Abstract – Fusing out-of-sequence information is a very important problem for multi-sensor target tracking systems. The challenge is in dealing with measurements that arrive from the various sensors at a central processor out-of-sequence; that is, signals arriving with a measurement relating to a time previous to the time of the current state. The problem of how to deal with these updates has received much attention in recent years and has been solved optimally and sub-optimally by various authors. Most of the solutions, however, treated only the problem of updating a track by out-of-sequence measurements (OOSM), but did not deal with the updating of a central track by local tracks. In this paper we adopt three solutions proposed by Bar-Shalom, for the OOSM problem, and adapt them to solving the out-of-sequence Track (OOST) problem. We then demonstrate their application to the case of track-to-track fusion.

Keywords: Track Fusion, Out-of-Sequence Information, Out-of-Sequence Measurements, Out-of-Sequence Tracks, OOSI, OOSM, OOST, GMTI.

1 Introduction

Multisensor tracking systems usually operate in a centralized manner. There are several methods for the fusion of track data in this architecture [1]. In the ideal case all the sensor level tracks are updated at about the same time. Thus, the global (fused) tracks are updated promptly after the new states from all sensors have been received. However, in most real cases each sensor collects and transmits its data without synchronization with the other sensors. Therefore, the central track is updated each time an update is received from one of the sensors. Central tracks are fused with the sensor tracks.

The out-of-sequence tracks (OOST) problem [2] arises because the time delays in the arrival of the updated states from various sensors to the central tracker are usually different. Therefore, an updated state may reach the central tracker out-of-sequence. It may occur that a central track was already updated up to a given time and then, an updated state from a "late" sensor arrives with an earlier time stamp. This problem is related to the more general well known out-of-sequence measurements (OOSM) problem [3] where the measurements, (rather then the updated states) are received out-of-sequence by the central tracker.

Many OOSM filtering algorithms were developed in recent years (see for example [3] – [7]). These algorithms update the latest states and covariance matrices using the OOSM. In ref. [4] three algorithms are presented for updating the state and the corresponding covariance matrix with an OOSM that lies between the last two measurements, the one-step-lag case, where one algorithm is optimal and the two other are suboptimal. In ref. [5] these three algorithms were generalized to the case of l-step lags OOSM. The idea was to introduce an equivalent measurement that represents all measurements with time stamps later than the OOSM, thus reducing the l-step lag to an effective one step lag scenario. Actually, all three algorithms in [5] are sub-optimal for the multiple-lag problem.

In comparison with the OOSM problem the OOST problem has received little attention and it is not fully understood [2]. The unknown cross correlations between sensor level tracks and the central fused track is the crucial point. Therefore, not many approaches were developed for dealing with the OOST problem. For examples of various approaches to the problem see ref. [2]. Ref. [2] formulates a Bayesian solution to the OOST problem. Ref. [8] developed the OOST filtering using a decorrelated pseudo measurement approach.

In this paper we assume that each sensor transmits to the central tracker only the last updated state for each track and the corresponding covariance matrix. The central tracker fuses this updated state with its own predicted state to obtain a new central update state with its corresponding covariance matrix. This straightforward track-to-track fusion algorithm is presented in Section 2.

In this paper we generalize the fusion algorithm presented in Section 2 to treat also OOST scenarios. We adopt the three OOSM algorithms, derived in ref. [4], and adapt them to the OOST problem. The formulation of the problem and our approach to its solution are presented in Section 3. In Section 4, we derive the equations of the three algorithms for the OOST problem and point out the differences between these equations and the corresponding equations in the OOSM problem [4].

We apply the three algorithms on two GMTI scenarios and evaluate their effectiveness using different time delays. The numerical results are presented in Section 5. Conclusions are given in Section 6.
2 Algorithm for track-to-track fusion

In this section we present a simple algorithm for the centralized system track-to-track fusion. This algorithm is applicable to cases in which the central tracker is updated after all new updated states, from the last "scan" of one of the sensors are received.

Assume that all the tracks of the central tracker were already updated up to time $k$. Assume also that updated states and covariance matrices from one of the sensors arrive after this time. Consider one sensor update state $x'(\kappa \mid \kappa)$ and its covariance matrix $P'(\kappa, \kappa)$ that arrive with time stamp $\kappa > k$ i.e. in sequence. To update the appropriate central state $x(k \mid k)$ we extrapolate the central state up to the time $\kappa$ as

$$x(\kappa \mid k) = F(\kappa, k)x(k \mid k)$$

where $F(\kappa, k)$ is the state transition matrix from time $k$ to time $\kappa$, i.e. forward transition. Similarly, the central track last covariance $P(k, k)$ has to be extrapolated to the time $\kappa$ through

$$P(\kappa, k) = F(\kappa, k)P(k, k)F(\kappa, k)' + Q(\kappa, k)$$

where $Q(\kappa, k)$ is the process noise covariance from time $k$ to $\kappa$. We assume, for simplicity, that the process noise is zero-mean and white and that its covariance matrix is the same for the sensor and the central tracker. Then we apply the fusion equations from ref. [3] and obtain the new fused (central) state as

$$x(\kappa \mid \kappa) = x(\kappa \mid k) + P(\kappa, k)(P(\kappa, k) + P'(\kappa, \kappa))^{-1}(x'(\kappa \mid \kappa) - x(\kappa \mid k))$$

Similarly, the corresponding fusion covariance matrix is obtained as follows:

$$P(\kappa, \kappa) = P(\kappa, k)(P(\kappa, k) + P'(\kappa, \kappa))^{-1} \times P'(\kappa, \kappa)$$

3 Formulation of the OOST problem

We now deal with the OOST problem related to a system that operates in a centralized manner as described in the previous section. The OOST problem arises when an update state, from one of the sensors, (related to a given central track) arrives out-of-sequence, i.e., the updated state arrives with time stamp $\tau$ where the appropriate central track is already updated up to the later time $k > \tau$. The corresponding last state and covariance matrix of the central track must be updated with the update state and covariance of the arriving sensor track.

This OOST problem is quite common in real multisensor systems, but its solution is not straightforward. In order to solve the OOST problem we have to apply a backward prediction algorithm on the central track and then we can apply the fusion procedure with the sensor update state. However, it is not simple to apply a backward prediction here, since such a retrodiction requires use of the process noise which depends, in this case, on the updated central state [4] which is yet unknown.

In ref. [4], Bar-Shalom addressed the process noise problem, related to the retrodiction of a given track when an "old" measurement is received i.e., the OOSM problem. He developed an optimal algorithm and two suboptimal algorithms to solve the problem for the one-step-lag case. In the next section, we apply these algorithms to the OOST problem, where an "old" updated track is received rather than a measurement.

4 Algorithms for the OOST problem

In this section we show how to implement the algorithms A, B, and C from ref. [4] to solve the OOST problem. Since algorithm C is the simplest, we describe it first and then describe algorithms B and A.

4.1 Algorithm C for OOST

Assume that an updated state of one of the sensors arrives at the central system with time stamp $\tau < k$ (i.e., out-of-sequence). In order to update the central state $x(k \mid k)$ with the sensor state $x'(\tau \mid \tau)$ we interpolate the central state backward to time $\tau$. Similarly, we interpolate the covariance matrix $P(k, k)$ to time $\tau$. Since we "ignore" the process noise covariance in this algorithm the state is interpolated as follows:

$$x'(\tau \mid k) = F(\tau, k)x(k \mid k)$$

where $F(\tau, k)$ is the state transition matrix from time $k$ to time $\tau$ i.e. backward transition. Similarly, the covariance $P(k, k)$ is interpolated:

$$P'(\tau, k) = F(\tau, k)P(k, k)F(\tau, k)'$$

The sum of the covariance matrices at time $\tau$ is defined as:

$$S'(\tau) = P'(\tau, k) + P'(\tau, \tau)$$

The gain used for updating the state and the covariance matrix is:

$$w'(k, \tau) = F^{-1}(\tau, k)P'(\tau, k)S(\tau)^{-1}$$
Finally, the central state that was updated with the sensor state is:

\[ x^c(k \mid \tau) = x(k \mid k) + w^c(k, \tau)[x^c(\tau \mid \tau) - x^c(k \mid k)] \]  \hspace{1cm} (9)

Similarly, the central covariance matrix that was updated with the sensor covariance matrix is:

\[ P^c(k, \tau) = P(k, k) - w^c(k, \tau)P^c(\tau, k)F^{-1}(\tau, k)' \]  \hspace{1cm} (10)

Note that the \( H \) matrix (used in ref. [4] for transformation from the sensor coordinates to the tracker coordinates) is equal to the identity matrix here since the sensor update state and the central tracker state have the same coordinates.

4.2 Algorithm B for OOST

Algorithm B is too a suboptimal solution to the OOST problem. In this algorithm we do take the process noise covariance into account to interpolate the covariance matrix to time \( \tau \), but do not take it into account when interpolating the central state. Therefore, in this algorithm

\[ x^B(\tau \mid k) = F(\tau, k)x(k \mid k) \] \hspace{1cm} (11)

On the other hand, the central track covariance matrix is interpolated to time \( \tau \) as follows:

\[ P^B(\tau, k) = F(\tau, k)[P(k, k) + P^B_v(k, \tau \mid k)] - P^B_v(k, \tau \mid k) - P^B_v(k, \tau \mid k)'F(\tau, k)' \] \hspace{1cm} (12)

where \( P^B_v(k, \tau \mid k) \) is the "net" contribution of the process noise covariance

\[ P^B_v(k, \tau \mid k) = Q(k, \tau) \] \hspace{1cm} (13)

and \( P^B_v(k, \tau \mid k) \) is the cross covariance between the process noise and the predicted state of the central tracker

\[ P^B_v(k, \tau \mid k) = Q(k, \tau) - P(k, k - 1)S^B(k)^{-1}Q(k, \tau) \] \hspace{1cm} (14)

In eq. (14) \( S^B(k) \), the sum of the covariance matrices at time \( k \) (before fusion), is defined as

\[ S^B(k) = P(k, k - 1) + P^B_v(k, k) \] \hspace{1cm} (15)

and the sum at time \( \tau \) before fusion is:

\[ S^B(\tau) = P^B(\tau, k) + P^B_v(\tau, k) \] \hspace{1cm} (16)

The superscript \( s \) in eq. (15) refers to the sensor which was used to update the central track at time \( k \). This sensor is necessarily not the sensor which produced the OOST which we are currently fusing.

The gain for updating the state and the covariance matrix is:

\[ w^B(k, \tau) = P^B_v(k, \tau \mid k)S(\tau)^{-1} \] \hspace{1cm} (17)

where, following ref. [4] we have

\[ P^B_v(k, \tau \mid k) = [P(k, k) - P^B_v(x, \tau \mid k)] \times F(\tau, k)' \] \hspace{1cm} (18)

The final central state, obtained after fusion with the sensor update state is

\[ x^B(k \mid \tau) = x(k \mid k) + w^B(k \mid \tau)[x^c(\tau \mid \tau) - x^B(\tau \mid k)] \] \hspace{1cm} (19)

and, similarly, the final central covariance matrix, obtained after fusion with the sensor data is

\[ P^B(k, \tau) = P(k, k) - P^B_v(k, \tau \mid k)P^B_v(k, \tau \mid k)' \] \hspace{1cm} (20)

4.3 Algorithm A for OOST

Algorithm A is an optimal algorithm, in which we use the process noise covariance to retrodict, both the central state and its covariance matrix to time \( \tau \). Therefore, the equations of this algorithm are the same as those of algorithm B except that equations (11) and (13) are replaced by the following equations:

\[ x^A(\tau, k) = F(\tau, k)x(k \mid k) \times [x(k \mid k) - Q(k, \tau)S^A(k)^{-1}V(k)] \] \hspace{1cm} (21)

and

\[ P^A_{vv}(k, \tau \mid k) = Q(k, \tau) - Q(k, \tau)S^A(k)^{-1}Q(k, \tau) \] \hspace{1cm} (22)

where, \( S^A(k) = S^B(k) \) (eq. (15) ) and \( V(k) \) is the "innovation" vector at time \( k \), i.e.,

\[ V(k) = x^v(k \mid k) - x(k \mid k - 1) \] \hspace{1cm} (23)

The superscript \( s \) in this equation has the same meaning as in eq. (15). The "innovation" vector in eq. (23) is additional data that must be saved and will be used in the next step of the algorithm. As noted above, all other equations of algorithm A are the same as those of algorithm B, except that the superscript
\( B \) is replaced by \( A \). For example, the final central state and its covariance matrix are

\[
x^A(k, \tau) = x(k, k) + w^A(k, \tau)[x^A(\tau, \tau) - x^A(\tau, k)]
\]

\[ P^A(k, \tau) = P(k, k) - w^A(k, \tau)P^A_{xc}(k, \tau)w^A(k, \tau)' \]  

(25)

It is important to emphasize that algorithm \( A \), presented in this subsection, is optimal. In this algorithm the cross-correlation between the central track and the sensor-level track is computed exactly. This cross-correlation is subsequently subtracted in the process of retrodiction from the central state and from its covariance.

5 Numerical results

We applied the three algorithms, described in the previous section, to a simulated GMTI tracking system. In this section we compare the tracking results obtained using each of the algorithms \( A, B, \) and \( C \) described above and those obtained by ignoring the OOST data altogether.

The simulated system consists of two GMTI sensors positioned at 80 degrees relative to each other. The two sensors differ in their revisit times—8 and 15 seconds—but otherwise have the same typical properties, i.e., a probability of detection \( Pd = 0.9 \) and a standard deviation of 5 m in range, 0.002 radians in azimuth and 0.6 m/sec in Doppler. The process noise spectral density for the sensors and the central tracker is \( q = 0.2 m^2 / s^3 \).

The state (of each sensor and of the central tracker) consists of position and velocity in two dimensions i.e., \((x, v_x, y, v_y)\). We apply the fusion procedure to the central tracker states each time a new state from one of the sensors is received. Then we calculate the distance between the real location of the target and the location of the new fused central state. The rms value of this distance serves as a test to evaluate the quality of the various OOST algorithms.

We considered two ground scenarios consisting of a single target moving along a straight line and a circle. The speed of the target, \(|v|\), along its track was taken to be varying in each of the following two scenarios we considered:

Scenario 1: A target moving along a straight line with the velocity changing according to \(|v| = 12 + 4\cos(4t/100)\) in m/sec.

Scenario 2: A target moving in a circular path with the velocity changing according to \(|v| = 9 + 5\cos(4t/100)\) in m/sec.

It is necessary to consider a scenario with a varying velocity for two important reasons. First, this is the realistic case. Second, this motion differs significantly from the linear system model and hence the cross-correlation, related to the process noise covariance, becomes important.

Without fusion, the rms values of the distance between the ground true position of the target and the sensor tracks are 85 m and 90 m for the two sensors of scenario 1, and 95 m and 98 m for the two sensors of scenario 2. In addition, after applying fusion in the ideal case, where there is no out-of-sequence data, the rms value is 11.2 m in scenario 1 and 12.6 m in scenario 2.

OOST events are introduced by enforcing a constant delay of 10 seconds on one sensor (the one with revisit time of 8 seconds). Therefore, the received times in the central system of the data from the last sensor are 18, 26, 34, 42, 50, ... seconds, whereas the real measurement times are 8, 16, 24, 32, 40, ... seconds. However, the data from the other sensor are obtained with no delay i.e., 15, 30, 45, ... seconds. Then, for example, the data that arrived at time 18, 34, and 50 seconds from the first sensor are out-of-sequence.

Typical post-fusion results of the three algorithms for the distance rms values of both scenarios are shown in Table 1.

Table 1 - comparison of post-fusion distance rms values of both scenarios are shown in Table 1.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Example 1</th>
<th>Example 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12.94</td>
<td>16.17</td>
</tr>
<tr>
<td>B</td>
<td>12.99</td>
<td>16.23</td>
</tr>
<tr>
<td>C</td>
<td>14.11</td>
<td>17.60</td>
</tr>
<tr>
<td>ignoring OOST</td>
<td>23.11</td>
<td>30.49</td>
</tr>
</tbody>
</table>

From the comparison in Table 1 we deduce the following important results. First, we see that the OOST data cannot be ignored. Making use of the OOST data by either of the algorithms \( A, B \) or \( C \) significantly improves the tracking results. Second, we see that algorithm \( C \) did not perform as well as algorithms \( A \) and \( B \); The results of tracking using algorithms \( A \) and \( B \) exceed those obtained using algorithm \( C \) by approximately 9%. This is reasonable since algorithms \( A \) and \( B \) make use of the process noise covariance whereas algorithm \( C \) neglects it. Our scenarios 1 and 2 realistically contain significant model noise, as alluded to above, and therefore algorithm \( C \) does badly in neglecting it. Our calculations indeed confirmed that in a case of constant velocity where the contribution of the process noise covariance is negligible (or absent), algorithm \( C \) yields superior results. Finally, algorithm \( A \) is superior to \( B \). This is expected since algorithm \( B \) neglects the model noise covariance when interpolating the central state.
In Figures 1 and 2 we present a short section of the tracks obtained in scenario 1 and 2 respectively. The red stars are the true locations (ground truth) of the target. In scenario 1, the target moves in a straight line with varying velocity. Since some of the update states from one sensor are obtained out-of-sequence, the central track deviates from the actual track as long as updates are received from only one sensor. When the OOST data are received from the other sensor, the central track is updated and reverts to the true path. This behavior is more pronounced in Fig. 2 since the target is moving in a circle with varying velocity. In this case, if we miss the update state from one sensor the central track might exit from the circle, as we see in Fig. 2. By using the OOST data the central track stays on course.

6 Conclusions

In this paper we presented an extension to the OOST problem of the three algorithms originally developed by Bar-Shalom for the OOSM problem (see ref. [4]). The significant difference between the two approaches is that in the OOSM approach there are no correlations between the fused entities (measurements) whereas in the OOST approach correlations do exist between the fused entities (tracks). The algorithms are important for real cases, where the contribution of the process noise to the tracker is not negligible.

We presented two examples of the application of the three algorithms. The examples are based on fusion of two GMTI sensors, positioned 80 degrees relative to each other. We enforced a constant delay on one sensor and thus obtained several OOST cases. The target moves with varying velocities in both examples. Algorithm C, corresponding to a complete neglect of process noise, did not perform well in these examples. Algorithm A performed best, and algorithm B performed almost as well. We expect, however, that in cases where the OOST problem is more severe than the two examples presented here, algorithm A will yield significantly superior performance to algorithm B.

Acknowledgments:

The authors would like to thank Mr. Eliezer Haber for his help in preparing this paper.

References