COVER PAGE

9th International Command and Control Research and Technology Symposium

Coalition Transformation: An Evolution of People, Processes and Technology to Enhance Interoperability

September 14–16, 2004
Copenhagen, Denmark

Title of paper:

Methods and System Design of the IFD03 Information Fusion Demonstrator

Name of Authors:

Johan Schubert, Christian Mårtenson, Hedvig Sidenbladh, Pontus Svenson, Johan Walter

Point of Contact:

Johan Schubert

Organization:

Department of Data and Information Fusion,
Division of Command and Control Systems,
Swedish Defence Research Agency

Complete address:

SE-172 90 STOCKHOLM, Sweden

Telephone/Fax Number:

Tel +46 8 5550 3702
Fax +46 8 5550 3700

E-mail:

schubert@foi.se

URL:

http://www.foi.se/fusion

Topic:

Network Centric Applications

Methods and System Design of the IFD03 Information Fusion Demonstrator

Johan Schubert*, Christian Mårtenson, Hedvig Sidenbladh,
Pontus Svenson, and Johan Walter

Department of Data and Information Fusion
Division of Command and Control Systems
Swedish Defence Research Agency
SE 172 90 Stockholm, Sweden

{schubert, cmart, hedvig, ponsve, johanw}@foi.se
http://www.foi.se/fusion/

Abstract

The Swedish Defence Research Agency has developed a concept demonstrator for demonstrating information fusion methodology focused on intelligence processing at the division level for a future Network Based Defence (NBF) / Network Centric Warfare (NCW) C4ISR system. The demonstrator integrates force aggregation, particle filtering and sensor allocation methods to construct, dynamically update and maintain a situation picture.

1 Introduction

The Swedish Defence Research Agency (FOI) has developed a concept demonstrator called the Information Fusion Demonstrator 2003 (IFD03) for demonstrating information fusion methodology for a future Network Based Defence (NBF) / Network Centric Warfare (NCW) C4ISR system. The demonstrator's focus is on the intelligence processing at the division level in an ground warfare scenario.

The demonstrator consists of two main parts, an analysis module and a simulation module. In this paper we give a description of the system design and fusion methods of the IFD03 analysis module. This paper is a companion paper to one published at The Sixth International Conference on Information Fusion (Svensson and Hörling, 2003) describing the scenario, sensors and simulation of the simulation module. Together they fully describe the IFD03.

In a recent work, Ahlberg et al. (2004) describe the essential experience from the use and development of IFD03.

*Corresponding author.

In Section 2 we describe the fusion methods of the IFD03 analysis module. Section 2.1 focuses on force aggregation to construct a situation picture. Following this we describe particle filter tracking to dynamically update the situation picture (Section 2.2). In Section 2.3 we discuss sensor management for maintaining the situation picture. In Section 3 we give a description of the general design of IFD03. Finally, the IFD03 visualizer is described in Section 4.

2 Methods

The analysis module has three main tasks and uses four different methods. The tasks are force aggregation, ground vehicle tracking and sensor allocation. They are performed using Dempster-Shafer clustering and Dempster-Shafer template matching for force aggregation, probability hypothesis density (PHD) particle filtering for ground vehicle tracking, and random set simulation for sensor allocation.

2.1 Force Aggregation

In the force aggregation we use intelligence reports with given position, time, and type information. We define force aggregation as a combination of two processes. First, an association of intelligence reports, objects or units (depending on hierarchical level) by a clustering process. Secondly, a classification of cluster content through a comparison with templates.

We start by evaluating all pairs of intelligence reports, to find whatever is against that two reports are referring to the same object: Wrong type of vehicle? Is distance too long or too short? Wrong direction? Wrong relative positions? etc. This yields a potential conflict between each pair of intelligence reports. A conflict matrix is constructed and supplied to the clustering algorithm. We use the Dempster-Shafer clustering algorithm (Schubert, 1993; Bengtsson and Schubert, 2001; Schubert, 2003a) to partition the set of reports into subsets, each subset corresponding to one object, and classify the objects by fusing all intelligence using Dempster's rule. This method continues upwards level by level. At the vehicle to platoon level, vehicles are clustered and groups of vehicles are classified using Dempster-Shafer matching against templates (Schubert, 2003b). At all levels in clustering and template matching we use the full descriptive power of Dempster-Shafer theory, carrying several alternative hypotheses represented by a belief function that is the result of fusing all intelligence in the cluster. Each alternative hypothesis is matched and evaluated against all templates and a weighted average is calculated for each potential template.

A screen picture from the demonstrator showing the result of automated force aggregation at the platoon level is shown in Figure 1. This method is currently developed up to the battalion level. A few other approaches to force aggregation than the one described here are (Biermann, 1998; Lorenz and Biermann, 2002; Johnson and Chaney, 1999).

2.1.1 Conflict Matrix

There is one conflict matrix for each aggregation level. The conflict matrix $\mathbf{C_{ij}}$ contains the conflict between the entities i and j. The matrix is symmetric and contains zeros on the diagonal.

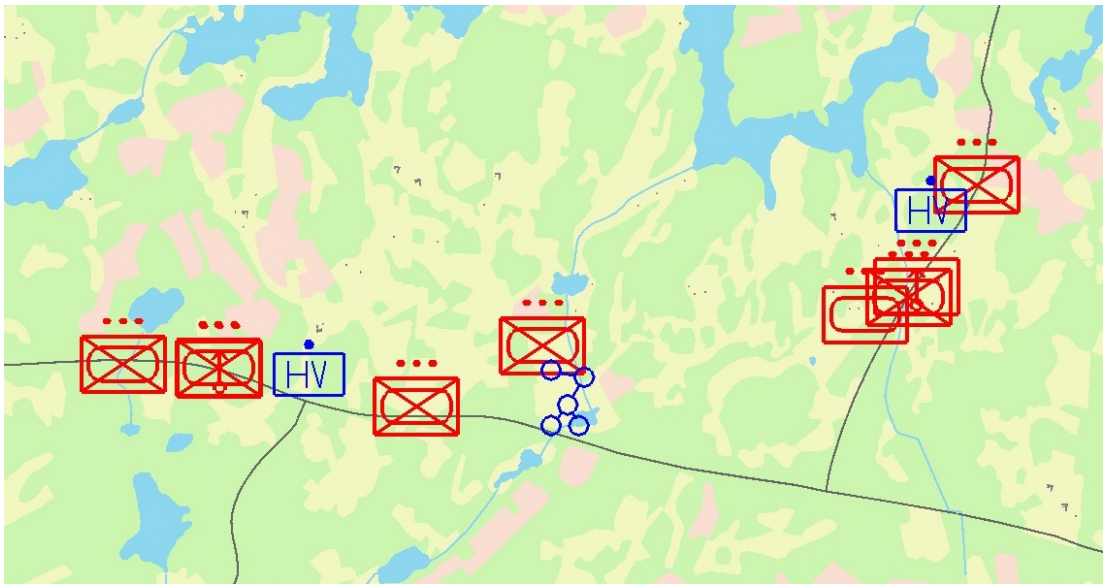

Figure 1: Force aggregation of vehicles into platoons.

There are two different ways of computing the conflict matrix. When computing the conflict matrix for the reports, the conflict between two reports is based on their type, on how fast a vehicle must travel in order to cause the two reports and on how much their directions differ.

When computing the conflict matrix for vehicles and units, the conflicts are based on doctrine data that specify how far apart the objects appear within their unit.

All entities – reports, vehicles and units – contain a classification of types, $\mathbf{T}$. However, the classification is uncertain, so we can only give probabilities for sets of types. The basic belief mass supporting that an entity is of type $\mathbf{A} \in \mathbf{T}$ is denoted $m(\mathbf{A})$. The believed type of an entity is defined by all $m(\cdot)$ associated with the entity.

The conflict matrix for the reports. The conflict $\mathbf{C}$ is computed from the type conflict $\mathbf{C^t}$, the speed conflict $\mathbf{C^s}$ and the direction conflict $\mathbf{C^d}$.

$$\mathbf{C} = 1 - (1 - \mathbf{C^t})(1 - \mathbf{C^s})(1 - \mathbf{C^d}) \tag{1}$$

The type conflict between the entities E_i and E_j is given by Dempster's rule of combination:

$$\mathbf{C^t}_{E_i E_j} = \sum_{\substack{A \in E_i, B \in E_j \\ A \cap B = \emptyset}} m(A) \cdot m(B) \tag{2}$$

The speed conflict, $\mathbf{C^s}$, is obtained by calculating the speed at which a vehicle must travel, in order to have caused the reports. The speed is then normalized, as shown in Figure 2, to get the conflict.

The direction conflict, $\mathbf{C^d}$, is obtained in a similar way by computing the difference between the two directions in the reports. For details see (Cantwell et al., 2001).

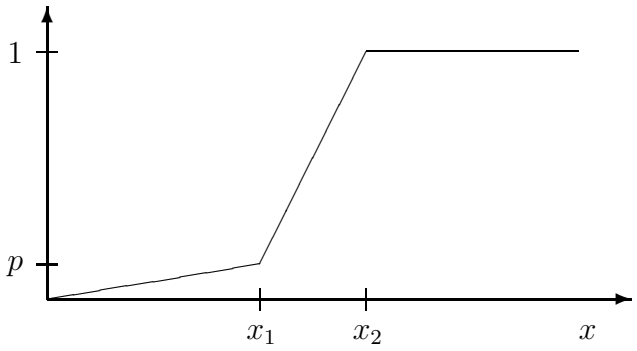

Figure 2: The normalization function.

The conflict matrix for the vehicles and units. If the believed types of two entities do not appear in any common unit template, there should be a large conflict. Otherwise the conflict is based on their relative distance and the maximum allowed distance for all units, according to the doctrine data.

For each level – vehicles, platoons, companies... – a destination matrix, **DM** is defined. The allowed distance between $\mathbf{T}_a$ and $\mathbf{T}_b$ is $\mathbf{DM}_{ab}$.

For types $\mathbf{T}_a$ and $\mathbf{T_b}$ that do not appear in any common unit template, $\mathbf{DM}_{ab} = -1$ The conflict between entity E_i and E_j is given by:

$$\mathbf{C}_{E_iE_j} = \sum_{A \in E_i, B \in E_j} C_{AB} \cdot m(A) \cdot m(B) \tag{3}$$

where A and B are focal elements, and

$$C_{AB} = \min_{a \in A, b \in B} c_{ab} \tag{4}$$

where a and b are types, and

$$c_{ab} = \begin{cases} 1 & d > \mathbf{DM}_{ab} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where d is the distance between entities E_i and E_j.

2.1.2 Clustering

We have developed a method for managing intelligence reports that concern different objects. This is the situation when it is not known a priori to which object each intelligence report is related. The intelligence reports are clustered into groups that should be handled independently.

In (Bengtsson and Schubert, 2001) a method for clustering intelligence reports based on their pairwise conflict was developed. This method was extended into a method capable of also handling pairwise attractions (Schubert, 2003a).

Such evidence is not generated intrinsically in the same way as conflicts. Instead, we assume that it is given from some external source.

As an example let us look at a real-world problem from intelligence analysis that we are studying (Cantwell et al., 2001). In intelligence analysis we may have conflicts between two different

intelligence reports about sighted objects, indicating that two objects probably do not belong to the same unit (cluster). Such conflicts arise when reports about objects are compared under the hypothesis that they refer to the same unit, e.g., report object types, times, positions and directions may be incompatible given constraints about unit structure. At the same time we may have information from communication intelligence as an external source, indicating that the two objects probably do belong to the same unit (cluster) as they are in communication. Such information is made available from studying communication patterns obtained through COMINT, e.g., if two objects are transmitting in sequence we may calculate a probability that they are in communication and thus belong to the same unit structure.

As conflicts push reports apart (into different clusters) and attractions pull them together (into the same cluster), using both leads to an improved clustering result and faster computation.

We use Dempster-Shafer theory (Dempster, 1968; Shafer, 1976) and can then use the conflict of Dempster's rule when the two intelligence reports are combined as an indication of whether they belong together. Combining two mass functions is done by calculating their orthogonal combination using Dempster's rule.

Combining m_1 and m_2 of the two pieces of evidence may result in a conflict defined as

$$c_{ij} = \sum_{A_i \cap B_j = \emptyset} m_1(A_i) B_2(B_j) \tag{6}$$

whenever there are at least one focal element from $\{A_i\}$ and one focal element from $\{B_j\}$ such that $A_i \cap B_j = \emptyset$. This number is between zero and one. The higher this value is, the more conflict there is between the two intelligence reports.

This conflict is the basis for separating intelligence reports into clusters. A high conflict between the two intelligence reports is an indication of repellency that they do not belong to the same cluster. The higher the conflict is, the less credible that they belong to the same cluster.

Consider all pairwise conflicts between the intelligence reports in a cluster χ_a, where c_{ij} is the conflict of Dempster's rule when combining e_i and e_j.

In addition to the conflicting metalevel evidence induced by the internal conflict between intelligence reports belonging to the same cluster, in many applications it is important to be able to handle attracting metalevel evidence from some external source stating that two intelligence reports concern the same object.

Such an external metalevel evidence is represented as a pairwise piece of evidence, where p_{ij} is a degree of attraction.

The best partitioning of all intelligence reports is found by a cluster process (Schubert, 2004) that minimizes a function $m_{\{\chi_a\} \oplus \chi}(\neg AdP)$ with a proposition that this is not an "adequate partition" AdP.

Approximately this function can be written as

$$m_{\{\chi_a\} \oplus \chi}(\neg AdP) \approx [1 - \prod_{(ij)|\forall a. e_i \wedge e_j \notin \chi_a} (1 - p_{ij}) \times \prod_a \prod_{(ij)|e_i \wedge e_j \in \chi_a} (1 - c_{ij})] \tag{7}$$

The clustering process of the intelligence reports is done by neural clustering using Potts spin theory (Potts, 1952; Chaikin and Lubensky, 1995). The Potts spin problem consists of minimizing

an energy function

$$E = \frac{1}{2} \sum_{i,j=1}^{N} \sum_{a=1}^{q} (J_{ij}^{-} - J_{ij}^{+}) S_{ia} S_{ja} \tag{8}$$

by changing the states of the spins S_{ia}'s, where $S_{ia} \in \{0, 1\}$ and $S_{ia} = 1$ means that report i is in cluster a. N is the number of intelligence reports and q the number of clusters. This model serves as a clustering method if J_{ij}^{-} is used as a penalty factor when report i and j are in the same cluster, and J_{ij}^{+} when they are in different clusters.

The minimization is carried out by simulated annealing. For computational reasons we use a mean field model, where $V_{ia} =< S_{ia} >$, $V_{ia} \in [0, 1]$, in order to find the minimum of the energy function. The Potts mean field equations are formulated (Peterson and Söderberg, 1989) as

$$V_{ia} = \frac{e^{-H_{ia}[V]/T}}{\sum_{b=1}^{q} e^{-H_{ib}[V]/T}} \tag{9}$$

where

$$H_{ia}[V] = \sum_{j=1}^{N} J_{ij} V_{ja} - \gamma V_{ia} \tag{10}$$

V, T, H_{ib} and γ are parameters of the annealing process.

In order to map the function $m_{\{\chi_a\} \oplus \chi}(\neg AdP)$ onto a Potts spin neural network we must rewrite the function as a sum of terms that is to be minimized.

Minimizing this function is equivalent to

$$min \sum_{(ij)|\forall a. e_i \wedge e_j \notin \chi_a} -log(1 - p_{ij}) + \sum_{a} \sum_{(ij)|e_i \wedge e_j \in \chi_a} -log(1 - c_{ij}) \tag{11}$$

In order to apply the Potts model to Dempster-Shafer clustering we use interactions $J_{ij}^{-} = -log(1 - c_{ij})\delta_{|A_i \cap A_j|}$ and $J_{ij}^{+} = -log(1 - p_i j)\delta_{|A_i \cap A_j|}$ in the energy function (Equation (8)), where δ is the Kronecker delta with

$$\delta_{|A_i \cap A_j|} \equiv \begin{cases} 1 & A_i \cap A_j = \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where A_i and A_j are two focal elements and

$$\delta_{ij} \equiv \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{13}$$

in Figure 3.

By minimizing the energy function we also minimize $m_{\{\chi_a\} \oplus \chi}(\neg AdP)$. In Figure 3 an algorithm for minimizing the energy function through iteration of Equations (9) and (10) is shown.

2.1.3 Number of Clusters

In order to find the correct number of clusters we run the clustering process for different number of clusters and observe the remaining conflict after convergence of the process. If the number of

INITIALIZE

K (number of clusters); N (number of intelligence reports)

$J_{ij}^{-} = -log(1 - c_{ij})\delta_{|A_i \cap A_j|} \forall i, j;$

$J_{ij}^{+} = -log(1 - p_{ij})(1 - \delta_{|A_i \cap A_j|}) \forall i, j;$

$s = 0; t = 0; \epsilon = 0.001; \tau = 0.9; \gamma = 0.5;$

$T^0 = T_c$ (a critical temperature) $= \frac{1}{K} \cdot max(-\lambda_{min}, \lambda_{max})$,
where λ_{min} and λ_{max} are the extreme eigenvalues of M,
where $M_{ij} = J_{ij}^{-} - J_{ij}^{+} - \gamma\delta_{ij}$;

$V_{ia}^0 = \frac{1}{K} + \epsilon \cdot rand[0, 1] \forall i, a;$

REPEAT

REPEAT-2

$\forall i$ Do:

$$H_{ia}^s = \sum_{j=1}^{N} (J_{ij}^{-} - J_{ij}^{+}) V_{ja}^{\begin{cases} s+1 & j<i \\ s & j \geq i \end{cases}} - \gamma V_{ia}^s \forall a;$$

$$F_i^s = \sum_{a=1}^{K} e^{-H_{ia}^s / T^t};$$

$$V_{ia}^{s+1} = \frac{e^{-H_{ia}^s / T^t}}{F_i^s} + \epsilon \cdot rand[0, 1] \forall a;$$

$s = s + 1;$

UNTIL-2

$\frac{1}{N} \sum_{i,a} |V_{ia}^s - V_{ia}^{s-1}| \leq 0.01;$

$T^{t+1} = \tau \cdot T^t;$

$t = t + 1;$

UNTIL

$\frac{1}{N} \sum_{i,a} (V_{ia}^s)^2 \geq 0.99;$

RETURN

$\{\chi_a | \forall S_i \in \chi_a, \forall b \neq a V_{ia}^s > V_{ib}^s\};$

Figure 3: Pseudo code for clustering algorithm.

clusters was too small we will have a high conflict since we are trying to squeeze a large problem into a too small number of clusters. On the other hand if the number of clusters is too large we will have a very small conflict from measurement errors. Thus, when the logarithm of the total weight of conflict is plotted in a graph it is often easy to find the inflexion point corresponding to the correct number of clusters, Figure 4.

2.1.4 Classification

We allow for any number of nonspecific and uncertain propositions in each intelligence report. With this we may handle any general intelligence report.

The classification process deals with intelligence reports on a cluster-by-cluster basis. Looking at intelligence in one of the clusters, the classification from intelligence using templates takes place in two phases. First, we combine all intelligence reports within the cluster, and secondly, we compare the combined intelligence with all available templates.

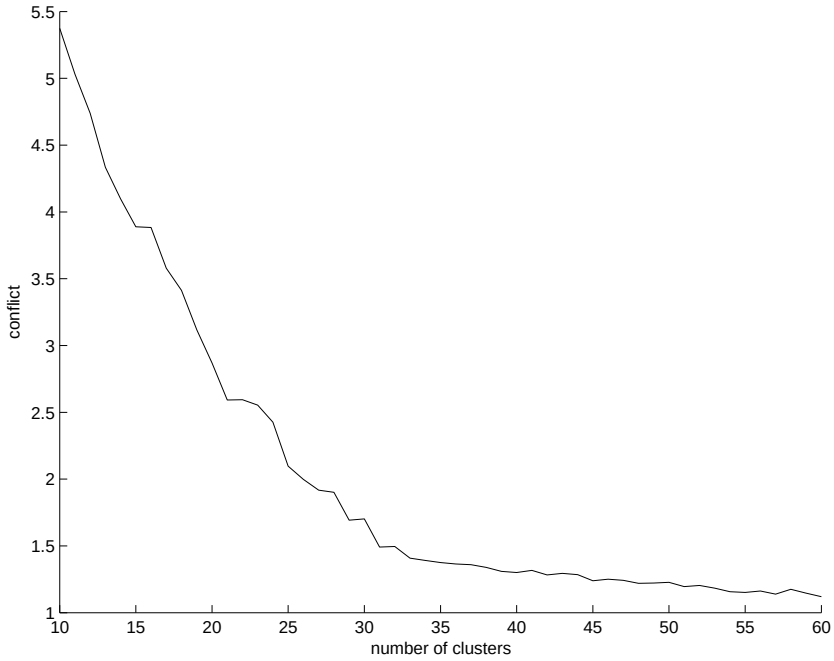

Figure 4: Logarithm of the total weight of conflict.

In the combination of intelligence a special concern is the representation used. As the reports in general are not reports about the same object or group of objects, we can not use a simple representation dealing only with object type. Instead, we must use a more advanced representation that allows us to keep track of different objects and their possible types. Intelligence reports that actually are referring to the same object or group of objects are precombined, and henceforth viewed as one intelligence report. When this is done, all intelligence reports in the cluster under investigation can be combined, giving us the possibility to investigate the different resulting hypotheses regarding force composition.

When selecting a template for the current cluster we search for a maximum matching between template and fused intelligence. Since intelligence consists of multiple alternative hypotheses with an accompanying uncertainty, we must take every hypothesis into account, to its degree of uncertainty, when evaluating a template. As these hypotheses are also nonspecific regarding object type, i.e., they refer to a subset of all possible types instead of to a single type, we cannot expect a perfect matching for each type of object in the template. Instead, we look for a possibility of matching between intelligence and template, i.e., the absence of conflicts in number of items between what the intelligence propose and what each available template requests for all subsets of types. With this measure we can select a template for intelligence with nonspecific propositions.

Here, we will investigate the representation and combination of all intelligence referring to the same unit. We assume that a number of intelligence reports about different sets of objects are available. These reports have already been partitioned into subsets where each subset corresponds to a unit on one higher hierarchical level (Bengtsson and Schubert, 2001; Schubert, 2000). Let us hereafter focus on one such subset χ_a and the aggregation of the intelligence in this subset.

Let TY be a set of all possible types of objects $\{TY_x\}$ where TY_x is a type of vehicle or a type of unit depending on which hierarchical level we are at.

The frame of discernment when fusing reports regarding different sets of objects that should be combined as fragments of a larger unit structure becomes

$$\Theta_{I_a} = \{\langle x_1, x_2, \ldots, x_{|I_a|}\rangle\} \tag{14}$$

where $x_i = (x_{i\bullet}n, x_{i\bullet}pt)$ is information regarding the i^{th} set of objects with $x_{i\bullet}n \subseteq \{1, \ldots, N_{C_a}\}$ and $x_{i\bullet}pt \subseteq TY$.

Comparing templates having specific propositions that are certain in what they are requesting with intelligence propositions that are not only uncertain but may also be nonspecific in what they are supporting can be a difficult task. The idea we use to handle this problem is to compare a candidate template with intelligence from the perspective of each and every subset of all possible types of objects TY.

In doing this we investigate how much support a subset of TY receives both directly and indirectly from intelligence and template, respectively. The support for a subset of TY is summed up from all propositions that are equal to or itself a subset of this subset of TY. This is similar to the calculation of belief from basic probability numbers in Dempster-Shafer theory, except that we are not summing up basic probability numbers but natural numbers representing the number of objects of the proposed types.

Let T be a set of all available templates $\{T_i\}$. Each template is represented by any number of slots S_i^j where $S_{i\bullet}^j pt \subseteq TY$ is a possible type from the set TY and $S_{i\bullet}^j n$ is the number of that type in T_i.

Based on the combination of all intelligence reports we evaluate all templates of $\{T_i\}$.

As we have several different alternative propositions in the intelligence regarding the type of objects and their corresponding number of objects, we need to compare each potential template with these alternatives and let each proposition influence the evaluation. For each template we find a measure of fitness between the template and each proposition in the intelligence, separately.

We then make a linear combination where each measure of fitness is weighted by the basic probability number of that proposition,

$$m_{\bigoplus J_a}(\langle x_1, x_2, \ldots, x_{|I_a|}\rangle) \tag{15}$$

We get

$$\begin{aligned}\pi_{\bigoplus J_a}(T_i) = \frac{1}{2} \sum_{\langle x_1, x_2, \ldots, x_{|I_a|}\rangle} & m_{\bigoplus J_a}(\langle x_1, x_2, \ldots, x_{|I_a|}\rangle) \\ & \left[\max_{n \in SC_a(TY)} \left\{\min\left[\frac{n}{ST_i(TY)}, \frac{ST_i(TY)}{n}\right]\right\}\right. \\ & \left. + \min_j \left[\begin{cases} \max_{n \in SC_a(S_{a\bullet}^j pt)} \left\{\min\left[\frac{n}{ST_i(S_{a\bullet}^j pt)}, \frac{ST_i(S_{a\bullet}^j pt)}{n}\right]\right\}, & ST_i(S_{a\bullet}^j pt) > 0 \\ 1, & ST_i(S_{a\bullet}^j pt) = 0 \end{cases}\right]\right]\end{aligned} \tag{16}$$

where $S_{a\bullet}^j pt \subseteq TY$.

For each potential template T_i we calculate the number of objects requested by the template from the perspective of subset $X \subseteq TY$ in Equation (16) as

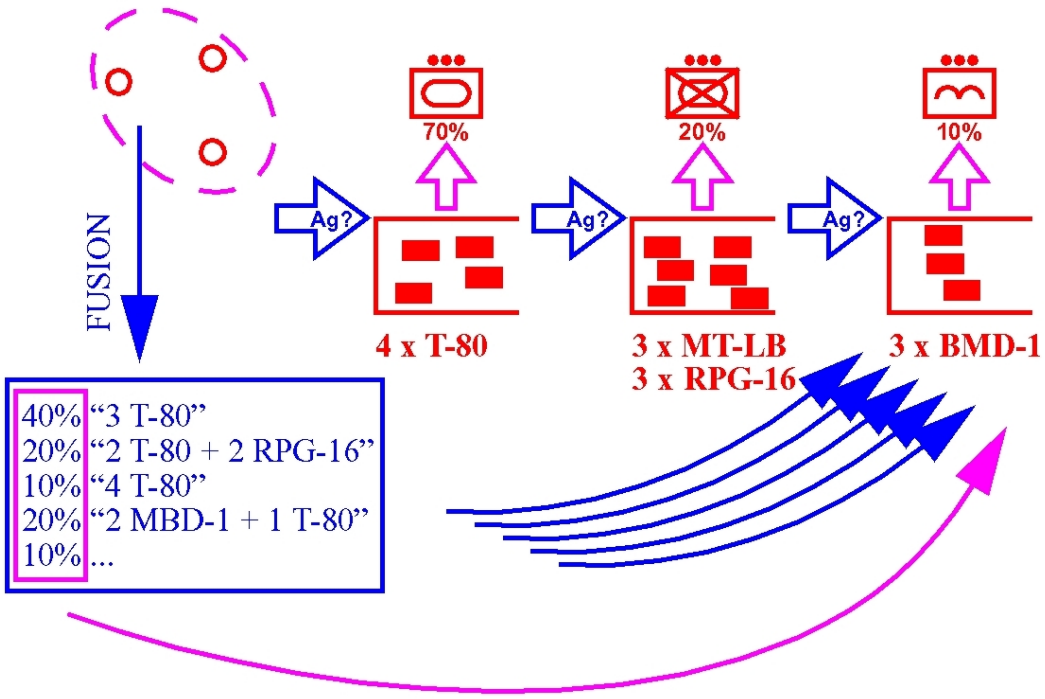

Figure 5: The intelligence is fused into several alternative hypotheses. Each hypotheses is evaluated against all templates to give an overall fitness for each template.

$$ST_i(X) = \sum_{j|S^j_{i\bullet}pt \subseteq X_\bullet pt} S^j_{i\bullet}n, \forall X \subseteq TY \tag{17}$$

and the number of objects supported by proposition $\langle x_1, x_2, \ldots, x_{|I_a|}\rangle$ of the intelligence from the perspective of subset $X \subseteq TY$ as

$$SC_a(X|\langle x_1, x_2, \ldots, x_{|I_a|}\rangle) = \sum_{i \;\middle|\; \substack{x_i \in \langle x_1, x_2, \ldots, x_{|I_a|}\rangle \\ x_{i\bullet}pt \subseteq X_\bullet pt}} x_{i\bullet}n, \forall X \subseteq TY \tag{18}$$

The best template is selected if its matching value is above some threshold, figure 5.

While the fitness measure $\pi_{\oplus J_a}(\cdot)$ is used for aggregation from the current hierarchical level, we also need the basic probability of the highest ranked template for any further aggregation from the next hierarchical level. Through a fitness weighted transformation, these templates will share this support in relation to their fitness towards the corresponding focal element in the intelligence.

We find the basic probability number of a template T_i as

$$m_{\oplus J_a}(T_i) = \sum_{\langle x_1, x_2, \ldots, x_{|I_a|}\rangle \supseteq T_i} \left[m_{\oplus J_a}(\langle x_1, x_2, \ldots, x_{|I_a|}\rangle) \frac{\pi_{\langle x_1, x_2, \ldots, x_{|I_a|}\rangle}(T_i)}{\sum_{\langle x_1, x_2, \ldots, x_{|I_a|}\rangle \supseteq T_j} \pi_{\langle x_1, x_2, \ldots, x_{|I_a|}\rangle}(T_j)} \right] \tag{19}$$

The evidential force aggregation method makes it possible to aggregate uncertain intelligence reports with multiple uncertain and nonspecific propositions into recognized forces using templates.

All templates used in IFD03 are completely specific. This allows us to optimize the implementation of Equations (16) and (17).

2.2 Tracking

In the tracking module, the states of an unknown number of ground vehicles moving in terrain are maintained. The tracking is based on observations in the form of intelligence reports of ground position $\mathbf{y}$, ground speed v, and direction of motion θ.

When tracking multiple targets in general, the size of the state-space for the joint distribution over target states grows exponentially with the number of targets. When the number of targets is large, this makes it impossible in practice to maintain the joint distribution over target states. However, if the targets can be assumed to move independently, the joint distribution does not have to be maintained.

A mathematically principled approach is to propagate only the first moment of the joint distribution, the *probability hypothesis density (PHD)* (Mahler and Zajic, 2001). This entity is briefly described in Section 2.2.1. It has the property that for each sub-area S in the state-space, the integral of the PHD over S is the expected number of targets within this area. Thus, peaks in the PHD can be regarded as estimated target states. Since the identities of objects are not maintained, there is no model-data association problem.

This paper makes three contributions to applied research on PHD filtering.

1. In Section 2.2.2, a particle filter (Gordon et al., 1993; Isard and Blake, 1998) implementation of PHD tracking, the *PHD particle filter*, is briefly described. For a thorough description, see (Sidenbladh, 2003).

2. Particle filtering is suited for tracking with non-linear and non-Gaussian motion models, and is thus suitable for ground target tracking. The non-linear terrain-dependent motion model is described in Section 2.2.3.

3. In its original formulation, the sensor visibility is assumed constant with respect to position and time. We incorporate knowledge of sensor quality and field of view into the filter. This is described in Section 2.2.4.

2.2.1 PHD filtering

The number of vehicles (called targets below) to track is unknown and varies over time. This means that the targets at time t is a *random set* (Goodman et al., 1997; Mahler, 2000) $\Gamma_t = \{\mathbf{X}_t^1, \ldots, \mathbf{X}_t^{N_t}\}$, where $\mathbf{X}_t^i$ is the state vector of target i and N_t is the number of targets in the set. A certain outcome of the random set Γ_t is denoted $X_t = \{\mathbf{x}_t^1, \ldots, \mathbf{x}_t^{n_t}\}$. Similarly, the set of observations received at time t is a random set $\Sigma_t = \{\mathbf{Z}_t^1, \ldots, \mathbf{Z}_t^{M_t}\}$, where M_t can be larger than, the same as, or smaller than N_t. A certain outcome of the random set Σ_t is denoted $Z_t = \{\mathbf{z}_t^1, \ldots, \mathbf{z}_t^{m_t}\}$.

For a large number of targets, it will be computationally intractable to keep track of every single target. However, if the signal to noise ratio (SNR) is high and the targets move independently

of each other, the full joint distribution $f_{\Gamma_t \mid \Sigma_{1:t}}(X_t \mid Z_{1:t})$ over all targets can in each time step be approximately recovered from the first moment of this distribution, the probability hypothesis density (PHD) $D_{\mathbf{X}_t \mid \Sigma_{1:t}}(\mathbf{x}_t \mid Z_{1:t})$ (Mahler and Zajic, 2001; Sidenbladh, 2003), which is defined over the state-space Θ of one target instead of the much larger joint target space Θ^{N_t}. Thus, the computational cost of propagating the PHD over time is much lower than propagating the full distribution.

The PHD has the properties that, for any subset $S \subseteq \Theta$, the integral of the PHD over S is the expected number of targets in S at time t:

$$E[|\Gamma_t \cap S|] = \int_S D_{\mathbf{X}_t \mid \Sigma_{1:t}}(\mathbf{x}_t \mid Z_{1:t}) \, d\mathbf{x}_t \, . \tag{20}$$

In other words, it will have local maxima approximately at the locations of the targets. The integral of the PHD over Θ is the expected number of targets, n_t.

We now describe one time-step in the PHD filter, which is propagated using Bayes' rule (Mahler and Zajic, 2001; Sidenbladh, 2003). First, a *prior* PHD is predicted from the PHD and observations at the previous time-step. Then, new observations are used to compute the *likelihood* of this prior PHD. This results in a new *posterior* PHD. The steps are described below.

Prediction. The temporal model of the targets include birth (appearance of a target in the field of view), death (disappearance of a target from the field of view) and temporal propagation. Probability of target death is p_D and of target birth p_B.

Target hypotheses are propagated from earlier hypotheses according to the dynamical model

$$\mathbf{X}_t = \phi(\mathbf{X}_{t-1}, \mathbf{W}_t) \tag{21}$$

where $\mathbf{W}_t$ is a noise term independent of $\mathbf{X}_{t-1}$ (Section 2.2.3). This gives

$$f_{\mathbf{X}_t \mid \mathbf{X}_{t-1}, \mathbf{Z}_{1:t-1}}(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}) \equiv f_{\mathbf{X}_t \mid \mathbf{X}_{t-1}}(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

with no dependence on the history of observations $\mathbf{z}_{1:t-1}$.

Other target hypotheses are born from observations at the previous time instant (Sidenbladh, 2003) according to the model

$$\mathbf{X}_t = \phi(h^{-1}_{\mathbf{X}_t}(\mathbf{Z}_{t-1}, \mathbf{V}_{t-1}), \mathbf{W}_t) \tag{22}$$

where $\mathbf{V}_t$ is a noise term (Section 2.2.3). This model defines the birth pdf $f_{\mathbf{X}_t \mid \mathbf{Z}_{t-1}}(\mathbf{x}_t \mid \mathbf{z}_{t-1})$. To take all observations $\Sigma_t = \{\mathbf{Z}_t^1, \ldots, \mathbf{Z}_t^{M_t}\}$ into account for target birth, a birth PHD is defined from the set of birth pdf:s as

$$D_{\mathbf{X}_t \mid \Sigma_{t-1}}(\mathbf{x}_t \mid Z_{t-1}) = \sum_{\mathbf{z}_{t-1}^i \in Z_{t-1}} f_{\mathbf{X}_t \mid \mathbf{Z}_{t-1}}(\mathbf{x}_t \mid \mathbf{z}_{t-1}^i) \, . \tag{23}$$

Given the models of motion, death and birth, the prior PHD (Mahler and Zajic, 2001) is estimated from the posterior PHD at the previous time instant as

$$\begin{aligned} &D_{\mathbf{X}_t \mid \Sigma_{1:t-1}}(\mathbf{x}_t \mid Z_{1:t-1}) = \\ &p_B D_{\mathbf{X}_t \mid \Sigma_{t-1}}(\mathbf{x}_t \mid Z_{t-1}) + \int (1 - p_D) f_{\mathbf{X}_t \mid \mathbf{X}_{t-1}}(\mathbf{x}_t \mid \mathbf{x}_{t-1}) D_{\mathbf{X}_{t-1} \mid \Sigma_{1:t-1}}(\mathbf{x}_{t-1} \mid Z_{1:t-1}) \, d\mathbf{x}_{t-1} \, . \end{aligned} \tag{24}$$

Observation. We define p_{FN} as the probability that a target is *not* observed at a given time step (the probability of false negative). This entity is further discussed in Section 2.2.4. Assuming that there are no spurious observations (a good approximation in our application), the posterior PHD distribution is computed (Mahler and Zajic, 2001) from the prior as

$$D_{\mathbf{X}_t \mid \Sigma_{1:t}}(\mathbf{x}_t \mid Z_{1:t}) = \sum_{\mathbf{z}_t^i \in Z_t} f_{\mathbf{X}_t \mid \mathbf{Z}_t, \Sigma_{1:t-1}}(\mathbf{x}_t \mid \mathbf{z}_t^i, Z_{1:t-1}) + p_{FN} D_{\mathbf{X}_t \mid \Sigma_{1:t-1}}(\mathbf{x}_t \mid Z_{1:t-1}) \tag{25}$$

where

$$f_{\mathbf{X}_t \mid \mathbf{Z}_t, \Sigma_{1:t-1}}(\mathbf{x}_t \mid \mathbf{z}_t^i, Z_{1:t-1}) \propto f_{\mathbf{Z}_t \mid \mathbf{X}_t}(\mathbf{z}_t^i \mid \mathbf{x}_t)\, D_{\mathbf{X}_t \mid \Sigma_{1:t-1}}(\mathbf{x}_t \mid Z_{1:t-1})\,, \tag{26}$$

which is a pdf (with the integral 1 over the state-space).

Using Equations (23), (24) and (25), the PHD can be propagated in time. The result of the tracking is the estimated number of targets, and the location of the detected maxima in the posterior PHD in each time step.

2.2.2 Particle implementation

We will now describe the particle filter implementation of Equations (23), (24) and (25).

A pdf (with integral 1) is usually represented with $\mathcal{N}$ particles. Here, a PHD (with integral n_t) is represented with $n_t\mathcal{N}$ particles, n_t being the expected number of targets at time t. One time-step proceeds as follows (Figure 6):

Prediction. The posterior PHD at time $t-1$ is represented by a set of particles $\{\boldsymbol{\xi}_{t-1}^1, \ldots, \boldsymbol{\xi}_{t-1}^{n_{t-1}\mathcal{N}}\}$. These are propagated in time by sampling from the dynamical model $f_{\mathbf{X}_t \mid \mathbf{X}_{t-1}}(\mathbf{x}_t \mid \boldsymbol{\xi}_{t-1}^j)$ for $j = 1, \ldots, n_{t-1}\mathcal{N}$. The propagated particles are each given a weight $\varpi_t^j = (1 - p_D)/\mathcal{N}$. The set of weighted propagated particles represent the second term in Equation (24).

For each of the observations $\mathbf{z}_{t-1}^i, \in Z_{t-1}, i = 1, \ldots, m_{t-1}$, $\mathcal{N}$ particles are sampled from the birth model $f_{\mathbf{X}_t \mid \mathbf{Z}_{t-1}}(\mathbf{x}_t \mid \mathbf{z}_{t-1}^i)$ (Equation (23)). Each particle is given a weight $\varpi_t^j = p_B/\mathcal{N}$. The resulting set of weighted particles represent the first term in Equation (24).

The two weighted particle clouds are concatenated to form a set of particles with attached weights,

$$\{(\tilde{\boldsymbol{\xi}}_t^1, \varpi_t^1), \ldots, (\tilde{\boldsymbol{\xi}}_t^{(m_{t-1}+n_{t-1})\mathcal{N}}, \varpi_t^{(m_{t-1}+n_{t-1})\mathcal{N}})\}$$

that represent the approximate prior PHD (Equation (24)) at time t.

Observation. For each new observation $\mathbf{z}_t^i \in Z_t, i = 1, \ldots, m_t$, a copy i of the prior particle set is made. New weights $\pi_t^{i,j} \propto \varpi_t^j f_{\mathbf{Z}_t \mid \mathbf{X}_t}(\mathbf{z}_t^i \mid \tilde{\boldsymbol{\xi}}_t^j)$ are computed. For each set i, the weights are thereafter normalized to sum to one. The re-weighted particle set represents the i:th term $f_{\mathbf{X}_t \mid \mathbf{Z}_t, \Sigma_{1:t-1}}(\mathbf{x}_t \mid \mathbf{z}_t^i, Z_{1:t-1})$ in the sum in Equation (25).

The original prior particle set is down-weighted according to $\pi_t^{0,j} = p_{FN}\varpi_t^j$. This set now represent the last term in Equation (25).

The concatenation of these $m_t + 1$ sets,

$$\{(\tilde{\boldsymbol{\xi}}_t^1, \pi_t^1), \ldots, (\tilde{\boldsymbol{\xi}}_t^{(m_t+1)(m_{t-1}+n_{t-1})\mathcal{N}}, \pi_t^{(m_t+1)(m_{t-1}+n_{t-1})\mathcal{N}})\}$$

is a weighted representation of the posterior PHD.

```
% prediction:
for j ← 1 : n_{t−1}N
    % from previous time-step:  posterior ξ^j_{t−1}.
    sample prior ξ̃^j_t from f_{X_t | X_{t−1}}(x_t | ξ^j_{t−1}).
    compute prior weight ϖ^j_t ← (1 − p_D)/N.
end
for i ← 1 : m_{t−1}
    % from previous time-step:  observation z^i_{t−1}.
    J ← j.
    for j ← (J + 1) : (J + N)
        sample prior ξ̃^j_t from f_{X_t | Z_{t−1}}(x_t | z^i_{t−1}).
        compute prior weight ϖ^j_t ← p_B/N.
    end
end

% observation:
for j ← 1 : (m_{t−1} + n_{t−1})N
    compute likelihood π^{0,j}_t ← p_{FN} ϖ^j_t.
end
for i ← 1 : m_t
    for j ← 1 : (m_{t−1} + n_{t−1})N
        compute likelihood π̃^{i,j}_t ← ϖ^j_t f_{Z_t | X_t}(z^i_t | ξ̃^j_t).
    end
    for j ← 1 : (m_{t−1} + n_{t−1})N
        normalize likelihood π^{i,j}_t ← π̃^{i,j}_t / Σ_k π̃^{i,k}_t.
    end
end

% resampling:
expected number of targets n_t = Σ_{i=0}^{m_t} Σ_{j=1}^{(m_{t−1}+n_{t−1})N} π^{i,j}_t.
for j ← 1 : n_t N
    monte carlo sample posterior ξ^j_t from weighted set
    ⋃_{i=0}^{m_t} {(ξ̃^1_t, π^{i,1}_t), ..., (ξ̃^{(m_{t−1}+n_{t−1})N}_t, π^{i,(m_{t−1}+n_{t−1})N}_t)}.
end
```

Figure 6: Pseudo code for time-step t in a PHD particle filter.

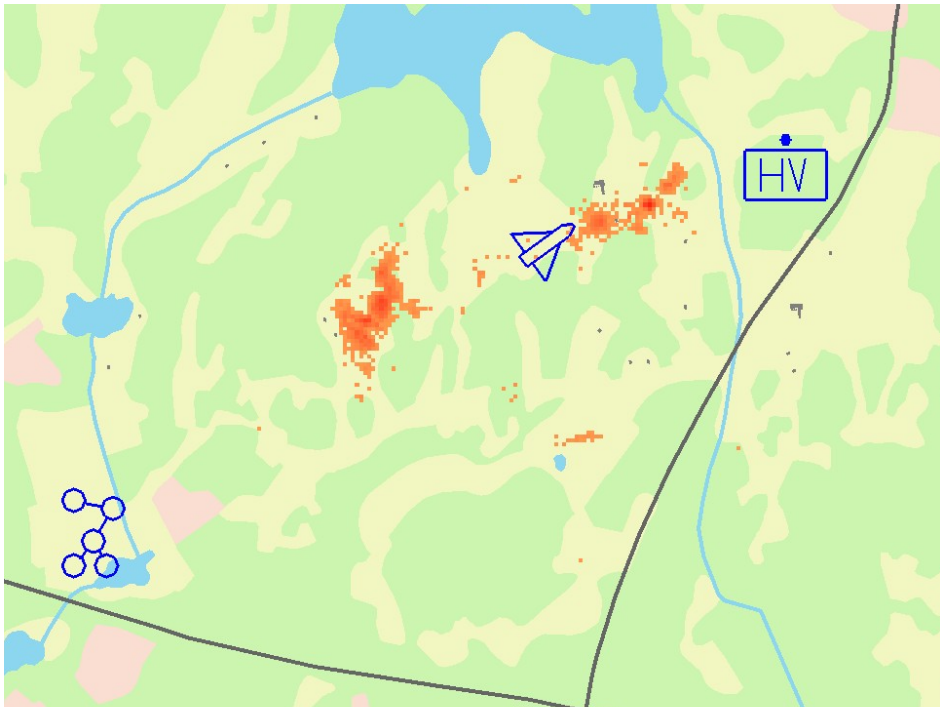

Figure 7: The posterior PHD represented as a set of particles. For greater visibility, the histogram over particle position is shown; the saturation of red in a certain sub-area (i.e., histogram bin) represents the particle concentration in this area.

Resampling. An unweighted representation of the posterior PHD is now obtained by resampling the weighted particle set. The expected number of targets is computed as the sum over all weights in this set: $n_t = \sum_{i=1}^{(m_t+1)(m_{t-1}+n_{t-1})\mathcal{N}} \pi_t^i$. Now, $n_t\mathcal{N}$ new particles are Monte Carlo sampled from the weighted set. The result is an unweighted particle set $\{\boldsymbol{\xi}_t^1, \ldots, \boldsymbol{\xi}_t^{n_t\mathcal{N}}\}$ that represents the approximate posterior PHD $D_{\mathbf{X}_t \mid \Sigma_{1:t}}(\mathbf{x}_t \mid Z_{1:t})$ at time t.

An example of a posterior PHD is shown in Figure 7.

2.2.3 Terrain Dependent Motion and Birth Model

The state of a vehicle hypothesis at time t depends according to Equation (21) on the state of the hypothesis at the previous time-step $t-1$, and of the terrain at the vehicle position $\mathbf{y}_t$. Likewise, the state of a newly born vehicle hypothesis depends according to Equation (22) on the observation from which it was born, and on the terrain at its position.

While the former dependence can be modeled using linear dynamics, the terrain dependence is highly non-linear. In this system, the terrain is only evaluated with respect to the vehicle position $\mathbf{y}_t$, not speed and direction. For each position, the terrain can be retrieved from the database (see Section 3.2). The terrain influence on the vehicle position is represented as probability ratios $\pi_{\text{water}} = p_{\text{water}}/p_{\text{road}} = 0$, $\pi_{\text{forest}} = p_{\text{forest}}/p_{\text{road}} = 0.04$, $\pi_{\text{field}} = p_{\text{field}}/p_{\text{road}} = 0.2$, and $\pi_{\text{road}} = 1$ of the vehicle being positioned in different type of terrain.

The sampling from the conditional pdf $f_{\mathbf{X}_t \mid \mathbf{X}_{t-1}}(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ is performed in two steps:

1. Each particle in the old posterior cloud $\{\boldsymbol{\xi}_{t-1}^1, \ldots, \boldsymbol{\xi}_{t-1}^{n_{t-1}\mathcal{N}}\}$ is propagated using a first order linear dynamical motion model.

2. Each new particle is given a weight $\pi_{\text{terrain type}}$ depending on the terrain type at its position. A new particle cloud is then Monte Carlo sampled from the weighted particles.

Likewise, the sampling from the conditional pdf $f_{\mathbf{X}_t \,|\, \mathbf{z}_{t-1}}(\mathbf{x}_t \,|\, \mathbf{z}_{t-1}^i)$ for each old observation $\mathbf{z}_{t-1}^i$ is performed as

1. $\mathcal{N}$ particles are sampled from observation $\mathbf{z}_{t-1}^i$ using a linear Gaussian model. The cloud is propagated using a first order linear dynamical motion model.

2. Identical to step 2. above.

2.2.4 Sensor Position Dependent Detection Rate

The probability of missed detection p_{FN} varies over space and time, due to the type and fields of view of the different sensors. To achieve a correct PHD estimate it is important to model this variance.

For each sensor i in the system, the target detection probability p_t^i and the present field of view A_t^i is known at a given time-step t (see also Section 3.1). The probability of missed detection in a certain position $\mathbf{y}$ can then be derived as

$$p_{FN}(\mathbf{y}) = \prod_{\mathbf{y} \in A_t^i} 1 - p_t^i \,. \tag{27}$$

This varying p_{FN} is used for propagation of the PHD over time as described in Equation (25).

This corresponds to intuition. If there are no sensors nearby, the prior particle distribution is accepted as posterior distribution as is. However, prior particles that come inside the field of view of a sensor are suppressed if there are no observations to support them. Accurate sensors with high p_t^i suppress particles to a higher degree than sensors with a low p_t^i.

2.3 Sensor Management

The aggregation module in IFD03 implements a simple version sensor management based on *random set simulation*. As in the tracking module (Section 2.2), random sets (Goodman et al., 1997; Mahler, 2000) are used to formally describe the operation of our algorithm, and the probability hypothesis density is used to render the method computationally feasible.

The purpose of the sensor adaptation implemented in IFD03 is to determine which of several sensor adaptation schemes should be used in a given tactical situation. Inputs to the module are a list of such sensor schemes or plans, a road network that describes the geography of the situation of interest, and positions of enemy units.

Pseudo code for our sensor adaptation algorithm is shown in Figure 8. The algorithm, which will be described in detail in future work, works as follows. We are given a density vector x_0, which describes the positions of the units of interest at time $t = 0$. We also assume a set S of sensor adaptation schemes and information on the road network on which the enemy is assumed to move.

We now introduce three different random sets:

1. $\mathbf{X}(t)$ denotes the positions of the enemy units at time t, conditioned on them being at $\mathbf{x}_0$ at time 0. It can be seen as representing a simulation of ground truth: the instance $x(t)$ of $\mathbf{X}(t)$ occurs with probability $P[\mathbf{X}(t) = x(t)|\mathbf{X}(0) = x_0]$. For simplicity of notation, we will not explicitly show the conditioning on x_0 in the following.

2. For each sensor scheme $s \in S$ and instance $x(t)$ of the future ground truth, we calculate a set of possible observations $\mathbf{Z}(x(t), s, t)$ at time t. $\mathbf{Z}$ is also a random set; note that it depends on ground truth as well as sensor scheme.

3. Finally, we determine what our view of ground truth would be, given the set of observations $\mathbf{Z}$. This gives rise to the final random set, $\mathbf{Y}(t)$. $\mathbf{Y}(t)$ is our fusion system's approximation of the (simulated) ground truth $\mathbf{X}(t)$ using the observations $\mathbf{Z}$ obtained by deploying sensors according to scheme s_i.

Note that all of the random sets introduced are explicitly time-dependent. When writing an expression like $P[\mathbf{X}(t)]$, we mean the probability of the entire time-evolution of $\mathbf{X}(t)$, not just the probability at a specified time. $P[]$ can thus be seen as a sort of "probability density functional" in the space of all explicitly time-dependent random sets. Further mathematical details on this will be presented in future work.

Determining which sensor scheme to use is now done simply by comparing the assumed ground truth $x(t)$ to the fusion system's simulated view $y(t)$. For each instance $x(t)$ of $\mathbf{X}(t)$, we can easily determine the best s by averaging over the ensembles of observations $\mathbf{Z}$ and simulated filter output $\mathbf{Y}$ entailed by that simulated ground truth. A sensor scheme is good if the simulated filter gives a good approximation of the simulated ground truth. The fit of a specific sensor scheme s for a certain simulated ground truth $x(t)$ can be written as

$$H(x(t), s) = \\ \int P[\mathbf{Z}(t) = z(t)|\mathbf{X}(t) = x(t), s] \times P[\mathbf{Y}(t) = y(t)|\mathbf{Z}(t) = z(t)] \times h(x(t), y(t))dx(t)dy(t) \tag{28}$$

where h is a functional that compares $x(t)$ and $y(t)$ and the integrals are functional integrals over all random sets $y(t)$ and $z(t)$. In IFD03, we used four different h-functionals: two that computed the entropy of y at either a user-specified target-time or averaged over all time, and two that calculated the L_2 distance between x and y, again either at a specific time or averaged over all times. The difference between the entropy-like measure and the distance measure is that the entropy measure rewards sensor schemes that give rise to peaked distributions, but might miss some of the enemy units. A measure that uses a specific time is termed a local measure, while global h-measures average over all times.

The overall best sensor scheme is then determined by averaging also over the random set $\mathbf{X}(t)$, as

$$s^{\text{best}} = \arg\min_{s \in S} \int P[\mathbf{X}(t) = x(t)]H(x(t), s)dx(t) \tag{29}$$

Implementing Equations (28) and (29) would thus entail averaging over three different random sets, which is clearly computationally infeasible. There are several possible ways of approximating these equations. One way is to use approximations of the probabilities P appearing in them, perhaps doing some sort of Monte Carlo sampling instead of the ensemble averages. In the implementation used in IFD03, we use a number of approximations:

```
% Pseudo code for the four major steps of the adaptation algorithm.
% The module simulates N_t ground truths and does
% N_o realizations of the observation process for each
% ground truth, averaging the fitnesses.  This process is
% repeated for each sensor scheme s, and the best s
% is selected.
% Note that x_t, y_t and z_t here are vectors;
% we have discretised space to only include nodes that
% are present on the road network.  T and δ below
% represent the motion model constrained to this network.
% T is the end time of simulation, while p_d is the
% assumed detection probability of a sensor.

% Simulating ground truth:
x_1 = x_0
for t = 1 : T − 1
    x_{t+1} = x_t + δ where δ is randomly selected
end for

% Generate fictitious observations:
for t = 1 : T
    if s has sensor with view of x_t at time t
       generate observation z_t = x_t with probability p_d
    end if
end for

% Simulate filter:
y_1 = x_1
for t = 1 : T − 1
    y_{t+1} = T y_t + z_{t+1}
    where T is a transfer matrix corresponding to the road network
end

% Compare simulated ground truth and filter:
h_1(s) = ||y_T − x_T||_2
h_2(s) = Σ_t ||y_t − x_t||_2
h_3(s) = H(y_T)
h_4(s) = Σ_t H(y_t)
where H(x) is the entropy of the vector x
```

Figure 8: Pseudo code for sensor adaptation.

1. As stated above, we constrain all motion of adversary units to a road network. We also use discretised time instead of continuous.

2. Instead of full random sets for simulated ground truth, observations, and simulated filter, we use PHD's for these. This means that, for instance, $x(t)$ only gives the expected number of units at different positions in the road network.

3. We use a very simple model for determining $P[\mathbf{X}(t) = x(t)]$ and averaging over all $x(t)$: we assume that enemy movement can be described by a motion model T. This model is used to determine paths for all enemy units present at time $t = 0$. Instead of averaging over all possible futures, a certain number N_f of such paths are generated and assumed to have equal probabilities of occurring.

4. A similar motion model in the form of a transition matrix T is used to simulate the filter determining $\mathbf{Y}$, and we average only over a number N_o of possible observations (*i.e.*, realizations of $\mathbf{Z}$).

The adaptation module returns the best found sensor scheme s as well as a quality measure that simply gives the fractions of the number of simulated ground truths and observations in which s dominated all other schemes.

The adaptation module is the least mature module in IFD03, and we see a number of possible improvements to both the algorithm and the implementation.

3 System Description

In this section we describe the general design of the IFD03. The demonstrator utilizes all of the methods described in Section 2, and also makes use of an advanced terrain database (Section 3.2) that has been integrated into the simulation framework Flames (Section 3.1). An overview of how the different components fit together is shown in Figure 9.

The connections between the parts can be summarized as follows.

All data originate in the scenario simulator. For IFD03, we chose to use the Flames system (see Section 3.1). For the demonstrations performed, we have used a standard scenario from the Swedish armed forces.

The parts of IFD03 that handle the scenario simulation are written in C and directly linked into the Flames suite of programs. We also implemented a number of sensor models to make observations on the red team.

The various fusion modules impose different requirements on what should be in an observation report. The format chosen for our reports is described in Section 3.4. Note that the aggregation module also generates reports of vehicles, platoons, companies, and battalions — these follow basically the same format as those for observations, with obvious redefinitions.

All of our fusion modules are implemented in compiled Matlab code, which is linked into the Flames program `Fire` to produce an executable.

When a sensor model has made an observation, it communicates this to the Fusion Node component of IFD03, which is implemented in C and Matlab. The Fusion Node, upon receiving an observation, stores this in an internal format (detailed in Section 3.4). Detailed analyses are not

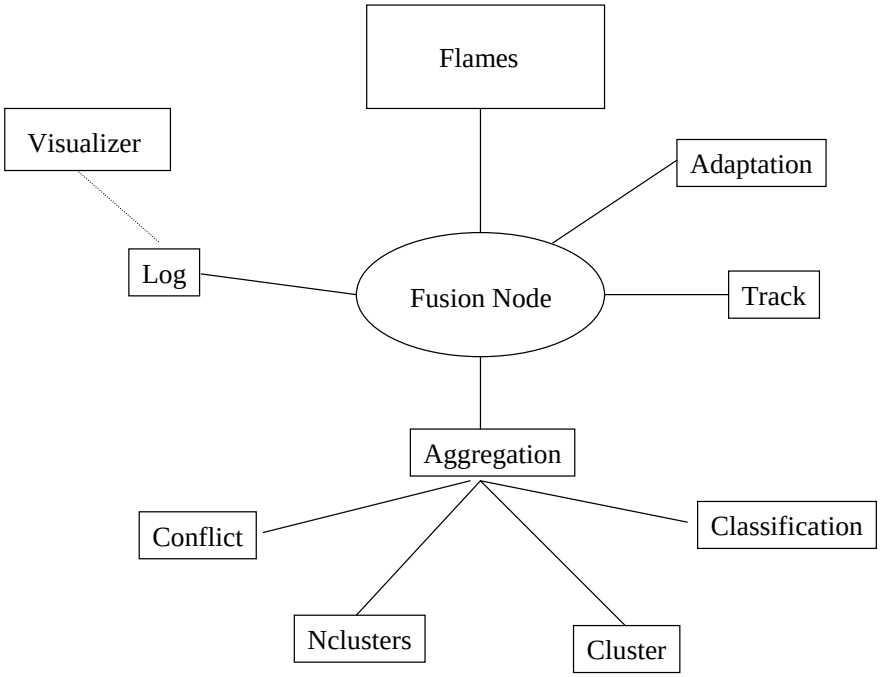

Figure 9: The connections between the different parts of the IFD03. Lines between modules mean that the modules exchange data. Note that the Visualizer is a separate program, while all the other modules are linked into the `Fire` program.

performed for each observation that is logged; instead different analysis modules are called at different pre-specified times.

The three fusion modules implemented in IFD03 have somewhat different requirements on the data supplied to them. The particle filter implemented in the Track module needs to get observations fairly regularly, while the Aggregation module can be called at greater time-intervals. The sensor Adaptation module is currently only used once in the scenario — this occurs late in the scenario when most of the blue force's mobile sensor resources have been destroyed and they need an accurate estimation of the future position of an enhanced tank company.

If the Fusion Node determines that a certain fusion module should be called, it collects the appropriate input data for the module and then takes care of its output. The output is logged to data files for later playback in the visualizer, see Section 4.

For the Track module, the appropriate input is a list of observations that have occurred since the last time Track was called. Its output consists of histograms of the particles representing the PHD of the hostile units. The Track module could also be used with aggregated platoons or companies as inputs; this would enable us to follow units instead of vehicles.

Input to Adaptation (Section 2.3) consists of a list of positions of enemy units, as well as a set of possible sensor schemes and the road network used in the random set simulation.

The Aggregation module utilizes all observations collected so far, and attempts to build an hierarchical situation picture of the opposing units. Its first step consists of calculating a conflict matrix (Section 2.1.1) and then clustering the observations (Section 2.1.2). The number of clusters to use is unknown, so a special procedure is used to determine this, see Section 2.1.3. Once the

proper clusters are found, a classification procedure that compares the cluster to a pre-specified set of templates is performed, Section 2.1.4. The output of the aggregation module consists of a list of clusters and their classification in terms of templates.

This output forms the input for the next level of aggregation: classified clusters of sensor observations are termed vehicles, and further analyzed to get platoons, clusters of which in turn give rise to companies.

Each iteration of the aggregation module thus produces lists of vehicles, platoons, and companies. For details on how to connect units observed at one time-step with those observed at another, see Section 3.3.

3.1 Simulator

We chose to base the simulator used for IFD03 on the commercial simulation framework *Flames*, developed by Ternion Corporation. The reasons for this choice are discussed in detail in (Hörling et al., 2002). Flames offers an infrastructure that includes common facilities for models, *e.g.*, object management, time management, memory management, execution control, data base management, and so on. It also provides a set of standard applications to support scenario definition and execution.

The primary objects of the simulation fall into three categories: actors, terrain model and Fusion Node.

- Actors. The actors in a scenario simulation consist of units from two teams, the "red" team and the "blue" team. Each unit is equipped with a platform model and a radio model for within-team communication. Additionally, units in the blue team are equipped with various sensors for target detection and classification. Sensor types modeled are a video and IR camera, a "soldier" sensor (visual), a ground target multi-sensor system (acoustic/seismic) and a communication intelligence surveillance system. The video/IR camera can be attached to an unmanned aerial vehicle (UAV), which can also carry and drop a ground target multi-sensor system. To enable target detection and classification, visual and acoustic/seismic signatures are attached to the platforms of the red team. When a detection is made a report is sent to the Fusion Node with target information. The format of the reports is described in Section 3.4.

- Terrain. As further detailed in Section 3.2, the terrain model is structured as a triangulated terrain skin with additional vector data, describing the features of the environment as polygons. The platforms and sensors in a scenario use this information for movability and visibility calculations. The same information is also available for the Fusion Node, which currently uses it only in the tracking algorithm. In future versions, the terrain model will possibly also be used for path-planning in the sensor management method.

- Fusion Node. In a strict meaning, the Fusion Node is also an actor in the scenario. It is implemented as a Flames cognitive model attached to an immobile blue unit. The other blue units constantly feed the Fusion Node with target reports and upon request, sensor status reports. By sending sensor status reports a unit can update the Fusion Node on the current coverage of its sensors. This information is used by the tracking algorithm, see Section 2.2.4.

The Fusion Node acts in the scenario by making sensor status requests on suitable units and by directing blue sensor resources.

The scenario demonstrated in IFD03 takes place in May 2015. Tension in the Baltic Sea area has grown gradually over several years and the state of alert of the Swedish defense has been raised. At the outbreak of the war a number of concurrent events occur. One of these is a ”Trojan horse” enemy landing at the ferry harbor at Kapellskär. A mechanized enemy battalion (the red team) moves inland. The defending battalion commander wants to obtain a detailed picture of the enemy's size, composition, and activity in order to be able to judge the enemy's action options and decide his own. The intelligence sources available at the time of the landing (the blue team) are five Home Guard patrols, three UAV:s, a traffic surveillance camera, a communication intelligence surveillance system and three ground target multi-sensor systems.

3.2 Terrain Model

When planning IFD03 in 2001, the requirements for geographic information system (GIS) services and the ability to use Swedish and European geodata formats was not satisfied by Flames. Ternion Corporation proposed an augmentation of the capabilities of Flames which led to FOI placing an order with Ternion for Flames terrain modeling enhancements. In 2003, Flames was equipped with an advanced terrain model that offers user access for queries of altitude, line-of-sight and terrain feature data. The model data is provided by a third party terrain database generation tool, Terra Vista by Terrex Inc., that can import data from different sources in different formats and export a single correlated terrain description. Terra Vista also has the ability to write the correlated data in a variety of formats. Flames uses ARC Shape files as input to its terrain model. The model is structured as a TIN DEM (Triangulated Irregular Network Digital Elevation Model) representing the terrain skin, and additional vector data describing terrain features, such as roads, rivers, lakes and houses. For the scenario used in IFD03, data come from the Swedish Land Survey and describe a 45 x 20 km^2 area over the peninsula of Rådmansö, north-west of Stockholm.

3.3 Data management

All reports that are generated are saved in the global variable *all_reports*. Some sensors are tracking sensors. Those sensors generates many reports about the tracked vehicle. This is useful for the particle filtering methods but may cause the clustering algorithm to overflow. Therefore the sensor sets a *tracking flag* when it generates a consecutive report. All reports that do not have the tracking flag set, are saved in *all_reports_to_cluster*.

For computational reasons we can not use all reports in *all_reports_to_cluster* when we cluster them into vehicles. Instead we consider only the last *max_number_to_cluster* reports. This means that we could lose vehicles when they stop generating reports. To prevent this we save the vehicle observations in a *vehicle record*.

At startup time the vehicle record is empty.

When we have clustered reports into new vehicles, the vehicle observations in the vehicle record are removed and the new vehicle observations are put in the vehicle record. Then the old vehicles – that were removed from the vehicle record – are put back in the vehicle record if they are not too old and if they consist entirely of reports that were not considered in the last clustering.

3.4 Report format

The report format, as shown in Figure 3.4, is the Matlab structure that defines the reports sent from Flames to the Fusion Node.

Position
A 3-valued vector [latitude, longitude, height (meter)]
TARGET_POS_ERR_TYPE_FLAG
1 = no position error.
2 = circle, standard deviation in target_pos_err
3 = ellipse, major and minor axis + rotation angle in target_pos_err_nswerot
4 = polygon, Nx2 matrix target_pos_err_polygon contains coordinates
Class_focals
el_i = Class_focals{i} contains the i:th set of elements the vehicle can be.
$el_i\{j\}$ contains the j:th element the vehicle can be.
The type of $el_i\{j\}$ is string.
Class_masses
Class_masses(i) is the mass of focal elements focals{i}
TARGET_SPEED_ERR_TYPE_FLAG
0 = no speed error given
1 = stddev in target_speed_err
TARGET_HEADING_ERR_TYPE_FLAG
0 = no heading error given
1 = stddev in target_heading_err

When the reports have been clustered into vehicles, the structure used for representing a vehicle is almost identical to the report format structure. In addition to the report format structure, the vehicle structure also contains a list of the reports referring to the vehicle, and the field : *target_class_explainstr (string)*. The same is true for the *Platoon structures*, the *Company structures* and the *Battalion structures*.

4 Visualization

For demonstrational purposes it is of great importance to have a flexible and functional visualization tool. The IFD03 Visualizer is a strongly modified version of the original Flames visualizer Flash aimed at illustrating the full functionality of the IFD03 Fusion Node. The simulation results can be visualized synchronously in multiple parallel visualizers, making it possible to use several computers and screens simultaneously. New views can easily be created and customized.

Two categories of views were used in IFD03:

- The Analyst views. These views could help an analyst building a situation picture. An Analyst view is a window with a 2D projection of the terrain model, with zooming and panning possibilities. On top of the map is a layer of Open GL graphics and Flames icons, representing sensor reports, unit positions or particle filter histograms. For IFD03 the following views were designed and demonstrated: ground truth, sensor reports, vehicle, platoon and

```
report_format = struct(
   'report_id',               ID
                              (number, set by Flames_Report),

   'sensor_id',               ID (number),
   'sensor_pos',              Position,
   'sensor_type',             TYPE (string),
   'sensor_continuous_tracking',
                              REPORT_NOT_TO_CLUSTRING_FLAG,

   'target_class_focals',     Class_focals,
   'target_class_masses',     Class_masses,

   'target_pos',              Position,
   'target_pos_err_type',     TARGET_POS_ERR_TYPE_FLAG,
   'target_pos_err',          standard_deviation (meter),
   'target_pos_err_nswerot',  [a, b, θ],
   'target_pos_err_polygon,   <Nx2 matrix>,

   'target_heading',          θ (−π <= θ <= π, 0 is north),
   'target_heading_err_type', TARGET_HEADING_ERR_TYPE_FLAG,
   'target_heading_err',      Δθ (radians),

   'target_speed',            speed (m/s),
   'target_speed_err_type',   TARGET_SPEED_ERR_TYPE_FLAG,
   'target_speed_err',        Δs (m/s),

   'target_correct_name',     NAME
                              (string, only for debugging
                              and evaluation),

   'time_detected',           detection_time (string),
   'time_detected_num',       dtime (set by MatLab),
   'time_received',           received_time (string),
   'time_received_num',       rtime (set by MatLab),
)
```

Figure 10: The report format

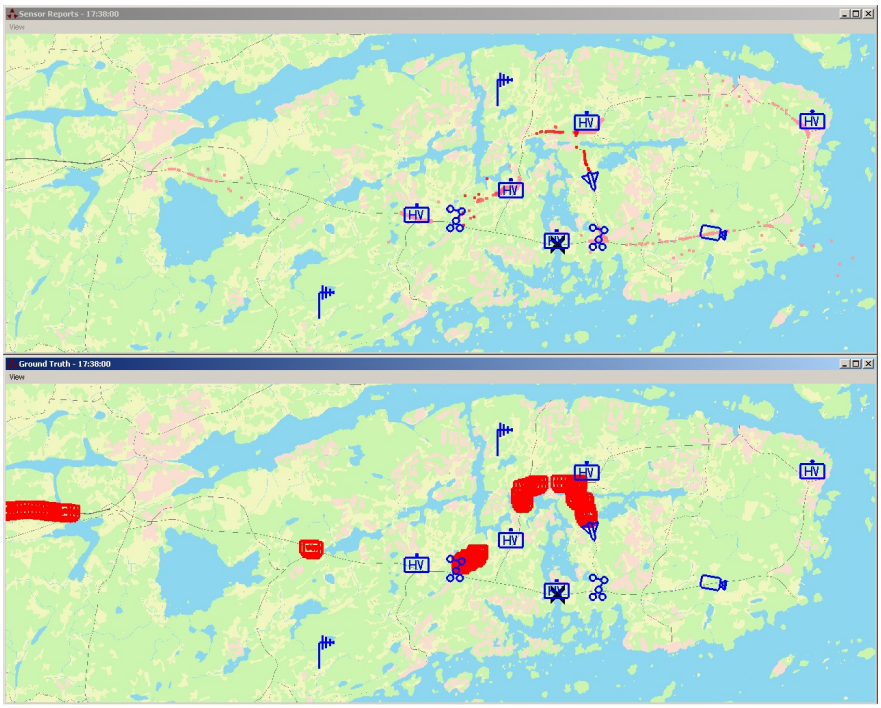

Figure 11: Snapshot of the ground truth and sensor report view in IFD03.

company aggregates and particle filter histograms on vehicle level. Snapshots of the ground truth and sensor report views are shown in Figure 11, and vehicle and platoon aggregate views in Figure 12.

Additional information for the analyst can be obtained by clicking on the symbols in a view. An information box then appears with details specific for each type of symbol. As an example, clicking on a platoon symbol gives information on the age of the aggregate, the classification of it and its certainty, the number of vehicles in the cluster forming the platoon and the best alternative classification.

- The Status views. These views display technical data from the analysis methods. Two Status views were designed for the IFD03, using basic Matlab plotting tools. The first view consists of four simple sub-plots (Figure 13). One shows the stream of incoming reports as a floating table with information on the most recent reports. The other three sub-plots show, respectively, the estimated number of vehicles, platoons and companies over time. In the case of estimated number of vehicles, two graphs are shown, representing the estimate given by the aggregation module and by the tracking module respectively. For higher aggregates no information is currently provided from the tracking, but can of course be added when tracking is performed on all levels.

  The second status view is used for demonstrating the sensor management. The alternative paths available for the sensor platform are shown and the final choice of the sensor management method is highlighted.

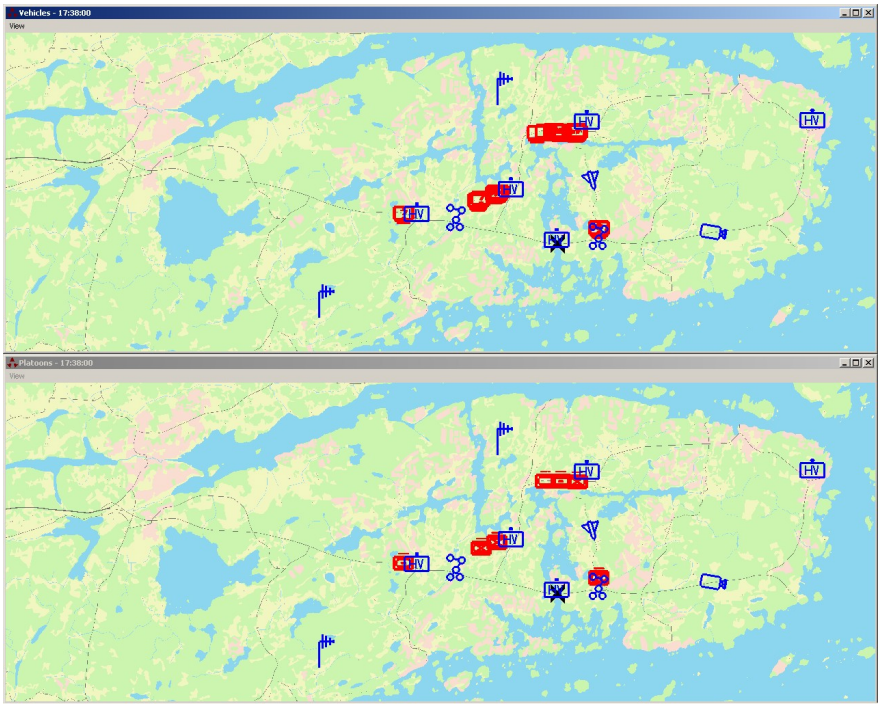

Figure 12: Snapshot of the vehicle and platoon aggregate view in IFD03.

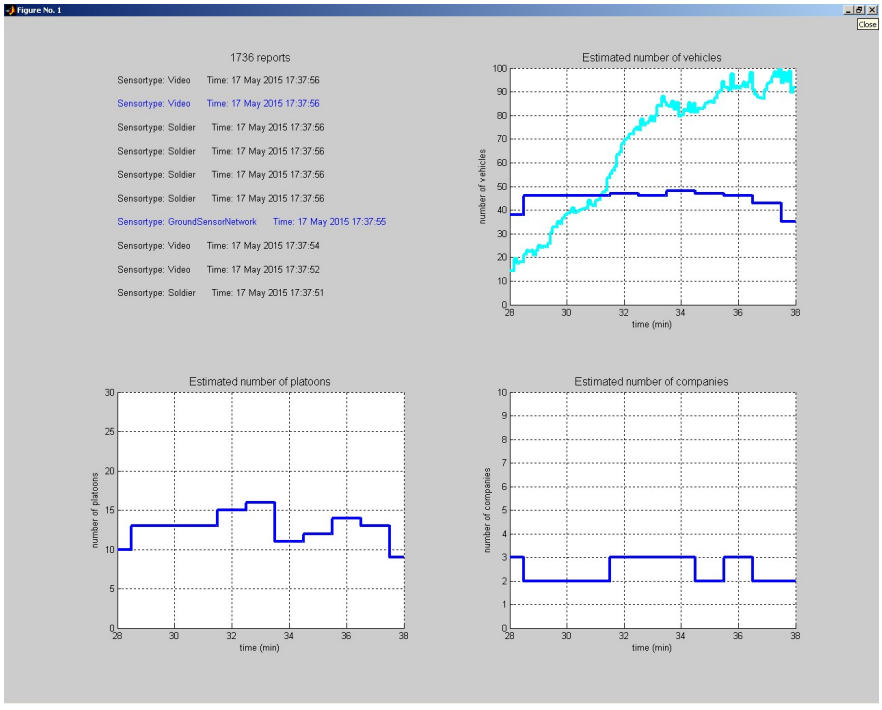

Figure 13: Snapshot of the standard status view in IFD03. The sub-plots show incoming reports and the estimated number of vehicles, platoons and companies over time.

Technically the IFD03 Visualizer consists of four entities, out of which three are executable applications. The database, handled by a MySQL database manager, stores simulation result data to be visualized. The Postprocessor application is responsible for creating tables in the database and for converting and transferring simulation result data into the database. The Playback Control application is responsible for synchronizing the playback of the scenario across the different connected visualizers. The user controls the playback of the scenario from this application. The user can move freely in scenario time and the clients will be updated accordingly. The application works as a server to which the visualizers, clients, can connect. The modified `Flash` application is responsible for the actual visualizing of the data. The different kinds of data are visualized in views and `Flash` connects to the playback control as a client and fetches the data to be visualized from the database.

5 Conclusions

The Swedish Defence Research Agency has developed a demonstrator IFD03 for demonstrating information fusion methodology focused on intelligence processing at the division level. With this demonstrator we are able to demonstrate possible methods for a future Network Based Defence (NBF) / Network Centric Warfare (NCW) C4ISR system. The demonstrator also functions as an internal development tool for testing newly developed methods in an established environment together with previously developed methods.

A demonstration of IFD03 in December 2003 for the Swedish Armed Forces was a great success.

References

Ahlberg, S., Hörling, P., Jöred, K., Mårtenson, C., Neider, G., Schubert, J., Sidenbladh, H., Svenson, P., Svensson, P., Undén, K. and Walter, J. (2004). The IFD03 information fusion demonstrator, *Proceedings of the Seventh International Conference on Information Fusion*, International Society of Information Fusion, Mountain View, CA, USA, pp. 936–943.

Bengtsson, M. and Schubert, J. (2001). Dempster-Shafer clustering using Potts spin mean field theory, *Soft Computing* **5**(3): 215–228.

Biermann, J. (1998). HADES – A knowledge-based system for message interpretation and situation determination, *in* A. P. del Pobil, J. Mira and M. Ali (eds), *Tasks and methods in Applied Artificial Intelligence, Proceedings of the Eleventh International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, LNCS 1416, Springer-Verlag, Berlin, Germany, pp. 707–716.

Cantwell, J., Schubert, J. and Walter, J. (2001). Conflict-based force aggregation, *Cd Proceedings of the Sixth International Command and Control Research and Technology Symposium*, US Dept. of Defence CCRP, Washington, DC, USA, Track 7, Paper 031, pp. 1–15.

Chaikin, P. and Lubensky, T. C. (1995). *Principles of condensed matter physics*, Cambridge University Press, Cambridge, UK.

Dempster, A. P. (1968). A generalization of Bayesian inference, *Journal of the Royal Statistical Society B* **30**(2): 205–247.

Goodman, I. R., Mahler, R. P. S. and Nguyen, H. T. (1997). *Mathematics of Data Fusion*, Kluwer Academic Publishers, Dordrecht, Netherlands.

Gordon, N., Salmond, D. and Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *IEE Proceedings F (Radar and Signal Processing)* **140**(2): 107–113.

Hörling, P., Mojtahed, V., Svensson, P. and Spearing, B. (2002). Adapting a commercial simulation framework to the needs of information fusion research, *Proceedings of the Fifth International Conference on Information Fusion*, International Society of Information Fusion, Sunnyvale, CA, USA, pp. 220–227.

Isard, M. and Blake, A. (1998). Condensation – conditional density propagation for visual tracking, *International Journal of Computer Vision* **29**(1): 5–28.

Johnson, J. K. and Chaney, R. D. (1999). Recursive composition inference for force aggregation, *Proceedings of the Second International Conference on Information Fusion*, International Society of Information Fusion, Mountain View, CA, USA, pp. 1187–1195.

Lorenz, F. P. and Biermann, J. (2002). Knowledge-based fusion of formets: discussion of an example, *Proceedings of the Fifth International Conference on Information Fusion*, International Society of Information Fusion, Sunnyvale, CA, USA, pp. 374–379.

Mahler, R. (2000). *An Introduction to Multisource-Multitarget Statistics and its Applications*, Lockheed Martin Technical Monograph.

Mahler, R. and Zajic, T. (2001). Multitarget filtering using a multitarget first-order moment statistic, SPIE Vol. 4380 *Signal Processing, Sensor Fusion and Target Recognition X*, SPIE, Bellingham, WA, USA, pp. 184–195.

Peterson, C. and Söderberg, B. (1989). A new method for mapping optimization problems onto neural networks, *International Journal of Neural Systems* **1**(1): 3–22.

Potts, R. B. (1952). Some generalized order-disorder transformations, *Proceedings of the Cambridge Philosophical Society* **48**: 106–109.

Schubert, J. (1993). On nonspecific evidence, *International Journal of Intelligent System* **8**(6): 711–725.

Schubert, J. (2000). Managing inconsistent intelligence, *Proceedings of the Third International Conference on Information Fusion*, International Society of Information Fusion, Sunnyvale, CA, USA, pp. TuB4/10–16.

Schubert, J. (2003a). Clustering belief functions based on attracting and conflicting metalevel evidence, *in* B. Bouchon-Meunier, L. Foulloy and R. R. Yager (eds), *Intelligent Systems for Information Processing: From Representation to Applications*, Elsevier Science, Amsterdam, Netherlands, pp. 349–360.

Schubert, J. (2003b). Evidential force aggregation, *Proceedings of the Sixth International Conference on Information Fusion*, International Society of Information Fusion, Sunnyvale, CA, USA, pp. 1223–1229.

Schubert, J. (2004). Clustering belief functions based on attracting and conflicting metalevel evidence using potts spin mean field theory, *Information Fusion*, in press .

Shafer, G. (1976). *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ, USA.

Sidenbladh, H. (2003). Multi-target particle filtering for the probability hypothesis density, *Proceedings of the Sixth International Conference on Information Fusion*, International Society of Information Fusion, Sunnyvale, CA, USA, pp. 800–806.

Svensson, P. and Hörling, P. (2003). Building an information fusion demonstrator, *Proceedings of the Sixth International Conference on Information Fusion*, International Society of Information Fusion, Sunnyvale, CA, USA, pp. 1316–1323.