PNT Fusion Algorithms: Architectures and Approaches

Murat Efe
Ankara University
50. Yil Yerleskesi, L Blok
Electrical and Electronics Eng. Dept.
06830, Golbasi – Ankara
TÜRKİYE

efe@eng.ankara.edu.tr

ABSTRACT

The concept of Position, Navigation and Timing (PNT) inherently contains estimation of a quantity using noisy sensor measurements. The fact that estimation will have to be performed is independent of the number of systems/sensors producing the measurements and whether those sensors transparently output the measured value or systems output a processed entity. In all cases a decision needs to be made regarding the architecture and/or the algorithm to be employed. This paper is the first part of a two-part paper aiming to outline the mostly employed fusion architectures and integration algorithms for PNT applications. The papers present the basics of those architectures/algorithms introduced and provide some simulation results to facilitate better comprehension.

1.0 INTRODUCTION

According to the widely accepted descriptions given in [1], positioning is the ability to accurately and precisely determine one's location and orientation anywhere in 2D or 3D, navigation is the ability to determine current and desired position and apply corrections to course, orientation, and speed to attain the desired position anywhere, whereas timing is the ability to acquire accurate and precise time from a standard anywhere, which also includes time transfer. There could be a single source of information providing the user with all necessary outputs. However, it is unlikely that a single source would be able to sustainably offer all outputs all the time and under all conditions. Hence some kind of mechanism needs to be developed to collect outputs of separate (and possibly disparate) sources (systems/sensors) and process them to produce a single value for the parameter of interest. Such a framework is often called the fusion of available information and the process of producing a single output from multiple inputs is called integration, though these two terms are frequently use interchangeably. There is yet another term, i.e., estimation, to add to this bunch that is used to describe a similar process. Although estimation is the process of determining the value of a constant of dynamically varying parameter based on noisy measurements, in a broader sense it could be considered as integration. In other words, despite subtle differences it is not wrong to conclude that all these terms correspond to the same process.

When dealing with multiple sources, decision on the fusion architecture as well as the integration algorithm depends on the available communication and/or computation resources. When the resources are adequate, then sometimes how quickly one desires to produce an output is the main deciding factor and sometimes time is not the issue but the accuracy is. Whether it is designing a new system from scratch or designing an algorithm for a legacy system, how well one knows the tools to be utilized often determines the quality of the output.

This is first part of a two-part paper aiming to introduce most commonly used architectures and algorithms for PNT applications and incite better understanding pertaining to the specifics of the presented tools. Note that, the tools presented here are not specific to PNT applications but could be designed with the specifics of such systems in mind.

This paper is organized as follows. Next section introduces the fusion architectures and outlines the approaches employed in those architectures, whereas the Kalman filter and modifications to Kalman filter to relax the assumptions to the standard approach are given in Section 3. Simulation results to facilitate better understanding of how Kalman based approaches are presented in Section 4 and some concluding remarks are given in the last section.

2.0 FUSION ARCHITECTURES

There are two main architectures where information from multiple sources is handled in order to produce single and more accurate information when compared to individual contributing sources, namely *i)* centralized and *ii)* distributed. Figure 1 illustrates generic structures of both architectures where the main difference is a central processing unit handling all the measurements from multiple sources to come with a single piece of information which is encapsulate in the term "state". State in this context is assumed to contain its uncertainty information, i.e., covariance, as well. On the other hand, in the distributed architecture, each source has its own processing unit to produce the local state, then the local states are shared between the sources for the global/fused state. In this distributed representation, it is assumed that each local unit also acts as a global unit, however this is not necessary and more common approach is to send local states to a central unit for processing and producing the global state.

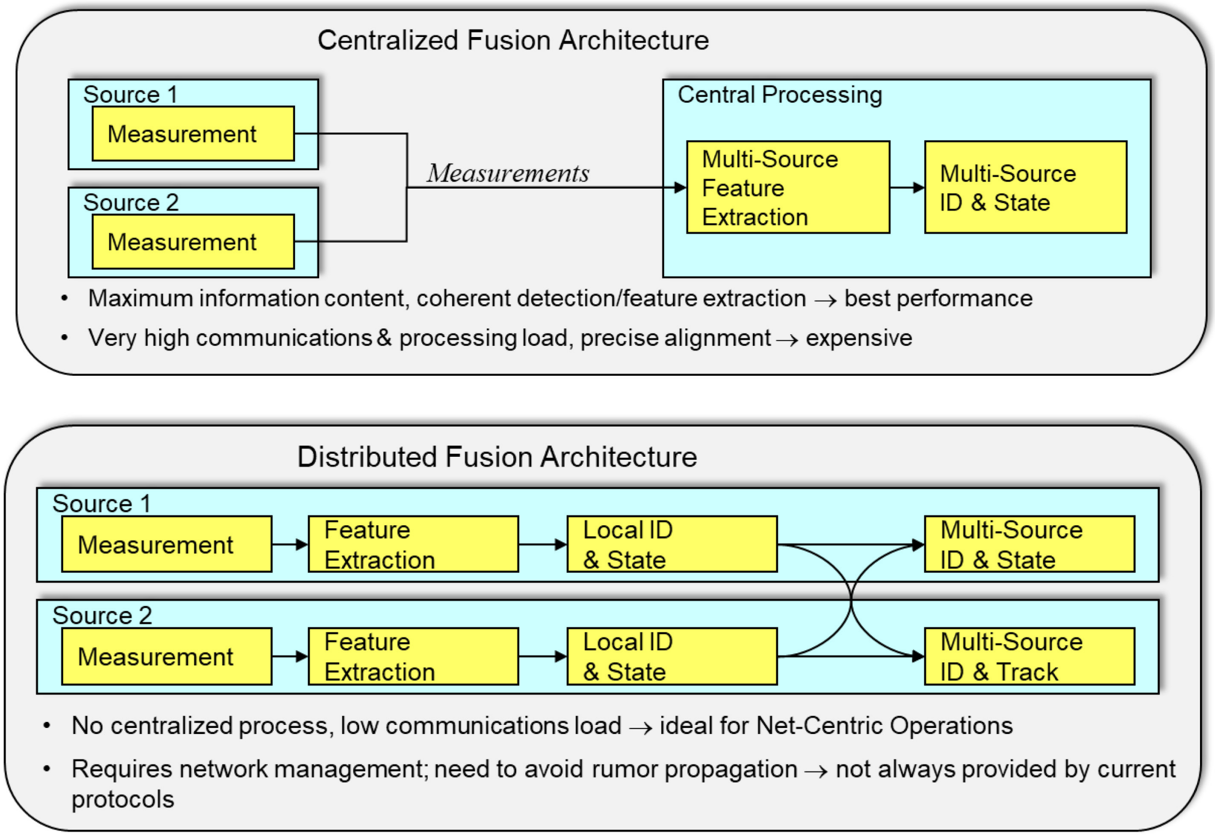
Centralized Fusion Architecture
Source 1
Measurement
Source 2
Measurement
Measurements
Central Processing
Multi-Source Feature Extraction
Multi-Source ID & State
• Maximum information content, coherent detection/feature extraction → best performance
• Very high communications & processing load, precise alignment → expensive
Distributed Fusion Architecture
Source 1
Measurement
Feature Extraction
Local ID & State
Multi-Source ID & State
Source 2
Measurement
Feature Extraction
Local ID & State
Multi-Source ID & Track
• No centralized process, low communications load → ideal for Net-Centric Operations
• Requires network management; need to avoid rumor propagation → not always provided by current protocols

Figure 1: Main Fusion Architectures.

In the centralized architecture, sending the information from multiple sources to a central unit, then allowing all information being processed together will result in near-optimal solution as in this way almost no information loss shall occur. On the other hand, such an architecture requires a high bandwidth, mostly dedicated, communications channel. However, having a distributed information processing units working locally, then sharing the processed output within the network is more attractive and practically achievable. When centralized and distributed architectures are compared:

- Communication channel bandwidth may not be enough to send all information to a central processing unit.
- In the centralized approach the performance of the central processing unit may be greatly affected by the problems encountered in the local nodes. On the other hand, in the distributed system each local unit produces a processed output, hence even with a single operational local unit an output is guaranteed.
- Distributed architecture provides modularity and it is easier to add or omit a local unit in comparison to the centralized approach.

Nevertheless, the biggest disadvantage of distributed processing is that the information processed by a local unit is never independent of the information being processed at another unit, which creates a big obstacle when the two information from different local units is fused. One way to tackle this problem is tracking the common information across local units, which is easier said than done. Ignoring this common information is another way to deal with this problem, however, in this case common information will be used more than once during fusion creating overconfidence and inconsistency with the actual situation.

There are two types of common information to be considered in distributed fusion. Measurement errors by individual sources within the distributed sensor network could be considered independent. However, when the observed target (the platform that navigation is performed in a PNT application) is common then the common process noise, through modelling, will create the common information. In other words, filtering is performed on common process noise which requires cross-state correlation to be taken into account during the fusion process. In addition to the correlation caused by the common process noise, existence of a feedback mechanism cerates another source for common information. In some applications, the fused state is fed back to the local units to replace locally estimated states. In this case, the common information presents itself in the form of common prior distribution and will have to be mitigated while fusion. Four different approaches could be considered to address the problems caused by common information in a distributed fusion network [2]:

- Naive fusion
- Cross-correlation fusion
- Information-matrix fusion
- Covariance intersection.

There are additional approaches for the same problem such as tracklet fusion and maximum a posteriori (MAP) fusion presented in the literature. However, when studied, although derived differently, it is apparent that the tracklet fusion mathematically converges to the information-matrix fusion approach. On the other hand, MAP fusion in which both common prior distribution and cross-correlation between local estimates are taken into account, is one of most complicated fusion approach requiring high level of communication between local units with high computational load. Therefore, these two approaches are not covered here.

2.1 Naïve Fusion

In the naïve fusion both common process noise (cross-correlation error) and common prior distribution is ignored. This is the simplest approach in distributed fusion where the fused state and the corresponding covariance is given in Eq 1, where $\hat{x}$, $\hat{x}_i$ and $\hat{x}_j$ are the fused and individual state estimates respectively and P, P_i and P_j are the corresponding covariances.

$$\hat{x} = P\left(P_i^{-1}\hat{x}_i + P_j^{-1}\hat{x}_j\right) \qquad (1)$$
$$P = \left(P_i^{-1} + P_j^{-1}\right)^{-1}$$

Eq. 1 indicates that the fused covariance is the inverse of sum of inverse of individual covariances. Considering the fact that covariance matrices are positive definite, then, for the same confidence interval, error volume that P matrix represents will always be smaller than individual covariance matrix being fused. For instance if $P_i = P_j$ then, $P = 0.5P_i$, indicating that *rms* error for the fused estimate is approximately reduced by one third when compared to the *rms* error of individual estimates that enter the fusion process. Such a reduction in uncertainty is not realistic especially when the filter model does not match the platform model, for instance when the platform maneuvers.

If the origin of the states to be fused is not known, there is a simple test to perform to determine if those states correspond to the same source. In such cases a similarity metric, given by Eq. 2, is calculated. Under the assumption that state estimations are Gaussian distributed, then C_{ij} is a chi-squared random variable with degree of freedom equal to the number components in the state matrix allowing for selecting a threshold with respect to a confidence level.

$$C_{ij} = \left(\hat{x}_i - \hat{x}_j\right)^T \left(P_i + P_j\right)^{-1} \left(\hat{x}_i - \hat{x}_j\right) \qquad (2)$$

2.2 Cross-Correlation Fusion

Cross-correlation fusion takes into account the common process noise that is ignored in naïve fusion, however, common prior distributions are still ignored. Fused state estimate and the corresponding covariance are given in Eq. 3

$$\hat{x} = \hat{x}_i + \left(P_i - P_{ij}\right)\left(P_i + P_j - P_{ij} - P_{ji}\right)^{-1}\left(\hat{x}_j - \hat{x}_i\right) \qquad (3)$$
$$P = P_i - \left(P_i - P_{ij}\right)\left(P_i + P_j - P_{ij} - P_{ji}\right)^{-1}\left(P_i - P_{ji}\right)$$

Careful eyes will notice that Eq. 3 is equivalent to Eq. 1 (naïve fusion) when the cross-correlation terms P_{ij} and P_{ji} ($P_{ij} = P_{ji}^T$) are zero. [4] reported that, based on an application where the cross-correlation approach has 7% larger uncertainty ellipse compared to the optimal global estimation, when applied correctly, the performance of cross-correlation fusion approach is very close to optimal. For the cross-covariance approach to be applied correctly, each local unit needs to calculate the cross-correlation term and share it with the central processing unit. Eq. 4 is used to calculate the cross-correlation in a Kalman filter where *K* is Kalman filter gain matrix, *H* is the measurement matrix and *Q* is the process noise matrix. As it will be explained in the next section, for nonlinear state and measurement equations, linearized *K* and *H* matrices could be used. An important issue related to cross-correlation fusion is that, for nonlinear cases, in addition to the states, corresponding covariance matrices and the calculated cross-covariance terms, linearized *K* and *H* matrices must also be communicated for the fusion process.

$$P_{ij}(k) = \left(I - K^i(k)H^i(k)\right)F(k-1)P_{ij}(k-1)F(k-1)^T\left(I - K^j(k)H^j(k)\right)^T + \qquad (4)$$
$$\left(I - K^i(k)H^i(k)\right)Q(\mathrm{k}-1)\left(I - K^j(k)H^j(k)\right)^T$$

The similarity metric to determine if states to be fused correspond to the same source when cross-correlations are taken into account is given in Eq. 5. Similarly, under the assumption that state estimations are Gaussian distributed, then is a chi-squared random variable with degree of freedom equal to the number components in the state matrix allowing for selecting a threshold with respect to a confidence level.

$$C_{ij} = \left(\hat{x}_i - \hat{x}_j\right)^T \left(P_i + P_j - P_{ij} - P_{ji}\right)^{-1} \left(\hat{x}_i - \hat{x}_j\right) \tag{5}$$

2.3 Information Matrix Fusion

Information matrix fusion approach assumes that common prior distribution is the dominant common information and common process noise can be ignored. Estimated state and the corresponding covariance is calculated through Eq. 6.

$$\begin{aligned} \hat{x} &= P\left(P_i^{-1}\hat{x}_i + P_j^{-1}\hat{x}_j - \overline{P}^{-1}\overline{x}\right) \\ P &= \left(P_i^{-1} + P_j^{-1} - \overline{P}^{-1}\right)^{-1} \end{aligned} \tag{6}$$

where the common prior information is represented by $\overline{x}$ and $\overline{P}$. The information matrix fusion approach is only slightly more complicated than naïve fusion. If the common priors were negligible, the information fusion becomes the naïve fusion rule. This is consistent with the assumption that the local estimates are conditionally independent [[2], Ch. 17]. Also, there is no need to maintain cross-covariance information between the local nodes. This significantly reduces communication bandwidth requirements. This approach is optimal when there is no (significant) process noise—the target dynamic model is deterministic, or each local update is transmitted to the central processing center at each update (full-rate communications).

The similarity metric to determine if states to be fused correspond to the same source when information matrix fusion approach is employed is given in Eq. 7.

$$C_{ij} = \left(\hat{x} - \hat{x}_i\right)^T \left(P_i\right)^{-1} \left(\hat{x} - \hat{x}_i\right)^T + \left(\hat{x} - \hat{x}_j\right)^T \left(P_j\right)^{-1} \left(\hat{x} - \hat{x}_j\right) + \left(\hat{x} - \overline{x}\right)^T \left(\overline{P}\right)^{-1} \left(\hat{x} - \overline{x}\right) \tag{7}$$

2.4 Covariance Intersection

Unlike the approaches detailed above Covariance Intersection (CI) approach does not make any conditional independence between local sources. CI explicitly recognizes that local state estimates are indeed correlated and was pro conditional independence is “an extremely rare property”. Thus, performing fusion based on this assumption will not lead to a robust fused state. The approach defines robustness through Eq. 8

$$P_{CI} - P_M > 0 \tag{8}$$

in which P_{CI} represents the covariance obtained employing CI and P_M represents the covariance obtained through centralized fusion utilizing full information. On the condition that the difference matrix is positive definite, the CI approach seeks for the P_{CI} matrix that will result in the smallest uncertainty volume, through Eq. 9.

$$\begin{aligned}
&\hat{x} = P_{CI}\left(wP_i^{-1}\hat{x}_i + (1-w)P_j^{-1}\hat{x}_j\right) \\
&P_{CI} = \left(wP_i^{-1} + (1-w)P_j^{-1}\right)^{-1} \\
&w \in [0,1]
\end{aligned} \tag{9}$$

The CI method calculates the weight w, using the optimization defined in Eq. 10, that will lead to the P_{CI} matrix with the smallest uncertainty volume.

$$\underset{w\in[0,1]}{\arg\min}\left(\det\left(P_{CI}\right)\right) \tag{10}$$

Alternatively, an optimization approach employing the trace of the P_{CI} matrix could be used for reducing the computational load, Eq. 11.

$$\underset{w\in[0,1]}{\arg\min}\left(\operatorname{trace}\left(P_{CI}\right)\right) \tag{11}$$

In the CI approach, the weight w is calculated using Eq. 10 oar Eq. 11, then Eq. 9 is employed to calculate the fused state and its covariance. Close examination of Eq. 9 will reveal that CI is very similar in form to naïve fusion algorithm. The difference is that covariance matrices of the states to be fused is multiplied by w and $(1-w)$ in the CI approach. For instance for $w = 0.5$, the state estimate is the same as in the naïve fusion, but the covariance error would be twice as large, which eliminates the false over confidence caused by the common information.

3.0 INTEGRATION ALGORITHMS

An integration algorithm is expected to track the status of dynamic systems represented in the state vector when subject to random disturbances. The Kalman filter [5] does exactly that. The name filter stems from dealing with noisy measurements but it is in fact a real time estimator. Its popularity is due, in large part, to the fact that so many practical estimation problems can be modeled and solved in this way, and the fact that the Kalman filter has performed so well in so many of these applications. It has been justifiably labelled "navigation's integration workhorse" [6] because it has become an essential part of modern navigation systems – especially for integrating navigation systems as disparate as GNSS and inertial navigation system (INS) [7].

This section outlines the Kalman filter and two modifications to the Kalman filter to deal with nonlinearity, namely the extended Kalman filter (EKF) and unscented Kalman filter (UKF). Before giving details about the Kalman filter let us first describe the system state space model given by Eq. 12 and Eq. 13 that we will use in the filter. In this model x represents the time varying state parameter (position, velocity, acceleration, time drift, etc.) and z represents the measurements observed at discrete times that are used to estimate the state.

$$x_k = f(x_{k-1}, v_k) \tag{12}$$

$$z_k = h(x_k, \mathrm{w}_k) \tag{13}$$

The state is assumed to be a Markov process and evolving according to Eq. 12 and Eq. 13 describes the measurements. v_k and w_k are process and measurement noise respectively, assumed to be random variables with known distribution and independent of each other. Moreover, it is assumed that the functions f and h are known.

3.1 Kalman Filter

When the system dynamics are linear and noises are Gaussian distributed then Kalman filter is the optimal minimum mean square error (MMSE) state estimator [8]. For the dynamic system described by Eqs. 12 and 13, under the assumption that f and h are linear functions and the noises are additive zero mean white Gaussian, system state space representation can be re-written as

$$x_k = F_{k-1}x_{k-1} + v_{k-1} \tag{14}$$

$$z_k = H_k x_k + \mathrm{w}_k \tag{15}$$

where,

$$E\left[v_k v_k'\right] = Q_k \tag{16}$$

$$E\left[w_k w_k'\right] = R_k \tag{17}$$

Also, it assumed that F , H , Q and R are known time invariant matrices, hence the time index drops. Then, the Kalman filter equations for the system state space model described by Eqs. 12 and 13 are as follows;

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1} \tag{18}$$

$$P_{k|k-1} = FP_{k-1|k-1}F' + Q \tag{19}$$

$$\hat{z}_{k|k-1} = Hx_{k|k-1} \tag{20}$$

$$S_k = HP_{k|k-1}H' + R \tag{21}$$

$$K_k = P_{k|k-1}H'S_k^{-1} \tag{22}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(\mathrm{z} - \mathrm{z}_{k|k-1}) \tag{23}$$

$$\hat{P}_{k|k} = P_{k|k-1} - K_k S_k K_k \tag{24}$$

The state and its covariance are predicted based on the estimated state and its covariance at time *k*-1 using Eqs. 17 and 18. Kalman gain is calculated through Eq. 22 and state estimation and the corresponding covariance is obtained using Eqs. 23 and 24.

Some Properties of the Kalman filter [8]:

- Kalman filter is MMSE and MAP optimal and the estimates are unbiased when state and measurements are linear functions and noises are normally distributed.
- On the other hand, when the state and measurement equations are linear but the noise distributions are not Gaussian, then the Kalman filter is MMSE optimal but MAP optimal. Also, the estimation is biased.
- When the linearity and normality conditions hold, the Kalman filter is an efficient estimator, i.e., posterior covariance coincides with the Cramer Rao Lower Bound (CRLB).
- Kalman filter is not robust because of assumptions on noise.
- The filter is sensitive to modelling errors.

3.2 Extended Kalman Filter

When the functions f and/or h in the system state space model given by Eqs. 12 and 13 are nonlinear, but the assumptions about the process and measurement noises are still valid, then the standard Kalman filter cannot be used. However, the advantages of Kalman filter could still be obtained through some modification to the standard filter. Linearization of state and/or measurement functions through Taylor series expansion leads to a suboptimal estimator called the extended Kalman filter (EKF). Although several orders of expansion is possible, due to the resulting computational complexity and the fact that gain obtained with higher order expansion is minimal, usually a first order Taylor expansion is employed. EKF works rather well when the nonlinearity is not severe and even higher order expansion does not help with the increasing nonlinearity level.

First order Taylor series expansion for the functions f and h are given in Eq. 25 and Eq. 26 respectively.

$$F_k = \left.\frac{\partial f}{\partial \hat{x}}\right|_{\hat{x}_{k-1}} \tag{25}$$

$$H_k = \left.\frac{\partial h}{\partial \hat{x}}\right|_{\hat{x}_{k|k-1}} \tag{26}$$

The most common problem with the use of EKF is that the filter is sensitive to selection of modeling parameters and may quickly diverge due to errors caused by linearization.

3.3 Unscented Kalman Filter

Another, Kalman based MMSE estimator to deal with nonlinear system dynamics is the unscented Kalman filter (UKF) [9], [10]. UKF deviates from the standard Kalman filter and EKF by dropping Gaussian distributed noise assumption. The idea behind UKF is the intuition that estimating a distribution would be easier than estimating a nonlinear function or transform. The approach deterministically assigns a set of points that represent the sample mean $\overline{x}_k$ and sample covariance P_k, and puts these points through the nonlinear transform to obtain the posterior state and the corresponding covariance. For a dynamic system with a state vector x_k of n dimension, $2n+1$ deterministically assigned sigma point will be needed.

3.3.1 The UKF Algorithm

Assume that a state variable x with mean $\bar{x}$ and covariance P changes its state in time according to the dynamic equations defined by Eqs. 27 and 28.

$$x_k = f(x_{k-1}) + v_{k-1} \tag{27}$$

$$z_k = h(x_k) + \mathrm{w}_k \tag{28}$$

where v_k and w_k are process noise with a known covariance Q_k and measurement noise with a known covariance R_k respectively.

Step 1: Calculate the sigma points

At time $k-1$ deterministic sample points and the associated weights are determined through Eqs. 29-34.

$$\chi_{0,k-1} = \bar{x}_{k-1} \tag{29}$$

$$\chi_{\mathrm{i},k-1} = \bar{x}_{k-1} + \left(\sqrt{(L+\gamma)P_{k-1}}\right)_i, \qquad i = 1,2,\ldots,L \tag{30}$$

$$\chi_{\mathrm{i},k-1} = \bar{x}_{k-1} - \left(\sqrt{(L+\gamma)P_{k-1}}\right)_{i-L}, \qquad i = L+1, L+2,\ldots,2L \tag{31}$$

$$w_0^{(c)} = \frac{\gamma}{L+\gamma} \tag{32}$$

$$w_0^{(c)} = \frac{\gamma}{L+\gamma} + \left(1-\alpha^2+\beta\right) \tag{33}$$

$$w_i^{(m)} = w_i^{(c)} = \frac{1}{2(L+\gamma)}, \qquad i = 1,2,\ldots,L \tag{34}$$

where χ_{i} and w_{i} are the sigma points and the associated weights respectively. γ is the scaling parameter calculated as,

$$\gamma = \alpha^2(L+\kappa) - L \tag{35}$$

where L is the state vector dimension, α is a small positive number that determines how the sigma points will distributed around $\bar{x}$ ($10^{-4} \le \alpha \le 1$), κ is the second scaling parameter whose value is usually taken as zero and β is the parameter to take the prior distribution of x into account ($\beta = 2$ is the optimal value when the prior distribution is Gaussian). Also, $\left(\sqrt{P}\right)_i$ represents i^{th} column of the square root of the covariance matrix.

Step 2: Time update

State prediction and its covariance is calculated by replacing the determined signa points in functions f and h as follows;

$$\chi_{i,k|k-1} = f\left(\chi_{\mathrm{i},k-1}\right) \tag{36}$$

$$x_{k|k-1} = \sum_{i=0}^{2L} w_i^{(m)} \chi_{i,k|k-1} \tag{37}$$

$$P_{k|k-1} = \sum_{i=0}^{2L} w_i^{(\mathrm{c})} \left(\chi_{i,k|k-1} - x_{k|k-1}\right)\left(\chi_{i,k|k-1} - x_{k|k-1}\right)^T + Q_k \tag{38}$$

$$\chi^*_{i,k|k-1} = \left[\chi_{0:i,k|k-1} \quad \chi_{0,k|k-1} + v\sqrt{Q_k} \quad \chi_{0,k|k-1} - v\sqrt{Q_k}\right]_i \tag{39}$$

$$z_{i,k|k-1} = h\left(\chi^*_{i,k|\mathrm{k}-1}\right) \tag{40}$$

$$z_k^- = \sum_{i=0}^{2M} w_i^{*(m)} z_{i,k|k-1} \tag{41}$$

$$P_{zz,k} = \sum_{i=0}^{2M} w_i^{*(c)} \left(z_{i,k|k-1} - z_k^-\right)\left(z_{i,k|k-1} - z_k^-\right)^T + R_k \tag{42}$$

$$P_{xz,k} = \sum_{i=0}^{2M} w_i^{*(c)} \left(\chi_{i,k|k-1} - x_{k|k-1}\right)\left(z_{i,k|k-1} - z_k^-\right)^T \tag{43}$$

where, $v = \sqrt{L+\gamma}$. Additionally, moving from Eq. 38 to Eq. 39 the sigma points are augmented with additional points derived from the matrix square root of the process noise covariance. This requires setting $M = 2L$ and recalculating the weights $w_i^{*(m)}$ and $w_i^{*(c)}$ as described in Step 1.

Step 3: State update using the measurements

Kalman gain is calculated and state and its covariance are updated as in the standard Kalman filter.

$$K_k = P_{xz,k} P_{zz,k}^{-1} \tag{44}$$

$$x_k = z_{k|k-1} + K_k \left(z_k - z_k^-\right) \tag{45}$$

$$P_k = P_{k|k-1} - K_k P_{yy,k} K_k^T \tag{46}$$

4.0 SIMULATION RESULTS

In this section some simulation results are presented in order to illustrate the performance of fusion and integration algorithms covered in this paper. The aim here is not to be comprehensive but to give some understanding towards the comparative performance of these approaches for the selected scenarios.

4.1 Performance of Fusion Architectures/Algorithms

A passive multistatic scenario is considered to compare the performance of various fusion architectures/algorithms. The scenario comprises two transmitters and two receivers in which locations of the receiver 1, transmitter 1 and receiver 2, transmitter 2 are given as $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$, $\begin{bmatrix}20000 & 0 & 0\end{bmatrix}^T$, $\begin{bmatrix}-5000 & 0 & 0\end{bmatrix}^T$ and $\begin{bmatrix}-20000 & 0 & 0\end{bmatrix}^T$ respectively. Receivers are supposed make observations in accordance with the following error values, standard deviation of range, angle and range rate errors are 100m, 1deg and 1m/s respectively. Also, measurement frequency is 1Hz, probability of detection is 1 and probability of false alarm is 0. Moreover, receivers are assumed to be synchronized and sending their observations to the fusion center without a delay.

In this setting the platform is assumed to be moving for 50s according to the white noise acceleration model at 45deg heading with respect to the horizontal axis and 141.42m/s initial velocity. Initial state $\begin{bmatrix}x & vx & y & vy\end{bmatrix}^T$ is taken as $\begin{bmatrix}10000 & 100 & 15000 & 100\end{bmatrix}^T$. Platform altitude is assumed constant and known at all times. The scenario is depicted in Figure 2.

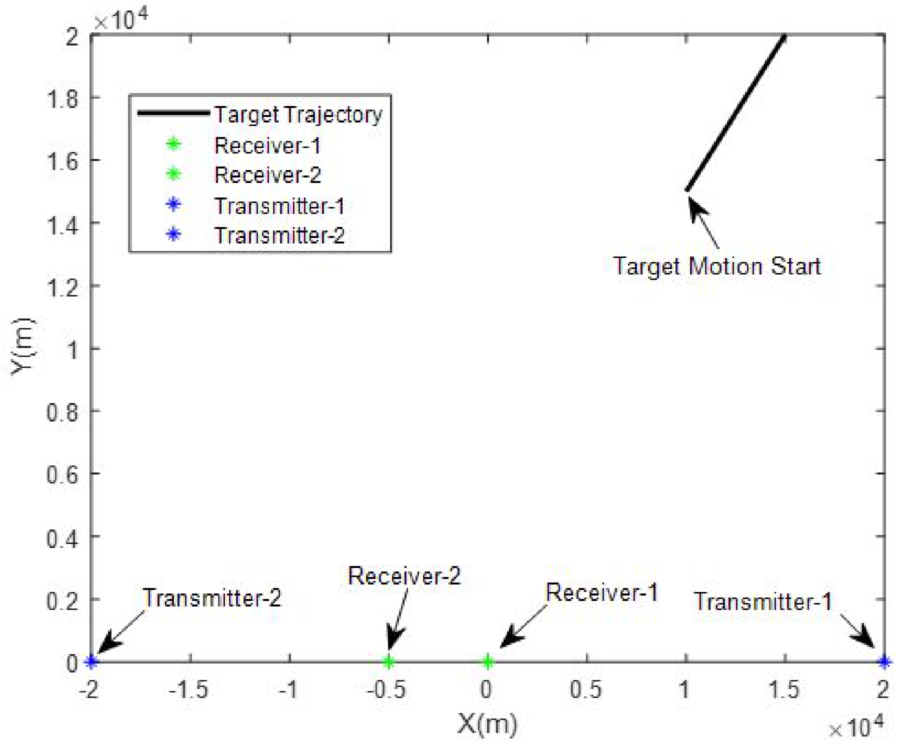

Figure 2: Scenario for testing fusion algorithms.

An extended Kalman filter has been employed for this scenario, where all fusion algorithms have been tested with varying process noise values (intensity of acceleration standard deviation in this case) both when feedback from the fusion center back to the nodes is available and not available. Results, in terms of RMS position error, are presented in Table 1.

Table 1: Fusion algorithms performance.

	Without Feedback							With Feedback	
Process Noise Std	**Filter 1**	**Filter 2**	**Central Fusion**	**Naïve Fusion**	**CI**	**IM Fusion**	**CC Fusion**	**IM Fusion**	**CI**
1	125.8	125.8	41.6	34.3	48.4	48	44.5	41.6	48.1
5	160.5	160.5	42.1	35.4	50.1	49.6	67.5	42.1	48.6
10	185.4	185.4	37.6	37.6	53.1	52.7	75	42.9	49.2
20	229.2	229.2	45.9	45.9	64.9	64.4	97.3	46.9	54.1
40	267	267	53.2	56.3	79.6	78.6	109.8	53.2	61.6

4.2 Performance of Integration Algorithms

In this section performance of integration algorithms are compared using a simple scenario in which the nonlinearity is introduced within the measurement equation. Performance of the standard Kalman filter is not included in the discussions for brevity, so EKF and UKF are compared. The scenario considers a lateral movement where the measurements are taken by a radar in range and azimuth. Since the filters are designed to process measurements in Cartesian coordinates, radar measurements will have to be transformed, hence nonlinearity is introduced. Figure 3 shows how the nonlinear measurements occur and Figure 4 illustrates the scenario.

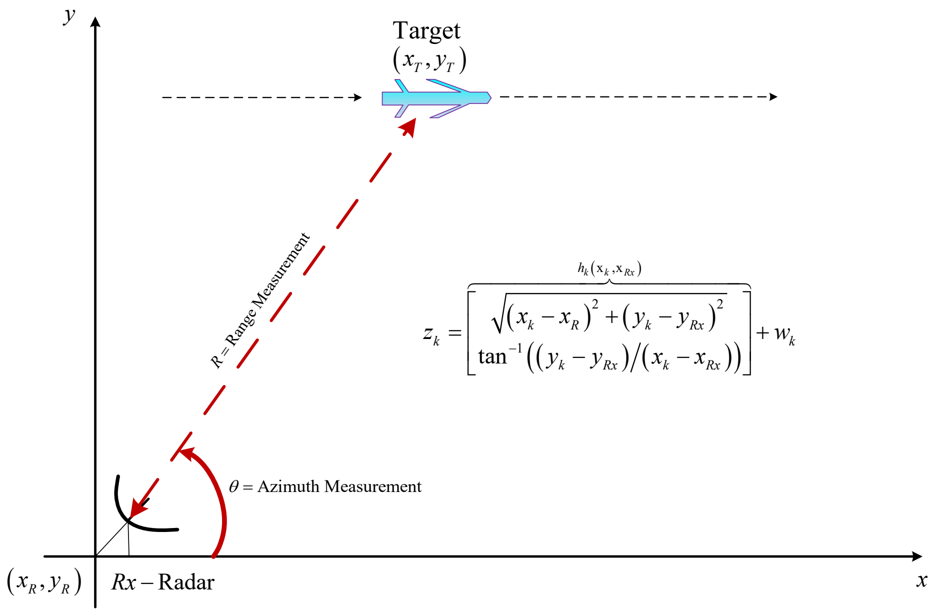
y
Target
(x_T, y_T)
R = Range Measurement
$z_k = \overbrace{\begin{bmatrix} \sqrt{(x_k - x_R)^2 + (y_k - y_{Rx})^2} \\ \tan^{-1}((y_k - y_{Rx})/(x_k - x_{Rx})) \end{bmatrix}}^{h_k(x_k, x_{Rx})} + w_k$
θ = Azimuth Measurement
(x_R, y_R)
Rx – Radar
x

Figure 3: Nonlinear measurements.

Since we are only interested in the performance comparison, for fairness, both EKF and UKF have been designed to model target motion perfectly. Nonlinearity introduced by measurement transformation is not too severe, hence, the RMSE performances for both filters are similar as seen in Figure 5. On the other hand UKF convergence is faster when compared to EKF. Similar results can also be observed when consistency of the filters are analyzed. Consistency indicates whether actual errors are consistent with the variances calculated by the estimator and absence of large deviations from confidence interval borders is a sign of consistent estimation.

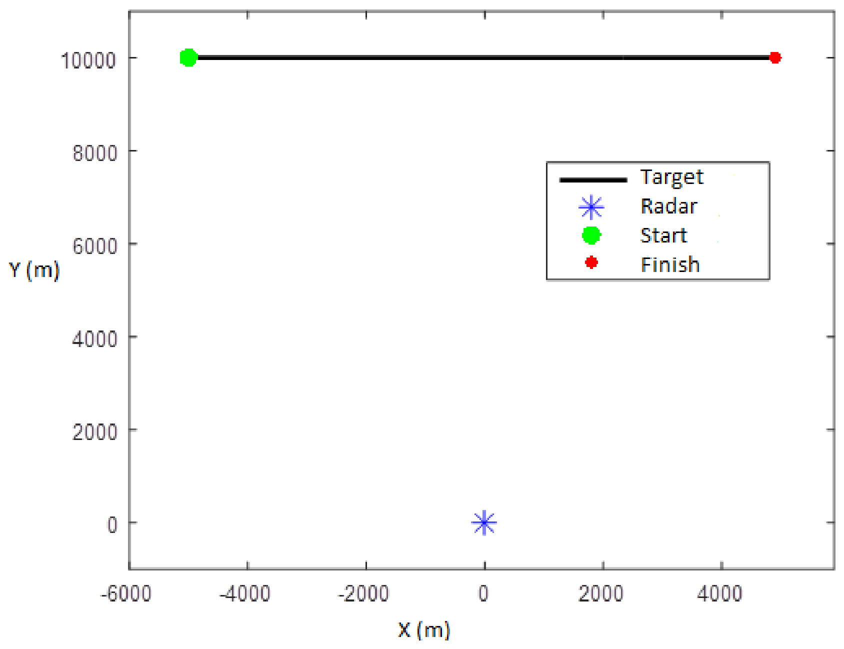

Figure 4: Scenario.

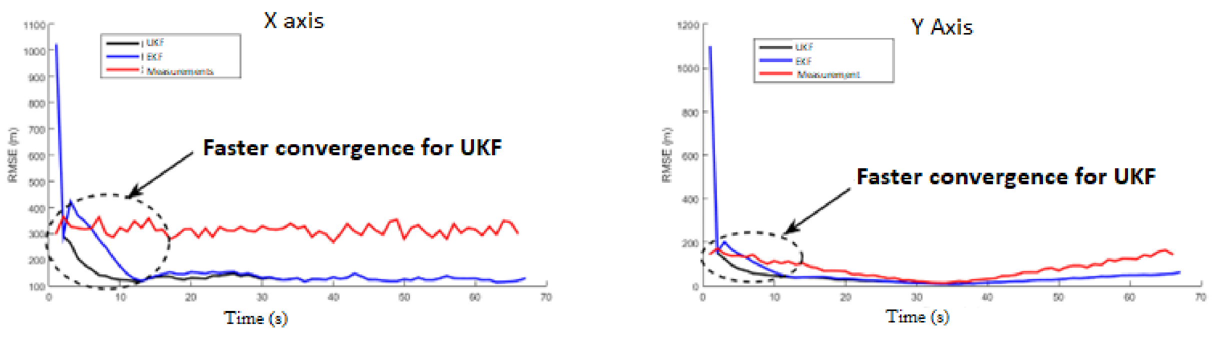

Figure 5: RMSE errors.

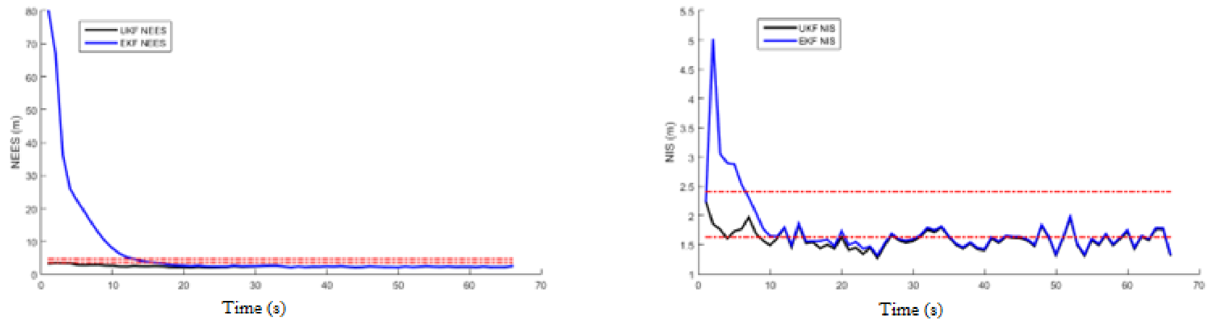

Figure 6: Consistency analysis.

5.0 CONCLUSIONS

This paper is a first part of a two-part paper aiming to introduce PNT fusion architecture/algorithms and integration algorithms. Fundamental fusion architectures and algorithms are introduced here as well as basic integration algorithms. Performance comparisons are given based on simulations. More elaborate integration algorithms, such as the particle filter, will be covered in part 2.

6.0 REFERENCES

[1] US Army Positioning, Navigation, and Timing (PNT) Reference Architecture, v1.0, 1 July 2018.

[2] Liggins II, Martin, David Hall, and James Llinas, eds. Handbook of multisensor data fusion: theory and practice. CRC press, 2017.

[3] Drummond, Oliver E. “Track and tracklet fusion filtering.” Signal and Data Processing of Small Targets 2002. Vol. 4728. SPIE, 2002.

[4] Bar-Shalom, Yaakov, X. Rong Li, and Thiagalingam Kirubarajan. Estimation with applications to tracking and navigation: theory algorithms and software. John Wiley & Sons, 2001.

[5] Kalman, Rudolph Emil. “A new approach to linear filtering and prediction problems.” (1960).

[6] Levy, Larry J. “The Kalman filter: navigation’s integration workhorse.” GPS World 8.9 (1997): 65-71.

[7] Grewal, Mohinder S., Angus P. Andrews, and Chris G. Bartone. Global navigation satellite systems, inertial navigation, and integration. John Wiley & Sons, 2020.

[8] Bar-Shalom, Yaakov, X. Rong Li, and Thiagalingam Kirubarajan. Estimation with applications to tracking and navigation: theory algorithms and software. John Wiley & Sons, 2001.

[9] Simon Julier, Jeffrey Uhlmann, and Hugh F. Durrant-Whyte, “A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators”, IEEE Transactions on Automatic Control, Vol. 45, No. 3, March 2000.

[10] Simon Julier, “The Scaled Unscented Transform”, Proceedings of the American Control Conference Anchorage, AK May 8-1 0,2002.