Sensor Fusion and Decentralized Control in Robotic Systems III

Peter W. Gibbens
The University of Sydney
35 PUBLICATIONS **734** CITATIONS

Ali Haydar Göktoğan
The University of Sydney
110 PUBLICATIONS **1,153** CITATIONS

Multiple-platform localization and map building

Eric W. Nettleton, Hugh F. Durrant-Whyte, Peter W. Gibbens, Ali H. Goektogan

Eric W. Nettleton, Hugh F. Durrant-Whyte, Peter W. Gibbens, Ali H. Goektogan, "Multiple-platform localization and map building," Proc. SPIE 4196, Sensor Fusion and Decentralized Control in Robotic Systems III, (16 October 2000); doi: 10.1117/12.403733

Event: Intelligent Systems and Smart Manufacturing, 2000, Boston, MA, United States

Multiple Platform Localisation and Map Building

Eric W. Nettleton, Hugh F. Durrant-Whyte, Peter W. Gibbens and Ali H. Göktoğan

Australian Centre for Field Robotics
Department of Aeronautical, Mechanical and Mechatronic Engineering
The University of Sydney,
NSW 2006, Australia

ABSTRACT

This paper presents current work on decentralised data fusion (DDF) applied to multiple unmanned aerial vehicles. The benefits of decentralising algorithms, particularly in this field, are enormous. At a mission level, multiple aircraft may fly together sharing information with one another in order to produce more accurate and coherent estimates, and hence increase the their chances of success. At the single platform level, algorithms may be decentralised throughout the airframe reducing the probability of catastrophic failure by eliminating the dependency on a particular central processing facility.

To this end, a complex simulator has been developed to test and evaluate decentralised picture compilation, platform localisation and simultaneous localisation and map building (SLAM) algorithms which are to be implemented on multiple airborne vehicles. This simulator is both comprehensive and modular, enabling multiple platforms carrying multiple distributed sensors to be modelled and interchanged easily. The map building and navigation algorithms interface with both the simulator and the real airframe in exactly the same way in order to evaluate the actual flight code as comprehensively as possible. Logged flight data can also be played back through the simulator to the navigation routines instead of simulated sensors.

This paper presents the structure of both the simulator and the algorithms that have been developed. An example of decentralised map building is included, and future work in decentralised navigation and SLAM systems is discussed.

Keywords: Simulator, Picture Compilation, SLAM, Information Filter, Multiple Platform

1. INTRODUCTION

This paper is concerned with the development of algorithms for decentralised, multiple vehicle, map building. The motivating scenario is that of multiple unmanned flight vehicles, each equipped with one or more terrain sensors, cooperatively building a map of the terrain over which they are flying. Ideally, any number of vehicles, with any number of payloads in any configuration should be accommodated by the map building algorithm.

Decentralised architectures offer a number of advantages over conventional centralised, hierarchical or federated architectures for data fusion and map building tasks. First, they allow networks of sensors to communicate information in an efficient, modular and scalable manner. Second, they allow information-theoretic ideas to be used to control and manage the flow of information between different systems. For this reason, there has been a great deal of research in the area over recent years.[1–5]

There are three basic constraints to a general decentralised system:

1. There is no single central fusion centre and no node should be central to the operation of the network.
2. There is no common communications facility - communications must be kept on a strictly node-to-node basis.
3. Each node has knowledge only of its immediate neighbours - there is no global knowledge of the network topology.

The algorithms developed are based on the information filter form of the conventional Kalman filter. The information form of the Kalman filter employs the inverse covariance matrix as the main means for communicating information. It yields exactly the same numerical result as the conventional Kalman filter but allows simple decentralisation of algorithms.

Further author information:
E.W.N.: E-mail: ericn@mech.eng.usyd.edu.au

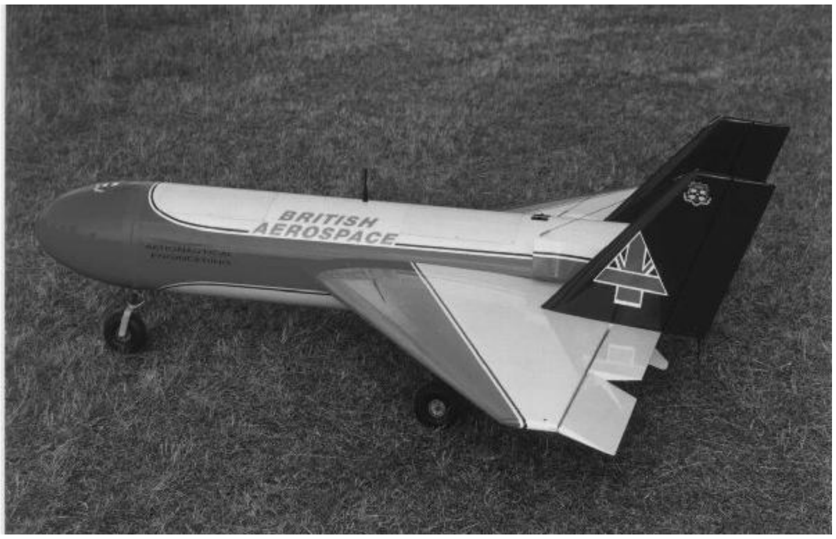


Figure 1. A Brumby Mk.1 Airframe.

The work described in this paper is part of the Autonomous Navigation and Sensing Experimental Research (ANSER) project and is aimed at the development of a multiple flight vehicle demonstration of decentralised data fusion. The system under development consists of four unmanned flight vehicles of the type shown in Figure 1. Each flight vehicle is equipped with GPS and inertial sensors and carries two terrain payloads; a vision system and either a mm-wave radar or laser sensor. Each payload incorporates it's own modular, fully decentralised processing hardware. On-board, the payloads communicate with each other using a CAN bus. Inter-vehicle communication is via spread-spectrum radio. Each payload processor implements a fully decentralised data fusion algorithm. Payloads communicate with each other directly in terms of terrain information; all data fusion and assimilation occurs at the payload site. There is no separate fusion centre on any flight platform and no fusion centre elsewhere on the ground. The architecture is thus decentralised, fully modular and scalable.

In order to develop and test these decentralised algorithms successfully, a simulator has been developed to mimic the flight vehicle and its on-board sensors. The data fusion nodes can be connected to the simulator or the airframe transparently.

The initial development has centred on decentralised picture compilation, or map building. This problem involves multiple platforms flying and building a map of features on the ground. Note that this does not involve any pose estimation of the flight vehicle. Each platform maintains a decentralised information filter for the features it observes, and transmits the information to all other platforms. Thus, the net result is that each node on each platform has a full map of all features that all platforms have seen. In addition to this, if multiple aircraft are looking at the same features then the extra information they communicate to each other results in a more accurate estimate of the features location.

The next stage of development is decentralised vehicle navigation. This involves a single airborne platform with a decentralised onboard architecture flying over and making observations of features at known locations and using these observations to estimate the vehicle pose.

Decentralised simultaneous localisation and map building (SLAM) is a combination of both the aforementioned problems. Multiple platforms with decentralised onboard architectures fly with no *a priori* map knowledge. They start at an unknown location in an unknown environment and simultaneously make relative observations to features in the environment, building a relative map, and use this map to navigate. By using multiple platforms, it is possible to share map information and hence increase the map accuracy. As each platform uses the map to navigate, the result is a more accurate estimate of the vehicle states.

There has been a significant amount of research in the standard centralised state space SLAM problem over the last few years in areas such as convergence and large scale implementations.[6–12] Recently, research has begun on information space implementations of the problem,[13] leading to a decentralised SLAM architecture. Closed form solutions to the continuous time problem have been presented.[14,15]

This paper focuses on a description of the decentralised architecture for this system; how the algorithms are mapped to the architecture, and some preliminary results from simulation of these algorithms.

2. SYSTEM DESIGN

2.1. Decentralised Information Filter

Consider a system described in linear form

$$\mathbf{x}(k) = \mathbf{F}(k)\mathbf{x}(k-1) + \mathbf{w}(k), \tag{1}$$

where $\mathbf{x}(j)$ is the state of interest at time j, $\mathbf{F}(k)$ is the state transition matrix from time $k-1$ to k, and where $\mathbf{w}(k)$ is the associated process noise modeled as an uncorrelated white sequence with $\mathrm{E}\{\mathbf{w}(i)\mathbf{w}^T(j)\} = \delta_{ij}\mathbf{Q}(i)$. The system is observed by a sensor according to the linear equation

$$\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{v}(k) \tag{2}$$

where $\mathbf{z}(k)$ is the vector of observations made at time k, $\mathbf{H}(k)$ the observation matrix or model, and where $\mathbf{v}(k)$ is the associated observation noise modeled as an uncorrelated white sequence with $\mathrm{E}\{\mathbf{v}(i)\mathbf{v}^T(j)\} = \delta_{ij}\mathbf{R}(i)$. The Kalman filter algorithm generates estimates for the state $\hat{\mathbf{x}}(k \mid k)$ at a time k given all observations up to time k, together with a corresponding estimate covariance $\mathbf{P}(k \mid k)$ as:

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}(k)\left[\mathbf{z}(k) + \mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1)\right] \tag{3}$$

$$\mathbf{P}(k \mid k) = \mathbf{P}(k \mid k-1) - \mathbf{W}(k)\mathbf{S}(k)\mathbf{W}^T(k) \tag{4}$$

where the gain matrix $\mathbf{W}(k)$ is given by

$$\mathbf{W}(k) = \mathbf{P}(k \mid k-1)\mathbf{H}^T(k)\mathbf{S}^{-1}(k), \tag{5}$$

and the innovation covariance by

$$\mathbf{S}(k) = \mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}^T(k) + \mathbf{R}(k). \tag{6}$$

The information form of the Kalman filter is obtained by re-writing the state estimate and covariance in terms of two new variables

$$\hat{\mathbf{y}}(i \mid j) \overset{\triangle}{=} \mathbf{P}^{-1}(i \mid j)\hat{\mathbf{x}}(i \mid j), \qquad \mathbf{Y}(i \mid j) \overset{\triangle}{=} \mathbf{P}^{-1}(i \mid j), \tag{7}$$

and also the information associated with an observation in the form

$$\mathbf{i}(k) \overset{\triangle}{=} \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{z}(k), \qquad \mathbf{I}(k) \overset{\triangle}{=} \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k) \tag{8}$$

With these definitions, the Kalman filter (Equations 3, 4, 5 and 6) become

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \mathbf{i}(k), \tag{9}$$

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \mathbf{I}(k). \tag{10}$$

The information form of the Kalman filter, while widely known, is not commonly used because the update terms are of dimension the state, whereas in the distributed Kalman filter updates are of dimension the observation. For single sensor estimation problems, this argues for the use of the Kalman filter over the information filter. However, in multiple sensor problems, the opposite is true. The reason is that with multiple sensor observations

$$\mathbf{z}_i(k) = \mathbf{H}_i(k)\mathbf{x}(k) + \mathbf{v}_i(k), \qquad i = 1, \cdots, N$$

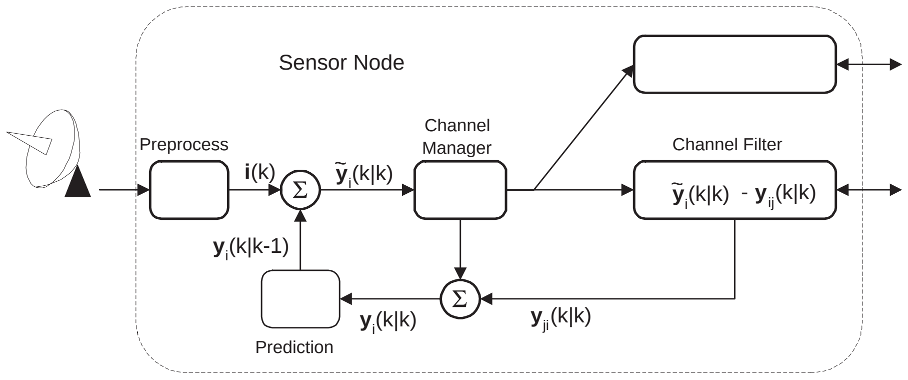


Figure 2. Algorithmic structure of a decentralised sensing node.

the estimate can not be constructed from a simple linear combination of contributions from individual sensors

$$\hat{\mathbf{x}}(k \mid k) \neq \hat{\mathbf{x}}(k \mid k-1) + \sum_{i=1}^{N} \mathbf{W}_i(k) \left[\mathbf{z}_i(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}(k \mid k-1)\right],$$

as the innovation $\mathbf{z}_i(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}(k \mid k-1)$ generated from each sensor is correlated because they share common information through the prediction $\hat{\mathbf{x}}(k \mid k-1)$. However, in information form, estimates can be constructed from linear combinations of observation information

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \sum_{i=1}^{N} \mathbf{i}_i(k),$$

as the information terms $\mathbf{i}_i(k)$ from each sensor are uncorrelated. Once the update equations have been written in this simple additive form, it is straight-forward to distribute the data fusion problem (unlike for a Kalman filter); each sensor node simply generates the information terms $\mathbf{i}_i(k)$, and these are summed at the fusion center to produce a global information estimate.

To decentralise the information filter all that is necessary is to replicate the central fusion algorithm (summation) at each sensor node and simplify the result. This yields a surprisingly simple nodal fusion algorithm. The algorithm is described graphically in Figure 2. Essentially, local estimates are first generated at each node by fusing (adding) locally available observation information $\mathbf{i}_i(k)$ with locally available prior information $\hat{\mathbf{y}}_i(k \mid k-1)$. This yields a local information estimate $\tilde{\mathbf{y}}_i(k \mid k)$. The difference between this local estimate and prediction (corresponding to new information gained) is then transmitted to other nodes in the network. In a fully connected or broadcast network, this results in every sensing node getting all new information. Communicated information is then assimilated simply by summing with the local information. An important point to note is that, after this, the locally available estimates are *exactly* the same as if the data fusion problem had been solved on a single central processor using a monolithic formulation of the conventional Kalman filter.

2.2. Decentralised Architecture

As was mentioned previously, the ANSER project aims to demonstrate a complete decentralised data fusion architecture. This means that the algorithms are decentralised throughout each individual platform as well as across multiple aircraft. This concept is illustrated in Figure 3. Within each platform, each sensor is attached to a different processing node.

Figure 2 shows the structure of a typical sensor/processing node, i, in a decentralised data fusion system. This corresponds to each of the sensor/estimate/channel sets in Figure 3. The node generates information measures $\hat{\mathbf{y}}_i(k \mid k)$ at a time k given observations made locally and information communicated to the node up to time k. The node implements a local prediction stage to produce information measure predictions $\hat{\mathbf{y}}_i(k \mid k-1)$ at time k given

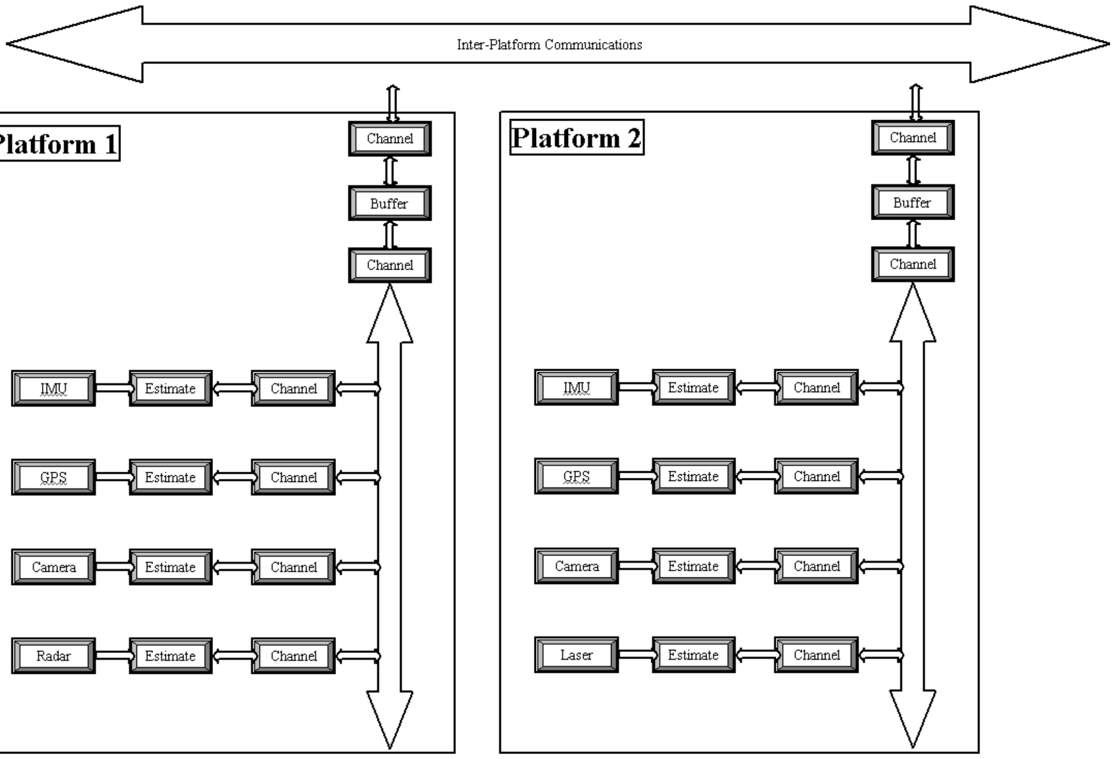


Figure 3. Multiple platform DDF block diagram

all local and communicated data up to time $k-1$ (this prediction stage is often the same on each node and may, for example, correspond to the path predictions of a number of common targets). At this time, local observations produce local information measures $\mathbf{i}(k)$ on the basis of local observations. The prediction and local information measures are combined, by simple addition, into a total local information measure $\tilde{\mathbf{y}}_i(k \mid k)$ at time k. This measure is handed down to the communication channels for subsequent communication to other nodes in the decentralised network. Incoming information from other nodes $\hat{\mathbf{y}}_{ji}(k \mid k)$ is extracted from appropriate channels and is assimilated with the total local information by simple addition. The result of this fusion is a locally available global information measure $\hat{\mathbf{y}}_i(k \mid k)$. The algorithm then repeats recursively.

The communication channels exploit the associativity property of information measures. The channels take the total local information $\tilde{\mathbf{y}}_i(k \mid k)$ and subtract out all information that has previously been communicated down the channel, $\hat{\mathbf{y}}_{ij}(k \mid k)$, thus transmitting only new information obtained by node i since the last communication. Intuitively, communicated data from node i thus consists only of information not previously transmitted to a node j; because common data has already been removed from the communication, node j can simply assimilate incoming information measures by addition. As these channels essentially act as information assimilation processes, they are usually referred to as channel filters

Channel filters have two important characteristics:

1. Incoming data from remote sensor nodes is assimilated by the local sensor node *before* being communicated on to subsequent 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. This leads to an important conclusion: *A decentralised data fusion system can scale up in size indefinitely without any increase in node-to-node communication bandwidth.*

2. A channel filter compares what has been previously communicated with the total local information at the node. Thus, if the operation of the channel is suspended, the filter simply accumulates information in an additive

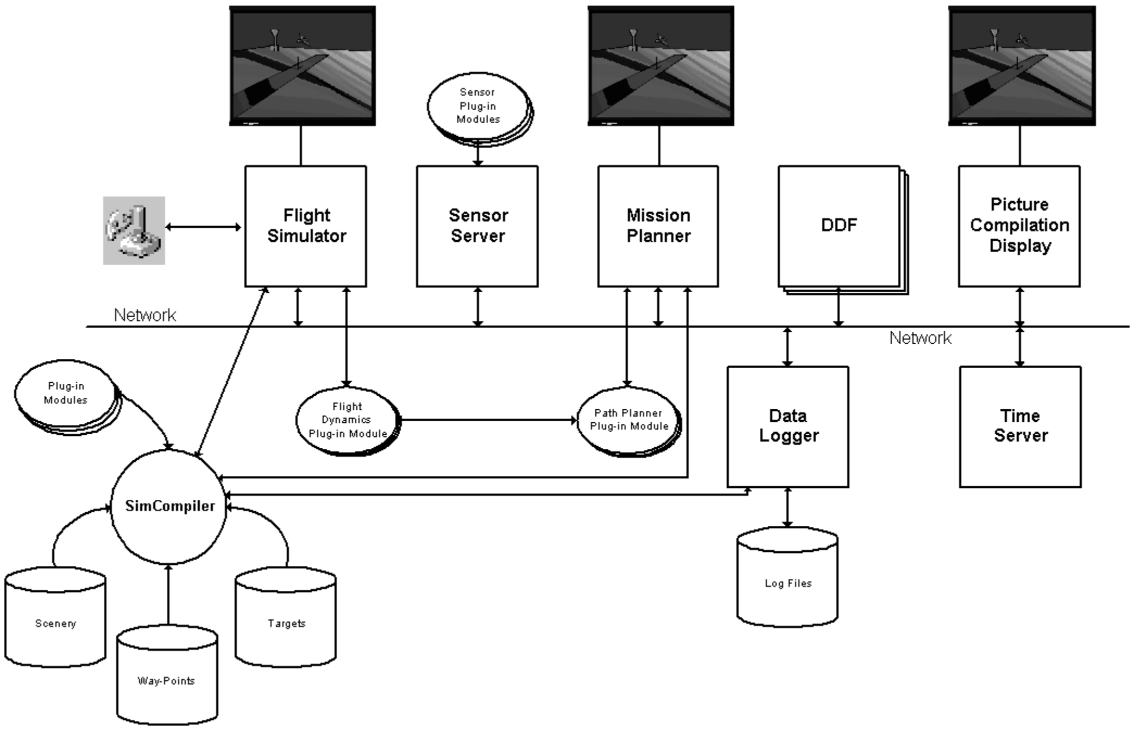


Figure 4. Simulator block diagram

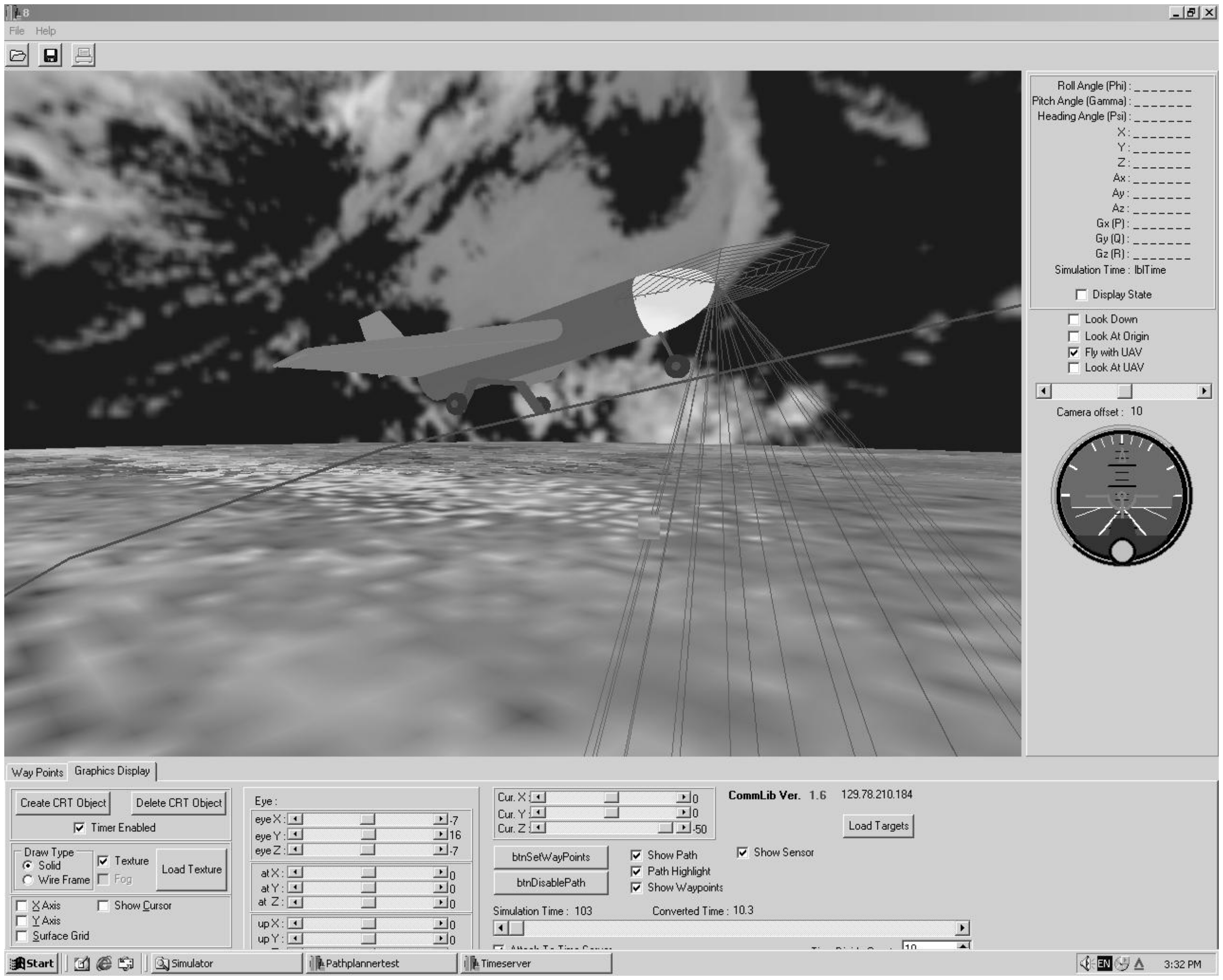
fashion. When the channel is re-opened, the total accumulated information in the channel is communicated in one single message. The consequences of this are many-fold; burst transmission of accumulate data can be employed to substantially reduce communication bandwidth requirements (and indeed be used to manage communications); if a node is disconnected from the communications network, it can be re-introduced and information synchronised in one step (the same applies to new nodes entering the system, dynamic network changes and signal jamming). This leads to a second important conclusion: *A decentralised data fusion system is robust to changes and loss of communication media.*

Together, the node and channel operation define a system which is fully modular in algorithmic construction, is indefinitely scalable in numbers of nodes and communication topology, and is survivable in the face of failures of both sensor nodes and communication facility.

3. SIMULATOR AND ALGORITHM DESIGN

In order to successfully develop a decentralised data fusion network, an effective mechanism to test it is required. The requirements for such a tool are significantly different from those of a standard centralised architecture, where all data simply needs to be sent to the one location. Rather, in the decentralised case, there are a large number of parallel operations taking place which cannot be effectively simulated using a standard sequential processing environment. To do so would require ignoring many practical considerations, not least of which are timing and bandwidth. To solve this problem, a decentralised multiprocessor simulator was developed to mimic a real operating environment as closely as possible. Figure 4 illustrates the structure of the simulator in a block diagram.

Each block in the schematic represents a different module in the simulator, and can be run on any machine in the system. Starting with the DDF nodes, each would be run on a separate computer in exactly the same way as in the actual implementation. The DDF nodes receive simulated observations from their respective sensors from the sensor server, which interacts with the stored terrain and features via the user plug in sensor models. A new sensor model can be developed and added (or modified) very easily using this structure, and as the sensor server has the true



Roll Angle (Phi) : _______
Pitch Angle (Gamma) : _______
Heading Angle (Psi) : _______
X : _______
Y : _______
Z : _______
Ax : _______
Ay : _______
Az : _______
Gx (P) : _______
Gy (Q) : _______
Gz (R) : _______
Simulation Time : lblTime
☐ Display State
☐ Look Down
☐ Look At Origin
✓ Fly with UAV
☐ Look At UAV
Camera offset : 10

✓ Timer Enabled
Draw Type: ● Solid ○ Wire Frame
✓ Texture ☐ Fog
☐ X Axis ☐ Show Cursor
☐ Y Axis
☐ Surface Grid

Eye :
eye X : -7
eye Y : 16
eye Z : -7
at X : 0
at Y : 0
at Z : 0
up X : 0
up Y : 0

Cur. X : 0
Cur. Y : 0
Cur. Z : -50
✓ Show Path
✓ Path Highlight
✓ Show Waypoints
✓ Show Sensor
CommLib Ver. 1.6 129.78.210.184
Simulation Time : 103
Converted Time : 10.3

3:32 PM

Figure 5. Simulator screen capture. The line along the axis of the aircraft is the flightpath it is following and the conical shape extending down from the platform nose represents the radars field of view.

state of the entire simulated world, the model can be made as simple or as complex as is required. The sensor server receives the true state of all of the platforms in the world from the flight simulator, which can be controlled by using joystick inputs to the flight model to fly the platform, by replaying recorded flight data or by using the waypoint generator to calculate a trajectory for the platform. In addition to the algorithmic components of the simulator, open GL graphic interfaces have been developed to display live movies of both the true simulator world and the estimates of the world generated by the DDF nodes. Figure 5 shows a screen shot of the vehicle following a predetermined flightpath. All data can also be logged to file for post run analysis. Finally, the simulator timeserver acts as the common clock source for all modules in the system, and can be sped up or slowed down to suit the requirements of the simulation.

The primary purpose of this simulator is to provide a tool for developing and testing the real flight algorithms. It has been structured in this modular form with the specific objective of allowing the flight code to be comprehensively evaluated and further, to allow for easy evaluation of logged flight data.

3.1. Algorithm Design

3.1.1. Picture Compilation

The decentralised picture compilation implementation is an exercise in map building. Multiple platforms fly around an area using an external sensor such as a radar, laser, camera or a combination making relative observations to features in the environment. In this problem, localisation is not considered and the pose of the platform is known at all times. Platforms estimate the location of features on the ground. Assuming a sensor made an observation of the x,y,z location of a feature i, then $\mathbf{z}_i(k) = [x, y, z]^T$ and $\mathbf{H}(k)$ is the 3x3 identity matrix.

Each platform maintains a separate filter for each feature, and upon making on observation and associating it correctly, transmits a new estimate to all other nodes in the same platform as well as to all other aircraft. Thus, all nodes in the system share all information, which results in a steady state where each node has the same complete map of all features seen by any of the sensors. Should any one sensor or node fail, it is clear that the rest will continue to perform as normal rather than experience any catastrophic failure.

3.1.2. Vehicle Localisation

Decentralised vehicle localisation is a scenario where multiple sensors are distributed throughout a single platform. The sensors are an IMU, GPS and some combination of a radar, laser and camera. The external sensors make observations of features on the ground, the locations of which are known *a priori*. These are used with a standard decentralised information filter to estimate the platform pose.

The IMU node runs an extended information filter to act as the prediction stage for the platform. All other nodes use a simpler linear prediction model to predict delayed observation information, and read in the information vectors from the inertial node as their prediction. That is, the information vector output by the inertial node actually contains a loss in information as it is a prediction, which means that when other nodes read it in the channel filter will calculate a negative information gain and the covariance of the system will grow as expected. All other nodes make observations in the normal way, and perform updates that add information to the system.

3.1.3. Simultaneous Localisation and Map Building

In the SLAM problem, platforms have no *a priori* knowledge of their location or of the location of the features. The internal structure of each platform is as in the vehicle localisation case, where the IMU drives the primary platform prediction model. All external sensors make relative observations to external features and then augment the feature states into the information state vector. This then results in a SLAM information matrix with platform states in the top left and the map in the bottom right. However, rather than communicate the whole SLAM information matrix between platforms, it is proposed to communicate only the map. It may well be possible to exploit the fact that the platform to platform cross information is zero[15] in information space to simplify the problem, however, this is an area which is still the subject of research.

4. SYSTEM ANALYSIS

Consider a scenario where two platforms are flying on a picture compilation mission. That is, each node on each platform is estimating the location of features observed on the ground and transmitting the information vector and matrix to other nodes as described in Section 2. Figure 3 details the sensors each platform is equipped with and the communication links and filters associated with each node. For the purpose of this example, it is assumed that there are ten targets observed as shown in Figure 6.

At time $k = 0$, all nodes on all platforms have zero information; that is, $\mathbf{Y}(0) = 0$ and $\mathbf{y}(0)$=0.

At the first time step, say $k = 1$, the camera on platform 1 has observed four features. Similarly, the radar on platform 1 has observed three features, two of which were also seen by the camera. These observations are handled as follows:

- Both the camera and the radar use their observations locally to update their map filters, which results in the camera node having a map of four and the radar having a map of three features.
- The camera outputs its information $\mathbf{Y}(k)$ and $\mathbf{y}(k)$ onto the bus.

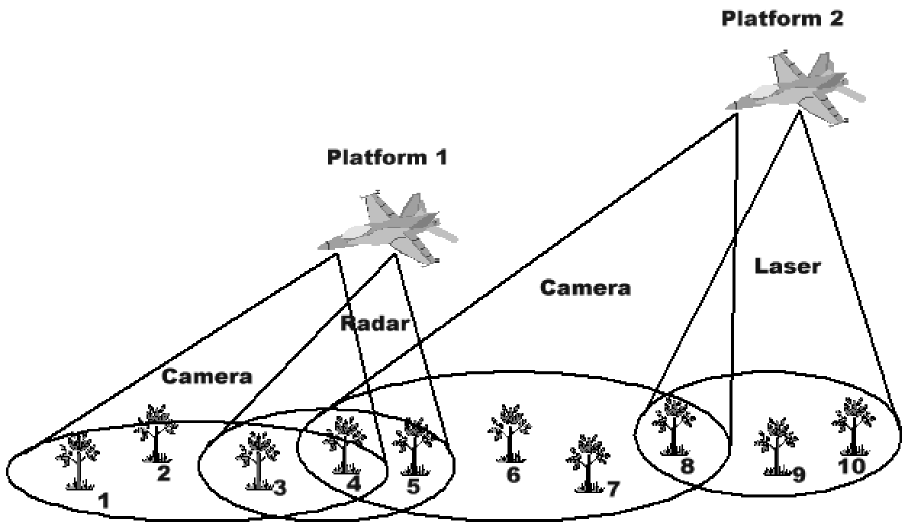


Figure 6. Features observed by each sensor.

- All nodes read in the information, which contains the information matrix and vectors for the four features. The channel filter determines the new information and the filter progress to the update stage. For the GPS, IMU and inter-platform nodes, the update then simply requires adding this information to their own estimate that is currently still zero. However, the radar node must perform data association on all four features in the map to determine if they match features in its own local map. This should result in the two features common to both nodes (features 3 and 4) being identified. The camera information for these two is then added accordingly, and two remaining new features are added to the local map by starting two new filters. At this time, all nodes on platform 1, except the radar, have a map of the cameras four features, while the radar has a map of five (it includes the extra feature only it has seen).
- The radar now outputs its own full map to the DDF bus as it has not yet transmitted. It will send the full information state $\mathbf{Y}(k)$ and $\mathbf{y}(k)$ for each of the five features in its map.
- Each node will receive this information and will use the data association algorithm to determine which features are located in their own local maps. The new information is then determined in the channel filter and added to the current estimate. Now, all nodes in platform 1 contain the complete map of five features.

Consider now platform 2. Over the same period of time as platform 1, platform 2 has been observing features as well. The laser on platform 2 observed three features which no platform has detected previously, while the camera saw five features. Of these five, one is common with the laser on platform 2 (feature 8), one is common with the radar on platform 1 (feature 5), one is common with both the camera and radar on platform 1 (feature 4) and the remaining two are completely new. Figure 6 illustrates the features observed by each sensor.

The DDF process on platform 2 then occurs exactly as the sequence on platform 1 such that after both the camera and laser have transmitted their information, all nodes on platform 2 have a map of seven features.

The system is now in a state where each platform has all information from its own onboard sensors, but the platforms have not yet communicated with each other. Note that there is zero map information in the channel filter between the platforms. This occurs as information has yet to travel though this link. The information is passed between the platforms in the following sequence.

- Platform 1 communicates the full information state $\mathbf{Y}(k)$ and $\mathbf{y}(k)$ for each feature in its map to platform 2.

- The inter-platform channel filter on platform 2 receives this information, performs data association to match common features and calculates the new information for each feature. As this channel filter is currently at zero, the step is trivial and the system sends the new information to the buffer. At this point, the buffer contains the full map of platform 2, so the buffer must therefore perform data association to match features of the new information from platform 1 with the current map of platform 2. This results in the two common features being detected (features 4 and 5 from Figure 6), which allows the map to be updated fully.
- Platform 2 now transmits the contents of the buffer to the DDF bus. That is, it sends the full $\mathbf{Y}(k)$ and $\mathbf{y}(k)$ containing all ten features to all nodes on platform 2.
- Each node performs the local data association and updates such that all nodes on platform 2 now have a complete map of all ten features.
- Platform 2 transmits its own full map back to platform 1. The inter-platform channel filter at platform 1 performs data association to match the features it has, then determines the new information. In determining the new information, it will subtract what it just sent to platform 2. This means that there will be no new information about features 1, 2 and 3, some new information about the two common features (features 4 and 5) and all new information about the five features platform 1 did not observe. This information is transmitted to the buffer, where it is fused with the current map at platform 1.

As a final point, consider now that the radar on platform 1 observes a feature that only the camera on platform 2 has previously seen (say, feature 6 from Figure 6). It makes this observation and sends its map of six features to the DDF bus on platform 1. Note that the information from platform 2 has not yet been transmitted to all nodes on platform 1. It is for this reason that there is a channel filter between the DDF bus and the buffer. All sensor nodes, including the buffer, update in the usual fashion using the information from the radar.

Platform 1 can now transmit the contents of the buffer to the DDF bus. This information contains all data from both platforms, including the latest observation made by the radar on platform 1. Each node on platform 1 receives this information and performs the usual data association and update. The final result is that both platforms have a full map of all ten features. Platform 1 has slightly more information about feature 6 at this time due to the latest observation by the radar, however, this will be sent to platform 2 at the next inter-platform broadcast.

This example demonstrates how the decentralised map building system functions. It is important to note that all nodes, including the GPS and IMU which do not observe any features, can still build a map.

5. CONCLUSIONS AND FUTURE WORK

Looking at the structure of the decentralised picture compilation and vehicle navigation problems gives valuable insights into the decentralised SLAM algorithm. In the picture compilation or map building problem, the communication and update of map information is trivial as the features are stationary. This then means that the prediction stage of the filter is omitted. The implication of this is that for map information, there is no delayed data problem. This extends directly to the SLAM problem if the platforms are passing the map between each other, and also means that the transmission rate can be as fast or as slow as necessary as new map data can be added at any time.

The issue of delayed data is one of the most important in decentralised data fusion. The current method of handling this is to feed the observation information through the standard prediction cycle to get it to the current time, and then perform the usual update cycle. However, while this method is consistent, it is suboptimal. For this reason, one of the key areas of research is aimed at determining a more accurate method of fusing old information without adding a large degree of complexity.

Future work in this project is aimed at decentralising the SLAM algorithm. Looking at the structure of the information form of the problem has revealed a number of interesting features, such as the platform to platform cross information being zero and the total map information being the sum of the maps at each individual platform. The major problem in this form is that it is not possible to directly extract the map from the information matrix due to the platform to map cross information being non-zero. It is not desirable to transmit the entire map and vehicle information as this would necessitate all platforms augmenting the states of all other platforms with their own. This also has significant implications on the amount of bandwidth required.

Decentralising map building and localisation algorithms for multiple flight vehicles offers enormous advantages for increasing the survivability of an individual platform and for increasing the chances of a group of aircraft being able to successfully complete a given task. The onboard decentralisation means that the aircraft will undergo a gradual performance degradation as nodes fail rather than a catastrophic failure, and the decentralisation across multiple vehicles increases the amount of information to the whole group. Indeed, should a node fail on a given platform, information from other platforms may well be sufficient to enable it to complete the task at hand. It is for this reason that decentralised data fusion, and decentralised SLAM in particular, are such hotly researched topics.

There is still a good deal more research required to solve the decentralised SLAM problem, however, this work in decentralised map building and vehicle navigation provides some of the necessary information. Further, a comprehensive and tested simulation environment has now been constructed that is ideal for development of decentralised algorithms such as SLAM.

ACKNOWLEDGMENTS

This work is funded by BAE SYSTEMS UK and BAE SYSTEMS Australia as part of the Autonomous Navigation and Sensing Experimental Research (ANSER) project. The first author is funded by BAE SYSTEMS Australia and would like to thank them for his PhD scholarship.

REFERENCES

1. S. Grime and H. Durrant-Whyte, "Data fusion in decentralized sensor networks," *Control Engineering Practice* **2, No 5**, pp. 849–863, 1994.
2. S. Utete, *A Network Manager for a Decentralized Sensing System*, Masters Thesis, University of Oxford, 1992.
3. S. Utete, *Network Management in Decentralized Sensing Systems*, PhD Thesis, University of Oxford, 1994.
4. D. Nicholson and R. Deaves, "Decentralised track fusion in dynamic networks," in *Signal and Data Processing of Small Targets*, *Proc. SPIE* **4048**, 2000.
5. R. Deaves, *The Management of Communications in Decentralised Bayesian Data Fusion Systems*, Ph.D Thesis, Bristol University, Dept of Electrical and Electronic Engineering, 1999.
6. R. Smith and P. Cheeseman, "On the representation of spatial uncertainty," *Int J. Robotics Research* **5, No 4**, pp. 56–68, 1987.
7. W. Rencken, "Concurrent localisation and map building for mobile robots using ultrasonic sensors," in *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2192–7, 1993.
8. J. Castellanos, J. Montiel, J. Neira, and J. Tardos, "Sensor influence in the performance of simultaneous mobile robot localization and map building," in *6th International Symposium on Experimental Robotics*, pp. 203–212, (Sydney), 1999.
9. M. Dissanayake, P. Newman, H. Durrant-Whyte, S. Clark, and M. Csorba, "An experimental and theoretical investigation into simultaneous localisation and map building," in *6th International Symposium on Experimental Robotics*, pp. 171–180, (Sydney), 1999.
10. J. Leonard and H. Durrant-Whyte, "Simultaneous map building and localisation for an autonomous mobile robot," in *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pp. 1442–1447, (New York), 1991.
11. J. Leonard, R. Carpenter, and H. Feder, "Stochastic mapping using forward looking sonar," in *FSR '99 International Conference on Field and Service Robotics*, pp. 69–74, (Pittsburg), 1999.
12. R. Deaves, D. Nicholson, D. Gough, L. Binns, P. Vangasse, and P. Greenway, "Multiple robot system for decentralized slam investigations," in *Sensor Fusion and Decentralised Control in Robotic Systems III*, 2000.
13. R. Deaves, "Covariance bounds for augmented state kalman filter application," *Electronic Letters, 11th November 1999* **35, No 23**, p. 2062, 1999.
14. P. Gibbens, G. Dissanayake, and H. Durrant-Whyte, "A closed form solution to the single degree of freedom simultaneous localisation and map building (slam) problem," in *IEEE Conference on Decision and Control*, 2000.
15. E. Nettleton, P. Gibbens, and H. Durrant-Whyte, "Closed form solutions to the multiple platform simultaneous localisation and map building (slam) problem," in *Sensor Fusion: Architectures, Algorithms, and Applications IV*, B. V. Dasarathy, ed., *Proc. SPIE* **4051**, pp. 428–437, 2000.