Multisensor Particle Filter Cloud Fusion for Multitarget Tracking

Daniel Danu, Thia Kirubarajan
Department of Electrical and Computer Engineering
McMaster University, Hamilton, Canada
danudg@univmail.cis.mcmaster.ca
kiruba@mcmaster.ca

Thomas Lang
General Dynamics, Ottawa, Canada
Tom.Lang@gdcanada.com

Michael McDonald
DRDC Ottawa, Canada
Mike.Mcdonald@drdc-rddc.gc.ca

Abstract – Within the area of target tracking particle filters are the subject of consistent research and continuous improvement. The purpose of this paper is to present a novel method of fusing the information from multiple particle filters tracking in a multisensor multitarget scenario. Data considered for fusion is under the form of labeled particle clouds, obtained in the simulation from two probability hypothesis density particle filters. Different ways of data association and fusion are presented, depending on the type of particles used (e.g. before resampling, resampled, of equal or of different cardinalities). A simulation is presented at the end, which shows the improvement possible by using more than one particle filter on a given scenario.

Keywords: Tracking, particle filtering, data association, finite random sets.

1 Introduction

Within the last decade and a half, particle filters have been the subject of consistent research and improvements, with special emphasis on target tracking [2]-[10]. The essential work in [2] takes on a single target and, using the bootstrap particle filter, expands tracking performance beyond the classical Extended Kalman Filter (EKF) for nonlinear state dynamics and measurements. In [3] several versions of sampling and resampling developed in the mean-time are presented, while in [4] the authors present an extension of the particle filter to the multitarget case. The fundamental work in [5] on multitarget tracking based on random finite sets (RFSs) and finite set statistics (FISST) derives equations for sequential estimation of the first order multitarget moment aka probability hypothesis density (PHD). Based on the FISST and PHD approaches, new particle filters devised for multitarget scenarios, with the ability to handle target births, deaths and spawning were developed by authors in [6]-[8]. The PHD particle filter was improved in [9]-[10] with track identities, or track labels, use of which is made in the current work during the fusion process. The purpose of this paper is to present a novel method of fusing the information from multiple such particle filters tracking in a multisensor multitarget scenario. Specifics of the particle filters used in local estimators are detailed in section 2. The algorithms for associating and fusing particle clouds (where a cloud is a cluster of particles labeled as pertaining to the same target) generated by two different particle filters are presented in section 3. Ground target tracking simulation and results are given in section 4, while conclusions are drawn in section 5.

2 Particle Filters

Because of its ability to handle target birth (or death), track initiation and spawning in a multitarget scenario, the PHD implementation of a particle filter (PHD-PF) was considered for generating particle clouds at local sensor level. Further in this paper sensor and local estimator have a similar meaning, both denoting a local sensor-estimator entity that sends its output data for fusion.

2.1 PHD Particle Filter

The PHD definition is based on the theory of RFSs, whose statistics a.k.a. FISST define the targets in a scenario as a RFS (meta-target) and the set of observations as another RFS (meta-observation) [5]. The PHD was introduced in [5] as the first order multitarget moment – the density function whose integral over a region is the expected number of targets in that region. Being defined in single-target state space, its value at a given point is the probability density function of target presence, therefore provides target localization. In the same work [5] the author derives, within the Bayesian framework of FISST, the sequential estimation of the first order multitarget moment, further known within the target tracking community as the PHD filter. The PHD filter has the ability to initiate tracks of newly born targets, spawning targets, as well as to terminate tracks for dead targets in a multitarget scenario of dynamic target cardinality, while also able to account for clutter within the measurement set. A sequential Monte Carlo implementation of the PHD filter, known as the PHD particle filter, was derived in [6]-[8].

2.2 Track labeling

One main step of the data fusion process in target tracking is data association, in which data (sets) from different
local estimates are established (grouped) as pertaining to the same target. The fused estimate results from the combination of data contained within such an already associated group (the two steps might overlap or even pass through several iterations in some novel approaches). Data association is a very computationally intensive task, in most cases increasing exponentially with the number of sensors and targets. When fusing particle clouds, the problem is complicated even more, by having for each target a high number of representing particles. This problem is greatly alleviated by labeling particles estimated as representing the same target at the sensor level, prior to fusion. A PHD particle filter with the particle labeling capability was developed in [10]. The main idea is to run in parallel another estimator in order to preserve track identities (e.g. Kalman filter) and associate PHD peaks to current tracks identities. As a result, the PHD particles entering the fusion process, beside the state and weight, have the track label information, which basically them in several herein called clouds. For example, after processing the measurements received at a given time \( k \), the output of a given particle filter \( s \) (where \( s = 1..S \) and \( S \) is the number of sensors) is the set of labeled particles corresponding to the estimated PHD, \( \hat{D}_{s,k,k} \):

\[
\hat{D}_{s,k,k} = \left\{ \left( w_{l,s,k}^l, \xi_{l,s,k}^l \right) \right\}, \quad l_i = 1..\hat{N}_{s,k,k}, \quad i = 1..L_{s,k,k} \quad (1)
\]

Here \( \xi_{l,s,k}^l \) is a particle of track label \( l_s \) and index \( i \), \( w_{l,s,k}^l \) is its weight, \( \hat{N}_{s,k,k} \) is the estimated number of targets, \( l_i = 1..\hat{N}_{s,k,k} \) are all targets labels (numbered here from 1 to \( \hat{N}_{s,k,k} \), though this is seldom possible), and \( L_{s,k,k} \) is the number of particles in the cloud corresponding to track labeled \( l_s \), all at time \( k \) and at sensor \( s \). In further notation the time index \( k \) is dropped for simplification, as fusion is performed statically (at a common instance, thus time being irrelevant for this paper purpose).

One advantage of using labeled particles in the particle filter sequential estimation is that new particles can be thrown into processing at any frame around every measurement (true or false) and no estimation of false measurements spatial distribution is needed for generating these new particles. The particles corresponding to false measurements are eliminated in the next frames, as they will not generate usually confirmed tracks, and particles of non-confirmed tracks are not propagated for more than two or three frames. (False tracks, when generated, are usually of small duration.) The advantage of using labeled particles for fusion is that particles of non-confirmed tracks do not enter the fusion process, thus decreasing a lot the data association problem size and communication.

3 Fusion of labeled particle clouds

The fusion part is divided into two main stages, the first being the data association and second the estimation from combined associated data. Data to be fused here consists of labelled particles and their corresponding weights. The two stages are treated separately next, in developing the fusion method for two sensors.

3.1 Association of particle clouds

Using notations in (1) with time index dropped, we denote by \( \Xi_1^h \) a \( l_1 \)-labeled cloud at sensor one, and by \( \Xi_2^h \) a \( l_2 \)-labeled cloud at sensor two, with labels \( l_1 = 1..\hat{N}_1 \) and \( l_2 = 1..\hat{N}_2 \). (We assume that on both sensors labels \( l_1, l_2 \) are contiguous and start from 1 only for the purpose of simplifying the notation). We denote by \( \text{cost} \left( \Xi_1^h, \Xi_2^h \right) \) the cost of the hypothesis that clouds \( \Xi_1^h \) and \( \Xi_2^h \) correspond to the same target. Similarly to the case of track-to-track association, if we assume that the cloud association events among different cloud pairs are independent, the most likely cloud-to-cloud association hypothesis can be found, within the 2D\(^1\) assignment formulation, by solving the following constrained optimization:

\[
\text{min} \sum_{\xi_1^{1..l_1}} \sum_{\xi_1^{1..l_2}} \chi_{\xi_1^h,\xi_2^h} \text{cost} \left( \Xi_1^h, \Xi_2^h \right) \quad (2)
\]

subject to

\[
\sum_{l_1=1}^{L_1} \chi_{\xi_1^{l_1},\xi_2^h} = 1, \quad l_1 = 1..L_1 \quad (3)
\]

\[
\sum_{l_2=1}^{L_2} \chi_{\xi_1^{l_1},\xi_2^{l_2}} = 1, \quad l_2 = 1..L_2
\]

where \( \chi_{\xi_1^{l_1},\xi_2^{l_2}} \) is a binary variable. The labels \( l_1 = 0 \) and \( l_2 = 0 \) in (2-3) are used to denote the association with dummy (therefore no association for the cloud on the other sensor). The minimization in (2)-(3) can be easily solved using for example auction algorithms [12]-[13]. It remains to determine the evaluation of the cloud-to-cloud association cost, \( \text{cost} \left( \Xi_1^h, \Xi_2^h \right) \), which is treated in the next section. In computing the cloud-to-cloud association cost, two types of clouds are considered (from the particle resampling viewpoint): clouds of unresampled particles and clouds of resampled particles. A gating between clouds has to be used first, such that some costly multiassignments (defined next) are skipped.

3.2 Cost of cloud to cloud association

A particle cloud is a finite set of particles, therefore the cost of associating two clouds can be seen as the distance

\(^1\) The optimization problem generalizes to \( N \)-D assignment when the number of sensors \( S > 2 \).
between the two sets (of \( M_1 \) and \( M_2 \) particles) defining the clouds. Prior to computing the cost of associating two clouds
\[
\Xi_s = \left\{ w_s^i, \xi_s^i \right\}_{i=1}^{M_s}, \ s = 1, 2
\]
on two sensors (labeled clouds, however, with label dropped here for notation simplification), both clouds are normalized such that their weights sum to one. This is done because we wish to re-interpret the particle cloud as a discretized representation of the target’s probability density function in 2D-Cartesian space, of which integral is one:
\[
\int_{\mathbb{R}^2} p_s(x, y) \cdot dA(x, y) = 1
\]
where \( R \) is the region over which the density is estimated, \( p_s \) is the density function and \( dA \) is the elementary surface area (for estimation in \( xy \) plane). A cloud is sent for fusion only if its track was confirmed, therefore the target is almost surely present and (5) holds over the estimation surface \( R \). For the normalized weights we have
\[
\sum_{i=1}^{M_s} w_s^i = \sum_{i=1}^{M_s} \frac{w_s^i}{\Delta A(\xi_s^i)} \Delta A(\xi_s^i) = \sum_{i=1}^{M_s} p_s(\xi_s^i) \cdot \Delta A(\xi_s^i) = 1
\]
From (6), upon normalization each particle weight corresponds to the pdf of the cell surrounding the particle (of area \( \Delta A(\xi_s^i) \) around \( \xi_s^i \)) multiplied by the area cell \( w_s^i = p_s(\xi_s^i) \cdot \Delta A(\xi_s^i), \) representing the cell probability mass. The Wasserstein distance defines the dissimilarity between two probability densities \( f, g \), using the distance \( d \) (e.g. Euclidian for the case of particles here) and the joint distribution \( h(x, y) \) whose marginals are the two densities
\[
\int h(x, y)dx = g(y) \text{ and } \int h(x, y)dy = f(x)
\]
\[
d_W^p(f, g) = \inf_{\delta} \sqrt{\int d(x, y)^p h(x, y)dx\,dy}
\]
The inclusion of state velocity component into particle distance could be added (i.e. as \( d = ||x1-x2|| + \varepsilon \ ||v1-v2|| \)) with \( \varepsilon \) having a the physical meaning small time interval. In [11] the multitarget miss distance is introduced based on the Wasserstein distance as a means to assess the multitarget tracking performance (through providing a distance between the set of estimated tracks and true targets). Here we use the adjustment of Wasserstein distance to finite sets as suggested in [11], the difference being that instead of being applied to sets of targets, it is applied to sets of weighted particles. We use for functions \( f \) and \( g \) the sum of Dirac delta functions \( \delta_{\Xi_s}(\xi_s) \) and \( \delta_{\Xi_s}(\xi_s) \), defined on the surfaces \( \Xi_s = \{(x, y)\} \), covered by both sensors \( s = 1, 2 \) and with values on the discrete sets corresponding to the two particle clouds \( \Xi_1 = \{\xi_1^1, \ldots, \xi_{M_1}^1\} \), \( \Xi_2 = \{\xi_2^1, \ldots, \xi_{M_2}^1\} \)
\[
\delta_{\Xi_s}(\xi_s) = \sum_{i=1}^{M_s} w_s^i \cdot \delta_{\Xi_s}(\xi_s), \ s = 1, 2
\]
and \( h(x, y) \) is defined as
\[
h(x, y) = \sum_{i=1}^{M_1} \sum_{j=1}^{M_2} w_{ij} \delta_{\Xi_1}(\xi_s) \delta_{\Xi_2}(\xi_s)
\]
where \( w_{ij} \) are elements of the transportation matrix \( W \) known under this name from the linear programming special case of transportation problem. Introducing (8.1) and (8.2) in (7), the Wasserstein distance applied to two particle clouds becomes:
\[
d_W^p(\Xi_1, \Xi_2) = \inf_{\delta} \sqrt{\sum_{i=1}^{M_1} \sum_{j=1}^{M_2} \sum_{s=1}^{2} w_{ij} \delta_{\Xi_s}(\xi_s) \delta_{\Xi_s}(\xi_s)}
\]
with \( p = 1 \) or \( p = 2 \). Different methods for computing the cost based on (9) are derived further, depending on the type of particles used (i.e. resampled or before resampling).

3.2.1 Using clouds of resampled particles
Using clouds of (labeled) resampled particles at the particle filter level produces a fixed number of particles per cloud, all having same weight (i.e. 1/M for \( M \) particles per cloud). The constant weight of all particles results in the special form of \( \delta_{\Xi_s}(\xi_s) \) as
\[
\delta_{\Xi_s}(\xi_s) = \frac{1}{M_s} \sum_{i=1}^{M_s} \delta_{\Xi_s}(\xi_s), \ s = 1, 2
\]
In this case the transportation matrix \( W \) has the sums on all rows equal, and the same holds for columns:
\[
\sum_{i=1}^{M_1} w_{ij} = \frac{1}{M_2}, \ \sum_{j=1}^{M_2} w_{ij} = \frac{1}{M_1}
\]
Equations (9) and (11) represent the constrained optimization to be solved in estimating the distance between the clouds (objective function). Two sub-cases arise in the computation of this cost: when \( |\Xi_1| = |\Xi_2| \) (i.e. clouds are of equal cardinalities) and when \( |\Xi_1| \neq |\Xi_2| \).

3.2.1.1 Resampled particle clouds of equal cardinalities
For equal cardinalities, \( M_1 = M_2 = M \) and all particles in both clouds have identical weights 1/M. Consequently the cloud-to-cloud association cost can be simplified to the optimal objective function of a simple 2-D assignment (i.e. \( W \) to have elements in \( \{0, 1\} \) only). The minimization (9) becomes
\[
d_W^p(\Xi_1, \Xi_2) = \min_{\delta} \sqrt{\frac{1}{M} \sum_{i=1}^{M} d(\xi_s, \xi_s)}
\]
with \( p = 1, 2 \), where \( \sigma_i \) is a permutation of \( i = 1, \ldots, M \). Equation (12) represents the 2-D assignment problem (without dummy association allowed, as all particles should be assigned), which can be solved by finding optimum \( \sigma_i \) permutation using auction algorithms [12]-[13].

### 3.2.1.2 Resampled particle clouds of different cardinalities

For clouds of different cardinalities, \( M_1 \neq M_2 \), even though particles in each cloud have equal weights (i.e. \( 1/M_1 \) for \( s = 1 \) and \( 1/M_2 \) for \( s = 2 \)), they are different than particles weights in the other cloud. In order to reduce this problem to a similar 2-D problem as in 3.2.1.1, both clouds need to be reduced to the same number of particles and of equal weights (in order to allow one-to-one associations only and therefore \( W \) with elements in \( \{0,1\} \)). Using a method similar to the one in [11], each particle \( i = 1 \ldots M_1 \) in the cloud \( \Xi_1 \) will be resampled into \( \Xi_1 N_1 = \Xi_1 \frac{M_1}{\gcd(M_1, M_2)} \) new particles, where \( \gcd \) stands for the greatest common divisor, thus obtaining a cloud of \( \Xi_1 \) particles of weights \( w = \frac{\gcd(M_1, M_2)}{M_1} \). The same is done to the particles in the cloud \( \Xi_2 \), in consequence obtaining \( \Xi_2 N_2 = \Xi_2 \frac{M_2}{\gcd(M_1, M_2)} \) particles of same weight \( w \). At this stage the cost can be computed following the same method in 3.2.1.1.

### 3.2.2 Using clouds of unresampled particles

For two given particle filters \( s = 1, 2 \), with \( M_1 \) respectively \( M_2 \) total particles per filter at a given time instance, the unresampled particles \( \{ \tilde{w}_i^s, \tilde{z}_i^s \}_{i=1}^{M_s} \) have unequal weights. By its nature, for the PHD-PF, the total sum of particles weights is equal to the estimated number of targets (rational number before rounding) \( \sum_{i=1}^{M_s} \tilde{w}_i^s = \tilde{N}_s \). For the labeled particle filter, the number of targets is also given (integer) by the number of labels, with only the labels of confirmed tracks counted. A labeled track is considered confirmed only if it has measurements associated to it in several consecutive frames. Only these particles, labeled as pertaining to confirmed tracks, are propagated for fusion. The cost of associating any two clouds is computed as:

\[
\text{cost} (\Xi_1, \Xi_2) = \min \sqrt{\sum_{i=1}^{M_1} \sum_{j=1}^{M_2} w_{ij} d \left( \tilde{\xi}_i, \tilde{\xi}_j \right)} \tag{13}
\]

satisfying the constraints

\[
\sum_{j=1}^{M_2} w_{ij} = \tilde{w}_i^1, \quad j = 1..M_2 \tag{14}
\]

\[
\sum_{i=1}^{M_1} w_{ij} = \tilde{w}_j^2, \quad i = 1..M_1 \tag{15}
\]

Constraints (14) and (15) assure that each particle in both clouds is fully assigned (using its full weight) to particles from the other cloud. The minimization problem (13)-(15) is a multiassignment problem, similar to the transportation problem, which is a special case of linear programming.

### 3.3 Multiassignment solution using linear programming interior point methods

In order to translate (13)-(15) in a classical linear program

\[
\min d^T w \tag{16}
\]

subject to

\[
Aw = b, \quad w \geq 0, \quad w \leq 1 \tag{17}
\]

we use the following for \( d \in \mathbb{R}^{M_1+M_2}, w \in \mathbb{R}^{M_1} \), \( A \in \mathbb{R}^{M_1+M_2, M_1+M_2}, b \in \mathbb{R}^{M_1+M_2} \):

\[
d = [d_{11}, d_{12}, \ldots, d_{1M_1}, d_{21}, d_{22}, \ldots, d_{2M_1}, \ldots, d_{M_1M_2}, \ldots, d_{M_2M_2}]
\]

represents the vector of distances \( d_{ij} = d \left( \tilde{\xi}_i, \tilde{\xi}_j \right) \),

\[
w = [w_1, w_2, \ldots, w_{M_1}, w_{M_1+1}, \ldots, w_{M_1+M_2}, \ldots, w_{M_1+M_2}]
\]

is the content of transportation matrix \( W \) in vector form,

\[
A = \begin{bmatrix} 1 & 1 & \ldots & 1 & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\
0 & 0 & \ldots & 0 & 1 & 1 & \ldots & 1 & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 1 & 1 & \ldots & 1 \\
1 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \\
0 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 1 & 0 & 0 & \ldots & 1 & 0 & 0 & \ldots & 1 \end{bmatrix}
\tag{18}
\]

and \( b = [\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_{M_1}, \tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_{M_2}]^T \) being the vector of particle weights of both clouds.

A sample package solving the above minimization is LIPSOL (Linear-programming interior point solver) [14].

### 3.4 Multiassignment solution as sequential 2D associations

For clouds with very high particle counts, a suboptimal way of performing the multiassignment between
unresampled clouds of different cardinalities is as a sequence of 2D unweighted assignments.

### 3.5 Cloud to cloud fusion

For two associated particle clouds (as corresponding to same target), $\Xi_1$ and $\Xi_2$, of unresampled particles with normalized weights (each cloud weights sum to one), three methods for computing the estimated fused state are presented below.

**Method 1** – Derives from a direct combination of particles in both clouds, as below

$$\hat{x} = 0.5 \left( \sum_{i=1}^{M_1} \tilde{z}_i^1 \tilde{w}_i^1 + \sum_{j=1}^{M_2} \tilde{z}_j^2 \tilde{w}_j^2 \right)$$  \hspace{1cm} (19)

**Method 2** – Clouds are weighted by their sample covariances, following the information filter approach [1]:

$$\hat{x} = \left( P_1^{-1} + P_2^{-1} \right)^{-1} \left( P_1^{-1} \sum_{i=1}^{M_1} \tilde{z}_i^1 \tilde{w}_i^1 + P_2^{-1} \sum_{j=1}^{M_2} \tilde{z}_j^2 \tilde{w}_j^2 \right)$$  \hspace{1cm} (20)

where $P_1, P_2$, are sample covariances of clouds $\Xi_1$ and $\Xi_2$, computed through:

$$P_s = \sum_{i=1}^{M_s} \tilde{w}_i^s (\tilde{z}_i^s - \overline{\tilde{z}}^s) (\tilde{z}_i^s - \overline{\tilde{z}}^s)^T$$  \hspace{1cm} (s = 1, 2)  \hspace{1cm} (21)

where $\overline{\tilde{z}}_s$ is the sample weighted mean. In this method independence of the estimation errors of both clouds is assumed.

**Method 3** – Fusion is carried out making use of the estimated sample cross-covariance matrices between clouds, $P_{12}$, $P_{21}$. These are estimated based on the transportation matrix $W$ between clouds, derived in the association step

$$P_{12} = \sum_{i=1}^{M_1} \sum_{j=1}^{M_2} \tilde{w}_i^1 (\tilde{z}_j^2 - \overline{\tilde{z}}^1) (\tilde{z}_i^2 - \overline{\tilde{z}}^2)^T$$  \hspace{1cm} (22)

and used as

$$P = P_1 + P_2 - P_{12} - P_{21}$$  \hspace{1cm} (23)

in the computation of fused estimate

$$\hat{x} = (P - P_{12}) \sum_{i=1}^{M_1} \tilde{z}_i^1 \tilde{w}_i^1 + (P - P_{21}) \sum_{j=1}^{M_2} \tilde{z}_j^2 \tilde{w}_j^2$$  \hspace{1cm} (24)

### 4 Simulation

The simulation scenario contains several targets which occur and disappear at different moments, as shown through trajectories in Figure 1 below.

Figure 1 Targets evolution in time in the x, y directions.

Figure 2 Targets and measurements in xy plane for the entire scenario simulation.

Figure 2 shows the true measurements (circles) and false alarms (crosses). The two local estimators used at the two sensors $s = 1, 2$, are PHD particle filters with track-valued (labeled) particle clouds. The equations and implementation of the track-labelled PHD particle filter follow the one in [10], here extended to two-dimensional tracking, briefly described below, with $k$ denoting the time sample, and filter index $s$ dropped. Particle states contain two-dimensional position and velocity in Cartesian coordinates, while measurements provide only position information.

**Initialization**

For $i = 1, ..., L_1$ sample position $(x, y)$ of particles as $\tilde{z}_i^{(x,y)} \sim N(0, \sigma_z)$, where $Z_i$ is the set of all initial measurements, and $N$ stands for normal distribution. The velocity components of each particle are initialized uniformly distributed within maximum velocities allowed for a target $\tilde{v}_i^{(x,y)} \sim U(-v_{\text{max}}, v_{\text{max}})$, with $v_{\text{max}} = 6$ here.

Initial weights are computed as $\tilde{w}_i^1 = \tilde{z}_i^1 / (L_1 \cdot n_i)$, where
$ς_i$ is the estimated number of targets and $n_1$ is the number of measurements.

**Prediction**

For $i = 1, \ldots, L_{k-1}$ (confirmed particles at time $k-1$), sample $\xi^i_k \sim q(\cdot | \xi^i_{k-1})$, where $q$ is the proposal density, which translates into

$$\xi^i_k = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \xi^i_{k-1} + \nu_k$$  \hspace{1cm} (25)

with $\nu_k$ denoting the process noise ($v_{x,y} = 1.0$, $v_{x,vy} = 0.5$) and $T = 1$ the sampling time step. Assuming no target spawning, weights are computed as $w^i_k = w^i_{k-1} \cdot e^i_k$, where $e^i_k = 0.95$ is the probability of target survival.

For $i = L_{k-1}+1, \ldots, L_{k-1}+J_k$ newborn particles are sampled based on new measurements in the set $Z_k$. For this purpose $Z_k$ is partitioned with respect to $Y_{k-1}$, the set of tracks at time $k-1$, into

$Z_{k,in} = \{ z^i_k | z^i_k \text{ in validation region of a track in } Y_{k-1} \}$

and

$Z_{k,out} = \{ z^i_k | z^i_k \text{ in no validation region of any track in } Y_{k-1} \}$

Particles position components are sampled for all measurements as $\xi^i_k(\cdot,x,y) \sim N(z,\sigma_z)$, where $\sigma_z$ is the standard deviation of measurement position error. For measurements in $Z_{k,in}$ particles velocity components are sampled from $\xi^i_k(\cdot,v_{x,y}) \sim N(\tilde{x}^i_{k-1},\sigma)$, where $\tilde{x}^i_{k-1}$ is the velocity estimate of track $n$ at time $k-1$, in which region the measurement $z$ was accepted. For measurements in $Z_{k,out}$ particles velocity components are sampled uniformly within $[-v_{x,max}, v_{x,max}]$ and $[-v_{y,max}, v_{y,max}]$. The weights of all newborn particles are computed as

$$w^i_{k|k-1} = \frac{b_k(\xi^i_k)}{J_k \cdot p_k(\xi^i_k | Z_k)}$$  \hspace{1cm} (26)

where $b_k(\xi^i_k)$ is the intensity function of the point process (considered Poisson here) generating the target birth and $p_k(\xi^i_k | Z_k)$ the proposal density. By considering normal densities for $b_k$ and $p_k$ as in [8],[10] the weights are computed as $1 / J_k$, where the constant intensity of the point process was taken as $I = 0.2$ (e.g. one target birth expected every 5th sampling time on average).

**Update**

For $i = 1, \ldots, L_k$, where $L_k = L_{k-1} + J_k$, the updated weights are computed as

$$w^i_k = w^i_{k|k-1} \cdot \left[ 1 - P_D + \sum_{z \in Z_{k,in}} \frac{\beta P_D h(z | \xi^i_{k|k-1})}{\lambda \cdot c + \beta C_i(z)} + \sum_{z \in Z_{k,out}} \frac{\beta P_D h(z | \xi^i_{k|k-1})}{\lambda \cdot c + \beta C_i(z)} \right]$$  \hspace{1cm} (27)

where $P_D$ is the probability of detection (considered unity here), $h$ is the sensor measurement function, considered as normal with standard deviation 2.5 for both sensors, $\lambda = 4$ is the average rate of clutter points per scan, $c = 0.000025$ is the uniform clutter density for the whole surveyed area, $\beta_1=1.1, \beta_2 = \beta_1/25$ are design parameters as in [10] and

$$C_i(z) = \sum_{i=1}^{J_k} P_D h(z | \xi^i_{k|k-1})w^i_{k|k-1}$$  \hspace{1cm} (28)
(Root Mean Square Errors) computed over time for each run and averaged over 100 MC runs are shown in tables below for single filter and fused results.

Table 1 RMSE (x, vx, y, vy) for filter 1.

<table>
<thead>
<tr>
<th>Target</th>
<th>Target 2</th>
<th>Target 3</th>
<th>Target 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE x</td>
<td>1.7658</td>
<td>1.8079</td>
<td>1.9992</td>
</tr>
<tr>
<td>RMSE vx</td>
<td>2.7756</td>
<td>1.6699</td>
<td>1.8474</td>
</tr>
<tr>
<td>RMSE y</td>
<td>2.2746</td>
<td>2.0457</td>
<td>1.8778</td>
</tr>
<tr>
<td>RMSE vy</td>
<td>1.1663</td>
<td>1.3235</td>
<td>0.7606</td>
</tr>
</tbody>
</table>

Table 2 RMSE (x, vx, y, vy) for Method 1 of fusion.

<table>
<thead>
<tr>
<th>Target</th>
<th>Target 2</th>
<th>Target 3</th>
<th>Target 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE x</td>
<td>1.3192</td>
<td>1.3582</td>
<td>1.5353</td>
</tr>
<tr>
<td>RMSE vx</td>
<td>2.6851</td>
<td>1.5078</td>
<td>1.8009</td>
</tr>
<tr>
<td>RMSE y</td>
<td>1.4959</td>
<td>1.4196</td>
<td>1.2962</td>
</tr>
<tr>
<td>RMSE vy</td>
<td>1.1812</td>
<td>1.1471</td>
<td>0.7752</td>
</tr>
</tbody>
</table>

Table 3 RMSE (x, vx, y, vy) for Method 2 of fusion.

<table>
<thead>
<tr>
<th>Target</th>
<th>Target 2</th>
<th>Target 3</th>
<th>Target 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE x</td>
<td>1.7658</td>
<td>1.3442</td>
<td>1.5765</td>
</tr>
<tr>
<td>RMSE vx</td>
<td>2.6158</td>
<td>1.4193</td>
<td>1.7308</td>
</tr>
<tr>
<td>RMSE y</td>
<td>1.5531</td>
<td>1.4254</td>
<td>1.2717</td>
</tr>
<tr>
<td>RMSE vy</td>
<td>1.1109</td>
<td>1.0945</td>
<td>0.7242</td>
</tr>
</tbody>
</table>

From results above it can be seen that position estimates are improved more than velocity estimates, and that the usage of sample covariances in Method 2 does not increase consistently the accuracy.

4.1 Monte Carlo simulation for Gaussian distributed clouds

In order to assess the proposed association cost for clouds of different distributions, Monte Carlo simulations were performed for estimating the association cost of two clouds, normally distributed (corresponding to the unresampled case), with different standard deviations. The number of particles considered was M = 200 on both clouds. The cost distributions obtained in 50 runs are compared with the distribution of the distance between the clouds centers estimated using the weighted sum of particles

\[
d_{12} = \left\| \sum_{i=1}^{\mu} w_i \xi^i - \sum_{j=1}^{\nu} w_j \xi^j \right\| \tag{29}
\]

From the histograms of the distances computed using (9) and (29), compared in Fig. 5-9 for clouds of different distributions, it can be seen that the cost computed based on (29) is insensitive to the variance of the clouds, as only the first order estimate enters (29), while the one computed based on (9) increases with the difference between clouds standard deviation (variance). While these simulations were using Gaussian clouds, however the cost in (9) is not making any assumption on normality, therefore differences in higher order moments of the clouds are expected to be reflected in the costs as well.
5 Conclusion

A method of fusing density estimates of particle filters at the particle cloud level was proposed. Cloud-to-cloud association costs depending on particles types were developed together with several fusion methods. Results obtained for a simulated multitarget scenario of dynamic cardinality were presented. The fusion at particle level results in a fused estimate of target density, not only in a fused estimate of the target state; the improvement of the target density estimate will be further investigated.

References


