Comparison of Angle-only Filtering Algorithms in 3D Using EKF, UKF, PF, PFF, and Ensemble KF

Syamantak Datta Gupta, Jun Ye Yu, Mahendra Mallick, Mark Coates, Mark Morelande

1McGill University, Montreal, Quebec, Canada, syamantak.dattagupta@mcgill.ca
2McGill University, Montreal, Quebec, Canada, jun.yu@mail.mcgill.ca
3Independent Consultant, Anacortes, WA., USA, mahendra.mallick@gmail.com
4McGill University, Montreal, Quebec, Canada, mark.coates@mcgill.ca
5National Australia Bank, Melbourne, VIC, Australia, m.morelande@gmail.com

Abstract—In our previous work, we compared the performance of the extended Kalman filter (EKF), unscented Kalman filter (UKF), and particle filter (PF) for the angle-only filtering (AOF) problem in 3D using Cartesian coordinates and modified spherical coordinates (MSC) for the relative state vector. We found that the UKF-MSC and EKF-MSC had the best performance in accuracy, the UKF-MSC being slightly better than the EKF-MSC. The PF didn’t perform well compared with the EKF and UKF and had a higher computational cost. In this work, we compare the performance of the particle flow filter (PFF) with the other filters for the AOF problem. In addition, we also analyze the performance of two versions of the ensemble Kalman filter (EnKF) in this comparative study. We present numerical results from Monte Carlo simulations to analyze the state estimation accuracy and computational cost of these filters.

Keywords: Angle-only filtering (AOF) in 3D, Modified spherical coordinates (MSC), Extended Kalman filter (EKF), Unscented Kalman filter (UKF), Particle filter (PF), Particle flow filter (PFF), and Ensemble Kalman filter (EnKF).

I. INTRODUCTION

The angle-only filtering (AOF) problem in 3D using bearing and elevation angles occurs in a number of important real-world problems. These problems include passive ranging using an infrared search and track (IRST) sensor, passive sonar, passive radar in the presence of jamming, satellite-to-satellite passive tracking, ballistic object (a missile or satellite) tracking by optical telescopes, and ground target tracking by small unmanned aerial vehicles by video cameras. Significant research has been performed for the bearings-only filtering (BOF) problem in 2D, see [6], [35] and the references therein. However, the number of papers on the AOF problem in 3D is relatively small [1], [22], [28], [31]—[33], [34], [36], [38], [39].

Three types of coordinate systems, Cartesian coordinates, modified spherical coordinates (MSC), and log spherical coordinates (LSC) are used to represent the relative state vector in the near-constant velocity model [4] in 3D and nonlinear measurement model for bearing and elevation does not face the collapse of the covariance matrix as in the BOF. However, as in MPC, the dynamic model using MSC is nonlinear and complex. Since the bearing and elevation are components of the MSC, the measurement model is linear and the measurement update step in MSC is straightforward [4].

The particle flow filter (PFF) developed by Daum and Huang [10], [11], [15] has received a great deal attention in the Fusion community because it has been claimed to have superior accuracy and computational speed. This has led to publications by other researchers [5], [9], [16], [29], [30]. In many cases, detailed steps for implementing the PFF algorithm are missing; the paper [16] provides algorithm details for the benefit of a graduate student or researcher.

Daum and Huang studied the AOF problem of a long-range ballistic missiles using angle-only measurements in [10] and used the MSC for the relative state vector. They compared the performance of the EKF-MSC and the particle flow filter (PFF) using MSC (PFF-MSC) using a time interval of 400 seconds. They used 500 particles in the PFF-MSC and used 100 Monte Carlo simulations. They studied various geometries by varying the angle between the velocity of the missile and the line-of-sight (LOS) of the sensor and presented the median error for missile position. Details on the missile dynamic model, sensor trajectory, and the measurement accuracy were not presented in [10]. It was claimed that for certain difficult geometries, the PFF-MSC was more accurate than the EKF-MSC by an order of magnitude [10], [15]. A close examination of the plots in Figs. 2–6, however, shows that the median position error...
The range vector can be expressed in terms of range, bearing, and elevation as follows, in Cartesian coordinates:

\[ \mathbf{r} = r \cos \epsilon \sin \beta, \quad \mathbf{r} = r \sin \epsilon, \quad z = z. \]

Then the range vector of the target can be defined by

\[ \mathbf{r} := [x, y, z]'. \]

The range is defined by

\[ \rho := \sqrt{x^2 + y^2 + z^2}, \quad \rho > 0. \]

The range vector can be expressed in terms of range, bearing (\( \beta \)) and elevation (\( \epsilon \)), as defined in Figure 1, by

\[ \mathbf{r} = r \left[ \begin{array}{c} \cos \epsilon \sin \beta \\ \cos \epsilon \cos \beta \\ \sin \epsilon \end{array} \right], \quad \beta \in [0, 2\pi], \quad \epsilon \in [-\pi/2, \pi/2]. \]

Finally, the ground range is defined as follows:

\[ \rho := \sqrt{x^2 + y^2} = r \cos \epsilon, \quad \rho > 0. \]
B. Modified Spherical Coordinates for Relative State Vector

We follow the conventions of Stallard [38], and define $\omega$ as a component of the MSC, where

$$\omega(t) := \tilde{\beta}(t) \cos \epsilon(t).$$  

(8)

Let $\zeta(t)$ denote the logarithm of range $r(t)$

$$\zeta(t) := \ln r(t).$$  

(9)

Then

$$r(t) = \exp[\zeta(t)].$$  

(10)

Differentiating (9) with respect to time, we get

$$\dot{\zeta}(t) = \dot{r}(t)/r(t).$$  

(11)

The relative state vector of the target in MSC is defined by [31], [38]

$$\xi(t) := \left[ \begin{array}{c} \omega \, t \varepsilon \end{array} \right]'$$  

(12)

The components of the MSC defined in [38] and [31] are the same; however, the ordering is different.

III. Dynamic Models

We begin by presenting the discrete-time dynamic models of the target and ownership in the T frame. Next, we present the corresponding dynamic models. The target follows the near-constant velocity model in 3D, while the ownership follows a sequence of constant velocity (CV) and coordinated turn (CT) models in a plane parallel to the XY plane of the T frame.

A. Dynamic Model for State Vector and Relative State Vector in Cartesian Coordinates

At time $t_k$, let the Cartesian state vector of the target be given by $x_k^t$. Following [4], [19], the discrete-time dynamic model of the target can then be defined as follows

$$x_k^t = F_{k-1} x_{k-1}^t + w_{k-1}^t,$$  

(13)

where $F_{k-1}$ and $w_{k-1}^t$ are the state transition matrix [4], [19] and integrated process noise [19], respectively, for the time interval $[t_k-1, t_k]$,

$$F_{k-1} = F(t_k, t_{k-1}) := \begin{bmatrix} 1 & \Delta_k \\ 0 & 1 \end{bmatrix} \otimes I_3,$$  

(14)

$$w_{k-1}^t := \int_{t_{k-1}}^{t_k} F(t_k - t) w(t) \, dt,$$  

(15)

where $\Delta_k := t_k - t_{k-1}$ is the sampling interval and $\otimes$ refers to the Kronecker product. Using the properties of the continuous-time process noise $w(t)$, state transition matrix $F_{k-1}$ defined in (14), and the definition of $w_{k-1}^t$ in (15), it can be shown that $w_{k-1}^t$ is a zero-mean Gaussian white integrated process noise with covariance $Q_{k-1}$

$$w_{k-1}^t \sim N(w_{k-1}^t; 0, Q_{k-1}),$$  

(16)

$$Q_{k-1} = \begin{bmatrix} \Delta_k^3/3 & \Delta_k^2/2 \\ \Delta_k^2/2 & \Delta_k \end{bmatrix} \otimes \text{diag}(q_x, q_y, q_z).$$  

(17)

Here, $q_x, q_y, q_z$ are the power spectral densities of the process noise [19] along the $X, Y$, and $Z$ axes respectively.

As the dynamic model of the ownership is deterministic, the process noises are not used to simulate its motion. The CV and CT motions of the ownership are defined as follow:

$$x_k^o = F_{k-1} x_{k-1}^o + w_{k-1}^o,$$  

(18)

$$x_k^o = F_{CT}^o(\Delta_0, \omega) x_{k-1}^o,$$  

(19)

where

$$F_{CT}^o(\Delta, \omega) = \begin{bmatrix} 1 & 0 & 0 & \sin(\omega\Delta)/\omega & -[1 - \cos(\omega\Delta)]/\omega & 0 \\ 0 & 1 & 0 & [1 - \cos(\omega\Delta)]/\omega & \sin(\omega\Delta)/\omega & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta \\ 0 & 0 & 0 & \cos(\omega\Delta) & \sin(\omega\Delta) & 0 \\ 0 & 0 & 0 & \sin(\omega\Delta) & \cos(\omega\Delta) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$  

(20)

Using (13) in the definition (3) of the relative state vector and then adding and subtracting $F_{k-1} x_{k-1}^o$, we get

$$x_k = F_{k-1} x_{k-1} + w_{k-1}^o - u_{k-1},$$  

(21)

where

$$u_{k-1} := x_{k-1}^o - F_{k-1} x_{k-1}^o.$$  

(22)

We note that when $\omega_k^o$ approaches zero, $F_{CT}^o(\Delta_0, \omega_k^o)$ reduces to $F_{k-1}$ and $u_{k-1}$ becomes zero.

B. Dynamic Model for Relative State Vector in Modified Spherical Coordinates

Let $f_{C}^{MSC} : \mathbb{R}^6 \rightarrow \mathbb{R}^6$ denote the transformation from relative Cartesian coordinates to MSC. Similarly, let $f_{C}^{MSC} : \mathbb{R}^6 \rightarrow \mathbb{R}^6$ denote the inverse transformation from MSC to relative Cartesian coordinates. Then,

$$\xi_k = f_{C}^{MSC}(x_k),$$  

(23)

$$x_k = f_{C}^{MSC}(\xi_k).$$  

(24)

Functional forms of $f_{C}^{MSC}(x_k)$ and $f_{C}^{MSC}(\xi_k)$ are presented in [33]. Then, using (21) in (23), we get

$$\xi_k = f_{C}^{MSC}(F_{k-1} x_{k-1} + w_{k-1}^o - u_{k-1}).$$  

(25)

We have

$$x_{k-1} = f_{C}^{MSC}(\xi_{k-1}).$$  

(26)

Substitution of (26) in (25) gives

$$\xi_k = f_{C}^{MSC}(F_{k-1} f_{C}^{MSC}(\xi_{k-1}) + w_{k-1}^o - u_{k-1}).$$  

(27)

Formally, we can write (27) as

$$\xi_k = b(\xi_{k-1}, u_{k-1}, w_{k-1}),$$  

(28)

where $b$ is a nonlinear function of $\xi_{k-1}, u_{k-1},$ and $w_{k-1}^o$. We note that a closed form analytic expression for the nonlinear function $b$ is difficult to obtain. However, it is straightforward to calculate the predicted state estimate $\hat{\xi}_{k|k-1}$ approximately given $\xi_{k-1|k-1}$, using nested functions in (27). It can be
seen from (28) that the process noise \( \mathbf{w}_{k-1} \) is non-linearly transformed in the MSC dynamic model. This makes the EKF and UKF slightly more complicated compared to the usual dynamic models in which the process noise is additive.

### C. Measurement Models

The passive sensor collects bearing and elevation measurements \( \{ z_k \} \) at discrete times \( \{ t_k \} \). The measurement model for the bearing and elevation angles using the relative Cartesian state vector \( \mathbf{x}_k \) is

\[
\mathbf{z}_k = \mathbf{h} (\mathbf{x}_k) + \mathbf{n}_k, \quad (29)
\]

where \( \mathbf{h}(\mathbf{x}_k) := \begin{bmatrix} \beta_k \\ \epsilon_k \end{bmatrix} = \begin{bmatrix} \tan^{-1}(x_k, y_k) \\ \tan^{-1}(z_k, \rho_k) \end{bmatrix}, \quad (30) \]

where the ground range \( \rho_k \) is defined in (7) and \( \mathbf{n}_k \) is a zero-mean white Gaussian measurement noise with covariance \( \mathbf{R} \)

\[
\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \quad \mathbf{R} := \text{diag}(\sigma_\beta^2, \sigma_\epsilon^2). \quad (31)
\]

The measurement model for the bearing and elevation angles \( \beta_k, \epsilon_k \) using MSC is

\[
\mathbf{z}_k = \mathbf{H} \mathbf{z}_k + \mathbf{n}_k, \quad (32)
\]

\[
\mathbf{H} := \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (33)
\]

### IV. Nonlinear Filtering Using Cartesian and Modified Spherical Coordinates

The dynamic model (near-constant velocity model in 3D) for Cartesian co-ordinates is linear and the measurement model for bearing and elevation is non-linear for this case. On the other hand, the dynamic model using MSC is non-linear as in (27) or (28). In addition, the process noise is not additive. Since, the bearing and elevation are elements of the MSC, the measurement model (32) is linear in this case.

Detailed descriptions of algorithms used for the Cartesian and Spherical (MSC) versions of the UKF, EKF and BPF can be found in [32] and [33]. The BPF algorithm was slightly modified in this study by introducing a resampling step when the effective sample size fell below a certain threshold, as opposed to resampling at every step. In this section, we present the algorithms used for the particle flow filter (PFF) and the ensemble Kalman filter (EnKF).

#### A. Particle Flow Filter

In the particle flow filter (PFF), [12], [14], [13], a set of particles are migrated smoothly from the previous to the next time instant using a particle flow derived through a differential equation. The unnormalized conditional probability density functions can be expressed as a product of two probability density functions as follows:

\[
\tilde{p}(\mathbf{x}_k|z_{1:k}) := p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|z_{1:k-1}). \quad (34)
\]

We denote \( p(\mathbf{z}_k|\mathbf{x}_k) \), and \( p(\mathbf{x}_k|z_{1:k-1}) \) by \( \psi(\mathbf{x}_k) \) and \( g(\mathbf{x}_k) \) respectively. Daum and Huang introduce a function \( \phi(\mathbf{x}, \lambda) \) as follows:

\[
\phi(\mathbf{x}, \lambda) := \log g(\mathbf{x}_k) + \lambda \log \psi(\mathbf{x}_k). \quad (35)
\]

with \( \lambda \in [0, 1] \). At each time step, the particle flow is created by varying \( \lambda \) from 0 to 1. The objective is to evaluate \( \frac{\partial \phi}{\partial \lambda} \) for each particle and migrate them to the next time instant accordingly. In [13], Daum and Huang generalized and improved the filter using the Fokker-Planck equation with zero process noise. Denote \( \frac{\partial \phi}{\partial \lambda} \) by \( f(\mathbf{x}, \lambda) \). Under the condition that the function \( \phi \) is smooth and non-zero, the following relation between the function \( \phi \) and \( f \) is derived:

\[
\frac{\partial \log \phi}{\partial \lambda} f(\mathbf{x}, \lambda) + \log(\psi) = -\text{Tr} \left( \frac{\partial f}{\partial \lambda} \right). \quad (36)
\]

The exact flow of the probability density function can be evaluated by solving the above differential equation. For a nonlinear measurement model with Gaussian noise, a closed form solution for \( \frac{\partial \phi}{\partial \lambda} \) can be obtained [16], using a first order Taylor series approximation of the measurement function, and it is given by

\[
\frac{\partial \phi}{\partial \lambda} = A(\lambda) \mathbf{x} + \mathbf{b}(\lambda), \quad (37)
\]

where

\[
A = -\frac{1}{2} \mathbf{P} \mathbf{H}^\top (\lambda \mathbf{H}_k \mathbf{P} \mathbf{H}^\top - \mathbf{R})^{-1} \mathbf{H}_k, \quad (38)
\]

\[
b = (\mathbf{I} + 2 \lambda \mathbf{A}) (\mathbf{I} + \lambda \mathbf{A}) \mathbf{P} \mathbf{H}^\top (\mathbf{R}^{-1} (\mathbf{z}_k - \mathbf{e}_k) + \mathbf{A} + \mathbf{x}_k). \quad (39)
\]

Here, \( \mathbf{x}_k \) denotes the predicted value of \( \mathbf{x}_k \), \( \mathbf{P} \) is the predicted covariance matrix, \( \mathbf{H}_k := \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} |_{\mathbf{x} = \mathbf{x}_k} \), and \( \mathbf{e}_k = h(\mathbf{x}_k|k-1) - \mathbf{H}_k (\mathbf{x}_k|k-1) \) denotes the error induced in approximating \( h(\mathbf{x}_k) \) by \( \mathbf{H}_k (\mathbf{x}_k) \). The predicted value of the covariance \( \mathbf{P} \), can be evaluated by an unscented or extended Kalman filter running in parallel to the particle flow filter to improve performance. Pseudocode used for the particle flow paper can be found in Algorithm 1. A slight modification of the original PFF algorithm by Daum and Huang was suggested in [16], where linearization of the measurement function is performed at each particle location, instead of only at the mean. In this filter, which we refer to as the PFF (localized), the pseudocode from the second for loop onwards was replaced with the pseudocode provided in Algorithm 2.

Algorithms 1 and 2 provide the recursion steps for Cartesian co-ordinates. For the MSC versions of these two filters, the difference is in the propagation step, where one has to use (28) to propagate the particles, and in the evaluation of \( \mathbf{A} \) and \( \mathbf{b} \). As the measurement model (32) is linear, a Taylor series approximation of \( h(\cdot) \) is not required for the MSC case, and the term \( \mathbf{e}_k \) in (39) is zero.

#### B. Ensemble Kalman Filter

The ensemble Kalman filter (EnKF) [17], [18], [20], [37], [41] is an extension of the Kalman filter for non-linear models, where instead of a single state estimate the filter maintains a statistical sample of state estimates, termed as the ensemble. This filter executes the task of estimation by minimizing the error covariance, where error statistics are modeled using the ensemble of predicted states. At the forecast step, using the ensemble of forecast states, the sample mean and error covariances are evaluated. The forecast ensemble members
are combined to evaluate the ensemble covariance of the predicted observations and the cross covariance of the state and observation ensembles. These are used to compute a single Kalman gain matrix, which is subsequently used to update the ensemble using new observation.

Different versions of the EnKF have been proposed, and these can be classified into two categories, stochastic [17], [18], [20] and deterministic [37], [41]. The two classes are identical in the forecast step and differ only in the analysis step. While the stochastic version of the filter computes the observation error covariance by perturbing the observation, the deterministic version avoids this step.

The stochastic version of the EnKF implemented for this paper follows the steps discussed in [20] and is described in the pseudocode given in Algorithm 3. To implement the deterministic version, we used a Taylor series approximation for the matrix square root, as suggested in [37]. The steps are elaborated in Algorithm 4.

For the relative MSC, these algorithms differ only in the propagation step, where (28) is used to obtain the forecast ensemble of state variables, and in evaluating the corresponding forecast value of the observation, which is evaluated by using the linear measurement equation given in (32).

Algorithm 1: A recursion of the particle flow filter with Cartesian coordinates (CPFF).

Input: EnKF/UKF estimates $m_{k-1|k-1}$, covariance matrix $P_{k-1|k-1}$ and the particle cloud $\{x_{k-1}^i\}$ at time $t_{k-1}$ and the measurement $z_k$.

Output: updated EKF/UKF mean $m_{k|k}$, covariance matrix $P_{k|k}$ and cloud $\{x_{k}^i\}$ at time $t_k$.

for $k = 1, \ldots, T$ do
  Propagate particles: $x_{k}^i = F(\Delta k)x_{k-1}^i - u_{k-1}$
  Calculate the sample mean $\bar{x}_k = \frac{1}{N} \sum_{i=1}^{N} x_{k}^i$
  Apply UKF/EKF Prediction
    $(m_{k-1|k-1}, P_{k-1|k-1}) \rightarrow (m_{k|k-1}, P_{k|k-1})$
  for $j = 1, \ldots, N$ do
    Set $\lambda = j \Delta \lambda$
    Calculate $H_x$ by linearizing $h(\cdot)$ at $\bar{x}_k$
    Calculate $A$ and $b$ using (38) and (39)
    for $i = 1, \ldots, N$ do
      Evaluate $\frac{dx_k}{d\lambda}$ from (37) for each particle
      Migrate particles $x_{k}^i = x_{k}^i + \Delta \lambda \frac{dx_k}{d\lambda}$
    end
    Re-evaluate sample mean $\bar{x}_k$ using the updated particles $\{x_{k}^i\}$.
  end
  Apply UKF/EKF update :
    $(m_{k|k-1}, P_{k|k-1}) \rightarrow (m_{k|k}, P_{k|k})$
  Estimate $x_k$ from the particles $\{x_{k}^i\}$ (sample mean or robust mean)
  Optional: redraw particles $x_{k}^i \sim N(\bar{x}_k, P_{k|k})$.
end

Algorithm 2: A recursion of the particle flow filter-localized with Cartesian coordinates (CPFF-localized).

for $j = 1, \ldots, N$ do
  Calculate $\frac{dx_k}{d\lambda}$ from (37) for each particle
  Migrate particles $x_{k}^i = x_{k}^i + \Delta \lambda \frac{dx_k}{d\lambda}$
end

V. NUMERICAL SIMULATIONS AND RESULTS

The scenario used in our simulation is the same as in [32], [33], and is similar to that used in [7], with changes to make it three dimensional in nature. The initial height $z_0$ of the ownship is 10.0 km. The target’s initial ground range $r_0$, bearing $\beta_1$, and height $z_0$ are shown in Table I. Then the initial elevation $\epsilon_1$ can be calculated. Table I also shows the target’s initial speed $s_1$, course $c_1$ in the $XY$-plane and the $Z$ component of velocity $z_0$. The target’s initial position and velocity in Cartesian coordinates can be found from Table I as $(138/\sqrt{2}, 138/\sqrt{2}, 9$) km and $297/\sqrt{2}(-1, -1, 0)$ m/s, respectively. The motion of the target is governed by the near-constant velocity model. The power spectral densities $(q_x, q_y, q_z)$ of the zero-mean white acceleration process noise along the $X$, $Y$, and $Z$ axes of the T frame are $(0.01, 0.01, 0.00011)$ m$^2$s$^{-3}$, respectively.

The ownship moves in a plane parallel to the $XY$-plane at a fixed height of 10 km, with a motion profile as described in Table II. In Table II, $\Delta \phi$ represents the duration of the segment, $\Delta \phi$ is the total angular change during the segment, and $\omega_0$ is the angular velocity of the ownship during the segment. The ownship trajectory and the true target trajectory from the first Monte Carlo run are shown in Figure 2.

The measurement sampling time interval is 1.0 s. The bearing and elevation measurement error standard deviations used in our simulation were 1, 15 and 35 mrad (0.057, 0.857 and 2 degrees), representing high, medium, and low measurement accuracy, respectively. We used 1000 Monte Carlo simulations to calculate various statistics such as the root mean square (RMS) position and velocity errors.

The filters are initialized as described in [33] with the parameters shown in Table I. PFs are implemented with a sample size of 10,000. The EnKFs and PFFs are run with 500 ensemble members and 500 particles respectively. These numbers were so chosen that increasing them further did not
Algorithm 3: A recursion of the stochastic ensemble Kalman filter with Cartesian coordinates (CSEnKF).

**Input:** The posterior ensemble \{x_{k-1}^i\} at time \(t_{k-1}\) and the measurement \(z_k\).

**Output:** updated ensemble \{x_k^i\} at time \(t_k\).

for \(k = 1, \ldots, T\) do

Propagate particles to obtain the forecast state ensemble \(x_k^i = F(\Delta k)x_{k-1}^i - u_{k-1}\).

Calculate the sample mean of the forecast state ensemble \(\bar{x}_k = \frac{1}{N} \sum_{i=1}^{N} x_k^i\).

for \(i = 1, \ldots, N\) do

Compute the predicted observations: \(\hat{z}_k^i = h(x_k^i)\).

Perturb the observations by adding noise \(\hat{z}_k^i + n_k^i\) where \(n_k^i \sim N(0, R)\).

end

Calculate the sample mean of the forecast observation ensemble \(z_k = \frac{1}{N} \sum_{i=1}^{N} \hat{z}_k^i\).

Evaluate the error matrices \(E_k^x = [x_k^1 - \bar{x}_k \ldots x_k^N - \bar{x}_k]\) and \(E_k^z = [z_k^1 - \bar{z}_k \ldots z_k^N - \bar{z}_k]\).

Evaluate the ensemble covariance terms \(P_k^{x,x} = \frac{1}{N-1}E_k^x(E_k^x)^\top\) and \(P_k^{z,z} = \frac{1}{N-1}E_k^z(E_k^z)^\top\).

Compute the Kalman gain \(K = P_k^{z,x}(P_k^{z,z})^{-1}\).

Update ensemble members:

for \(i = 1, \ldots, N\) do

\(x_k^i = x_k^i + K(z_k - \hat{z}_k^i)\)

end

end

Algorithm 4: A recursion of the deterministic ensemble Kalman filter with Cartesian coordinates (CDEnKF).

**Input:** The posterior ensemble \{x_{k-1}^i\} at time \(t_{k-1}\) and the measurement \(z_k\).

**Output:** updated ensemble \{x_k^i\} at time \(t_k\).

for \(k = 1, \ldots, T\) do

Propagate particles to obtain the forecast state ensemble \(x_k^i = F(\Delta k)x_{k-1}^i - u_{k-1}\).

Calculate the sample mean of the forecast state ensemble \(\bar{x}_k = \frac{1}{N} \sum_{i=1}^{N} x_k^i\).

for \(i = 1, \ldots, N\) do

Compute the predicted observations: \(\hat{z}_k^i = h(x_k^i)\).

end

Calculate the sample mean of the forecast observation ensemble \(z_k = \frac{1}{N} \sum_{i=1}^{N} \hat{z}_k^i\).

Evaluate the error matrices, normalized by \(\sqrt{N-1}:\)

\(E_k^x = \sqrt{N-1}[x_k^1 - \bar{x}_k \ldots x_k^N - \bar{x}_k]\) and \(E_k^z = \sqrt{N-1}[z_k^1 - \bar{z}_k \ldots z_k^N - \bar{z}_k]\)

\(S = E_k^x(E_k^x)^\top + R\)

\(U = I - (E_k^z)^\top S^{-1}E_k^z\)

Use a first order Taylor series approximation to evaluate the square root of \(U: T = I - 0.5U\).

Compute the Kalman gain \(K = E_k^z(E_k^z)^\top S^{-1}\).

Update the mean of the state ensemble \(\bar{x}_k = E_k^x + K(z_k - \bar{z}_k)\).

Update the state error ensemble \(E_k^x = E_k^z T\).

Update the ensemble members:

for \(i = 1, \ldots, N\) do

\(x_k^i = \bar{x}_k + \sqrt{N-1}E_k^x\)

end

end

Table 1: Initial parameters of target.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\rho_1) (km)</td>
<td>138.0</td>
</tr>
<tr>
<td>(\beta_1) (deg)</td>
<td>45.0</td>
</tr>
<tr>
<td>(\epsilon_1) (deg)</td>
<td>-0.415</td>
</tr>
<tr>
<td>(\beta_1) (km)</td>
<td>9.0</td>
</tr>
<tr>
<td>(v_1) (m/s)</td>
<td>297.0</td>
</tr>
<tr>
<td>(c_1) (deg)</td>
<td>-135.0</td>
</tr>
<tr>
<td>(z_1) (m/s)</td>
<td>0.0</td>
</tr>
</tbody>
</table>

lead to any substantial improvement in tracking performance. The Cartesian versions of the PFF were run alongside an EKF to propagate the covariance matrices while the PFF-MSC was implemented using a UKF-MSC.

Two measures of performance are used to compare the algorithms. The first is the RMS position error averaged from the end of the last observer maneuver to the end of the surveillance period. The second performance measure is the RMS position error at the end of the surveillance period. The results are shown in Tables III and IV. For the filters implemented in MSC, these errors are evaluated using the relative Cartesian coordinates calculated exactly from MSC, following Appendix 1B in [33]. The execution times of the algorithms are given in Table V.

The best performance is achieved by the two implementations of the PFF in Cartesian coordinates, followed closely by the MSC versions of the PFF, EKF, UKF and the DEEnKF. Especially for low measurement accuracy, the performance improvements of these filters over the others are substantial.

Among the two versions of the ensemble Kalman filters, the deterministic version was found to outperform to a relative extent. For both filters, the performance was better when MSC was used as compared to Cartesian co-ordinates.

It is interesting to note that the particle flow filter demonstrated a significant improvement over the bootstrap particle filter. The former achieves this performance improvement by using only 500 particles, as compared to 10,000 particles used by the bootstrap filter. The PFF (local) algorithm provided marginal improvements over the original PFF and had a higher cost of computation. Regularization of the covariance matrix following [33], and an introduction of a resampling step improved the performance of the PF. The relative performance of the various filters is very similar when we compare the tracking errors for the velocity components of the state.
TABLE II: Motion profile of ownship for various segments.

<table>
<thead>
<tr>
<th>Time Interval (s)</th>
<th>∆t (s)</th>
<th>∆φ (rad)</th>
<th>Motion Type</th>
<th>ω (rad/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 15]</td>
<td>15</td>
<td>0</td>
<td>CV</td>
<td>0</td>
</tr>
<tr>
<td>[15, 31]</td>
<td>16</td>
<td>−π/4</td>
<td>CT</td>
<td>−π/64</td>
</tr>
<tr>
<td>[31, 43]</td>
<td>12</td>
<td>0</td>
<td>CV</td>
<td>0</td>
</tr>
<tr>
<td>[43, 75]</td>
<td>11</td>
<td>0</td>
<td>CT</td>
<td>π/64</td>
</tr>
<tr>
<td>[75, 86]</td>
<td>16</td>
<td>−π/4</td>
<td>CT</td>
<td>−π/64</td>
</tr>
<tr>
<td>[86, 102]</td>
<td>108</td>
<td>0</td>
<td>CV</td>
<td>0</td>
</tr>
</tbody>
</table>

VI. CONCLUSIONS

In this paper, we consider the angle-only filtering problem in 3D using bearing and elevation measurements, and present a comparison of performance of various nonlinear filters using relative Cartesian coordinates and modified spherical coordinates (MSC). Of the algorithms considered here, the EKF, UKF and DEnKF in MSC, and the PFF are found to provide the best performance.

The MSC is the 3D analogue of the well-known modified polar coordinates (MPC) in 2D. In MPC, the bearing, bearing-rate, range-rate divided by range form the observable components of the state vector; while the inverse of range is the unobservable component. Ill-conditioning of the covariance matrix is prevented by decoupling the observable and unobservable components in MPC, which yields better filter performance than that using Cartesian coordinates [23]. For MSC, the first five components are observable and the last component is unobservable. Our results indicate that tracking using MSC is beneficial because of this decoupling, although the benefits are modest in our scenario.

As extensions of this work, we would like to consider the realistic scenario of a maneuvering target. It would be interesting to calculate the posterior Cramér Rao lower bound (PCRLB) [40], [24] for the filtering problem which represents the best achievable accuracy. This would enable a rigorous comparison of the accuracy of the various filtering algorithms discussed in this paper.

ACKNOWLEDGMENT

The authors thank Sam Blackman for providing the details of the tracking scenario and useful discussions and advice.
TABLE V: CPU times of filtering algorithms for the measurement standard deviation of 15 mrad.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CPU (s)</th>
<th>Relative CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKF</td>
<td>0.023</td>
<td>1.00</td>
</tr>
<tr>
<td>CUKF</td>
<td>0.052</td>
<td>2.26</td>
</tr>
<tr>
<td>CSEnKF</td>
<td>0.080</td>
<td>3.91</td>
</tr>
<tr>
<td>UKF-MSC</td>
<td>0.136</td>
<td>5.91</td>
</tr>
<tr>
<td>SenKF-MSC</td>
<td>0.201</td>
<td>8.74</td>
</tr>
<tr>
<td>BPF</td>
<td>0.503</td>
<td>21.87</td>
</tr>
<tr>
<td>DEnKF</td>
<td>0.977</td>
<td>42.48</td>
</tr>
<tr>
<td>EKF-MSC</td>
<td>1.142</td>
<td>49.65</td>
</tr>
<tr>
<td>CPEF</td>
<td>1.257</td>
<td>54.65</td>
</tr>
<tr>
<td>DPEF-MSC</td>
<td>1.270</td>
<td>55.21</td>
</tr>
<tr>
<td>CPF</td>
<td>1.335</td>
<td>58.04</td>
</tr>
<tr>
<td>CPF(FC(1))</td>
<td>6.784</td>
<td>294.95</td>
</tr>
</tbody>
</table>

REFERENCES


