Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) for Dynamic-nodes Tracking

Chen Qiu (cqiu29@uwo.ca)

***Abstract*—In this report, the performance of Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) is compared in the application of tracking two dynamic nodes with different moving patterns. The criteria for comparing the performance of tow algorithms are the ensemble averaged root mean squared error (RMSE) curves of position and velocity and the total running time for each algorithm. The results demonstrate that for this particular problem, the EKF outperforms the UKF from the perspective of these two criteria.**

I. Introduction

Kalman filter (KF) theory is named after Rudolf E. Kalman who is one of the primary developers. It is well known for its advantage of including knowledge of how the state vector behaves in time compared with the complete family of recursive least-sqaures (RLS) filters in the applications of tracking time-discrete systems [1]. If the driving noise and measurement noise are Gaussian and both the state model and the measurement model are linear, KF is optimal to minimize the mean square error of the estimated parameters.

However, in practical applications, non-Gaussian noise or nonlinear models are more common to be encountered. Particle filters [2] were developed to deal with the non-Gaussian noise while extended Kalman filter (EKF) and unscented Kalman filter (UKF) were developed to deal with the nonlinearity in models [1]. This project focuses on the algorithms, i.e. EKF and UKF, which solve the problem with nonlinear measurement model.

In this report, the tracking performance of EKF and UKF based on the same state model and measurement model with the same distance measurements and initial knowledge about the positions of two nodes is compared. The criteria measuring the performance of two algorithms are the ensemble averaged root mean squared error (RMSE) curves of position and velocity and the total running time of each algorithm. The results show that for this tracking problem, the EKF outperforms the UKF with respect to the comparison criteria.

II. System Model

A. Problem definition

Fig. 1 visualizes the topology of the system and the true trajectories of two nodes. Consider two nodes i and j are moving on a plane with area of $A = 50m \times 50m$. Node i is in linear motion with constant accelerator a_i while node j is in curvilinear motion with constant accelerator a_j. Both nodes receive distance measurements from four anchors which have accurate knowledge of positions. The EKF and UKF are implemented to track these two nodes and their performance is compared in terms of tracking accuracy and consumed running time.

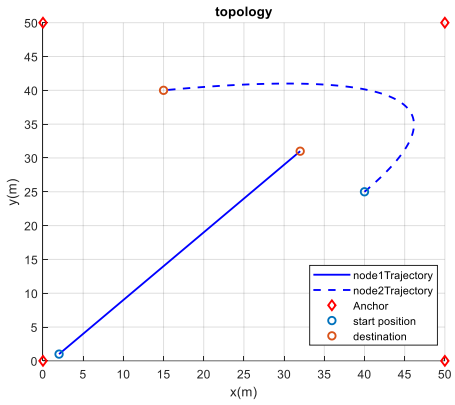

Fig. 1. Topology of the system and true trajectories

B. State Model and Measurement Model

The state vector to be estimated consists of position and velocity parameters. Both node i and node j have the same state model and measurement model, so node $i \in \mathbf{I} = \{1,2\}$ is denoted for simplicity. For a node $i \in \mathbf{I}$, the state model is

$$\mathbf{x}_i^{(n)} = \begin{bmatrix} x_{1,i}^{(n)} & x_{2,i}^{(n)} & \dot{x}_{1,i}^{(n)} & \dot{x}_{2,i}^{(n)} \end{bmatrix}^{\mathrm{T}} \tag{1}$$

where $\mathbf{x}_i^{(n)}$ contains two-dimensional (2D) position $\mathbf{p}_i^{(n)} = \begin{bmatrix} x_{1,i}^{(n)} & x_{2,i}^{(n)} \end{bmatrix}^{\mathrm{T}}$ and velocity $\mathbf{v}_i^{(n)} = \begin{bmatrix} \dot{x}_{1,i}^{(n)} & \dot{x}_{2,i}^{(n)} \end{bmatrix}^{\mathrm{T}}$. The temporal evolution of the state $\mathbf{x}_i^{(n)}$ is modeled as

$$\mathbf{x}_i^{(n)} = g_i\left(\mathbf{x}_i^{(n-1)}\right) + \mathbf{u}_i^{(n)} \tag{2}$$

$$g_i\left(\mathbf{x}_i^{(n-1)}\right) = \phi \mathbf{x}_i^{(n-1)} = \begin{bmatrix} 1 & 0 & \mathrm{T} & 0 \\ 0 & 1 & 0 & \mathrm{T} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_i^{(n-1)} \tag{3}$$

where ϕ is the transition matrix, and the driving noise $\mathbf{u}_i^{(n)}$ is zero-mean Gaussian and independent across n and i. Since the state vector does not include the accelerators of a node i, the stochastic model of this state model is velocity random walk, which means that the accelerator vector $\boldsymbol{a}_i^{(n)} = \begin{bmatrix} \ddot{x}_{1,i}^{(n-1)} & \ddot{x}_{2,i}^{(n-1)} \end{bmatrix}^{\mathrm{T}}$ is white Gaussian random noise process with uncertainty $\sigma_{\mathbf{a}_i}^2$.

$$\mathbf{u}_i^{(n)} = \Gamma \boldsymbol{a}_i^{(n)} \tag{4}$$

where Γ is the shaping matrix.

$$\Gamma = \begin{bmatrix} t^2/2 & 0 \\ 0 & t^2/2 \\ t & 0 \\ 0 & t \end{bmatrix} \tag{5}$$

In terms of measurement model, at each time step n, a node $i \in \mathbf{I}$ receives distance measurements $\mathbf{z}_{ik}(n)$ from the four anchors $k \in \{1,2,3,4\}$. Note that the position of each anchor k is accurately known, i.e. $\mathbf{p}_k = [x_{1,k} \quad x_{2,k}]^{\mathrm{T}}$.

$$\mathrm{z}_{ik}(n) = h_k\left(\boldsymbol{x}_i^{(n)}\right) + w_{ik}^{(n)} \tag{6}$$

$$h_k\left(\mathbf{x}_i^{(n)}\right) = \sqrt{(x_{1,i}^{(n)} - x_{1,k})^2 + (x_{2,i}^{(n)} - x_{2,k})^2} \tag{7}$$

where $w_{ik}^{(n)}$ denotes the measurement noise matrix, modeled as Gaussian distribution $w_{ik}^{(n)} \sim \mathcal{N}(0, \sigma_{wik}^2)$. Collect all measurements $\mathrm{z}_{ik}(n)$ for a device i at time step n, the augmented measurement model can be obtained.

$$\mathbf{z}_i(n) = h\left(\mathbf{x}_i^{(n)}\right) + \mathbf{w}_i^{(n)} \tag{8}$$

In addition, the driving noise $\mathbf{u}_i^{(n)}$ and measurement noise $\mathbf{w}_i^{(n)}$ are both assumed to be white and uncorrelated with each other, as well as uncorrelated with the state vector $\boldsymbol{x}_i^{(n)}$.

III. Algorithms

Both EKF and UKF are the extensions of linear Kalman Filter when encountering nonlinearity in system models. On the one hand, the solution of EKF is to linearize the nonlinear model using Taylor series expansion and to remain only the first two terms as the approximation to the nonlinear model. Thus, the partial derivative with respect to each element of the state vector should be derived. On the other hand, the solution of UKF is to avoid linearization process by sampling sigma points with respect to the state vector and transforming the sigma points using the nonlinear model in order to obtain the first and second order statistical properties of the transformed sigma points. In this section, the two algorithms are discussed based on the system models introduced in Section II. Note that both algorithms are discussed at the situation where the state vector is updated from time step $n-1$ to time step n.

A. EKF

The EKF generally consists of two phases, temporal prediction and the updating phase using the measurements. At the beginning of this time step n, the state vector from the previous time step $\mathbf{x}_i^{(n-1)}$ and the corresponding state covariance matrix $\mathbf{C}_{\mathbf{x}_i}^{(n-1)}$ are provided.

In the temporal prediction, the state vector and the corresponding state covariance matrix are first predicted based on the state evolution model, which is a linear function $g_i(\cdot)$ mentioned in Section II-B.

$$\mathbf{x}_i^{(n,n-1)} = g_i(\mathbf{x}_i^{(n-1)}) \tag{9}$$

$$\mathbf{C}_{\mathbf{x}_i}^{(n,n-1)} = \phi \mathbf{C}_{\mathbf{x}_i}^{(n-1)} \phi^{\mathrm{T}} + \Gamma \mathbf{C}_{\mathbf{a}_i}^{(n-1)} \Gamma^{\mathrm{T}} \tag{10}$$

where $\mathbf{C}_{\mathbf{a}_i}^{(n-1)}$ is the covariance matrix of accelerators determined by $\sigma_{\mathbf{a}_i}^2$.

In the updating phase, the state vector and the corresponding covariance matrix are updated through the information from the measurements based on the linearized measurement model. This is achieved by the following steps:

Step 1: The design matrix $\mathbf{H}$ is calculated by Taylor series expansion with only the first two terms remained.

$$\mathbf{H} = \frac{\partial h_k}{\partial \mathbf{x}_i}\Big|_{\mathbf{x}_i = \mathbf{x}_i^{(n-1)}} \tag{11}$$

Step 2: The Kalman gain $\mathbf{K}$ is calculated using the design matrix $\mathbf{H}$, the measurement noise covariance matrix $\mathbf{C}_{\mathbf{w}_i}^{(n)}$ and the predicted state covariance matrix $\mathbf{C}_{\mathbf{x}_i}^{(n,n-1)}$.

$$\mathbf{K} = \mathbf{C}_{\mathbf{x}_i}^{(n,n-1)} \mathbf{H}^{\mathrm{T}} (\mathbf{H} \mathbf{C}_{\mathbf{x}_i}^{(n,n-1)} \mathbf{H}^{\mathrm{T}} + \mathbf{C}_{\mathbf{w}_i}^{(n)})^{-1} \tag{12}$$

Step 3: The innovation vector $\delta^{(n)}$ is calculated by the obtained measurement vector $\mathbf{z}_i(n)$.

$$\delta^{(n)} = \mathbf{z}_i(n) - h\left(\mathbf{x}_i^{(n,n-1)}\right) \tag{13}$$

where $h(\cdot)$ is the nonlinear measurement function mentioned in Section II-B.

Step 4: The state vector and the corresponding state covariance matrix are corrected by the Kalman gain $\mathbf{K}$ and the innovation vector $\delta^{(n)}$.

$$\mathbf{x}_i^{(n)} = \mathbf{x}_i^{(n-1)} + \mathbf{K}\delta^{(n)} \tag{14}$$

$$\mathbf{C}_{\mathbf{x}_i}^{(n)} = (\mathbf{I} - \mathbf{K}\mathbf{H}) \mathbf{C}_{\mathbf{x}_i}^{(n,n-1)} \tag{15}$$

Finally, the updated state vector and the corresponding state covariance matrix are considered as the initial information in the next time step $n+1$.

B. UKF

The UKF generally consists of three phases, sigma points sampling, temporal prediction and updating phase using the measurements. At the beginning of this time step n, the state vector from the previous time step $\mathbf{x}_i^{(n-1)}$ and the corresponding state covariance matrix $\mathbf{C}_{\mathbf{x}_i}^{(n-1)}$ are provided.

In the phase of sigma points sampling, a set of $2L+1$ sigma vectors $\{\boldsymbol{\chi}_l\}_0^{2L}$ are generated, each of which has dimension L. Note that L is the dimension of the state vector $\mathbf{x}_i$. The first vector $\boldsymbol{\chi}_0^{(n-1)}$ is set equal to the mean of $\mathbf{x}_i^{(n-1)}$. The remaining $2L$ vectors are obtained from the columns of the matrix square roots

$$\boldsymbol{\chi}_l^{(n-1)} = \mathbf{x}_i^{(n-1)} \pm \left(\sqrt{(L+\lambda)\mathbf{C}_{\mathbf{x}_i}^{(n-1)}}\right)_l \quad l = 1,2,\ldots,2L \tag{16}$$

Where $\sqrt{(L+\lambda)\mathbf{C}_{\mathbf{x}_i}^{(n-1)}}$ is the lth column of the matrix square root, and $\lambda = \alpha^2(L+\kappa) - L$ is a scaling parameter. Besides, the weighting factors W_i for the sampled vectors are defined by

$$W_0^{(m)} = \frac{\lambda}{L+\lambda} \quad W_0^{(c)} = \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta) \tag{17}$$

$$W_l^{(m)} = W_l^{(c)} = \frac{1}{2(L+\lambda)} \quad l = 1,2,\ldots,2L \tag{18}$$

where the superscript m refers to the mean while the superscript c refers to covariance. α, β and κ are the tunable parameters. α determines the spread of the sigma points, usually between 0 .001 to 1; β takes account of prior knowledge on the distribution of the state vector, usually set to 2 for Gaussian distribution; and κ is a secondary scaling parameter usually set to 0 or $3 - L$ [1].

After generating the sigma vectors, in the temporal prediction, these sampled sigma vectors are transformed using the state evolution model mentioned in Section II-B. The mean and covariance matrix of the transformed vectors are then calculated.

$$\boldsymbol{\chi}_l^{(n,n-1)} = \phi \boldsymbol{\chi}_l^{(n-1)} \tag{19}$$

$$\mathbf{x}_i^{(n,n-1)} = \sum_{l=0}^{2L} \mathrm{W}_l^{(m)} \boldsymbol{\chi}_l^{(n,n-1)} \tag{20}$$

$$\mathbf{C}_{\mathbf{x}_i}^{(n,n-1)} = \sum_{l=0}^{2L} \mathrm{W}_l^{(c)} (\boldsymbol{\chi}_l^{(n,n-1)} - \mathbf{x}_i^{(n,n-1)})(\cdot)^{\mathrm{H}} + \mathrm{Q} \tag{21}$$

where $\mathrm{Q} = \Gamma \mathbf{C}_{\mathbf{a}_i}^{(n-1)} \Gamma^{\mathrm{T}}$ denotes the driving noise covariance matrix mentioned in Section II-B, and $(\cdot)^{\mathrm{H}}$ denotes the Hermitian transpose of $(\boldsymbol{\chi}_l^{(n,n-1)} - \mathbf{x}_i^{(n,n-1)})$.

When it comes to the phase of updating using the measurements, the state vector and the corresponding covariance matrix are updated through the information from the measurements based on the linearized measurement model. This is achieved by the following steps:

Step 1: The sigma vectors $\boldsymbol{\chi}_l^{(n-1)}$ $l = 0,1,\dots,2L$ are transformed using the nonlinear measurement function $h(\cdot)$.

$$\hat{\mathbf{z}}_l^{(n,n-1)} = h\left(\boldsymbol{\chi}_l^{(n,n-1)}\right) \quad l = 0,1,\dots,2L \tag{22}$$

Step 2: The mean and the corresponding covariance matrix of the transformed sigma vectors are calculated.

$$\bar{\mathbf{z}}_l^{(n,n-1)} = \sum_{l=0}^{2L} W_l^{(m)} \hat{\mathbf{z}}_l^{(n,n-1)} \quad l = 0,1,\dots,2L \tag{23}$$

$$\mathbf{C}_{\mathbf{z}}^{(n)} = \sum_{l=0}^{2L} \mathrm{W}_l^{(c)} (\hat{\mathbf{z}}_l^{(n,n-1)} - \bar{\mathbf{z}}_l^{(n,n-1)})(\cdot)^{\mathrm{H}} \tag{24}$$

where $(\cdot)^{\mathrm{H}}$ denotes the Hermitian transpose of $(\hat{\mathbf{z}}_l^{(n,n-1)} - \bar{\mathbf{z}}_l^{(n,n-1)})$.

Step 3: The covariance matrix between the transformed sigma vectors using the nonlinear measurement function $h(\cdot)$ and the transformed sigma vectors using the linear temporal evolution function $g(\cdot)$ is calculated.

$$\mathbf{C}_{\boldsymbol{\chi}\mathbf{z}}^{(n)} = \sum_{l=0}^{2L} \mathrm{W}_l^{(c)} \, a \cdot b \tag{25}$$

where $a = \boldsymbol{\chi}_l^{(n,n-1)} - \mathbf{x}_i^{(n,n-1)}$, $b = (\hat{\mathbf{z}}_l^{(n,n-1)} - \bar{\mathbf{z}}_l^{(n,n-1)})^{\mathrm{H}}$

Step 4: The Kalman gain is calculated using the covariance matrices.

$$\mathbf{K} = \mathbf{C}_{\boldsymbol{\chi}\mathbf{z}}^{(n)} (\mathbf{C}_{\mathbf{z}}^{(n)})^{-1} \tag{26}$$

Step 5: The innovation vector is calculated by the obtained measurements.

$$\delta^{(n)} = \mathbf{z}_i(n) - \bar{\mathbf{z}}_l^{(n,n-1)} \tag{27}$$

where $\mathbf{z}_i(n)$ is the obtained measurement vector at current time step.

Step 6: The state vector and the corresponding covariance matrix are corrected by the Kalman gain $\mathbf{K}$ and the innovation vector $\delta^{(n)}$.

$$\mathbf{x}_i^{(n)} = \mathbf{x}_i^{(n-1)} + \mathbf{K}\delta^{(n)} \tag{28}$$

$$\mathbf{C}_{\mathbf{x}_i}^{(n)} = \mathbf{C}_{\mathbf{x}_i}^{(n,n-1)} - \mathbf{K}\mathbf{C}_{\mathbf{z}}^{(n)}\mathbf{K} \tag{29}$$

Finally, the updated state vector and the corresponding state covariance matrix are considered as the initial information in the next time step $n + 1$.

IV. Simulations

Before starting the simulations of the tracking algorithms, the true trajectories of two nodes are generated. The time step duration is set to $t = 0.2\ sec$ and there are in total 200 time steps. As showed in Fig. 1, the true initial and end positions of two nodes are $\mathbf{p}_1^{initial} = [2,1]^{\mathrm{T}}$, $\mathbf{p}_2^{initial} = [40,25]^{\mathrm{T}}$, $\mathbf{p}_1^{end} = [32,31]^{\mathrm{T}}$, $\mathbf{p}_2^{end} = [15,40]^{\mathrm{T}}$. The true initial velocities of two nodes are provided, i.e. $\mathbf{v}_1^{initial} = \mathbf{v}_2^{initial} = [2,2]^{\mathrm{T}}$ .Both nodes have constant accelerators.

$$\mathbf{a}_i = [a_{i1}, a_{i2}]^{\mathrm{T}} \quad i = 1,2 \tag{30}$$

$$a_{i1} = \frac{\mathbf{p}_{i1}^{end} - \mathbf{p}_{i1}^{initial} - \mathbf{v}_{i1}^{initial} \times (200 \times t)}{(t \times 200)^2} \quad i = 1,2 \tag{31}$$

$$a_{i2} = \frac{\mathbf{p}_{i2}^{end} - \mathbf{p}_{i2}^{initial} - \mathbf{v}_{i2}^{initial} \times (200 \times t)}{(t \times 200)^2} \quad i = 1,2 \tag{32}$$

where $\mathbf{p}_{i1}^{end}$ denotes the first element of $\mathbf{p}_i^{end}$ for $i = 1,2$. Provided all these initial parameters, the true trajectories of the nodes are generated by the temporal evolution function $g_i(\cdot)$.

When it comes to the simulation setup, three different standard deviation values of driving noise are considered, i.e. $\sigma_{\mathbf{a}_i} \in \{\frac{1m}{s^2}, \frac{5m}{s^2}, \frac{10m}{s^2}\}$. Besides, three different standard deviation values for measurement noise are also considered, i.e. $\sigma_{\mathbf{w}_i} \in \{0.1m, 1m\}$. In terms of the initial state vectors and the corresponding state covariance matrices, it is assumed that the system does not have much initial knowledge about two nodes. Therefore, the initial states are set to $\mathbf{x}_1^{(0)} = [15m, 15m, 5m/s, 5m/s]$, $\mathbf{x}_2^{(0)} = [39m, 40m, 5m/s, 5m/s]$.They are given the same state covariance matrix $\mathbf{C}_{\mathbf{x}_i}^{(0)} = [(2m)^2 \ (2m)^2 \ (1m/s)^2 \ (1m/s)^2]^{\mathrm{T}} \ \ i = 1,2$.

The EKF and UKF are simulated and compared. In the simulated algorithms, the tuning parameters of UKF are set to $\alpha = 0.6, \beta = 2, \kappa = 3 - L = -1$. For both algorithms, there are 50 simulation runs. Besides, the criteria to measure the performance are the ensemble averaged RMSE curves of both position and velocity and the total running time of each algorithm.

Fig.2 and Fig. 3 present the results when $\sigma_{\mathbf{a}_i} = 5m/s^2$ and $\sigma_{\mathbf{w}_i} = 0.1\ m$. Specifically, Fig. 2 shows the tracking results of two algorithms while Fig. 3 and Fig.4 show the ensemble averaged RMSE curves of position and velocity with respect

to time steps for two nodes with different moving patterns. As can be seen in Fig.2, for node 1, both EKF and UKF can track the trajectory of the dynamic node accurately. However, for node 2, the UKF keeps tracking but with more bias and noise than EKF. More specifically, as showed in Fig.3 and Fig.4, in terms of the ensemble averaged RMSE curves of position for both nodes, the error of UKF converges more slowly, with over 180 time steps for node 1 and over 100 time steps for node 2. Besides, the position error of UKF does not remain stable after the first convergence. As for the ensemble averaged RMSE curves of velocity, it is obviously that the estimation of UKF has bias of around 5 meters than that of EKF. Note that the amount of bias is equal to the difference between the prior initial velocity values and the true initial velocity values. It is also noticeable that for the velocity error of node 1, the error curve does not converge of UKF. In short summary, the tracking of UKF on position or velocity is less accurate and less robust than that of EKF based on the provided prior knowledge of system.

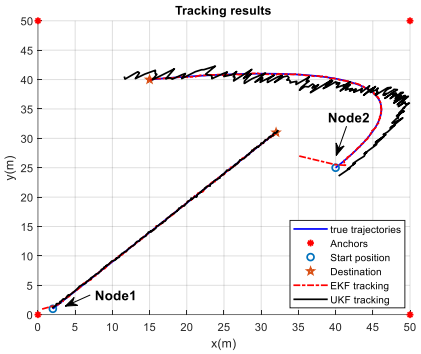

Fig. 2. Tacking results of EKF and UKF

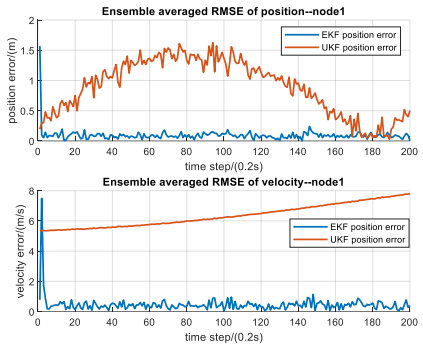

Fig. 3. Ensemble averaged RMSE of position for node 1

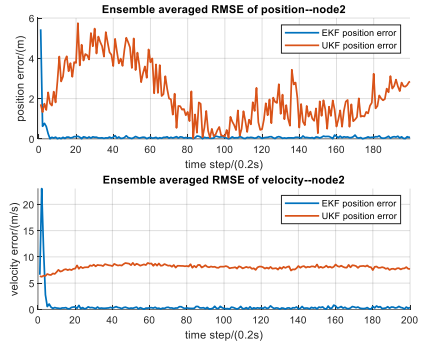

Fig. 4. Ensemble averaged RMSE of position for node 2

Other values of $\sigma_{\mathbf{a}_i} \in \{\frac{1m}{s^2}, \frac{10m}{s^2}\}$ and $\sigma_{\mathbf{w}_i} = 1m$ are also considered in the simulation. The ensemble and temporal averaged error of tracking position and velocity by both algorithms are listed in Table I (at the bottom of this report). As can be seen in Table I, on the one hand, when increasing the standard deviation $\sigma_{\mathbf{a}_i}$ of driving noise, the tracking errors of EKF are increased slightly with error of velocity increased more than error of position. However, the tracking errors of UKF are not affected obviously by the changing of $\sigma_{\mathbf{a}_i}$. On the other hand, when increasing the standard deviation $\sigma_{\mathbf{w}_i}$ of measurement noise, the tracking errors of both EKF and UKF are more sensitive compared with the responses to the change of $\sigma_{\mathbf{a}_i}$. Especially, the position error of UKF is significantly larger with greater measurement noise.

In addition to the algorithms responses to various driving noise and measurement noise, the UKF performance of different tuning parameters are also investigated. Since β and κ have the optimal values for Gaussian distributions according to [1], the value of α is tuned. The value of α is tuned when $\sigma_{\mathbf{a}_i} = 5m/s^2$ and $\sigma_{\mathbf{w}_i} = 1\,m$ .The results show that the optimal α for node 1 is $\alpha = 0.3$ while the optimal value for node 2 is $\alpha = 0.009$. Fig. 5 and Fig. 6 show the ensemble averaged RMSE of position and velocity for both nodes.

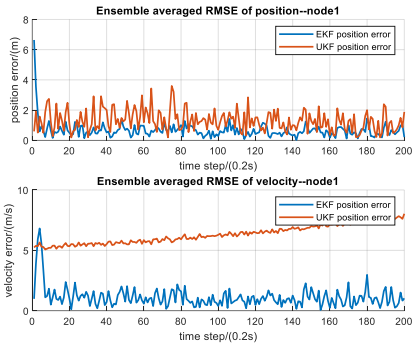

Fig. 5. Ttracking node 1 with $\alpha = 0.3$ in UKF

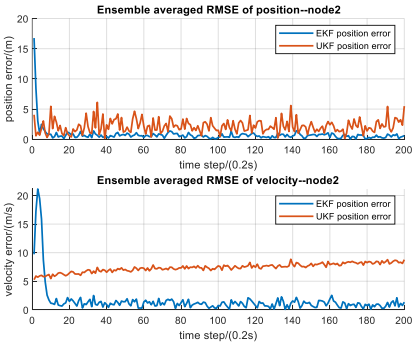

Fig. 6. Tacking node 2 with $\alpha = 0.009$ in UKF

As can be seen, with the optimal α, the tracking performance on position of UKF has almost same level of accuracy (around 1 meter) as that of EKF. It is noticeable that the ensemble averaged RMSE curve of position of UKF does not have the peak at the beginning, which indicates that UKF can capture the position of nodes faster and more accurate at the very beginning compared with EKF.

In terms of the tracking on velocity, the UKF still has bias which is probably relative to the prior knowledge on velocity. When the prior knowledge on velocity is better known, i.e.

$\mathbf{v}_1^{(0)}$ and $\mathbf{v}_2^{(0)}$ are set closer to the true values $\mathbf{v}_i^{initial} = [2,2]^{\mathrm{T}}$, the velocity tracking for UKF would probably less biased. Fig. 7 and Fig. 8 demonstrate this notion. With the same better prior knowledge of velocity, UKF can track velocity better than EKF. However, when the prior knowledge of velocity is changed, the optimal α also changes. In Fig. 8, the optimal α is 0.3 instead of 0.09.

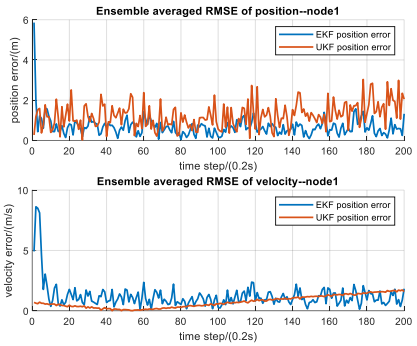

Fig. 7. Tacking node 1 with $\mathbf{v}_1^{(0)} = [1,1]^{\mathrm{T}}$

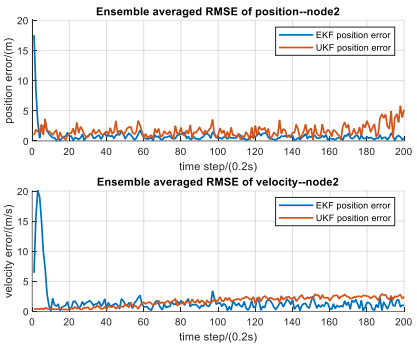

Fig. 8. Tacking node 2 with $\mathbf{v}_2^{(0)} = [1,1]^{\mathrm{T}}$

After parameters tuning, the optimal performance of UKF can be achieved when $\sigma_{\mathrm{w}_i} = 1m$, $\alpha = 0.3$ $\sigma_{\mathrm{a}_i} = 5m/s^2$, $\mathbf{v}_i^{(0)} = [1,1]^{\mathrm{T}}$. The ensemble temporal RMSEs of position and velocity are listed in Table II. As can be seen, the tracking accuracy on position and velocity of the two algorithms is on the same level with EKF better at tracking position and UKF better at tracking velocity.

TABLE II. OPTIMAL PERFORMANCE OF UKF COMPARED WITH EKF

Ensemble temporal averaged RMSE	**EKF**		**UKF**	
	Node 1	Node 2	Node 1	Node 2
$e_{\mathbf{p}}$ (m)	0.6694	0.8200	1.2192	1.5569
$e_{\mathbf{v}}$(m/s)	1.2386	1.6407	0.7359	1.6141

Finally, the ensemble averaged running time of two algorithms are recorded in Table III. The result shows that the UKF consumes around ten times longer running time than EKF. The reason is probably that both inverse calculation and square root of matrix should be operated in UKF which may require more time than Tayler series calculation in EKF. Besides, in EKF, only inverse calculation of covariance matrix should be computed.

TABLE III. OPTIMAL PERFORMANCE OF UKF COMPARED WITH EKF

Algorithms	**EKF**	**UKF**
Ensemble averaged running time (s)	0.0276	0.2284

V. CONCLUSIONS

In conclusion, this project compares the performance of EKF and UKF on tracking dynamic nodes with different motion pattern. Both EKF and UKF can track position and velocity of dynamic nodes with accuracy of around 1 m for position and less than 2 m/s for velocity.

The shortcoming for EKF is that partial derivative should be computed with respect to every element of the state vector. However, EKF tracking is not sensitive to the prior knowledge of initial state vector and does not require tuning parameters when the driving noise is properly modeled.

In terms of UKF, the advantage is that it avoids linearization process of nonlinear model, which is a distinguished advantage especially when the nonlinear function of model is too complicated to compute partial derivatives. However, the drawback is that when UKF is used to track both the position and velocity, its performance is affected by the prior knowledge on the initial state vector. Besides, the parameter which determines the spread of sigma points should be tuned. Only with the proper prior knowledge of initial state vector and the optimal tuning parameter, can the UKF tracking performs as accurately as EKF tracking. It is noticeable that once the optimal parameter and proper knowledge of initial state vector is provided, the UKF can track faster and more accurate at the very beginning than the EKF.

As for the future works, it is not yet explained in this report that why the tracking performance of UKF on dynamic nodes with different motion pattern varies. This reason should be worked out in the future research. In addition, algorithm that is able to intelligently find the optimal tuning parameter of UKF can also be an interesting topic.

REFERENCES

[1] Haykin, S. (2014). Adaptive Filter Theory. 4th ed. Harlow: Pearson, pp.773-778.

[2] C. Wu and C. Han, "Quadrature Kalman particle fitler," in Journal of Systems Engineering and Electronics, vol. 21, no. 2, pp. 175-179, April 2010.

TABLE I. VARIOUS DRIVING NOISE AND MEASRUEMENT NOISE

RMSE	Noise / Method	$\sigma_{w_i} = 0.1m$				$\sigma_{a_i} = 5m/s^2$			
		$\sigma_{a_i} = 1$		$\sigma_{a_i} = 10$		$\sigma_{w_i} = 0.1$		$\sigma_{w_i} = 1$	
		Node1	Node2	Node1	Node2	Node1	Node2	Node1	Node2
e_p(m)	EKF	0.0795	0.1135	0.0926	0.1131	0.0992	0.1118	0.6582	0.7891
	UKF	0.8815	2.2100	0.8942	2.2563	0.8883	2.2866	24.623	6.5268
e_v(m/s)	EKF	0.2320	0.4610	0.9409	1.0958	0.6253	0.7566	1.1613	1.5300
	UKF	6.3485	7.6215	6.3469	7.6143	6.3486	7.5988	6.9670	7.4864