# A Two-Stage Approach to Multi-Sensor Temporal Data Fusion

D.Hutber and Z.Zhang
INRIA,
2004 Route des Lucioles,
B.P.93,
06902 Sophia Antipolis Cedex,
France.
dhutber@sophia.inria.fr, zzhang@sophia.inria.fr

**Abstract**

This paper proposes a two-stage architecture for multi-sensor temporal data fusion. The first stage uses extended Kalman filters to track tokens seen by each sensor, and the second stage links the tokens corresponding to the same real-world event. Two pairs of strategies are presented relating to the initial data association between tokens and filters, together with decision rules for switching between them. One pair is for the 'bootstrap' phase and the 'continuous' phase for an event, and the other distinguishes between a complex tracking task and a simpler one. The application of the techniques to a driver's assistant system is described.

## 1 Introduction

The problem of multi-sensor data fusion has been extensively formulated by others [4, 3] without the explicit use of time. The work described here concentrates on the subset of possible applications for which a temporal model of the dynamics of the 'world' is available. The problem is stated in terms of a (non-linear) state-space model, for which the method of Extended Kalman Filters is chosen to estimate the state of the world from limited observations. Typical applications of this approach are robot navigation and surveillance.

## 2 Problem Formulation

In this section the formalism being used to describe the proposed architecture and the principal problems involved are presented.

### 2.1 Extended Kalman Filters

An uncontrolled non-linear system with state variables $\mathbf{x}(t)$ of dimension $n$ and observation variables $\mathbf{y}(t)$ of dimension $m$ is described by the following equations:

BMVC 1994 doi:10.5244/C.8.71

$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + \mu_k$
$\mathbf{y}_k = C_k \mathbf{x}_k + \nu_k$

where $A(\mathbf{x}_k)$ is the linearised state transition matrix and $C(\mathbf{x}_k)$ is the linearised observation matrix the value $\mathbf{x}_k$, $\mu_k$ is the state noise and $\nu_k$ is the observation noise. The covariance of the state variables is $P$.

In the linear case, provided the additive noise is Gaussian, a Kalman filter provides an optimal way of estimating the state variables $\mathbf{x}$ given the observations $\mathbf{y}$ and known values of the state noise and observation noise.

The update of the estimated state variables $\mathbf{x}$ is carried out in two stages, a prediction (denoted by the suffix $k/k-1$) followed by a correction (denoted by $k/k$):

$$\mathbf{x}_{k/k-1} = A_k \mathbf{x}_{k-1/k-1} \tag{1}$$

$$P_{k/k-1} = A_k^T P_{k-1/k-1} A_k + Q_k \tag{2}$$

and

$$\mathbf{x}_{k/k} = \mathbf{x}_{k/k-1} + P_{k/k-1} C_k^T R_k^{-1} (\mathbf{y}_k - C_k \mathbf{x}_{k/k-1}) \tag{3}$$

$$P_{k/k}^{-1} = P_{k/k-1}^{-1} + C_k^T R_k^{-1} C_k \tag{4}$$

where $R$ and $Q$ are the observation noise and the model noise covariance matrices.

## 2.2 Asynchronous and Non-Monotonic Observations

In order to motivate the attention given to this particular aspect of the multi-sensor temporal data fusion problem, consider the case of a mobile robot moving through the environment, equipped with multiple sensors. The raw data acquired by these sensors will usually need some degree of processing prior to the fusion process, and this processing time will usually be data-dependent. In addition, active strategies may be employed by the robot to acquire data from those parts of the environment that are more complex, more relevant or less accessible. This work assumes that the time at which data is collected by the various sensors is known, and the time of the last observation for each sensor $i$ is denoted by $t_i$.

As a result of these constraints, the data arriving from the sensors to be fused will in general be asynchronous, and due to the variable processing time involved may well be 'out of order' (in the sense of being non-monotonic in time, clearly!). Thus any architecture designed to cope with this type of data must be able to handle asynchronous and non-monotonic data sequences. We now re-formulate the problem for the asynchronous constraint, the monotonicity being taken care of by the choice of architecture presented in Section3

The equations (1)–(4) are for the discrete case, with the index $k$ running in our case over time but more generally over the number of observations made. Since we are interested in using this formulation for the temporal evolution of a physical system, we first change from the use of constant discrete time intervals between observations of $\delta t = constant$ to the use of a variable time interval. In the general case this makes the matrix $A = A(\mathbf{x}, \delta t)$.

The other change that is made concerns the evolution of the covariance matrix $P$ in time. We estimate the noise in our model of the dynamics of the system $Q_k$ by an expected error in the state vector, typically in the higher order derivatives.

In the application that will be described more in Section 5, a state vector comprising position and velocity is used, together with an expected acceleration **a** that is modelled as noise. Thus the matrix $Q$ becomes:

$$Q(\delta t) = diag \begin{pmatrix} 1/2\mathbf{a}\delta t^2 \\ \mathbf{a}\delta t \end{pmatrix}$$

Here **a** defines the (assumed known) physical limits of the objects being tracked.

### 2.3 Data Association

The problem of data association arises when data corresponding to several 'real-world events' is sensed by the sensors, when erroneous detections are made (false alarms), and when events are not detected (missed targets). Thus in our formulation there may be several perceived events that are modelled by separate E.K.F.'s, which may or may not correspond to real-world events. More will be said about this in Section3

Within the E.K.F. formulation of this problem, we can use the maximum likelihood estimate to match any of the $\#Obs_i(t)$ observations with timestamp $t$ from sensor $i$ $y_{ij}^{(t)}, j = 1, 2, .., \#Obs_i(t)$. The observation which satisfies this is the one which is closest in the Mahalanobis distance sense from the estimated state variables of the perceived events. The closest observation must also satisfy the Mahalanobis distance test to decide whether it is more likely to have arisen from another random event.

The Mahalanobis distance test between two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ of dimension $m$ who have covariance matrices $P_1$ and $P_2$ is given by:

$$(\mathbf{x}_1 - \mathbf{x}_2)(P_1 + P_2)^{-1}(\mathbf{x}_1 - \mathbf{x}_2)^T \leq \kappa$$

where $\kappa$ is a threshold chosen from a $\chi^2$ distribution table with $m$ degrees of freedom.

The maximum likelihood estimate will not, however, always match up the 'correct' observation to an E.K.F., especially when the covariance of the vectors is large. This is generally the case when not many observations of the event have yet been made. Various strategies are possible such as best first association, or probabilistic methods [2] which maintain multiple hypotheses with probabilities attached to them.

For this reason, the strategy that will be presented in Section 4 differs according to whether 'few' observations of an event have been made (the bootstrap phase) or whether 'many' observations have been made (the continuous phase). A decision rule for switching between the two will also be presented.

## 3 A Two-Stage Architecture

This section presents an architecture to handle multi-sensor temporal data fusion in an environment where there are potentially many events. It is in two stages,

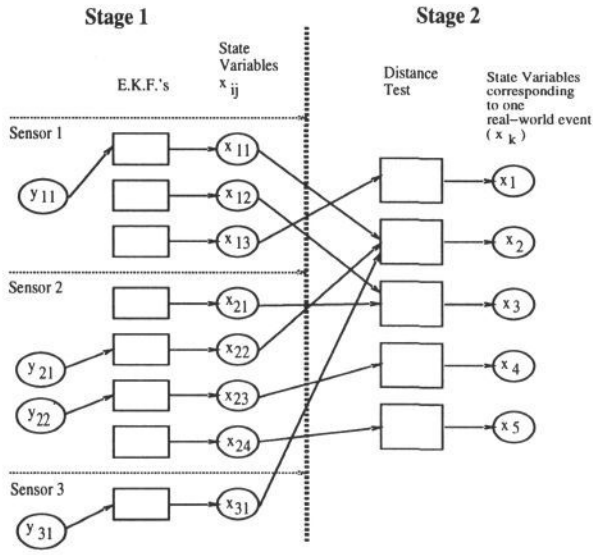


Figure 1: The Two-Stage Filter Method

the first of which consists of temporally fusing the data from one sensor, and the second of linking up all the sensors to produce a single output. It takes advantage of the (generally true) fact that a single sensor will yield a series of observations that are monotonic in time, which will in practice be true when the logical sensor is implemented on a single CPU or in a pipeline architecture. Thus the problem has effectively been sidestepped by using the monotonic subsets of observations from each sensor, hence a conventional (asynchronous) filter can be used for each subset. If the data is not monotonic, another architecture using the propagation of observations forward in time has been proposed [5].

Thus (see Figure 1), for each observation $\mathbf{y}_{ij}$ of a perceived event $(j)$ in each sensor $(i)$, in the first stage the observation is first associated with an E.K.F., which calculates a set of state variables $\mathbf{x}_{ij}$. In the example there are three logical sensors, two with one observation and the other with two observations which arrive asynchronously. The data association is done using a strategy that depends on whether the observation is associated with an existing filter (and hence is in the continuous phase) or whether a potentially new event has been sensed (bootstrap phase). An E.K.F. is created *only* when a large enough set of observations have been associated together to make the probability that the perceived event corresponds to a real event large enough. This is described in more detail in the next section.

The second stage is to combine the $\mathbf{x}_{ij}$ corresponding to the same event in different sensors to give a better estimate of the state variables $\mathbf{x}_k$. This is done by first asynchronously propagating each of the filter state variables up to a time T, where $T \geq t_i, \forall i$, using the prediction stage of the E.K.F. equations.

Now, at any given time there will be two types of filter variables $\mathbf{x}_{ij}$. The first are those that at a previous time have already been associated with event $k$. These continue to be associated with the event, and there is currently no retrying of possible associations. The second type are those that have not yet been

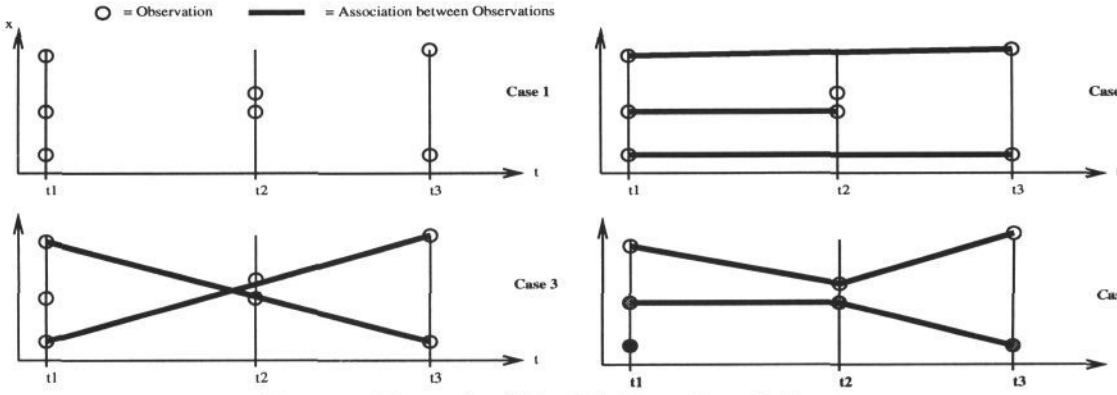


Figure 2: Example of Possible Data Associations

allocated to an event. Remember that a filter is created only when it has a good chance of corresponding to a real event. We also use the constraint that at most one filter from each sensor can correspond to the event. As a result, all the existing filters $\mathbf{x}_{lj}$ that have no link to a filter from sensor $i$ ($l \neq i$) are tested using the generalised Mahalanobis test:

$$(\mathbf{x}_{lj} - \mathbf{x}_{ij})^T (P_{lj}(T) + P_{ij}(T))^{-1} (\mathbf{x}_{lj} - \mathbf{x}_{ij}) \leq \kappa$$

where the dimension of the test is now equal to $n$.

The estimated state vector and covariance for each event $\mathbf{x}_k (= \mathbf{x}_j)$ can now be calculated from the state and covariance of the linked filters. This may be done optimally using the formula:

$$P_j^{-1} \mathbf{x}_j = \sum_{i=1}^{N} \mathbf{x}_{ij} P_{ij}^{-1}$$

where:

$$P_j^{-1} = \sum_{i=1}^{N} (P_{ij})^{-1}$$

- a formula which can be derived directly from the E.K.F. equations. In the example there are five resulting events.

# 4 A Strategy for Data Association

The choice of a strategy for data association is still somewhat of an art, with parameters that are set on a trial and error basis. Here, given that the context of the two-stage method is targeted at applications having a time evolution model, we try to use a strategy that is determined as much as possible by the data themselves, and by measurable physical characteristics of the system that is being modelled.

Consider the following example (Figure 2):

The figure represents a simple 1D case of data association, with time running horizontally. Thus in the example there are three observations at time $t_i(1)$, two

at time $t_i(2)$ and two at time $t_i(3)$ In the first case, we claim it is impossible, or at least very unwise, to try to associate any of the tokens together, in the absence of any further information.

In the second case, if it was known that the tokens' states could change by only a small amount between the times shown, the data association shown by the lines would become more likely.

The association shown for the third case would be more likely if we knew that false alarms and missed targets were rare.

Finally, the fourth case shows yet another association if we were somehow able to 'colour-code' the observations.

The purpose of this example is firstly to show that the problem of data association is not necessarily easy, and secondly to show how additional information can help to ease the decision-making process. The above example is quite a complex task, needing a strategy that postpones decision-making, such as [6]. However, especially in the context of real-time systems, such a strategy can be expensive to implement. One solution is to be able to detect when a data association problem is becoming complex or simple, and switch between strategies accordingly.

We choose two data association methods, one simple and the other more complex, and present a criterion for switching between the two that gives an indication of the likely complexity of the association task. In both methods the current observations at time $t(k)$ are $\mathbf{y}_{ij}^{t(k)}, j = 1, 2, .., \#Obs_i$ are associated with a representation of the observations at time $t(k-1)$. The representation is different for the two strategies.

**Best First**

The simpler method uses the previous set of observations $\mathbf{y}_{ij}^{t(k-1)}$. Here the idea is that since the association task has already been determined as being relatively easy, the closest observation to the last one is a good choice. We use a distance function, e.g. the Mahalanobis distance, with an associated test threshold based on 95% or 99% confidence, to calculate the distance between all pairs of observations at the two times $d(\mathbf{y}_{ij}^{t(k-1)}, \mathbf{y}_{ij}^{t(k)})$. We find the minimum distance of all the $\#Obs_i(t(k-1)).\#Obs_i(t(k))/2$ pairs, and then associate these two observations together, eliminating all others pairs that contain one of these two observations. This forms the next link of a chain of observations at times $t(k), t(k-1), ...$ Thus the association is at most 1-1, any remaining observations that are unmatched from time $t(k-1)$ are eliminated, and any unmatched from time $t(k)$ start off new observation sequences. A variant on this method, more in the asynchronous spirit of this paper, is to use the idea of a 'lifetime' of an observation. The observations at $t(k), t(k-1)$ are replaced by observations that are not necessarily from consecutive time-frames, but within the lifetime defined. Thus an observation $\mathbf{y}_{ij}^{t_2}$ may only be associated with an existing chain culminating in $\mathbf{y}_{ij}^{t_1}$ if $t_2 - t_1$ is less than the defined lifetime. This technique also gracefully deals with the problem of missed targets in the bootstrap phase.

**Beam Search**

The beam search method is different from the best first one in that multiple hypotheses are allowed at each stage. Instead of choosing the closest observation, we split or duplicate an existing filter into a number of filters, with the number of duplications equal to the number of observations found in the Mahalanobis

neighbourhood. We leave the forthcoming observations to decide which match is correct. The filter resulting from the correct match will be confirmed by subsequent observations, while those resulting from incorrect matches will, in general, not. A support of existence measure is then defined to prune out the unlikely branches of the split filters. The multiple-matches problem is thus handled gracefully. The method is described in greater length in [6].

The only remaining decision is when to switch between the bootstrap phase for an event and the track phase. At present this is still a parameter which is set manually, dependent on the sensor being treated.

## 4.1 A Tracking Complexity Measure

Since we are dealing with the time evolution of physical systems, we use some simple physical measures to help define the tracking complexity measure.

Firstly note that as in the second case of Figure 2, for a physical system there is usually a maximum rate of change in the observation variables (generalised velocity or smoothness condition) which we will denote by $V_{max}$, a vector of dimension $m$. This is different to the model noise that was discussed in Section 2 in that it relates to the observations rather than the state variables, and is valid over a comparatively short timescale. If $V_{max}$ is large, then we must search in a large space for the data associations, and the complexity is high.

Secondly observe that the observation noise, as modelled by the covariance matrix $R$, largely determines how many other observations are within the Mahalanobis neighbourhood of an observation. If $R$ is large, the distinction between neighbouring observations is small. This implies that it will be difficult to associate an observation to a given set of past observations (either a sequence or branched filter), and the tracking complexity will again be high.

We thus define a tracking complexity measure (T.C.M.) [1] as the following:

$$\frac{1}{N}\sum_{j=1}^{\#Obs_i}\sum_{l=1}^{j-1}[(\mathbf{y}_{ij}-\mathbf{y}_{il})^T(R_{ij}+R_{il}+\mathbf{V})^{-1}(\mathbf{y}_{ij}-\mathbf{y}_{il})]^{-1} \tag{5}$$

where $N = \#Obs_i(\#Obs_i+1)/2$ where all the quantities are evaluated at time $t_i^{(k)}$ for observations in the $i$th sensor, and:

$$\mathbf{V} = diag\{[1/3V_{max}(t_i(k)-t_i(k-1))]^2\} \tag{6}$$

The idea, borrowed from pattern recognition, is to determine an average intra-cluster distance on which to base the strategy decision. The measure gives weight to observations that are close together in a neighbourhood definition which includes an element of the expected change in the observation over time. The factor of $1/3$ relates the maximum velocity to the Mahalanobis test, since we are using a $3\sigma$ probability threshold. This test still needs a threshold $T$ to be determined, but since it takes into account the error expectations it is relatively robust.

The strategy for data association in Stage 1 of the two-stage architecture now becomes:

[1] This can be either global, or a bucketing technique can be used

*Try to match* $\mathbf{y}_{ij}^{t(k)}$ *to existing filters* $\mathbf{x}_{ij}(t(k)), P_{ij}(t(k))$ *using best first method.*

*For all those observations remaining unmatched, calculate the tracking complexity measure (TCM).*

*If* $TCM > T$*, use beam search strategy allowing track splits.*

*If* $TCM < T$*, use best first strategy to associate observations.*

*For all sequences that have sufficient observations (another threshold), use these observations to initialise a Kalman filter.*

# 5 Application to a Driver's Assistant System

This two-stage approach to multi-sensor temporal data fusion has been implemented as part of the Eureka PROMETHEUS ProArt project. The driver's assistant system uses several types of visual sensors, including telemetry, linear stereo, movement and other logical sensors to provide information about the environment of the vehicle. The fusion module constructs a map of the environment, giving the relative position, velocity and size of all the perceived obstacles. The map is then analysed by a copilot module in order to determine which obstacles, if any, pose a danger to our vehicle.

An interesting aspect of this application is the relative importance of the Type I errors (false alarms) and the Type II errors (missed targets). Clearly a missed target (a car or other real event not detected by the system) could be very serious for the safety system, since it represents an unknown hazard. At the front of the vehicle the system has a controllable time-of-flight telemetry sensor [1] that enables a focus of attention mechanism to be used. For any perceived event that represents a safety hazard, more information can be collected relating to that event. Thus a false alarm can usually be resolved later by collecting more data, whereas a missed target is more difficult to resolve. The fusion parameters are thus biased heavily towards Type I errors.

The experimental results presented are for simulated noisy data using parameters estimated from a knowledge of the real sensors. The tracking complexity measure is calculated for each sensor independently. Here the telemetry sensor is simulated in scan or uncontrolled mode (as opposed to the controlled focus-of-attention mode). There are two vehicles in front of us.

The two sets of figures represent, for two different noise levels, the performance of the two algorithms. The sensor (our vehicle) is at the bottom of the figure with the field of view upwards. Each rectangle represents an observation, its width across the page representing the perceived size of the event.

The first figure of the sets represent the input to the data fusion program, integrated over about 60 frames, for two rates of false alarms and missed targets. The first set shows an expected number of false alarms of 2 and a missed target probability of 0.1. Occlusion is taken into account for seen events, thus at the top left of the first figure false alarms are seen because one of more of the real events have been missed.

The second and third figures in the sets show the output of the fusion algorithm for the two cases of best-first and beam search respectively. In these figures the size of the rectangle now indicates the *uncertainty* of the position state variables calculated from $P_j$. Thus a series of rectangles getting larger generally indicates

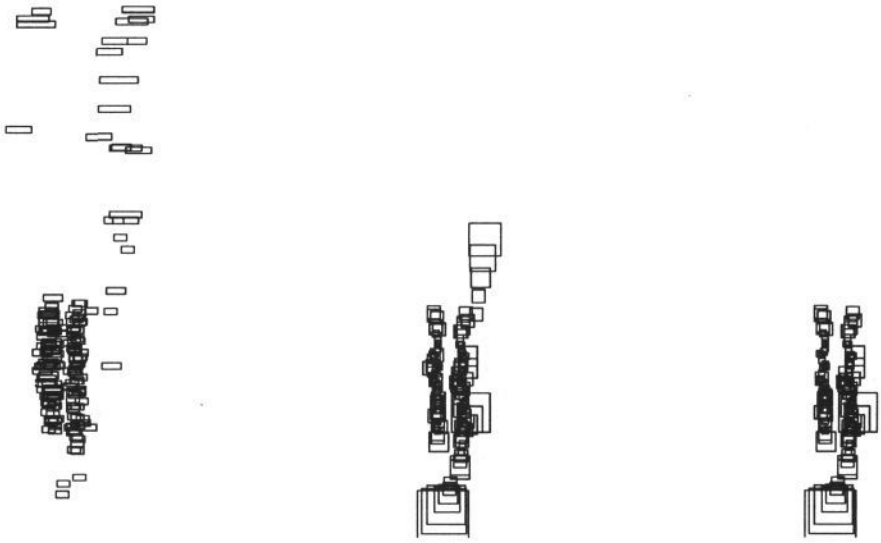

Figure 3: Input/Outputs from Fusion - Simple Case

a false alarm that has received no observations to decrease its uncertainty. The heavy bias towards Type I errors is reflected by the persistence of a false alarm for a relatively long time after observations in its gate have ceased. In principle this may be resolved by using the active mode telemetry. In the first set of figures it can be seen that there is little difference between the performance of the algorithms (the best-first has two major false alarms, the beam search only one). The extra overhead of maintaining multiple hypotheses is not justified and this is reflected by an peak value of the T.C.M. of 0.13.

Figure 4 has the same ground truth data, but with 5 false alarms per frame and missed target probability 0.1. Here the beam-search algorithm out-performs the best-first considerably, making the extra computational expense more worthwhile. The peak T.C.M. here is 0.55.

# 6 Discussion and Conclusions

An architecture for multi-sensor temporal data fusion and a strategy for data association have been presented. The key design points are the following:

An attempt has been made to relate the system parameters to physical quantities. In the context of the driver's assistant application this means that the copilot has the opportunity to meaningfully change the model used by the fusion module, or to reason about errors in the environment map. Of the two remaining thresholds, the first determining the switch between the bootstrap and continuous modes is a function of the false alarm and missed target rates, which have to be empirically determined for any given application. The second threshold on the tracking complexity is relatively robust since observations that are close (in the $R_{ij} + R_{il} + \mathbf{V}$ sense) contribute largely to the measure. A bucketing technique helps to localise the data association problems, so that different strategies can be

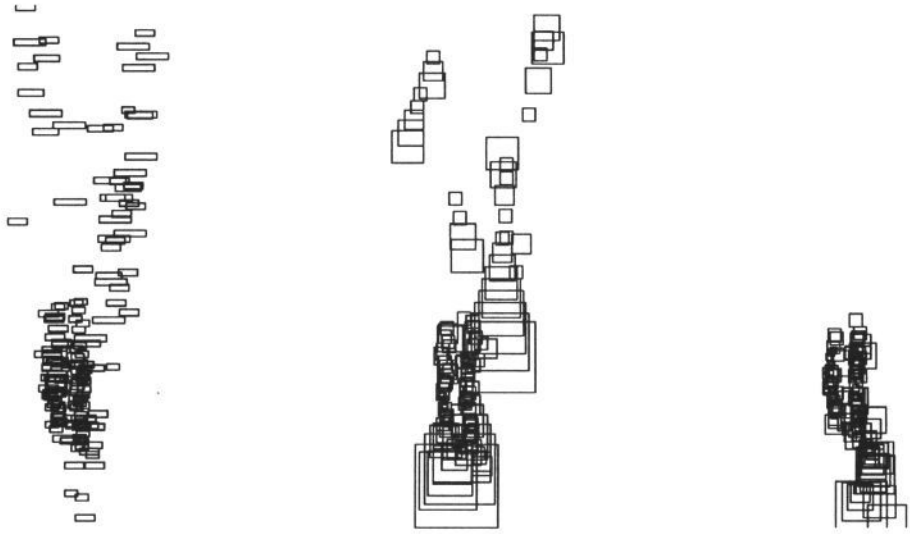

Figure 4: Input/Outputs from Fusion - Complex case

running in different buckets at the same time.

These switches enable a real-time system to make opportunistic savings in time of calculation. However, for the tracking complexity measure, there is certainly an overhead in calculating the measure, and if in the application being considered the switch is made very infrequently, an alternative strategy such as calculating the measure every $N$ seconds, (say), may be better.

# References


[1] J. Alizon, J. Gallice, L. Trassoudaine, and S. Treuillet. Multisensory data fusion for obstacle detection and tracking on motorway. In *Proceedings of the 3rd PROMETHEUS Workshop*, pages 85–94, April 1990.

[2] Y. Bar Shalom and T. E. Fortmann. *Tracking and Data Association.* Academic-Press, Boston, 1988.

[3] James J. Clark and Alan L. Yuille. *Data Fusion for Sensory Information Processing Systems.* Kluwer Academic Publishers, 1990.

[4] H. Durrant-Whyte. *Integration, Coordination and Control of Multi-Sensor Robot Systems.* Kluwer Academic Publishers, 1988.

[5] D. Hutber, T.Vieville, and G. Giraudon. Data fusion for reliable detection and tracking of multiple obstacles in a road environment - an asynchronous approach. In *Proceedings of SPRANN 94*, April 1994.

[6] Z. Zhang. Token tracking in a cluttered scene. *Image and Vision Computing*, 12(2):110–120, March 1994.