbf{x}$ | $\mathbf{u}_k$ | Innovation sequence | $\mathbf{P}^{xz}$ | Cross covariance matrix |
| $\bar{\mathbf{x}}$ | Prediction of $\mathbf{x}$ | $\hat{\mathbf{P}}$ | Estimation of $\mathbf{P}$ | $\mathbf{P}^{zz}$ | Innovation covariance matrix |
| $\theta_0, \theta_1$ | Parameters of $P_D$ | $\mathbf{R}^0$ | Initial $\mathbf{R}$ of AUKF | $\mathbf{P}^{xx}$ | Error covariance of state |
| $\tau_0, \tau_1$ | Thresholds of $e_{s_i}$ | $\Omega_0, \Omega_1$ | Thresholds of $N_k$ | $e_{s_i}$ | Remaining energy of $s_i$ |
| $\lambda, \beta$ | Parameters of $p_d$ | $b_c$ | Bits of data packets | $e_t, e_r, e_d$ | c of energy cost |

### 2.2. Event-Detection Model and Tracking-Probability Definition

Event-based methods for sensing target in WSNs must consider the detection probability model. Several factors may influence the detection efficiency, such as sensor reliability, environmental conditions, and target characteristics [22]. This paper uses a hybrid detection model similar to that in the work [13,23] which merges the binary and probabilistic exponential detection model. This model is based on two thresholds $r$, $r_t$ ($r > r_t$) and considers three situations:

$$p_d(s_i) = \begin{cases} 1, & \mathrm{d}(\mathrm{s}_i, \Im) < r - r_t, \\ e^{-\lambda a^{\beta}} & \mathrm{r} - \mathrm{r}_t \leq \mathrm{d}(\mathrm{s}_i, \Im) \leq r + r_t, \\ 0 & \mathrm{d}(\mathrm{s}_i, \Im) > r + r_t, \end{cases} \tag{1}$$

where $d(s_i, \Im)$ is the distance between sensor $s_i$ and target $\Im$, $a = \mathrm{d}(\mathbf{s}_i, \Im) - (r - r_t)$ represents the target characteristic, and $0 < \lambda, \beta \leq 1$ represent the sensor technology and environment factors.

According to the above equation, if $d(s_i, \Im) \leq r + r_t$, the target could be detected with a probability. In this work, the cluster nodes will be selected considering the predicted position of target, which we will describe later. However, the predicted next position of target is not always very accurate. The nodes that closes to the predicted position of target may detect the target with a high probability at next timestep [24]. Thus, we take into account the distance-to-target of the SNs and ensure the target

is detected by the task cluster with a high tracking-probability when selecting the task cluster nodes. Suppose $n$ task cluster nodes track the target, the average of the detection probability of current task cluster nodes to the target is defined as the tracking-probability of current cluster

$$P_D = \frac{1}{n}\sum_{i=1}^{n} p_d(s_i). \tag{2}$$

See Figure 2 for an example that uses the model of target detection and tracking-probability when selecting task cluster nodes.

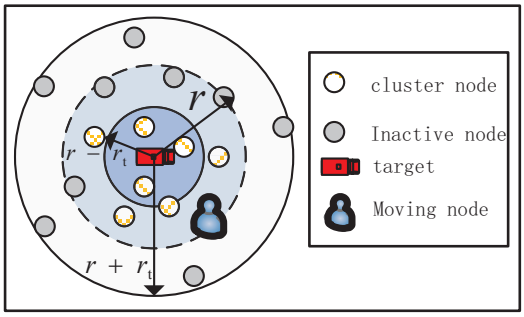


**Figure 2.** An example of target detection model and tracking-probability.

### *2.3. Motion and Measurement Models*

This paper considers only a single-target tracking problem. A four-dimensional state vector, $\mathbf{x}_k = [x(k), v_x(k), y(k), v_y(k)]^T$, denotes target state at timestep $k$, which includes the position vector $\mathbf{c}_k = [x(k), y(k)]^T$ and the velocity vector $\mathbf{v}_k = [v_x(k), v_y(k)]^T$. $\mathbf{l}_i = [s_x(i), s_y(i)]^T$ is the location of sensor nodes $s_i$. In this article, we assume that the sampling time interval between two successive timesteps, $\Delta$, is a constant under normal conditions. The motion state of target evolves according to the following discrete-time dynamic model [25]:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}) + \mathbf{w}_{k-1}, \tag{3}$$

where $f(*)$ is the state transition function of target, and $\mathbf{w}_{k-1}$ is the process noise vector, assumed the zero-mean white Gaussian with covariance matrix $\mathbf{Q}_{k-1}$. The statistics of initial state vectors $\hat{\mathbf{x}}_0$ and its error covariance matrix $\hat{\mathbf{Q}}_0$ are assumed to be known.

Suppose node $s_i$ is used to detect target $\Im$ at timestep $k$, the measurement $z_i(k)$ is given by [11]

$$z_i(k) = h_i(\mathbf{x}_k) + v_{i,k}, \tag{4}$$

where $h_i(*)$ is the measurement function and $v_{i,k}$ is the measurement noise of $s_i$ at the $k$th timestep. Although the practical measurement noise distribution of each sensor node is very complex, $v_{i,k}$ is assumed as an independent and identically distributed (i.i.d.) Gaussian random variable with zero mean and identical $\sigma^2$ to simplify the model, as in [11]. Note that the task cluster node set $\Upsilon_k$ will vary with different timestep. Denote $N_k$ be the number of $\Upsilon_k$. Then the sensor measurements at the $k$th time step can be indicated in a vector form [26]:

$$\mathbf{z}_k = \mathbf{h}(x_k) + \mathbf{v}_k = \begin{bmatrix} h_1(\mathbf{x}_k) \\ \vdots \\ h_i(\mathbf{x}_k) \\ \vdots \\ h_{N_k}(\mathbf{x}_k) \end{bmatrix} + \begin{bmatrix} v_{1,k} \\ \vdots \\ v_{i,k} \\ \vdots \\ v_{N_k,k} \end{bmatrix}. \tag{5}$$

Then, the measurement noise covariance matrix $\mathbf{R}(k)$ can be obtained:

$$\mathbf{R}(k) = diag(\sigma^2, ..., \sigma^2)_{N_k \times N_k}. \tag{6}$$

For SNs $s_i$, its measurement function is given by

$$h_i(\mathbf{x}_k) = \sqrt{(s_x(i) - x(k))^2 + (s_y(i) - y(k))^2}. \tag{7}$$

For MNs $m_j$, suppose it locates in $\mathbf{m}_j(k) = [u_{j,x}(k), u_{j,y}(k)]$ at timestep $k$, then its measurement function can be given as follows

$$\hbar_j(\mathbf{x}_k) = \begin{bmatrix} h_j(\mathbf{x}_k) \\ \varnothing_j(\mathbf{x}_k) \end{bmatrix} = \begin{bmatrix} \sqrt{(u_{j,x}(k) - x(k))^2 + (u_{j,y}(k) - y(k))^2} \\ \arctan((u_{j,y}(k) - y(k))/(u_{j,x}(k) - x(k))) \end{bmatrix}, \tag{8}$$

where $h_j(\mathbf{x}_k)$ is the measurement function of distance sensor which measures the distance-to-target of the MNs $m_j$ and $\varnothing_j(\mathbf{x}_k)$ is the measurement function of angular sensors which measures the angular between the $m_j$ and the target.

Note that, under normal conditions, the MNs only turn on their distance sensors, and once the target is declared to be lost, their angular sensors will be also switched on to detect the lost target.

### *2.4. Energy Consumption Model*

Energy consumption is considered as the most important tracking cost. The proposed energy consumption model is based on the power and activation time of different functional modules: Sensor, Microprocessor, and Transceiver [27]. Thus, there are three main aspects consuming energy for task nodes, namely, target sensing, data processing and data communication [28]. Task cluster nodes, in active state, always have their modules on to acquire and process information data about the target and transmit or receive data, which results in most of the energy consumption. The inactive SNs which spend most of their time in sleep state during which they only periodically sense the target and receive messages. To simplify system models, we assume that there is no energy consumption of SNs in inactive state. In addition, the energy consumption of MNs is also negligible for they can replenish their energy because of the mobility [29].

Static nodes adopt the energy consumption model similar to work [11]. Let $E_{sp}$ denote the energy consumption of static node $s_i$ in target sensing and data processing, regarded as a constant in this work. For data transmitting from $s_i$ to $s_j$, the energy cost to transmit $b_c$ bits data with a distance $r_{ij}$ is given by

$$E_t(i, j) = (e_t + e_d r_{ij}^{\iota}) b_c, \tag{9}$$

where $e_t$ and $e_d$ are decided by the transmitter, and $\iota$ depends on the channel characteristics, assumed to be time-invariant; the energy cost in receiving data by $s_i$ from other nodes is

$$E_r(i) = e_r b_c, \tag{10}$$

where $e_r$ is decided by receiver install in sensor node $s_j$. Hence, the total energy consumption of $s_i$ as a CM during a timestep yields

$$E_{con}(i) = E_t(i, ch) + E_{sp}, \tag{11}$$

and the total energy cost of a CH during a timestep is

$$E_{con}(ch) = E_r(ch) * N_k + E_{sp} + E_t(ch), \tag{12}$$

where $E_t(ch)$ stands for the energy cost of transmitting the fusing result to the related nodes (e.g., the MNs and the nest CH).

## 3. Adaptive Unscented Kalman Filter Algorithm for Target Tracking

Each CM will measurement the distance between the target and itself. When the preset time is up, they send their measurements to their CH, in which a filter algorithm will be performed to fuse the inaccurate measurements and product some accurate estimations [1]. Unscented Kalman filter (UKF) is a considerably typical nonlinear filter algorithm, which was proposed for state estimation in nonlinear dynamic system. As it has many merits such as simplicity in realization, high accuracy, and rapid convergence [30,31]. In this work, a nonlinear distance-based observation model is adopted, and UKF is used due to its superior performance for maneuvered targets [32]. However, when the maneuvering target moves with time-varying speed in the monitoring area, the standard UKF may cannot estimate the target state robustly because of the highly time-varying process noise [33], and then current task cluster may lost the target. Therefore, we put forward a robust adaptively UKF (AUKF) algorithm to estimate the maneuvering target with time-varying speed.

### *3.1. Standard Unscented Kalman Filter: A Brief Review*

With respect to the motion and measurement models which have been described in Section 2.3, the nonlinear estimation based on standard UKF can be briefly expressed as [13,30]:

1. Compute weights with the initial parameter $0 < \omega_0 < 1$:

$$\omega_j = \frac{(1-\omega_0)}{2n_x} \tag{13}$$

$$\mathbf{c}^0 = \sqrt{\frac{n_x}{1-\omega_0}}, \quad \mathbf{c}^j = \sqrt{\frac{n_x}{1-\omega_0}}\mathbf{r}^j, j = 1, \cdots, 2n_x, \tag{14}$$

where $n_x$ is the dimension of the state vector, $\{\mathbf{r}^j; j = 1, \cdots, n_x\}$ is the unit vector of the $j$th dimension and $\mathbf{r}^j = -\mathbf{r}^{(j-n_x)}$ when $j = n_x + 1, \cdots, 2n_x$.

2. At timestep $k$, establish symmetric sigma points $\boldsymbol{\phi}$ about the previous state estimation with the last estimation of target state $\hat{\mathbf{x}}_{k-1|k-1}$ and error covariance matrix $\hat{\mathbf{P}}^{xx}_{k-1|k-1}$:

$$\boldsymbol{\phi}^{(j)}_{k-1|k-1} = \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{D}_{k-1|k-1}\mathbf{c}^j, j = 0, \cdots, 2n_x, \tag{15}$$

where $\mathbf{D}_{k-1|k-1} = (\hat{\mathbf{P}}^{xx}_{k-1|k-1})^{1/2}$ is the square root of $\hat{\mathbf{P}}^{xx}_{k-1|k-1}$.

3. Predict the target state at timestep $k$ $\bar{\mathbf{x}}_{k|k-1}$ and its error covariance matrix $\bar{\mathbf{P}}^{xx}_{k|k-1}$:

$$\bar{\boldsymbol{x}}_{k|k-1} = \sum_{j=0}^{2n_x} \omega_j f(\boldsymbol{\phi}^{(j)}_{k-1|k-1}) \tag{16}$$

$$\bar{\mathbf{P}}^{xx}_{k|k-1} = \sum_{j=0}^{2n_x} \omega_j [f(\boldsymbol{\phi}^{(j)}_{k-1|k-1}) - \bar{\mathbf{x}}_{k|k-1}] * [f(\boldsymbol{\phi}^{(j)}_{k-1|k-1}) - \bar{\mathbf{x}}_{k|k-1}]^T + \mathbf{Q}_{k-1}, \tag{17}$$

where $\mathbf{Q}_{k-1}$ is the process noise covariance matrix at timestep $k-1$.

4. Establish symmetric sigma points $\boldsymbol{\phi}$ about the state prediction:

$$\boldsymbol{\phi}^{(j)}_{k|k-1} = \bar{\mathbf{x}}_{k|k-1} + \mathbf{D}_{k|k-1}\mathbf{c}^{(j)}, j = 0, \cdots, 2n_x, \tag{18}$$

where $\mathbf{D}_{k|k-1}$ is also the square root of $\bar{\mathbf{P}}^{xx}_{k|k-1}$.

5. Predict the innovation covariance matrix $\bar{\mathbf{P}}^{zz}_{k|k-1}$ and cross covariance matrix $\bar{\mathbf{P}}^{xz}_{k|k-1}$:

$$\bar{\mathbf{P}}^{zz}_{k|k-1} = \sum_{j=0}^{2n_x} \omega_j [h(\boldsymbol{\phi}^{(j)}_{k|k-1}) - \bar{\mathbf{z}}_{k|k-1}] * [h(\boldsymbol{\phi}^{(j)}_{k|k-1}) - \bar{\mathbf{z}}_{k|k-1}]^T + \mathbf{R}_k \tag{19}$$

$$\bar{\boldsymbol{P}}^{xz}_{k|k-1} = \sum_{j=0}^{2n_x} \omega_j [f(\boldsymbol{\phi}^{(j)}_{k-1|k-1}) - \bar{\boldsymbol{x}}_{k|k-1}] * [h(\boldsymbol{\phi}^{(j)}_{k|k-1}) - \bar{\boldsymbol{z}}_{k|k-1}]^T, \tag{20}$$

where $\bar{\mathbf{z}}_{k|k-1} = \sum_{j=0}^{2n_x} \omega_j h(\boldsymbol{\phi}^{(j)}_{k|k-1})$ is the prediction of measurement and $R_k$ is the measurement noise covariance matrix at timestep $k$.

6. Calculate current Kalman gain $\mathbf{K}_k$ and then obtain the estimation of current state $\hat{\mathbf{x}}_{k|k}$ and its error covariance matrix $\hat{\mathbf{P}}^{xx}_{k|k}$ using current actual measurement $\mathbf{z}^0_k$.

$$\mathbf{K}_k \triangleq \bar{\mathbf{P}}^{xz}_{k|k-1} (\bar{\mathbf{P}}^{zz}_{k|k-1})^{-1} \tag{21}$$

$$\hat{\mathbf{x}}_{k|k} = \bar{\mathbf{x}}_{k|k-1} + \mathbf{K}_k (\mathbf{z}^0_k - \bar{\mathbf{z}}_{k|k-1}) \tag{22}$$

$$\hat{\mathbf{P}}^{xx}_{k|k} = \bar{\mathbf{P}}^{xx}_{k|k-1} - \mathbf{K}_k \bar{\mathbf{P}}^{xz}_{k|k-1} (\mathbf{K}_k)^T. \tag{23}$$

As shown in Equations (17) and (19), to run UKF, users need to provide noise covariance $\mathbf{Q}_{k-1}$ and $\mathbf{R}_k$. Thus, performance of UKF depends on how well users can configure the $\mathbf{Q}_{k-1}$ and $\mathbf{R}_k$ for current applications. Conventionally, they are often configured as constant matrices during the running of standard UKF using a trial-and-error approach, which relies on the experience and background of users.

### *3.2. Adaptive Unscented Kalman Filter*

The standard UKF algorithm works well under suitable prior $\mathbf{Q}$ and $\mathbf{R}$. However, when the target moves with time-varying noise, the standard UKF may fail and thus its estimation results become inaccurate and not robustness due to the mismatch between the prior process noise distribution and the actual one [34]. To address this challenge, we propose a robust and efficient adaptive unscented Kalman filter (AUKF) algorithm. The algorithm adaptively adjusts the prior process noise covariance matrix $\mathbf{Q}$ based on the weighting combination of its current theoretical estimation value and the last data. It should be noted that, in this paper, we only update $\mathbf{Q}$ rather than $\mathbf{R}$, because the measurement noise $v$ of each sensor nodes is assumed to be a small variation.

The innovation sequence $\boldsymbol{\mu}_k$ denotes the additional information available to the filter as a consequence of the incoming new measurements. Hence, it is considered as the most relevant information for the filter adaptation and can be used to estimate the noise covariance [35]. According to Equation (3), the process noise can be represented as $\mathbf{w}_{k-1} = \mathbf{x}_k - f(\mathbf{x}_{k-1})$. Furthermore, from Equation (22) in Section 3.1, it yields

$$\begin{aligned} \hat{\mathbf{w}}_{k-1} &= \hat{\mathbf{x}}_k - f(\hat{\mathbf{x}}_{k-1|k-1}) = \hat{\mathbf{x}}_k - \bar{\mathbf{x}}_{k|k-1} \\ &= \mathbf{K}_k (\mathbf{z}^0_k - \bar{\mathbf{z}}_{k|k-1}) = \mathbf{K}_k \boldsymbol{\mu}_k. \end{aligned} \tag{24}$$

Therefore, the estimation of $\mathbf{Q}_{k-1}$ can be estimated as:

$$\mathbf{Q}_{k-1} = cov(\hat{\mathbf{w}}_{k-1}) = \mathbf{K}_k E[\boldsymbol{\mu}_k \boldsymbol{\mu}_k^T] \mathbf{K}_k^T, \tag{25}$$

where $E(*)$ is the expectation operation. To implement the above equation, $E(\boldsymbol{\mu}_k \boldsymbol{\mu}_k^T)$ is usually approximated by means of averaging $\boldsymbol{\mu}_k \boldsymbol{\mu}_k^T$ over time using a windowing method. Instead of using moving window methods (like in works [35]), this paper adaptively adjusts $\mathbf{Q}$ by utilizing a weighting factor $\lambda$ to balance the last noise covariance value and its current estimation. The weighting factor $\lambda \in (0, 1)$ is used to ensure the update strength. Therefore, the $\mathbf{Q}$ is updated as:

$$\mathbf{Q}_{k-1} = (1 - \lambda) \mathbf{Q}_{k-1} + \lambda (\mathbf{K}_k \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T \mathbf{K}_k^T), \tag{26}$$

When the covariance matrices $\mathbf{Q}_{k-1}$ are updated, state estimations $\hat{\mathbf{x}}_{k|k}$ and $\hat{\mathbf{P}}^{xx}_{k|k}$ should be corrected with the new $\mathbf{Q}_{k-1}$, which are given as follows:

$$\hat{\mathbf{P}}^{xz}_{k|k} = \sum_{j=0}^{2n_x} \omega_j [\boldsymbol{\phi}^{(j)}_{k|k} - \hat{\mathbf{x}}_{k|k}] * [h(\boldsymbol{\phi}^{(j)}_{k|k}) - \hat{\mathbf{Z}}_{k|k}]^T \tag{27}$$

$$\bar{\mathbf{P}}^{xx}_{k|k} = \sum_{j=0}^{2n_x} \omega_j [\boldsymbol{\phi}^{(j)}_{k|k} - \hat{\mathbf{x}}_{k|k}]^T * [\boldsymbol{\phi}^{(j)}_{k|k} - \hat{\mathbf{x}}_{k|k}] + \mathbf{Q}_{k-1} \tag{28}$$

$$\hat{\mathbf{K}}_k \triangleq \hat{\mathbf{P}}^{xz}_{k|k} (\bar{\mathbf{P}}^{zz}_{k|k})^{-1} \tag{29}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k} + \hat{\mathbf{K}}_k (\mathbf{z}^0_k - \hat{\mathbf{Z}}_{k|k}) \tag{30}$$

$$\hat{\mathbf{P}}^{xx}_{k|k} = \bar{\mathbf{P}}^{xx}_{k|k} - \hat{\mathbf{K}}_k \hat{\mathbf{P}}^{xz}_{k|k} (\hat{\mathbf{K}}_k)^T. \tag{31}$$

The overall procedure of the proposed AUKF algorithm is summarized in Algorithm 1.

**Algorithm 1:** The adaptive Unscented Kalman filter (AUKF) algorithm.

**Input:** $f(*), h(*), \hat{\mathbf{x}}_0, \mathbf{Q}_0, \mathbf{R}_1, \hat{\mathbf{P}}_0, \lambda$.

Initialization:

$\omega_j = (1-\omega_0)/2n_x$; $\mathbf{c}^0 = \sqrt{\frac{n_x}{1-\omega_0}}$; $\mathbf{c}^j = \sqrt{\frac{n_x}{1-\omega_0}}\mathbf{r}^j, j = 1, \cdots, 2n_x$.

**for** $k = 1 \to K$ **do**

  Implement the standard UKF to obtain $\hat{\mathbf{x}}_{k|k}$, $\bar{\mathbf{P}}^{zz}_{k|k-1}$, $\mathbf{K}_k$, $\hat{\mathbf{P}}^{xx}_{k|k}$.

  Update the $\mathbf{Q}_{k-1}$:

    $\boldsymbol{\mu}_k = \mathbf{z}^0_k - h(\bar{\mathbf{x}}_{k|k-1})$

    $\mathbf{Q}_{k-1} \leftarrow (1-\lambda)\mathbf{Q}_{k-1} + \lambda(\mathbf{K}_k \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T \mathbf{K}_k^T)$;

  Correct state estimations:

    $\hat{\mathbf{K}}_k \triangleq \hat{\mathbf{P}}^{xz}_{k|k} (\bar{\mathbf{P}}^{zz}_{k|k})^{-1}$;

    $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k} + \hat{\mathbf{K}}_k (\mathbf{z}^0_k - \hat{\mathbf{Z}}_{k|k})$;

    $\hat{\mathbf{P}}^{xx}_{k|k} = \bar{\mathbf{P}}^{xx}_{k|k} - \hat{\mathbf{K}}_k \hat{\mathbf{P}}^{xz}_{k|k} (\mathbf{K}_k)^T$;

    $\mathbf{Q}_k \leftarrow \mathbf{Q}_{k-1}$, $\mathbf{R}_{k+1} \leftarrow \mathbf{R}_k$.

    Save the $\hat{\mathbf{x}}_{k|k}$ and $\hat{\mathbf{P}}^{xx}_{k|k}$.

**end for**

## 4. Selection of Task Cluster

In WSNs, each SN usually has limited bandwidth and energy resources. Additionally, not all nodes that detect the target contribute equally to the tracking. Therefore, to increase the lifetime of a WSN, only some SNs should be activated to act as task cluster nodes and other SNs should keep being asleep.

According to Section 2.2, in which we discuss the detection model and tracking-probability definition, the selected SNs should locate as close to the target as possible. Thus, nodes that close to the predicted target position will be selected as cluster nodes with a high priority. In addition, the candidate SNs with high remaining energy should also be given preference to act as cluster nodes to balance the energy distribution. Therefore, we cast such a selection problem as an optimization problem as

$$\begin{cases} \min \ \Phi_k = \sum\limits_{s_i \in \Gamma_k} \frac{(d(s_i, \Im))^2}{e_{s_i}} \\ s.t. \quad P_D = \frac{1}{N_k} \sum\limits_{s_i \in \Gamma_k} p_d(s_i) \geq \frac{\theta_0}{N_k} \\ \qquad \mathrm{N}_k > \Omega_0 \\ \qquad e_{s_i} > \tau_0 \end{cases}, \tag{32}$$

where $N_k$ is the node number of selected set $\Gamma_k = \{s_i; i = 1, \cdots, N_k\}$ which acts as the task cluster to detect the current target at timestep $k$, and $e_{s_i}$ is the remaining energy of node $s_i$. As described in Equation (32), there are three requirements to restrict the selected task cluster $\Gamma_k$: (1) the predicted tracking-probability of the selected $\Gamma_k$, $P_D$, should first exceed a threshold $\frac{\theta_0}{N_k}$; (2) $N_k$ should also exceed a threshold $\Omega_0$ to guarantee tracking precision (note that, if the number of candidate nodes is less than $\Omega_0$ and more than $\Omega_1$, all candidate nodes will form the task cluster, otherwise the target may enter a coverage hole and the recovery mechanism should be performed); and (3) node should be equipped with enough energy (at least $\tau_0$ J) to work normally. After satisfying the three requirements, we try to keep $\Phi_k$ as smaller as possible to save energy. As for the selection of CH, the node $s_j$ with the minimum $\frac{(d(s_j,\Im))^2}{e_{s_j}}$ in $\Gamma_k$ will be selected as the CH, which is described as

$$s_j = \underset{\forall s_i \in \Gamma_k, e_{s_i} > \tau_1}{\arg\min} \left(\frac{(d(s_i,\Im))^2}{e_{s_i}}\right), \tag{33}$$

where $\tau_1$ is the least energy to ensure a CH work normal. Figure 3 describes the process of selecting next task nodes.

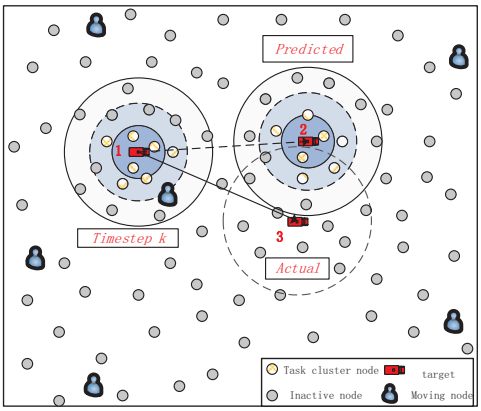


**Figure 3.** An example of selecting the next cluster nodes. At timestep $k$, the CH will predict the target position at timestep $k+1$ according to the current estimations of target state. Then, the SNs close to the predicted position and equipping with much energy will be activated as cluster nodes. However, if the maneuvering target changes its trajectory or speed, the selected task cluster may fail to detect it. Then, the target recovery mechanism will be implemented, which we will introduce later.

## 5. Tracking the Target with Mobile Sensors

### *5.1. Description of Tracking Process with Mobile Nodes*

After deployment of the sensor network, all sensor nodes are assumed to have known their own positions and then acquire the location information of their neighbor nodes by exchanging beacon messages. At first, all SNs are in sleep mode, but periodically awake to receive messages. The MNs are assumed to locate at the area boundary and be in charge of detecting whether a target is approaching the monitoring area. Once a target is detected, the MNs sensing the target will compute the target position by performing the trilateration. After the target is located, the MN nearest the target will select some SNs around the target to form a task cluster and it will also approach the predicted position as soon as possible and cooperate with the static task cluster to perform the tracking task.

Once a SN is activated by the last CH, its PIR sensor will be turned on and the tracking task begins. Note that we assume that each sensor node can compute and store data locally, as well as replying with data packets locally.

In general, each CM (not the CH and the MN) will perform the following tasks during the timestep $k$:

1. Once the PIR sensors make a positive detection, it will turn on the distance-measuring sensor to achieve the distance-to-target.
2. When the preset time interval is up, the node will send a data packet which includes its current measurements and remaining energy information to the CH and the closest MN after a random delayed time with the conflict detect mechanism, CSMA/CA.
3. Once sending the data packet successfully, it will shut down its sensors and turn into sleep mode again to save energy until awakened next time.

As for the CH, which acts as the scheduler of the task cluster, it needs to perform the following operations:

1. After receiving the activated message packet from the last CH, it extracts and saves the previous state information of the target, and then it will also execute the detection task like that in the CM.
2. When the preset time interval is up, it begins to receive the data packets from its CMs and the MN. Then, it carries out standard UKF algorithm to fuse different measurements with its own measurements and then obtains current estimations of target state as well as its predictions.
3. It extracts the remaining energy information of its neighbour nodes from the data packet coming from the MN and then chooses appropriate cluster nodes and a new CH for next cluster according to the method described in Section 4.
4. It sends a data packet which includes current estimations of target state and its predictions to the MN and activates the next cluster nodes.
5. After reporting the results to related nodes, it also closes its sensors and puts into sleep state until awakened next time.

In this paper, the selected MN services as a sink node due to its superior communication ability and sufficient energy. Thus, it will perform the following works during a timestep under normal conditions:

1. It will approach the predicted position of target at current timestep as soon as possible and then implement the detection task like the cluster node.
2. When the preset time interval is up, it sends a data packet including its measurements and the remaining energy information of the neighbour nodes of current CH.
3. Once receiving the data packet from the CH, it forwards the current state information of the target to the remote end by some internets (e.g., the cellular network) and also shares the information with other MNs.
4. It will select the MN nearest to the predicted position of target as the next mobile sink.

### *5.2. Analysis of Mobile Nodes in Tracking*

In this work, we assume that the MNs could move anywhere in a random way and their sensing and communication radius are much greater than that of static nodes. Before the target appears at the monitor area, these MNs will locate in the area boundary to detect if there exists a target that will enter the area. During one timestep, the MN will service as a mobile sink and participate in detecting the target. Furthermore, at the end of current timestep, the MN closest to the predicted position of the target in the next timestep will be selected in advance, and then the selected MN will approach the predicted position as soon as possible. In addition, the selected MN could detect the target with a probability 1 when it arrives at the predicted position of target owing to superior sensing ability. That is to say, only one suitable mobile node will take part in current tracking task during one timestep, and other mobile nodes will go on keeping detection state or move to someplace to replenish their energy if they are running out their energy.

According to the description in Section 5.1, the selected MN will service as a mobile sink to collect and forward relevant information as well as participating in detecting the target in this work. Thus,

there are two functions that the MN preforms in tracking the mobile target, namely, mobile sink and tracking node. Next, take Figure 4 as an example to illustrate the performance of the MN in tracking a mobile target.

- *Performance as the mobile sink.* As the sinks, node needs to gather information from current cluster head and forward it to a remote end. As shown in Figure 4a, four fixed sinks are involved in the monitor area. If current cluster head closes to one of sinks, it could communicate with the sink directly. When current cluster head is far away the fixed sinks, it has to depend on a relay node to communicate with the closest sink, which brings in a heavy communication burden. While, in this work, the selected MN will service as a mobile sink and keep close to current cluster during a timestep. Hence, current cluster head can directly communicate with the mobile sink without any relay nodes as shown in Figure 4b.
- *Performance as the tracking node.* To ensure a high tracking accuracy, the tracking scheme should select a task cluster with a tracking-probability. Thus, as shown in Figure 4a, six static nodes are selected as current task nodes to ensure a high tracking-probability. Nevertheless, when a mobile node is involved, only two static nodes are required to ensure a high tracking-probability, which can been seen in Figure 4b. That is because the selected MN will move close to target, improving the detecting probability and saving the energy consumption of static nodes [21].

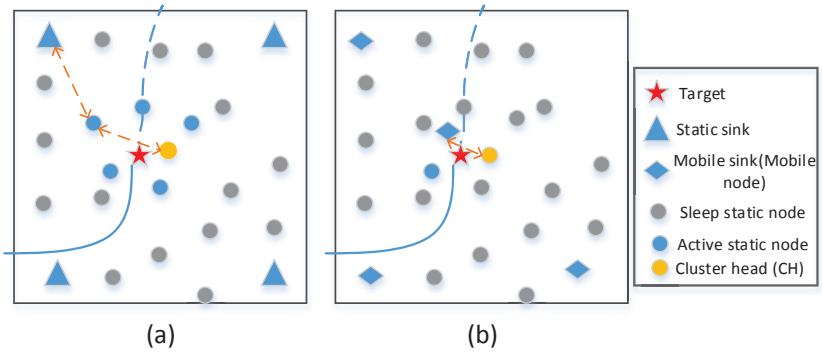


**Figure 4.** Illustration of the tracking methods based on the static nodes (left, the method used in work [13]) and hybrid nodes (right, the method used in this work): (**a**) four fixed sinks are involved and six static nodes required to form a task cluster to track the target in current timestep; and (**b**) four mobile nodes works in the monitor area and one of them cooperates with the task cluster that only consists of two static nodes.

## 6. Recovery Mechanism for Target Lost

In the cluster-based network, the CH has responsibility to predict the next position of mobile target and activates the next respective CH and its CMs that the target is approaching in advance to carry out further tracking. The prediction is only based on target's present speed and direction and thus the network may lose the target [36]. The reasons of losing the target can be summarized as follows [18]:

- *Localization errors:* As mentioned earlier, only some sensor nodes are awakened to track the target for saving energy. Localization is never perfect no matter what estimation methods (e.g., EKF, UKF or PF) are used. Furthermore, the estimation errors may have a cumulative effect on estimating the target state. Then, an inaccurate estimation of target location may result in prediction errors which can further lead to target loss, since an unsuitable cluster is wakened in advance.
- *Communication failures:* Sensor nodes may be unable to communicate due to some obstacles, such as trees, stones, and buildings. Moreover, packet loss and delay in response owing to communication breakdown, overload, and environmental factors can also be considered in this case.

- *Node failures:* Sensor nodes in WSNs have limited battery capacity and unreliable components in order to reduce costs. Thus, node failures may occur due to software or hardware failure, battery discharge, enemy action, etc.
- *Abrupt change in target's speed or direction:* The target may change its trajectory or speed suddenly because of the internal or external factors. In this case, the difference between actual and prior prediction position of target becomes so large that the active cluster cannot track the target efficiently.
- *Target enters the coverage hole in WSN:* The coverage holes exist in the sensor networks due to the uneven deployment of the sensor nodes [15]. The tracking network system may lose the target when it enters the holes where only few nodes could detect the target.

In this paper, the problem we discuss is to track a maneuvering target with time-varying speed. Thus, we only consider the following failure reasons: the case "*Abrupt change in target's speed or direction*" and the case "*Target enters the coverage hole in WSN*". Without loss of generality, the recovery mechanism can be divided into two distinct phases: (1) declaration of lost target; and (2) target recovery.

### *6.1. Declaration of Lost Target*

The recovery mechanism is initiated when cluster reports loss of target. Thus, before initiating the target recovery mechanism, we should first confirm whether the target has been lost. As the target moves away from current cluster, the current CH will send wake-up message to activate the next cluster nodes and the MN closest to the predicted target position will also follow the target. If the selected cluster cannot sense the target well in some stipulated time, it will declare that the target is lost and inform the nearest MN to start up the target recovery mechanism. The criterion of judging that a cluster could sense the target well yields

$$P_D > \theta_1 \quad \text{and} \quad \mathrm{D}_k > \Omega_1, \tag{34}$$

where $\mathrm{D}_k$ is the number of active nodes which could detect current target and $\theta_1$ is a parameters of $P_D$. Otherwise, the task cluster will declare that the target is lost.

Then, we will describe the decision process in detail by taking an example. As shown in Figure 3, the target is tracked by the cluster 1 during timestep $k$. At the end of timestep $k$, the CH in cluster 1 predicts the target will be likely to move to the cluster 2 according to the current direction and speed of target. However, the maneuvering target suddenly changes its speed or direction during timestep $k+1$, and then the target locates actually at cluster 3. Therefore, there are only three cluster nodes in cluster 2 cloud sense the target, and other selected cluster nodes will turn into sleep state after sending a report message to their CH. Clearly, the $P_D$ of cluster 2 is less than $\theta_1$ or the $\mathrm{D}_k \leq \Omega_1$. Under this circumstance, the CH of cluster 2 will inform the nearest MN that the target has been lost.

### *6.2. Target Recovery Method*

To enable an energy-efficient and robust target recovery method, one needs to consider from both the MN and SNs. In this paper, a novel target recovery method is employed to continue acquiring the target state during the period at which the task cluster loses the target. On the other hand, based on the estimation information of the target, the MN can efficiently recover the tracking of target, while saving energy by decreasing the number of awakened SNs. Our proposed recovery method has following three steps:

(1) *The MN detects and tracks the target:* Once receiving the message about the loss of target, the MN will continue to detect and track the target by using AUKF algorithm. The initialize process noise covariance matrix and error covariance matrix in AUKF are both set as the values of their previous timestep. However, the initialize measurement noise covariance matrix will be updated as a new value $\mathbf{R}^0$ Furthermore, to improve the tracking accuracy, the MN will reduce its sampling time interval to $\Delta/N_\Delta$. After $N_\Delta$ samples, the target position at next $N_\Delta$ timestep will be predicted and all SNs whose

sensing range covers the position will be acquired. Then, if the number of those nodes is no less than $\Omega_1$, the nodes will be activated to form a recovery cluster and detect the target, otherwise the MN will continue to execute the above operations.

(2) *The recovery cluster detects the target:* The recovery cluster nodes will detect the target as soon as they are activated. If the cluster cannot sense the target well in some stipulated time, the CH will inform the MN the target is still lost, and then the MN will go on detecting the target as described in step 1.

If the recovery cluster could detect the target well, the location process will be implemented. In this paper, we use the classic trilateration method to acquire the target position. Readers could refer to work [37] for more details about the trilateration method. The location process of trilateration is described in Figure 5. Note that if the number of detection nodes is two, the MN will serve as a complement node and if the number is more than three, the three nodes with the minimum $\frac{(d(s_j,\Im))^2}{e_{s_j}}$ will be chosen for saving energy. Suppose the computational position of target at timestep $K1 - N_\Delta/2$ and $K1$ are $(p_x^1, p_y^1)$ and $(p_x^2, p_y^2)$, respectively. Then, the target velocity at timestep $K1$ is given by

$$\begin{cases} \mathrm{v_x} = (p_x^2 - p_x^1)/(\Delta/2) \\ \mathrm{v}_y = (p_y^2 - p_y^1)/(\Delta/2) \end{cases} \tag{35}$$

After obtaining the new estimation of target position and velocity, the sampling time interval of the tracking system will be recovered to $\Delta$.

(3) *The downstream cluster tracks the target:* Once the recovery cluster obtains the new target state information, the standard UKF algorithm will be used to predict the next target state. Furthermore, the related nodes will be activated as the downstream task cluster and all the active nodes involved in recovery fall asleep as soon as the target recovery message is received except those that are selected as downstream cluster nodes.

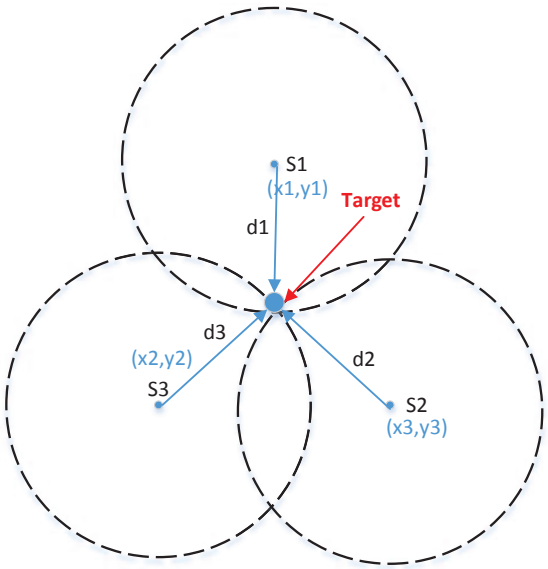


**Figure 5.** Description of the trilateration method.

The target recovery algorithm is summarized in Algorithm 2.

**Algorithm 2:** The target recovery mechanism.

**Step 1: The MN detects and tracks the target:**
Reduce its sampling time interval to $\Delta/N_\Delta$, and implement AUKF to estimate the position of target at each timestep.
After $N_\Delta$ samples, predict the next position of target.
Activate all static nodes whose sensing range covers the predicted position.
**If** there are no appropriate nodes to form a recovery task cluster, **then**
  continue to implement the **step 1**.
**end if**.
**Step 2: The recovery cluster detects the target:**
**If** the cluster could track the target well according to the Equation (34), **then**
  Execute the location process two times to obtain the target position and velocity.
  Recover the sampling time interval to $\Delta$
**else**
  the current CH informs the MN that the target is lost and skip to **step 1**.
**end if**.
**Step 3: The downstream cluster tracks the target:**
Initialize the noise covariance with $\mathbf{Q}_0$ and $\mathbf{R}_0$, target state with the position and velocity of target.
Perform the standard UKF to predict the next target state, and select the downstream cluster nodes.
The recovery cluster broadcasts a target recovery message and activate the downstream task cluster to work.

## 7. Simulation and Performance Evaluation

In this section, we evaluate the performance of the proposed robust tracking scheme with a simulation framework in three different cases, as well as compare it with other related methods. Our experiments are performed on an Intel Core i7-6700 3.4 GHz PC with 16G memory and implemented in Matlab R2015b.

### *7.1. Simulation Setup*

In our simulation, we consider the tracking scenario as shown in Figure 1. A maneuvering target with time-varying speed (e.g., vehicle) moves in a 100 m $\times$ 100 m square area with coordinates from $[0, 100]$ to $[0, 100]$ which is covered by $N_s$ SNs and a few MNs. In terms of the target motion, we use a simple linear model to represent the moving target with the discrete time dynamic state equation. More complex models require a priori knowledge, often unavailable in most situations, hence is not considered in this paper [13].

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}) + \mathbf{w}_{k-1} = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{w}_{k-1}, \tag{36}$$

where $\mathbf{A}$ is the state transition matrix. The sampling time interval $\Delta$, process noise covariance matrix $\mathbf{Q}$ and measurement noise covariance matrix $\mathbf{R}$ are constant under normal conditions during the tracking process. However, if the target is lost, the three parameters will be changed with the timestep by the scheduler of the recovery method. Without loss of generality, the initial energy of each SN distributes uniformly in $[0, 1](J)$, and the energy consumption model of SN in different roles has been described in Section 2.4. Additionally, it is assumed that there is no wireless transmission error when nodes communicate with each other.

In this paper, the root mean-squared error in position at each timestep, $RMSE_p$, and its average, $ARMSE_p$, are adopted as the indications of tracking accuracy, since it yields a combined measurement of the bias and variance of a filter estimate [38]. The $ARMSE_p$ is given by

$$ARMSE_p = \sqrt{\frac{1}{N_m K}\sum_{i=1}^{N_m}\sum_{k=1}^{K}[(\hat{x}_k^{(i)} - x_k)^2 + (\hat{y}_k^{(i)} - y_k)^2]}, \tag{37}$$

and the $RMSE_p$ at timestep $k$ yields

$$RMSE_p(k) = \sqrt{\frac{1}{N_m}\sum_{i=1}^{N_m}[(\hat{x}_k^{(i)} - x_k)^2 + (\hat{y}_k^{(i)} - y_k)^2]}, \tag{38}$$

where $(\hat{x}_k^{(i)}, \hat{y}_k^{(i)})$ is the estimated target position in timestep $k$ at $i$-th Monte Carlo run. $N_m = 1000$ is the number of Monte Carlo runs and $K = 60$ is the number of sample steps in one run. Other parameter settings in the simulations are shown in Table 2.

**Table 2.** The settings of system parameters in our simulation environment.

| | |
|---|---|
| $\mathbf{Q} = 2 * \begin{bmatrix} \frac{\Delta^3}{3} & \frac{\Delta^2}{2} & 0 & 0 \\ \frac{\Delta^2}{2} & \Delta & 0 & 0 \\ 0 & 0 & \frac{\Delta^3}{3} & \frac{\Delta^2}{2} \\ 0 & 0 & \frac{\Delta^2}{2} & \Delta \end{bmatrix}$, | $\mathbf{A} = \begin{bmatrix} 1 & \Delta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta \\ 0 & 0 & 0 & 1 \end{bmatrix}$, |
| $\hat{\mathbf{x}}_0 = [16.18, 2.14, 81.32, -4.75]$, | $\hat{\mathbf{P}}_0 = diag([0.2, 0.3, 0.2, 0.3])$, |
| $\sigma^2 = 2$, | $\Delta$ = 0.5 s, |
| r = 10 m, | t = 2 m, |
| $\lambda = 0.5$, | $\beta = 0.5$, |
| $\tau_0$ = 0.05 J | $\tau_1$ = 0.2 J, |
| $\Omega_0 = 4$, | $\Omega_1 = 2$, |
| $\theta_0 = 2.5$, | $\theta_1 = 1$, |
| $\mathbf{R}^0 = diag([1, 0.0001])$, | $\iota = 2$, |
| $e_t = 4.5 \times 10^{-5}$ J/bit, | $e_d = 1.0 \times 10^{-8}$ J/bit, |
| $e_r = 1.35 \times 10^{-4}$ J/bit, | $e_{sp} = 8.0 \times 10^{-7}$ J/bit, |
| $b_c$ = 48 bits, | $N_\Delta = 4$. |

We carry out our simulation experiment under two typical scenarios, including both the normal situation and target is lost due to the uneven distributed nodes or the abrupt change of the target speed. Then, the performance of the proposed tracking scheme can be evaluated under these situations. There is no need to change the scheme itself when facing the different situations. Note that all results are averaged by $N_m = 1000$ Monte Carlo runs.

### *7.2. Tracking Performance under Normal Circumstances*

In this section, to evaluate the tracking performance of our proposed scheme, we assume that the target will not be missing during the tracking. Thus, to ensure the assumption, there is no coverage hole in the sensor network and target speed would not change suddenly and greatly. For fully evaluating the tracking performance, we take into account two different sensor networks: uniformly and randomly distributed sensor networks.

Two metrics have been used in the performance analysis.

(1) *Tracking errors*. As shown with the red dotted line in Figure 6, a maneuvering target move along a curve trajectory in the monitored area which is assumed to be covered by $N_s$ uniformly distributed SNs. One of the estimated target trajectories is displayed with green solid line. The tracking errors shown in Figure 7 is indicated by the RMSE in position ($RMSE_p$) at each

timestep. The minimum and maximum $RMSE_p$ are separately 0.5046 m and 1.9921 m, and the $ARMSE_p$ is 0.8920 m. As for the tracking errors in randomly distributed sensor networks, Figures 8 and 9 show, respectively, one of the estimated target trajectories and tracking errors. The minimum and maximum $RMSE_p$ are respectively 0.5684 m and 1.9463 m, and the $ARMSE_p$ is 0.8670 m.

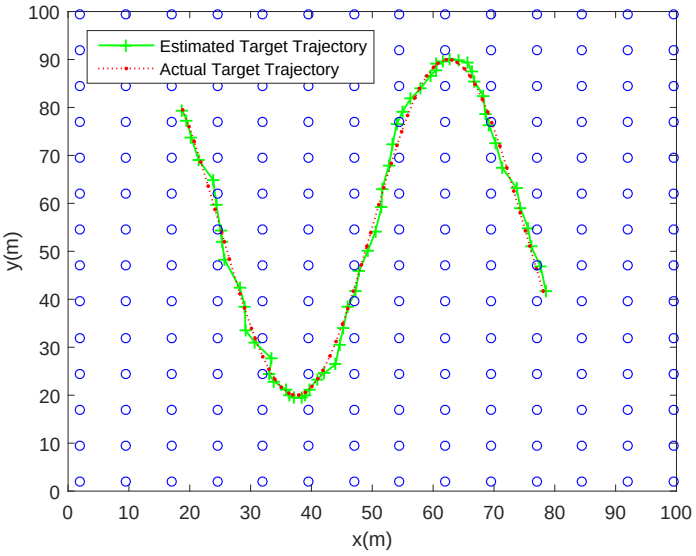


**Figure 6.** One of tracking trajectories using our proposed tracking scheme in a uniformly distributed sensor network.

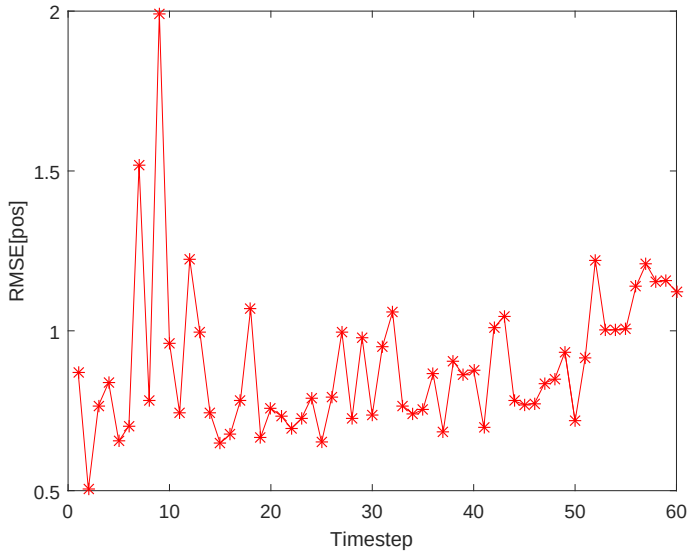


**Figure 7.** Tracking errors ($RMSE_p$) at each timestep using our proposed tracking scheme.

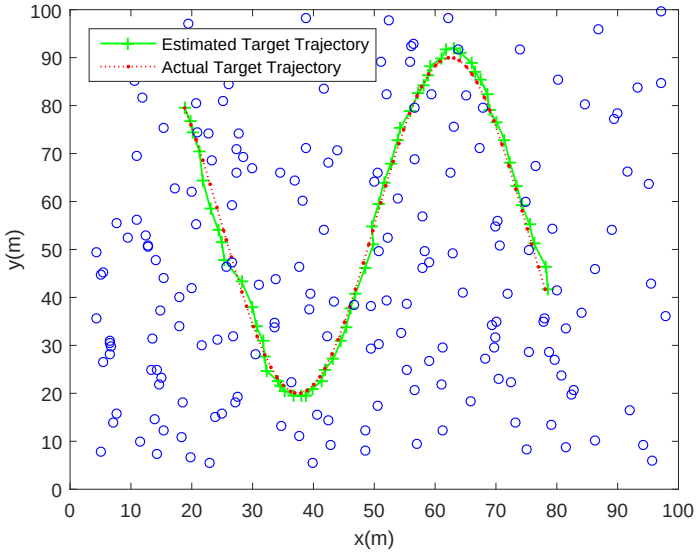


**Figure 8.** One of tracking trajectories using our proposed tracking scheme in a randomly distributed sensor network.

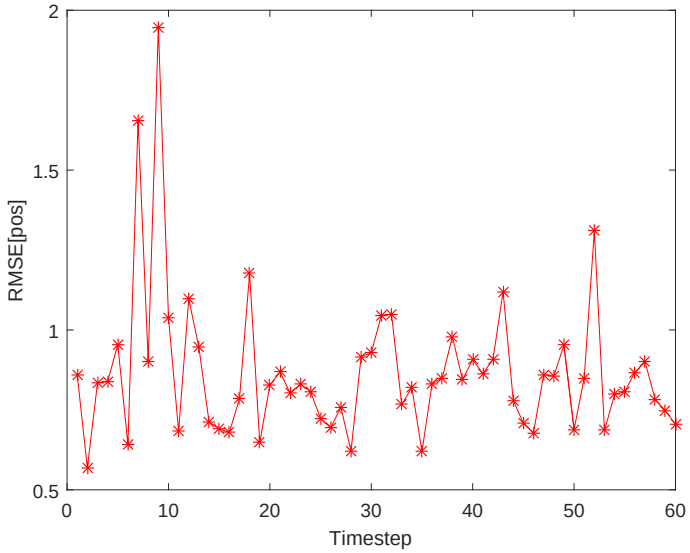


**Figure 9.** Tracking errors ($RMSE_p$) at each timestep using our proposed tracking scheme in a randomly distributed sensor network.

(2) *Total energy consumption*. The amount of energy consumed by the whole network to monitor the mobile target is another important metric to measure the practicality of our scheme. The averaged energy consumption of the proposed tracking scheme in one tracking action used in the randomly distributed sensor network is 2.4623 J, higher than that in the uniformly distributed sensor network a bit (2.3449 J). The reason for this is that the proposed method may activate more SNs due to the uneven distribution in the randomly distributed sensor network.

### *7.3. Performance Analysis of Mobile Nodes in Tracking the Target*

To evaluate the benefits of using mobile nodes in this work, we compare the tracking based on hybrid nodes (our proposed method, THN) with the tracking only based on static nodes (TSN). To be fair, the two methods use the same cluster node selection mechanism and UKF algorithm as described

in this paper. To clearly present the difference between two methods, $\Omega_0$ and $\theta_1$ are separately set to 3 and 3.5, and other parameter settings are similar to Table 2.

Figure 10 shows the number of activated cluster nodes in each timestep used the two different methods. In this figure, we can see that the number of activated cluster nodes in TSN is greater than or equal to that in THN at each timestep. As for the comparisons of tracking errors, the two methods have nearly the same good performance, which is shown in Figure 11. The $ARMSE_p$ of THN and TSN are, respectively, 0.9286 m and 0.9163 m. Therefore, in Figures 10 and 11, we can find that the use of the MNs in target tracking could decrease the number of activated task nodes and then obtain an energy-saving with a good performance in tracking errors.

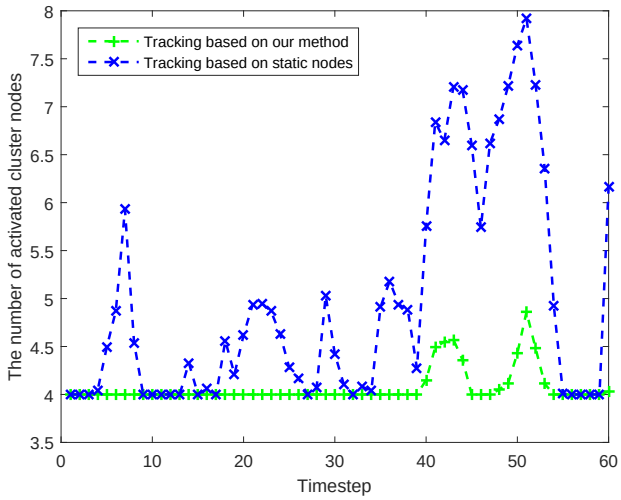


**Figure 10.** Comparisons of the number of activated cluster nodes in each timestep.

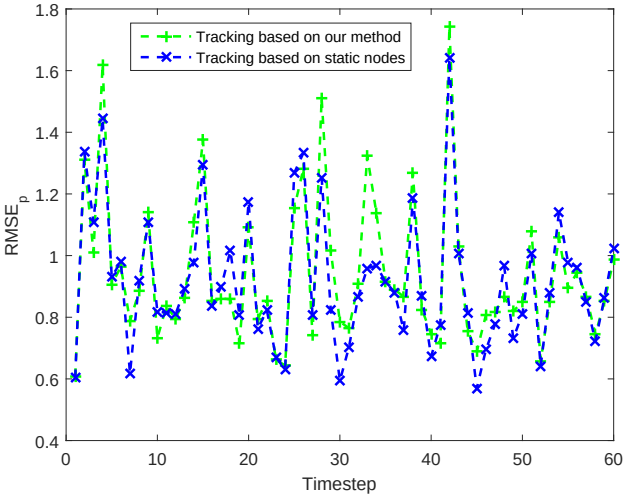


**Figure 11.** Comparisons of tracking errors ($RMSE_p$) at each timestep.

### *7.4. Recovery Performance When Target Is Lost*

In this section, we will evaluate the performance of our loss recovery mechanism and compare it with the classic source recovery mechanism (SRM) used in work [17]. Readers can refer to work [17] for more details about the source recovery mechanism. Meanwhile, to further illustrate the superiority of the proposed AUKF algorithm in the recovery mechanism, we also compare it with the recovery mechanism with the standard UKF algorithm.

#### 7.4.1. Abrupt Change in Target's Speed or Direction

In this case, we will test the recovery mechanism of our tracking scheme in the situation that the target is lost as its speed or direction is changed suddenly during moving in the monitor area. To avoid the situation that the target may enter the coverage hole, we assume that the monitored area is covered by $N_s = 196$ uniformly distributed SNs.

As shown in Figure 12, the target suddenly increases its speed at timestep 3 and decreases it speed at timestep 10, and then it begin to change its direction. In Figures 12 and 13, it can be seen that the proposed recovery mechanism with AUKF could works well when the target is lost. After the task cluster declares that target is lost, the MN goes on tracking the target by using the AUKF. However, if the MN tracks the target by using UKF, the tracking performance will suffer from degradation and even divergence after losing the target. That is because the actual process noise distribution will mismatch with the assumed one when the motion state of the target occurs abrupt change, leading to a biased or even divergent filter solutions [34]. Therefore, the proposed recovery mechanism with UKF could not recover the target tracking in this case. As for the recovery performance of the classic SRM, the current task cluster nodes will activate their neighboring nodes if the target is not in its detection area. Furthermore, if the target is still not found, all the sensor nodes in the network will be activated to looking for the target. Therefore, in this case, the SRM could ensure to recover the target, despite missing the target state in some timesteps.

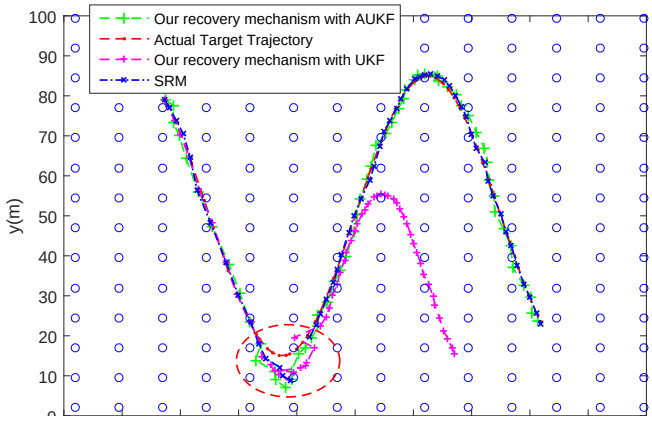


**Figure 12.** Tracking trajectories of the target under different recovery mechanism when the target suddenly change its speed or direction.

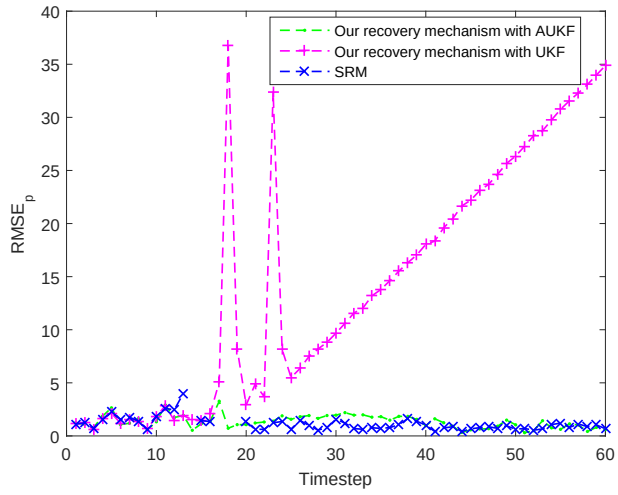


**Figure 13.** Tracking errors ($RMSE_p$) at each timestep under different recovery mechanism when the target suddenly change its speed or direction.

Table 3 summarizes the averaged activated node amount in one tracking action for recovering the target tracking and the $ARMSE_p$ of the three different methods. Our proposed recovery mechanism with AUKF outperforms the SRM by about 75% in the averaged activated node amount in one tracking action which can be adopted as the indication of energy consumption with an almost identical $ARMSE_p$. Obviously, the $ARMSE_p$ of the proposed recovery mechanism with UKF is the largest and we have explained the reasons in the above discussion.

**Table 3.** Features of different recovery mechanisms.

| Recovery Mechanisms | Averaged Amount of Activated Nodes in One Tracking Action | $ARMSE_p$ |
|---|---|---|
| Our recovery mechanism with AUKF | 6.010 | 1.581 m |
| Source recovery mechanism (SRM) | 24.505 | 1.378 m |
| Our recovery mechanism with UKF | - | 18.360 m |

7.4.2. Target Enters Coverage Holes in the Monitoring Area

In this case, we will test the recovery mechanism of our tracking scheme in the situation that the target will enter a coverage hole due to the uneven distribution of SNs. Similarly, we assume that the target would not suddenly change its speed to focus on the problem of coverage hole.

As shown in Figure 14, the target will enter a coverage hole at the timestep 41 from which there is no suitable node could track the target. In this case, the proposed recovery mechanism with AUKF and UKF are both works well. When the target enters a coverage hole, the related MN is informed that the target is lost and begins to track the target alone. Although the node number is less, the sample rate is raised and the motion state of target does change significantly. Thus, the MN can obtain a good estimation of target state by using AUKF or UKF algorithm. However, from the Figures 14 and 15 we can find that the SRM could not track the target when the target locates in the coverage hole. That is because the SRM recovers the target tracking by means of activating related neighbour nodes to find the lost target, but there is no nodes in the coverage hole.

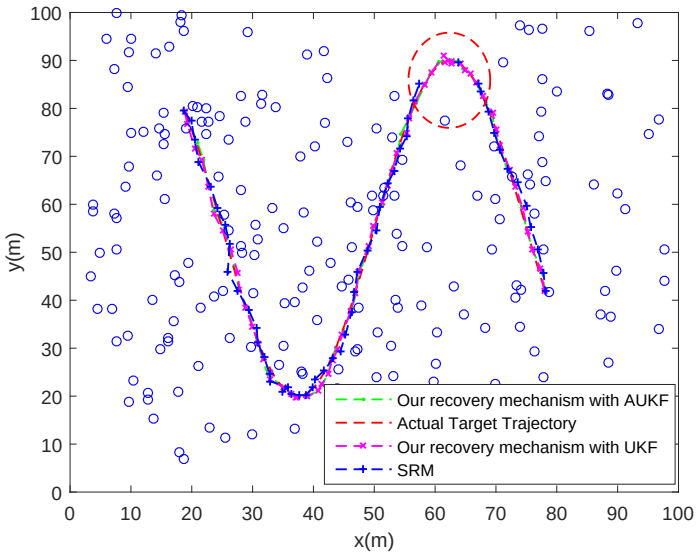


**Figure 14.** Tracking trajectories of the target under different recovery mechanism when the target enters a coverage hole.

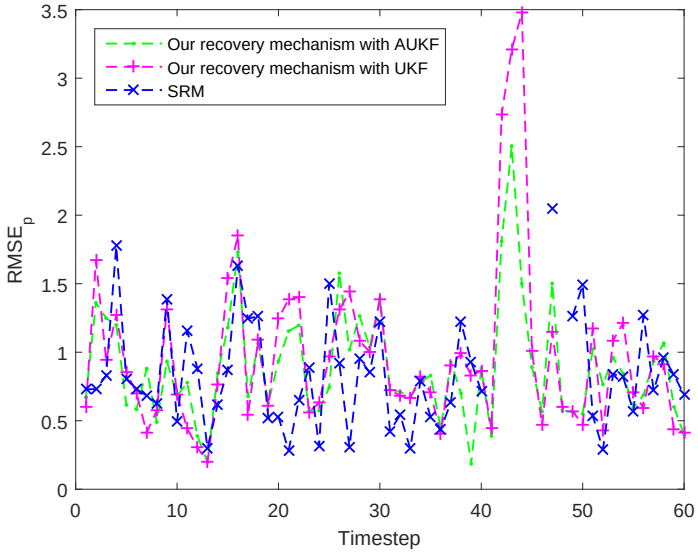


**Figure 15.** Tracking errors ($RMSE_p$) at each timestep under different recovery mechanism when the target enters a coverage hole.

Table 4 also summarizes the averaged activated node amount in one tracking action for recovering the target tracking and the $ARMSE_p$ of the three different methods in this case. We can find the proposed recovery mechanism with AUKF achieves the best performance in both two features. Additionally, the performance of the proposed recovery mechanism with UKF is very close to that with AUKF in this case. As for the SRM, the averaged activated node amount of it in one tracking action is far more than that of the previous two, which will consume a great deal of energy. The reason for this is, when the activated neighbour nodes also cannot find the lost target, all nodes in the network will be activated to look into the target according to the theory of SRM.

**Table 4.** Features of different recovery mechanisms.

| Recovery Mechanisms | Averaged Amount of Activated Nodes in One Tracking Action | $ARMSE_p$ |
|---|---|---|
| Our recovery mechanism with AUKF | 3.5 | 0.977 m |
| Source recovery mechanism (SRM) | 433.996 | 2.140 m |
| Our recovery mechanism with UKF | 3.7 | 1.153 m |

## 8. Conclusions and Future Work

In this paper, we present a novel loss recovery and tracking scheme for maneuvering target in hybrid WSNs where a few MNs are used to cooperate with SNs to build up robust and efficient tracking networks. Based on the hybrid WSN, we consider a cluster-based single target-tracking scene. By dynamically scheduling the MNs and static cluster nodes, the tracking probability and accuracy can be effectively guaranteed with fewer cluster nodes and less energy consumption compared with the tracking only based on static nodes. In addition, in view of the the fact that the task cluster may lose the mobile target when the target abruptly changes its target's velocity or enters coverage holes in the deployment monitoring area, we propose a novel loss recovery mechanism by using the characteristics of the hybrid WSNs. Furthermore, an adaptive UKF (AUKF) is proposed for the MN to track the lost target robustly. The simulation results demonstrate that the proposed loss recovery and tracking scheme behaves really well in improving the robustness and accuracy of recovering and tracking the mobile target as well as decreasing the amount of the activated task nodes.

In our future endeavors, we will aim to carry out our work on investigating the multi-target tracking schemes in hybrid WSNs, which is more complicated than tracking in the single tracking

scenario. Furthermore, the loss recovery mechanism will be also extended to the multi-target tracking scenario. Additionally, the number of mobile nodes which take part in the tracking task in one timestep is fixed with one. In our following work, we will research the influence of the number of mobile nodes which take part in the tracking task in one timestep on the tracking accuracy and the energy-saving in hybrid WSNs. Moreover, we will also focus on how to move the mobile nodes to save their energy and ensuring the tracking-probability of task nodes.

**Author Contributions:** The work presented here was carried out in collaboration between all authors. Hanwang Qian and Xiaobing Yuan conceived and designed the experiments; Pengcheng Fu performed the experiments; Baoqing Li analyzed the data; Jianpo Liu contributed reagents/materials/analysis tools; and Hanwang Qian wrote the paper.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Souza, E.L.; Nakamura, E.F.; Pazzi, R.W. Target Tracking for Sensor Networks: A Survey. *ACM Comput. Surv.* **2016**, *49*, 1–31.
2. Brunelli, D.; Minakov, I.; Passerone, R.; Rossi, M. POVOMON: An Ad-hoc Wireless Sensor Network for indoor environmental monitoring. In Proceedings of the 2014 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems, Naples, Italy, 17–18 September 2014; pp. 1–6.
3. Yoo, J.; Kim, H.J. Target Localization in Wireless Sensor Networks Using Online Semi-Supervised Support Vector Regression. *Sensors* **2015**, *2015*, 12539–12559.
4. Cenedese, A.; Luvisotto, M.; Michieletto, G. Distributed clustering strategies in industrial wireless sensor networks. *IEEE Trans. Ind. Inform.* **2017**, *13*, 228–237.
5. Akyildiz, I.F.; Weilian, S.; Sankarasubramaniam, Y.; Cayirci, E. A survey on sensor networks. *IEEE Commun. Mag.* **2002**, *40*, 102–114.
6. Han, G.; Shen, J.; Liu, L.; Shu, L. BRTCO: A Novel Boundary Recognition and Tracking Algorithm for Continuous Objects in Wireless Sensor Networks. *IEEE Syst. J.* **2017**, *PP*, 1–10.
7. Enayet, A.; Razzaque, M.; Hassan, M.; Almogren, A.; Alamri, A. Moving Target Tracking through Distributed Clustering in Directional Sensor Networks. *Sensors* **2014**, *14*, 24381–24407.
8. Shi, K.; Chen, H.; Lin, Y. Probabilistic coverage based sensor scheduling for target tracking sensor networks. *Inf. Sci.* **2015**, *292*, 95–110.
9. Shang, C. An Efficient Target Tracking Mechanism for Guaranteeing User-Defined Tracking Quality in WSNs. *IEEE Sens. J.* **2015**, *15*, 5258–5271.
10. Wen, Y.; Gao, R.; Zhao, H. Energy Efficient Moving Target Tracking in Wireless Sensor Networks. *Sensors* **2016**, *16*, 29.
11. Hu, X.; Hu, Y.H.; Xu, B. Energy-Balanced Scheduling for Target Tracking in Wireless Sensor Networks Sensor Networks. *ACM Trans. Sens. Netw. (TOSN)* **2014**, *11*, 21.
12. El-Fouly, F.H.; Ramadan, R.A.; Mahmoud, M.I.; Dessouky, M.I. REBTAM: Reliable energy balance traffic aware data reporting algorithm for object tracking in multi-sink wireless sensor networks. *Wirel. Netw.* **2016**, 1–19, doi:10.1007/s11276-016-1365-1.
13. Fu, P.; Tang, H.; Cheng, Y.; Li, B.; Qian, H.; Yuan, X. An energy-balanced multi-sensor scheduling scheme for collaborative target tracking in wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2017**, *13*, doi:10.1177/1550147717698968.
14. Zhou, Z.; Du, C.; Shu, L.; Hancke, G.; Niu, J.; Ning, H. An Energy-Balanced Heuristic for Mobile Sink Scheduling in Hybrid WSNs. *IEEE Trans. Ind. Inform.* **2016**, *12*, 28–40.
15. Patil, S.; Gupta, A.; Zaveri, M. Recovery of Lost Target Using Target Tracking in Event Driven Clustered Wireless Sensor Network. *J. Comput. Netw. Commun.* **2014**, *2014*, 15.
16. Hsu, J.M.; Chen, C.C.; Li, C.C. POOT: An efficient object tracking strategy based on short-term optimistic predictions for face-structured sensor networks. *Comput. Math. Appl.* **2012**, *63*, 391–406.
17. Samarah, S.; Al-Hajri, M.; Boukerche, A. A Predictive Energy-Efficient Technique to Support Object-Tracking Sensor Networks. *IEEE Trans. Veh. Technol.* **2011**, *60*, 656–663.

18. Gupta, A.; Patil, S.; Zaveri, M. Lost Target Recovery in Wireless Sensor Network Using Tracking. In Proceedings of the International Conference on Communication Systems and Network Technologies, Rajkot, India, 11–13 May 2012; pp. 352–356.
19. Mahboubi, H.; Masoudimansour, W.; Aghdam, A.G.; Sayrafian-Pour, K. An Energy-Efficient Target-Tracking Strategy for Mobile Sensor Networks. *IEEE Trans. Cybern.* **2017**, *47*, 511–523.
20. Lembke, K.; Kietlinski, L.; Golanski, M.; Schoeneich, R. RoboMote: Mobile Autonomous Hardware Platform for Wireless Ad-hoc Sensor Networks. In Proceedings of the 2011 IEEE International Symposium on Industrial Electronics (ISIE), Gdansk, Poland, 27–30 June 2011; pp. 940–944.
21. Wang, T.; Peng, Z.; Liang, J.; Wen, S.; Bhuiyan, M.Z.A.; Cai, Y.; Cao, J. Following targets for mobile tracking in wireless sensor networks. *ACM Trans. Sens. Netw. (TOSN)* **2016**, *12*, 31.
22. Nakamura, E.F.; Souza, E.L. Towards a flexible event-detection model for wireless sensor networks. In Proceedings of the IEEE Symposium on Computers and Communications, Riccione, Italy, 22–25 June 2010; pp. 459–462.
23. Liu, W.R.; He, Y.; Zhang, X.Y.; Jiang, F.; Gao, K.; Xiao, J.M. Energy-Efficient Node Scheduling Method for Cooperative Target Tracking in Wireless Sensor Networks. *Math. Probl. Eng.* **2015**, *2015*, doi:10.1155/2015/627479.
24. SanMiguel, J.C.; Cavallaro, A. Cost-Aware Coalitions for Collaborative Tracking in Resource-Constrained Camera Networks. *IEEE Sens. J.* **2015**, *15*, 2657–2668.
25. Oguz-Ekim, P.; Gomes, J.P.; Xavier, J.; Oliveira, P. Robust Localization of Nodes and Time-Recursive Tracking in Sensor Networks Using Noisy Range Measurements. *IEEE Trans. Signal Process.* **2011**, *59*, 3930–3942.
26. Lin, J.; Xiao, W.; Lewis, F.L.; Xie, L. Energy-Efficient Distributed Adaptive Multisensor Scheduling for Target Tracking in Wireless Sensor Networks. *IEEE Trans. Instrum. Meas.* **2009**, *58*, 1886–1896.
27. Zhou, H.Y.; Luo, D.Y.; Gao, Y.; Zuo, D.C. Modeling of Node Energy Consumption for Wireless Sensor Networks. *Wirel. Sens. Netw.* **2011**, *3*, 18–23.
28. Fu, P.; Cheng, Y.; Tang, H.; Li, B.; Pei, J.; Yuan, X. An Effective and Robust Decentralized Target Tracking Scheme in Wireless Camera Sensor Networks. *Sensors* **2017**, *17*, 639.
29. Xing, G.; Li, M.; Wang, T.; Jia, W.; Huang, J. Efficient Rendezvous Algorithms for Mobility-Enabled Wireless Sensor Networks. *IEEE Trans. Mob. Comput.* **2012**, *11*, 47–60.
30. Julier, S.J.; Uhlmann, J.K. Unscented filtering and nonlinear estimation. *Proc. IEEE* **2004**, *92*, 401–422.
31. Gustafsson, F.; Hendeby, G. Some Relations between Extended and Unscented Kalman Filters. *IEEE Trans. Signal Process.* **2012**, *60*, 545–555.
32. Julier, S.J.; Uhlmann, J.K. Corrections to "Unscented Filtering and Nonlinear Estimation". *Proc. IEEE* **2005**, *92*, 1958.
33. Hajiyev, C.; Soken, H.E. Robust adaptive unscented Kalman filter for attitude estimation of pico satellites. *Int. J. Adapt. Control Signal Process.* **2014**, *28*, 107–120.
34. Song, Q.; Han, J. An Adaptive UKF Algorithm for the State and Parameter Estimations of a Mobile Robot. *Acta Automatica Sin.* **2008**, *34*, 72–79.
35. Zhou, J.; Knedlik, S.; Loffeld, O. INS/GPS tightly-coupled integration using adaptive unscented particle filter. *J. Navig.* **2010**, *63*, 491–511.
36. Khare, A.; Sivalingam, K.M. On recovery of lost targets in a cluster-based wireless sensor network. In Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops, Seattle, WA, USA, 21–25 March 2011; pp. 208–213.
37. Fang, B.T. Trilateration and Extension to Global Positioning System Navigation. *J. Guid. Control Dyn.* **1986**, *9*, 715–717.
38. Li, X.R.; Zhao, Z. Evaluation of estimation algorithms part I: Incomprehensive measures of performance. *IEEE Trans. Aerosp. Electron. Syst.* **2006**, *42*, 1340–1358.