Toward a Fully Decentralized Architecture for Multi-Sensor Data Fusion

H. F. Durrant-Whyte, B.Y.S. Rao and H. Hu

Department of Engineering Science
University of Oxford

Abstract

The complexity of the multi-sensor data fusion problem has given rise to a considerable interest in the development of suitable architectures for multi-sensor systems. Most current data fusion architectures rely on having a central processor where all sensor information is combined. Such centralized architectures give rise to problems with communication and computational bottlenecks, and are not able to deal with sensor failure in a graceful manner. This paper describes a fully decentralized architecture for data fusion problems. This architecture takes the form of a network of sensor nodes, each with its own processing facility, which together do not require any central processor or any central communication facility. In this architecture, communication is performed locally and communication occurs between any two nodes. Such an architecture has many desirable properties including robustness to sensors failure, and flexibility to the addition or loss of one or more sensors. This architecture is appropriate for the class of Extended Kalman Filter (EKF) based geometric data fusion problems. The starting point for this architecture is an algorithm which allows the complete decentralization of the multi-sensor EKF equations amongst a number of sensing nodes. We describe this algorithm and show how it may be applied to a number of different data-fusion problems. An implementation of this algorithm to the problem of multi-camera, real-time, tracking of objects and people moving through a room is described.

1 Introduction

There has been a considerable amount of recent interest in the development of algorithms to process and fuse information obtained from a multi-sensor system. This interest arises from the need to combine information from a number of different sensors to obtain a full and complete description of the environment. For example, algorithms that use linear estimators have been employed in visual data fusion problems [1, 3, 10] and in range data integration [6, 11]. Knowledge-based or logical methods have also been used in these applications [15, 16, 10]. One overriding factor with all multi-sensor systems is that their inherent complexity requires considerable effort to be devoted to organization and architectural considerations [13]. Hierarchical organizations have proved popular [18] because of their ability to hide the complexity of low-level processing with higher level computational layers. Blackboard architectures have also been extensively employed in multi-sensor systems [20, 13, 7], because of an apparent ability to modularize sensor competences in terms of a number of communicating "agents".

The motivation behind all these organizations is to hide complexity and make each sensor function as modular as possible. Typically, in these organizations, the architecture comes first, and the algorithm is then "designed to fit". This often leads to severe problems in communication between different modules, and in the overall control of the system — who talks to whom, and when and about what. Such problems have resulted in these organizations being controlled by a central processing facility - either to take care of high-level fusion functions, or to serve as a communication medium. This goes against some of the original aims of these architectures; the sensing modules cease to become "autonomous agents" and the central processor becomes a communication and computational bottleneck.

This paper describes a fully decentralized architecture for multi-sensor data-fusion. This architecture has no central processing facility and no centralized communications medium, it does not require any ad hoc structure, such as a hierarchy, to be imposed on it, and it does not require a controller to organize it. The structure of this architecture resembles a network of sensor nodes (Fig. 1). Each sensor node has its own processing element, and its own communications facilities. It does not need to know what other sensors are in the network and it does not need to know what information they can provide. Communications can take place between any two connected sensor nodes, and they can independently assimilate each others information. The starting point for this architecture is an algorithm which permits the full decentralization of the Extended Kalman Filter (EKF) equations. We call this algorithm the Decentralized Kalman Filter (DKF). The DKF is an implementation of a multi-sensor EKF which has been divided up into modules, one associated with each sensor. In the EKF, a joint prediction is made about what each sensor is expected to observe and when these observations are made, they are centrally combined into a single composite description of the environment. In a DKF, each individual node makes its own predictions about what will be observed, and initially only integrates its own observation to obtain a local, partial estimate of the environment. An additional stage is then evoked in which these partial estimates are broadcast to other nodes where they are assimilated to provide the full environment estimate. The DKF algorithm has a number of features which make it more than just a mathematical curiosity:

- The estimates arrived at by each node are guaranteed to be exactly the same as those obtained by a fully centralized EKF (Our algorithm differs from the sub-optimal documention described in [5]). Thus, although processing is now distributed, there is no degradation in performance.
- Each sensor node deals with its own pre-processing and estimation problems; the failure of any one of the nodes will not result in a whole system failure. Thus, the decentralized sensing architecture is highly survivable and robust, being able to degrade gracefully in the face of sensor or processor failure.
- The amount of additional computation required is quite small and is certainly outweighed by the advantages of distributing computation.
- The communication overhead is also low, indeed it is actually less than is required for similar hierarchical organizations [14].

These advantages have led us to pursue the DKF as an algorithmic foundation for developing fully decentralized data-fusion architectures. Given the basic DKF algorithm, the next step is to determine just how applicable it is to data fusion problems in robotics. This question has a quite a simple (and in retrospect, obvious) answer; any data-fusion problem which can be expressed in terms of an EKF can be decomposed into an equivalent DKF. This encompasses a large number of different techniques including the visual data fusion methods of Faugeras and Ayache [1, 9], the linear estimators used by Chaiti et al. [6, 11] and Crowley [6], and the geometric data fusion methods described in [7]. This constitutes an important class of data fusion methods.

In this paper we describe the basic DKF algorithm, in both the linear and non-linear case, and demonstrate its application in two decentralized sensing systems. Section 2 introduces the algorithm by developing
the equations for a system in which each sensor takes purely linear observations of a common state. Section 3 describes an implementation of this linear DKF to the problem of real-time visual target tracking using four cameras. Section 4 extends these ideas to non-linear systems and in particular, sensors that observe different types of geometry. Section 5 describes some recent progress toward our first fully decentralized sensing network in which each sensor node comprises a simple infra-red sensor together with its own dedicated processor. In conclusion we discuss some limitations with our current algorithms and ways in which we intend to extend them.

2 The Decentralized Kalman Filter

The main stages in the operation of the DKF are shown diagrammatically in Figure 2. Each node is initialized with estimates of the state of the system together with an associated variance of that estimate. These initial values for state estimates and variances may be obtained in several ways and the methods used by the implementations given in this paper are outlined in later sections. The main loop begins with each node taking an observation (possibly asynchronously) from its sensor of the state of the environment. With this observation (and its associated variance) the node is able to compute conventional Kalman Filter equations to reach a partial estimate of the state based on only its own observation. Each node then broadcasts this estimate to the other nodes and receives information being broadcast to it. Lastly, each node computes assimilation equations to take into account the data it has just received; each node has locally computed a global estimate.

In this section, we derive the equations for describing the linear DKF; where the observations made are a linear function of the state to be observed. This derivation is intended to demonstrate the principles behind the DKF algorithm.

2.1 Centralized Kalman Filter Equations

We begin by reviewing the centralized Kalman Filter, both as a means of introducing notation and for later comparison with the decentralized filter. It is assumed that the reader is familiar with the basic Kalman Filter mechanism (the recent book by Bar-Shalom and Fortmann [2] is an excellent introduction to the Kalman Filter and its many extensions -- our notation derives from this book).

Consider a system represented by a state vector $x(k)$ at time $k$, whose dynamics are described by the state transition equation

$$x(k+1) = F(x(k))x(k) + G(k)w(k)$$

where $F(x)$ is the system model $G(k)$ is the noise model, and $w(k)$ is the input noise. A number of sensors are considered to take observations $a_i(k)$, $i = 1, \ldots, m$, of the state $x(k)$ according to the observation equation

$$z(k) = H_i(x(k)) + v(k)$$

where $z(k) = [z_1^T(k), \ldots, z_m^T(k)]^T$ is the stacked observation vector, $H_i$ is the observation model, and $v(k)$ is the observation noise. We assume $E[w(k)] = E[v(k)] = 0$, $E[w(k)^T] = Q(k)I$, $E[v(k)^T] = I$, and $E[w(k)v(k)^T] = 0$.

For all $k \geq 0$ we define

$$\hat{x}(k | j) = E[x(k) | z(1), \ldots, z(j)]$$

to be the estimate (expected value) of the state $x(k)$ at time $k$ based on observations taken up to time step $j$, and

$$P(k | j) = E[(x(k) - \hat{x}(k | j))(x(k) - \hat{x}(k | j))^T]$$

to be the mean squared error in this estimate.

For a system which is described by Equation 1 and which is being observed according to Equation 2, the Kalman Filter provides a recursive solution for the estimate $\hat{x}(k+1 | k+1)$ of the state $x(k+1)$ in terms of the estimate $\hat{x}(k | k)$ and the new observation $z(k+1)$. This solution has two stages: prediction and update.

**Prediction:**

$$\hat{x}(k+1 | k) = F(x(k))\hat{x}(k | k)$$

$$P(k+1 | k) = F(x(k))P(k | k)F(x(k)) + G(k)Q(k)G(k)^T$$

**Update:**

$$\hat{x}(k+1 | k+1) = \hat{x}(k+1 | k) + W(k+1)(z(k+1) - H(k+1)\hat{x}(k+1 | k))$$

$$P^{-1}(k+1 | k+1) = P^{-1}(k+1 | k) + H(k+1)R_{k+1}^{-1}(k+1)H(k+1)$$

where

$$W(k+1) = P(k+1 | k)H(k+1)R_{k+1}^{-1}(k+1)$$

is the Kalman gain matrix. Note that the information form of the variance update has been employed (Equation 6).

2.2 Decentralized Nodal Equations

The partitioning of these centralized equations follows the work of Keshmiri [14] in which a hierarchical formulation of the Kalman Filter is derived. We first unstack the observation vector $z(k)$ into $m$ subvectors of dimension $m_i$ corresponding to the observations made by each individual sensor

$$z(k) = [z_1^T(k), \ldots, z_m^T(k)]^T$$

and partition the observation matrix into submatrices corresponding to these observations

$$H(k) = [H_1^T(k), \ldots, H_m^T(k)]^T.$$  

The noise vector is also partitioned

$$v(k) = [v_1^T(k), \ldots, v_m^T(k)]^T$$

and it is assumed that these parts are uncorrelated

$$E[v(k)v^T(k)] = \text{blockdiag}(R_1(k), \ldots, R_m(k))$$

The system model $F(k)$ and the state $x(k)$ cannot, in general, be partitioned in the same way as the observations. However, it is often the case that each individual sensing node can have its own model of the system. This may occur in a number of ways: when different sensors do indeed observe different linear combinations of the state (different viewpoints for example), or when coupling between states is explicitly accounted for.

Letting $F_i(k)$ be the sensors local system model, and letting $x_i(k)$ be the states associated with this system, the local state transition and observation equations can be written as:

$$x_i(k+1) = F_i(k)x_i(k) + G_i(k)w_i(k)$$

$$z_i(k) = H_i(k)z_i(k) + v_i(k).$$

With these transition and observation equations, each node $i$ starts by implementing its own local form of Equations 3-7.

**Prediction:**

$$\hat{x}_i(k+1 | k) = F_i(k)\hat{x}_i(k | k)$$

$$P_i(k+1 | k) = F_i(k)P_i(k | k)F_i(k) + G_i(k)Q_i(k)G_i(k)^T$$

**Update:**

$$\hat{x}_i(k+1 | k+1) = \hat{x}_i(k+1 | k) + W_i(k+1)(z_i(k+1) - H_i(k+1)\hat{x}_i(k+1 | k))$$

$$P_i^{-1}(k+1 | k+1) = P_i^{-1}(k+1 | k) + H_i(k+1)R_i^{-1}(k+1)H_i(k+1)$$

where

$$W_i(k+1) = P_i(k+1 | k)H_i(k+1)R_i^{-1}(k+1)$$

and the tilde denotes a partial estimate based only on the $i$th node's own observation.

2.3 Derivation of Assimilation Equations

After each node has computed its partial estimate $\bar{x}_i(k+1 | k+1)$, it must communicate information to the other nodes and be prepared to assimilate the information it receives. To determine what information needs to be communicated, we first derive the assimilation equations.

**Assimilation of Variance**

The partitioning of the observation model and the observation noise allows us to write
Now, rearranging Equations 6 and 13 gives:

$$H_t^T(k-1)R_t^{-1}(k)H_t(k) = \sum_{i=1}^{m} H_t^T(k)R_t^{-1}(k)H_t(k). \quad (15)$$

By substituting these into Equation 15

$$P_t^{-1}(k+1 | k+1) = P_t^{-1}(k+1 | k) + \sum_{j=1}^{m} [P_t^{-1}(k+1 | k+1) - P_t^{-1}(k+1 | k)]$$

and placing this assimilation equation at each node (decentralizing) we obtain:

$$P_t^{-1}(k+1 | k+1) = P_t^{-1}(k+1 | k) + \sum_{j=1}^{m} [P_t^{-1}(k+1 | k+1) - P_t^{-1}(k+1 | k)]. \quad (18)$$

Assimilation of State

Again, from the partition of the observation vector and observation model, we have

$$H_t^T(k)R_t^{-1}(k)u_t(k) = \sum_{i=1}^{m} H_t^T(k)R_t^{-1}(k)u_t(k). \quad (19)$$

Premultiplying Equation 16 by $P_t(k+1 | k+1)$ and using the definition of the Kalman gain from Equation 17 gives

$$I - W_t(k+1 | k+1)H_t(k+1) = P_t(k+1 | k+1)P_t^{-1}(k+1 | k). \quad (20)$$

Premultiplying Equation 5 by $P_t^{-1}(k+1 | k)$ and employing Equation 20 and rearranging gives

$$P_t^{-1}(k+1 | k+1)u_t(k+1) = P_t^{-1}(k+1 | k+1)u_t(k+1 | k+1) - P_t^{-1}(k+1 | k)\hat{x}(k+1 | k). \quad (21)$$

Similarly, premultiplying Equation 17 by $\hat{P}_t^{-1}(k+1 | k+1)$ and using the definition of the Kalman gain from Equation 14 gives

$$I - W_t^{-1}(k+1 | k+1)H_t(k+1) = \hat{P}_t(k+1 | k+1)\hat{P}_t^{-1}(k+1 | k). \quad (22)$$

then premultiplying Equation 12 by $\hat{P}_t^{-1}(k+1 | k+1)$, employing Equation 22 and rearranging gives

$$H_t^T(k+1)\hat{R}_t^{-1}(k+1)\hat{x}_t(k+1) = \hat{P}_t^{-1}(k+1 | k+1)\hat{x}_t(k+1 | k+1) - \hat{P}_t^{-1}(k+1 | k)\hat{x}_t(k+1 | k). \quad (23)$$

Substituting Equations 21 and 23 into Equation 19

$$\hat{x}(k+1 | k+1) = P_t^{-1}(k+1 | k+1)\hat{x}_t(k+1 | k+1)$$

and placing this assimilation equation at each node (decentralizing) results in:

$$\hat{x}_t(k+1 | k+1) = \hat{P}_t^{-1}(k+1 | k+1)\hat{x}_t(k+1 | k+1) + \sum_{j=1}^{n} \left[ P_t^{-1}(k+1 | k)\hat{x}_j(k+1 | k) - \hat{P}_t^{-1}(k+1 | k)\hat{x}_j(k+1 | k) \right].$$

and

$$\hat{x}_t(k+1 | k+1) = \hat{P}_t^{-1}(k+1 | k+1)\hat{x}_t(k+1 | k+1) + \sum_{j=1}^{n} \left[ P_t^{-1}(k+1 | k)\hat{x}_j(k+1 | k) - \hat{P}_t^{-1}(k+1 | k)\hat{x}_j(k+1 | k) \right].$$

$$\hat{x}_t(k+1 | k+1) = \hat{P}_t^{-1}(k+1 | k+1)\hat{x}_t(k+1 | k+1) + \sum_{j=1}^{n} \left[ P_t^{-1}(k+1 | k)\hat{x}_j(k+1 | k) - \hat{P}_t^{-1}(k+1 | k)\hat{x}_j(k+1 | k) \right].$$

$$\hat{x}_t(k+1 | k+1) = \hat{P}_t^{-1}(k+1 | k+1)\hat{x}_t(k+1 | k+1) + \sum_{j=1}^{n} \left[ P_t^{-1}(k+1 | k)\hat{x}_j(k+1 | k) - \hat{P}_t^{-1}(k+1 | k)\hat{x}_j(k+1 | k) \right].$$

$$\hat{x}_t(k+1 | k+1) = \hat{P}_t^{-1}(k+1 | k+1)\hat{x}_t(k+1 | k+1) + \sum_{j=1}^{n} \left[ P_t^{-1}(k+1 | k)\hat{x}_j(k+1 | k) - \hat{P}_t^{-1}(k+1 | k)\hat{x}_j(k+1 | k) \right].$$

2.4 Communication and Assimilation

To summarize, each node makes an observation according to Equation 9 and then computes a partial estimate using Equations 10–14. The nodes then communicate with each other (see below) and assimilate received information according to:

$$\hat{x}_t(k+1 | k+1) = \hat{P}_t^{-1}(k+1 | k+1)\hat{x}_t(k+1 | k+1) + \sum_{j=1}^{n} \left[ P_t^{-1}(k+1 | k)\hat{x}_j(k+1 | k) - \hat{P}_t^{-1}(k+1 | k)\hat{x}_j(k+1 | k) \right].$$

The terms state error info and variance error info are the two terms that need to be transmitted by each node to each other node. It is also worth emphasizing that this DSKF is mathematically equivalent to the centralized KF. Thus the local estimates obtained by each node will be identical to those obtained from a centralized scheme.

2.5 Discussion

Each node needs to compute the prediction and update Equations 10–14, followed by the assimilation Equations 20 and 21. Equations 10–14 are no more than a "parallelized" form of Equations 3–7. Clearly for these equations, each node will not need to perform as much computation as the centralized processor would simply because the dimensions of the observation vectors and observation model are smaller. However, the sum of all computations performed by the nodes will exceed the amount of processing done by a central processor. Thus, although the speed-up gained by parallelizing will be large, but will not be linear in the number of nodes. The additional computation required for assimilation is almost negligible, representing a summation of vectors, a summation of matrices and one matrix-vector multiply. Consequently it is purely the parallelisation which gives this decentralized filter computational advantage over its centralized equivalent.

Each node must communicate one vector (state error info) and one matrix (variance error info) to each other node. Assuming there are $m$ sensing nodes and each node estimates a full state vector of dimension $n$, then a total of $(n^2 + n)(m - 1)$ numbers need be communicated in each estimation cycle. This is obviously more than if each sensor just communicated observations directly to a central fusion center, but less than if an equivalent hierarchical decentralisation scheme were used $(3mn^2 + n)$ [14]. There is a problem with designing a convenient communication topology, which nodes should be talking to each other. With four nodes, a fully-connected topology is possible with three communication channels per processor, which is the maximum allowable with the Tranputer technology we have chosen for our implementations. With any more nodes, a compromise must be found whereby alternative routes can be found through intermediate switching nodes. We have so far not investigated these alternative communication topologies. In summary, the DSKF significantly reduces the computation any one processor needs to undertake whilst incurring only a relatively small communication overhead.

Most standard extensions to the centralized Kalman filter can also be applied to the DSKF; multi-target tracking, or clutter models, for example [2]. This is because the basic prediction and estimation process is encapsulated in Equations 10–14, which are no more than local forms of the usual centralized equations, assimilation in fact only a means of combining estimates that have already been obtained.

3 Implementing the Linear DSKF

Not many sensing problems are linear in the strict sense. However they can often be made to look linear by suitable processing of sensor data and by suitable definition of state. The implementation described in this section falls into this category. This implementation involves the use of four widely spaced cameras to track, in real-time, objects and people moving through a room [1]. The sensing system used consists of four CCD cameras mounted at the top corners of a room (10mx10m in size), and pointing at the approximate center of the floor. Special purpose hardware (DataCube) was used to capture and preprocess images at frame-rates. In addition to this, the cameras each have three 2800 Transputers associated with them. Two of these processors are used to extract targets from the processed image, the third is employed to run the local nodes DSKF.

The image preprocessing performed by each camera system in each time frame was as follows:

- The current image for the time frame is differenced with the previous image and the resulting image "deghosted" to reveal the new
events.
- Since the resulting image usually reveals events that suffer from 'shatter' (ie. several pixels around a small area all associated with the same event) these shattered events are clumped together about their centres of gravity to give cleaner distinct events.
- Knowing the camera calibration the events are then converted from (x, y) pixel coordinates to (z, y) floor coordinates using a simple perspective projection. Within this is the implicit assumption that all the events we are interested in occur on the floor. In this case the assumption is valid and makes event detection far simpler.

Processing of images in this way occurs at about 20Hz. The result of this pre-processing stage are primitive events in (two-dimensional) (x, y) coordinates. It is these events that are treated as observations - consequently the linear DKF may be applied. The events found from this pre-processing stage are passed to the processor running the DKF algorithm-one for each camera.

The DKF algorithm uses the same single-target third-order model of motion dynamics $x = (x_n \dot{x}_n \ddot{x}_n)^T$ for each node. Each filter is initialised by batch processing the first three events to obtain initial location and velocity estimates. Local predictions are then calculated using Equations 10 and 11. Each DKF then accepts a new event and calculates a local estimate of the state according to Equations 12 and 13. If more than one event is found, then the event closest to the prediction is used as the observation. If no events are found, then the prediction becomes the estimate. Each node then communicates its estimate to the other nodes and assimilates those estimates that are communicated to it according to Equations 26 and 27. Should the new event deviate sufficiently from the predicted path, the filter will reinitialize itself and calculate a new state based on the new event position and its estimated velocity.

Figure 4 shows the output of the system. The square panel represents the floor of the room. The small crosses and squares show the valid events detected by each of the four cameras. The large triangle shows the current global estimate of the position of the object, (taken from any one of the cameras since they all arrive at the same global consensus) and the circle shows the area in which the next event is predicted to fall. Should the event actually fall outside this area the filter will reinitialize.

The filter can operate at a maximum speed of 20ms per iteration for the combined estimation and assimilation stages. In practice, the speed is limited by the time it takes to preprocess the images (20Hz). The system can reliably track the motion of objects (in this example, a toy robot moving in circles), as well as people moving through the room. Although the results shown are for the single-target case we have now extended the filter to cope with multiple target tracking in clutter.

### 4 Decentralized Geometric Data Fusion

Not many data-fusion problems are as conveniently linear as the one just described. To apply the DKF in more general data-fusion problems, in which non-trivial geometries are observed and integrated, we need to develop the corresponding non-linear filter. Ultimately the type of problems that we would like to treat with the DKF are the class of geometric data-fusion methods. These techniques treat sensors as "geometric extractors", sources of uncertain, partial, geometric information about the environment [8]. The geometric information obtained from these sensors is combined using any number of different types of linear estimator, including the Extended Kalman Filter. These methods are powerful, if limited class of data-fusion techniques, which are being increasingly applied in robotics problems [1, 9, 6, 11, 7, 12].

In this section, we will deal with two types of non-linearities; those associated with non-linear observations and state transitions, and those associated with non-linear coordinate transforms between different sensors.

#### 4.1 The Non-Linear DKF

The extension of the linear DKF to deal with non-linear state transitions and non-linear observations is quite straightforward. This is because these non-linearities only affect the local estimation equations and not the assimilation equations. As the local estimation equations (Equations 10-14) are no more than nodal equivalents of the well-known central estimation equations (Equations 3-7) the inclusion of non-linear terms in the DKF follows exactly the same pattern as for the centralized EKF. Thus if the nodal transition and observation equations are described by

$$x_i(k+1) = F_i(x_i(k), k) + G_i(k)w_i(k)$$

$$x_i(k) = H_i(x_i(k), k) + v_i(k)$$

then the nodal estimation equations are given by

**Prediction:**

$$\hat{x}_i(k+1 | k) = F_i(\hat{x}_i(k), k)$$

$$P_i(k+1 | k) = (\nabla F_i(k))P_i(k | k)(\nabla F_i(k))^T + G_iQ_iG_i^T$$

**Update:**

$$\tilde{x}_i(k+1 | k+1) = \hat{x}_i(k+1 | k) + W_i(k+1 | k)[x_i(k+1) - \nabla H_i(k+1)\hat{x}_i(k+1 | k)]$$

$$P_i^{-1}(k+1 | k+1) = P_i^{-1}(k+1 | k) + W_i(k+1 | k)$$

$$\nabla H_i(k+1)\tilde{x}_i(k+1)$$

where

$$W_i(k+1) = \hat{P}_i(k+1 | k)\nabla H_i(k+1)^T\tilde{P}_i^{-1}(k+1)$$

and

$$\nabla H_i(k+1)\hat{P}_i(k+1 | k)$$

are the Jacobians of $F_i(k)$ and $H_i(k+1)$ evaluated at $\hat{x}_i(k | k)$ and $\hat{x}_i(k+1 | k)$ respectively. The terms that need to be communicated are the state error info and varance error info in the assimilation equations (Equations 26 and 27) remain unchanged.

The second non-linear case concerns the transmission of geometric information between geographically distributed sensor systems. We are particularly interested in this problem because it has direct application to the decentralized sensing network described in Section 5.

In this case, the non-linearities do affect the assimilation equations as estimates in one sensor node's coordinate system are transformed into other nodes viewpoints. However, the nodal estimation equations (Equations 10-14 in the linear case and Equations 30-34 in the non-linear case) are not affected by this transformation and can take place in any suitable view-centered reference frame.

The assimilation equations are altered as follows [8]: Let

$$\hat{z}_i(k) = h_i(\hat{x}_i(k))$$

describe (the geometric) transformation of the jth node's state vector as viewed from its own coordinate system, into an equivalent state vector as viewed from the jth node coordinate system. Estimates of state and variance (at time k given observations up to time l) in different coordinate systems can then be transformed using linearization arguments as:

$$\tilde{z}_j(k | l) = h_i^T(\hat{z}_i(k | l))$$

and

$$\hat{P}_j(k | l) = (\frac{\partial h}{\partial x_j} \hat{P}_i(k | l) \frac{\partial h}{\partial x_j})^T$$

These transformations result in the modified assimilation Equations

$$\tilde{z}_j(k+1 | k+1) = \hat{P}_i(k+1 | k+1)\tilde{z}_j(k+1 | k+1) + \sum_{j=1}^{\infty} \alpha_j$$

$$\hat{P}_j^{-1}(k+1 | k+1) = \hat{P}_i^{-1}(k+1 | k) + \sum_{j=1}^{\infty} \beta_j$$

where

$$\alpha_j = (\hat{P}_j^{-1}(k+1 | k+1))^T\tilde{z}_j(k+1 | k+1) - \hat{P}_j^{-1}(k+1 | k)\tilde{z}_j(k+1 | k)$$

$$\beta_j = (\hat{P}_j^{-1}(k+1 | k+1))^T\tilde{z}_j(k+1 | k)$$

are the state error info and varance error info of the jth sensor node transformed into the jth nodes view frame.

In this case, each node computes first it's own local estimate based on its own observation according to Equations 10-14 or Equations 30-34 depending on whether state transitions and observations are linear or non-linear respectively. Then each node computes an $\alpha_j$ and $\beta_j$ for each node that intends to communicate with. This requires a prior knowledge of how pairs of nodes are placed in relation to each other and so some location information must also be communicated. These two terms are communicated, and assimilated by each node according to Equations 26 and 39.

This work was conducted in collaboration with British Aerospace Sensor Research Center.
4.3 Simulation of Non-Linear Assimilation

The non-linear assimilation process has been investigated in simulation, on a transputer system running under the control of a Sun workstation. The simulation is of the case where a target is tracked by observers able to take angle-only measurements. Complete state recovery can only be achieved by communicating information between different observers. We are currently implementing this algorithm on the sensor net described in Section 5.

5 A Decentralized Sensing Network

We are in the process of building a fully decentralized sensor network. This network will ultimately consist of six sensor nodes. Our goals in building this sensor network are:

- To physically realize a geographically distributed fully decentralized data fusion architecture.
- To demonstrate the robustness, survivability and graceful degradation of this architecture by allowing new nodes to be added and failed nodes to be removed from the network without catastrophic consequences.
- To provide a platform for further work in fully decentralized sensing architectures.

A schematic describing one of the sensor nodes is shown in Figure 5 and a photograph of the node is shown in Figure 6. Each node consists of a steerable active infra-red ranging sensor, a TRAM2/8800 Transputer with 128K of local memory, two CO11 transputer-link to parallel I/O lines, and two transputer-to-transputer (long distance) communication channels. All the electronics associated with the sensor and motor are also incorporated within the node.

The infra-red sensor is amplitude-based; a modulated infra-red signal is emitted from a power LED and the returned reflections are focussed by a parabolic to an infra-red detector. The returning signals are passed through an analog matched-filter which extracts the signal coincident with the modulations of the transmitted pulse. The amplitude of this returned signal is then used as a continuous 8-bit measure of range. It is important to realize that this sensor is in fact a reflectance measuring device; the range values obtained are not a good indication of the real range to a target because the returned amplitude depends on the re-reflectance properties of the target. Although absolute range is difficult to determine with this sensor, changes in reflectance can be determined with very high angular resolution. These reflectance changes correspond to either range discontinuities or changing surface albedo. Thus this sensor can be used to detect the heading to a target which is distinguished from its background by either range or reflectance. The reason for using this device rather than a more conventional ranging technology, such as ultra-sonics, is that we can achieve high data-acquisition rates (typically 200Hz), comparable to the speed at which the DKF can be run on the Transputer processors used (25ns).

6 Conclusions

We have described an algorithm that allows a class of geometric data fusion problems to be fully distributed amongst a network of sensing nodes. This algorithm provides a means of developing a fully decentralized architecture for data-fusion, which eliminates computational bottlenecks, and is able to degrade gracefully in the face of sensor and processor failure. An implementation of the linear case of this algorithm has been described which enables information obtained from four cameras to be fused in real time and used to track objects moving across a room. The non-linear case has been demonstrated in simulation and a fully-decentralized sensing network has been described which is intended to implement this simulation.

The DKF algorithm is only applicable to problems that can be reduced to an equivalent Extended Kalman Filter. Although this includes a wide variety of data fusion methods, including geometric data-fusion techniques, it excludes many other methods. In particular it precludes the logical-sensing techniques popularized by Henderson [15, 16], and the image-based methods used by Aggarwal [17].

Our immediate research goals are to complete the work on the multi-target implementation of the visual event tracking system and to complete a 3-node implementation of the decentralized sensor network. The sensor net that we are currently building will not only be use to test and demonstrate the DKF algorithm, but will also as a platform for investigating other aspects of fully decentralized data-fusion architectures. In particular, we are interested in developing methods to increase the amount of "local intelligence" that can be embedded in each sensing node. This includes not just stochastic models of the sensor and the information it obtains, but also logical formalisms able to describe some of the physics of the device. We feel that a transputer-based architecture is a natural choice for investigating these problems.

References

Figure 3: Example images from each of the four cameras. Each row shows the same time frame view from camera one at the left to camera four at the right. Each next row shows the subsequent time frames' views. The event to be detected is the movement of the small white box.

Figure 2: The DKF Algorithm

Figure 5: A schematic describing the main components of the sensor node.

Figure 4: The output of the system. The small crosses and squares are readings from each camera. The large triangle is the current global estimate of the position of the object. The circle is the uncertainty region within which the filter expects the next event to fall.