# Decentralized Data Fusion and Control in Active Sensor Networks

**Alexei Makarenko, Hugh Durrant-Whyte**
ARC Centre of Excellence in Autonomous Systems (CAS)
The University of Sydney, Australia
www.cas.edu.au
{alexei,hugh}@cas.edu.au

**Abstract –** *The paper presents an algorithm for Bayesian decentralized data fusion (BDDF) and its extension to information-theoretic control. The algorithm is stated for a feature represented by a general probability density function. Several specific representations are then considered – Gaussian, discrete, Certainty Grid, and hybrid. Well known algorithms for these representations are shown to fit the general BDDF pattern. Stating the algorithms in Bayesian terms has a practical advantage of allowing a generic software implementation. It is also hoped that a clear general formulation will stimulate extensions to efficient non-parametric representations of arbitrary distributions. The algorithms are described in the context of the Active Sensor Network architecture – a modular framework for decentralized cooperative data fusion and control. The approach is illustrated with the results of two deployment scenarios with an indoor sensor network.*

**Keywords:** Decentralized data fusion, decentralized control, sensor networks, mobile robots.

## 1 Introduction

Large numbers of autonomous sensing platforms connected into a network promise better spatial coverage, higher responsiveness, survivability and robustness compared to a single vehicle solution. The need for such systems exists in many applications involving tasks in which timely fusion and delivery of heterogeneous information streams is of critical importance. Examples include military and civilian surveillance, fire fighting, intelligent buildings, etc.

The Active Sensor Network (ASN) project at the University of Sydney aims to combine decentralized data fusion and control algorithms into a unified yet flexible system architecture suitable for a wide range of sensing tasks. The ASN can be described from three viewpoints: the architecture [1], the algorithms, and the concrete implementation [2]. The focus of this paper is on the algorithmic side of the framework but the general approach is briefly described to provide the necessary background.

We seek a solution to the problem of distributed information gathering (DIG). We consider a distributed phenomenon which can be described by a state vector $x$. There is a set of heterogeneous robotic platforms equipped with sensors and actuators. There is also a set of operators who observe the phenomenon directly using human senses or by interacting with the network through a user interface (UI). All entities, human and robotic, are thought of as members of a team.

ASN is an architecture for *cooperative autonomous* sensing platforms. Autonomy implies that a platform is able to work in isolation and does not rely on infrastructure services, remote control, or other external inputs. Cooperative means that the platforms share common goals and, when possible, work together to achieve them. Platforms are likely to have different capabilities but each comes equipped with power, processing and communication facilities, sensors and actuators. Each one fuses local observations with information communicated from neighboring nodes into a synchronized view of the world. Similarly, each one makes local control decisions based on the knowledge of local platform capabilities and the global synchronized world view.

The fundamental principle of the ASN architectural style is decentralization. Compared to a centralized or a distributed system, a decentralized system is characterized by two key constraints [3]: a) no central services and facilities and b) no knowledge of global topology. The resulting system offers a number of advantages over other architectures. *Scalability:* the computational and communication load at each node is independent of the size of the network. *Robustness:* no element of the system is mission critical, so that the system is survivable in the event of run-time loss of components. *Modularity:* components can be implemented and deployed independently from each other.

The ASN system is composed of software components communicating asynchronously with each other. Component types correspond roughly to the functional breakdown. With respect to the environment information, a component can be a Source (producer), a Sink (consumer), or a Fuser/Distributor. Similarly, with respect to control commands, a component can be a Source (decision maker) or a Sink (controlled object). A particular component can play several of these roles at once. To make the reference clear, the component types will be capitalized.

This paper is organized as follows. A brief review of related work is given first. Sec. 3 describes the ASN data fusion layer. It includes the description of Bayesian decen-

tralized data fusion algorithm and its application to several probability density function (*pdf*) representations. Sec. 4 describes two algorithms to decentralized control. Sec. 5 demonstrates the architecture in two sets of experiments on an indoor sensor network.

## 2 Related Work

In terms of data fusion algorithms, Bayesian non-linear filtering is clearly stated for a single sensing platform in [4]. A broad review of distributed data fusion architectures can be found in [5]. An interesting query-response approach to distribute a particle filter is given in [6]. A general data fusion framework is proposed in [7] but its extension to a decentralized case is not clear.

In terms of application this work is most closely related to the field of sensor networks (SN) [8]. It is convenient to view the SN research by dividing it into three broad categories: multi-robot systems (MRS), macro SN (MSN) and micro SN ($\mu$SN). Small to medium team sizes (up to 100 platforms) in MRS are best handled by centralized and hierarchical approaches to data fusion and control [9, 10]. Very large team sizes, envisioned for $\mu$SN, require decentralized approaches. However, due to limited processing power the issues of information fusion (or "aggregation") are either not addressed or handled in a non-probabilistic fashion. The Directed Diffusion (DD) protocol [11] propagates data from sources to sinks and its non-probabilistic data aggregation approach is typical: duplicate target location estimates are suppressed by the intermediate nodes. Platforms are usually static, so control issues are not considered.

The MSN field is the closest to this work. The original ideas of channel filters used in present work were demonstrated on a model process control plant comprising over 150 distributed sensors [12]. More recently, Decentralized Data Fusion (DDF) was applied to tracking ground targets using four purpose-built fixed-wing Unmanned Aerial Vehicles [13]. Up to three of them were flown simultaneously during a three year flight program with up to six fusion nodes operating simultaneously and in real time. The program also dealt with such issues as track initialization and deletion, delayed and asequent data [14], timing, etc. Decentralized control (DC) was added to DDF in [15]. This work extends the DDF and DC, which are limited to Gaussian uncertainties, to the general Bayesian case.

In military systems, Cooperative Engagement Capability (CEC) is an operational US Navy multi-sensor tracking system [16]. It uses a fully connected architecture which limits its scalability. Tactical Component Network (TCN) [17] advocates in-network processing similar to DDF but algorithmic details are not available.

## 3 Bayesian Decentralized Data Fusion (BDDF) Algorithm

This section describes the algorithm underlying the data fusion layer of the ASN architecture. It fulfills the function of sensing, fusion, and dissemination of information.

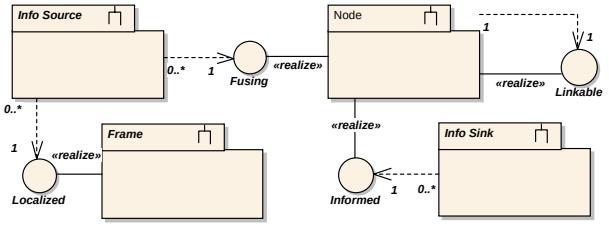



Fig. 1: Structural diagram of the data fusion layer.

### 3.1 Architecture and Interfaces

Data fusion design involves making many architectural choices, including: data fusion method, distribution in processing and storage, communication topology, type of exchanged information, degree of preprocessing, and many others. Listing our choices in the same order, the ASN approach to data fusion is: Bayesian, decentralized in processing and storage, over a tree or general network, utilizing both scan-to-track and track-to-track fusion, feature-based.

Because ASN is component-based, understanding interactions between components is required to understand the fusion algorithm. Fig. 1 shows component types implementing the data fusion layer of the system. Information Sources observe the environment, Nodes fuse and distribute information, Sinks request and use information, and Frames are responsible for localization and other platform related functions. The relationship between component types is described by the services each type provides and requires.

The Fusing interface is provided by Nodes. It accepts observations from Information Sources. Its multiplicity is one-to-many and it accepts "scans" in the form of observation likelihoods $L(z \mid \mathbf{x})$, where $z$ is a particular observation and $\mathbf{x}$ is the underlying state. The Linkable interface is used to connect Nodes into a network – the information backbone of the system. It represents a strictly one-to-one relationship between pairs of Nodes and exchanges tracks in the form of a *pdf* $P(\mathbf{x})$ over the state $\mathbf{x}$. The Informed interface is used to serve information to information consumers. It allows Sinks to specify quality of service (QoS) requirements which Nodes attempt to satisfy. Its multiplicity is one-to-many and the main information flow is in the form of track estimates $P(\mathbf{x})$. The Localized interface plays an important role of providing global localization to Information Sources. The localization method is chosen by individual Frames. The next two sections describe the internal structure of Sensor and Node components.

### 3.2 Sensor Realization

The Sensor component plays the role of an information source. First, the Sensor reads a raw measurement $z$ from the sensing hardware and converts it into the global coordinate frame. To perform the transformation, the Sensor needs to know its current global pose. It is calculated based on the current Frame pose (Localized interface) and the known offset of the physical sensor. The next step is to calculate an observation likelihood $L_s(z \mid \mathbf{x})$ based on

the observation function. Optionally, initial data association may be performed at the Sensor as well. Some sensors may have extra information useful for data association that may not be available at the Node. Finally, observations are submitted to the Node through the Fusing interface.

### 3.3 Node Realization

Fig. 2 shows the internal structure and the interdependencies between the subsystems of the Node component. The three main internal parts are the local filter, several channel filters, and the topology manager. The role of the local filter is to maintain the estimate of the state of the world. It realizes two interfaces: Fusing and Informed, used by Information Sources and Sinks respectively. Each Node has one local filter. The channel filter is used to manage communication between Nodes. It serves two main functions: to keep track of information previously communicated between the nodes ("through the channel"), and to synchronize incoming and outgoing information with the local filter. The topology manager is responsible for reconfiguration of the network topology. The three subsystems will be described in more detail below but first, the information flow between the local and channel filters is described.

Fig. 3(a) shows four triggers for activity on Node $i$. The four activities occur asynchronously – the fact illustrated in the Figure by separate "swim lanes". The first two are triggered by events external to the Node (arrival of observation and channel update messages), while the last two are triggered internally. Local updates are executed when local observations arrive. Channel filter updates are done using time horizon. Node-to-Node synchronization is triggered primarily by availability of information. Node-to-Sink connections are updated based on the requested accuracy (QoS). The algorithmic steps taken inside the local and channel filters for each of the events will be described in the sections below.

When an observation $L_s(z \mid \mathbf{x})$ arrives from one of the local Sensors through the Fusing interface, it is passed to the local filter where it is fused with the local estimate. When a channel update message arrives from one of the connected Nodes through the Linkable interface it is first processed internally in the channel filter. Then the current remote estimate $P_j(x \mid \mathbf{Z}^j)$ and the estimate of common information between the local and remote Nodes $P_{ij}(x)$ are passed to the local filter.

When an internal event triggers synchronization between the local filter and one of the channel filters, the current local estimate $P_i(x)$ is passed to the channel filter. The channel filters are periodically synchronized with the corresponding remote channel filters on the linked Nodes. Two possibilities exist for the form of the exchanged information: the current estimates $P_i(x)$ and $P_j(x)$ or the new information accumulated since the last update $M_{ij}(x)$ and $M_{ji}(x)$. Transmitting the current state and not just the new information offers a certain degree of robustness to packet loss in the channel. All information contained in the lost messages is implicitly present in future messages received

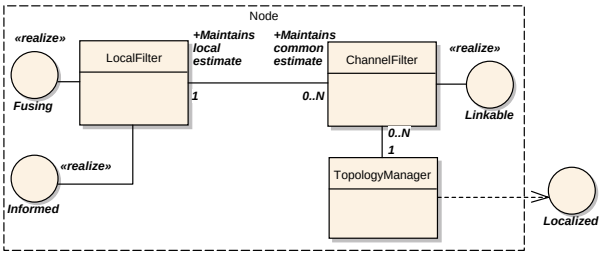



Fig. 2: Internal structure of the Node component.

after communication link is re-opened. Transmitting only the new information may save bandwidth for certain representations. A mixed policy is also possible. In the following discussion only the case of transmitting the full estimate is considered.

When an internal event triggers a state update of Information Sinks through the Informed interface, the current local estimate $P_i(x)$ is passed to the Sink. The same choice between sending the estimate or only the new information applies here as well.

The scalability of this approach stems from the fact that each node performs fusion. Incoming data from remote nodes is assimilated by the local filter *before* being passed on to the linked nodes. Therefore, no matter the number of incoming messages, there is only a single outgoing message to each node. Consequently, as the sensor network grows in size, the amount of information sent down any one channel remains constant and the system as a whole can scale indefinitely. Each node stores a local copy of all feature estimates. Thus, if the operation of the channel is suspended, the filter simply accumulates information in an additive fashion. When the channel is re-opened, the total accumulated information in the channel is communicated in one single message. This feature allows burst transmission of data to reduce communication bandwidth requirements.

The operation of Node's subsystems is described next. The algorithmic steps in processing of incoming information is described in terms of basic operations: associate, predict, subtract, fuse. These will be defined for an arbitrary *pdf* in Sec. 3.4 and then more precisely for several *pdf* representations in Sec. 3.5.

#### *3.3.1 Local Filter*

The local filter generates state estimates on the basis of observed, predicted and communicated information. Other infrastructure such as channel filters and the topology manager exist only to support the proper functioning of the local filter. The local filter contains an array of Bayesian filters representing individual features of the environment. Different feature types may be stored side by side and the filters may use different *pdf* representations.

The local filter receives observations from one or more local sensors as shown in the top row of Fig. 3(a). A sequence diagram in Fig. 3(b) shows the steps inside the local filter. Observations arriving asynchronously are stored in a buffer and the local filter is notified. Inside the local filter,

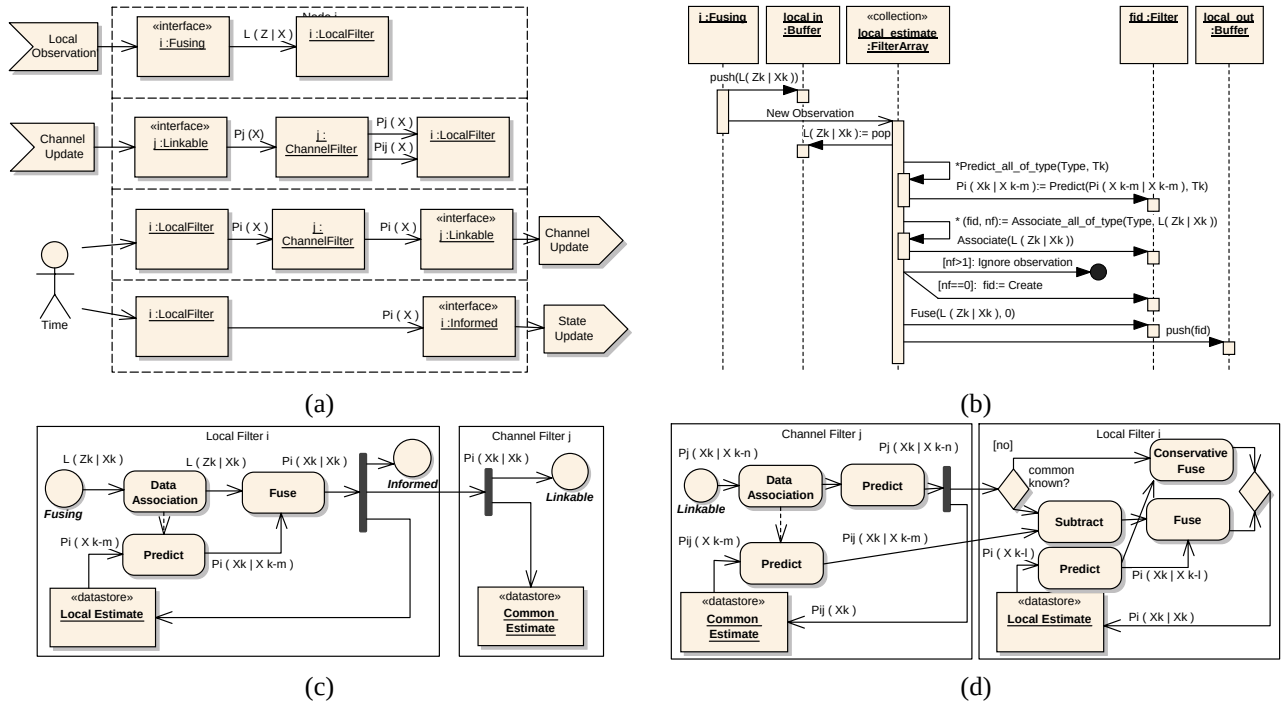



Fig. 3: External and internal information flow in the Node component: (a) four activity triggers and the information flow they cause; (b) detailed sequence diagram for the local filter with a sensor observation; (c) and (d) abbreviated information flow diagrams for local observation and channel update.

all features of the matching type are first predicted forward to the observation time. Data association is performed by matching the observation with the local estimate. The results of data association performed by the Sensor may also be used. If there is no match to an existing feature, a new one is created. Multiply associated observations are ignored. With correct association, the local filter fuses the observation and the prediction assuming conditional independence. The updated feature is marked as modified. Panel (c) shows the same procedure in less detail.

Processing of a channel update is shown in panel (d). The fusion procedure depends on whether the common information contained in the two estimates is known. If it is, then optimal fusion is possible. The common information is first subtracted from the remote estimate and, if the information gain is positive, the new information is fused with the latest local estimate which has been predicted to the time horizon. If common information is not known, then the two estimates are fused conservatively.

### *3.3.2 Channel Filter*

The channel filter maintains the information which is in common between two directly linked nodes. The information flow inside the channel filter is illustrated at the bottom of Fig. 3. The common estimate $P_{ij}(\cdot \mid \cdot)$ is updated in two cases: when node $i$ updates its local estimate through local observations or information from nodes other than $j$ (c) and when node $i$ receives information from node $j$ (d).

In the latter case, when node $i$ has information unknown to node $j$, a channel message is sent. The current local estimate $P_i(x_k)$ is placed into the outgoing buffer and sent at the appropriate time. The common estimate is updated accordingly. When a channel update message is received from node $j$, all estimates of common information are predicted to the next time horizon, and data association is performed. If there is no match to an existing feature, a new one is initialized. Multiply associated observations are ignored. If a positive match is found, the remote estimate is predicted to time horizon and placed into the local filter's incoming buffer.

### *3.3.3 Topology Manager*

A long-lived distributed system must be able to reconfigure during its life time. Common reasons for reconfiguration are addition, exit and especially failure of components; motion of platforms and features, etc. A strictly reactive topology control policy is currently implemented: connecting to services and responding to service outages. Channel filters require an acyclic network to function properly. Addition of fusion Nodes does not present a problem but exit or failure of Nodes in non-leaf positions may lead to cyclic networks possibly causing overconfidence in feature estimates. A decentralized algorithm for rebuilding a tree network after a Node failure is described in [18]. Proactive algorithms for network topology control is a subject of future research.

## 3.4 Data Fusion Primitives

A closer examination of the data flow diagrams in Fig. 3 reveals that, regardless of the implementation of the proba-

bilistic filter, the BDDF algorithm can be expressed in terms of five basic operations:

1. Data association
2. Prediction (motion update)
3. Scan-to-track fusion (fusing an estimate and an independent observation)
4. Track-to-track fusion (fusing two estimates with known common information)
5. Conservative track-to-track fusion (fusing two estimates with unknown common information)

These basic operations will be referred to as DDF primitives. In the remainder of this section we will define these primitives for a general *pdf*. The next section will specialize them to specific *pdf* representations, including Gaussian point features.

We consider an environment feature described by a state vector $\mathbf{x}_k = \mathbf{x}(t_k)$. The state is unknown and has to be estimated, it assumed to be Markovian. The feature is modelled with a probabilistic state transition function $P(\mathbf{x}_k \mid \mathbf{x}_{k-1})$. Let $\mathbf{z}_k = \mathbf{z}(t_k)$ be an observation of the feature with state $\mathbf{x}_k$. The sensor likelihood function is $L(\mathbf{z}_k \mid \mathbf{x}_k)$ and is assumed to depend only on the state of the feature at the time of the observation. The general Bayesian filtering problem is to find the posterior probability $P(\mathbf{x}_k \mid \mathbf{Z}^k, \mathbf{x}_0)$ at time $t_k$ based on the initial prior and the history of observations. The solution uses the recursive form of Bayes' theorem:

$$P(\mathbf{x}_k \mid \mathbf{Z}^k, \mathbf{x}_0) = \frac{P(\mathbf{z}_k \mid \mathbf{x}_k) P(\mathbf{x}_k \mid \mathbf{Z}^{k-1}, \mathbf{x}_0)}{P(\mathbf{z}_k \mid \mathbf{Z}^{k-1})} \quad (1)$$

It is convenient to separate Eq. 1 into two parts: the motion update which *predicts* the state from one time step to the next (the initial prior is omitted)

$$P(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{Z}^{k-1}) = \int P(\mathbf{x}_k \mid \mathbf{x}_{k-1}) P(\mathbf{x}_{k-1} \mid \mathbf{Z}^{k-1}) d\mathbf{x}_{k-1} \quad (2)$$

and the information update which *fuses* the information from the predicted estimate and an independent observation

$$P(\mathbf{x}_k \mid \mathbf{Z}^k) = \frac{1}{C} L(\mathbf{z}_k \mid \mathbf{x}_k) P(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{Z}^{k-1}) \quad (3)$$

The fusion of estimates (tracks) held by two Nodes requires identification of new information contained in one distribution relative to the other. The new information is easy to calculate if the common information between the two distributions $P(\mathbf{x}_k \mid \mathbf{Z}^k_{i \cap j})$ is known:

$$P(\mathbf{x}_k \mid \mathbf{Z}^k_{j \setminus i}) = \frac{1}{C} \frac{P(\mathbf{x}_k \mid \mathbf{Z}^k_j)}{P(\mathbf{x}_k \mid \mathbf{Z}^k_{i \cap j})}. \quad (4)$$

The combined estimate based on the observations of both Nodes can then be found:

$$P(\mathbf{x}_k \mid \mathbf{Z}^k_{i \cup j}) = P(\mathbf{x}_k \mid \mathbf{Z}^k_i) P(\mathbf{x}_k \mid \mathbf{Z}^k_{j \setminus i}). \quad (5)$$

Channel filters are introduced specifically to maintain the information common to two Nodes. If for some reason the channel filter cannot be trusted, then it is necessary to perform a conservative fusion of the two distributions. One way to do this is to simply keep the more informative one of the two and discard the other one. Entropy can be a measure of informativeness in this case. It is possible to salvage more information under some circumstances (see Sec. 3.5.1.)

The *data association* step is required if filtering uses observations of uncertain origin. It is often the most challenging part of the problem. Distance measures can be used for general distributions [19] but this remains a research topic.

## 3.5 Specialized Representations

The BDDF algorithm is defined for an arbitrary *pdf* but implementation details differ depending on the *pdf* representation. This section discusses specialization of DDF primitives to Gaussian point features, discrete general distributions, Certainty Grids, and mixed representations.

### *3.5.1 Gaussian Point Features*

Point features with Gaussian position uncertainty are most commonly used in the context of DDF due to the algorithmic efficiency which they allow. Consider a discrete system described in standard linear form

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{G}_k \mathbf{w}_k; \qquad \mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k, \quad (6)$$

where $\mathbf{x}_k$ is the state vector at time $t_k$, $\mathbf{F}_k$ is the state transition matrix from time $k-1$ to $k$, $\mathbf{G}_k$ the influence matrix for process noise, and $\mathbf{w}_k$ is the associated process noise modelled as an uncorrelated white sequence with $\mathrm{E}\left[\mathbf{w}_i \mathbf{w}_j^T\right] = \delta_{ij} \mathbf{Q}_i$. A vector of observations $\mathbf{z}_k$ is obtained at time $t_k$ according to the linear observation model $\mathbf{H}_k$ with the associated observation noise $\mathbf{w}_k$ modelled as an uncorrelated white sequence with $\mathrm{E}\left[\mathbf{v}_i \mathbf{v}_j^T\right] = \delta_{ij} \mathbf{R}_i$.

By direct substitution of the Gaussian *pdf* into the log-likelihood form of Bayes theorem, the information form of the Kalman filter (KF) can be derived [18]. This algorithm, called the *Information Filter* (IF), is numerically equivalent to the Kalman filter but is more suitable for decentralization. In the IF, the standard state $\mathbf{x}$ and covariance matrix $\mathbf{P}$ are replaced by the information matrix $\mathbf{Y} = \mathbf{P}^{-1}$ and the information vector $\hat{\mathbf{y}} = \mathbf{Y}\mathbf{x}$.

For *data association*, a simple innovation-based gating mechanism can be set up which dismisses new observations if considered unlikely. The *prediction* stage of the IF is more complicated compared to the KF form

$$\begin{aligned} \mathbf{Y}_{k,k-1} &= \mathbf{M}_k - \mathbf{M}_k \mathbf{G}_k \mathbf{\Sigma}^{-1} \mathbf{G}_k^T \mathbf{M}_k \\ \hat{\mathbf{y}}_{k,k-1} &= [1 - \mathbf{M}_k \mathbf{G}_k \mathbf{\Sigma}^{-1} \mathbf{G}_k^T] \mathbf{F}_k^{-T} \hat{\mathbf{y}}_{k-1} \\ &\quad + \mathbf{Y}_{k,k-1} \mathbf{B}_k \mathbf{u}_k, \end{aligned} \quad (7)$$

with $\mathbf{\Sigma}_k = \mathbf{G}_k^T \mathbf{M}_k \mathbf{G}_k + \mathbf{Q}_k^{-1}$ and $\mathbf{M}_k = \mathbf{F}_k^{-T} \mathbf{Y}_{k-1} \mathbf{F}_k^{-1}$. But the *update* stage is remarkably simple

$$\begin{aligned} \hat{\mathbf{y}}_{k,k} &= \hat{\mathbf{y}}_{k,k-1} + \sum \mathbf{i}_k \\ \mathbf{Y}_{k,k} &= \mathbf{Y}_{k,k-1} + \sum \mathbf{I}_k \end{aligned} \quad (8)$$

where $\mathbf{i}_k \equiv \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{z}_k$ and $\mathbf{I}_k \equiv \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k$. *Track-to-track fusion* with valid channel filter information is also straightforward:

$$\begin{aligned} \mathbf{Y}_{k,k}^{i\cup j} &= \mathbf{Y}_{k,k}^{i} + \mathbf{Y}_{k,k}^{i} - \mathbf{Y}_{k,k-1}^{i\cap j} \\ \hat{\mathbf{y}}_{k,k}^{i\cup j} &= \hat{\mathbf{y}}_{k,k}^{i} + \hat{\mathbf{y}}_{k,k}^{i} - \hat{\mathbf{y}}_{k,k-1}^{i\cap j} \end{aligned} \tag{9}$$

with $\mathbf{Y}^{i\cap j}$ and $\hat{\mathbf{y}}^{i\cap j}$ maintained by the channel filter. If the channel filter is invalid then a *conservative track-to-track fusion* method must be used. The Covariance Intersect (CI) algorithm allows to combine two Gaussian random variables when the correlation between them is unknown [20]:

$$\begin{aligned} \mathbf{Y}_{k,k}^{i\cup j} &= \omega \mathbf{Y}_{k,k}^{i} + (1-\omega)\mathbf{Y}_{k,k}^{j} \\ \hat{\mathbf{y}}_{k,k}^{i\cup j} &= \omega \hat{\mathbf{y}}_{k,k}^{i} + (1-\omega)\hat{\mathbf{y}}_{k,k}^{j}, \end{aligned} \tag{10}$$

where $\omega$ is selected based on a heuristic. The most commonly used heuristic is to select $\omega$ which minimizes the determinant of the resulting covariance matrix.

#### 3.5.2 General Discrete Distributions

Despite their computational advantages, parameterized probability distributions are often inadequate, e.g. in nonlinear filtering [4] and discrete identification [21]. It is quite easy in principle to decentralize the maintenance of a general distribution but the computational and communication burden required in practice presents a problem. The most common method of representing general distributions in centralized data fusion is the spatial grid. In this case the DDF primitives are the same as in Sec. 3.4 with the integral in Eq. 2 replaced by a discrete sum.

#### 3.5.3 Certainty Grids

The Certainty Grid (CG) [22] allows a simple and intuitive representation of distributed spatial information such as occupancy for indoor spaces or traversability for the outdoors. Formally, the certainty grid is a discrete-state binary random field. Each element encodes the probability of the corresponding grid cell $C_i$ being in a particular state, e.g. occupied $\hat{\mathbf{y}} = \log P(s(C_i) = \text{occ})$, shown in log-likelihood form. The representation can be extended (similar to [23]) to include more than two states. The decentralized CG may be viewed as a decentralized identification problem [21] and a special case of the discrete distributions discussed above. The identification is performed between occupied and empty states of each cell. Information accumulated by the decentralized network is the certainty of occupancy of each individual cell.

*Data association* is not required because the "features" (the CG cells) are referred to by their location. The locations of the cells in the OG map are assumed to be constant and known, so the *prediction* step typical of target tracking applications is not necessary. It is possible, however, to introduce a certain amount of information "forgetting" or blurring [22]. The time scale associated with information loss may reflect the expected dynamic aspect of the environment. Any information entered into the map is not permanent and has to be verified periodically. This feature, combined with the ability of mobile platforms to seek out new information, provides a limited ability of dealing with non-static environments.

*Scan-to-track fusion* fuses local observations with the local predicted estimate

$$\hat{\mathbf{y}}_{k,k}^{i} = \hat{\mathbf{y}}_{k,k-1}^{i} + \sum \mathbf{i}_k + C, \tag{11}$$

where $C$ is a normalization constant. *Track-to-track fusion* with known common information is a matter of simple addition and subtraction

$$\hat{\mathbf{y}}_{k,k}^{i\cup j} = \hat{\mathbf{y}}_{k,k-1}^{i} + \hat{\mathbf{y}}_{k,k-1}^{j} - \hat{\mathbf{y}}_{k,k-1}^{i\cap j} + C. \tag{12}$$

*Conservative track-to-track fusion* must be employed if the observation history is not known. No special algorithms are available.

#### 3.5.4 Mixed Representations

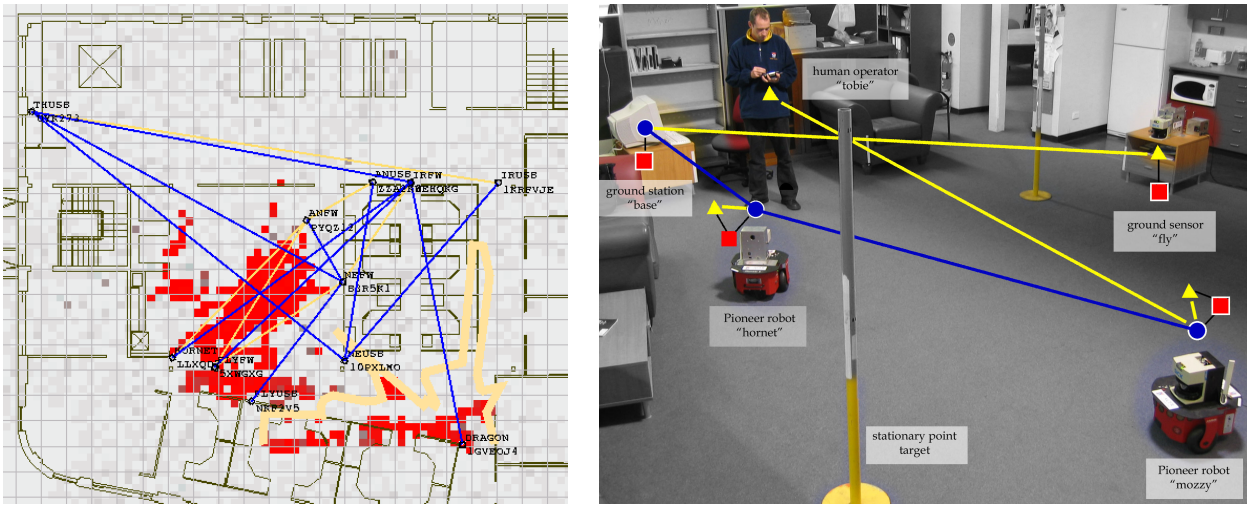
When choosing the right *pdf* representation for a feature, it is often possible to break up the state vector into independent subsets and represent each subset with the most appropriate representation. One use for a hybrid state is in joint target tracking and classification algorithms [24]. In [25], the identity of the target is represented by a a particle filter and its position by a Gaussian. In another application, data association and probability of track existence are combined in the Integrated Probabilistic Data Association (IPDA) algorithm [26]. These hybrid states and others can be decentralized provided that required algorithms exist for the representations chosen for each state subset.

## 4 Decentralized Control

The data fusion layer of the ASN leads to a synchronized view of the state of the environment. Based on this belief, sensing platforms equipped with actuators can make individual control decisions to maximize the team utility function. Two algorithms have been examined within the ASN framework: coordinated and cooperative control.

### 4.1 Coordinated Control

The coordinated control algorithm [15] predicts and maximizes the expected information gain from local sensors without any knowledge of the choices made by other decision makers. The ASN propagates observed information influencing the locally optimized sensing plans. Consequently, by simply activating the data fusion layer of ASN and keeping the control policies independent at each platform, a coordinated control solution is obtained. Zero-look-ahead solution, or "information surfing", is of special interest because of the low computational effort required.




(a) (b)

Fig. 4: The experiments: the "motion" map (a) and point feature tracking (b). Blue lines stand for DDF links.

### 4.2 Cooperative Control through Negotiation

An interesting extension of BDDF is a decentralized cooperative control scheme [27] which engages each decision maker in anonymous negotiation based on propagation of *expected* observation information through special *negotiation channels*. These channels are identical to the regular BDDF channels in all respects except that they exchange expected information gains instead of the actual ones. Controller components act as both sources and sinks of expected information. Nodes fuse and disseminate global estimates. The same BDDF primitives and specialized representations apply.

Team decisions are reached as follows. Each decision maker updates its sensing plan using a better-response procedure and communicates the change in expected observation information. Plans are withdrawn by submitting negative information gains. The negotiation cycle is repeated to determine the sensing actions that optimize the team utility. Experimental validation of this algorithm remains a subject of future work.

## 5 Experiments

Two sets of indoor experiments differ in environment representation, team makeup, and deployment patterns. Both use a version of the BDDF algorithm and are implemented using the ASN application framework.

The task in the first set of experiments is to build a dynamic map of motion in the office space stored in the Certainty Grid format (Sec. 3.5.3). The task is performed by a team of stationary lasers and video cameras. Fig. 4(a) shows the resulting "motion" map built by a system with up to 39 components on 11 hosts. The dark (red) cells mark the locations where motion has been detected. The office layout is superimposed by hand for clarity.

The task in the second set of experiments is to estimate locations of stationary point targets [28]. This time the team is a mix of two Pioneer robots, a stationary sensor module (all equipped with laser range finders), and two operators. Fig. 4(b) shows the overall view of the system which involved 13 components on 5 hosts: 3 nodes (blue circles) execute the IF algorithm (Sec. 3.5.1); 3 Sensors convert laser scans to Gaussian point observations; 2 Controllers implement the coordinated information surfing algorithm (Sec. 4.1). Features 0 and 1 in Fig. 5(a) were acquired by the robotic Sensors, feature 2 is outside of the platforms' sensor range and was entered by the operator. The mobile platforms respond to the new information by driving to the feature and observing it as shown in Fig. 5(b).

## 6 Conclusions

The BDDF algorithm presented in this paper extends decentralized data fusion techniques to general probability distribution. BDDF is a direct extension of DDF in that it uses the idea of channel filters but without the limiting Gaussian assumption. Similarly, the earlier work on decentralized information-theoretic control is reformulated for the case of arbitrary probability distributions.

Future work in the area of data fusion involves specializing BDDF to non-parametric representations of arbitrary distribution. The general Bayesian formulation opens the door for a generic Node implementation with a provision for various representation plug-ins. On the control side, experimental validation of the cooperative control algorithm is needed.

## Acknowledgment

This work is partly supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government, and by AFOSR/AOARD under contract 03-13.

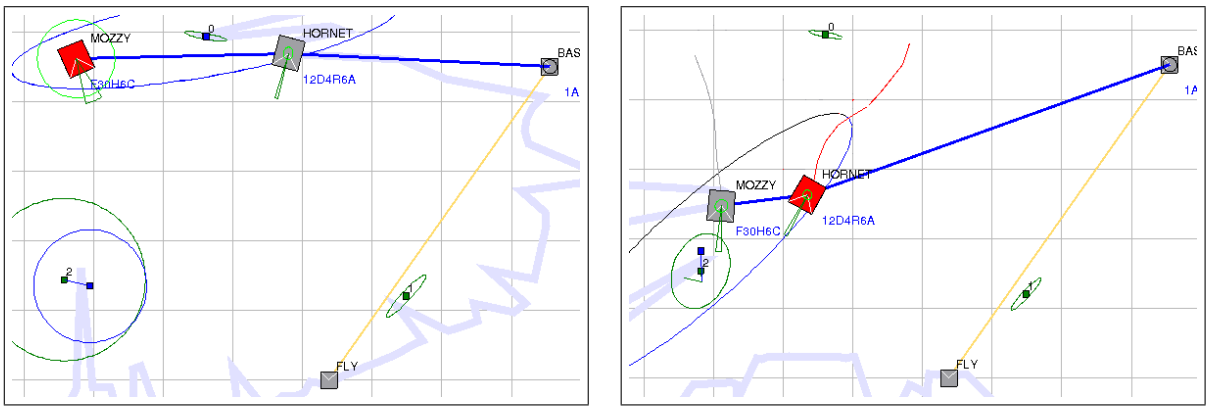



Fig. 5: Tracking point features: operator enters feature 2 (left); platforms drive to and observe it (right).

## References


[1] A Makarenko, A Brooks, SB Williams, HF Durrant-Whyte, and B Grocholsky. An architecture for decentralized active sensor networks. In *IEEE ICRA*, New Orleans, LA, 2004.

[2] A Brooks, A Makarenko, T Kaupp, S Williams, and H Durrant-Whyte. Implementation of an indoor active sensor network. In *Int. Symp. on Exp. Robotics*, Singapore, 2004.

[3] J Manyika and HF Durrant-Whyte. *Data Fusion and Sensor Management: a decentralized information-theoretic approach*. Ellis Horwood, 1994.

[4] L Stone, C Barlow, and T Corwin. *Bayesian multiple target tracking*. Artech House, London, 1999.

[5] ME Liggins II, C-Y Chong, I Kadar, MG Alford, V Vannicola, and S Thomopoulos. Distributed fusion architectures and algorithms for target tracking. *Proc. of the IEEE*, 85(1):95–107, 1997.

[6] M Rosencrantz, G Gordon, and S Thrun. Decentralized sensor fusion with distributed particle filters. In *UAI-03*, 2003.

[7] H Carvalho, W Heinzelman, A Murphy, and C Coelho. A general data fusion architecture. In *Int. Conf. on Info. Fusion*, pages 1465–1472, 2003.

[8] H Qi, SS Iyengar, and K Chakrabarty. Distributed sensor fusion – a review of recent research. *J. of the Franklin Inst.*, 388:655–668, 2001.

[9] RG Simmons, , and et al. Coordination for multi-robot exploration and mapping. In *AAAI Nat. Conf. on AI*, pages 852–8, Austin, TX, 2000.

[10] K Konolige, C Ortiz, R Vincent, A Agno, M Eriksen, B Limketkai, M Lewis, L Brieseister, and E Ruspini. CentiBOTS: Large scale robot teams. In *Int. Workshop on Multi-Robot Systems*, pages 193–204, Washington, DC, 2003.

[11] C Intanagonwiwat, R Govindan, D Estrin, J Heidemann, and F Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. on Networking*, 11(1):2–16, 2003.

[12] S Grime and H Durrant-Whyte. Data fusion in decentralized sensor networks. *Control Eng. Practice*, 2(5):849–63, 1994.

[13] S Sukkarieh, E Nettleton, J-H Kim, M Ridley, A Goktogan, and HF Durrant-Whyte. The ANSER project: Data fusion across multiple uninhabited air vehicles. *Int. J. of Rob. Research*, 22(7/8):505–540, 2003.

[14] E Nettleton and H Durrent-Whyte. Delayed and asequent data in decentralised sensing networks. In *SPIE Photonics*, pages 1–9, Boston, MA, 2001.

[15] B Grocholsky, A Makarenko, T Kaupp, and HF Durrant-Whyte. Scalable control of decentralised sensor platforms. In *Int. Workshop on Info. Processing in Sensor Networks*, pages 96–112, Palo Alto, CA, 2003.

[16] CB Sheehy. Data selectivity vital to operational picture. *SIGNAL Magazine*, (May), 2001.

[17] Solipsys Corp. Tactical component network: Overview. White paper, http://www.solipsys.com/tcn.php, 2000.

[18] E Nettleton. *Decentralised Architectures for Tracking and Navigation with Multiple Flight Vehicles*. PhD, U. of Sydney, 2003.

[19] JN Kapur. *Measures of Information and their Application*. John Wiley, 1994.

[20] S Julier and J Uhlmann. General decentralised data fusion with covariance intersection (CI). In D Hall and J Llinas, editors, *Handbook of Data Fusion*. CRC Press, 2001.

[21] J Berger. *Statistical decision theory and Bayesian analysis*. Springer-Verlag, New York, 1985.

[22] A Elfes. Robot navigation: Integrating perception, environmental constraints and task execution within a probabilistic framework. In *Reasoning with Uncertainty in Robotics. Int. Workshop*, pages 93–129, Amsterdam, Netherlands, 1995.

[23] C Stachniss and W Burgard. Mapping and exploration with mobile robots using coverage maps. In *IEEE/RSJ IROS*, pages 467–472, Las Vegas, NV, 2003.

[24] S Challa and GW Pulford. Joint target tracking and classification using radar and esm sensors. *IEEE Trans. on Aerospace and Electronic Sys.*, 37(3):1039–1055, 2001.

[25] D Fox, J Hightower, L Liao, D Schulz, and G Borriello. Bayesian filtering for location estimation. *IEEE Pervasive Comp. Mag.*, 2(3):24–33, 2003.

[26] D Musicki, R Evans, and S Stankovic. Integrated probabilistic data association. *IEEE Trans. on Auto. Contr.*, 39(6):1237–41, 1994.

[27] B Grocholsky. *Information-Theoretic Control of Multiple Sensor Platforms*. PhD, U. of Sydney, 2002.

[28] A Makarenko, T Kaupp, and HF Durrant-Whyte. Scalable human-robot interactions in active sensor networks. *IEEE Pervasive Computing Mag.*, 2(4):63–71, 2003.