Adaptive Track Fusion in a Multisensor Environment

Céline Beugnon *
Graduate Student
Mechanical & Aerospace Engineering
SUNY at Buffalo
Buffalo, NY 14260, U.S.A.
beugnon@eng.buffalo.edu

Tarunraj Singh
Associate Professor
Mechanical & Aerospace Engineering
SUNY at Buffalo
Buffalo, NY 14260, U.S.A.
tsingh@eng.buffalo.edu

James Llinas
Center for Multisource Information Fusion
SUNY at Buffalo
Buffalo, NY 14260, U.S.A.
llinas@acsu.buffalo.edu

Rajat K. Saha
Nova Research Corporation
Burlington, MA, USA
Rajatksaha@aol.com

Abstract - ***The aim of this paper is to derive an adaptive approach for track fusion in a multisensor environment. The measurements of two sensors tracking the same target are processed by linear Kalman Filters. The outputs of the local trackers are sent to the central node. In this node, a decision logic, which is based on the comparison between distance metrics and thresholds, selects the method to obtain the global estimate. Numerical simulations assess the influence of the thresholds and of the sensor noise ratio on the adaptive algorithm performance. The values of the thresholds govern the trade-off between accuracy and computational burden. The main advantage of the adaptive fusion is its ability to react to changes in the system characteristics.***

Keywords: Track-to-Track Fusion, Multisensor, Adaptive Fusion

Introduction

In this paper an approach to adaptive fusion is developed in the case of two trackers (local and external) which are unable to send their raw measurements to the central node. Consider the system whose architecture is illustrated by Fig. (1).

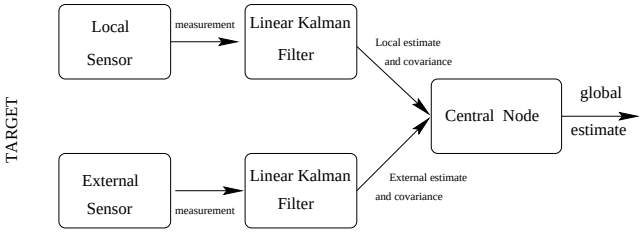

Figure 1: General Architecture of the Tracking System

The two sensors are coupled with linear Kalman Filters tracking the same target. The aim of the central node is to compute the fused estimate given the outputs of the local and external trackers. The problems of alignment and track association are not considered in this work. It is therefore assumed that the local nodes generate tracks that relate to the same target and are derived in the same system of coordinates. It is also assumed that the system is observable.

The central node has to obtain the "best" global estimate given system constraints such as global accuracy, computation load, and bandwidth capacity. Most of the techniques available to obtain the global estimates rely on data fusion and none of them is perfect. A high accuracy in the global estimate leads to large computation load and simplified fusion methods yield poor track quality. In order to accommodate the possibly varying accuracy requirement, system parameters, and resource availability, an adaptive approach to track fusion is desirable.

In Section (1) some methods to obtain the global estimates are reviewed. In Section (2) an adaptive fusion algorithm is derived, which chooses the method for calculating the global estimate according to a logic-based decision tree. In Section (3) the distance metrics, on which the decisions are based, are described. In Section (4) the performances of the four resulting adaptive algorithms are compared with other methods in terms of accuracy and computation load. Finally, conclusions are drawn that outline the improvements to be done on the proposed approach.

1 Fusion methods in a multisensor environment

Given the outputs of the local and external trackers various approaches are available to obtain the global estimates. In this section are presented the fusion methods considered for adaptive fusion.

*Exchange student from the ENSICA, Toulouse, France

1.1 Measurement Fusion

Given the architecture of Fig. (1), the optimal method is Measurement Fusion [11] where the local and external sensors send their measurements to the central node which processes them through an optimal linear Kalman Filter. The global estimates therefore yield the smallest of the estimation errors among the class of linear estimators. The main drawback of this method is the need for the measurements and the lack of robustness of the centralized approach. Moreover, the nature of the measurement can be widely varying (infrared, radar, etc.) and treating all of them at the same time can be very challenging. If the central node is unable to deal with the measurements, the global estimate has to be based on the tracker outputs through the Track-to-Track Fusion techniques.

1.2 Simple Fusion: SF

Among the Track-to-Track Fusion methods, the first one derived by Singer [10] is labeled Simple Fusion. The local nodes process their respective measurements through optimal linear Kalman Filters. The local and external trackers send their estimates, x_l and x_e, and their estimation error covariance matrices, P_l and P_e. The global estimate at time t for Simple Fusion is a simple convex combination as shown below,

$$\begin{aligned}\hat{x}_{SF}(t/t) =& P_e(t/t)[P_l(t/t)+P_e(t/t)]^{-1}\hat{x}_l(t/t) \\ &+ P_l(t/t)[P_l(t/t)+P_e(t/t)]^{-1}\hat{x}_e(t/t),\end{aligned} \quad (1)$$

with the covariance matrix of the fused estimate, P_{SF}, given by,

$$P_{SF}(t/t) = P_l(t/t)[P_l(t/t)+P_e(t/t)]^{-1}P_e(t/t). \quad (2)$$

The algorithm is based on the hypothesis that the local and external tracks are uncorrelated and can therefore be used as measurements by a global Kalman Filter. Even though this method leads to good tracking accuracy, it is suboptimal because the tracks are actually correlated through their common process noise.

1.3 Weighted Covariance Fusion: WCF

In order to account for the correlation between the local tracks, Bar-Shalom [1, 2] derived the Weighted Covariance Fusion method. The Cross Covariance matrix is a measure of the correlation due to common process noise. It is defined by $P_C = E\{\tilde{x}_l\tilde{x}_e^T\}$ with $\tilde{x}_i$ standing for the estimation error in tracker i. Consider the linear system,

$$\begin{aligned} x(t+1) &= \Phi x(t) + Gw(t), \\ z_i(t) &= H_i x(t) + v_i(t), \end{aligned} \quad (3)$$

with $Q = E\{ww^T\}$ and $R_i = E\{v_i v_i^T\}$. It satisfies the following difference equation,

$$\begin{aligned}P_C(t/t) = (I-K_lH_l)[\,&\Phi P_C(t-1/t-1)\Phi^T \\ &+ GQG^T](I-K_eH_e)^T,\end{aligned} \quad (4)$$

where K_i stands for the local Kalman gains. The fused estimate for the Weighted Covariance Fusion is given by,

$$\begin{aligned} \hat{x}_{WCF}(t/t) &= \hat{x}_l(t/t) + W[\hat{x}_e(t/t) - \hat{x}_l(t/t)], & (5)\\ \text{with } W &= [P_l(t/t) - P_C(t/t)]P_E^{-1}(t/t), & (6)\\ \text{and } P_E(t/t) &= P_l(t/t) + P_e(t/t) - P_C(t/t) & (7)\\ &\quad - P_C^T(t/t). \end{aligned}$$

Finally, its covariance is given by,

$$\begin{aligned}P_{WCF}(t/t) = P_l(t/t) - [P_l(t/t) - P_C(t/t)]P_E^{-1}(t/t) \\ [P_l(t/t) - P_C(t/t)]^T.\end{aligned} \quad (8)$$

The inclusion of the Cross Covariance matrix is beneficial if and only if this matrix is positive definite [8]. In the non positive definite case, the WCF performance degrades and the algorithm can even perform worse than Simple Fusion. WCF is only optimal in the Maximum Likelihood sense and therefore suboptimal in the Minimum Mean Square Error sense because of the lack of prior information [3, 4]. The degradation in the global track quality is small (only a few percents according to [7]) and even nullifies when the sensor measurement noise standard deviations become increasingly dissimilar [6].

Finally WCF reduces to SF when the Cross Covariance matrix is omitted.

2 Adaptive track fusion

The motivation behind adaptive track fusion comes from two observations. First, doing fusion is not always worth the required computation and bandwidth capacity. Second, the inclusion of the Cross Covariance matrix is sometimes useless and Simple Fusion may perform as well as the complete Weighted Covariance Fusion. To overcome the changes in the system characteristics and in the requirements, an adaptive approach to fusion is developed here. Its architecture is presented in Fig. (2). The sensors and the local trackers

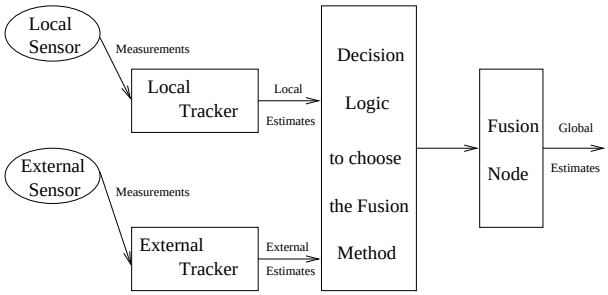

Figure 2: Architecture of the Adaptive Track Fusion Method

form the local nodes. The central node is divided into the decision logic, which chooses the method for evaluating the global estimates, and the fusion node, which evaluates the global estimates. The decision logic is based on the computation of two distance metrics and

a simple decision tree. A first metric allows the system to choose between using the local track as the global track and performing track fusion. In the latter case, another distance is computed to decide whether to use Simple Fusion or Weighted Covariance Fusion. The architecture of the decision tree is presented in Fig. (3).

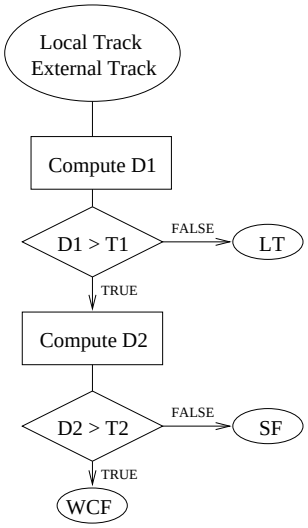

Figure 3: Decision Tree for Adaptive Fusion

The local and external tracker outputs are used to compute the first distance (D_1). If this distance is smaller than a given threshold (T_1), the global estimate is equal to the Local Track (LT). Otherwise, a second distance (D_2) is computed and compared to the second threshold T_2. Depending on this last criterion, either Simple Fusion (SF) or Weighted Covariance Fusion (WCF) is chosen to calculate the global estimate. The distances are functions of the tracking system and target characteristics whereas the thresholds are human inputs reflecting the trade-off between desired accuracy and computational load.

2.1 First distance

The first distance, between the local track and the fused estimate, is defined by,

$$\begin{aligned} D_1 &= \| \hat{x}_l - \hat{x}_{SF} \|_{(P_l+P_{SF})^{-1}}, \\ D_1 &= (\hat{x}_l - \hat{x}_{SF})^T (P_l + P_{SF})^{-1} (\hat{x}_l - \hat{x}_{SF}). \end{aligned} \tag{9}$$

If the distance is smaller than the threshold T_1 related to the desired global accuracy, the output of the adaptive filter is the local track. If the distance is greater than the threshold, fusion is performed. In this case the external track has valuable information to improve the quality of the global estimate.

The computation of the first distance is done using Simple Fusion which is the simplest fusion approach. The difference between the estimate based on Simple Fusion and the one based on Weighted Covariance Fusion is generally negligible with respect to the difference between the fused estimate and the local track.

Considering Eqs. (9), one can notice that the fused estimate and its covariance matrix appear in the calculation of the distance. This distance is computed to decide whether to use fusion or not to obtain the global estimates. The fused outputs should not appear in the distance calculations in order to allow saving in computation. Algebraic manipulations allow to express the distances only in terms of the local and external outputs. To get rid of the fused outputs, Eqs. (1) and (2) are used to obtain,

$$\begin{aligned} P_l + P_{SF} &= P_l + P_l(P_l + P_e)^{-1} P_e, \\ &= P_l[I + (P_l + P_e)^{-1} P_e], \\ &= P_l(P_l + P_e)^{-1}(P_l + 2P_e). \end{aligned} \tag{10}$$

and,

$$\begin{aligned} \hat{x}_l - \hat{x}_{SF} &= \hat{x}_l - P_e(P_l + P_e)^{-1}\hat{x}_l - P_l(P_l + P_e)^{-1}\hat{x}_e, \\ &= [P_l + P_e - P_e](P_l + P_e)^{-1}\hat{x}_l \\ &\quad - P_l(P_l + P_e)^{-1}\hat{x}_e, \\ &= P_l(P_l + P_e)^{-1}(\hat{x}_l - \hat{x}_e). \end{aligned} \tag{11}$$

Combining the two previous results, the first distance is finally expressed by,

$$D_1 = (\hat{x}_l - \hat{x}_e)^T (P_l + P_e)^{-1} P_l (P_l + 2P_e)^{-1} (\hat{x}_l - \hat{x}_e), \tag{12}$$

using the symmetry of the covariance matrices, P_i.

2.2 Second distance

If the first distance is greater than the given threshold T_1, track fusion is chosen to be performed. A second distance is then computed to choose between Simple Fusion and Weighted Covariance Fusion that is to know if the Cross Covariance matrix has to be included in the calculation of the fused estimates. Define the second distance as,

$$\begin{aligned} D_2 &= \| \hat{x}_l - \hat{x}_{WCF} \|_{(P_l+P_{WCF})^{-1}}, \\ D_2 &= (\hat{x}_l - \hat{x}_{WCF})^T (P_l + P_{WCF})^{-1} (\hat{x}_l - \hat{x}_{WCF}). \end{aligned} \tag{13}$$

Using Eqs. (5)-(8), we obtain:

$$\begin{aligned} \hat{x}_l - \hat{x}_{WCF} &= (P_l - P_C) P_E^{-1} (\hat{x}_l - \hat{x}_e), \\ P_l + P_{WCF} &= (P_l - P_C) P_E^{-1} P_A, \end{aligned} \tag{14}$$

where the matrix P_E is given by Eq. (8) and the matrix P_A is defined by,

$$P_A = P_l + P_C^T + 2(P_e - P_C^T)(P_l - P_C)^{-1} P_l. \tag{15}$$

The second distance is expressed as a function of the local agent outputs by,

$$D_2 = (\hat{x}_l - \hat{x}_e)^T P_E^{-1} (P_l - P_C)^T P_A^{-1} (\hat{x}_l - \hat{x}_e). \tag{16}$$

If the second distance is smaller than the threshold T_2, fusion is performed according to the Simple Fusion algorithm described by Eqs. (1) and (2). If the distance is larger than T_2, the Weighted Covariance Fusion approach is used according to Eqs. (5)-(8).

3 Simplification of the distance calculations

The advantage of the adaptive fusion algorithm is to perform fusion only when it is really worth it and therefore to save on computation overhead. However, what can be saved by avoiding involved calculations can be spent in the distance calculations. Different methods are presented to reduce the cost of computing the distance metrics.

3.1 Use of the steady state Cross Covariance matrix

First, to compute the distance D_2, the value of the Cross Covariance matrix has to be known. The exact value satisfies the time difference Equation (4) initialized with the zero matrix since the first local and external tracks are not correlated yet through common process noise. The propagation in time of this matrix requires that the local and external trackers send their Kalman gains, K_l and K_e, to the fusion node. When these gains are not available or when one wants to avoid the computationally expensive propagation in time, the steady state Cross Covariance can be employed. It satisfies the following asymmetric Lyapunov equation,

$$P_C = F_l P_C F_e^T + Q^*, \tag{17}$$

where

$$\begin{aligned} F_l &= (I - K_l H_l)\Phi, \\ F_e &= (I - K_e H_e)\Phi, \\ Q^* &= (I - K_l H_l) G Q G^T (I - K_e H_e). \end{aligned}$$

By applying a bilinear transformation [9] to the previous equation, the steady state Cross Covariance matrix satisfies,

$$AP_C + P_C B^T = \bar{Q}, \tag{18}$$

where

$$\begin{aligned} A &= (F_l - I)^{-1}(F_l + I), \\ B &= (F_e - I)^{-1}(F_e + I), \\ \bar{Q} &= -2(F_l - I)^{-1} Q^* (F_e^T - I)^{-1}. \end{aligned}$$

The steady state Cross Covariance matrix is calculated once for the given target and sensor characteristics. It is a good approximation of the true Cross Covariance as long as the system characteristics do not significantly change during the tracking interval.

3.2 Use of partial tracks and covariance matrices

Second, to avoid large amount of computation, the calculation of the distances can be based only on position. More precisely, instead of using the complete tracks and covariance matrices, only the (1,1) elements of the local and external tracks and covariance matrices are used to calculate the distances. The first distance based on partial local outputs becomes,

$$\begin{aligned} D_1^{'} = [\hat{x}_l(1,1) - \hat{x}_e(1,1)]^T [P_l(1,1) + P_e(1,1)]^{-1} P_l(1,1) \\ [P_l(1,1) + 2P_e(1,1)]^{-1} [\hat{x}_l(1,1) - \hat{x}_e(1,1)]. \end{aligned} \tag{19}$$

And the second distance becomes,

$$\begin{aligned} D_2^{'} = &[\hat{x}_l(1,1) - \hat{x}_e(1,1)]^T P_E^{-1}(1,1) [P_l(1,1) - P_C(1,1)]^T \\ &P_A^{-1}(1,1) [\hat{x}_l(1,1) - \hat{x}_e(1,1)]. \end{aligned} \tag{20}$$

Use of the partial local outputs is done to decrease the computation load of the distance calculations. The distance metrics obtained $D_1^{'}$ and $D_2^{'}$ are only a partial representation of the real distance between the local and the fused tracks. Using these distances corresponds to underestimating the real distances, since the effect of velocity is discarded.

3.3 Resulting adaptive algorithms

Combining the two previously described simplifications offers further saving in computation and leads to four adaptive methods based on the same decision tree of Fig. (3) but on different metrics.

1. With Complete Metrics: Adapt1 The distances are calculated using the local agent full tracks and covariance matrices. The Cross Covariance matrix is propagated at each instant in time and used when necessary.

2. With Complete Metrics and Steady State Covariance Matrix: Adapt2 The distances are similarly to Adapt1 computed using all the information coming from the local trackers. The only difference lies in the fact that the Cross Covariance matrix is assumed to be constant. It is equal to its steady state value which is found by numerically solving for the asymmetric Lyapunov equation as detailed in Section (3.1).

3. With Partial Metrics: Adapt3 The distances are calculated by only using the (1,1) elements of the local agent tracks, covariance and cross covariance matrices. The lack of accuracy in the distance calculations is compensated by the small amount of computation required since using scalar components avoids expensive matrix inversions.

4. With Partial Metrics and Steady State Covariance Matrix: Adapt4 The (1,1) element of the steady state Cross Covariance matrix is used in the calculation of the second partial distance, $D_2^{'}$. It leads to a reduction of the computation load in the central node since the Cross Covariance matrix no longer has to be propagated in time.

4 Performance Assessment

4.1 Example system

The considered system is made of two sensors and a central node as in Fig. (2). The target is a second order system driven by process noise $w(t)$ whose covariance matrix, Q equals 10^4. The sampling time, T is equal to 0.2 second. Hence, we have for System (3), $\Phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}$, $G = \begin{bmatrix} T^2/2 \\ T \end{bmatrix}$, and $H_i = \begin{bmatrix} 1\ 0 \end{bmatrix}$.

4.2 Means of Comparison

The performances of the adaptive algorithms are compared with the performances of the Local Track, the Simple Fusion and finally the Weighted Covariance Fusion methods. Two criterion are considered to assess the performance of a method: the loss in accuracy and the computation load.

The loss in accuracy is defined as the difference between the accumulative root mean square fusion error of the considered method and the accumulative root mean square fusion error of the optimal Measurement Fusion method. The errors are summed over the time interval of the simulation ($t_f = 100$s) and averaged over 200 Monte Carlo simulations to obtain smooth results. The loss in accuracy for the method M is therefore defined by,

$$L_M = \underset{over\ MC\ Simulations}{Mean} \{ \sum_{t=t_0}^{t=t_f} rms_M(t) - rms_{MF}(t) \}, \tag{21}$$

where rms stands for the root mean square fusion errors.

The evaluation of the computation load is less straightforward. Only the computation that takes place into the central node at each iteration is considered. The amount of calculation done in the local and external trackers is not involved. Calculations that are done once, like the evaluation of the steady state Cross Covariance, are not added either. The computation load is evaluated through the mean number of flops per iteration. This number is calculated by averaging the number of flops given by Matlab over the time of simulation and the number of Monte Carlo simulations. The numbers of flops are not accurate and only their relative values are important. Indeed, code written is Matlab is not optimal in terms of computation efficiency. The calculations of the flops is furthermore specific to the language as it is specified in the Matlab online help:

> FLOPS returns the cumulative number of floating point operations. It is not feasible to count absolutely all floating point operations, but most of the important ones are counted. Additions and subtractions are one flop if real and two if complex. Multiplications and divisions count one flop each if the result is real and six flops if it is not. Elementary functions count one if real and more if complex.

4.3 Influence of the threshold for the first distance

The value of the first distance is compared to the threshold T_1 to decide whether fusion is performed or the local track is preferred. In this section, the influence of this threshold on the performances of the adaptive algorithms is assessed for 200 Monte Carlo simulations while the value for the second threshold is set to 0.5. The sensors have the dissimilar measurement noise standard deviations, $\sigma_l = 10$, $\sigma_e = 5$. In Fig. (4) are simultaneously plotted the loss in accuracy (*top*) and the mean number of flops per iteration (*bottom*) as a function of the first threshold value.

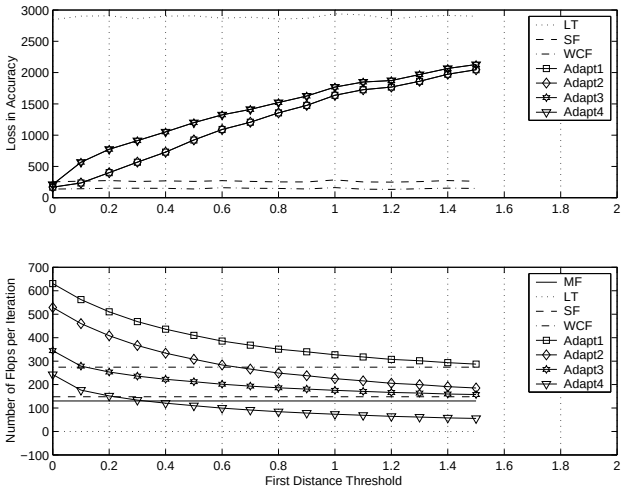

Figure 4: Influence of the First Distance Threshold: $\sigma_l = 10$, $\sigma_e = 5$. *top*: On the Mean Loss in Accuracy - *bottom*: On the Mean Number of Flops per Iteration.

The numbers of flops and the loss in accuracy for the conventional methods are constant, they do not depend on the value of the threshold. The little variations observed in the Fig. (4) are due to an insufficient number of Monte Carlo simulations. The loss in accuracy is the largest for the Local Track (LT). In this case, the external tracker outputs are not considered and their information lost. When applying this method, no computation at all is required in the central node but the performance is poor since the information contained in the external track is discarded. The loss in accuracy for Simple Fusion (SF) is larger than the loss for Weighted Covariance Fusion (WCF). Considering the computation load, SF is less resource consuming and the number of flops for this method is smaller than the number of flops for WCF.

Considering the threshold and its influence on the performances of the adaptive approaches, we observe that increasing T_1 leads to an increase in the loss in accuracy and simultaneously a decrease in the computation load. Indeed, the smaller the threshold is, the more fusion is carried out. Performing fusion induces an increase in the computation load but also improves the quality of the global track because the external track has valuable information.

Considering the adaptive algorithms, the use of the steady state Cross Covariance does not affect the loss

in accuracy but only the computation requirement. The difference in the numbers of flops is due to the necessity of propagating the Cross Covariance matrix in time in the central node if its steady state value is not used.

Using the partial distances leads to a large decrease in the computation load but also induces a larger loss of accuracy. The partial distances underestimate the real distance between the local track and the fused track. The Local Track is chosen more often than it should. As a result, the computation burden decreases faster and the greater use of the inaccurate Local Track leads to larger estimation errors. In order to obtain the same behavior with the complete or the partial metrics, given the same set of data, the value of the first threshold used for comparison with the partial distances should be reduced to account for the underestimation of the true distance.

The available computation capacity influences the choice of method for the distance estimations. The choice is also depending on the system architecture. For instance, the propagation in time of the Cross Covariance matrix, Eq. (4), requires the local and external Kalman gains, which may not be available. In such cases, only Adapt2 and Adapt4 are conceivable.

Depending on the global accuracy desired, the threshold is chosen. If the global estimate has to be accurate, the threshold is set to a small value. On the contrary, if the accuracy is not vital, a larger value of the threshold will lead to savings in the computation load.

Fig. (4) clearly shows the trade-off between the global accuracy requirement and the computation capacity through the influence of the first threshold value on the performances of adaptive approaches. The most interesting feature of the adaptive algorithms is their ability to react to changes in desired global accuracy and in the amount of available computation capacity, only the value of the threshold has to be modified. The benefit of their flexibility is to avoid fusion when it is useless in terms of accuracy or even penalizing in terms of computation load.

4.4 Influence of the threshold for the second distance

The choice of a threshold for the first distance is guided by the trade-off between global accuracy and computation load. The second threshold T_2 is useful to choose between Simple Fusion (SF) and Weighted Covariance Fusion (WCF). If the second distance is greater than this threshold, WCF is used to calculate the output of the adaptive filter, else Simple Fusion is performed. To study the influence of the second threshold on the filter performances, the first threshold is set to zero, therefore fusion is always considered. The other parameters of the system are identical to those of the previous Section (4.1). The influence of the second threshold is illustrated by Fig. (5) where the *top* subfigure represents the loss in accuracy with respect to the optimal Measurement Fusion method and the *bottom* subfigure shows the average number of flops per iteration.

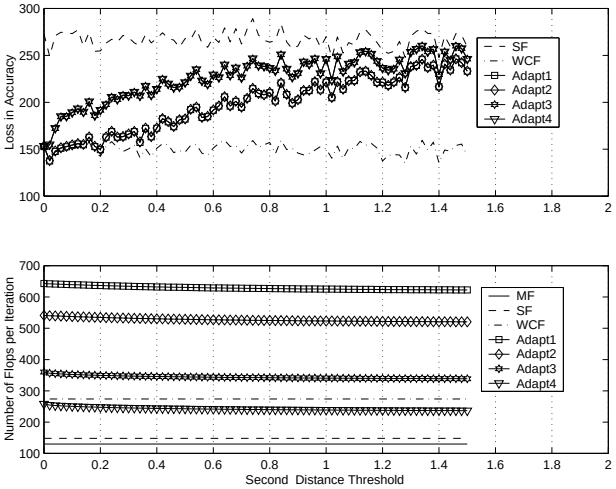

Figure 5: Influence of the Second Distance Threshold. *top*: On the Mean Loss in Accuracy - *bottom*: On the Mean Number of Flops per Iteration.

The largest loss in accuracy takes place for Simple Fusion and the smallest for Weighted Covariance Fusion. The loss for the adaptive approaches varies between these two values as a function of the threshold value, T_2. For small values of T_2, it is almost equal to the loss for WCF. When the threshold increases, the loss increases and tends to the loss for SF.

The computation loads for the adaptive methods are larger than the load for the classical fusion methods, with the exception of Adapt4. The global estimate is calculated using either SF or WCF because setting T_1 to zero suppresses the Local Track option. The computation load for the adaptive methods includes the calculations of the first and the second distances, plus the calculation of the fused estimate. The number of flops for the adaptive methods is therefore larger than the number for SF. The difference is a function of the method used to calculate the distances. If the distance evaluation is not resource consuming as for Adapt4, the computation load can be less than the load for WCF. The number of flops per iteration for the adaptive methods slightly decreases when the threshold value increases. In this case, the global estimate is calculated using more and more the SF approach, which is less computationally expensive than WCF.

Both the loss in accuracy and the computation load for the adaptive methods are ranging between the SF value and the WCF value. The value of T_2 governs the choice of a fusion method. For small values of the threshold, WCF is preferred and leads to a good accuracy of the global estimate but a high computation load. WCF usually performs better than SF because of the inclusion of the Cross Covariance matrix to account for common process noise. For large values of T_2, the accuracy requirement on the global estimate diminishes and Simple Fusion is preferred. Therefore, saving on the computation load is obtained but the quality of the global estimate degrades. The choice of a threshold for the second distance is therefore a

trade-off between the global accuracy requirement and the computing capacity.

The distances based on the partial outputs (Adapt3 and Adapt4) underestimate the real distance since the information based on velocity is not considered. For the same threshold value, these adaptive algorithms switch more often than they should to the SF, leading to a larger loss in accuracy than Adapt1 and Adapt2. Therefore the threshold value to prefer Simple Fusion over Weighted Covariance Fusion depends on the type of distance calculation and should be chosen accordingly.

In this example, the use of the steady state Cross Covariance does not affect the accuracy of the global estimates but allows a decrease in the computation load by avoiding the propagation in time of the Cross Covariance matrix in the central node.

Notice has to be taken that the difference in accuracy between the Simple Fusion and the Weighted Covariance Fusion methods is small in comparison with the difference between the fusion methods and the local track. If the loss in accuracy was plotted in Fig. (5), it would be around 3000 which is more than 10 times the loss for the less accurate fusion method. The choice of the fusion method is therefore a second level decision. The importance of the second threshold value increases when the value of the first threshold tends to zero. The influence of the fusion method is however small in terms of performance compared to the influence of the first decision to use the Local Track or to perform fusion.

The threshold T_2 influences the performance of the adaptive methods. It represents the trade-off between accuracy and computation load. Its value determines the fusion method according to the required global accuracy and the computing capacity. Its influence is however small and is only important for small values of the first threshold.

4.5 Influence of the sensor noise ratio

Up to now, the standard deviations of the local and external tracker have been fixed and the influences of changing the thresholds have been assessed. Now, the thresholds are fixed and equal 0.3 for the first distance and 0.5 for the second. Define the sensor noise ratio as the ratio of the standard deviation of the measurement noise for the external sensor to the standard deviation of the measurement noise for the local sensor. This ratio is varying from 10^{-5} to 10^{5} with the standard deviation of the local sensor noise equals to 10. The performance of the adaptive fusion algorithm is assessed by 200 Monte Carlo simulations. In Fig. (6) is plotted the loss in accuracy with respect to the optimal Measurement Fusion (*top*) and the average number of flops per iteration (*bottom*).

For large values of the sensor noise ratio, the qualities of the adaptive methods are better than the quality of Simple Fusion. The losses in accuracy are small and almost equal to the WCF loss. The computation load for WCF is moreover larger than the load for the adaptive methods. When the sensor noise ratio is large, the

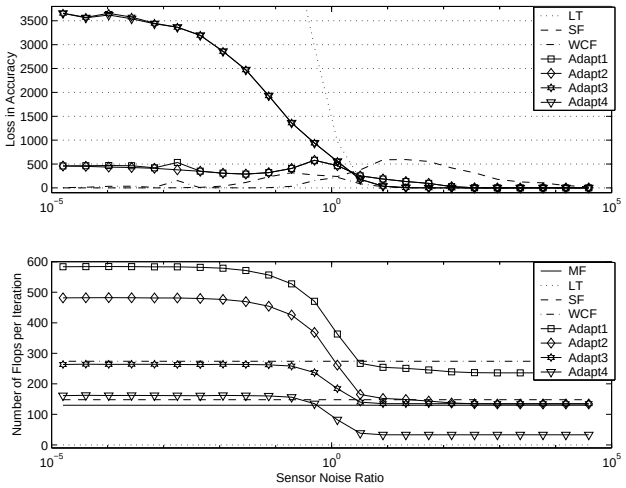

Figure 6: Influence of the Sensor Noise Ratio. *top*: On the Mean Loss in Accuracy - *bottom*: On the Mean Number of Flops per Iteration.

accuracy of the local tracker is much better than the accuracy of the external tracker. In that case, the improvement of the fused track accuracy over the local track accuracy is small and not worth the computation load. The adaptive algorithms prevent useless computations by choosing the local track.

For small values of the ratio, the quality of the local track is really poor in comparison with the quality of the external track. Since the threshold for the first distance is set to quite a large value (0.3), the local tracker output is not necessarily always discarded. Using the poorly accurate local track leads to a loss in accuracy larger than the losses for SF and WCF. The reduction in the computation load is only clear for Adapt3 and Adapt4, which matches the result of Fig. (4) for $T_1 = 0.3$.

For the same small value of the sensor noise ratio, the losses in accuracy for the partial distance methods (Adapt3 and Adapt4) are larger than the losses for the complete distance approaches (Adapt1 and Adapt2) and vice versa for the computation load. Using the partial distances leads to underestimating the real distance. The adaptive methods based on partial distances therefore switch to the Local Track option more often than the other adaptive methods. As stated in the previous example, the choice of a value for the thresholds depends on what is the trade-off between accuracy and computing capacity and also on which method is preferred to evaluate the distances. In this example, the threshold values for comparison with the partial distances should be decreased to obtain the same behavior as with the complete metric.

For a ratio close to unity, the loss in accuracy for the Simple Fusion equals the loss for the more sophisticated WCF and its data processing requires less computation power. Considering the adaptive approaches, the ones based on complete metrics perform well. Their losses in accuracy are almost equal to the classical method losses. Their computation requirement are however more important. The adaptive

approaches based on partial metrics allow saving in computation but their accuracies degrade. If the ratio is close to one but greater, i.e. the local track is slightly more accurate, the partial distances are beneficial. Their computation loads are smaller than the loads for the classical fusion methods and their degradation in accuracy has the order of magnitude of the degradation for SF.

Varying the sensor noise ratio outlines the ability of the adaptive algorithms to react to changes in the system characteristics. When the local track carries a lot more information than the external track, fusion is not considered. The global estimate is almost as accurate as the WCF estimate but the computation load is much smaller. On the contrary case when the external sensor is much more accurate, the performances of the adaptive algorithms degrade. To improve the adaptation for small values of the sensor noise ratio, a good alternative would be to allow the decision node to choose the external track. The decision could be made on the respective level of local and external sensor measurement noise. Improvement would be obtained by constructing statistical tests to choose the "best" track when the sensors have largely dissimilar accuracy. This way more useless computation would be avoided while maintaining a good tracking accuracy.

5 Conclusions

In this paper, a simple approach for adaptive fusion has been derived. It is based on the computation of two distances and their comparison to thresholds. Four adaptive fusion methods have been introduced. The differences lie in the calculation of the distances (with the propagation in time of the Cross Covariance or with its steady state value) and also in the use of the complete or the partial local outputs.

The performance of the adaptive algorithms is expressed in terms of computation load and loss in accuracy with respect to the optimal Measurement Fusion. Their performances are studied as a function of the threshold values and the sensor noise ratio. The main advantage of adaptive fusion is its ability to react to the changes in the sensor environment and to avoid useless or even penalizing fusion. Only the values of the thresholds have to be changed to reflect the changes in accuracy requirement or computing capacity.

To empower adaptive fusion, some improvements should be done on the distances and on the decision tree. Refinement of the distances includes normalization and simplification of the calculations. Better performance requires the extension of the decision tree through the inclusion of more fusion options like Best Track or Information Fusion [5] and through the possibility to choose among the various distance calculations. For instance, a higher level of decision should decide either to use or avoid the steady state Cross Covariance according to the availability of the local gains or the quality of the approximation. A last step would be the generalization to more practical cases: multiple asynchronous sensors, communication and/or processing delays, etc.

References

[1] Y. Bar-Shalom. **On the Track-to-Track Correlation Problem**. *IEEE Transactions on Automatic Control*, TAC 26(2):571–572, Apr 1981.

[2] Y. Bar-Shalom and L. Campo. **The Effect of Common Process Noise on the Two-Sensor Fused-Track Covariance** . *IEEE Transactions on Aerospace and Electronic Systems*, AES 22(6):803–805, Nov 1986.

[3] K.C. Chang, R.K. Saha, and Bar-Shalom Y. **On Optimal Track-to-Track Fusion**. *IEEE Transactions on Aerospace and Electronic Systems*, AES 33(4):1271–1275, Oct 1997.

[4] K.C. Chang, Zhi Tian, and R.K. Saha. **Performance Evaluation of Track Fusion With Information Filter** . In *Proceedings of the International Conference on Multisource-Multisensor Information Fusion*, pages 648–655, July 1998.

[5] Yaakov Bar-Shalom Editor. **Multitarget-Multisensor Tracking: Advanced Applications**. Artech House, 1990.

[6] A.M. Haimochiv, J. Yosko, R.J. Greenberg, M.A. Parisi, and D. Becker. **Fusion of Sensors with Dissimilar Measurement/Tracking Accuracies**. *IEEE Transactions on Aerospace and Electronic Systems*, AES 29(1):245–249, Jan 93.

[7] J.A. Roecker and C.D. McGillem. **Comparison of Two-Sensor Tracking Methods Based on State Vector Fusion and Measurement Fusion**. *IEEE Transactions on Aerospace and Electronic Systems*, AES 24(4):447–449, July 1988.

[8] R.K. Saha. **Effect of Common Process Noise on Two-Sensor Track Fusion**. *Journal of Guidance, Control, and Dynamics*, 19(4):829–835, July-August 1996.

[9] R.K. Saha. **Track-to-Track Fusion With Dissimilar Sensors**. *IEEE Transactions on Aerospace and Electronic Systems*, AES 32(3):1021–1028, July 1996.

[10] R.A. Singer and A.J. Kanyuck. **Computer Control of Multiple Site Track Correlation** . *Automatica*, 7:455– 462, 1971.

[11] D. Willner, C.B. Chang, and K.P. Dunn. **Kalman Filter Algorithms for a Multi-Sensor Systems**. In *Proceedings of the IEEE Conference on Decision and Control*, pages 570–574, 1978.