Thomas Bak

Lecture Notes - Estimation and Sensor Information Fusion

November 14, 2000

Aalborg University

Department of Control Engineering
Fredrik Bajers Vej 7C
DK-9220 Aalborg
Denmark

Table of Contents

1. Introduction

This note discusses multi-sensor fusion. Through sensor fusion we may combine readings from different sensors, remove inconsistencies and combine the information into one coherent structure. This kind of processing is a fundamental feature of all animal and human navigation, where multiple information sources such as vision, hearing and balance are combined to determine position and plan a path to a goal.

While the concept of data fusion is not new, the emergence of new sensors, advanced processing techniques, and improved processing hardware make real-time fusion of data increasingly possible. Despite advances in electronic components, however, developing data processing applications such as automatic guidance systems has proved difficult. Systems that are in direct contact and interact with the real world, require reliable and accurate information about their environment. This information is acquired using sensors, that is devices that collect data about the world around them.

The ability of one isolated device to provide accurate reliable data of its environment is extremely limited as the environment is usually not very well defined in addition to sensors generally not being a very reliable interface.

Sensor fusion seeks to overcome the drawbacks of current sensor technology by combining information from many independent sources of limited accuracy and reliability to give information of better accuracy and reliability. This makes the system less vulnerable to failures of a single component and generally provide more accurate information. In addition several readings from the same sensor are combined, making the system less sensitive to noise and anomalous observations.

2. Sensors

2.1 Sensor Classification

Sensors may be classified into two groups: *active* or *passive*. *Active* sensors sends a signal to the environment and measures the interaction between this signal and the environment. Examples include sonar and radar. By contrast, *passive* sensors simply record signals already present in the environment. Passive sensors include most thermometers and cameras.

Sensors may also be classified according to the medium used for measuring the environment. This type of classification is related to the senses with which humans are naturally endowed.

Electromagnetic sensors (Human vision).

1. Charged Coupled Devices (CCD)
2. Radio Detection and Ranging (RADAR)
3. ect.

Sound wave sensors (Human hearing).

1. Sound Navigation and Ranging (SONAR)
2. ect.

Odor sensors (Human smell).

Touch sensors.

1. Bumpers
2. Light Emitting Diodes (LED)
3. ect.

Proprioceptive sensors (Human balance). The senses that tell the body of its position and movement are called *proprioceptive* senses. Examples:

1. Gyroscopes
2. Compass
3. GPS
4. ect.

2.2 Multiple Sensor Networks

Multiple sensor networks may be classified by how the sensors in the network interact. Three classes are defined:

Complementary. Sensors are complementary when they do not depend on each other directly, but can be combined to give a more complete image of the environment. Complementary data can often be fused by simply extending the limits of the sensors.

Competitive. Sensors are competitive when they provide independent measurements of the same information. They provide increased reliability and accuracy. Because competitive sensors are redundant, inconsistencies may arise between sensor readings, and care must be taken to combine the data in away that removes the uncertainties. When done properly, this kind of data fusion increases the robustness of the system.

Cooperative. Sensors are cooperative when they provide independent measurements, that when combined provide information that would not be available from any one sensor. Cooperative sensor networks take data from simple sensors and construct a new abstract sensor with data that does not resemble the readings from any one sensor.

The major factors dictating the selection of sensors for fusion are: the compatibility of the sensors for deployment within the same environment, and the complementary nature of the information derived from the sensors. If the sensors were to merely duplicate the information, the the fusion process would merely be the equivalent of building in redundancy for the enhancement of reliability of the overall system.

On the other hand, the sensors should be complementary enough in terms of such variables as data rates, field of view, range, sensitivity to make fusion of information meaningful.

3. Fusion

3.1 Levels of Sensor Fusion

Observational data may be combined, or fused, at a variety of levels from the raw data level to a state vector (feature) level, or at the decision level. A common differentiation is among high-level fusion, which fuses *decisions*, mid-level fusion, which fuses parameters concerning *features* sensed locally, and low-level fusion, which fuses the raw *data* from sensors. The higher the level of fusion, the smaller the amount of information that must be processes.

This hierarchy can be further detailed by classification according to input and output data types, as seen in Figure 3.1.

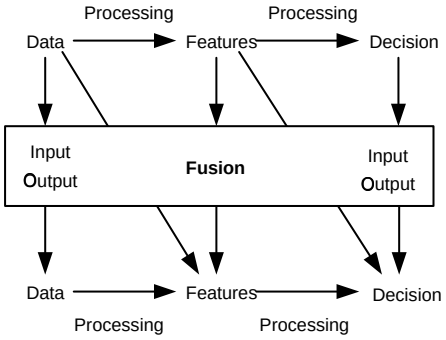

Fig. 3.1. Levels of sensor fusion. The fundamental levels of data-feature-decision are broken down into a mode classification based on input-output relationship.

An additional dimension to the fusion process, is temporal fusion, that is fusion of data or information acquired over a period of time. The can occur on all levels listed above and hence is viewed as a part of the the categorization above.

The human brain is working on all levels in the categorization of Figure 3.1 as well as in temporal fusion. An important characteristic of the brain is the decision fusion because of the ability to take a global perspective. The machine is most efficient relative to humans at the data level because of its ability to process large amounts of raw data in a short period of time.

3.2 An I/O-Based Characterization

By characterizing the three fundamental levels in Figure 3.1, *data-feature-decision* by their input output relationship we get a number of input-output modes:

Data In-Data Out (DIDO). This is the most elementary for of fusion. Fusion paradigms in this category are generally based on techniques developed in the traditional signal processing domain. For example sensors measuring the same physical phenomena, such as two competitive sensors may be combined to identify obstacles at each side of a robot.

Data In-Feature Out (DIFO). Here, data from different sensors are combined to extract some form of feature of the environment or a descriptor of the phenomenon under observation. Depth perception in humans, by combining the visual information from two eyes, can be seen as a classical example of this level of fusion.

Feature In-Feature Out (FIFO). In this stage of the hierarchy, both input and output are features. For example shape features from and imaging sensor may be combined with range information from a radar to provide a measure of the volumetric size of the object being studied.

Feature In-Decision Out (FIDO). Here, the inputs are features and the out of the fusion process is a decision such as a target class recognition. Most pattern recognition systems perform this kind of fusion. Feature vectors are classified based on a priori information to arrive at a class or decision about the pattern.

Decision In-Decision Out (DIDO). This is the last step in the hierarchy. Fusion at this level imply that decision are derived at a lower level, which in turn imply that the fusion process at previously discussed levels have also been performed. Examples of decision level fusion involves weighted methods (voting techniques) and inference.

In practical problems, it is likely that fusion in many of the modes defined above will be incorporated into the design to achieve optimal performance. The process of extracting relevant information from the data, in terms of features and decisions, on the one hand may be throwing away information, but on the other hand may also be reducing the noise that degrade the quality of the ensuing decision.

3.3 Central versus Decentral Fusion

Centralized architectures assume that a single processor performs the entire data fusion process. With the growing complexity of systems, centralized fusion is becoming less attractive due to the high communication load and reliability concerns. An alternative is to perform a local estimation of states or parameters from available data followed by a fusion with other similar features to form a global estimate. When done right, this assures a graceful degradation of the system as nodes fail.

A fully is comprised of a number of nodes. Each nodes process sensor data locally, validates them, and uses the observations to update its state estimate via

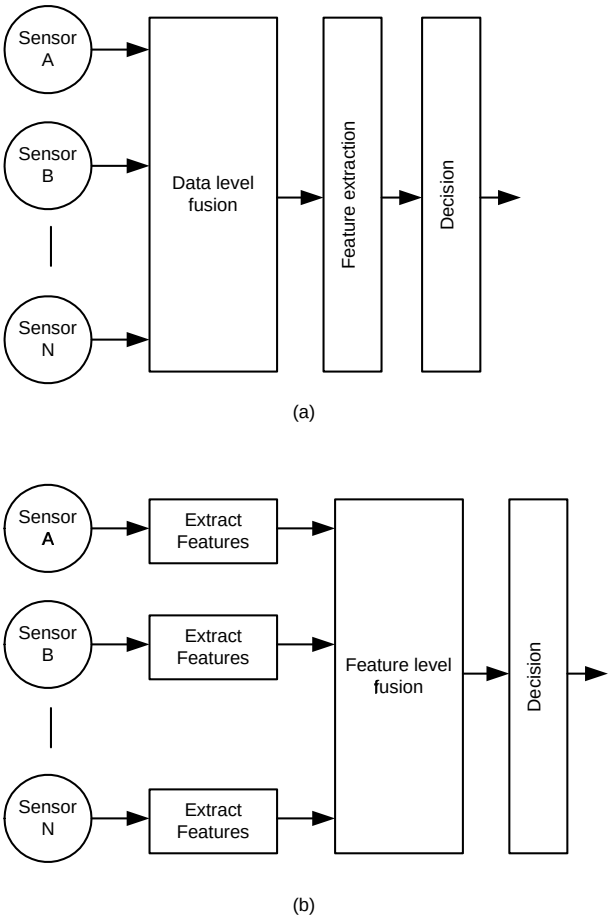

Fig. 3.2. Alternate architectures for multi-sensor identity fusion. (a) Data level fusion, (b) Feature level fusion.

standard Kalman filtering. The nodes then communicates their results in terms of state and possibly parameter estimates to other similar nodes that are then used locally in the other nodes to update their local estimate in a second update. The estimate available is now a global estimate and it may be verified to be the same as would be achieved using a centralized architecture.

The decentralized architecture is more resistant to node failures.

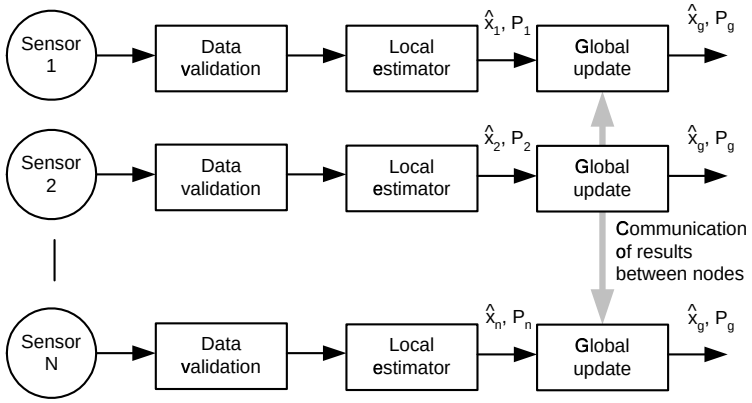

Fig. 3.3. Decentralized fusion of data. Local estimates of state and parameter is updated locally by data from other fusion nodes achieving a global estimate.

4. Multiple Model Estimation

This chapter extends the consideration of estimation theory to consider the use of multiple process models. Multiple process models offer a number of important advantages over single model estimators.

Multiple models allow a modular approach to be taken. Rather than develop a single model which must be accurate for all possible types of behaviour by the true system, a number of different models can be derived. Each model has its own properties and, with an appropriate choice of a data fusion algorithm, it is possible to span a larger model space.

Four different strategies for multiple model estimation are examined in this chapter

- multiple model switching
- multiple model detection
- multiple hypothesis testing
- multiple model fusion

Although the details of each scheme is different, the three first all use fundamentally the same approach. The designer specifies a set of models. At any given time only one of these models is *correct* and all the other models are *incorrect*. The different strategies use different types of test to identify which model is correct and, once this has been achieved, the information from all the other models is neglected.

The latter strategy, *model fusion* utilize that process models are a source of information and their predictions can be viewed as the measurement from a virtual sensor. Therefore, multiple model fusion can be cast in the same framework as multisensor fusion and a Kalman update rule can be used to consistently fuse the predictions of multiple process models together. This strategy has many important benefits over the other approaches to multiple model management. It includes the ability to exploit information about the differences in behaviour of each model. As a result, the fusion is synergistic: the performance of the fused system can be better than that of any individual model.

4.1 Benefits of Multiple Models

Increasing the complexity of a process model does not necessarily lead to a better estimator. As the process model becomes more complicated, it makes more and

more assumptions about the behaviour of the true system. These are embodied as an increasing number of constraints on situations in which the model fits the real world. When the behaviour of the true system is consistent with the assumptions in the model, its performance is very good. However, when the behaviour is not consistent, performance can be significantly worse. Simply increasing the order of the model without regard to this phenomena could lead to a very complicated, high order model which works extremely well in very limited circumstances. In most other situations, its performance could be poor.

Multimodel estimation techniques are a method which addresses this underlying problem. Rather than seek a single model which is sufficiently complicated and flexible that it is consistent with all possible types of behaviour for the true system, a number of low order approximate systems are developed. Each approximate system is developed with a different assumptions about the behaviour of the true system. Since these assumptions are different, the situations in which each model fits is different.

By ensuring a suitable complement over the range of operation of the system, good performance can be achieved under all conditions. Although the accuracy of a single model in a specific circumstance is lost, it is balanced by the range of situations over which performance is good. The models are combined so that the strengths of one covers for the weaknesses of the others. Figure 4.1 provides a representation of this.

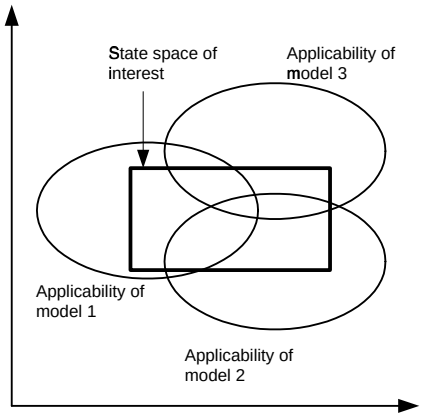

Fig. 4.1. The use of multiple models. Three different models have been derived with different but overlapping *domains*. Although a single model is not capable of describing the entire behaviour of the system, a suitable combination of all of the models can.

4.2 Problem Formulation

Suppose a set of n process models have been derived. Each uses a different set of assumptions about the true system, and the result is a set of n approximate systems. The i'th approximate system is described using the process and observation models

$$x_i(k) = f(x_i(k-1), t_{k-1}) + v(k-1) \tag{4.1}$$
$$z_i(k) = h(x_i(k), t_k) + w(k) \tag{4.2}$$

The state space of the approximate system is related to the true system state, x_{T}, according to the structural equations

$$x_i(k) = g_i(x_{\mathrm{T}}(k), t_k)$$

The problem is to find the estimate of the variables of interest $y(k|k)$[1] with covariance $P_{yy}(k|k)$ which takes account of all of the different approximate models such that the trace of $P_{yy}(k|k)$ is minimized. The strategy is not limited to just combining the estimates of the variables in an output stage, the estimate may also be feed back to each of the approximate systems, directly affecting their estimates.

This is reflected in the model management strategy which may be summarized in terms of two sets of functions – the critical value output function

$$y(k) = m(y_1(k), \ldots, y_n(k)) \tag{4.3}$$

which combine the multiple model estimates and the approximate system update function

$$x_i(k) = n_i(x_1(k), \ldots, x_n(k)) \tag{4.4}$$

that updates the i'th model state. A successful multimodel management strategy should have the following properties:

Consistency. The estimate should always be consistent[2]. The estimates become inconsistent when the discrepancy between the estimated covariance and the true covariance is not positive semidefinite.

Performance. There should be a performance advantage over using a single process model. This justifies the extra complexity of using multimodels instead of a single model. These benefits include more accurate estimates and more robust solutions.

Theory. The method should have a firm theoretical foundation. Not only does this mean that the range of applicability is known, but it is also possible to generalize the technique to a wide range of applications.

[1] The whole state space may not be of interest, and $y(k|k)$ may thus be a subset of $x(k|k)$

[2] That is $P(i|j) - \mathrm{E}\{\tilde{x}_(i|j)\tilde{x}_(^\top i|j)|Z_j\} < 0$ where $\tilde{x}$ are state errors, and Z_j are measurements up to time t_j

4.3 Methods of Multi-Model Estimation

The principle and practice of multiple model estimation has been used extensively for many years. This section reviews various multimodel schemes.

4.3.1 Model Switching Methods

The simplest approach for estimation with multiple models is model switching. It is simple to understand, computationally efficient, and readily facilitates the use of many different models. The essential idea is shown in Figure 4.2.

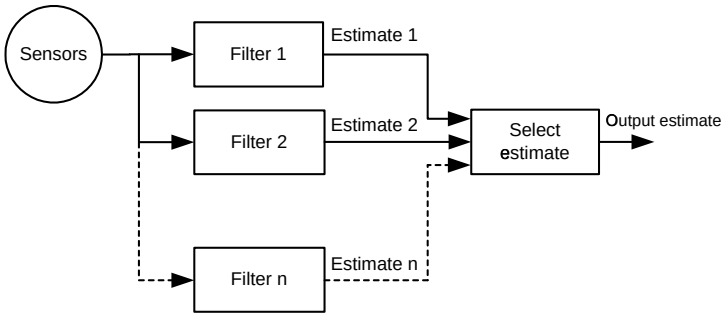

Fig. 4.2. The model switching method to estimation with multiple models. The output from the bank of filters is that of the filter which uses an appropriate model for the given situation.

Given that n filters are implemented, one for each process model. All the filters operate completely independently and in parallel with one another. They make predictions, perform updates, and generate the estimates of the critical values of the system. Combining the information from the different models is very simple. It is assumed that, at any time t_k, only one model is *appropriate*. The output from the bank of filters is that of the filter which uses the appropriate model. The problem is to identify the appropriate model. The output model management function is

$$m(y_1(k), \ldots, y_n(k)) = \delta_1(k)y_1(k) + \ldots \delta_n(k)y_n(k) \tag{4.5}$$

where

$$\delta_i(k) = \begin{cases} 1 & \text{if model } i \text{ is selected,} \\ 0 & \text{if model } i \text{ is not selected.} \end{cases} \tag{4.6}$$

Since only one model is chosen at any time $\delta_i(k)$ sums to unity. The filters combine information only for output purposes and their respective state spaces are unchanged.

The key design issue with this approach is to choose the switching logic which determines the values of $\delta_i(k)$ which is dependent on the specific application. The

landbased navigation of the Space Shuttle (Ewell 1988), for example, uses two different process models which correspond to accelerating and cruising flight. The accelerating model is used when the acceleration exceeds a predefined limit. When acceleration is below some critical level, the shuttle is assumed to be in cruise mode. In these cases, complex gravitational and atmospheric models are incorporated into the equations of motion. The threshold was selected using extensive empirical tests.

Switching on the basis of measurements (or estimates) may lead to jitter if the switching criterion is noisy and its mean value is near the switching threshold. A more refined solution is to employ hysteresis switching which switches only when there is a significant difference between the performance of the different models.

Although the model switching method and its variants are simple to understand and use, there are a significant number of problems. The most important is that the choice of the switching logic is often arbitrary, and has to be selected empirically – there is no theoretical justification that the switching logic is correct.

4.3.2 The Model Detection Method

The model detection approach is a more sophisticated version of the model switching method. Rather than switch on the basis of various ad hoc thresholds, it attempts to detect which model is the least complex model (parsimonious ((Ljung 1987))), which is capable of describing the most significant features of the true system. The issue is a bias/variance tradeoff. As a model becomes more complex, it becomes a better description of the true system, and the bias becomes progressively smaller. However, a more complex model includes more states. Given that there is only a finite amount of noisy data, the result is that the information has to be spread between the states and the variance on the estimates of all of the states increase.

For example, if a vehicle is driving in a straight line a very simple model may be used. However, when the vehicle turn, the effects of dynamics and slip may have to be included. By ensuring that the least complex model is used at any given point in time, the *variance* component of the bias/variance tradeoff is minimized so that the model is not overly complicated for the maneuvers in question.

The output model management function is of the same form as Equations (4.5) and (4.6). No information is propagated from one model to the next and the approximate system update is hence unity.

One method for testing whether a model is parsimonious or not is to use model reduction techniques. These examine how a high order model can be approximated by a low order model. A crucial component for any model reduction scheme is the ability to assess the relative importance that different states have on the behaviour of the system. If a number of states have very little effect on output, then they can be deleted without introducing substantial errors.

To make the contributions from different modes clear, we turn to balanced realizations (Silvermann, Shookoohi, and Dooren 1983). In balanced realizations a linear transformation is applied to the state space of the system. In this form, the contribution of different modes is made clear, and decision rules can be easily formulated.

To calculate the balanced realizations for the linear system,

$$x(k+1) = F(k)x(k) + B(k)u(k) \tag{4.7}$$

$$z(k) = H(k)x(k) \tag{4.8}$$

its observability, M_{d}, and controllability, W_{d}, Grammians are evaluated across a window of length l according to the equations

$$M_{\mathrm{d}}(k, k-l) \triangleq \sum_{i=k-l+1}^{k} \Phi^{\top}(i-1,k)H^{\top}(i)H(i)\Phi(i-1,k) \tag{4.9}$$

$$W_{\mathrm{d}}(k, k-l) \triangleq \sum_{i=k-l+1}^{k} \Phi(k,i-1)B(i)B^{\top}(i)\Phi^{\top}(k,i-1) \tag{4.10}$$

where $\Phi(i,m)$ is the state transition matrix from discrete step *m* to *i*.

The Grammians are diagonalized by finding the matrices $T(k)$ and $\Sigma^2(k)$ such that

$$M_{\mathrm{d}}(k-l,k)W_{\mathrm{d}}(k-l,k) = T(k)\Sigma^2(k)T^{-1}(k) \tag{4.11}$$

If the linear transformation $T(k)$ is applied to the state vector, the calculated Grammians are diagonal and both equal to $\Sigma(k)$ which is the matrix of singular values. This matrix is diagonal:

$$\Sigma(k) = \begin{bmatrix} \sigma_1(k) & 0 & \dots & 0 \\ 0 & \sigma_2(k) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \sigma_n(k) \end{bmatrix} \tag{4.12}$$

The singular values reflect the significance that each state has for the behaviour of the system (Silvermann, Shookoohi, and Dooren 1983). Both the observability and controllability of the state are included in the measure. If a state is almost unobservable, it has little effect on the measured output of the system. If it is almost uncontrollable, the control inputs can exert little influence on its final value. If it is both unobservable and uncontrollable, its value is not affected by the control inputs and it does not change the output of the system.

Example 4.3.1. A vehicle dynamics model may include tire stiffness parameters in the state vector in order to model the a number of physical parameters in the tire that change over time. When the vehicle drives in a straight line the forces and slip angles are small, rendering the tire stiffness parameters unobservable. In this situation, the filter should switch to a lower order model without tire modeling.

As the method above is only used for the switching rule, linearization is acceptable for nonlinear models, but the overhead associated with this method is still significant. Moreover it may be difficult to evaluate the results.

The two approaches which have been described so far use deterministic tests to identify which model is appropriate. The alternative is to consider probabilistic methods which are described next.

4.3.3 Multiple Hypothesis Testing

Multiple Hypothesis Testing (Magill 1965) (MHT) is one of the most widely used approaches to multimodel estimation. Rather than using bounds to define regions of applicability, each model is considered to be a candidate for the true model. For each model, the hypothesis that it is the true model is raised, and the probability that the hypothesis is correct is evaluated using the observation sequence. Over time, the most accurate model is assigned the largest probability and so dominates the behaviour of the estimator.

Figure 4.3 shows the structure of this method. Each model is implemented in its own filter and the only interaction occurs in forming the output. The output is a function of the estimates made by the different models as well as the probability that each model is correct. The fused estimate is not used to refine the estimates maintained in each model and so the approximate system update functions are unity.

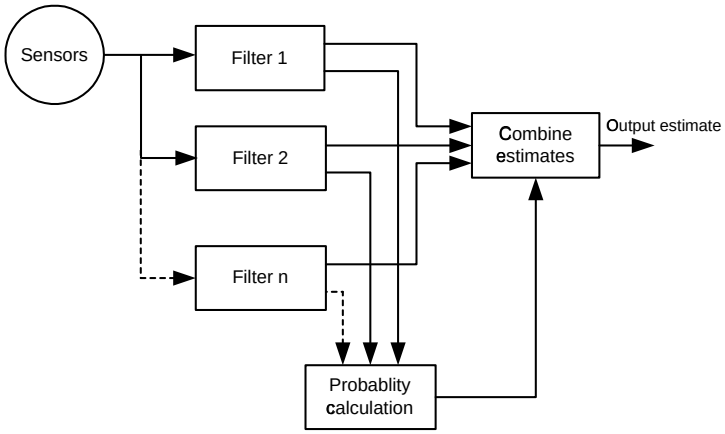

Fig. 4.3. The model hypothesis method. The output is a function of the estimates made by the different models as well as the probability that each model is correct.

The following two assumptions are made:

1. The true model is one of those proposed.
2. The same model has been in action since $t = 0$.

From these two assumptions, a set of n hypotheses are formed, one for each model. The hypothesis for the i'th model is

$$H_i = \text{Model } M_i \text{ is correct.} \tag{4.13}$$

By assumption 1 there is no null hypothesis because the true model is one of models in the set. The probability that model M_i is correct at time t_k, conditioned on the measurements up to that time, Z_k is

$$\mu_i(k) \triangleq p(M_i|Z_k) \tag{4.14}$$

The initial probability that M_j is correct is given by $\mu_j(0)$, which according to assumption 1 sum to unity over all models.

The probability assigned to each model changes through time as more information becomes available. Each filter predicts and updates according to the Kalman filter equations. Consider the evolution of μ_j from $k-1$ to k. Using Bayes' rule

$$\begin{aligned}
\mu_i(k) &\triangleq p(M_i|Z_k) \\
&= p(M_i|z(k), Z_{k-1}) \\
&= \frac{p(z(k)|Z_{k-1}, M_i)p(M_j|Z_{k-1})}{p(z(k)|Z_{k-1})} \\
&= \frac{p(z(k)|Z_{k-1}, M_i)\mu_i(k-1)}{\sum_{j=1}^{n} p(z(k)|Z_{k-1}, M_j)p(M_j|Z_{k-1})}
\end{aligned} \tag{4.15}$$

The $p(z(k)|Z_{k-1}, M_i)$ is the probability that the observation $z(k)$ would be made given that the model M_j is valid. This probability may be calculated directly from the innovations $r_j(k) \triangleq z(k) - \hat{z}(k|k-1)$. Assuming that the innovation is Gaussian, zero mean and has covariance $S(k|k-1)$ the likelihood is

$$\lambda_i(k) = \frac{1}{(2\pi)^{m/2}\det(S_i(k|k-1))^{1/2}} \exp\{\cdot\} \tag{4.16}$$

$$\{\cdot\} = \left(-\frac{1}{2} r_i^{\top}(k) S_i^{-1}(k|k-1) r_i(k)\right) \tag{4.17}$$

where m is the dimension of the innovation vector.

The innovation covariance $S(k)$ is maintained by the Kalman filter and given by

$$S_i(k|k-1) = H_i(k)P_i(k)H_i^{\top}(k) + R_i(k). \tag{4.18}$$

The MHT now works by recursion, first calculate $\lambda_i(k)$ for each model. The new probabilities are now given by

$$\mu_i(k) = \frac{\lambda_i(k)\mu_i(k-1)}{\sum_{j=1}^{n} \lambda_j(k)\mu_j(k-1)} \tag{4.19}$$

The minimum mean squared error estimate is the expected value. The output function, i.e. the output of the multiple model is thus given by

$$\hat{y}(k|k) = m\big(y_1(k), \dots, y_n(k)\big) \tag{4.20}$$

$$= \mathrm{E}\big(y(k)|Z_k\big) \tag{4.21}$$

$$= \mathrm{E}\left(\sum_{i=1}^{n} y_i(k)p(M_i|Z_k)\right) \tag{4.22}$$

$$= \sum_{i=1}^{n} \hat{y}_i \mu_i(k) \tag{4.23}$$

The covariance in the estimate can be found to (Bar-Shalom and Fortmann 1988)

$$P_{yy}(k|k) = \sum_{i=1}^{n} \mu_i(k) \left(P_{i,yy}(k|k) + (\hat{y}_i(k|k) - \hat{y}(k|k))(\hat{y}_i(k|k) - \hat{y}(k|k))^\top \right) \tag{4.24}$$

which is a sum of probability weighted terms from each model. In addition to the direct covariance from the models, a term is included that takes account for the fact that $\hat{y}_i$ is not equal to y_i.

The MHT approach was developed initially for problems related to system identification – the dynamics of a nominal plant were specified, but there was some uncertainty in the values of several key parameters. A bank of filters were implemented, one for each filter candidate parameter value. Over time, one model acquires a significantly greater probability than the other models, and it corresponds to the model whose parameter values are most appropriate.

The basic form of MHT has been widely employed in missile tracking where different motion models correspond to different types of maneuvers (Bar-Shalom and Fortmann 1988). Experience, however, shows that there are numerous practical problems with applying MHT. The performance of the algorithm is dependent upon a significant difference between the residual characteristics in the *correct* and the *mismatched* filters, which is sometimes difficult to establish.

The three methods we have reviewed until now each has a number of shortcomings, many of which stem from the fact that they use fundamentally the same principle. The methods all at each instance in time assumed that only one model is *correct*, and the multimodel fusion problem is to identify that model. However this includes the extremely strong assumption that all the other models, no matter how close they are to the current most appropriate model, provide no useful information whatever.

These shortcomings motivate the development of a new strategy for fusing information from multiple models. This approach, known as model fusion, is described next.

4.3.4 Model Fusion

The model fusion approach treats each process model as if it were a *virtual sensor*. The i'th process model is equivalent to a sensor which measures the predicted value $\hat{x}_i(k|k-1)$. The *measurement* is corrupted by noise (the prediction error) and the same principles and techniques as in multi-sensor fusion may be used.

Figure 4.4 illustrates the method for two approximate systems. Each model is implemented in its own filter and, in the prediction step, each filter predicts the future state of the system. Since each filter uses a different model and a different state space, the predictions are different.

Filter 1 (or 2) propagates its prediction (or some functions of it) to filter 2 (or 1) which treats it as if it were an observation. Each filter then updates using prediction and sensor information alike.

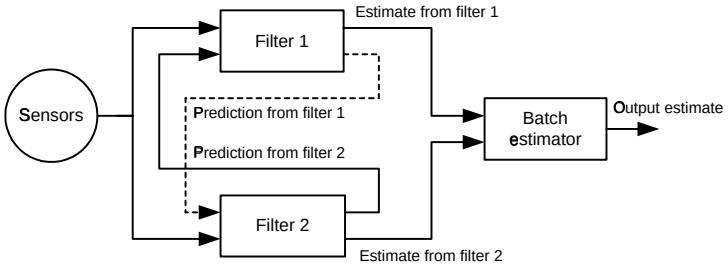

Fig. 4.4. Model fusion architecture for two approximate systems. Each filter predicts the future state of the system using the same inputs, and updates using sensor information and the prediction propagated from the other filter.

The process model is used to predict the future state of the system. Using the process model the prediction summarizes all the previous observation information. The estimate at t_k is thus not restricted to using the information contained in $\boldsymbol{z}(k)$. The prediction allow temporal fusion with data from all past measurements.

The fusion of data is best appreciated by considering the information form (Maybeck 1979) of the Kalman filter prediction. The amount of information maintained in an estimate $\hat{x}$ is defined to be the inverse of its covariance matrix. With the Kalman gain K, the covariance prediction may be rewritten as

$$P(k|k) = P(k|k-1) - KH(k)P(k|k-1) \tag{4.25}$$

$$P(k|k)P^{-1}(k|k-1) = 1 - KH(k) \tag{4.26}$$

Using this result in combination with the Kalman gain in terms of the predicted covariance, (Maybeck 1979)

$$K = P(k|k)H(k)^{\top}R(k)^{-1} \tag{4.27}$$

where $R(k)$ is the measurement covariance. The state measurement update may be written as

$$\hat{x}(k|k) = P(k|k)\left[P^{-1}(k|k-1)\hat{x}(k|k-1) + H(k)^{\top}R(k)^{-1}\boldsymbol{z}(k)\right] \tag{4.28}$$

A similar expression can be found for the predicted information matrix

$$P(k|k)^{-1} = P(k|k-1)^{-1} + H(k)^{\top}R(k)^{-1}H(k) \tag{4.29}$$

The estimate in Equation (4.28) is therefore a weighted average of the prediction and the observation. Intuitively the weights must be inversely proportional to their respective covariances. Seen in the light of Equation (4.28) predictions and observations are treated equally by the Kalman filter. It is therefore possible to consider the prediction from filter 1 (2) as an observation, $\hat{x}(k|k-1)$ with covariance $P^{-1}(k|k-1)$ that can be used to update filter 2 (1).

Batch Estimator. This approach is fundamentally different from the other approaches which are described in this chapter. Rather than assume that only one process model is *correct*, all the models are treated as valid descriptions of the true system. At some points in time one model might be better than another, but this does not mean that the poorer model yields no useful information. On the contrary, since all the models are capable of making consistent predictions, they all provide information.

Each process model makes its own estimates of the interesting variables. The different estimates can be fused together efficiently and rigorously using a batch estimator (Bierman 1977). Define

$$Y(k) = \begin{bmatrix} y_1(k) \\ \dots \\ y_n(k) \end{bmatrix}, \tag{4.30}$$

$$H = \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix}, \tag{4.31}$$

$$P_{YY}(k|k) = \begin{bmatrix} P_{y_1y_1}(k|k) & P_{y_1y_2}(k|k) & \dots & P_{y_1y_n}(k|k) \\ P_{y_2y_1}(k|k) & P_{y_2y_2}(k|k) & \dots & P_{y_2y_n}(k|k) \\ \vdots & \vdots & \ddots & \vdots \\ P_{y_ny_1}(k|k) & P_{y_ny_2}(k|k) & \dots & P_{y_ny_n}(k|k) \end{bmatrix} \tag{4.32}$$

the estimates of interest are then given by

$$P_{yy}(k|k) = \left(H^T P_{YY}^{-1}(k|k) H\right)^{-1} \tag{4.33}$$

$$y(k) = P_{YY}^{-1}(k|k) P_{yy}(k|k) H^T Y(k) \tag{4.34}$$

There are several advantages of this method. First, it is not necessary to assume that the real physical system switches from mode to mode, nor is it necessary to develop a rule to decide which mode should be used. Rather, the information is incorporated using the Kalman filter update equations.

Suppose that filter 1 is the *correct* model, characterized by the fact that its (consistent) prediction has a smaller covariance than that of filter 2. Filter 2 then weights the predictions from filter 1 very highly in its update. Conversely, filter 1 weights the predictions from filter 2 by a very small fraction. The result is that the information which is *most correct* is given the greatest weight. This method is also capable of exploiting additional information which none of the other filters used.

The Structure of Prediction Propagation. In general, the filters cannot directly propagate and fuse the entire prediction vectors with one another. The reason is that different models use different state spaces. One model might possess states which the other models do not have.

Conversely, two models might have states which are related to one another through a transformation (for example two filters might estimate the absolute position of two different points rigidly fixed to the same vehicle). Only information which can be expressed in a common space can be propagated between each filter.

A model translator function $T_{1\to 12}(x_1(k))$ propagates the prediction from filter 1 into a state space which is common to both filters. Similarly, the prediction from filter 2 is propagated using its own translator function. The fusion space for both filters consists of parameters which are the same in both filters. These parameters obey the condition that

$$T_{2\to 12}(\bar{x}_2(k)) = T_{1\to 12}(\bar{x}_1(k)) \tag{4.35}$$

for all choices of the true state space vector $x_{\mathrm{T}}(k)$ and $\bar{x}_i(k) = g_i(x_{\mathrm{T}}(k))$. Candidate quantities include

- States which are common to both models (such as the position of the same point measured in the same coordinate system).
- The predicted observations.
- The predicted variables (states of interest).

However, it is important to stress that many states do not obey this condition. This is because many states are lumped parameters: their value reflects the net effects of many different physical processes. Since different approximate systems use different assumptions, different physical processes are lumped together in different parameters.

Each filter treats the propagation from the other filter as a type of observation which is appended to the observation state space. Therefore, the observation vector which is received by filter 1 is

$$z_1(k) = \begin{bmatrix} z(k) \\ T_{2\to 12}(x_2(k)) \end{bmatrix} \tag{4.36}$$

The model fusion is a straightforward procedure. For each pair of models find a common state space in which the condition of Equation (4.35) hold. The observation space for each filter is augmented appropriately, and the application of the filter follows trivially.

The model fusion system relies on the Kalman filter for its information propagation, and we will hence have to take account of that the prediction errors in the different models are correlated with one another in order to get consistent estimates.

Figure 4.5 shows the information flows which occur when two models are fused. Information can be classified in two forms: information which flows from the *outside* (and is independent) and that which flows *within* (the predictions which propagate between the filters). The external flows are indicated by dashed lines, internal flows by solid ones. Both flows play a key role in the prediction and the update. In the prediction step each filter uses its own process model to estimate the future state of the system.

The process noises experienced by each model are not independent for two reasons. First, there is a component due to the true system process noise. Second, the modeling error terms for each filter are evaluated about the true system state. Since the process noises are correlated, the prediction errors are also correlated. Each filter

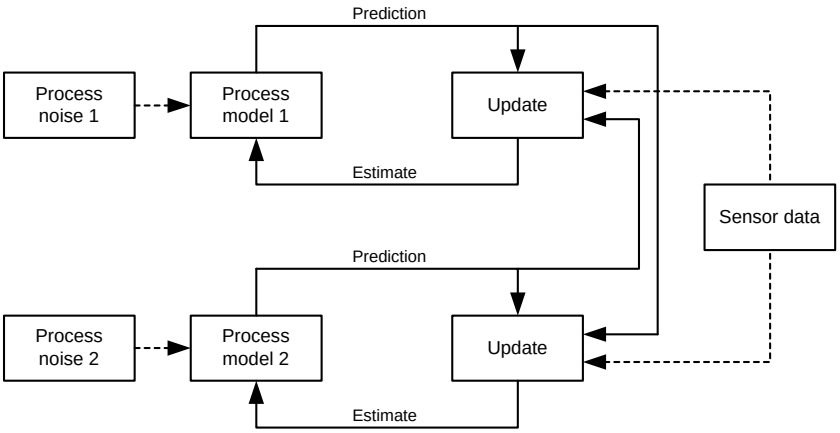

Fig. 4.5. Information flow in the fused two-model case. The external (and independent) flows are indicated by dashed lines, internal flows (the predictions) by solid ones.

updates using its own prediction, the prediction from the other filter, and the sensor information.

Since the process noises are correlated, the prediction errors committed by both models

$$\tilde{x}_1(k|k-1) = F_1\tilde{x}_1(k-1|k-1) + v_1(k-1) \tag{4.37}$$
$$\tilde{x}_2(k|k-1) = F_2\tilde{x}_2(k-1|k-1) + v_2(k-1) \tag{4.38}$$

are also correlated. The cross correlation is

$$P_{12}(k|k-1) = F_1P_{12}(k-1|k-1)F_2^\top + Q_12(k-1) \tag{4.39}$$

where Q_{12} is the cross correlation matrix between the two process models.

As can be seen, the cross-correlations between the prediction errors evolve in a similar fashion to the covariance of each state estimate. First, the diffusion step scales the cross correlation, second, the correlation is reinforced by adding process noise correlation.

More insight is gained by looking at a Taylor series expansion of the measurement vector given in Equation 4.36.

$$r_1(k) = \begin{bmatrix} z(k) - \mathrm{E}\Big(h\big(x_1(k-1)\big)|Z_{k-1}\Big) \\ T_{2\to12}\big(x_2(k)\big) - T_{1\to12}\big(x_1(k)\big) \end{bmatrix} \tag{4.40}$$
$$= \begin{bmatrix} H_1(k)\tilde{x}_1(k|k-1) \\ \nabla T_{2\to12}\big(\tilde{x}_2(k|k-1)\big) - \nabla T_{1\to12}\big(\tilde{x}_1(k|k-1)\big) \end{bmatrix} \tag{4.41}$$

where $\nabla T_{1\to12}$ represents the Jacobian, i.e. a linearization about the current state. Taking outer products leads to the following cross covariance equation

$$P_{x_1 r_1}(k|k-1) = \begin{bmatrix} P_{11}(k|k-1)H_1^T(k) \\ P_{11}(k|k-1)\nabla^\top T_{1\to 12} - P_{12}(k|k-1)\nabla^\top T_{2\to 12} \end{bmatrix} \quad (4.42)$$

which shows how information flows between the two filters. The first component describes how the observation information is injected. The second component, which determined the weight placed on the prediction propagated from filter 2, is the difference between the covariance of filter 1, and the crosscorrelation between filters 1 and 2 all projected into the same space.

In effect only the information which is unique to filter 2 is propagated to filter 1. As the two filters become more and more similar to one another, they become more and more tightly correlated. Less information is propagated from filter 2 and, in the limit when both filters use exactly the same process models, no information is passed between them at all. Model fusion does not *invent* information over and above what can be obtained from the sensors. Rather, it is a different way to exploit the observations. In the other limit, as the models become less and less correlated with one another, the predictions contain more and more independent information and so is assigned a progressively greater weight. The update enforces the correlation between the filters through the fact that the same observations are used to update both filters with the same observation noises.

The rather complex cross correlations in the observation noise may be simplified by treating the network of filters and cross correlation filters as components of a single, all encompassing system or combined system.

4.3.5 Combined System

The combined system is formed by stacking the state spaces of each approximate system. The combined state space vector x_c is

$$x_c(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_N(k) \end{bmatrix} \quad (4.43)$$

The dimension of x_c is the sum of the dimensions of the subsystem filters. The covariance of the combined system is

$$P_c(k|k) = \begin{bmatrix} P_{11}(k|k) & P_{12}(k|k) & \cdots & P_{1n}(k|k) \\ P_{21}(k|k) & P_{22}(k|k) & \cdots & P_{2n}(k|k) \\ \vdots & \vdots & \ddots & \vdots \\ P_{n1}(k|k) & P_{n2}(k|k) & \cdots & P_{nn}(k|k) \end{bmatrix} \quad (4.44)$$

The diagonal subblocks are the covariances of the subsystems, and the off diagonal subblocks are the correlations between them.

The combined system evolves over time according to a process model which is corrupted by process noise. These are given by stacking the components from each subsystem:

$$f_c(x_c(k|k), u(k), v_c(k), t_k) = \begin{bmatrix} f_1(x_1(k), u(k), v_1(k), t_k) \\ f_2(x_2(k), u(k), v_2(k), t_k) \\ \vdots \\ f_n(x_n(k), u(k), v_n(k), t_k) \end{bmatrix} \tag{4.45}$$

The associated covariance is

$$Q_c(k) = \begin{pmatrix} Q_{11}(k) & Q_{12}(k) & \cdots & Q_{1n}(k) \\ Q_{21}(k) & Q_{22}(k) & \cdots & Q_{2n}(k) \\ \vdots & \vdots & \ddots & \vdots \\ Q_{n1}(k) & Q_{n2}(k) & \ddots & Q_{nn}(k) \end{pmatrix} \tag{4.46}$$

The observation vectors from the individual subsystems are also aggregated to yield the combined system's observation vector. However, simply stacking the observation vectors from the individual subsystems can introduce redundant terms which must be eliminated. This problem can be illustrated by considering the innovation vectors for the two system case. Acting independently of one another, the innovations for the two approximate systems are

$$r_1(k) = \begin{bmatrix} z(k) - h_1\big(x_1(k-1)\big) \\ T_{2\to 12}\big(x_2(k)\big) - T_{1\to 12}\big(x_1(k)\big) \end{bmatrix} \tag{4.47}$$

$$r_2(k) = \begin{bmatrix} z(k) - h_2\big(x_2(k-1)\big) \\ T_{1\to 12}\big(x_1(k)\big) - T_{2\to 12}\big(x_2(k)\big) \end{bmatrix} \tag{4.48}$$

Both innovation vectors have the same model fusion component $T_{2\to 12}\big(x_2(k)\big) - T_{1\to 12}\big(x_1(k)\big)$ apart from a sign. Eliminating the redundant terms, the combined innovation vector becomes

$$r_c(k) = \begin{bmatrix} z(k) - h_1\big(x_1(k-1)\big) \\ T_{2\to 12}\big(x_2(k)\big) - T_{1\to 12}\big(x_1(k)\big) \\ z(k) - h_2\big(x_2(k-1)\big) \end{bmatrix} \tag{4.49}$$

To be continued ....

References

Bar-Shalom, Y. and T. E. Fortmann (1988). *Tracking and Data Assocations*. The Academic Press.

Bierman, G. J. (1977). *Factorization Methods for Discrete Sequential Estimation*, Volume 128 of *Mathematics in Science and Engineering*. Academic Press.

Ewell, J. J. (1988). Space shuttle orbiter entry through land navigation. In *IEEE International Conference on Intelligent Robots and Systems*, pp. 627–632.

Ljung, L. (1987). *System Identification, Theory for the User*. Prentice-Hall information and system sciences series. Prentice-Hall.

Magill, D. D. (1965, October). Optimal adaptive estimation of sampled stochastic processes. *IEEE transaction on automatic Control 10*(4), 434–439.

Maybeck, P. S. (1979). *Stochastic Models, Estimation, and Control*, Volume 1. Academic Press.

Silvermann, L., S. Shookoohi, and P. V. Dooren (1983, August). Linear time-variable systems: Balancing and model reduction. *IEEE transaction on automatic Control 28*(8), 810–822.