Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Energy-efficient localization and target tracking via underwater mobile sensor networks*

Hua-yan CHEN[1,2], Mei-qin LIU‡[1,2], Sen-lin ZHANG[2]

[1]*State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China*

[2]*College of Electrical Engineering, Zhejiang University, Hangzhou 310027, China*

E-mail: chenhuayan@zju.edu.cn; liumeiqin@zju.edu.cn; slzhang@zju.edu.cn

Received Sept. 11, 2017; Revision accepted Apr. 17, 2018; Crosschecked Aug. 15, 2018

Abstract: Underwater mobile sensor networks (UMSNs) with free-floating sensors are more suitable for understanding the immense underwater environment. Target tracking, whose performance depends on sensor localization accuracy, is one of the broad applications of UMSNs. However, in UMSNs, sensors move with environmental forces, so their positions change continuously, which poses a challenge on the accuracy of sensor localization and target tracking. We propose a high-accuracy localization with mobility prediction (HLMP) algorithm to acquire relatively accurate sensor location estimates. The HLMP algorithm exploits sensor mobility characteristics and the multi-step Levinson-Durbin algorithm to predict future positions. Furthermore, we present a simultaneous localization and target tracking (SLAT) algorithm to update sensor locations based on measurements during the process of target tracking. Simulation results demonstrate that the HLMP algorithm can improve localization accuracy significantly with low energy consumption and that the SLAT algorithm can further decrease the sensor localization error. In addition, results prove that a better localization accuracy will synchronously improve the target tracking performance.

Key words: Underwater mobile sensor networks; Energy-efficient; Sensor localization; Target tracking
https://doi.org/10.1631/FITEE.1700598 **CLC number:** TP274

1 Introduction

Seas and oceans that cover more than 70% Earth surface are mysterious and charismatic to us human beings because of huge amount of unexploited resources. Underwater wireless sensor networks (UWSNs) are developing gradually to enhance our abilities to discover resources in aquatic environments (Sozer et al., 2000; Cui et al., 2006; Lloret, 2013). UWSNs are three-dimensional (3D) networks, composed of specific wireless sensors which rely on acoustic communication and float at different depths of underwater environments. These cooperative sensors enable a broad range of applications, such as environmental monitoring, undersea exploration, disaster prevention, and distributed tactical surveillance. UWSNs are the extension of wireless sensor networks (WSNs), which are used in terrestrial environments (Calafate et al., 2013). The significant differences between underwater networks and terrestrial networks are as follows: (1) Underwater acoustic communication features low bandwidth, long propagation delay, and high energy consumption; (2) Sensor nodes are prone to move with water currents, which results in dynamic networks known as underwater mobile sensor networks

‡ Corresponding author

* Project supported by the NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Informatization (No. U1609204), the National Natural Science Foundation of China (Nos. 61531015 and 61673345), and the Key Research and Development Program of Zhejiang Province, China (No. 2018C03030)

ORCID: Mei-qin LIU, http://orcid.org/0000-0003-0693-6574

(UMSNs). All these characteristics of UWSNs give rise to research interests in data gathering, synchronization, localization, target tracking, energy optimization, media access control (MAC), etc.

In this study, we address the issues of tracking a target moving in UMSNs. A UMSN with free-floating sensors is one of the various architectures of UWSNs. The primary challenge for target tracking in UMSNs is the localization scheme. As underwater sensors are battery-powered and it is impracticable to replace batteries when exhausted, energy efficiency has a high priority in underwater networks. In UMSNs, node locations change continuously, which means conventional localization schemes designed for static networks are inapplicable in view of energy efficiency. Because those localization schemes need to run frequently to guarantee a sufficient accuracy for target tracking, which will dramatically increase the communication energy cost, it is critical to design an appropriate localization scheme for target tracking in UMSNs.

To solve the aforementioned issues, we present a high-precision mobility prediction based simultaneous localization and target tracking (SLAT) algorithm for UMSNs. SLAT algorithms were proposed in recent years to deal with the problem of target tracking in networks with uncertain sensor locations. They can be divided into two types according to network characteristics. For static networks, SLAT algorithms can not only track the moving target, but also gradually obtain accurate location estimates of unknown sensors with processes of target tracking (Aggarwal and Wang, 2011; Kantas et al., 2012). For dynamic networks, SLAT algorithms can modify sensor locations during each step of target tracking processes with the help of prior knowledge and motion pattern of sensors (Teng et al., 2012; Li et al., 2013). Since UMSNs are dynamic networks with changing sensor locations, we design a special SLAT algorithm as an effective solution to target tracking problems in UMSNs.

The main contributions of this paper are twofold: (1) We propose a sensor mobility prediction scheme to predict sensor locations in future periods. This periodic localization scheme saves energy from the continuous localization scheme. (2) We give a simple yet useful SLAT algorithm to modify sensor positions during each step of target tracking processes. Simulations show that we can obtain more accurate estimates of sensor positions and the target state.

2 Related work

Target tracking is a focused application for underwater defense systems. Intended targets to be tracked are unmanned underwater vehicles (UUVs) and submarines. Some related works for target tracking in UWSNs can be found in the literature. Isbitiren and Akan (2011) presented a target tracking method that uses only measurement information for 3D underwater networks. It results in tracking failure if not enough sensors are involved. Wang et al. (2012) proposed an algorithm that combines the interacting multiple model (IMM) with the particle filter (PF) to cope with uncertainties in target maneuvers. Yu and Choi (2014) provided an algorithm, which increases energy efficiency of each sensor by implementing target tracking in a distributed architecture scheme named wake-up/sleep (WuS). WuS means waking up sensors that have an opportunity to detect the target at the next time step and sleeping sensors that could not detect the target. This approach saves energy via replacing centralized architectures (requiring a significantly high bandwidth for communications) with distributed architectures. Zhang et al. (2015) studied the effect of node topology on target tracking in UWSNs with quantized measurements. They proposed a target tracking scheme which selects the optimal topology by minimizing the posterior Cramer-Rao lower bound (PCRLB). Chen et al. (2017) derived an energy-efficient filter that implements the tradeoff between communication cost and tracking accuracy. It saves much energy while loosing little tracking accuracy or improves tracking performance with less additional energy cost. However, all the results above are based on static networks with an accurate node location. They fail to consider the dynamic networks with node mobility, which is more realistic in underwater environments. This problem is a new and important ground in underwater networks. In our work, we propose an energy-efficient solution to the target tracking problem in UMSNs.

For target tracking in UMSNs, the primary challenge is identifying sensor locations efficiently. In general, due to sensor mobility, UMSNs have to do real-time localization to ensure the availability of

sensor positions. Guo and Liu (2013) addressed the localization problem of active-restricted underwater sensor networks. Nodes send messages by turns and all nodes keep receiving information to obtain their current locations. This kind of localization scheme does not meet the energy saving requirement of underwater networks. Zhou et al. (2011) exploited the spatially correlated mobility pattern of UMSNs and applied it to localize the sensors. Sensors can predict future locations by a linear prediction algorithm, which saves energy from constant communications. Our work was inspired by Zhou et al. (2011). Since target tracking has the requirement of high localization accuracy, we improve the prediction algorithm in Zhou et al. (2011) to meet that requirement. A periodical localization scheme is introduced to reduce energy costs. In addition, we propose a simultaneous localization and target tracking scheme to improve both types of performance.

3 Problem formulation

UMSNs are composed of different types of free-floating sensors at different seawater layers. In this study, we consider the network architecture proposed by Zhou et al. (2011), which is composed of three types of sensors as shown in Fig. 1. Surface buoys float on the water and they can obtain their location estimates by their GPS equipment. They work as 'satellite nodes' in the underwater mobile network architecture. Super nodes float at different water depths and they can obtain their location estimates by direct contact with surface buoys. The details can be found in Austin et al. (2000). They serve as 'intermediary nodes' in the underwater mobile network architecture. As we know, a common GPS cannot

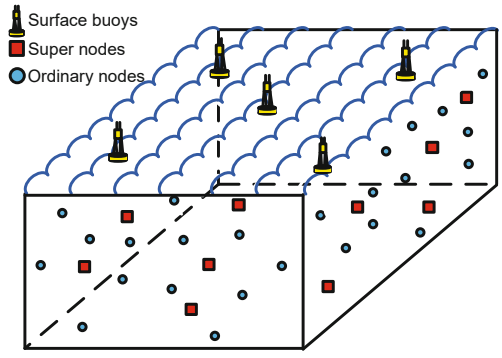

Fig. 1 Architecture of an underwater mobile sensor network

work in underwater environments. Underwater GPS systems, such as PARDIGM (Austin et al., 2000), have been proposed to obtain the absolute location information for underwater objects. In Kussat et al. (2005), simulation and experimental results showed that these underwater GPSs usually can provide position estimates with a centimetric accuracy even for mobile underwater nodes. Thus, it is safe for us to assume that surface buoys can be well localized by underwater GPS systems. In this study, we will not contribute to this part. Instead, we focus on tackling the localization challenge of ordinary nodes. Ordinary nodes are simple and widely distributed at different water depths. They play a major role in various underwater network applications. Since they are low-complexity sensors, ordinary nodes can communicate with only neighbors so they are not able to obtain their location estimates from surface buoys. These sensors must obtain their locations for the purpose that they can effectively participate in network applications.

The above descriptions of the three types of sensors tell that surface buoys and super nodes can locate themselves by convenient ways, but ordinary nodes are limited by their low-complexity hardware and need a special localization scheme. Most of the existing localization schemes were designed for static sensor networks, and localization processes need large energy consumption. In UMSNs, sensors move continuously with water current, so they are away from the original localization points. It is impractical to run those localization schemes frequently to obtain accurate location estimates due to limited battery capacity of underwater sensors, especially for ordinary nodes. However, the measurements for target tracking collected by these ordinary nodes are sensitive to their localization accuracy. So, we should find other ways to obtain locations of ordinary nodes and reduce their energy consumptions as far as possible.

3.1 Sensor mobility model

Research in hydrodynamics and relevant literature about object mobility in sea environments (Beerens et al., 1994; Mandal et al., 2017) showed that the movement of underwater sensors is not an irregular process. The mobility pattern of underwater sensors is determined mainly by tidal currents near the seashore. Fig. 2 depicts the sensor

mobility pattern in underwater environments. A sensor position changes periodically, forming a similar sine curve. We consider a kinematic model in Beerens et al. (1994), which is widely used (Zhou et al., 2011; Kim and Yoo, 2013) to model underwater environments. According to this model, velocities of a sensor in the x- and y-axis can be represented as

$$\begin{cases} V_x = k_1\lambda\sin(k_2x)\cos(k_3y) \\ \qquad + k_1\lambda\cos(2k_1t) + k_4, \\ V_y = -\lambda v\cos(k_2x)\sin(k_3y) + k_5, \end{cases} \tag{1}$$

where k_1, k_2, k_3, k_4, k_5, λ, and v are random variables related to environmental factors. This model manifests that velocities of a sensor in underwater environments are related to time t and its position. Fig. 2 and Eq. (1) indicate temporal and spatial correlations in this sensor mobility pattern, which tells us that sensor movements in underwater environment are predictable in nature.

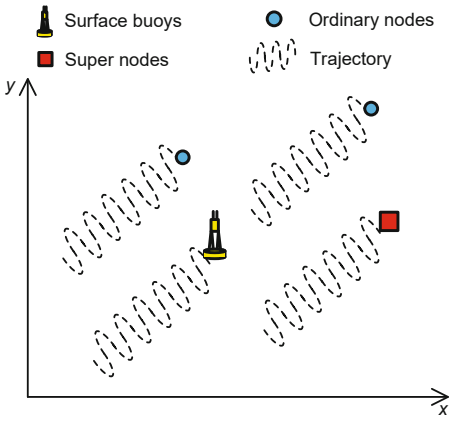

Fig. 2 Mobility pattern of underwater sensors

3.2 Multi-step linear prediction algorithm

Sensor movements in underwater environments are similar to the waveform of a voice signal which can be treated as the all-pole model or autoregressive (AR) model. Furthermore, referring to Pardey et al. (1996), the model of mobility pattern of underwater sensors is termed a non-adaptive AR model. For non-adaptive models, the parameters are chosen so as to give the best fit to a sequence of data samples. Because of this, non-adaptive models require that the signal should be stationary, which means that its statistical characteristics, such as average amplitude and frequency content, do not vary with time. Although the parameters in Eq. (1) will change over time, in different environments this change is usually slow and peaceful. So, it can also be considered locally stationary over short-time intervals. For non-adaptive AR models, the most commonly used algorithm is the Levinson-Durbin algorithm. In this study, we adopt the multi-step Levinson-Durbin prediction algorithm proposed by Brockwell and Dahlhaus (2004) for the purpose of a higher accuracy, as shown in Fig. 3. We give a brief introduction about this algorithm in the following.

Let X_t be a zero-mean stationary process and let

$$\hat{X}_{n+h}^{(h)}(m) = \sum_{j=1}^{m} \phi_m^{(h)}(j) X_{n+1-j} \tag{2}$$

be the best linear predictor of X_{n+h} given $X_n, X_{n-1}, \ldots, X_{n+1-m}$ with the corresponding mean squared error v_m^h, where m represents the number of historical data involved in the linear prediction and h represents the h^{th} future state to be predicted. Then the best linear predictor of X_{n+h} given $X_n, X_{n-1}, \ldots, X_{n-m}$ can be expressed as

$$\hat{X}_{n+h}^{(h)}(m+1) = \sum_{j=1}^{m+1} \phi_{m+1}^{(h)}(j) X_{n+1-j}. \tag{3}$$

The coefficients and mean squared errors can be expressed as

$$\begin{aligned} \phi_{m+1}^{(h)}(m+1) = & \Big(-\sum_{j=1}^{m} \phi_m(j)\gamma(m+1-j) \\ & + \gamma(m+h)\Big) v_m^{-1}, \end{aligned} \tag{4}$$

$$\phi_{m+1}^{(h)}(j) = \phi_m^{(h)}(j) - \phi_{m+1}^{(h)}(m+1)\phi_m(m+1-j), \tag{5}$$

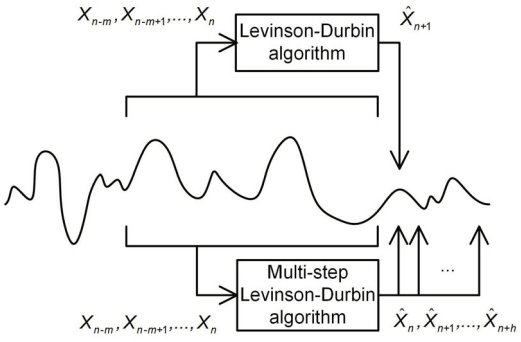

Fig. 3 Linear prediction using the Levinson-Durbin algorithm and the multi-step Levinson-Durbin algorithm

where $j = 1,2,\ldots,m$, $v_{m+1}^{(h)} = v_m^{(h)} - \phi_{m+1}^{(h)}(m+1)^2 v_m$, and $\gamma(i) = \lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} X_n X_{n-i}$ gives an infinite data sequence $X_{-\infty},\ldots,X_{\infty}$. In practice, we have only a finite data sequence $X_1, X_2, \ldots, X_N$ and $\gamma(i)$ will be

$$\gamma(i) = \frac{1}{N}\sum_{n=1}^{N} X_n X_{n-i}. \tag{6}$$

The initial conditions are $\phi_1^{(h)}(1) = \rho(h) = \gamma(h)/\gamma(0)$ and $v_1^h = \gamma(0)(1-\rho(h)^2)$. One-step prediction coefficients $\phi_m(j)$ $(j = 1,2,\ldots,m)$ and mean squared errors v_m are defined as in the traditional Levinson-Durbin algorithm. Details and derivation of the above algorithms can be found in Brockwell and Dahlhaus (2004).

4 High-precision localization with mobility prediction

Inspired by Zhou et al. (2011), we improve their scalable localization with mobility prediction (SLMP) algorithm and generate our high-precision localization with mobility prediction (HLMP) algorithm. Both localization schemes are designed to achieve a good balance between localization error and energy consumption. With mobility prediction, ordinary nodes can independently predict their location with less energy cost. Since the error of predicted location will increase gradually, ordinary nodes revise their location estimates by information received from super nodes after certain time periods, which guarantees the localization accuracy of ordinary nodes. The major superiorities of our HLMP algorithm are twofold: (1) For the purpose of eliminating cumulative errors, we substitute the multi-step Levinson-Durbin algorithm into the original Levinson-Durbin algorithm, which will be explained later; (2) Not only the mobility predictions but also their prediction errors are presented in our algorithm, which improves the localization accuracy and is in favor of further SLAT algorithms.

4.1 Super node mobility prediction

As we mentioned, in a UMSN architecture, a super node can communicate with surface buoys directly and estimate its current location at its will. So, a super node can calculate and record its velocity during each localization period T as

$$\boldsymbol{V}_k = (\boldsymbol{L}_k - \boldsymbol{L}_{k-1})/T + \boldsymbol{\Theta}_k, \tag{7}$$

where $\boldsymbol{L}_k$ denotes the super node location at time k and $\boldsymbol{\Theta}_k$ denotes the white Gaussian noise of the velocity sample due to the localization error of a super node. Then the super node stores a data sequence of velocity samples, $\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_N$, over time. The new velocity sample will edge out the oldest one to ensure the freshness of data and to save storage space. Given the sensor mobility pattern described in Eq. (1), this mobility pattern uses velocities of the x- and y-axis to represent the sensor mobility pattern in an underwater environment. Moreover, it indicates that sensors with the same (x, y) but different z (depth) move in the same mobility pattern, which is a common assumption in hydrodynamics. Thus, the velocity samples of super nodes contain their mobility pattern characteristics. According to Section 3.2, the mobility pattern of underwater sensors is termed a non-adaptive AR model, which can be solved by the Levinson-Durbin linear prediction algorithm. Using this algorithm, a super node can predict its next velocity based on previous velocity samples $[\boldsymbol{V}_k, \boldsymbol{V}_{k-1}, \ldots, \boldsymbol{V}_{k-(l-1)}]^{\mathrm{T}}$ and relevant coefficients $[\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_l]$, as follows:

$$\begin{aligned}\boldsymbol{V}_{k+1} &= \sum_{m=1}^{l} \boldsymbol{\phi}_m \boldsymbol{V}_{k+1-m} + \boldsymbol{\Delta}_{k+1} \\ &= \hat{\boldsymbol{V}}_{k+1} + \boldsymbol{\Delta}_{k+1},\end{aligned} \tag{8}$$

where l is the model order and $\boldsymbol{\Delta}_{k+1}$ denotes the prediction error with zero-mean white Gaussian distribution $\mathcal{N}(0, v_{k+1})$. Furthermore, based on the multi-step Levinson-Durbin algorithm, a super node can predict more velocities in future periods as

$$\begin{aligned}\boldsymbol{V}_{k+h} &= \sum_{m=1}^{l} \boldsymbol{\phi}_m^h \boldsymbol{V}_{k+1-m} + \boldsymbol{\Delta}_{k+h} \\ &= \hat{\boldsymbol{V}}_{k+h} + \boldsymbol{\Delta}_{k+h},\end{aligned} \tag{9}$$

where $h = 1, 2, \ldots, H$. This will result in a predicted velocity vector, $\hat{\boldsymbol{V}} = [\hat{\boldsymbol{V}}_{k+1}, \hat{\boldsymbol{V}}_{k+2}, \ldots, \hat{\boldsymbol{V}}_{k+H}]^{\mathrm{T}}$, with its prediction error covariance $\boldsymbol{v} = [v_{k+1}, v_{k+2}, \ldots, v_{k+H}]^{\mathrm{T}}$, and H is the length of the predicted velocity vector.

Given that super nodes float at different water depth and they have spatial correlations in the sensor mobility pattern, the predicted velocity vectors

of super nodes can be treated as velocity samples of the whole mobile network. These predicted velocity vectors can be regarded as references to assist ordinary nodes in localization processes.

Here, we make an explanation about why we substitute the multi-step Levinson-Durbin algorithm into the Levinson-Durbin algorithm. To decrease communication cost, a super node should deliver a relatively large predicted velocity vector in one broadcast. If we apply the Levinson-Durbin algorithm to calculate the predicted velocity vector, for example, when predicting $\boldsymbol{V}_{k+2}$ based on velocity samples $\boldsymbol{V}_k, \boldsymbol{V}_{k-1}, \ldots, \boldsymbol{V}_{k-(l-2)}$ and predicted velocity $\hat{\boldsymbol{V}}_{k+1}$ as the substitute (because they are unable to acquire velocity sample $\boldsymbol{V}_{k+1}$ at time k), the prediction error $\boldsymbol{\Delta}_{k+1}$ of $\hat{\boldsymbol{V}}_{k+1}$ will accumulate to prediction error $\boldsymbol{\Delta}_{k+2}$ of $\hat{\boldsymbol{V}}_{k+2}$, which is against our pursuit of high-accuracy predicted velocity vectors. The multi-step Levinson-Durbin algorithm calculates all elements of the predicted velocity vector based on common velocity samples, $\boldsymbol{V}_k, \boldsymbol{V}_{k-1}, \ldots, \boldsymbol{V}_{k-l-1}$, but different coefficients $\boldsymbol{\phi}^h$, $h = 1, 2, \ldots, H$. In addition, prediction errors $\boldsymbol{\Delta}_{k+1}, \boldsymbol{\Delta}_{k+2}, \ldots, \boldsymbol{\Delta}_{k+H}$ are mutually independent, which avoids accumulated errors.

4.2 Ordinary node prelocalization

Ordinary nodes with low-complexity hardware are unable to acquire location estimates directly from surface buoys. In our HLMP scheme, they achieve localization by predicted velocity vectors received from super nodes. Assume that ordinary node j receives N predicted velocity vectors from N super nodes. Then we can calculate the predicted velocity vector $\hat{\boldsymbol{V}}^j = [\hat{\boldsymbol{V}}_k^j]^{\mathrm{T}}$, $k = 1, 2, \ldots, H$ and its prediction error of ordinary node j, as follows:

$$\hat{\boldsymbol{V}}_k^j = \sum_{i=1}^{N} \mu_k^{ij} \hat{\boldsymbol{V}}_k^i, \tag{10}$$

$$v_k^j = \sum_{i=1}^{N} \mu_k^{ij} v_k^i, \tag{11}$$

where μ_k^{ij} denotes the interpolation coefficient between nodes i and j, calculated as

$$\mu_k^{ij} = \frac{(d_{ij} v_k^i)^{-1}}{\sum_{i=1}^{N} (d_{ij} v_k^i)^{-1}}, \tag{12}$$

where d_{ij} denotes the horizontal distance between super node i and ordinary node j, and v_k^i is the variance of the prediction error of $\hat{\boldsymbol{V}}_k^i$. Eq. (12) tells that the super node which is closer to ordinary node j at a horizontal distance or has a more accurate predicted velocity vector makes more contributions in predicting the velocity vector of ordinary node j. This equation makes sense. First, we take advantages of spatial correlations of underwater objects to facilitate mobility prediction. Since Eq. (1) indicates that sensors with the same (x, y) but different z (depth) move in the same mobility pattern, we consider only the horizontal distances between super nodes and ordinary nodes. Second, different predicted velocity vectors received from super nodes have different prediction errors. Specifically, some prediction errors may be very large if the linear prediction equation (Eq. (9)) is ill-conditioned. This factor should also be considered in determining velocity vectors of ordinary nodes. We prefer more accurate predicted velocity vectors by including the prediction accuracy in Eq. (12).

Fig. 4 shows the prelocalization process of an ordinary node. Ordinary nodes do prelocalization based on their predicted velocity vectors in prediction windows. Ordinary nodes receive broadcast packets from super nodes at prediction points, and then calculate their predicted velocity vectors in the next prediction window (Eq. (10)). Moreover, ordinary node j can measure Z_{ij}, the distance to super node i, using the well-known time-of-arrival (ToA) measurement as

$$Z_{ij} = h(\boldsymbol{L}^i, \boldsymbol{L}^j) + \nu_{ij}, \tag{13}$$

$$\begin{aligned} h(\boldsymbol{L}^i, \boldsymbol{L}^j) =& ((x_i - x_j)^2 + (y_i - y_j)^2 \\ &+ (z_i - z_j)^2)^{1/2}, \end{aligned} \tag{14}$$

where $h(\boldsymbol{L}^i, \boldsymbol{L}^j)$ denotes the real distance between nodes j and i, (x_i, y_i, z_i) and (x_j, y_j, z_j) are Cartesian coordinates of nodes i and j respectively, and ν_{ij} denotes the measurement error with zero-mean

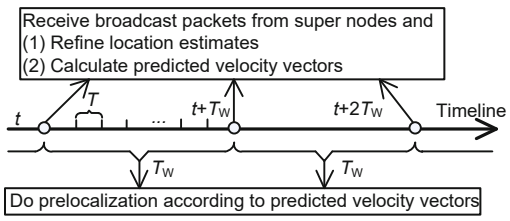

Fig. 4 Prelocalization process of an ordinary node ($\boldsymbol{T}$: localization period; $\boldsymbol{T_w}$: length of the prediction window; $\boldsymbol{t}$, $\boldsymbol{t + T_w}$, and $\boldsymbol{t + 2T_w}$: prediction points)

white Gaussian distribution $\mathcal{N}(0, R_{ij})$. Ordinary node location estimates will gradually deviate from true values if relying on only predicted velocity vectors. These measurements in Eq. (13) should be efficiently used to refine location estimates of ordinary nodes. Assume that the location estimate of ordinary node j is $\hat{\boldsymbol{L}}^j$ and that its distribution $p(\boldsymbol{L}^j)$ obeys $\mathcal{N}(\hat{\boldsymbol{L}}^j, \boldsymbol{\eta}^j)$. The refinement operation is realized by fusion of an a priori estimate and those measurements as

$$\begin{aligned} &p(\boldsymbol{L}^j_{\text{refined}}|\boldsymbol{L}^j, Z_{1j}, Z_{2j}, \ldots, Z_{Nj}) \\ &\propto p(\boldsymbol{L}^j)\prod_{i=1}^{N} p(Z_{ij}|\boldsymbol{L}^j), \end{aligned} \tag{15}$$

where N is the number of measurements. On account of the nonlinear measurement model in Eq. (13), the solution to Eq. (15) should be nonlinear filters, such as particle filter (PF) (Del Moral, 1997) and cubature Kalman filter (CKF)(Arasaratnam and Haykin, 2009). In this study, we apply the new CKF algorithm to deal with this nonlinear filtering problem. Assume that we have already acquired the refined location $\boldsymbol{L}^j_{\text{refined},i} \sim \mathcal{N}(\hat{\boldsymbol{L}}^j_{\text{refined},i}, \boldsymbol{P}^j_{\text{refined},i})$ of ordinary node j and its predicted velocity vector $\hat{\boldsymbol{V}}^j = [\hat{\boldsymbol{V}}^j_{i+1}, \hat{\boldsymbol{V}}^j_{i+2}, \ldots, \hat{\boldsymbol{V}}^j_{i+H}]$ with the corresponding prediction error covariance $\boldsymbol{v}^j = [v^j_{i+1}, v^j_{i+2}, \ldots, v^j_{i+H}]$. Then the prelocalization processes of ordinary node j in the next prediction window are as follows:

$$\begin{aligned} \boldsymbol{L}^j_{i+1} &= \hat{\boldsymbol{L}}^j_{\text{refined},i} + \boldsymbol{T}\hat{\boldsymbol{V}}^j_{i+1} + \boldsymbol{\psi}^j_{i+1} \\ &= \hat{\boldsymbol{L}}^j_{i+1} + \boldsymbol{\psi}^j_{i+1}, \end{aligned} \tag{16}$$

$$\boldsymbol{P}^j_{i+1} = \boldsymbol{P}^j_{\text{refined},i} + \boldsymbol{T}\boldsymbol{v}^j_{i+1}\boldsymbol{T}^{\mathrm{T}}, \tag{17}$$

$$\begin{aligned} \boldsymbol{L}^j_{i+h} &= \hat{\boldsymbol{L}}^j_{i+h-1} + \boldsymbol{T}\hat{\boldsymbol{V}}^j_{i+h} + \boldsymbol{\psi}^j_{i+h} \\ &= \hat{\boldsymbol{L}}^j_{i+h} + \boldsymbol{\psi}^j_{i+h}, \end{aligned} \tag{18}$$

$$\boldsymbol{P}^j_{i+k} = \boldsymbol{P}^j_{i+h-1} + \boldsymbol{T}\boldsymbol{v}^j_{i+h}\boldsymbol{T}^{\mathrm{T}}, \tag{19}$$

where $h = 2, 3, \ldots, H$, $\boldsymbol{\psi}^j_{i+1} \sim \mathcal{N}(0, \boldsymbol{P}^j_{i+1})$, and $\boldsymbol{\psi}^j_{i+h} \sim \mathcal{N}(0, \boldsymbol{P}^j_{i+h})$. Through the above equations, we obtain the prelocalization location estimates $\hat{\boldsymbol{L}}^j_{i+1}, \hat{\boldsymbol{L}}^j_{i+2}, \ldots, \hat{\boldsymbol{L}}^j_{i+H}$ with the corresponding covariance $\boldsymbol{P}^j_{i+1}, \boldsymbol{P}^j_{i+2}, \ldots, \boldsymbol{P}^j_{i+H}$ of sensor j in the next prediction window. The pseudo code of our HLMP algorithm is summarized in Algorithm 1.

Algorithm 1 High-accuracy localization with mobility prediction (HLMP) algorithm

1: Super nodes record velocity samples in each sampling period using Eq. (7).
2: **if** at the prediction points $t, t+T_{\text{w}}, \ldots, t+NT_{\text{w}}$ **then**
3: Super nodes calculate predicted velocity vectors $\hat{\boldsymbol{V}}^{\text{super}}$ using the multi-step Levinson-Durbin algorithm.
4: Super nodes broadcast their predicted velocity vectors.
5: Ordinary nodes receive predicted velocity vectors and acquire range measurements $Z_{\text{super,ordinary}}$ based on ToA measurements.
6: Ordinary nodes refine their position estimates base on the a priori estimate $\hat{\boldsymbol{L}}^{\text{ordinary}}$ and $Z_{\text{super,ordinary}}$. Then calculate $\hat{\boldsymbol{L}}^{\text{ordinary}}_{\text{refined}}$ and the corresponding covariance $\boldsymbol{P}^{\text{ordinary}}_{\text{refined}}$ using the CKF algorithm and fusion algorithm as Eqs. (22) and (23).
7: Ordinary nodes calculate their predicted velocity vectors $\hat{\boldsymbol{V}}^{\text{ordinary}}$ as Eq. (10).
8: **for** $h = 1, 2, \ldots, H$ **do**
9: Ordinary nodes estimate prelocalization position $\hat{\boldsymbol{L}}^{\text{ordinary}}_h$ with the corresponding covariance $\boldsymbol{P}^{\text{ordinary}}_h$ in the next prediction window as Eqs. (16)–(19).
10: **end for**
11: **end if**

5 Simultaneous localization and target tracking

With our HLMP scheme, ordinary nodes can obtain relatively precise location estimates with the help of super nodes. With these location estimates, ordinary nodes are qualified to accomplish target tracking tasks. Furthermore, in this section, we will improve localization accuracy during the process of target tracking using our SLAT algorithm. In general target tracking cases, we focus on modeling more practical target motions and acquiring more accurate target state estimates based on sensor measurements. We pay less attention to location estimates of sensors as they are immobile with minor localization errors and have less influence on target tracking. However, in the case of target tracking for UMSNs, sensor locations change continuously due to environmental forces such as tidal current. Sensor location estimates are usually coarse and seriously affect target tracking accuracy. Consider the measurement

model described by

$$Z_k^i = h(\boldsymbol{X}_k, \boldsymbol{L}_k^i) + \nu_k^i, \tag{20}$$

$$h(\boldsymbol{X}_k, \boldsymbol{L}_k^i) = \big((x_k - x_i)^2 + (y_k - y_i)^2 + (z_k - z_i)^2\big)^{1/2}, \tag{21}$$

where $h(\boldsymbol{X}_k, \boldsymbol{L}_k^i)$ denotes the real distance between the target and node i, (x_k, y_k, z_k) and (x_i, y_i, z_i) are Cartesian coordinates of the target and node i respectively, and ν_k^i denotes the measurement error with zero-mean white Gaussian distribution $\mathcal{N}(0, R_k^i)$. Note that the measurement in Eq. (20) contains location information of not only the target but also the sensor. So, it is certainly a feasibility to update sensor location estimates by measurements. The core idea of our SLAT algorithm is to take full advantage of information carried by measurements to update the target state and sensor location estimates. So, our SLAT algorithm contains two parts, target state update and sensor location update.

1. Target state update

Assume that we obtain the target state estimate $\hat{\boldsymbol{X}}_{k-1|k-1}$ with the corresponding covariance $\boldsymbol{P}_{k-1|k-1}^t$, sensor location estimates $\hat{\boldsymbol{L}}_k^j$ (replaced by $\hat{\boldsymbol{L}}_{\text{updated},k}^j$ if known), and measurements Z_k^j $(j = 1, 2, \ldots, n)$ at time k (n denotes the number of sensors participating in target tracking tasks). Then substitute the above values into CKF and generate local target state estimates $\hat{\boldsymbol{X}}_{k|k}^j$ with covariance $\boldsymbol{P}_{k|k}^{t,j}$. At last, local estimates will be sent to the fusion center (usually select one from those task sensors) to obtain the fusion target state estimate with covariance as (Zhu et al., 2001)

$$\boldsymbol{P}_{k|k}^{t^{-1}} = \boldsymbol{P}_{k|k-1}^{t^{-1}} + \sum_{i=1}^{n} (\boldsymbol{P}_{k|k}^{t,j^{-1}} - \boldsymbol{P}_{k|k-1}^{t^{-1}}), \tag{22}$$

$$\boldsymbol{P}_{k|k}^{t^{-1}} \hat{\boldsymbol{X}}_{k|k} = \boldsymbol{P}_{k|k-1}^{t^{-1}} \hat{\boldsymbol{X}}_{k|k-1} + \sum_{i=1}^{n} (\boldsymbol{P}_{k|k}^{t,j^{-1}} \hat{\boldsymbol{X}}_{k|k}^j - \boldsymbol{P}_{k|k-1}^{t^{-1}} \hat{\boldsymbol{X}}_{k|k-1}), \tag{23}$$

where $\hat{\boldsymbol{X}}_{k|k-1}$ and $\boldsymbol{P}_{k|k-1}^t$ are calculated using the CKF algorithm. The fusion target state estimate will be fed back to the task sensor to update location estimate.

2. Sensor location update

Assume that we obtain the location estimate $\hat{\boldsymbol{L}}_k^j$ (replaced by $\hat{\boldsymbol{L}}_{\text{updated},k}^j$, if known) with the corresponding covariance $\boldsymbol{P}_k^j$ (replaced by $\boldsymbol{P}_{\text{updated},k}^j$, if known), fusion target state estimate $\hat{\boldsymbol{X}}_{k|k}$, and measurements Z_k^j $(j = 1, 2, \ldots, n)$ at time k. Then take the sensor position as the state variable and substitute the above values into the CKF algorithm, which results in updated sensor location estimate $\hat{\boldsymbol{L}}_{\text{updated},k}^{\star,j}$ with covariance $\boldsymbol{P}_{\text{updated},k}^{\star,j}$. Finally, based on $\hat{\boldsymbol{L}}_{\text{updated},k}^{\star,j}$ and $\boldsymbol{P}_{\text{updated},k}^{\star,j}$, we update prelocalization location estimates in Eqs.(16)–(19) as

$$\hat{\boldsymbol{L}}_{\text{updated},k+1}^j = \hat{\boldsymbol{L}}_{\text{updated},k}^{\star,j} + \boldsymbol{T}\hat{\boldsymbol{V}}_{k+1}^j, \tag{24}$$

$$\boldsymbol{P}_{\text{updated},k+1}^j = \boldsymbol{P}_{\text{updated},k}^{\star,j} + \boldsymbol{T}\boldsymbol{v}_{k+1}^j \boldsymbol{T}^{\mathrm{T}}, \tag{25}$$

$$\hat{\boldsymbol{L}}_{\text{updated},k+h}^j = \hat{\boldsymbol{L}}_{\text{updated},k+h-1}^j + \boldsymbol{T}\hat{\boldsymbol{V}}_{k+h}^j, \tag{26}$$

$$\boldsymbol{P}_{\text{updated},k+h}^j = \boldsymbol{P}_{\text{updated},k+h-1}^j + \boldsymbol{T}\boldsymbol{v}_{k+h}^j \boldsymbol{T}^{\mathrm{T}}, \tag{27}$$

where $h = 2, 3, \ldots, i + H - k$. By our SLAT algorithm, sensor location estimates can be revised continuously during the process of target tracking. For clarity, the pseudo code of our SLAT algorithm in one prediction window is summarized in Algorithm 2.

6 Simulation results

6.1 Simulation scenario

We apply our HLMP and SLAT algorithms to a target tracking example in UMSNs for verification and appraisal. The target is assumed to move in a 3D underwater environment. Initially, super nodes are deployed as a $4 \times 4 \times 4$ uniform grid and 343 ordinary nodes are deployed randomly in a 1000 m $\times$ 1000 m $\times$ 1000 m region. The deployment error obeys normal distribution $\mathcal{N}(0, 5)$. For the sensor mobility model, related random variables and distributions are summarized in Table 1. The sampling interval T is 1 s and the length of prediction window H is set to 30 s. The N in Eq. (6) is 50 and the l in Eq. (9) is 10. The detection radius and the communication radius of sensors are assumed to be 200 m and 350 m, respectively.

Target motion is assumed to be maneuvering. The initial state of the target is assumed to be $\boldsymbol{X}_0 =$

Table 1 Random variables in the sensor mobility model

Variable	Distribution
k_1, k_2	$\mathcal{N}(\pi, (0.1\pi)^2)$
k_3	$\mathcal{N}(2\pi, (0.2\pi)^2)$
k_4, k_5, v	$\mathcal{N}(1, 0.1^2)$
λ	$\mathcal{N}(3, 0.3^2)$

Algorithm 2 Simultaneous localization and target tracking (SLAT) algorithm in one prediction window

1: Ordinary nodes, in one prediction window $k = 1, 2, \ldots, H$, do the following:
2: Assume that we know the target state estimate $\hat{\boldsymbol{X}}_{0|0}$ with the corresponding covariance $\boldsymbol{P}^{t}_{0|0}$ and location estimates $\hat{\boldsymbol{L}}^{\text{ordinary}}_{1,2,\ldots,H}$ with the corresponding covariance $\boldsymbol{P}^{\text{ordinary}}_{1,2,\ldots,H}$.
3: **for** $k = 1, 2, \ldots, H$ **do**
4: **for** $j = 1, 2, \ldots, n$ **do**
5: Ordinary nodes which detect the target become task-nodes and obtain range measurements Z^{j}_{k}, as Eq. (20).
6: Task-nodes update the local target state estimate $\hat{\boldsymbol{X}}^{j}_{k|k}$ and the corresponding covariance $\boldsymbol{P}^{t,j}_{k|k}$ using the CKF algorithm.
7: **end for**
8: The fusion center receives $\hat{\boldsymbol{X}}^{j}_{k|k}$ and $\boldsymbol{P}^{t,j}_{k|k}$, and then updates fusion state estimate $\hat{\boldsymbol{X}}_{k|k}$ and the corresponding covariance $\boldsymbol{P}^{t}_{k|k}$ using Eqs. (22) and (23).
9: **for** $j = 1, 2, \ldots, n$ **do**
10: Task-nodes receive $\hat{\boldsymbol{X}}_{k|k}$, and then with the help of Z^{j}_{k} update location estimates $\hat{\boldsymbol{L}}^{\star,j}_{\text{updated},k}$ and the corresponding covariance $\boldsymbol{P}^{\star,j}_{\text{updated},k}$ using the CKF algorithm.
11: **end for**
12: **for** $h = 1, 2, \ldots, H-k$ **do**
13: Task-nodes update their future location estimates $\hat{\boldsymbol{L}}^{j}_{\text{updated},k+h}$ and the corresponding covariance $\boldsymbol{P}^{j}_{\text{updated},k+h}$ as Eqs. (24)–(27).
14: Task-nodes substitute $\hat{\boldsymbol{L}}^{j}_{\text{updated},k+h}$ and $\boldsymbol{P}^{j}_{\text{updated},k+h}$ into $\hat{\boldsymbol{L}}^{\text{ordinary}}_{1,2,\ldots,H}$ and $\boldsymbol{P}^{\text{ordinary}}_{1,2,\ldots,H}$ respectively.
15: **end for**
16: **end for**

$[x_0, \dot{x}_0, y_0, \dot{y}_0, z_0, \dot{z}_0]^{\mathrm{T}} = [470, 7, 280, 5, 20, 4]^{\mathrm{T}}$, where (x_0, y_0, z_0) is the initial position of the target and $(\dot{x}_0, \dot{y}_0, \dot{z}_0)$ the corresponding velocity. From 1 s to 94 s, it moves at a constant velocity (CV). From 95 s to 184 s, it makes a coordinate turn (CT) with turn rate 0.035 rad/s. From 185 s to 200 s, it moves at a CV. CV and CT can be formulated as

$$\boldsymbol{X}_k = \boldsymbol{F}^{\mathrm{CV}} \boldsymbol{X}_{k-1} + \boldsymbol{w}_k, \tag{28}$$

$$\boldsymbol{X}_k = \boldsymbol{F}^{\mathrm{CT}} \boldsymbol{X}_{k-1} + \boldsymbol{w}_k, \tag{29}$$

where $\boldsymbol{F}^{\mathrm{CV}}$ and $\boldsymbol{F}^{\mathrm{CT}}$ are state transition matrices and $\boldsymbol{w}_k$ is the process noise with zero-mean white Gaussian distribution $\mathcal{N}(0, \boldsymbol{Q}_k)$. $\boldsymbol{F}^{\mathrm{CV}}$, $\boldsymbol{F}^{\mathrm{CT}}$, and $\boldsymbol{Q}_k$ are given as follows:

$$\boldsymbol{F}^{\mathrm{CV}} = \begin{bmatrix} 1 & T & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{30}$$

$$\boldsymbol{F}^{\mathrm{CT}} = \begin{bmatrix} 1 & \dfrac{\sin(wT)}{w} & 0 & \dfrac{\cos(wT)-1}{w} & 0 & 0 \\ 0 & \cos(wT) & 0 & -\sin(wT) & 0 & 0 \\ 0 & \dfrac{1-\cos(wT)}{w} & 1 & \dfrac{\sin(wT)}{w} & 0 & 0 \\ 0 & \sin(wT) & 0 & \cos(wT) & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{31}$$

$$\boldsymbol{Q}_k = q^2 \begin{bmatrix} \dfrac{T^3}{3} & \dfrac{T^2}{2} & 0 & 0 & 0 & 0 \\ \dfrac{T^2}{2} & T & 0 & 0 & 0 & 0 \\ 0 & 0 & \dfrac{T^3}{3} & \dfrac{T^2}{2} & 0 & 0 \\ 0 & 0 & \dfrac{T^2}{2} & T & 0 & 0 \\ 0 & 0 & 0 & 0 & \dfrac{T^3}{3} & \dfrac{T^2}{2} \\ 0 & 0 & 0 & 0 & \dfrac{T^2}{2} & T \end{bmatrix}, \tag{32}$$

where q is the intensity of the process noise. The measurement error ν obeys $\mathcal{N}(0, 5)$.

6.2 Performance verification

6.2.1 Performance of localization and target tracking

The simulation results are generated from 100 Monte-Carlo runs. We adopt root-mean-square error (RMSE) to assess the accuracy of localization and target tracking. In our simulation, we compare performances of three localization and target tracking schemes. The first one labeled by 'Reference' predicts velocity vectors based on the one-step Levinson-Durbin algorithm without precision control as in Zhou et al. (2011). In addition, it cannot update sensor location estimates during the process of target tracking. The second one labeled by 'HLMP and tracking' uses our HLMP algorithm for prelocalization but does not update sensor location estimates during the process of target tracking. The last one labeled by 'HLMP-SLAT' not only applies our HLMP algorithm for prelocalization but also updates the sensor location using our SLAT algorithm.

The real trajectory and target tracking performances of the three localization and target tracking schemes are shown in Fig. 5. It is clear that all the three schemes can successfully track the target, due to relatively accurate sensor position estimates generated from their localization algorithms, which is crucial for UMSNs. Fig. 6 lists the localization errors of the three schemes. The accuracy of the Reference scheme is significantly lower than those of others. Because in the Reference scheme, super nodes experience accumulated error when predicting velocity estimates using the one-step Levinson-Durbin algorithm. They broadcast only velocity estimates without their precisions, so ordinary nodes calculate interpolation coefficients based on only the distances to super nodes. The other two using our HLMP algorithm for localization manifesting high-precision certify the superiority of our algorithm. What is more, the HLMP-SLAT scheme acquiring the highest localization accuracy proves that our SLAT algorithm can surely further increase the localization accuracy during the process of target tracking. Obviously, improvements of localization will improve target tracking accuracy (Fig. 7). Fig. 6 shows that sensor localization errors will gradually rise if sensor location estimates rely on only prediction. The refined points marked in Fig. 6 represent the refinement operation (Eq. (15)) that ordinary nodes do at prediction points. The results show that the refinement operation based on range measurement between a super node and an ordinary node can decrease localization deviation efficiently when the localization error is significantly large. The refinement operation makes our HLMP algorithm more robust. Fig. 8 gives a quantitative comparison of the three schemes. Comparing the Reference scheme with the HLMP and tracking scheme, our HLMP algorithm improves localization accuracy by 69.26% and tracking accuracy by 65.74%. Comparing the HLMP and tracking scheme with the HLMP-SLAT scheme, our SLAT algorithm improves localization accuracy by 10.19% and tracking accuracy by 9.01%.

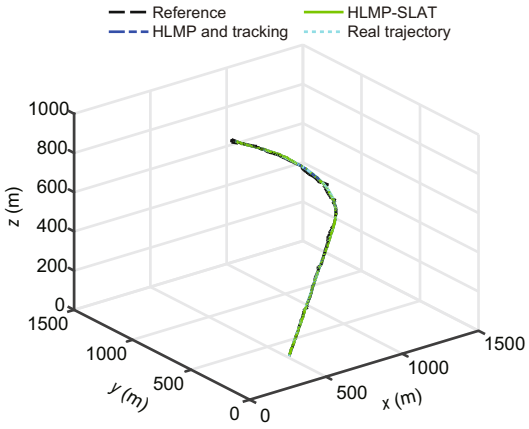

Fig. 5 Target tracking performances and the real trajectory

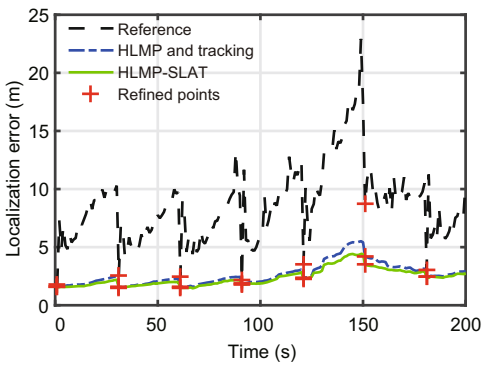

Fig. 6 Localization errors of the three schemes

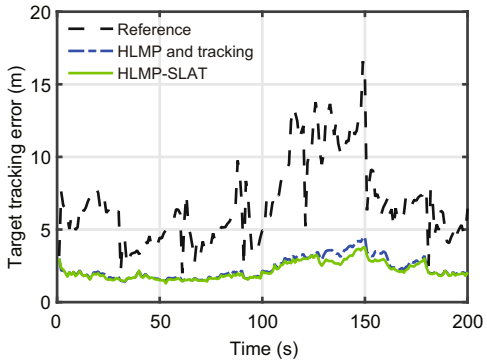

Fig. 7 Target tracking errors of the three schemes

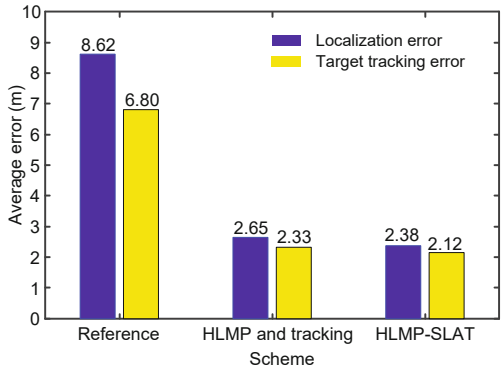

Fig. 8 Average localization errors and target tracking errors of the three schemes

6.2.2 Energy efficiency analysis

In this study, we focus on designing an energy-efficient localization scheme to address the target tracking problem in underwater wireless sensor networks. The energy model in Bhardwaj and Chandrakasan (2002) showed that the communication energy cost dominates the whole energy consumption compared with sensing and data processing. Since task-nodes during the process of target tracking in different schemes are the same, we cannot compare their energy cost in this process. Instead, we compare their communication cost in the process of localization. Since the accuracy of the Reference scheme is significantly lower than that of our HLMP-SLAT scheme, we introduce a rebroadcast mechanism for Reference to improve its accuracy. For super node i, if its prediction error is larger than a certain threshold, i.e.,

$$\sqrt{(\boldsymbol{L}^i-\hat{\boldsymbol{L}}^i)^{\mathrm{T}}(\boldsymbol{L}^i-\hat{\boldsymbol{L}}^i)}\geq D, \tag{33}$$

then super node i repeats the prediction algorithm and broadcasts new velocity vectors $\hat{\boldsymbol{V}}^i_{\mathrm{update}}$, where $\boldsymbol{L}^i$ and $\hat{\boldsymbol{L}}^i$ are current location and location estimate of super node i respectively, and D is a threshold. We fix the length of a prediction window as $H=50$ and change the threshold as $D\in\{2,4,7,10\}$. We count the total broadcast times of super nodes to compare the energy efficiency of different schemes. The performances under different D and our HLMP-SLAT scheme are listed in Table 2. With the decrease of rebroadcast threshold D, the localization performance of the Reference scheme becomes better. When $D=2$, the performance of the Reference scheme is close to that of our HLMP-SLAT scheme. However, the price is 989 broadcast times and our scheme needs only 192 broadcast times. This means that our HLMP-SLAT scheme saves 80.59% energy compared to the Reference scheme.

Table 2 Performances of the Reference (under different threshold D) and HLMP-SLAT schemes

Scheme	Localization error (m)	Broadcast times
Reference ($D=10$)	6.59	769
Reference ($D=7$)	5.11	816
Reference ($D=4$)	3.73	878
Reference ($D=2$)	2.76	989
HLMP-SLAT	2.62	192

6.3 Impacts of parameters

In the previous subsection, we have demonstrated the superiority of our HLMP-SLAT scheme. To study the characteristics of this scheme, we will analyze the impacts of parameters through further simulations in this subsection.

6.3.1 Performances of different H

Fig. 9 shows the localization errors of the HLMP and tracking scheme under different prediction window length H. Obviously, the smaller the H, the better localization performance we can have. This is because prediction window length H represents the minimum interval for ordinary nodes to receive new predicted velocity vectors from super nodes. Ordinary nodes need to do localization relying on only the last received predicted velocity vectors when they are waiting for new packets. This kind of localization has a problem of error accumulation, which means that the localization error will increase as the prediction window lengthens. Even though we can shorten the prediction window to obtain more accurate localization results, it is not advisable with the consideration of additional energy costs of communication. A better solution is presented in Fig. 10. It tells that our HLMP-SLAT scheme can weaken the influence of different H by revising location estimates during the process of target tracking. For the purpose of quantitative comparison, Table 3 displays the average localization errors and target tracking errors. The performance of the HLMP-SLAT scheme with $H=50$ is similar to that of the HLMP and tracking scheme with $H=30$, which means our HLMP-SLAT scheme can save almost 40% communication energy of super nodes under the same accuracy requirement.

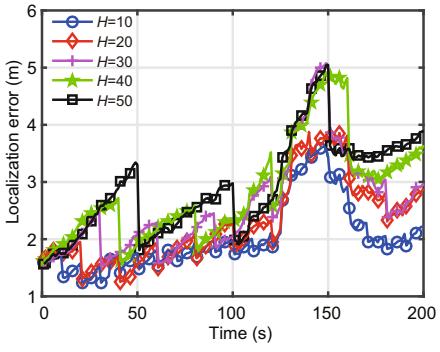

Fig. 9 Localization errors of the HLMP and tracking scheme under different prediction window length H

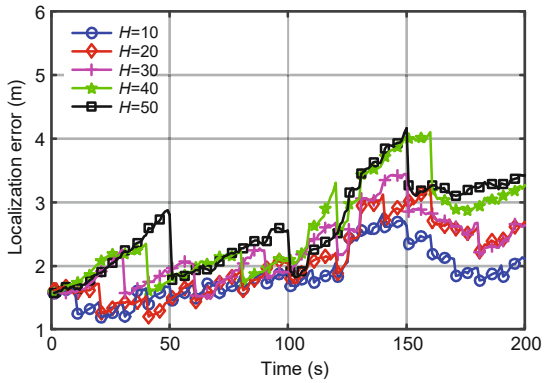

Fig. 10 Localization errors of the HLMP-SLAT scheme under different prediction window length *H*

Table 3 Average localization errors and target tracking errors under different prediction window length *H*

H	Localization error (m)		Target tracking error (m)	
	HLMP	HLMP-SLAT	HLMP	HLMP-SLAT
10	2.03	1.85	1.99	1.89
20	2.30	2.07	2.15	2.01
30	2.65	2.38	2.33	2.12
40	2.84	2.55	2.44	2.26
50	2.92	2.62	2.50	2.31

6.3.2 Performances of different N

The data sequence length N in Eq. (6) represents the amount of stored velocity data used to obtain AR model samples $\gamma(i)$. We fix the length of the prediction window as $H = 30$ and change the data sequence length as $N \in \{30, 50, 70, 90, 110\}$. Figs. 11 and 12 show the localization errors and target tracking errors of the HLMP-SLAT scheme under different data sequence length N, respectively. When N is small, taking $N = 30$ as an example, both the localization error and target tracking error are large. The localization and target tracking performances will be better with more velocity data used, which can be proved by $N = 50$, 70, and 90. However, when N is large enough, the increase of N has less influence on localization and target tracking performances, which can be proved by the comparison of $N = 90$ with $N = 110$. The quantitative results are listed in Table 4, implying that we should set the value of N to an optimal one given consideration to the memory size of devices.

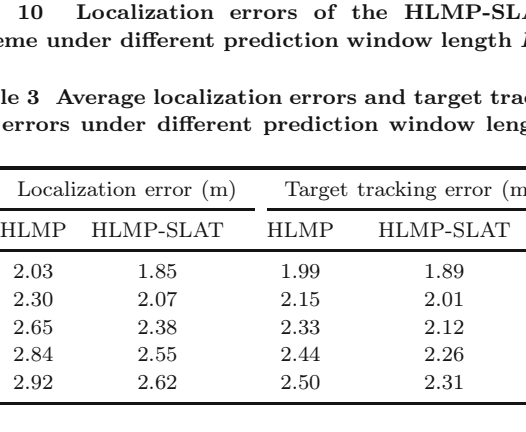

Fig. 11 Localization errors of the HLMP-SLAT scheme under different data sequence length *N*

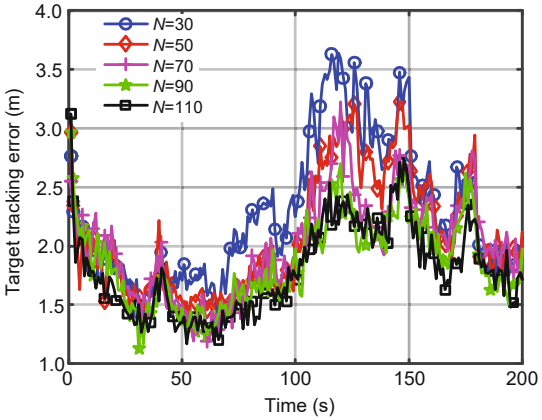

Fig. 12 Target tracking errors of the HLMP-SLAT scheme under different data sequence length *N*

Table 4 Performances of HLMP-SLAT under different data sequence length *N*

N	Localization error (m)	Tracking error (m)
30	2.57	2.27
50	2.38	2.12
70	2.10	2.00
90	1.88	1.86
110	1.78	1.79

6.3.3 Performances of different l

The model order l in Eq. (9) represents the number of velocity samples used to calculate the predicted velocity by a linear combination. We fix the length of the prediction window and the data sequence as $H = 30$ and $N = 50$, respectively. Then we change model order as $l \in \{2, 3, 4, 7, 10\}$. Figs. 13 and 14 show the localization errors and target tracking errors of the HLMP-SLAT scheme under different model order l, respectively. When $l = 10$, both the localization error and target tracking error are significantly larger than others. The localization and

target tracking performances with $l \in \{2, 3, 4, 7\}$ are very similar. For clarity, the average localization and target tracking errors are given in Table 5. When $l = 3$, we can obtain the best localization and target tracking performances. The performances become worse when l is significantly far away from 3, taking $l = 10$ as an example. This is due to the duality of increasing model order l. On the one hand, more model samples can provide more information about model characteristics; on the other hand, adding model samples will bring their inherent errors to the prediction result. So, the model sample length l should not be set too large.

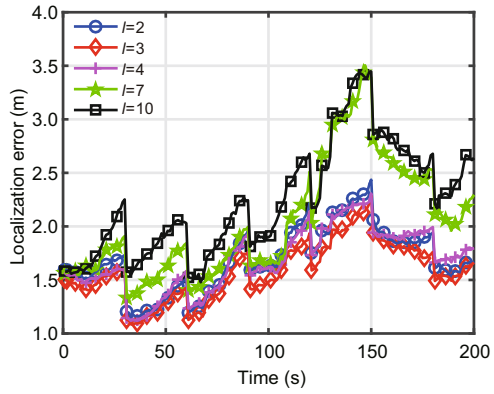

Fig. 13 Localization errors of the HLMP-SLAT scheme under different model order l

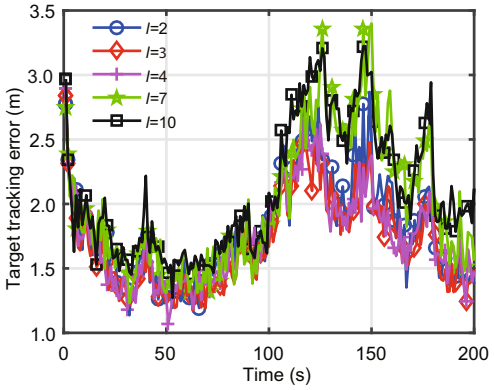

Fig. 14 Target tracking errors of the HLMP-SLAT scheme under different model order l

7 Conclusions

In this paper, we have presented an HLMP-SLAT scheme to solve the target tracking problem for UMSNs. Our HLMP-SLAT scheme contains two parts: HLMP algorithm and SLAT algorithm. Since sensor locations change continuously in UMSNs, which usually results in a large localization error, the HLMP algorithm gives a high-precision and energy-efficient solution to the localization problem with the help of sensor mobility prediction and the multi-step Levinson-Durbin algorithm. Moreover, our SLAT algorithm makes full use of measurement information to update sensor location estimates; the update of localization information feeds back to improve the target tracking accuracy. Compared to existing dynamic SLAT algorithms, an ordinary node uses only its own measurement without additional communication energy cost and figures out results fast without iteration operations. These advantages meet the energy saving and device simplifying requirements of ordinary nodes. Simulation results showed that our HLMP-SLAT scheme can fulfill the target tracking mission in UMSNs with higher localization and target tracking accuracies and save more energy than the schemes for reference. Besides, we have analyzed the influence of different parameters by carrying out more simulations, certifying the superiority of our HLMP-SLAT scheme ulteriorly.

Table 5 Performances of HLMP-SLAT under different model order l

l	Localization error (m)	Tracking error (m)
2	1.69	1.79
3	1.58	1.70
4	1.68	1.74
7	2.05	2.00
10	2.38	2.12

Acknowledgements

We would like to thank Professor Qun-fei ZHANG from School of Marine Science and Technology, Northwestern Polytechnical University, for his advice on simulation design.

References

Aggarwal P, Wang X, 2011. Joint sensor localisation and target tracking in sensor networks. *IET Radar Sonar Navig*, 5(3):225-233. https://doi.org/10.1049/iet-rsn.2010.0118

Arasaratnam I, Haykin S, 2009. Cubature Kalman filters. *IEEE Trans Autom Contr*, 54(6):1254-1269. https://doi.org/10.1109/TAC.2009.2019800

Austin TC, Stokey RP, Sharp KM, 2000. Paradigm: a buoy-based system for AUV navigation and tracking.

OCEANS MTS/IEEE Conf and Exhibition, p.935-938. https://doi.org/10.1109/OCEANS.2000.881376

Beerens SP, Ridderinkhof H, Zimmerman JTF, 1994. An analytical study of chaotic stirring in tidal areas. *Chaos Sol Fract*, 4(6):1011-1029. https://doi.org/10.1016/0960-0779(94)90136-8

Bhardwaj M, Chandrakasan AP, 2002. Bounding the lifetime of sensor networks via optimal role assignments. 21st Annual Joint Conf of the IEEE Computer and Communications Societies, p.1587-1596. https://doi.org/10.1109/INFCOM.2002.1019410

Brockwell PJ, Dahlhaus R, 2004. Generalized Levinson-Durbin and Burg algorithms. *J Econom*, 118(1-2):129-149. https://doi.org/10.1016/S0304-4076(03)00138-6

Calafate CT, Lino C, Diaz-Ramirez A, et al., 2013. An integral model for target tracking based on the use of a WSN. *Sensors*, 13(6):7250-7278. https://doi.org/10.3390/s130607250

Chen HY, Zhang SL, Liu MQ, et al., 2017. An artificial measurements-based adaptive filter for energy-efficient target tracking via underwater wireless sensor networks. *Sensors*, 17(5):971. https://doi.org/10.3390/s17050971

Cui JH, Kong JJ, Gerla M, et al., 2006. The challenges of building mobile underwater wireless networks for aquatic applications. *IEEE Netw*, 20(3):12-18. https://doi.org/10.1109/MNET.2006.1637927

Del Moral P, 1997. Nonlinear filering: interacting particle resolution. *Compt Rend Acad Sci Ser I-Math*, 325(6):653-658. https://doi.org/10.1016/S0764-4442(97)84778-7

Guo Y, Liu YT, 2013. Localization for anchor-free underwater sensor networks. *Comput Electr Eng*, 39(6):1812-1821. https://doi.org/10.1016/j.compeleceng.2013.02.001

Isbitiren G, Akan OB, 2011. Three-dimensional underwater target tracking with acoustic sensor networks. *IEEE Trans Veh Technol*, 60(8):3897-3906. https://doi.org/10.1109/TVT.2011.2163538

Kantas N, Singh SS, Doucet A, 2012. Distributed maximum likelihood for simultaneous self-localization and tracking in sensor networks. *IEEE Trans Signal Process*, 60(10):5038-5047. https://doi.org/10.1109/TSP.2012.2205923

Kim S, Yoo Y, 2013. High-precision and practical localization using seawater movement pattern and filters in underwater wireless networks. IEEE 16th Int Conf on Computational Science and Engineering, p.374-381. https://doi.org/10.1109/CSE.2013.64

Kussat NH, Chadwell CD, Zimmerman R, 2005. Absolute positioning of an autonomous underwater vehicle using GPS and acoustic measurements. *IEEE J Ocean Eng*, 30(1):153-164. https://doi.org/10.1109/JOE.2004.835249

Li WL, Jia YM, Du JP, et al., 2013. Distributed multiple-model estimation for simultaneous localization and tracking with NOLS mitigation. *IEEE Trans Veh Technol*, 62(6):2824-2830. https://doi.org/10.1109/TVT.2013.2247073

Lloret J, 2013. Underwater sensor nodes and networks. *Sensors*, 13(9):11782-11796. https://doi.org/10.3390/s130911782

Mandal AK, Misra S, Ojha T, et al., 2017. Oceanic forces and their impact on the performance of mobile underwater acoustic sensor networks. *Int J Commun Syst*, 30(1):e2882. https://doi.org/10.1002/dac.2882

Pardey J, Roberts S, Tarassenko L, 1996. A review of parametric modelling techniques for EEG analysis. *Med Eng Phys*, 18(1):2-11. https://doi.org/10.1016/1350-4533(95)00024-0

Sozer EM, Stojanovic M, Proakis JG, 2000. Underwater acoustic networks. *IEEE J Ocean Eng*, 25(1):72-83. https://doi.org/10.1109/48.820738

Teng J, Snoussi H, Richard C, et al., 2012. Distributed variational filtering for simultaneous sensor localization and target tracking in wireless sensor networks. *IEEE Trans Veh Technol*, 61(5):2305-2318. https://doi.org/10.1109/TVT.2012.2190631

Wang X, Xu MX, Wang HB, et al., 2012. Combination of interacting multiple models with the particle filter for three-dimensional target tracking in underwater wireless sensor networks. *Math Probl Eng*, 2012:829451. https://doi.org/10.1155/2012/829451

Yu CH, Choi JW, 2014. Interacting multiple model filter-based distributed target tracking algorithm in underwater wireless sensor networks. *Int J Contr Autom Syst*, 12(3):618-627. https://doi.org/10.1007/s12555-013-0238-y

Zhang Q, Liu MQ, Zhang SL, 2015. Node topology effect on target tracking based on UWSNs using quantized measurements. *IEEE Trans Cybern*, 45(10):2323-2335. https://doi.org/10.1109/TCYB.2014.2371232

Zhou XC, Shen HB, Ye JP, 2011. Integrating outlier filtering in large margin training. *J Zhejiang Univ-Sci C (Comput & Electron)*, 12(5):362-370. https://doi.org/10.1631/jzus.C1000361

Zhu YM, You ZS, Zhao J, et al., 2001. The optimality for the distributed Kalman filtering fusion with feedback. *Automatica*, 37(9):1489-1493. https://doi.org/10.1016/S0005-1098(01)00074-7