Department of Computing

Using Dynamic Time Warping for Multi-sensor Fusion

Ko Ming Hsiao

This thesis is presented for the Degree of
Master of Science (Computer Science
of
Curtin University of Technology

April 2009

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature: ………………………………………….

Date: ………………………...

Abstract

Fusion is a fundamental human process that occurs in some form at all levels of sense organs such as visual and sound information received from eyes and ears respectively, to the highest levels of decision making such as our brain fuses visual and sound information to make decisions. *Multi-sensor data fusion* is concerned with gaining information from multiple sensors by fusing across raw data, features or decisions. The traditional frameworks for multi-sensor data fusion only concern fusion at specific points in time. However, many real world situations change over time. When the multi-sensor system is used for situation awareness, it is useful not only to know the state or event of the situation at a point in time, but also more importantly, to understand the causalities of those states or events changing over time.

Hence, we proposed a multi-agent framework for temporal fusion, which emphasises the time dimension of the fusion process, that is, fusion of the multi-sensor data or events derived over a period of time. The proposed multi-agent framework has three major layers: *hardware*, *agents*, and *users*. There are three different fusion architectures: *centralized*, *hierarchical*, and *distributed*, for organising the group of agents. The temporal fusion process of the proposed framework is elaborated by using the *information graph*. Finally, the core of the proposed temporal fusion framework – Dynamic Time Warping (DTW) temporal fusion agent is described in detail.

Fusing multisensory data over a period of time is a challenging task, since the data to be fused consists of complex sequences that are multi–dimensional, multimodal, interacting, and time–varying in nature. Additionally, performing temporal fusion efficiently in real–time is another challenge due to the large amount of data to be fused. To address these issues, we proposed the DTW temporal fusion agent that includes four major modules: *data pre-processing*, *DTW recogniser*, *class templates*, and *decision making*. The DTW recogniser is extended in various ways to deal with the variability of multimodal sequences acquired from multiple heterogeneous sensors, the problems of unknown start and end points, multimodal sequences of the same class that hence has different lengths locally and/or globally, and the challenges of online temporal fusion.

We evaluate the performance of the proposed DTW temporal fusion agent on two real world datasets: 1) accelerometer data acquired from performing two hand gestures, and 2) a benchmark dataset acquired from carrying a mobile device and performing pre-defined user scenarios. Performance results of the DTW based system are compared with those of a Hidden Markov Model (HMM) based system. The experimental results from both datasets demonstrate that the proposed DTW temporal fusion agent outperforms HMM based systems, and has the capability to perform online temporal fusion efficiently and accurately in real–time.

Acknowledgements

First of all, I would like to thank my supervisor, Professor Geoff West, for his invaluable advices and comments on the thesis and all the published works. This thesis could not have been possible without the encouragement from him.

I would like to thank my supervisor, Professor Svetha Venkatesh, for her support and supervision. I would never have the opportunity to work for IPOM without her supports, and I would not be able to find a path in my research without her guidance.

I would like to thank my co-supervisor, Professor Mohan Kumar, for his ideas in sensor networks and data fusion. I am deeply appreciated for his support that allowed me to have the invaluable opportunity to visit him at The University of Texas, Arlington, and exchange research ideas with colleagues over there.

I would like to thank my family. They always encourage me to stand up where I fall, and not to give up my study. I am so grateful to have their endless love and support throughout my journey of life.

I would like to thank my dearest wife Grace Tseng for her love, care, patience, understanding, support, and encouragement. The coffee she made is full of love, and always inspires me.

Finally, I really felt an immense gratitude to all of you – Thank you.

Table of Contents

List of Figures

List of Tables

Published Work

This thesis is based upon three papers that have been published over the course of the research as listed below:

Conference papers:

- Ko, M. H., G. West, S. Venkatesh, and M. Kumar. 2005. *Workshop on Information Fusion and Dissemination in Wireless Sensor Networks (SENSORFUSION 2005) co-located with WICON 2005, July, 2005: Temporal Data Fusion in Multisensor Systems Using Dynamic Time Warping*. Budapest, Hungary
- Ko, M. H., G. West, S. Venkatesh, and M. Kumar. 2005. *Proceedings of the Second International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2005), December, 2005: Online Context Recognition in Multisensor Systems using Dynamic Time Warping*. Melbourne, Australia

Journal paper:

- Ko, M. H., G. West, S. Venkatesh, and M. Kumar. 2008. Using dynamic time warping for online temporal fusion in multisensor systems. *Information Fusion* 9 (3): 370-388.

Chapter 1 Introduction

Multi-sensor Data fusion is a diverse field that includes the theory, techniques, algorithms, and applications for optimal information processing in multi-sensor environments through intelligent integration of the multi-sensor data. Usually, data acquired from multiple sensors is fused at a variety of levels:

- The *raw data* level fuses the raw sensor data directly. If the multi-sensor data are commensurate (i.e., if the sensors are sensing the same physical phenomena such as two acoustic sensors in the same room). On the other hand, if the sensor data are not commensurate then the data must be combined at the feature vector level or decision level.

- The *feature* level extracts the feature vectors from sensor data, with subsequent fusion of the feature vectors into a single concatenated feature vector that is an input to pattern recognition techniques such as decision trees, neural networks, clustering algorithms, or self-organising maps.

- The *decision* level processes the sensor data to achieve high-level inferences or decisions, which are subsequently combined by methods such as weighted decision methods, voting techniques, Bayesian inference, and Dempster-Shafer theory.

Dasarathy (1994) further expands the three level hierarchy of fusion discussed above into five fusion process modes according to the joint input/output characteristics (refer to section 2.1.4 for details on these five fusion process modes). In addition to these five categories, this thesis suggests *Data In/Decision Out* should be considered. If the optimal feature extraction technique cannot be determined for a given data set, applying feature extraction might lead to information loss rather than improving the accuracy of classifiers. Thus the Data In/Decision

Out fusion process is required to go straight from the raw data to decisions without extracting features.

In performing sensor fusion, the aim is to associate, correlate, and combine data from multiple sensors and related information to achieve more specific inferences than could be achieved by using a single, independent sensor. Broadly speaking, multi-sensor data fusion may improve the performance of the system in four different ways as listed by Bellot, Boyer, and Charpillet (2002):

- **Representation**. Data fusion combines the information obtained to form an abstract level that provides a richer semantic on the data than each initial source of information.
- **Accuracy**. Data fusion can reduce or eliminate noise and errors in data.
- **Completeness**. Data fusion achieves a better situation awareness by integrating the new information into the current knowledge of an environment.
- **Certainty**. Data fusion can improve the prior probability of the sensor data. In general, the gain in accuracy and the gain in certainty are correlated.

Hence, over the past decade, multi-sensor fusion has received significant attention for a wide range of applications. The need for multi-sensor fusion has been recognised in many applications such as automatic target tracking (Aziz, Tummala, & Cristi, 1999), autonomous vehicles detection (Jouseau & Dorizzi, 1999), surveillance systems (Opitz, Henrich, & Kausch, 2004), automated manufacturing (Morriss, 1994), robotics (Castellanos & Tardós, 1999), fault detection and diagnosis of manufacturing processes (Venkatasubramaniam, 2003), wearable computers (DeVaul, Sung, Gips, & Pentland, 2003), and context–aware systems (Mantyjarvi, 2003). Many of these multi-sensor systems require not only fusion over sensors at specific points in time but also over periods of time. This is because the inference about the behavioural aspects of the environments or objects that are monitored by the multi-sensor systems is required, rather than merely their states at specific points in time or space. For instance, monitoring systems for batch processes (Lee, Yoo, & Lee, 2003), in which the abnormal processes are due to several causes over time; wearable computers for detection of human activities (Chambers, Venkatesh, West, & Bui, 2004; Krause, Siewiorek, Smailagic, & Farringdon, 2003), in which the complete gestures or activities are performed over periods of time; and context–aware mobile phones (Mantyjarvi, 2003), in which the user scenarios to be detected consist of different sub–actions over time. These systems require the accommodation of time into fusion processes to provide a window into the temporal arrangement of events, and the ability to suggest cause and effect among these events. However, the traditional fusion processes mentioned in Dasarathy (1994) do not consider these temporal relationships among events. Thus, there is a need for temporal fusion – the ability to fuse the raw data, features or decisions over a period of time to infer relationships of contextual and temporal proximity.

Temporal fusion over multiple heterogeneous sensors is a challenge. First, the complex multimodal sequences acquired are multi–dimensional, multimodal, interacting, time–varying, and are typically a mix of binary, discrete, and continuous variables. Second, the same class of multimodal sequences might vary in length locally e.g., the same human gestures and sub–gestures can be both performed at different speeds (Chambers et al., 2004), or the same batch processes can have different durations. Therefore, the sequences acquired can have similar waveforms but be different in length. This local variation further leads to global variation in length e.g., several continuous multi-sensor data streams that have the same gestures embedded might all have different lengths due to the variation in length of the same gesture. These variations make temporal fusion a difficult task. Moreover, performing temporal fusion efficiently in real–time is another challenge due to the large amounts of multi–sensory information. For example, some gesture data (Chambers et al., 2004) is described by six channels of continuous data including 1200 6-dimension vectors over 8 seconds; some user scenarios in the benchmark dataset (Mantyjarvi, Himberg, Kangas, Tuomela, & Huuskonen, 2004) are described by 29 variables (mix of binary, discrete and continuous variables) producing 284 29-dimension vectors over 5 minutes.

Most state of the art approaches tried to cope with this challenge by using a statistical framework (Damarla, 2008). The main advantage of a statistical approach is that explicit probabilistic models are employed to describe the various relationships between sensors and sources of information taking into account the underlying uncertainties. This thesis proposes a non-statistical based framework to solve the challenges of online temporal fusion, and the results are compared with statistical based approaches.

1.1 Aim and Approach

This thesis investigates the approaches for solving the challenges of temporal fusion in the field of multi-sensor data fusion. Our objectives are:

- To recognise and classify the complex multimodal sequences acquired from multiple heterogeneous sensors and related information to achieve more specific inferences than could be achieved by using a single, independent sensor.

- To recognise and classify the complex multimodal sequences acquired from multiple heterogeneous sensors efficiently at each sampling interval, in real-time.

- To establish a framework for online temporal fusion of the complex multimodal sequences acquired from multiple heterogeneous sensors.

To achieve the above aims, we investigated a bottom-up approach. First, two real-world benchmark datasets were selected for our experimental study: (1) six channels of continuous data acquired from accelerometers by performing two hand gestures (Chambers et al., 2004), and (2) a public dataset with 29 variables acquired from multiple heterogeneous sensors by carrying a mobile device and performing the pre-defined user scenarios (Mantyjarvi et al., 2004). Second, three prototypes were built using *Dynamic Time Warping* (DTW) (Sakoe & Chiba, 1978), a *Self-Organizing Map for structured data* (SOMSD) (Hagenbuchner, Sperduti, & Tsoi., 2003), and a *Hidden Markov Model* (HMM) (Rabiner, 1989) as the recogniser respectively. We then selected some segmented sequences with one dimension from the two datasets, and used the three prototypes to classify the selected sequences. Their classification rates are compared, and DTW outperformed both the SOMSD and the HMM.

Based on the results of the bottom-up approach, we proposed the use of *Dynamic Time Warping* (DTW) – a general time alignment and similarity measure for two one dimensional sequences, to construct a robust system framework for solving the challenges of temporal fusion. We theoretically extended the original *Dynamic Time Warping* introduced by Bellman (Bellman, 1957) to be the core recogniser of the proposed system framework and achieved temporal fusion in real-time. Thus the specific aims are as follows:

- To extend the DTW recogniser to deal with multimodal sequences which consist of the data or feature sequences acquired from multiple heterogeneous sensors over a period of time.

- To extend the DTW recogniser to be able to recognise the same class of multimodal sequences even though they have different lengths locally and/or globally.

- To extend the DTW recogniser to deal with the problem of unknown endpoints of "multimodal sequences" embedded in real-time continuous data streams. Most techniques for sequence recognition (e.g. HMM) require knowing the accurate locations of the endpoints for reliable and robust recognition. The DTW non–linearly warps one time sequence to match another given the start and end point correspondences, which means accurate location of the endpoints is also needed. DTW has been extended to deal with unknown start and end points of isolated words in speech (Myers, Rabiner, & Rosenberg, 1980b; Rabiner, Rosenberg, & Levinson, 1978), and connected word recognition (Myers & Rabiner, 1981; Myers, Rabiner, & Rosenberg, 1980a; Rabiner & Myers, 1981). However, the endpoint detection or segmentation problem is nontrivial and has been studied thoroughly in the speech recognition community, where "silence" between utterances is a useful clue (Crochiere, Tribolet, & Rabiner, 1981; Wilpon & Rabiner, 1987). Unlike continuous speech signals that only have one dimension for mono or two dimensions for stereo, it is more difficult to develop endpoint detection algorithms for the multi-dimensional and multimodal sequences. We will extend the DTW recogniser to solve this endpoint challenge in this thesis.

- To develop a robust and operational system framework that uses the enhanced DTW recogniser to achieve the *data input* or *feature in/decision out* fusion process on the complex sequences in real-time.

- To evaluate the performance of the proposed system framework, we use two real world benchmark datasets to see if the system can perform multi-sensor temporal fusion efficiently, coping with large amounts of multi–sensory information in real–time, and make classification decisions with high accuracy.

- To evaluate the robustness of the proposed system framework, the experimental results for two real world datasets are compared with the results from a Hidden Markov Model (HMM) based approach.

1.2 Significance and Contribution

This thesis makes two main contributions to the field of *Multi-sensor Data fusion*. First, we developed a robust and operational system framework that can solve the challenges of temporal fusion for complex multimodal sequences in real-time. The contributions and significance are:

- To the best of our knowledge, there has been little work on applying *Dynamic Time Warping* (DTW) (Sakoe & Chiba, 1978) in the field of Multi-sensor Data fusion. DTW is a general time alignment and similarity measure for two temporal sequences that was first introduced by Bellman (Bellman, 1957), and has been a popular research topic in the speech recognition community (Myers et al., 1980a). More recent research on DTW has been focused on applying it to mining patterns from one–dimensional time series (Keogh, 1997), and indexing and clustering for one dimensional time series (Keogh & Ratanamahatana, 2004; Lin, Vlachos, Keogh, & Gunopulos, 2004). In this thesis, we have successfully applied DTW for temporal fusion, which includes the developments of data pre-processing procedures, novel training methods for the DTW recogniser, DTW extensions, and decision making procedures. The results were validated with the real-world benchmark datasets, and demonstrated that the proposed system framework is robust and efficient.

- A novel multi-agent framework for temporal fusion. The proposed multi-agent temporal fusion framework includes three major layers: *hardware*, *agents*, and *users*. To differentiate our framework from other fusion frameworks in the literature, we emphasised the application of DTW temporal fusion agents for aggregation of information over time, and the three different architectures: *centralized*, *hierarchical*, and *distributed* for organising the DTW temporal fusion agents. In addition, we use an *information graph* to show the information flows among the DTW temporal fusion agents, and how the agents fuse the sensor data over the time dimension.

- A novel DTW based system framework for online human gesture recognition. The gesture dataset was captured by a person wearing watch-like wristbands (consisting of accelerometers) to record arm movement. The dataset consisted of a person mimicking the 12 gestures of a cricket umpire (Chambers et al., 2004). The proposed system can achieve a 97.95% classification rate for the 12 gestures. The experimental setup and results are detailed in section 4.2.

- A novel DTW based system framework for online context recognition of mobile applications. The context dataset has been proposed as a suitable benchmark for evaluating context recognition algorithms (Mantyjarvi et al., 2004). The dataset is obtained from sensors, including *3 axes accelerometers*, an *illumination sensor*, a *thermometer*, a *humidity sensor*, a *skin conductivity sensor*, and a *microphone*, placed in a small sensor box that is then attached to a mobile handheld device. The proposed system can achieve 97.5% classification rate for the five predefined user scenarios, which outperformed the results presented in Himberg et al (2003). The experimental setup and results are detailed in section 4.3.

- The most commonly used technique for temporal fusion or sequence recognition is the Hidden Markov Model (HMMs) (DeVaul et al., 2003; Hossain & Jenkin, 2005; Kallio, Kela, & Mantyjarvi, 2003; Rabiner, 1989; Wilson & Bobick, 2000). However, we demonstrate that the training and recognition procedures of the proposed DTW base system are potentially much simpler and faster. In this thesis, we explored the differences between HMM and DTW approaches and show the better performance of the DTW based approach. The comparisons are detailed in section 4.2.3 and section 4.3.1.

Second, we extended the original Dynamic Time Warping (DTW) to recognise the complex sequences which are multi–dimensional, multimodal, interacting, time–varying, and are typically a mix of binary, discrete, and continuous variables. The contributions and significance are:

- The original DTW is a general time alignment and similarity measure between two single dimension time series. We extend the DTW recogniser to deal with multi-dimensional and multimodal sequences acquired from multiple heterogeneous sensors over a period of time. We propose the extended *Euclidian distance* (Equation 3.1) and *Cosine correlation coefficient* (Equation 3.2) as local distance measures when performing the DTW algorithm.

- One of the major limitations in the original DTW algorithm is the “endpoint constraint” (refer section 3.5), which means accurate location of the endpoints of a target sequence is needed. However, the endpoint of a continuous stream is unknown when the data are sampled from multiple heterogeneous sensors in real time. We propose a variant of the original DTW algorithm which relaxes the endpoint constraints to solve this issue. The variant is described in section 3.5.2.

- We propose and explore the use of normalising factor (NF) in Equation 2.2 to allow the DTW recogniser to be able to recognise the same class of multimodal sequences even though they have different lengths locally or globally.

- In this thesis, we propose and explore the use of the DTW variant and the sliding windows operation to enable online temporal fusion. The details can be found in section 3.5.3.

1.3 Structure of the Thesis

This thesis is organised as follows. In Chapter 2, a survey of the related work in the fields of the formal frameworks for multi-sensor data fusion systems and the pattern recognition techniques for temporal fusion is presented.

Temporal fusion is an additional dimensionality to the fusion process that has not been emphasised in the reviewed data fusion frameworks. Thus, there is a need for a better temporal fusion framework. Chapter 3 describes our major contributions in develop a multi-agent framework for temporal fusion. In addition, the core of the proposed framework – the DTW temporal fusion agent, is presented.

Chapter 4 presents the experimental results on evaluating the performance of the proposed DTW temporal fusion agent. Two real-world datasets are used in the evaluation:

1. Two hand gestures dataset (Chambers et al., 2004).
2. Mobile context benchmark dataset (Mantyjarvi et al., 2004).

Performance results of the DTW temporal fusion agent are also compared with those of a Hidden Markov Model (HMM) based system.

Finally, Chapter 5 provides a summary of this thesis and discusses the potential future directions for improving the DTW based multi-agent framework for temporal fusion.

Chapter 2 Background

Fusion is a fundamental human process that occurs in some form at all levels of sense organs such as visual and sound information received from eyes and ears respectively, to the highest levels of decision making such as our brain that fuses visual and sound information to make decisions. In the context of multi-sensor data fusion, fusion is a way of processing, extracting, and combining data and information content about the world and the environment that is relevant to a human and the decisions that must be made.

This chapter provides a survey of both multi-sensor data fusion frameworks and pattern recognition techniques for temporal fusion. The section on frameworks presents some popular data fusion frameworks that are widely used for designing multi-sensor data fusion systems. The frameworks:

- identify the component functions that the data fusion systems have,
- define the connectivity among these components, and the data or information flows between them, and
- arrange these component functions as a set of processes that should be undertaken to make the data fusion system fully operational.

The core component function of a data fusion framework is the algorithm that can process and fuse the multi-sensor data. The section on pattern recognition techniques for temporal sequences focuses on data fusion algorithms that can be embedded in the context of a larger data fusion framework. Examples of pattern recognition techniques that can be used in data fusion frameworks for temporal fusion are hidden Markov models, dynamic time warping, and self-organizing maps for sequences.

2.1 Multi-sensor Data Fusion Frameworks

Multi-sensor data fusion systems are often of a complexity that requires the use of a formal framework and standards to specify data fusion processing and control functions, interfaces to sensor hardware and users, identification of the processes, categorisation of techniques, and specific techniques applicable to data fusion. The lack of a formal framework and standards for data fusion systems has been a major issue to integration, unification of terminology, and reuse of available technology. Hence, there is a need to formalise the framework and standards that crosses application-specific boundaries to improve communications among researchers, and to facilitate the cost-effective development, acquisition, integration and operation of multi-sensor data fusion systems.

2.1.1 Joint Directors of Laboratories (JDL) Data Fusion Framework

In 1986, the US Department of Defence established the *Joint Directors of Laboratories* (JDL) *Data Fusion Working Group* to begin the effort to formalise the framework and standards. The result of this effort was the creation of the *JDL data fusion framework* (Hall & Llinas, 1997), and this framework has been the most widely used for categorizing data fusion-related functions since then. However, the original JDL data fusion framework was designed with a military flavour. The more recent revised JDL data fusion framework have extended this limitation by broadening the functional model, relating the taxonomy to fields beyond the original military focus (Llinas et al., 2004; Steinberg, Bowman, & White, 1999). Furthermore, Bowman suggested the revised JDL data fusion framework as shown in Figure 2.1, which is designed to generalise the framework so that it can be useful across multiple application areas such as industrial, medical, wearable, ubiquitous, and pervasive systems (Bowman, 2004; Bowman, Steinberg, & White, 1999; Llinas et al., 2004).

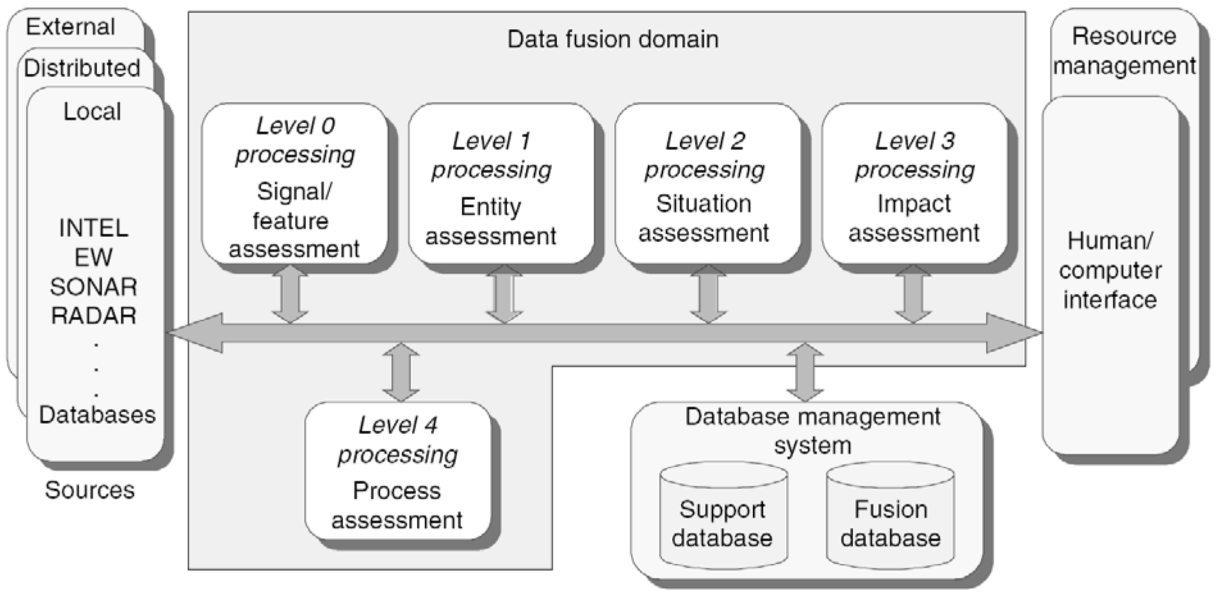

External
Distributed
Local
INTEL
EW
SONAR
RADAR
Databases
Sources
Data fusion domain
Level 0 processing Signal/ feature assessment
Level 1 processing Entity assessment
Level 2 processing Situation assessment
Level 3 processing Impact assessment
Level 4 processing Process assessment
Database management system
Support database
Fusion database
Resource management
Human/ computer interface

Figure 2.1: The revised Joint Directors of Laboratories (JDL) framework for data fusion. From: (Bowman, 2004)

The elements of the revised JDL data fusion framework are:

- **Sources**. The sources are the inputs to the data fusion systems, and provide information from sensors, prior information, databases, or human input.

- **Human/Computer interface**. This part provides an interface for human input and ensures all the information transmitted to the decision maker is transformed into a form which is intuitively usable for the decision-making process.

- **Database Management System**. The task of the database management system is to monitor, evaluate, add, update, and provide information for the fusion processes.

- **Level 0** (estimation of features). ***Signal/feature assessment*** is aimed at pre-processing data to correct biases, perform spatial and temporal alignment, standardize and normalise inputs, and extract features.

- **Level 1** (estimation of states of an entity). ***Entity assessment*** (i.e. an entity is considered as an individual) is aimed at associating and combining sensor data to infer the most reliable and accurate estimate of an entity's states of interest.

- **Level 2** (estimation of relationships among entities). ***Situation assessment*** is aimed at inferring from the estimated states of one entity in a situation to another and from the estimated attributes and relationships of entities to situations.

- **Level 3** (prediction of impacts). ***Impact assessment*** is aimed at determining the expected consequence of system plans or courses of action, given the current estimated entity's state and situational state.

- **Level 4** (estimation of prediction versus reality). ***Process assessment*** is aimed at monitoring the overall data fusion process to assess and improve real-time system performance as compared to desired states and measures of effectiveness.

For example, the signal features (e.g., mean & standard deviation) of one entity (e.g., the cricket umpire wearing the wristbands that consist of accelerometers) can be extracted from the entity's sensor observations (e.g., accelerometer data) via a **Level 0** data preparation process. The activity state of an entity (e.g. a gesture of the cricket umpire) can be estimated on the basis of attributes inferred from sensor observations; i.e. via a **Level 1** data preparation, association, and recognition process. The same entity's compositional or relational state (e.g. a gesture of the cricket umpire and its relations with the cricket team) can be inferred via a **Level 2** data association and recognition processes. Thus, a single entity – anything with internal structure, whether man, machine, or building – can be treated either as an individual, subject to a **Level 1** observation and state estimation – or as a "situation", subject to relational analysis via a **Level 2** entity to entity association and aggregate state recognition. The impact of a signal, entity, or situation on the user goal or mission can then be predicted based upon an association of these to alternative courses of action for each entity via a **Level 3** *reinforcement learning* process (Sutton & Barto, 1998). A **Level 4** function associates the system states and actions to desired system states and estimating or predicting the performance of the system.

In general, these five levels are different from each other in terms of the types of input data, models, outputs, and inferences involved in each level. The levels are not necessarily processed in order, and any level can be processed on its own given the corresponding inputs.

2.1.2 Waterfall Data Fusion Framework

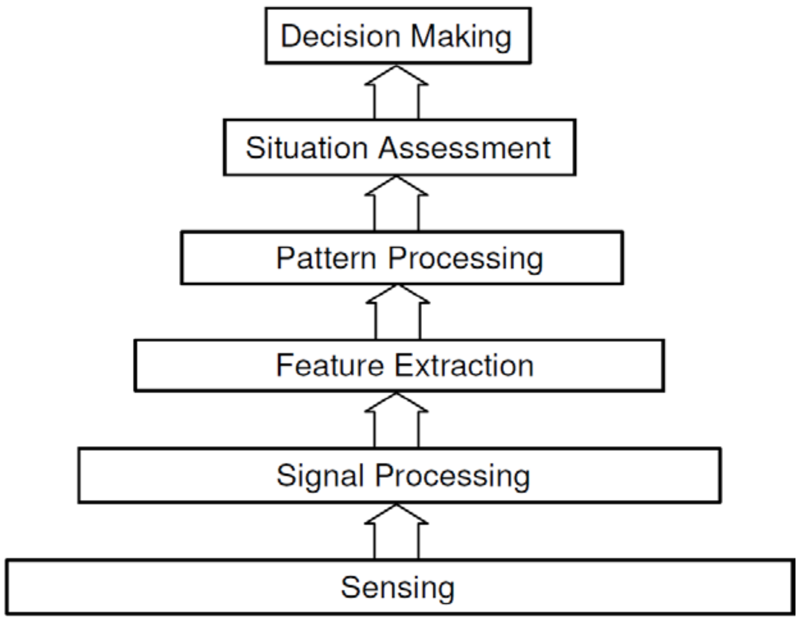


Figure 2.2: The Waterfall data fusion framework. From: (Markin et al., 1997)

Markin (1997) proposed another hierarchical framework commonly used by the data fusion community, called the *waterfall model* as depicted in Figure 2.2. This framework emphasises the flow of data operations from the data level to the decision making level. The stages relate to the levels 0, 1, 2, and 3 of the JDL model as follows:

- *Sensing*, *signal processing*, and *feature extraction* correspond to the Signal/feature assessment of the reviewed JDL level 0. The raw data is properly transformed to provide the required information about the environment sensed by sensors.

- *Pattern processing* matches the Entity assessment of the JDL level 1. This stage processes and fuses the features to obtain a symbolic level of inference about the data, and to minimise the data content whilst maximising information delivered.

- *Situation assessment* is similar to the situation assessment of the reviewed JDL level 2, and *decision making* corresponds to the Impact assessment of the reviewed JDL level 3. These stages relates object to the situations. Possible actions to take are predicted according to the information that has been gathered, the libraries and databases available, and the human interaction.

The waterfall data fusion framework is more exact in analysing the fusion process than other models. However, the major limitation of the waterfall model is the omission of any feedback information flow to advise the multi-sensor system on re-calibration, re-configuration and data gathering aspects. The waterfall model has been used in the defence data fusion community in Great Britain, but has not been significantly adopted elsewhere (Bedworth & Brien, 1999).

2.1.3 Boyd OODA Data Fusion Framework

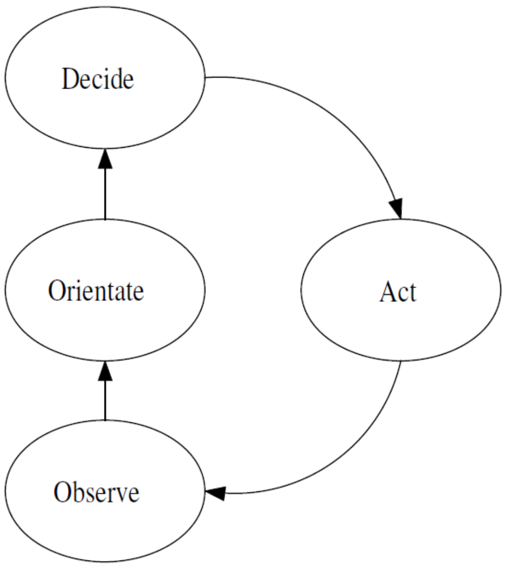


Figure 2.3: The Boyd OODA loop data fusion process model. From: (Boyd., 1987)

Boyd (1987) has proposed a *control cycle* or *OODA loop* that contains four phases as depicted in Figure 2.3. The OODA loop was first used for the classic decision-support mechanism in military information operations, but has since been widely used for data fusion because the decision-support systems for situational awareness are tightly coupled with fusion systems.

The four phases of Boyd's framework have distinct similarities with the reviewed JDL frameworks (Bedworth & Brien, 1999):

- **Observe –** is broadly comparable to signal assessment in the level 0 of the reviewed JDL framework.

- **Orient** – corresponds to functions of the levels 1 and 2 of the reviewed JDL framework.
- **Decide** – corresponds to level 3 of the reviewed JDL framework. It also includes much more e.g., logistics and planning.
- **Act** – has no direct counterpart in the reviewed JDL framework, because the JDL framework does not close the loop by taking the actuating part of the interaction into account.

The advantage of this framework is the fact that it closes the loop, and the actuators act on its environment and sensors. However, the framework does not show an appropriate structure for identifying and separating different sensor fusion tasks.

2.1.4 Dasarathy Data Fusion Framework

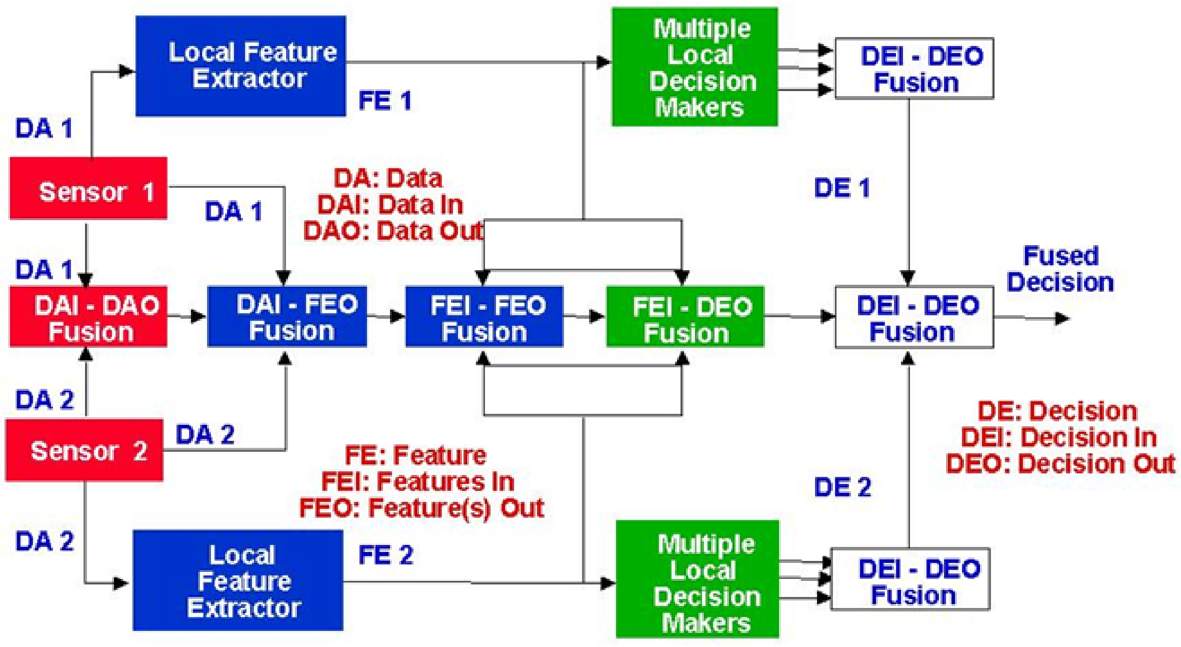


Figure 2.4: Flexible Global Fusion System Framework. From: (B.V. Dasarathy, 1997)

Dasarathy (1994, 1997, 1998, 2000) proposed the flexible global fusion system framework based on the I/O characteristics of fusion processes. Figure 2.4 shows the proposed framework which demonstrates the different ways of incorporating the following five fusion processes:

- **Data In/Data Out (DAI-DAO)**. This fusion process is accomplished at the front end of the processing stream to combine information from sensors with compatible data rates, data dimensionality and formats. For example, data pre-processing such as filtering and smoothing can be applied immediately on acquisition of the data from the different sensors.

- **Data In/Feature Out (DAI-FEO)**. Features are generated from the input data e.g. edge detection in an image. Data from different sensors are combined to derive some form of a feature of the object in the environment or a descriptor of the phenomenon under observation.

- **Feature In/Feature Out (FEI-FEO)**. Both the input and output of this fusion process are features. For example, shape features obtainable from an image can be combined with the range information obtainable from the radar sensor to derive a measure of the volumetric size of a target. This is one of the typical fusion processes at the feature level.

- **Feature In/Decision Out (FEI-DEO)**. Input features are fused together to give output decisions. Most pattern recognition systems involving inputs from multiple sensors belong to this form of fusion process. In the recognition phase, the feature vector is classified on the basis of a priori knowledge and/or training, to arrive at a class label (decision).

- **Decision In/Decision Out (DEI-DEO)**. Multiple input decisions are fused together to give a final output decision. This fusion process combines the decisions made at the local level at the fusion centre.

Figure 2.4 shows various fusion process flows. For example, one of the fusion process flows (top and bottom of Figure 2.4) show that each sensor has its own local feature extractor to perform the DAI-FEO fusion process, and the resulting feature is fed into its own local decision maker to perform the FEI-DEO fusion process. The decisions from each local decision maker are further fused to produce the final fused decision (DEI-DEO). One advantage of the DEI-DEO fusion process is that it is most tolerant to individual sensor subsystem failures, because even if no decision is made from one of the sensor subsystems, the others can still provide a decision to produce the final fused decision even if not at the same level of reliability. On the other hand, both DAI-DAO and DAI-FEO fusion processes are not only sensitive to the characteristics of the individual sensors (e.g., replication or complement), but also susceptible to individual sensor failures. This is because the final decision is not only affected by the performance of every sensor, but also the overall system may become incapable of making decisions in case of a failure of one or more of the sensor subsystems. In contrast, the FEI-DEO fusion process could derive reasonably good decisions with only a partial feature set but with much less reliability than one that uses the entire feature set.

To design the optimal fusion system, the potential for fusion in all the five fusion processes should be fully explored to take advantage of the information obtained from all the available sensors in the environment. In addition, the mix of inputs at different levels should be considered, for example, a mix of data and feature inputs to a fusion process with features as output can also be possible under some scenarios. Similarly, features from some sensors and decisions from the same or other sensors can be used as input to a fusion process whose output is at the decision level.

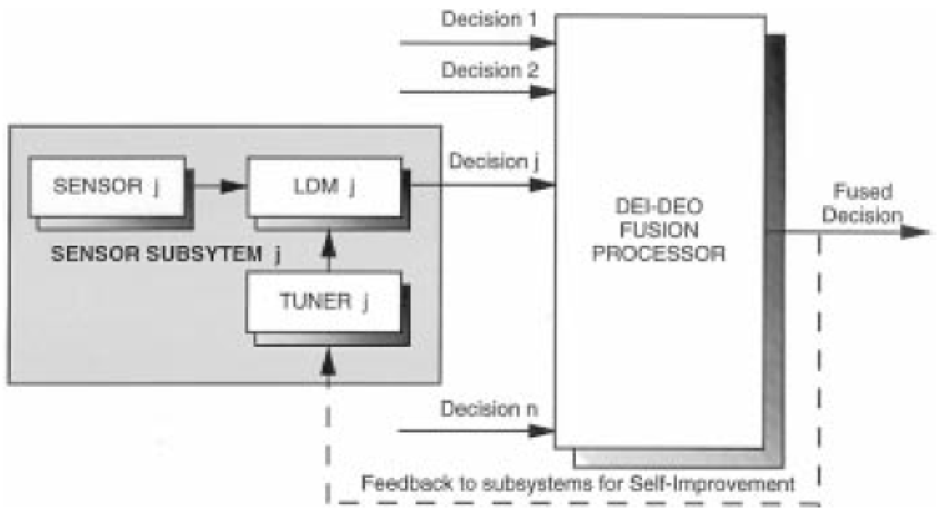


Figure 2.5: A self-improving multisensor fusion system framework. From: (B.V. Dasarathy, 1997)

In addition to the flexible global fusion system framework, Dasarathy (1998, 2000) also proposed the fusion system with self-improvement potential through mutual reinforcement (Sutton & Barto, 1998) between the central (DEI-DEO) fusion processor and the local sensor subsystems as depicted in Figure 2.5. The central (DEI-DEO) fusion processor not only fused the local decisions to form the final decision, but also sent the feedback to the sensor subsystems to improve their performance over time. The performance of sensor subsystems are improved by the tuners taking the feedback to fine-tune the local decision makers (LDM) as needed to reinforce their correct decisions and punish their wrong decisions so their decision making ability is improved. This improved performance of the sensor subsystems will in turn enhance the performance of central (DEI-DEO) fusion processor. This mutual reinforcement will tend to improve the overall system performance over time.

2.1.5 Omnibus Data Fusion Framework

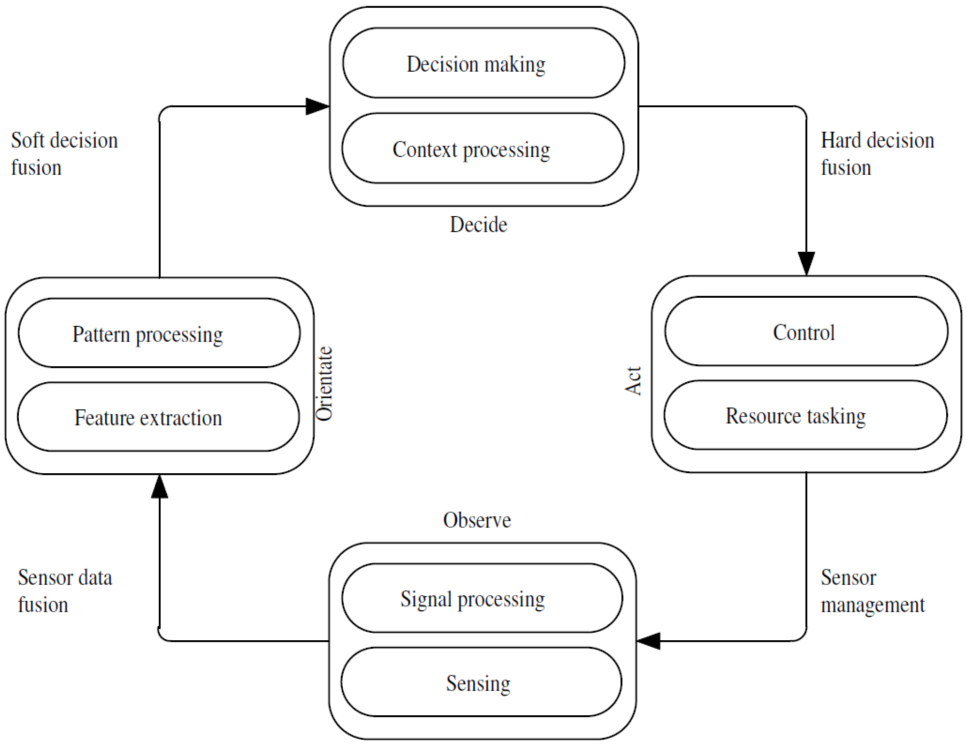


Figure 2.6: The Omnibus Data Fusion Framework. From: (Bedworth & Brien, 1999)

Bedworth and O’Brien have proposed the omnibus framework (Bedworth & Brien, 1999) as depicted in Figure 2.6. This framework is a hybrid of four other frameworks. It has the cyclic nature of the *Boyd OODA loop* but incorporates the finer definitions expressed by the Waterfall framework into each of the four main phases, and each phase can be associated with one of the levels in the reviewed JDL and Dasarathy frameworks.

The framework is intended to be used multiple times in the same application recursively at two different levels of abstraction. Firstly, it characterises and sub-divides the overall system with the aim of providing an ordered list of tasks. Secondly, the same structure may be used to organise the functional objectives of each such task. Using this approach a data fusion solution is categorised using a dual perspective – both by its system aim (i.e., sensing, situation and impact assessment) and its task objective (i.e., actuating and resource management).

There are two drawbacks of this framework (Shahbazian, ve, & Labbe, 2001). First, it combines too many concepts and types of other data fusion frameworks, which may cause the misconception that a specific architecture (e.g., soft decision, hard decision) is appropriate for only a certain specific phase of the loop. Second, the hierarchical separation of the sensor fusion tasks is very sophisticated in the omnibus model; it does not support a horizontal partitioning into tasks that reflect distributed sensing and data processing. Thus, the model does not support decomposition into modules that can be separately implemented, separately tested, and reused for different applications.

2.2 Pattern Recognition Techniques for Temporal Fusion

Temporal fusion is an additional dimension to the fusion process, which has not been emphasised in the previous data fusion frameworks. Temporal fusion is the fusion of data or information acquired over a period of time such as a simple temporal averaging of data acquired from different sensors is an example of the temporal fusion process. The temporal fusion process can be applied at any of the three levels of the hierarchy (i.e., Data/Feature/Decision levels) discussed earlier and hance in effect represents a horizontal dimension to the fusion process.

Sequential pattern recognition is another good example of the temporal aspects of the fusion process. Many algorithms for classification of temporal sequences have been developed and have been evaluated in the various domains, especially in the speech recognition, sound identification, and bioacoustics domains (N. Rabiner & Juang, 1993). We can group the large variety of algorithms into two groups:

1. Algorithms that make use of knowledge or assumptions about the temporal sequences to be recognised. For example, dynamic time warping (DTW) is one of the representatives of this group of algorithms, since it uses a class sequence to compare the similarity with the temporal sequences to be recognised (i.e., the training step is

achieved by selecting a class sequence that best represents the temporal sequence to be recognised instead of learning some parameters from the training dataset).

2. Algorithms that are able to learn temporal contexts such as self-organising maps for sequences (SOMSD) and hidden Markov models (HMM). The training step of this group involves learning of the parameters from the training dataset to build a representative model, the model is then used in the classification step to recognise the temporal sequences.

Both groups of algorithms have to take into account the following variability in performing the recognition:

- The temporal variability i.e., the same class of temporal sequences might have different lengths. The DTW uses both the warping paths and dynamic programming to cope with this temporal variability. On the other hand, the HMM utilises different state sequences (the states can be skipped or repeated) to deal with the variability.

- The variability of temporal sequence in the data or feature space i.e., to measure the difference in data or feature vectors. The DTW evaluates this variability by calculating Euclidian distances between data or feature vectors. Unlike the Euclidian distance that calculates the distance between the actual value of data or feature vectors, the HMM measures the likelihood by using probability density functions (pdf) of data or feature vectors.

In this section, we will review the DTW algorithm from the first group of algorithms and HMM and SOMSD from the second group.

2.2.1 Dynamic Time Warping (DTW)

The classic DTW algorithm uses a local distance measure to determine the distance between a class sequence and a test sequence by calculating a warping path on the DTW distance table. To formulate the problem of solving temporal fusion using a DTW recogniser, suppose we have:

- A test sequence $T(j)_{j=1}^{J}$ of length J, with $T(j) \in \boldsymbol{R}$ – real number.
- Several class sequences, each one represented as $C(i)_{i=1}^{I}$ of length I, with $C(i) \in \boldsymbol{R}$ – real number.

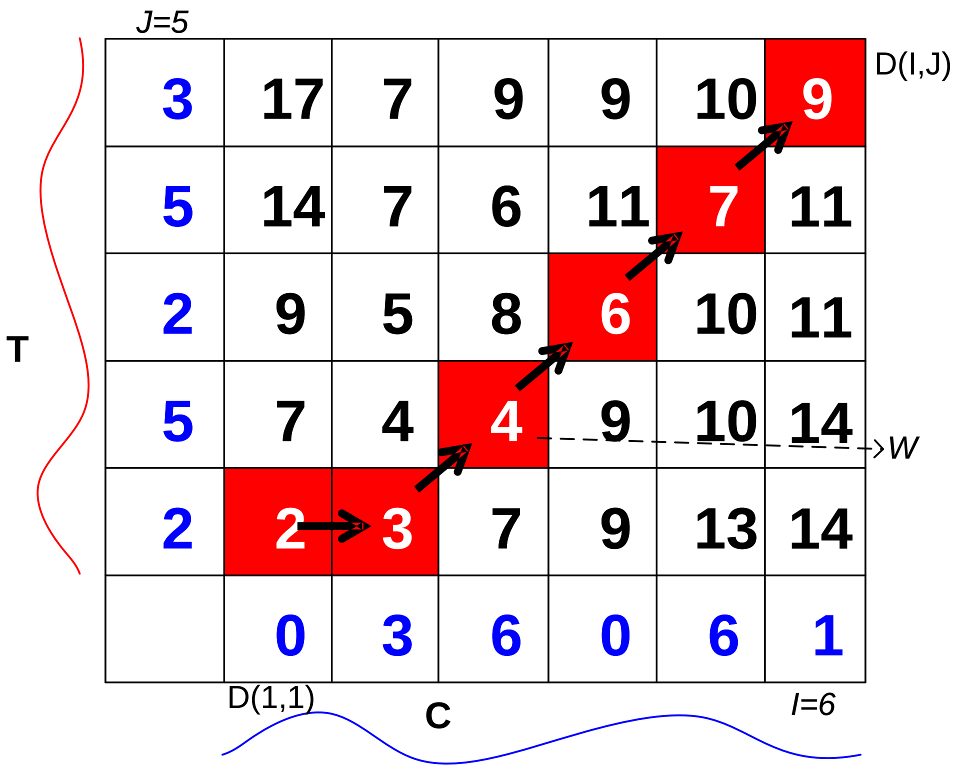


Figure 2.7: Example of a warping path in a distance table. Length of class sequence, $I = 6$ and length of test sequence, $J = 5$.

We want to know to which class the test sequence belongs to by measuring the similarity. To measure the similarity between the class and test sequences, an $I \times J$ distance table D is constructed, where $d(i,j)$ is the local distance between $C(i)$ and $T(j)$ as depicted in Figure 2.7. Typically, the Euclidean distance is used to measure these local distances, thus $d(i,j) =$

$(C(i) - T(j))^2$. A warping path W is then calculated from the distance table which consists of a set of table elements that defines a mapping and alignment between $C(i)$ and $T(j)$:

$$W = \left\{ w(i(q), j(q)) \middle| \begin{array}{c} q = 1, \dots, Q, \\ max(I,J) \leq Q \leq I + J - 1 \end{array} \right\}$$ **Equation 2.1**

with $i(q) \in \{1, \dots, I\}$ and$j(q) \in \{1, \dots, J\}$. The warping path is restricted by the following constraints (Sakoe & Chiba, 1978):

- **Continuity**: Given $w_q(i(q), j(q))$ and $w_{q-1}(i(q-1), j(q-1))$ where $i(q) - i(q-1) \leq \alpha$ and $j(q) - j(q-1) \leq \alpha$ with α an integer.
- **Endpoint**: The path must start at $i(1) = 1,\ j(1) = 1$ and finish at $i(Q) = I,\ j(Q) = J$.
- **Monotonicity**: Given $w_q(i(q), j(q))$ and $w_{q-1}(i(q-1), j(q-1))$ where $i(q) - i(q-1) \geq 0$ and $j(q) - j(q-1) \geq 0$.

The overall distance $DTW(C, T)$ between the class sequence and the test sequence is then calculated by summing the local distances over the warping path W. One popular choice for finding the best alignment between the class sequence and the test sequence is to search for the path with the minimal DTW distance over all possible warping paths:

$$DTW(C,T) = \underset{W}{arg\, min}\left(\frac{\sum_{q=1}^{Q} w(i(q), j(q))}{NF}\right)$$ **Equation 2.2**

where NF is a normalising factor to compensate for the fact that warping paths may have different lengths. Some common normalising factors have been discussed by Ratanamahatana and Keogh (2004a). We will discuss them further in Section 3.5.3.

During the calculation of the warping path between the class sequence and a set of test sequences, we would expect the warping path of the long test sequences to be longer than for the short test sequences, because of the endpoint constraints. We will discuss the method for choosing NF in Section 3.5.3. The optimal warping path can be found efficiently by using dynamic programming to evaluate the following recursive step:

$$D(i,j) = d(i,j) + min\begin{Bmatrix} D(i-1,j-1), \\ D(i-1,j), \\ D(i,j-1) \end{Bmatrix} \qquad \textbf{Equation 2.3}$$

This recursion is generally initialized as $D(1,1) = d(1,1)$. When calculating cells that are in the first row of D, only horizontal path extensions are considered (i.e., only $D(i-1,j)$ is considered). Similarly, when calculating cells that are in the first column of D, only vertical path extensions are considered (i.e., only $D(i,j-1)$ is considered). Finally, the recursion terminates when $i = I$ and $j = J$. The time and space complexity of this dynamic programming approach is $O(IJ)$.

In Equation 2.3, the warping paths are restricted by applying *local constraints* which decide the set of adjacent cells to be considered at each recursive step. There are various local constraints that can be applied e.g., Itakura (1975); Sakoe & Chiba (1978); Myers (1980a); Rabiner (1993); Keogh & Ratanamahatana (2004), and each of them defines a different set of admissible adjacent cells at each recursive step. Many local constraints have been examined in Sakoe and Chiba (1978), including some asymmetric ones, where the order of the two sequences matters. Our DTW recogniser uses the local constraint as defined in Equation 2.3.

In addition to local constraints, more global ones can be applied. For example the *Band-DP* (Kuhn & Tomaschewski, 1983) only allows the path to be some constant distance – E from the diagonal joining the start and finish corners of the space. Another is the *Itakura parallelogram* (Itakura, 1975) that allows the path to be at increasing distance from the diagonal towards the centre of the search space. In each case, the cost in cells outside of these regions is set to infinity to prevent these cells from being chosen. The Band-DP global constraint with width E is used in this paper so the warping path can be no more than E cells from the diagonal. If $i = mj$ defines the line between the start and finish points, then

$i = mj + E$ and $i = mj - E$ define the limits of the band. The use of the Band-DP can further reduce the time and space complexity of the warping path computation from $O(IJ)$ to $O(IE)$. Additionally, applying the Band-DP with width E leads to better results in classification tasks (Ratanamahatana & Keogh, 2004b).

To the best of our knowledge, the DTW based fusion framework proposed in this thesis is the first attempt to apply DTW in the domain of multi-sensor data fusion. We have use the DTW based framework for human gesture recognition (Ko, West, Venkatesh, & Kumar, 2005b), and online temporal context recognition for situation awareness (i.e., level 2 – Situation assessment of the JDL fusion framework) (Ko, West, Venkatesh, & Kumar, 2005a, 2008)

2.2.2 Hidden Markov Models (HMM)

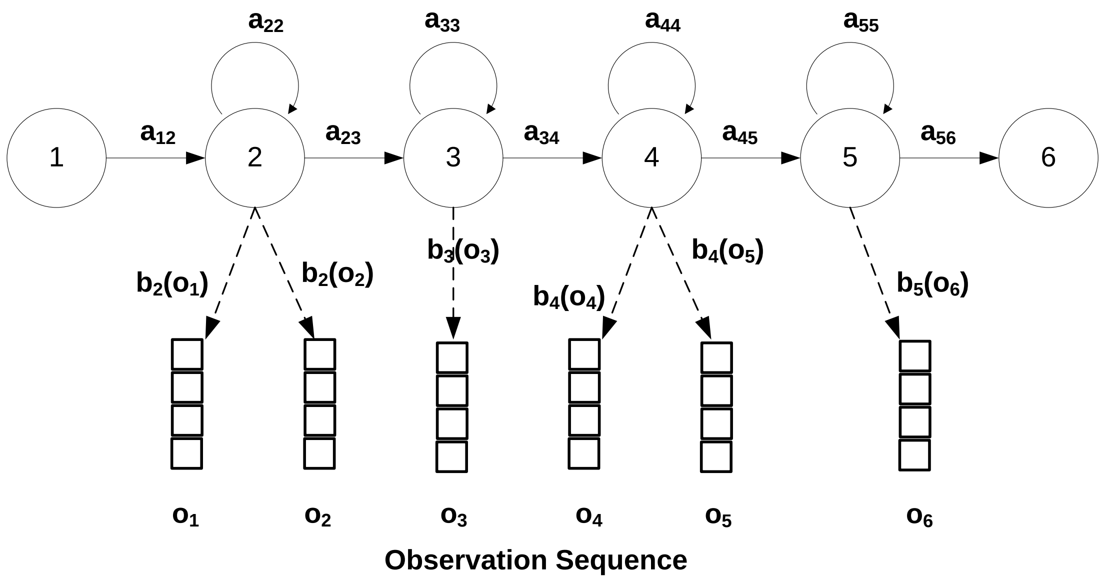


Figure 2.8: An example of Hidden Markov Model with five states

Figure 2.8 shows an example of a *Hidden Markov Model* (HMM). The HMM is the one of the most representative statistical approaches for temporal fusion or sequential data recognition (Murphy, 2002). The HMM model consists of the following parameters (Rabiner, 1989):

1. N, the number of states in the model e.g., Figure 2.8 has six states. Consider a model has N distinct states, $\{S_1, S_2, \ldots, S_N\}$. At regularly spaced discrete times, the system being modelled can change state to a new state or back to the same state. Denote the time

instants associated with the state changes as $t = 1,2, \ldots$, and denote the state variable at time t as q_t.

2. M, the number of distinct observation symbols per state e.g., the state 2 of Figure 2.8 has two observation symbols. The observation symbols are the physical output of the system being modelled. If the output is continuous (i.e., use mixture-of-Gaussians to model), M is infinite. The set of possible observation symbols is denoted as $V = V_1, V_1, \ldots, V_M$.

3. The state transition probability distribution $A = \{a_{ij}\}$ where

$$a_{ij} = P[q_{t+1} = S_j \,|\, q_t = S_i], \qquad i, j \in [1,2, \ldots, N] \qquad \textbf{Equation 2.4}$$

and q_t denotes the state at time t. The transition probabilities A satisfies the standard stochastic constraints $\sum_{\mathrm{j=1}}^{\mathrm{N}} \mathrm{a_{ij}} = 1, \mathrm{a_{ij}} \geq 0$.

4. The observation symbol probability distribution in state j, $B = \{b_j(k)\}$, where

$$b_j(k) = P[v_k \; at \; t \mid q_t = S_j]$$
$$j \in [1,2, \ldots, N], k \in [1,2, \ldots, M], t \in [1,2, \ldots, T] \qquad \textbf{Equation 2.5}$$

5. The initial state distribution $\pi = \{\pi_i\}$ where $\pi_i = P[q_1 = S_i], i \in [1,2, \ldots, N]$

$$\pi_i = P[q_1 = S_i], i \in [1,2, \ldots, N] \qquad \textbf{Equation 2.6}$$

With all these properties correctly defined, we can use the compact notation $\lambda = (A, B, \pi)$ to denote the HMM.

To formulate the problem of solving temporal fusion using a HMM recogniser, we have:

- An input sequence – $O = [o(1), o(2), o(3), \ldots, o(T)]$, where $o(1), o(2), o(3), \ldots, o(T)$ are raw data or features vectors calculated from the input signal, and T is time instance.

- A number of class models – λ.

Given the model λ and a sequence of observations O, the problem is how to calculate the probability that the observed sequence was produced by the model – $P(O \,|\, \lambda)$. Here, we want to determine the likelihood of the observed sequence O, given the model λ to determine how well a given model class matches a given observation sequence (i.e., classification). To calculate this probability using the brute force approach has time complexity $O(2T \times N^T)$, where T is the length of sequences and N is the number of symbols in the state. This is intractable for realistic problems, as the number of possible hidden node sequences typically is extremely high. Clearly a more efficient procedure is required to solve this problem.

The *Forward–Backward* procedure is such an efficient procedure that has time complexity $O(N^2T)$ (Rabiner, 1989). The procedure comprises three main steps: 1) computing forward probabilities. 2) computing backward probabilities. 3) computing smoothed values. The forward and backward steps are often called "forward message pass" and "backward message pass", and the message passing originates from the way the procedure processes the given observation sequence. First the procedure moves forward starting with the first observation in the sequence and going to the last, and then returning back to the first. At each single observation in the sequence, probabilities to be used for calculations at the next observation are computed. During the backward pass the procedure simultaneously performs the smoothing step. This step allows the procedure to take into account any past observations of output for computing more accurate results.

Before using the forward-backward procedure to classify the temporal sequence, the model λ needs to be derived (i.e., model learning). This model learning problem is to determine a method to adjust the model parameters (A, B, π) to maximise the probability of the observation sequence given the model. The learning problem is the most difficult problem to solve. There is no possible analytic method (Rabiner, 1989) i.e. given any finite observation sequence there is no optimal way of estimating the model parameters. However, we can choose $\lambda = (A, B, \pi)$ such that $P[O|\lambda]$ is locally maximized using an *Expectation–Maximization (EM) algorithm*. The EM algorithm is used for finding maximum likelihood estimates of parameters in probabilistic models, where the model depends on unobserved hidden variables. The EM algorithm alternates between performing an *expectation (E) step*, which computes an expectation of the likelihood by including the hidden variables as if they were observed, and *maximization (M) step*, which computes the maximum likelihood estimates of the parameters by maximizing the expected likelihood found on the E step. The parameters found on the M step are then used to begin another E step, and the process is repeated until the log-likelihood function converges to a (typically) local maximum.

Situational awareness (i.e., level 2 – Situation assessment of the JDL fusion framework) is achieved based on the sequence of events observed over a period of time (i.e., temporal sequences). Damarla (2008) has shown that each state in the HMM is an event that leads to a situation and the transition from one state to another is determined based on the probability of detection of certain events using multiple sensors of multiple modalities (e.g., acoustic, seismic, passive infrared (PIR), chemical, magnetic field (B-field) sensors, and electric field (E-field) sensors are used in the paper). Bruckner (Bruckner, Sallans, & Russ, 2007) has also proposed an automated method for traffic monitoring that uses HMMs to model the traffic behaviours. Moreover, Bernardin at el. (2003) has also demonstrated using HMM to learn hand grasping movements for robots.

2.2.3 Self-Organizing Map (SOM) for Temporal Sequences

Figure 2.9 shows the basic architecture of the *Self-Organizing Map* (SOM). The SOM, first proposed by Kohonen (Kohonen, 2000), is a type of artificial neural network that has three essential constituents: (1) each neuron – n_j in the SOM contains model vectors – m_j that they have been trained on, (2) the neuron neighbourhood – N_C given by connections in a low-dimensional grid, and (3) a metric to evaluate the distance of the neurons to the input vectors – $D_{SOM}(x_i, n_j)$.

The SOM training algorithm (Algorithm 2.1) uses a competitive and unsupervised process called *vector quantization* to build the SOM using a stream of input vectors. First, the number of neurons – C in the map is determined, where C should be greater than the number of clusters of input vectors so that all type of input vectors can be modelled. Second, the model vectors – m_j in the map will be initialised randomly, or by utilising the two principal components (PCA) of the input data vectors (Kohonen, 2000). Third, the SOM training algorithm changes the model vectors by repeating the following two steps as described in Algorithm 2.1:

1. Acquire an input vector x_i and compare with all the model vectors – m_j in the map by using the metric which can be one of:

   - *Euclidean distance*: $D_{SOM}(x_i, n_j) = \sqrt{\sum_{i=1}^{n}(x_i - m_j)^2}$

   - *Mahalanobis distance*: $D_{SOM}(x_i, n_j) = \sqrt{\sum_{i=1}^{n}\frac{(x_i - m_j)^2}{\sigma_i^2}}$

   - *Manhattan distance*: $D_{SOM}(x_i, n_j) = \sum_{i=1}^{n}|x_i - m_j|$

   The best matching unit (BMU) on the map is the neuron with the model vector that is most similar to the input vector.

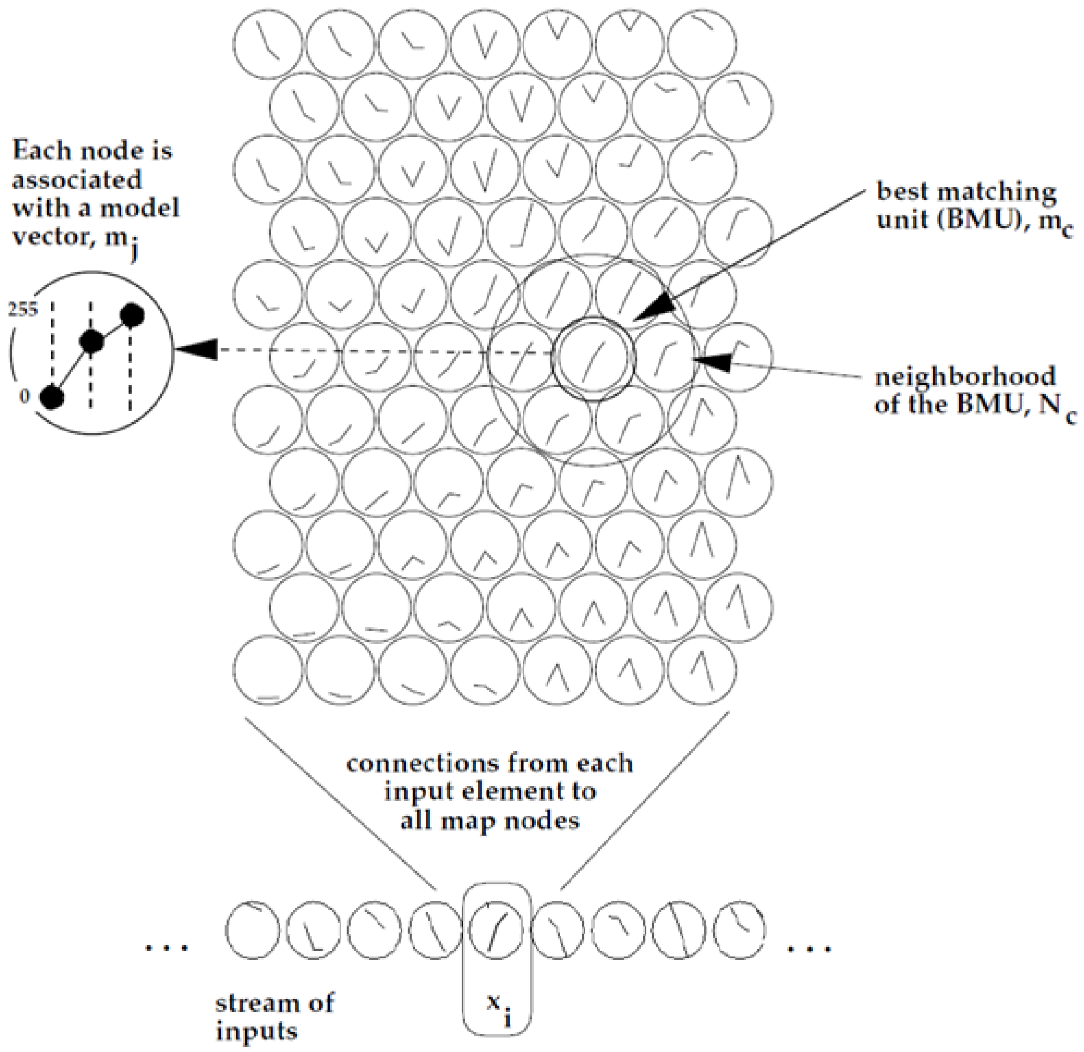

Each node is associated with a model vector, m_j
255
0
best matching unit (BMU), m_c
neighborhood of the BMU, N_c
connections from each input element to all map nodes
. . .
. . .
stream of inputs
x_i

Figure 2.9: The basic architecture of Self-Organizing Map (SOM). From:(Honkela, 1997)

```
select $\boldsymbol{C}$
Initialise $\boldsymbol{m_j}$ for all neurons
Repeat
    select $x_i$ from stream of inputs
    $m_C \leftarrow \underset{j}{argmin}\{D(x_i - m_j)\}$
    for each $m_j$ of $N_C$ do
        $m_j \leftarrow m_j + \gamma \times f(N_C) \times D_{SOM}(x_i, n_j)$
    end for
Until convergence
```

Algorithm 2.1: Basic SOM training algorithm

2. The model vectors of the BMU – m_C and its neighbouring neurons – N_C are moved closer towards the input vector x_i according to γ and $f(N_C)$. The learning rate – $\gamma \in (0,1)$ possibly decreases at each iteration to ensure the convergence of the training process. The update strength function – $f(N_C)$ calculates the amount of update for the current neuron, depending on its distance to the BMU in the vector space, the BMU adapts the strongest update and N_C weaker updates.

The metric and the update procedure are two main elements in the training process. The metric aims to measure the similarity of possibly sequential input vectors to the neurons to find the BMU. The model vector of the BMU and its neighbourhood neurons (N_C) are updated so that their responses to the input vectors is increased. To extend the standard SOM for classifying temporal sequences, these two main steps need to be extended to capture the temporal relationships among the input vectors. Extensions of the SOM for temporal sequence processing have been proposed in the literature such as *Temporal Kohonen map* (TKM) and *SOM for structured data* (SOMSD) (Hagenbuchner et al., 2003).

The Temporal Kohonen Map (TKM) proposed by Chappell and Taylor (1993) extends the original metric of the SOM by using *leaky integration* to capture the temporal knowledge. Let the temporal sequence be denoted by $s = (s_1, s_2, \dots, s_t)$ and t is the sequence length. The Temporal Kohonen Map (TKM) computes the distance of $s = (s_1, s_2, \dots, s_t)$ from neuron m_j by:

$$D_{TKM}(s, n_j) = \sum_{i=1}^{t} \alpha(1-\alpha)^{i-1} \left\| s_i - m_j \right\|^2$$ **Equation 2.7**

where $\alpha \in (0,1)$ is a memory parameter (Chappell & Taylor, 1993). The BMU is found when s_1 is closest to m_j as in the standard SOM, and the remaining sum – $(1-\alpha)\left\|s_2 - m_j\right\| + (1-\alpha)^2\left\|s_3 - m_j\right\| + \dots$ is also a minimum. The remaining sum integrates the distances between the neuron and each vector of the sequence weighted by an exponentially decreasing factor $(1-\alpha)^{i-1}$. Thus, the BMU will be a neuron with distance closest to the previous vectors in the sequence.

The update procedure of model vectors for the TKM is the same as for the standard SOM. Further studies pointed out that the training process of TKM is unstable and leads to only suboptimal results (Koskela, Varsta, Heikkonen, & Kaski, 1998). Consequently, the TKM has not been widely adapted for temporal sequence processing (Varsta, Milan, & Heikkonen, 1997).

The SOM for structured data (SOMSD) is another much more efficient and powerful SOM extension than TKM (Hagenbuchner et al., 2003). The SOMSD can process not only one dimensional sequence, but also a tree structured sequence. The SOMSD map assumes that the neurons are arranged on a rectangular l-dimensional lattice structure map so that each neuron can be indexed i.e, a neuron at (2, 2) of a 2-D lattice structure map. Each neuron of the SOMSD map has two vectors: model vector – $m_j \in R^c$ and an additional context vector – $k_j \in R^l$. The context vector represents temporal context by the corresponding index of the BMU in the previous time step. To find the BMU, the distance of sequence $s = (s_1, s_2, \dots, s_t)$ from a neuron is recursively computed by:

$$D_{SOMSD}\left(s, n_j\right) = \alpha_1 \left\| s_1 - m_j \right\|^2 + \alpha_2 \left\| K_{SOMSD}\left(s_2, \dots, s_n\right) - k_j \right\|^2 \qquad \textbf{Equation 2.8}$$

where $K_{SOMSD}(s_2, \dots, s_n)$ is the location of the BMU with the smallest $d_{SOMSD}\left(s_2, n_j\right)$ in the previous time step. The current context vector – k_j is represented by the index of the BMU of the map in the previous time step, and $\alpha_1, \alpha_2 > 0$ are constants to control mediation between the amount of model vector match versus context vector match. To find the BMU at the current step, the distance between the sequence and the BMU is defined as a mixture of two terms: the match of the neuron's model vector and the current sequence entry, and the match of the neuron's context vector and the context currently computed. This recursive process imposes a temporal bias towards those neurons whose context vector matches the index of the BMU of the previous time step.

In the update procedure of SOMSD, the well know *Hebbian learning* (Kohonen, 2000) takes place for both model vectors and context vectors i.e., the model vectors are moved toward the current entries of the sequence, and the context vectors are moved toward the index of the BMU of the previous time step.

The SOMSD has been successfully applied for the classification of artificially constructed pictures represented by tree structured sequences. A good classification accuracy of the objects in the picture (i.e., Level 1 – Entity assessment of the JDL fusion framework) can be obtained by attaching appropriate classes to the neurons of the SOMSD map. (Hagenbuchner et al., 2003; Hagenbuchner, Tsoi, & Sperduti, 2001). Frey (2008) has extended the SOM for the continuous diagnosis of the functionality, the early detection of potential failures, and the continuous monitoring of the underlying physical process. Furthermore, Lai et al. (2007) has also enhanced the SOM for situation awareness.

2.3 Chapter Summary

This chapter presents a survey of popular multi-sensor data fusion frameworks. The frameworks provide a common understanding of the component functions (e.g., source pre-processing, feature extraction and pattern processing, situation assessment, and decision making) for building multi-sensor data fusion systems. All of the reviewed frameworks have some strengths and weaknesses for the modelling of sensor fusion applications. For instances, many typical multi-sensor data fusion systems such as distributed control and SCADA systems, have a closed control loop, thus the Boyd ODDA loop and the Omnibus framework are more advantageous than the JDL and waterfall frameworks for this type of system. However, both the JDL and the waterfall frameworks are better guidelines for building embedded real-time fusion systems since they emphasise low-level fusion processing. Furthermore, Dasarathy's framework provides the best understanding of the data or information flows among the component functions, and the innovation of a self-improving data fusion framework. Bowman has proposed several revisions to the JDL framework, and generalizes a conclusion that the main functions of a data fusion systems are the encompassing duality between *data fusion* (i.e., estimate) and *resource management* (i.e., control) (Bowman, 1994; Bowman et al., 1999; Llinas et al., 2004; Steinberg & Bowman, 2004).

The section on pattern recognition techniques for temporal fusion describes the major algorithms for classification of temporal sequences. For instance, the algorithms which are able to learn the temporal contexts from training sequences such as self-organising maps for sequences and hidden Markov models have been described. The algorithms that can make use of knowledge or assumptions about the temporal sequences to be recognised (i.e., use of class sequences) such as dynamic time warping have been explained. DTW can adjust the constraints of warping path to cope with the variability in temporal sequences. In addition, the states of art applications of these major algorithms have been reviewed.

Temporal fusion is an additional dimensionality to the fusion process that has not been emphasised in the previously reviewed frameworks. In the next chapter, we will explore the role of temporal fusion in the fusion frameworks, and relate the reviewed frameworks with the DTW based framework that is proposed in this thesis.

Chapter 3 Multi-Agent Temporal Fusion Framework

Many real world situations change over time. When the multi-sensor system is used for situation awareness, it is useful not only to know the state or event of the situation at a point in time, but also, more importantly, to understand the causalities of those states or events as they change over time. This chapter describes the proposed framework that emphasises the horizontal dimension of the fusion process, that is, temporal fusion – fusion of the multi-sensor data or events derived from multi-sensor data over a period of time.

The main component of our framework is the DTW temporal fusion agent which uses dynamic time warping (DTW) as the core recogniser to perform temporal fusion. In this chapter, we will also describe the proposed extensions for the DTW recogniser to cope with the challenges and variability of temporal sequences.

3.1 Multi-agent temporal fusion framework

The previously reviewed fusion frameworks (refer to section 2.1) have a centralized architecture with all data processed by a single fusion node. When dealing with the large scale of sensor networks or wireless sensor networks, the distributed architecture is necessary to minimise the complexity of communication and computation. Figure 3.1 shows the proposed fusion framework that emphasises the DTW temporal fusion agents for aggregation of information over time, and the multi-agent architecture for scalability to large sensor networks. The proposed fusion framework includes three major layers: *hardware*, *agents*, and *users*:

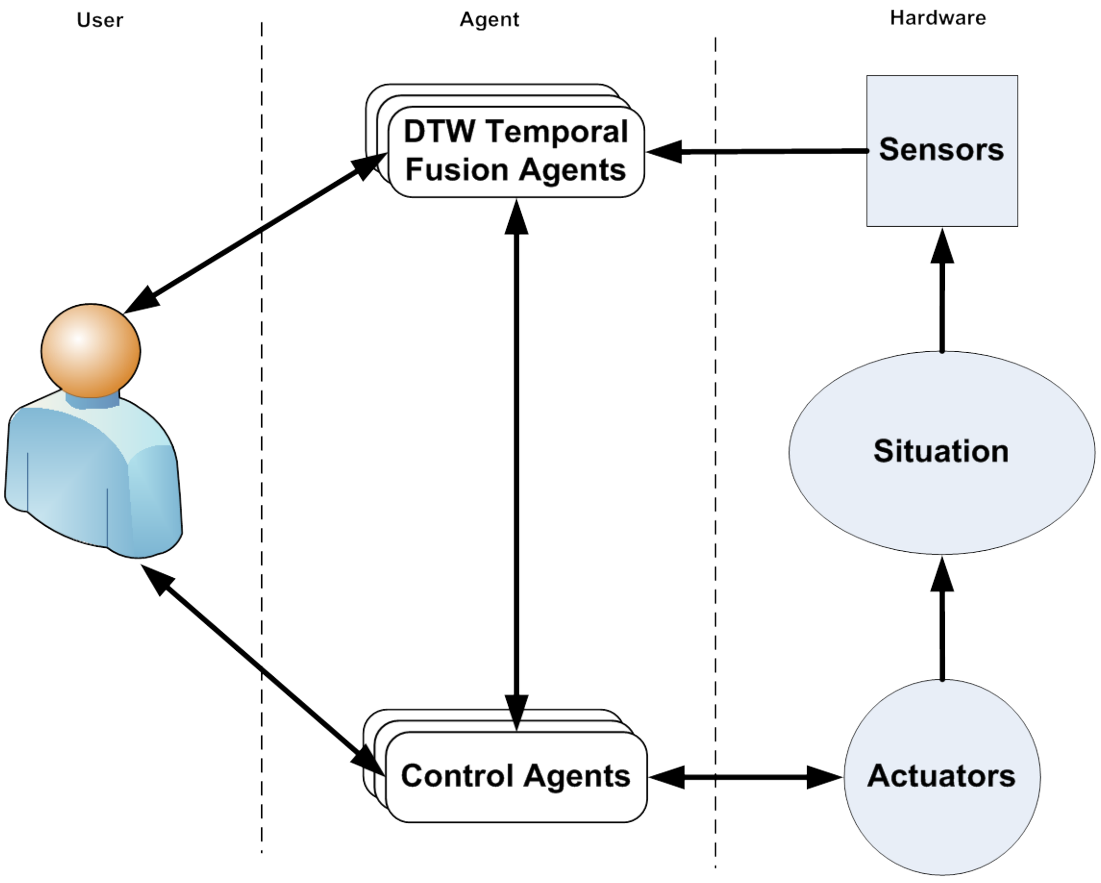


Figure 3.1: The multi-agent temporal fusion framework

The **user layer** represents the *human decision maker* of the temporal fusion system. The framework transforms information requirements of the decision makers through the software layer that will then allow the users to visualize the fusion products and generate feedback or control to enhance the fusion process. The *control agents* can convey the information needs of the decision-maker to the real-world devices (e.g., actuators, databases, etc.) that will either produce data relevant to the needs or perform automated tasks. In contrast, the *DTW temporal fusion agent* can aggregate the data acquired from the sensing devices (e.g., sensors or motes – self-contained sensing, computation, communication and power devices) into a form that can satisfy the decision maker and is intuitively usable by the user for the decision-making process.

The **hardware layer** contains *sensors, actuators*, and the *environments* being monitored. A *sensor* is a device that physically interacts with the monitored environment to perceive a property of the physical environment (e.g., temperature, pressure, light, sound, vision or motion) and transforms the result into a quantitative measurement that can be interpreted by the computer system. When the control agents are performing some automated tasks, some attributes of the external environment are required to be changed automatically without manual operations. In this case, the system will contain *actuators* which are able to move or control some environmental attributes.

The **agent layer** is the heart of the proposed temporal fusion framework, which consists of a group of the *DTW temporal fusion agents* and *control agents*: The function of control agents is to control the resources; on the other hand, the role of fusion agents is to derive the optimal estimates of the situation. In control theory, the relationship between control and estimation is expressed mathematically as a *duality* (Bowman, 2004). Similar to the duality that exists between estimation and control in control theory, Bowman (2004) has explored the duality between *data fusion* and *resource management*, which incorporates the *association*/*planning* duality as well as the *estimation*/*control* duality. To be more specific:

- The function of control agents is to control the hardware and produce the feedback to the fusion agents. The group of control agents is responsible for identifying the data sources to produce relevant input for a given fusion process i.e., *sensor selection* techniques can be applied in this process to find the sensors that maximise the information gain to meet the information needs of the decision maker (Zhang & Ji, 2005). In addition, the control agents coordinate the collection of the required data with appropriate actuating and sensing devices, or data and information from appropriate sources. This process that seeks to manage, or coordinate, the use of a set of actuators and sensors in a dynamic, uncertain environment, to improve the performance of the system is known as *sensor management* (Xiong & Svensson, 2002) or *resource management* (Bowman, 2004). Moreover, the control agents also perform the function of the level 4–process assessment of the JDL framework to monitor the overall temporal fusion process, and improve the overall system performance by sending feedback to the DTW temporal fusion agents. This process involves the use of reinforcement learning (Sutton & Barto, 1998) to improve the performance of the DTW temporal fusion agents. The control agent is itself a broad subject that requires the deep understanding of sensor selection (Zhang & Ji, 2005), sensor management (Xiong & Svensson, 2002), resource management (Bowman, 2004), and reinforcement learning (Sutton & Barto, 1998). Thus, the study of control agents is not within the scope of this thesis, and is suggested to be future work.

- The role of the DTW temporal fusion agent is to aggregate data, features, or decisions over a period of time to derive the optimal estimates of the *entities* (i.e., level 1 – entity assessment of JDL framework), *situations* (i.e., level 2 – situation assessment), and *impacts* (i.e., level 3 – impact assessment), which satisfies the information needs of decision-makers and minimises the decision-maker's uncertainty of information. Our framework contains a group of temporal fusion agents that use DTW as the core recogniser as shown in Figure 3.2. The rest of this chapter will be devoted to the architecture of the DTW temporal fusion agent in details.

Fundamentally the DTW temporal fusion agents of the agent layer can be arranged in three different topologies: *centralized*, *hierarchical*, and *distributed*. In addition, we use an *information graph* (Liggins et al., 1997) to show the information flows among the DTW temporal fusion agents, and how the agents fuse the sensor data over the time dimension, in these three fusion architectures.

In the *centralized fusion* architecture as shown in Figure 3.2, the DTW temporal fusion agent, F_1, is treated as a central processor that collects, processes, and fuses all information from the different sensors – S_1, S_2, S_3. On the right side of Figure 3.2, the solid circles represent the measurements generated by each sensor over time. The hollow rectangles represent the data vectors (has three elements) of the multimodal sequence that are buffered in a sliding window of the DTW temporal fusion agent, and this buffered sequence will be the input to the temporal DAI or FEI-DEO fusion process as described in section 3.2. Although the centralized fusion architecture has the best performance, it suffers from the two drawbacks (Liggins et al., 1997): *Communication* – there might be a communication bottleneck if all the sensors in the large sensor network communicate directly with the central fusion agent at the same time; *Vulnerability* – the entire fusion system fails. If there is a failure in the central DTW fusion agent or the central communications facility.

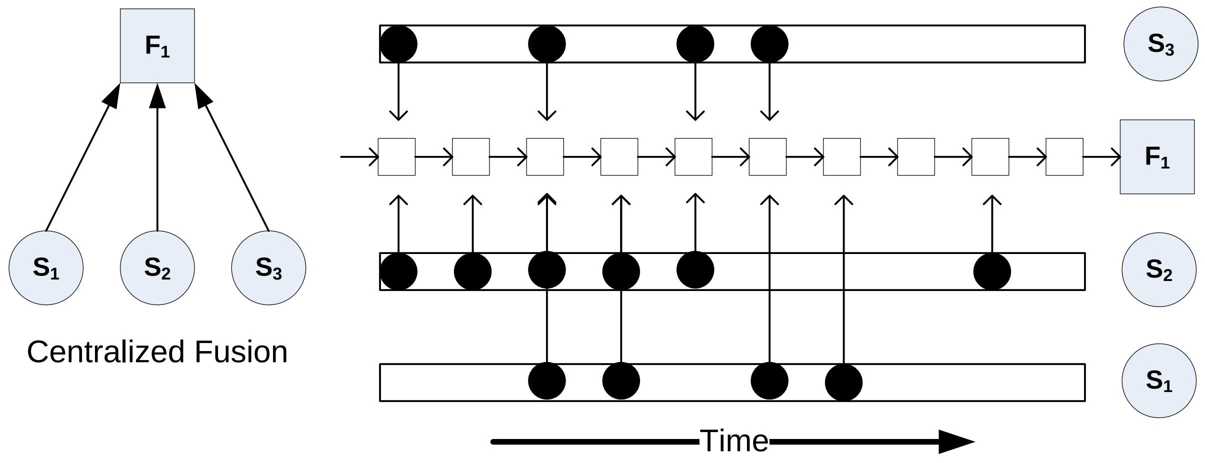


Figure 3.2: Centralized fusion architecture with one DTW temporal fusion agent and three sensors. (Liggins et al., 1997)

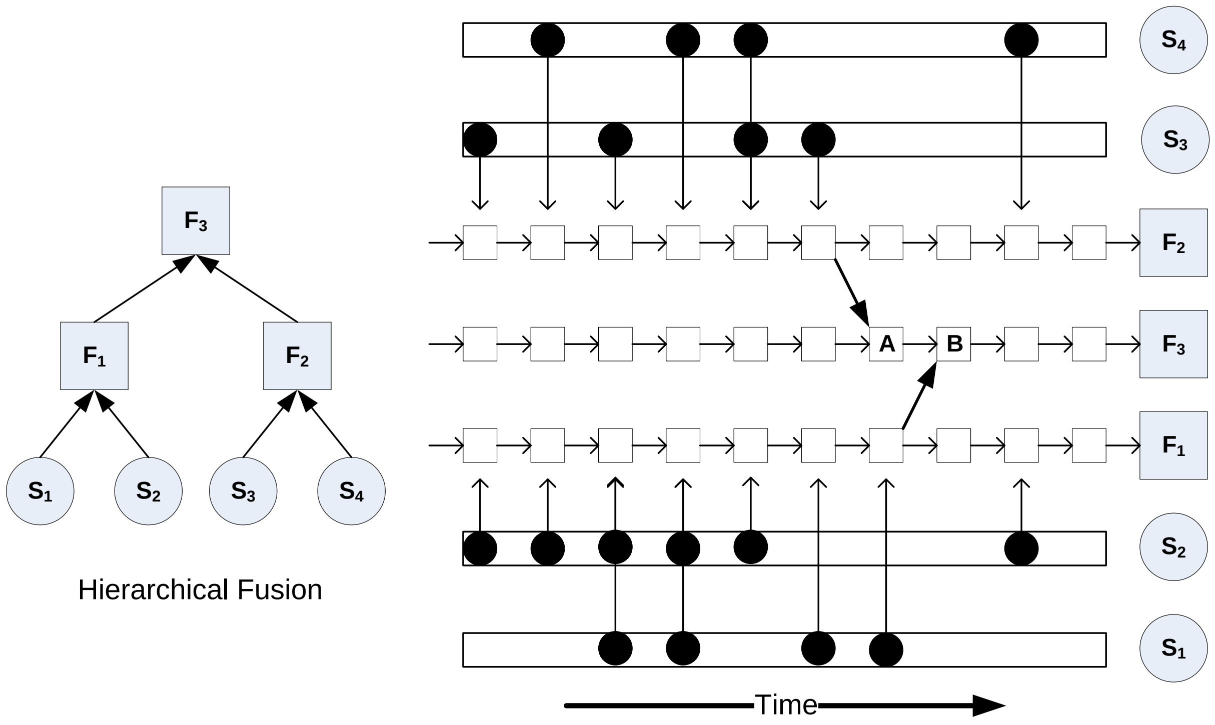


Figure 3.3: Hierarchical fusion architecture with three DTW temporal fusion agent and four sensors. (Liggins et al., 1997)

In the *hierarchical fusion* architecture, there are several levels where the top level contains a single central fusion agent (i.e., performing DEI-DEO), the bottom level consists of several local fusion agents (i.e., performing DAI or FEI-DEO), and several fusion agents in between the top and bottom levels also performing the DEI-DEO fusion process. Each of the local fusion agents (subordinates) may derive the local fused estimate from the inputs of a small group of sensors or an individual sensor, and send those local estimates to a central fusion agent (superior) to develop a global fused estimate. The left side of Figure 3.3 shows an example of the hierarchical fusion architecture that has two local DTW temporal fusion agents – F_1, F_2 (subordinates) with each fusing data from two sensors, and one central DTW temporal fusion agent – F_3 (superior). On the right side of Figure 3.3, the two local agents – F_1, F_2 both perform the temporal DAI or FEI-DEO fusion process over a period of time, and the central agent F_3 further performs the temporal DEI-DEO fusion process to combine the local decisions of A and B to make the global decision.

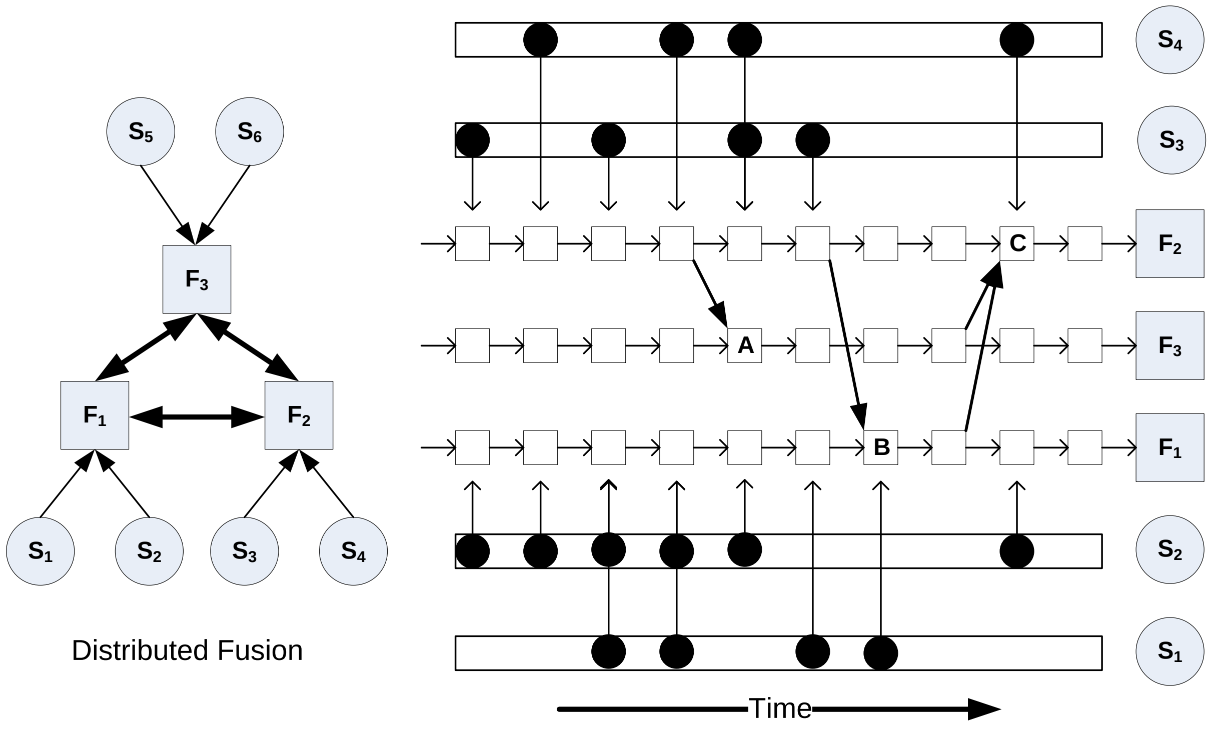


Figure 3.4: Distributed fusion architecture with three DTW temporal fusion agent and six sensors. (Liggins et al., 1997)

Figure 3.4 shows the *distributed fusion* architecture. There is no subordinate–superior relationship among the DTW temporal fusion agents. Each fusion agent can communicate its estimate to other fusion agents to assist them in making their local decisions, and in turn receives the estimates from other fusion agents to facilitate its own decision making. The left side of Figure 3.4 shows an example of the distributed fusion architecture that has three distributed DTW temporal fusion agents – F_1, F_2, F_3 with each deriving its own local estimate from two sensors. For instance, on the right side of Figure 3.4 agent F_2 sent its local estimate (i.e., fused data or feature from sensors S_3, S_4) to agent F_3 to assist F_3 in performing temporal FEI-DEO fusion process to derive *A*. Another example is that before agent F_2 can make decision *C*, it required the local estimates from both agents F_1, F_3 as depicted in Figure 3.4.

The distributed fusion architecture is not constrained by centralized computational limitation or communication bandwidth bottleneck, since the information flow between agents is reduced and the data processing load is distributed. All the fusion agents are complementary in distributed fusion, and this architecture is less vulnerable than a centralized fusion architecture when loss of sensors and any changes in the network occurs (Mitchell, 2007).

3.2 *DTW temporal fusion agent*

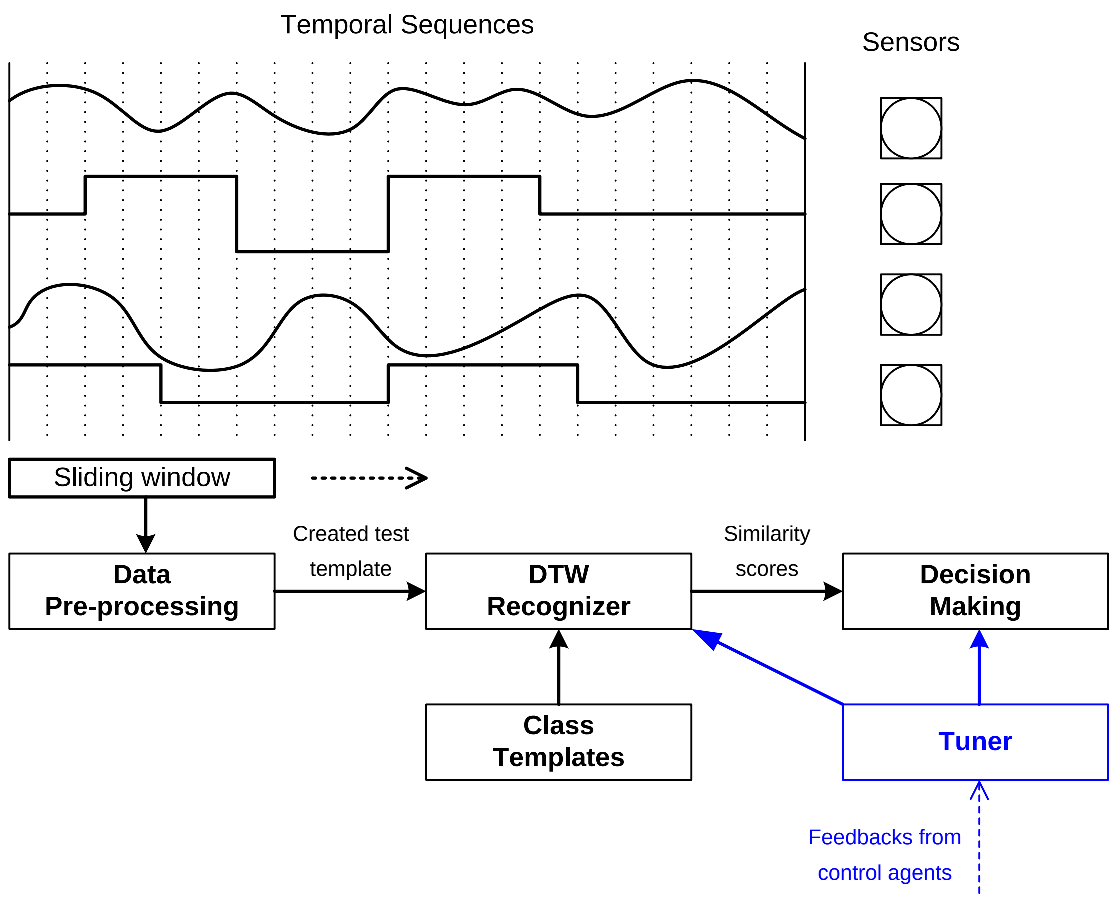


Figure 3.5: The modules of a DTW temporal fusion agent.

Figure 3.5 shows the software modular design of the DTW temporal fusion agent, and how the agent performs online temporal fusion of multimodal sequences using four heterogeneous sensors as an example. The multimodal sequences generated are a mix of binary, discrete, and continuous variables. There are four main modules implemented in each of the DTW temporal fusion agents: *data pre-processing*, *DTW recogniser*, *a database of class templates*, and *decision making*. The raw data input to the system is buffered in a sliding window. For the temporal data in/decision out fusion process, both the selected class templates and the raw data currently buffered in the sliding window are in raw data form. For the temporal feature in–decision out fusion process, the same pre-processing techniques are applied to the acquired data and the class templates. The data pre-processing module converts the raw data into the test template to improve the recognition accuracy and speed. Examples of such pre-processing are *filtering*, *variable normalisation*, *feature extraction*, and *segmentation*. The DTW recogniser then measures the similarity between the test template and all the selected class templates by calculating the DTW distances. Finally, the decision making module decides which class template is the most similar one to the input and hence outputs the best matching class.

3.3 Data pre-processing

In the presence of noise and outliers, the DTW might fail to measure the correct similarity between two sequences, since the DTW tries to match all elements between the two sequences. To eliminate noise and outliers, the system applies a suitable signal filter.

Multisensor data usually consists of a combination of both discrete and continuous variables. Normalisation of variables is required to ensure that the DTW distance calculated by using the local distance measure is representative and stable. Furthermore, normalisation ensures fairness across various sensors as well as features. Variable normalisation is typically performed to integrate vector components of varying dynamic range. Suppose some vector components have a variance that is significantly higher than those of others, the former will dominate the results of the local distance measure. Some common normalisation methods have been investigated to normalise the dynamic ranges of the vector components in the interval [0, 1] i.e., Variance normalisation, Min–max normalisation and Softmax normalisation.

The DTW recogniser can perform temporal fusion on raw data, but by performing fusion on features extracted from the data, both classification performance and computational speed may be improved. Mantyjarvi (2003) and Schmidt (2002) have studied extensively the suitable features to be extracted from different types of sensors for the recognition of various contexts.

While the multisensor data is acquired from heterogeneous sensors in real-time, there needs to be a method for segmenting areas of interest of multimodal sequences from the continuous data streams so that the DTW recogniser can perform temporal fusion. For example, in gesture recognition, the system is used to fuse data acquired from accelerometers worn on both wrists. Since the gestures are being performed continuously, there needs to be a method for selecting areas of movement to find candidate gestures from the continuous data streams. Finding periods of inactivity in the accelerometer data is one solution (Chambers et al., 2004). The system currently utilizes both the sliding window operation and a DTW variant to solve the unknown endpoints or the segmentation problem. If we have sufficient prior knowledge about the multisensor data to be recognised so that a suitable and robust segmentation method can be developed, the recognition accuracy of the system can be further improved.

3.4 Training of the DTW recogniser

Successful fusion processing, and particularly automated fusion, depends on the existence of a priori information, patterns, and models of behaviour (known as training data), without which it is difficult to predict potential hostile courses of action with any degree of confidence.

In the training stage, the most important task is the creation of reliable class templates for each class of multimodal sequences to be recognised. Although the template can be selected randomly (i.e., random selection), the accuracy of the DTW-based recognition system greatly relies on the quality of the created class templates. Some better methods than random selection are:

1. **Normal selection**: The normal procedure for selecting the class templates is to use each example as a template and determine its recognition rate when classifying the other examples of that class. The example with the best recognition rate is chosen as the template.

2. **Minimum selection**: Select the sample for the class template using the intra-class DTW distance that is the sum of DTW distances between the template and all other examples within the same class. The sample with minimum intra-class DTW distance is selected as the class template. In terms of clustering, the template selected by this method is the centre of the cluster.

3. **Average selection**: Generate the class template from a set of the best templates, e.g., extract the five templates that have the best minimum inter-class DTW distances, then take the average of these to produce the final class template. Averaging a collection of time series that are not perfectly time-aligned is non-trivial (Abdulla et al., 2003;

Ratanamahatana & Keogh, 2005). However, it has been demonstrated (Abdulla et al., 2003) that the speaker-dependent speech recognition rate is improved from 85.3%, using normal selection, to 99%, using this method.

4. **Multiple selection**: Use several class templates for each class, determine the classification for each template and then combine the results. This method usually achieves better classification rates than the aforementioned methods. However, this method is computationally inefficient because it increases the number of class templates to be compared. This method can also be used together with all the aforementioned methods. For instance, use minimum selection to sort all training templates in ascending order according to their intra-class DTW distances, and select the top five as the class templates (multiple minima). Usually, the results are better than minimum selection.

We will compare random selection, minimum selection and multiple selection in our experiments to explore the importance of selecting good class templates. After the creation of class templates, two other parameters are required when performing online temporal fusion. Given each class C^n ($1 \leq n \leq N$, $N =$ total number of classes), the following parameters need to be derived from the training templates of C^n (denote as TC^n):

- *End regions* (E_1 and E_2). We derive the end regions for each class C^n. E_2 and E_1 respectively are defined as the maximum and minimum lengths found within TC^n as depicted in Figure 3.6. The length of class templates of C^n and E_2 are used to determine the number of rows and columns of the DTW distance tables of C^n respectively. E_1 and E_2 together define all possible lengths of multimodal sequences of class C^n, and the difference between them is the Band-DP of C^n (i.e., $E = E_2 - E_1$). The use of the end region in the DTW recogniser will be discussed in Section 3.5.2.

- *Rejection threshold*. Since continuous data streams contain both target and unknown sequences, the rejection threshold is used to reject any unknown test template that should not belong to any of the N classes. Our current heuristic is to choose this threshold as the mean of the intra-class DTW distances between the selected class template and all TC^n plus a number of standard deviations. The number of standard deviations is determined such that the threshold will not reject any of the training templates TC^n. Other heuristics can be used such as the maximum intra-class DTW distance. During the sliding window operation, if the minimum DTW distance between the test template and the class template of C^n exceeds the threshold, it indicates the test template is unknown and does not belong to any of the N classes.

3.5 Dynamic time warping recogniser

After the class templates have been created, the next step is to measure the similarity between online data and the class templates. Since the length of multimodal sequences of the same class can vary greatly and the endpoints are unknown, the proposed system uses a variant of DTW to achieve both time alignment and the similarity measure. In the remainder of this section, we will develop dynamic time warping for online recognition of multimodal sequences generated from multiple sensors.

3.5.1 Multi-dimensional local distance measure for DTW

The class templates $C(I \times V)$ and test template $T(J \times V)$ represent multimodal sequences where V is the number of variables and I, J indicate the lengths of class and test templates respectively. To calculate the DTW distance between the test and class templates, the system uses the extended *Euclidian distance* (Equation 3.1) and *cosine correlation coefficient* (Equation 3.2) as the local distance measures to calculate the difference between the two vectors, C_i^V and T_j^V. They are defined as:

$$d_E(C_i^V, T_j^V) = \sqrt{\sum_{v=1}^{V} W(v)\left(C_{i(v)} - T_{j\,(v)}\right)^2}$$

Equation 3.1

$$d_C(C_i^V, T_j^V) = 1 - \frac{\sum_{v=1}^{V} W(v) C_{i(v)} T_{j\,(v)}}{\sqrt{\sum_{v=1}^{V} C_{i(v)}^2}\sqrt{\sum_{v=1}^{V} T_{j\,(v)}^2}}$$

Equation 3.2

where *W* is a positive definite weight vector. The weight vector *W* can be used in the DTW algorithm to give more weight to certain variables to improve the performance of online recognition. If every element of *W* is equal to 1, we obtain the normal *Euclidian distance* and *cosine correlation coefficient*. If prior knowledge of the importance of the sensors to the multimodal sequences under recognition is given, we can assign the elements of *W* to reflect this importance.

3.5.2 DTW variant for unknown start and end points

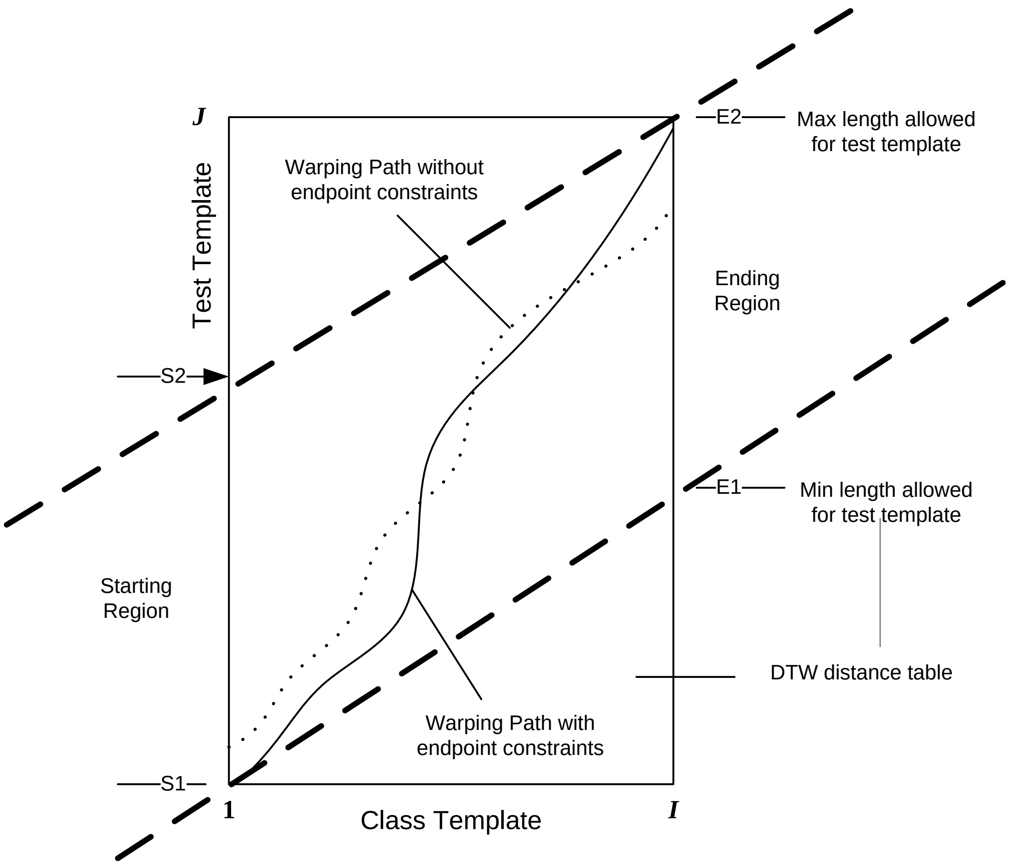

J
Test Template
Warping Path without endpoint constraints
E2
Max length allowed for test template
Ending Region
S2
E1
Min length allowed for test template
Starting Region
DTW distance table
Warping Path with endpoint constraints
S1
1
Class Template
I

Figure 3.6: DTW variant to solve the problem of unknown start or end points.

One of the major drawbacks in the original DTW algorithm is the endpoint constraints that require the warping path to start and finish in diagonally opposite corner cells of the distance table as shown by the solid curve in Figure 3.6. When reasonably accurate determination of the start and end points of the multimodal sequence has been made, this constraint is acceptable and does not harm overall performance of the DTW recogniser. However in the case of online recognition of multimodal sequences where the endpoints are unknown, the original DTW algorithm is inadequate. As such, we use a variant of the original DTW algorithm that replaces the endpoint constraints to allow the start point to fall in a defined start region (i.e., between S1 and S2) and the end point to fall in a defined end region (i.e., between E1 and E2) as shown by the bold dashed lines in Figure 3.2. The new definitions for the start and end regions are as follows:

$$i(1) = 1, j(1) = s, S_1 \leq s \leq S_2$$ **Equation 3.3**

$$i(Q) = I, j(Q) = e, E_1 \leq e \leq E_2$$ **Equation 3.4**

Hence the optimal warping path starts at the first vector of the class template and within the start region of the test template, and ends at the last vector of the class template and within the end region of the test template. Given a start and end region, the purpose of the DTW algorithm is to determine (in an optimal and efficient manner) the warping path, which provides the best time alignment between the class and the test template and has the minimum cumulative distance. The search for the optimal warping path for this variant of the DTW algorithm can be expressed as an extension of Equation 2.2:

$$DTW(C,T) = \min_{S_1 \leq s \leq S_2} \left(\min_{E_1 \leq e \leq E_2} \left(\frac{\sum_{q=1}^{Q} w(i(q), j(q))}{NF} \right) \right)$$ **Equation 3.5**

where s and e represent the start and end points of a warping path respectively. For computational efficiency, we need to keep both the start region $(S_2 - S_1)$ and end region $(E_2 - E_1)$ minimal. However, it is possible to expand either the start or the end region to incorporate the entire test template. In this case, this DTW variant can be used for *subsequence matching* (Vlachos et al., 2003), in which the class template is a subsequence within the longer test template. For example, if $S_1 = 1$ and $S_1 = J$, the dotted curve of Figure 3.6 illustrates that the class template is actually a subsequence of the test template.

3.5.3 Online DTW

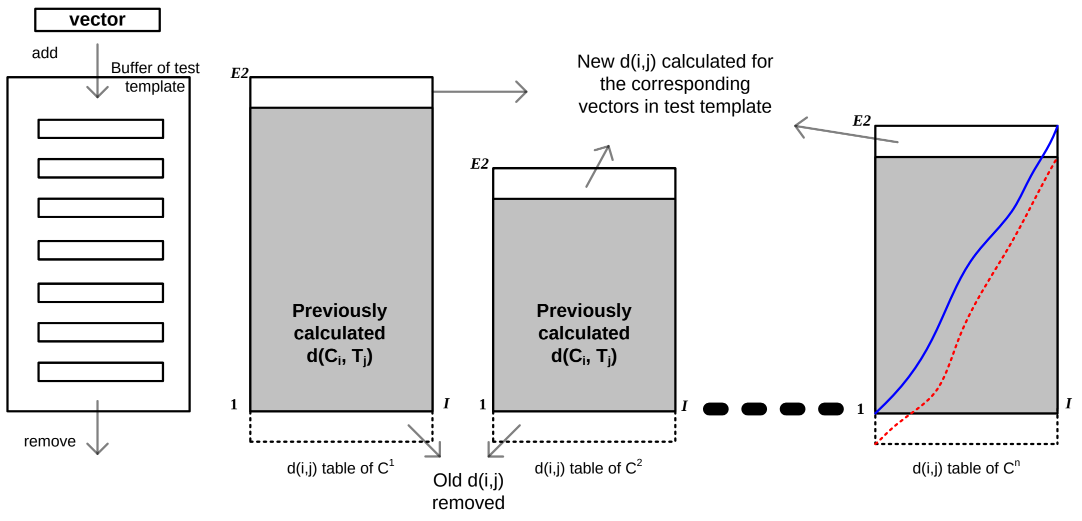

vector
add
Buffer of test template
remove
E2
Previously calculated d(C_i, T_j)
1
I
d(i,j) table of C^1
Old d(i,j) removed
New d(i,j) calculated for the corresponding vectors in test template
E2
Previously calculated d(C_i, T_j)
1
I
d(i,j) table of C^2
E2
1
I
d(i,j) table of C^n

Figure 3.7: The sliding window operation involves both the buffer of test template and DTW distance tables.

The DTW recogniser maintains an $I \times E_2$ distance table D for each class template C^n. Figure 3.7 shows that the online sliding windowing operation is applied on both the buffer of the test template T and the distance tables. The data pre-processing module uses a sliding window to extract one feature vector at a time from the streaming data which is then stored in the buffer of T. The system starts by buffering the feature vectors until it reaches the maximum size. The maximum size is defined as the largest E_2 learned during the training stage e.g., Figure 3.7 shows that the buffer size is the same as E_2 of D of C^1. Once the buffer reaches the maximum size, the DTW recogniser starts to calculate the elements of $d(i,j)$ and the warping path W for each D. From this time on, the buffer of T and D become sliding windows. When a new vector of T is stored, the first vector of T and the first row of D are deleted. Each D expands by one row and only the elements of $d(i,j)$ for the new row are calculated, i.e., when the jth vector of T is extracted, $d(i,j)$ is calculated for all $i(1 \leq i \leq I)$. In this way, the algorithm is efficient since it uses the previously calculated value. At each new step, the online DTW recogniser discards the old W (the dashed line of the table of C^n). A new W (the solid line of the table of C^n) is calculated in the updated D by fixing the start point ($S_2 = S_1 = 1$ in Equation 3.5) and relaxing the end point ($E_1 \leq e \leq E_2$ in Equation 3.5). Thus, the total time complexity of the online DTW involves both the update of distance tables $O(NI)$ and the calculation of the new W $O(NIE_2)$, where N is the number of class templates. If the *Band-DP* is used, the time complexity is reduced to $O(NE) + O(NIE)$, where $E = E_2 - E_1$.

If the lengths of class templates (between classes or within the class) are quite different, we need to normalise the returned DTW distance. We would expect long class templates to have higher DTW distances than short class templates, since they have longer warping paths on which to accumulate errors. The system uses the following options for the normalising factor (NF) in Equation 2.2:

- Option 1, NF = 1 (no normalisation on the distance).
- Option 2, NF = the length of the optimal warping path.
- Option 3, NF = the length of the longest template when comparing class and test templates.
- Option 4, NF = the length of the shortest template when comparing class and test templates.
- Option 5, NF = the sum of the lengths of both class and test templates.

At each new time instance, the normalised DTW distances are calculated for each class template and passed to the decision making procedure to perform classification.

3.6 Decision making procedure

The last major step in the proposed system (see Figure 3.5) is the decision making procedure which chooses the class template that most closely matches the unknown test template. There are two decision rules used in the system depending on which method is used to create the class template.

For *normal*, *minimum*, and *average* selections, a single class template is used for each class, and the *nearest neighbour* decision rule (NN rule) is used. We assume N classes, C^n,

$1 \leq n \leq N$, and for each test template we obtain the DTW distance D^n using our online algorithm. Then the decision rule is simply:

$$n^* = arg \min_{n}[D^n]$$ **Equation 3.6**

and the class template C^{n^*} is chosen as the winning class template. If the ranking of the class templates is required, the set of distances D^n can be used to give a new set of C^n which is ordered according to the DTW distance calculated.

For *multiple* selections in which each class is represented by two or more class templates, the K-nearest neighbour (KNN) decision rule can be used. Thus if we assume there are M class templates for all N classes, and we denote the *m*th class template of C^n as $C^{n,m}$, where $1 \leq n \leq N$, $1 \leq m \leq M$, then the DTW distance for the *m*th class template of C^n is $D^{n,m}$. If we select K $(1 \leq k \leq M)$ nearest neighbours out of M class templates of C^n that have the least DTW distances to the unknown test template (centre K samples in the cluster), and denote the DTW distance of the *k*th neighbour to the unknown test template as $D^{n,k}$. Then for each test template, we obtain the DTW distance for D^n as

$$D^n = \frac{1}{K}\sum_{K=1}^{K} D^{n,k}$$ **Equation 3.7**

and we choose the index of the recognised class using Equation 3.6. It should be noted that for $k = 1$, the K-nearest neighbour decision rule becomes the nearest neighbour rule. Similarly, we can compute the ranking of the class templates.

3.7 Chapter Summary

This chapter described the proposed multi-agent temporal fusion framework that includes three major layers: *hardware*, *agents*, and *users*. To differentiate our framework from the reviewed frameworks in the literature, we emphasised the application of DTW temporal fusion agents for aggregation of information over time, and the three different architectures: *centralized, hierarchical*, and *distributed* for organising the DTW temporal fusion agents. In addition, we use an *information graph* to show the information flows among the DTW temporal fusion agents, and how the agents fuse the sensor data over the time dimension.

In this chapter, the four main modules of the DTW temporal fusion agent: *data pre-processing, DTW recogniser, training of class templates*, and *decision making,* are described. We need to extend the DTW recogniser in various ways to be able to use it in the DTW temporal fusion agent to deal with multimodal sequences that consist of the data or feature sequences acquired from multiple heterogeneous sensors over a period of time. First, *Euclidian distance* (Equation 3.1) and *cosine correlation coefficient* (Equation 3.2) are proposed for measuring the local distance in the warping path of the temporal sequences with multi-dimensions. Second, we extended the DTW recognizer to deal with unknown start and end points so that it is able to recognize the same class of multimodal sequences even though they have different lengths locally or globally. The DTW non-linearly warps one time sequence to match another given start and end point correspondences, which means accurate location of the endpoints is also needed. We have also shown how the DTW recogniser deals with the problem of unknown endpoints of multimodal sequences embedded in continuous data streams. Third, we proposed an online DTW algorithm so that the DTW recogniser is able to perform real-time processing.

In next chapter, we will use two real world datasets to demonstrate that the proposed DTW temporal fusion agent can perform temporal fusion efficiently, coping with large amounts of multi-sensory information in real-time, and making classification decisions with high accuracy. We also show that the proposed DTW recogniser outperforms HMMs in both case studies.

Chapter 4 Experimental Results

This chapter explores the performance of the proposed DTW temporal fusion agent as shown in Figure 3.5. The performance is evaluated with two real world datasets:

1. A accelerometer dataset acquired from performing two hand gestures (Chambers et al., 2004). The gesture dataset contains six channels of continuous data that are acquired from two accelerometers.

2. A benchmark dataset acquired from carrying a mobile device that has six sensors, and performing the pre-defined user scenarios (Mantyjarvi et al., 2004). This dataset is a complex multimodal sequence that has 29 dimensions.

Performance results of the DTW temporal fusion agent are compared with those of a Hidden Markov Model (HMM) based system.

4.1 Experimental setup and evaluation

All the experiments in this thesis were conducted on a Pentium-4 3.2 GHz CPU with 1 GB of RAM running Windows XP. Most software components of the proposed system are currently implemented in MATLAB, except the DTW recogniser which is implemented in C++ for computational efficiency. We use two real world datasets to explore the performance of the proposed system for both offline and online temporal fusion. To compare our system with a HMM based system, we use the *Bayes Net Toolbox*[1] (Murphy, 2001) to construct an HMM based system for temporal fusion. The efficiency and accuracy of both systems are compared and analysed.

[1] Bayes Net Toolbox for Matlab is available at http://bnt.sourceforge.net/ (last accessed 01/2009).

For offline temporal fusion, we evaluate the performance of the system using the *classification rate* which is the ratio of the number of correctly classified multimodal sequences to the total number of multimodal sequences. For online temporal fusion, we evaluate the performance of the system using both *accuracy rate* (**AR**) and *local minimum deviation* (**LMD**) as illustrated in Figure 4.1. Figure 4.1 shows examples of multimodal sequences to be recognised, embedded in a continuous data stream. Figure 4.1b shows the normalised DTW distances of all the classes for all the sliding windows. The ground truth defines the start and end times of all the multimodal sequences to be recognised (depicted as *ST* and *ET* at the top of the upper waveforms in Figure 4.1a). The winning class for the multimodal sequence buffered in the current sliding window is defined as the one with the lowest DTW distance that is also less than the rejection threshold (the short red dashed lines at the bottom of the lower waveforms of Figure 4.1b). Hence, the **AR** is the ratio of the number of multimodal sequences, of which the winning class at time *ST* matches the ground truth, to the total number of multimodal sequences to be recognised in the data. When the winning class matches the ground truth, we define:

$$LMD \triangleq \frac{|ST - LMT|}{ET - ST} \qquad \textbf{Equation 4.1}$$

where **LMT** (ref. lower waveforms of Figure 4.1b) is defined as the time instance near *ST* where the local minimum of the DTW distances occurs, and indicates the time instance where the best match occurs. As depicted at the bottom left of Figure 4.1b, all the time instances near *ST* of the first multimodal sequence that have the DTW distances less than the rejection threshold (the red dashed line) are correctly classified as class-1 (the blue line), and **LMT** is the time instance where the local minimum of these DTW distances occurs. A reliable online recognition system will have high **AR** and low **LMD**.

4.2 Temporal fusion on gesture sequences acquired from accelerometers

This case study investigates the performance of the DTW temporal fusion agent for temporal fusion with six channels of continuous data. The six channels of continuous data are acquired from two homogenous sensors that consist of accelerometers (ADXL202 dual axis accelerometer from Analog Devices Inc.). The accelerometers can each measure the acceleration of up to ±2g at 150 samples/second in 3-D space. They are housed in small wristwatch sized enclosures worn in the form of a wristband on both wrists. The provided dataset (Chambers et al., 2004) was captured by people wearing the wristbands and mimicking the 12 gestures of a cricket umpire: Cancel Call, Dead Ball, Four, Last Hour, Leg Bye, No Ball, One Short, Out, Penalty Runs, Six, TV Replay, and Wide (Shepherd, 2005). The gesture sequences were captured over different days with four different actors to introduce variability between the movements.

In Section 4.2.1, we use the gesture sequences that have been segmented from the continuous data streams to evaluate the performances of the DTW temporal fusion agent for offline temporal *data in–decision out* and *feature in–decision out* fusion processes. The segmented data for the 12 gestures are shown in the first three columns of Figure 4.2. There are 65 instances (20 from the first actor, and 15 from each of the other three actors) for each of the 12 gestures (780 instances in total), and each instance is a multimodal sequence consisting of six continuous variables. We denote this gesture dataset as G1. Figure 4.2 also shows the significant variability in the data between gesture classes. Instances of the same gesture are also different in length, because the same movements can be performed at different speeds. The results of applying different data pre-processing techniques on the classification rate are studied.

In Section 4.2.2, we use a second set of gesture sequences to evaluate the performance of the DTW temporal fusion agent for the online temporal *feature in–decision out* fusion process. The test data is another set of four continuous data streams that are captured by the four actors respectively, performing each of the 12 gestures once (48 gestures in total which are embedded in four continuous data streams). We denote this gesture dataset as G2 to differentiate it from G1. Unlike the segmented sequences, the test data is a continuous data stream that consists of both gesture sequences and ‘‘noactivity’’ sequences. Figure 4.1a shows one of the continuous data streams that has 12 gestures embedded. We will compare *random*, *minimum*, *multiple*, and *multiple + minimum* selection methods (ref. Section 3.4) on both accuracy rate and local minimum deviation. The influence of different values of *K* (in the *K*-nearest neighbour decision rule ref. Section 3.6) and different options of normalisation factors (ref. Section 3.5.3) on AR are also analysed.

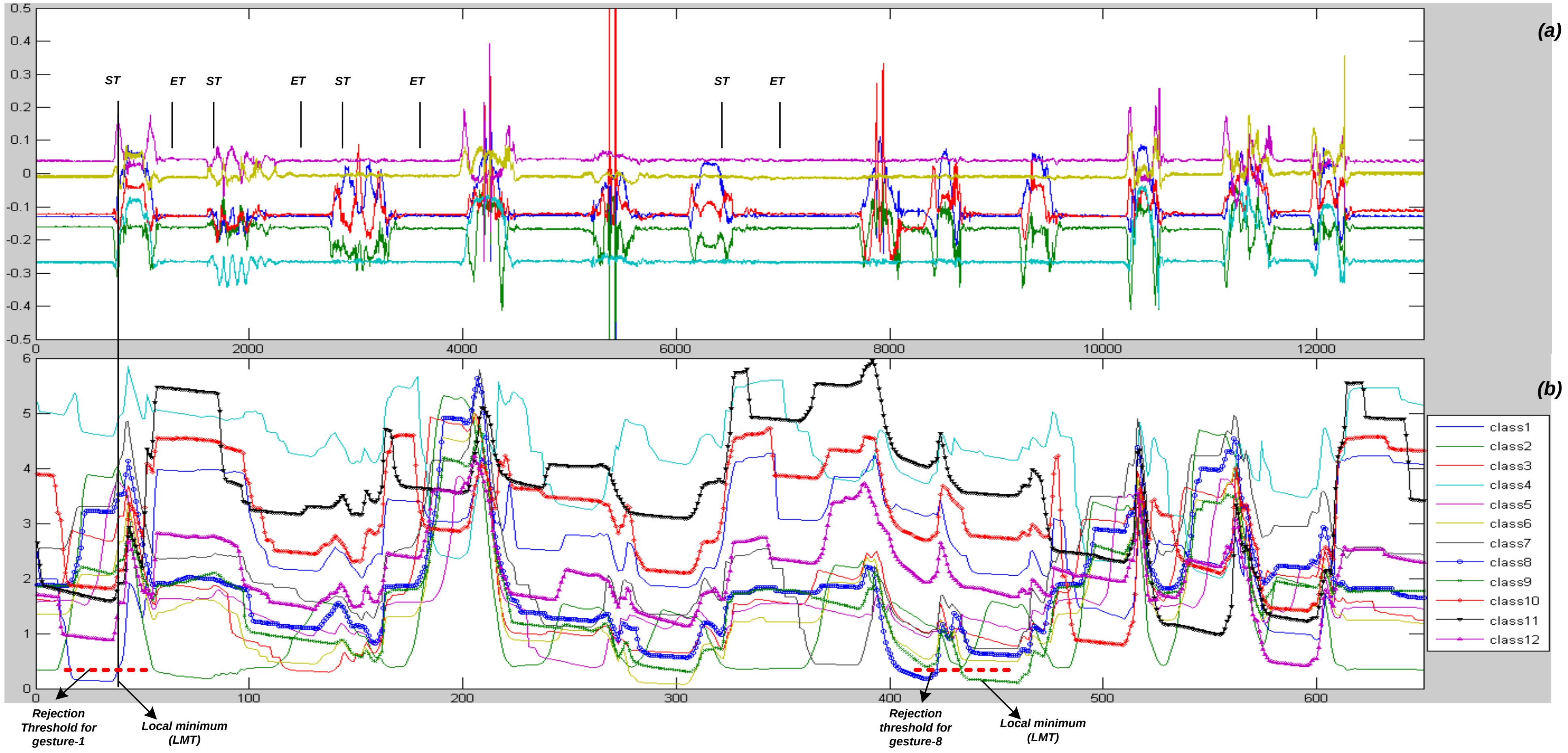


Figure 4.1: (a) The top figure shows the 12 gesture sequences embedded in the six channels of continuous data. (b) The bottom figure shows the DTW distances of all classes calculated at every time instance. The two red dashed lines are rejection thresholds for gesture 1 and 8. The two arrows also point out the local minimum of the DTW distances for gesture 1 and 8.

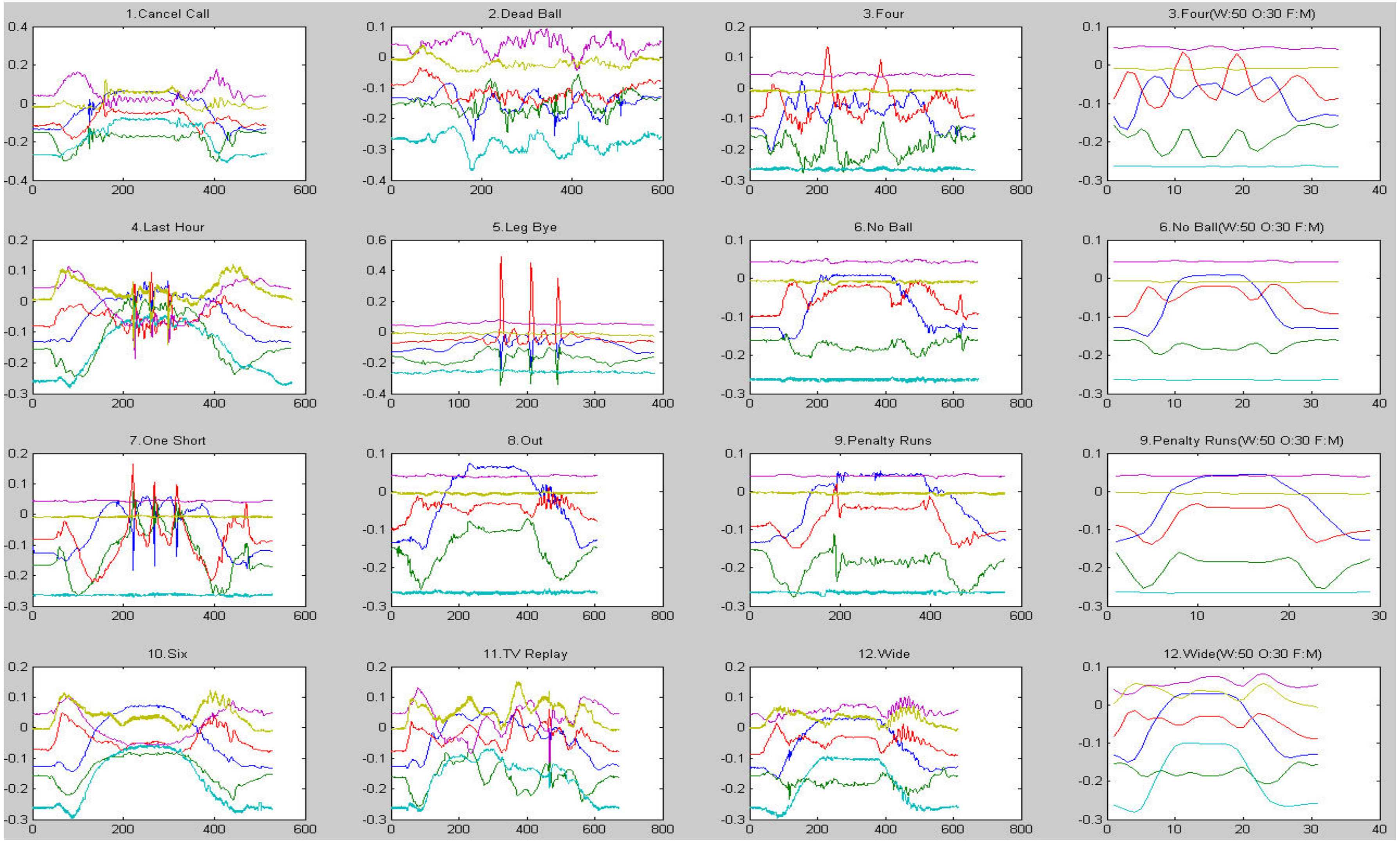


Figure 4.2: The first three columns show the raw data for 12 gestures. Each gesture consisted of six channels of continuous data. The 4th column shows the feature extraction on gestures 3, 6, 9, 12. The moving means are calculated using a sliding window of 50 samples with 30 samples overlap.

In Section 4.2.3, the performance of the DTW temporal fusion agent for offline temporal fusion is compared with a HMM based system by replacing the DTW recogniser of Figure 3.2 with the HMM recogniser. Since the dataset consists of six channels of continuous data and hence have continuous probability distributions, we use the HMM with the *mixture-of-Gaussians* (Rabiner, 1989) to estimate the probability directly instead of dividing the data into bins and making it discrete. For data-pre-processing, we extract the *mean* and *standard deviation* with various sizes of sliding window from the filtered data as described in Section 4.2.1. The database of class templates is replaced by the HMM models. We use various training setups to show the effects of different initializations of the *HMM parameters*, *insufficient training data*, and *different selections of models* on classification rates. After the HMM recogniser calculates the log-likelihood for each class, the nearest neighbour decision rule (NN rule) is used to find the best matching class with the maximum log-likelihood criteria. We also analyse the computations required by both the DTW and HMM recognisers. For online temporal fusion, we extend the DTW to deal with unknown endpoints (ref. Sections 3.5.2 and 3.5.3). However, we cannot compare the performances of both systems for online temporal fusion, since we do not have a reliable implementation of endpoint detection for the HMM recogniser.

4.2.1 Offline temporal fusion

To evaluate the performance of offline temporal data in–decision out and feature in–decision out fusion processes, we first create the database of class templates. Minimum selection is used to extract a single best template from each of the twelve gesture classes. With minimum selection, the intra-class DTW distance is calculated for each of the 65 instances within the same gesture class, and the one with the minimum intra-class distance is selected as the class template to represent that gesture class. For the data in–decision out fusion process, the DTW recogniser measures the similarity between each of the 780 instances and the 12 class templates (both in raw data form) by calculating the DTW distances. For the feature in–decision out fusion process, the same data pre-processing techniques are applied to both the instances and class templates before calculating the DTW distances. The DTW recogniser then passes the results to the decision making module to decide the best matching class.

	$W = 50$ $O = 30$ $F =$ M	$W = 50$ $O = 30$ $F =$ SD	$W = 50$ $O = 30$ $F =$ R	$W = 50$ $O = 30$ F = M&SD	$W = 50$ $O = 30$ $F =$ M&R	$W = 50$ $O = 30$ $F =$ SD&R	$W = 50$ $O = 30$ $F =$ M&SD&R
Euclidean distance	85.51%	76.67%	77.69%	90.13%	83.21%	84.10%	85.90%
Cosine correlation coefficient	91.15%	67.31%	82.95%	95.38%	90.26%	89.36%	92.18%

Table 4.1: W – size of sliding window, O – overlap, F – feature, M – mean, SD – standard deviation, R – root mean square. Euclidean distance and cosine correlation coefficient are the two local distance measure used in the DTW similarity measure. Using M&SD as the feature vector gives the best classification rates.

	$W = 50$ $O = 30$ $F =$ M&SD	$W = 30$ $O = 20$ $F =$ M&SD	$W = 20$ $O = 10$ $F =$ M&SD	$W = 10$ $O = 6$ $F =$ M&SD	$W = 0$ $O = 0$ $F =$ Raw data
Classification rate (Euclidean distance)	90.13%	90.26%	90.13%	88.97%	87.95%
Classification rate (Euclidean distance and filtering)	92.56%	92.82%	92.44%	91.15%	89.10%
Classification rate (Cosine correlation coefficient)	95.38%	96.41%	96.79%	97.05%	97.44%
Classification rate (Cosine correlation coefficient and filtering)	96.41%	97.05%	97.18%	97.18%	97.95%
Average running time (s)	6	20	19	131	2020

Table 4.2: The 2nd and 4th row show the classification rates achieved by using Euclidean distance and cosine correlation coefficient with different window sizes and overlap. The 3rd and 5th rows show that the better classification rates (bold text) can be achieved by applying filtering before feature extraction. The last row shows the average time spent on performing each classification.

	$W = 50,\ O = 30$ $F =$ M&SD, $N =$ Softmax	$W = 50,\ O = 30$ $F =$ M&SD, $N =$ Variance	$W = 50,\ O = 30$ $F =$ M&SD, $N =$ Min–max
Classification rate (Euclidean distance and filtering)	79.74%	90.26%	92.56%
Classification rate (Cosine correlation coefficient and filtering)	80.26%	67.95%	92.05%

Table 4.3: N – normalisation, Softmax, Variance and Min–max are three different normalisation methods. This table shows the classification rate achieved by applying normalisation after filtering and before feature extraction.

As mentioned in chapter 1, the *Data In/Decision Out* fusion process is required and should be considered in addition to the five fusion categories. Thus, the classification rates for the data in–decision out fusion process are shown in the last column of Table 4.2. This result shows that the DTW recogniser has the ability to achieve good classification rates on raw data (450–1200 vectors long with noise present). Although the classification rates are good (87.95 to 97.95%), the time spent on classifying all 780 instances is about 2020 s ($\approx$ 2.6 seconds per instance). The *Band-DP* is not used in offline temporal fusion, because we need a method to decide the size of the band (recall that the size of the band-E is learned only for online temporal fusion). Thus the time complexity for classifying one instance is $O(NIJ)$ (N = number of classes, I = length of class template, J = length of test template). The computation for the DTW recogniser depends on the length of class and test templates. Both raw data templates are six-dimensional sequences with 450–1200 vectors (3 to 8 seconds duration), and hence the time complexity is high. To speed up the fusion process, data pre-processing is required to transform the data for each gesture into a more precise and concise representation.

For the feature in–decision out fusion process, the classification rates for applying different data pre-processing techniques are shown in Table 4.1, Table 4.2, and Table 4.3. First, the classification rates achieved by using different feature vectors are compared in Table 4.1. The selected feature sets are combinations of *mean*, *standard deviation*, and *root mean square* calculated using a sliding window of 50 samples with 30 samples overlap. The 4th column of Figure 4.2 shows an example of extracting mean features from the raw data for gestures 3, 6, 9, 12 (if two features are extracted per sliding window, there will be 12 channels of data). Standard deviation describes the variation in acceleration, and hence should indicate the regions of sharp and smooth movements. Both the mean and root mean square can describe the average intensity of acceleration over regions of the gesture. The results show that using mean and standard deviation achieves the highest classification rates for both local distance measures – achieving 90.13% and 95.38% using Equation 3.1 and Equation 3.2 respectively.

Second, the effects of applying filtering before feature extraction are shown in Table 4.2. The raw data was filtered with a 5 point Gaussian smoothing kernel (s = 0.65) to reduce high frequency noise. The results show the advantage of applying filtering as it improves the classification rates by 1–3% for both Euclidean distance (Equation 3.1) and cosine correlation coefficient (Equation 3.2).

Third, using the Euclidean distance (Equation 3.1) as the local distance measure, the best result (90.26%) is achieved by using a window size of 30 samples with 20 samples overlap. Using the cosine correlation coefficient (Equation 3.2) as the local distance measure, the best result (96.79%) is achieved by using a window size of 20 samples with 10 samples overlap. By comparing the classification rates in Table 4.2, we can see that the cosine correlation coefficient as the local distance measure always outperforms the Euclidean distance.

The effects of applying different sizes of sliding windows and overlap to the classification rates can be seen in Table 4.2. The size of the sliding window determines the number of feature vectors to be extracted from the raw data. More feature vectors mean longer warping paths, and thus decreasing speed of classification. The last cell in the 2nd column of Table 4.2 shows that the DTW recogniser classified all 780 instances and achieved greater than 97% classification rates in about 6 seconds. The speed of classification indicates that our system is capable of performing online temporal fusion, as each gesture is classified in about 0.008 seconds (only 12 DTW comparisons are needed).

Finally, Table 4.3 shows the results of applying three different variable normalisation methods (ref. Section 3.3) after filtering but before feature extraction. Comparing the results of the 3rd cell in row 1 of Table 4.3 to the 2nd cell in column one of Table 4.2, we found that applying filtering and min–max normalisation generates the same result as applying only filtering (both 92.56%). However, if the cosine correlation coefficient is used, applying filtering and normalisation generates worse results than applying only filtering. Therefore, applying normalisation for this dataset is redundant as expected. The six channels of continuous data are acquired from the same type of accelerometer and have the same dynamic range of ±2g. Normalisation is required only if the data are acquired from different types of sensors which might contain both discrete and continuous outputs and have different dynamic ranges.

The 5th row of Table 4.2 shows the best results (i.e., 96% to 98%) that can be achieved using *minimum selection*. Only one gesture sequence is selected from one of the actors to be the class template (recall that the dataset used here is acquired from four actors). Thus these results also demonstrate that the DTW recogniser has the potential to perform actor independent classification with high accuracy. We can further improve the results by using multiple selection such as by selecting one template from each actor for each gesture class. However, the trade-off is that more computation is required. We show that multiple selection can achieve better results than minimum selection in the next section.

4.2.2 Online temporal fusion

One way to perform the online temporal fusion of gesture sequences from continuous data streams is to apply the online segmentation method mentioned in Chambers et al. (2004) in the data pre-processing module to detect regions of no activity. If the actor is stationary, the acceleration magnitude should be very close to the magnitude of gravity. It is unlikely for an actor to stay perfectly stationary, and there will inevitably be movement to some degree, for example very slow swaying from side to side, and hence some small changes in acceleration. The proposed approach (Chambers et al., 2004) is to model the magnitude of gravity (no activity) with a simple Gaussian model. A sliding window approach is then used for calculating the likelihood of the Gaussian model online. If the likelihood is high that means a region of no activity is detected and the data should be segmented at start and finish points given by the inactivity regions. For each gesture segmented by this method from continuous data streams, the DTW recogniser is run to measure the similarity by calculating the DTW distances. The decision making module then determines which of the 12 gesture classes is the best match.

In the proposed system framework, we utilize both the sliding window and the DTW variant to solve the segmentation or endpoints detection problem. We use the gesture sequences that were used in Section 4.2.1 as the training data from which we selected four different sets of class templates to represent each of the 12 gestures. We also derived the *End region* (E_1 and E_2) and the *Rejection threshold* respectively for each gesture as described in Section 3.4. The test data is another set of four continuous sequences that are captured by four actors respectively, performing each of the 12 gestures once (48 gestures in total which are embedded in four continuous data streams). Although we have demonstrated in Section 4.2.1 that the DTW recogniser can classify the gestures in raw data without feature extraction with good results, the consequence is the high computation required. To speed up the online temporal fusion, our data pre-processing module extracts features (*mean* and *standard deviation* within sliding windows of size 50 samples and overlap 30) from the data buffered in the current sliding window as well as the class templates. Therefore, the online DTW recogniser ends up dealing with shorter test and class templates, and is more efficient.

	Min-4	RD-4	Min-1	RD-1
$K = 1$, $NF = 1$	97.92%	93.75%	89.58%	83.33%
$K = 1$, $NF = 2$	93.75%	93.75%	91.67%	77.08%
$K = 1$, $NF = 3$	100.00%	97.92%	91.67%	83.33%
$K = 1$, $NF = 4$	100.00%	97.92%	93.75%	83.33%
$K = 1$, $NF = 5$	100.00%	97.92%	93.75%	87.50%
$K = 2$, $NF = 1$	95.83%	95.83%	92.71%	82.81%
$K = 2$, $NF = 2$	93.75%	93.75%		
$K = 2$, $NF = 3$	97.92%	97.92%		
$K = 2$, $NF = 4$	97.92%	97.92%		
$K = 2$, $NF = 5$	97.92%	97.92%		
$K = 3$, $NF = 1$	93.75%	91.67%		
$K = 3$, $NF = 2$	93.75%	93.75%		
$K = 3$, $NF = 3$	95.83%	95.83%		
$K = 3$, $NF = 4$	95.83%	93.75%		
$K = 3$, $NF = 5$	97.92%	95.83%		
$K = 4$, $NF = 1$	93.75%	95.83%		
$K = 4$, $NF = 2$	91.67%	93.75%		
$K = 4$, $NF = 3$	95.83%	95.83%		
$K = 4$, $NF = 4$	95.83%	95.83%		
$K = 4$, $NF = 5$	95.83%	95.83%		
Average time (s)	44	44	17	17

Table 4.4: The ARs achieved by using four sets of class templates: Min-4, RD-4, Min-1, and RD-1

	Min-4	RD-4	Min-1	RD-1
	$K = 1$	$K = 1$	$K = 1$	$K = 1$
	$NF = 5$	$NF = 5$	$NF = 5$	$NF = 5$
Gesture 1	0.83%	0.83%	0.83%	4.39%
Gesture 2	3.03%	2.02%	4.01%	11.94%
Gesture 3	2.86%	3.81%	3.81%	3.81%
Gesture 4	4.25%	3.33%	5.00%	3.33%
Gesture 5	9.50%	13.17%	12.00%	8.67%
Gesture 6	3.96%	3.96%	5.28%	1.92%
Gesture 7	4.46%	4.63%	4.46%	7.83%
Gesture 8	5.63%	7.49%	5.63%	7.51%
Gesture 9	4.32%	2.27%	4.32%	5.16%
Gesture 10	2.50%	2.63%	3.54%	2.22%
Gesture 11	3.36%	4.22%	3.36%	2.26%
Gesture 12	6.34%	2.50%	2.50%	7.50%

Table 4.5: The LMDs calculated for all 12 gestures by using Min-4, RD-4, Min-1, and RD-1

The results for AR are shown in Table 4.4. The values of AR were obtained for different sets of class templates, different values of K (in the K-nearest neighbour decision rule ref. section 3.6) from $K = 1$ to $K = 4$, and different options of normalisation factors from $NF = 1$ to 5 (ref. Section 3.5.3).

First, we compare the *random*, *minimum*, *multiple*, and *multiple + minimum* selection methods as mentioned in section 3.4. The four different sets of class templates are selected from the training data:

1. by using minimum selection to select four class templates per gesture that have least *intra-class DTW distances* (Min-4, *multiple minima*),

2. by selecting four class templates per gesture randomly (RD-4),

3. by using minimum selection to select one class template per gesture (Min-1),

4. and by selecting one class template per gesture randomly (RD-1).

Both values of AR obtained by using one template per gesture (Min-1 and RD-1) are worse than those ARs using four templates per gesture (Min-4 and RD-4). This shows that the AR of the DTW recogniser can be improved by increasing the number of class templates per class, but the trade-off is that the computational time is raised from an average of 17 to 44 s.

Second, the method used to select the class template is also critical. Minimum selection outperformed random selection in both cases as the average values of AR of Min-1 and Min-4 are about 10% and 1% higher than these of RD-1 and RD-4 respectively. The values of AR achieved by using both *normal* and *average* selection methods are not shown, because they produced similar values of AR for the minimum selection method.

Third, the value for K should be carefully chosen as the number of class templates increases per class. If there is only one class template per class, K can only be equal to 1. For four class templates per class, $K = 1$ and $K = 2$ rules yield higher values of AR than $K = 3$ or $K = 4$ rules. For $K = 1$, both Min-4 and RD-4 yield the best values of AR of 100% and 98% respectively. On the other hand, the best values of AR for Min-4 and RD-4 for $K = 4$ declined to 95.8%.

Finally, applying different options of normalisation factors also affects the values of AR (ref. Section 3.5.3). For $K = 1$, Min-4, RD-4 and Min-1 yield higher values of AR by applying

$NF = 3,4, or\ 5$, increasing AR about 2% to 4% than $NF = 1$ (no normalisation). For $K = 2, 3, and\ 4$ of Min-4, RD-4, applying $NF = 3, 4\ or\ 5$ always yields better values of AR than $NF = 1$. In this experiment, applying normalisation did not improve the values of AR dramatically (compare to $NF = 1$); because the 12 gesture sequences have similar lengths in the test data.

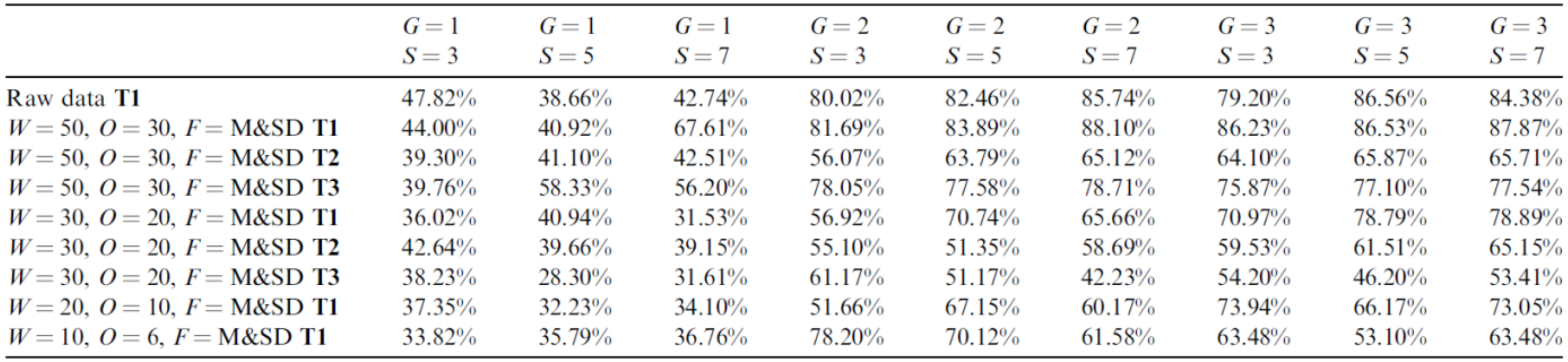
The *local minimum deviations* (LMDs) of all 12 gestures from Min-4, RD-4, Min-1, and RD-1 are shown in Table 4.5. Most of the values of LMD are less than 7%, except for two special cases. First, it should be noted that RD-1 has several large values of LMD that are greater than 7%. This is because one class template is selected randomly for each gesture, and the selected one might have a large DTW distance to the test sequences which results in higher values of LMD. Second, the values of LMD for the 5th gesture are greater than 8% for all four sets of class templates. This is because the 5th gesture sequence performed by the third actor is quite different from the training data, and it has a value of LMD greater than 30% for all four sets of class templates. From the values of LMD between the selected class templates and test data, we can determine if the selected class templates are good representatives.

The total length of the test data is about 56,000 samples (around 14,000 samples for each), corresponding to approximately 400 seconds duration. Using Min-1 or RD-1, the online recognition of the test data took about 17 seconds and achieved highest values of AR of 93.75% and 87.5% respectively. The highest values of AR of 100% and 97.92% were achieved in 44 seconds using Min-4 and RD-4 respectively, which only took about 0.016 second (performing 48 DTW comparisons) to classify the test templates starting at every time instance. The results demonstrate that the DTW recogniser is both fast and accurate for online recognition of gesture sequences acquired from the accelerometers.

4.2.3 Comparing the DTW and HMM based systems

Table 4.6 shows all the classification rates achieved by the HMM recogniser for offline temporal fusion. We applied the same data-pre-processing as in Table 4.2. In addition, all the gesture sequences are filtered before feature extraction. To train the HMM models, we use the Expectation–Maximization (EM) implementation (Murphy, 2001) of the *Baum–Welch algorithm* (Rabiner, 1989) to train the model parameters i.e., $\lambda = (A, B, \pi)$ as described in section 2.2.2. Like gradient methods, the EM algorithm also suffers from the problem of local maxima and its performance is sensitive to the initial estimates of the HMM parameters, e.g., the classification rate of a given bad initial estimate is much worse (e.g., 50%) than when a good initial estimate (e.g., 88%) is used. The first key problem is therefore how to choose the initial estimates of the HMM parameters so that the local maximum is the global maximum of the likelihood function. Unfortunately, there is no simple answer to this problem. The heuristic we used is to run the EM training with multiple initializations and choose the best five HMM models for each gesture class. In our initialization, $\boldsymbol{\alpha}$ and $\boldsymbol{\pi}$ are randomly initialized. We use the *Segmental K-means Segmentation* method to initialize $\boldsymbol{\beta}$ (Rabiner, 1989), since a good initialization of $\boldsymbol{\beta}$ is essential for continuous distributions (Rabiner, 1985). All the classification rates shown in Table 4.6 are the average of these classification rates obtained by the five HMM models for each gesture class.

The second problem associated with training HMM parameters is *insufficient training data*. To demonstrate this problem, we use three training datasets that are selected from G1: T1 – 5 instances are randomly selected from each of the 4 actors (20 training instances for each gesture class), T2 – only the 20 instances of the first actor are used for each gesture class, T3 – 1 training instance for each gesture class which are the class templates selected by minimum selection and used in Section 4.2.1. To show the impact of insufficient training data, we can compare the classification rates of the 3rd–5th rows and the 6th–8th rows of Table 4.6. With the same data pre-processing, most of the HMM models trained using T1 achieve much better classification rates than both T2 and T3. Applying data pre-processing with parameters W = 50, O = 30, almost all the results in the 3rd row are higher than those in the 4th and 5th rows. Similarly, applying W = 30, O = 20, almost all the results in the 6th row are also higher than these of the 7th and 8th rows. This is because the HMM models trained with both T2 and T3 did not capture the variability's introduced by different actors, and the insufficient number of training sequences was not adequate to learn the precise model parameters. Thus, if there are insufficient occurrences of different model events (e.g., symbol occurrences within states), the EM training may not give good estimates of the model parameters.



	$G=1$ $S=3$	$G=1$ $S=5$	$G=1$ $S=7$	$G=2$ $S=3$	$G=2$ $S=5$	$G=2$ $S=7$	$G=3$ $S=3$	$G=3$ $S=5$	$G=3$ $S=7$
Raw data **T1**	47.82%	38.66%	42.74%	80.02%	82.46%	85.74%	79.20%	86.56%	84.38%
$W=50$, $O=30$, $F=$ M&SD **T1**	44.00%	40.92%	67.61%	81.69%	83.89%	88.10%	86.23%	86.53%	87.87%
$W=50$, $O=30$, $F=$ M&SD **T2**	39.30%	41.10%	42.51%	56.07%	63.79%	65.12%	64.10%	65.87%	65.71%
$W=50$, $O=30$, $F=$ M&SD **T3**	39.76%	58.33%	56.20%	78.05%	77.58%	78.71%	75.87%	77.10%	77.54%
$W=30$, $O=20$, $F=$ M&SD **T1**	36.02%	40.94%	31.53%	56.92%	70.74%	65.66%	70.97%	78.79%	78.89%
$W=30$, $O=20$, $F=$ M&SD **T2**	42.64%	39.66%	39.15%	55.10%	51.35%	58.69%	59.53%	61.51%	65.15%
$W=30$, $O=20$, $F=$ M&SD **T3**	38.23%	28.30%	31.61%	61.17%	51.17%	42.23%	54.20%	46.20%	53.41%
$W=20$, $O=10$, $F=$ M&SD **T1**	37.35%	32.23%	34.10%	51.66%	67.15%	60.17%	73.94%	66.17%	73.05%
$W=10$, $O=6$, $F=$ M&SD **T1**	33.82%	35.79%	36.76%	78.20%	70.12%	61.58%	63.48%	53.10%	63.48%

Table 4.6: W – size of sliding window, O – overlap, F – feature, M – mean, SD – standard deviation, G –number of mixture-of-Gaussians per hidden state, S – number of hidden states, T1 – first training dataset, T2 – second training dataset, T3 – third training dataset. This table shows the average classification rates achieved by the HMM recogniser for different training setups.

The remaining issue in training HMM models with the mixture-of-Gaussians outputs is the choice of model size (i.e., number of hidden states) and choice of mixture size (i.e., number of Gaussian mixtures used for each hidden state). Unfortunately, there is no simple way to make such a choice. This choice must be made depending on the sequences being modelled. If the size of model and mixtures are too small for the sequences, the model does not capture all the events in the training sequences. If the size of model and mixtures are too large, some of the HMM model parameters do not have sufficient occurrences of events in the training sequences. Both cases result in low classification rates. Each column of Table 4.6 shows the classification rate achieved by different sizes of G (number of Gaussian mixtures per state) and $\boldsymbol{S}$ (number of hidden states). For instance, the models trained with raw data achieved the best classification rate when $G = 3$ and $S = 13$. The models trained with W = 50 and O = 30 achieve the best classification rate when $G = 2$ and $S = 7$.

In summary, compared to the simple training procedures of the DTW recogniser (ref. Section 3.4), the training of the optimal HMM model is complex. First, we have to try multiple initializations until good models are obtained to achieve good classification rates. For the DTW recogniser, we only need to select good class templates and the selected templates achieve the same classification rates consistently. Second, we must use sufficient training data to estimate the HMM model parameters. However, the DTW recogniser requires merely a good class template (i.e., minimum selection) for each of the multimodal sequence to be recognised, and the results are robust to the variability introduced by different actors. For example, the DTW recogniser that used T3 as class templates obtained much better results than the HMM recogniser trained with T3. Finally, the choice of model and mixture size for the HMM parameters crucially affect the classification rate. For the DTW recogniser, there are no additional parameters required in selecting the class template.

Our HMM recogniser uses the *Forward–Backward Procedure* (Rabiner, 1989) to measure the similarity between each of the 780 instances and the 12 trained HMM models by calculating the log-likelihood. The HMM recogniser then passes the results to the decision making module to make the correct class decisions. To perform the offline temporal *data in–decision out* fusion process, the gesture sequences G1 are filtered, but without applying feature extraction. The training dataset is T1. The classification rates achieved by the HMM recogniser are shown in the second row of Table 4.6. The best average classification rates are 86.5% and 85.7% when $G = 3$ and $S = 5$, and $G = 2$ and $S = 7$ respectively. Comparing these results to the last column of Table 4.2, the DTW recogniser outperforms the HMM recogniser and achieves better classification rates (e.g., 97.9%).

Table 4.6 shows the classification rates for the offline temporal *feature in–decision out* fusion process. The gesture sequences are filtered, and the *mean* and *standard deviation* are extracted in each sliding window. Using W = 50, O = 30 and training dataset T1, the best results achieved by the HMM recogniser are 88.1% and 87.8% when $G = 2$ and $S = 7$ and $G = 3$ and $S = 7$ respectively. The DTW recogniser obtained better results (e.g., 92.5% and 96.4%) as shown in the second column of Table 4.2. Unlike the HMM models that require training sequences from each actor, the DTW recogniser uses only one good gesture sequence from one of the actors as the class template. Although the HMM recogniser has the potential to perform actor independent classification, it requires sufficient training datasets that address all the variability introduced by different actors. For W = 30 and O = 20, W = 20 and O = 10, and W = 10 and O = 6, the best classification rates obtained by the HMM recogniser are 78.8% ($G = 3, S = 7$), 73.9% ($G = 3, S = 3$), and 78.2% ($G = 2, S = 3$). All these results are less than 80%. The DTW recogniser achieved much better results: 91–93% (Euclidean distance and filtering) and 96–98% (cosine correlation coefficient and filtering) respectively. The classification rates achieved by the DTW recogniser with different data pre-processing techniques outperform those achieved by the HMM, and the training procedures of DTW are also more efficient than for the HMM.

To compare the performance of both systems for online temporal fusion, the HMM recogniser must have a reliable solution for the endpoint detection of multimodal sequences embedded in continuous data streams. Since we do not have a reliable implementation for this endpoint detection, we cannot calculate the values of AR and LMD as in Section 4.2.2. However, we manually segmented the 48 gesture sequences from the second gesture dataset G2 according to the ground truth and classified them. Table 4.7 shows the classification rates achieved in this way. First, the HMM recogniser uses the best HMM models obtained in previous experiments (i.e., the models that can achieve >85% as shown in the third row of Table 4.6) to classify these 48 segmented gesture sequences. The classification rates achieved with different *G* and *S* are shown in the first row of Table 4.7. Second, we use the entire dataset G1 to train another set of HMM models for the HMM recogniser (65 training instances per gesture class). The HMM recogniser with the models trained using two different training datasets and different values of *G* and *S* all achieved the same classification rates – 70.83%. This result clearly shows that even if we have a robust implementation of endpoint detection, the HMM recogniser can only achieve a value of AR that is 70.83%. Comparing the results of Table 4.7 with the values of AR in Table 4.4, the DTW recogniser again outperforms the HMM recogniser.

	$G = 2$ $S = 7$	$G = 3$ $S = 3$	$G = 3$ $S = 5$	$G = 3$ $S = 7$
$W = 50$, $O = 30$, $F =$ M&SD **T1**	70.83%	70.83%	70.83%	70.83%
$W = 50$, $O = 30$, $F =$ M&SD **G1**	70.83%	70.83%	70.83%	70.83%

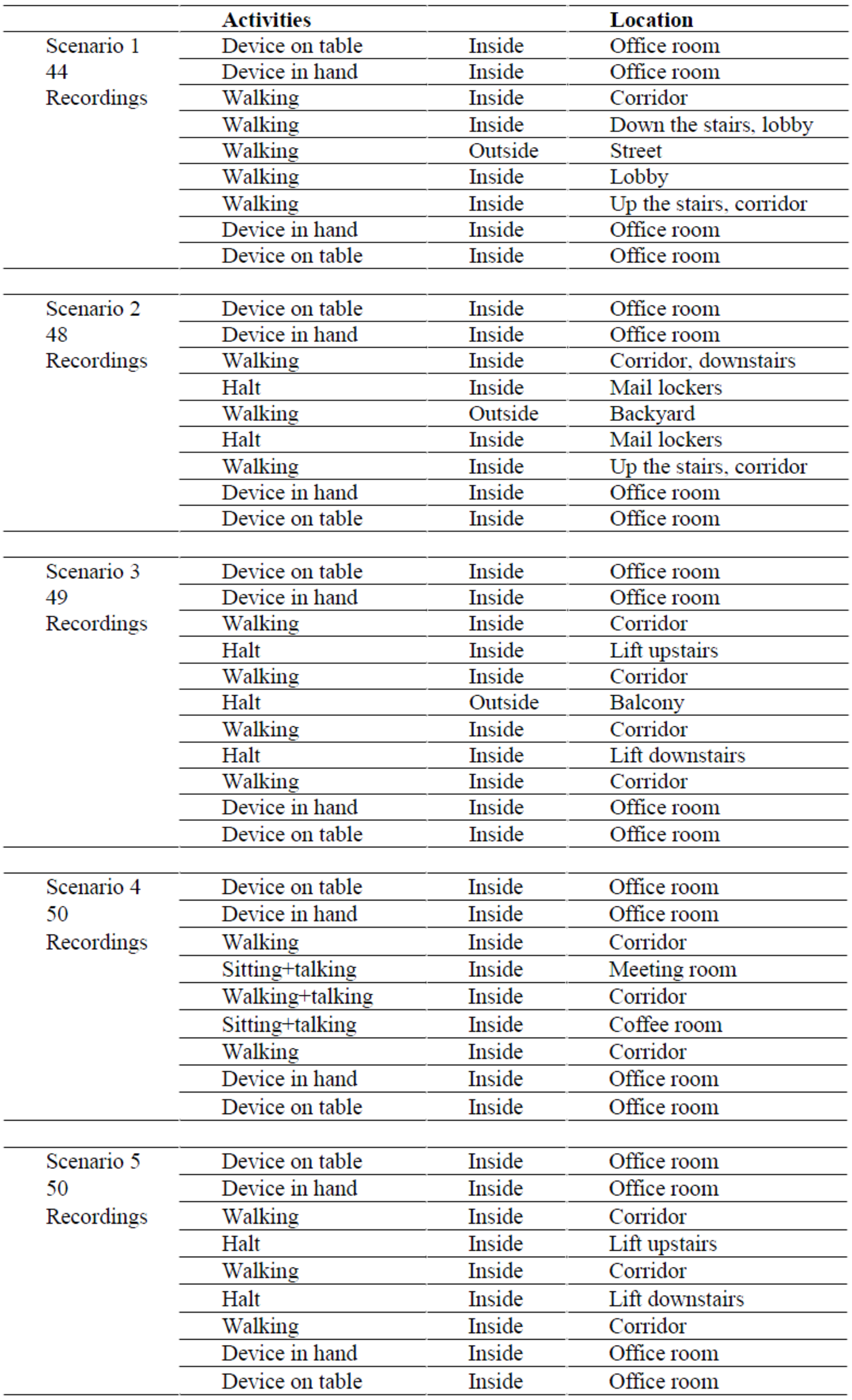
Table 4.7: W – size of sliding window, O – overlap, F – feature, M – mean, SD – standard deviation, G – number of the mixture-of-Gaussians per hidden state, S – number of hidden states, T1 – first training dataset, T4 – fourth training dataset. This table shows the average classification rates achieved by the HMM recogniser for gesture sequences G2.

In offline temporal fusion, the time complexity of the DTW recogniser to calculate the DTW distances for classifying one instance is $O(NIJL)$ (i.e., N = number of classes, I = length of class template, J = length of test template, and L = the time required to calculate the local distance measure (ref. Section 3.5.2)). The overall time complexity grows linearly as the length of test and class templates increases. The computation of L grows linearly as the dimensions of feature vector increase, but L has only a minor effect on the overall computation. On the other hand, the HMM recogniser uses the *Forward–Backward* Procedure to calculate the log-likelihoods for classifying one instance, and the time complexity is $O(NS^2JP)$ (i.e., N = number of classes, J = length of sequence, S = number of hidden states, and P = the time required to calculate the emitting probability of observation). The time complexity grows linearly as the length of sequences increases and exponentially as more hidden states are used. The computation of P also increases exponentially as the dimension of the feature vector increase, because it involves expensive matrix operations such as the calculation of the inverse and determinant of the covariance matrix (the size of the covariance matrix is V^2, where V is the dimension of the feature vector). Thus P has a major effect on the overall computation as V increases. Generally speaking, the time complexity of the DTW recogniser is lower than the HMM recogniser when the lengths of sequences are short. However, if the lengths of sequences are long, the HMM recogniser will be more efficient. For the best performance, we should always apply feature extraction from the raw data to keep the multimodal sequences short so that the fusion process can be achieved efficiently in real-time.

4.3 Temporal fusion on multimodal sequences acquired from heterogeneous sensors

The second case study aims at demonstrating that the system can achieve good results on more complex multisensory data. The data used here was presented in Mantyjarvi et al. (2004). This dataset has been proposed as a suitable benchmark for evaluating context recognition algorithms, and is publicly available online[2]. The data is obtained from sensors placed in a small sensor box that is then attached to a mobile handheld device. Heterogeneous sensors are used to capture this data and include *accelerometers with 3 axes*, an *illumination sensor*, a *thermometer*, a *humidity sensor*, a *skin conductivity sensor*, and a *microphone*. The data is collected by two users carrying the mobile device and repeating each of the five predefined user scenarios more than 44 times as shown in Table 4.8. The duration of each scenario varies from 1 to 5 min and the sensor data is sampled every second. The data set is provided in an ASCII flat file, and there are a total of 32 columns and 46,045 rows corresponding to approximately 12.8 hours of data. The first three columns include: the ID number of the scenario (SID), the ID number of the repetition of the scenario (RID) and time from the beginning of repetition in seconds. The rest of the 29 columns are *context atoms*.

[2] http://www.cis.hut.fi/jhimberg/contextdata/index.shtml (last accessed 01/2009).



	Activities		Location
Scenario 1	Device on table	Inside	Office room
44	Device in hand	Inside	Office room
Recordings	Walking	Inside	Corridor
	Walking	Inside	Down the stairs, lobby
	Walking	Outside	Street
	Walking	Inside	Lobby
	Walking	Inside	Up the stairs, corridor
	Device in hand	Inside	Office room
	Device on table	Inside	Office room
Scenario 2	Device on table	Inside	Office room
48	Device in hand	Inside	Office room
Recordings	Walking	Inside	Corridor, downstairs
	Halt	Inside	Mail lockers
	Walking	Outside	Backyard
	Halt	Inside	Mail lockers
	Walking	Inside	Up the stairs, corridor
	Device in hand	Inside	Office room
	Device on table	Inside	Office room
Scenario 3	Device on table	Inside	Office room
49	Device in hand	Inside	Office room
Recordings	Walking	Inside	Corridor
	Halt	Inside	Lift upstairs
	Walking	Inside	Corridor
	Halt	Outside	Balcony
	Walking	Inside	Corridor
	Halt	Inside	Lift downstairs
	Walking	Inside	Corridor
	Device in hand	Inside	Office room
	Device on table	Inside	Office room
Scenario 4	Device on table	Inside	Office room
50	Device in hand	Inside	Office room
Recordings	Walking	Inside	Corridor
	Sitting+talking	Inside	Meeting room
	Walking+talking	Inside	Corridor
	Sitting+talking	Inside	Coffee room
	Walking	Inside	Corridor
	Device in hand	Inside	Office room
	Device on table	Inside	Office room
Scenario 5	Device on table	Inside	Office room
50	Device in hand	Inside	Office room
Recordings	Walking	Inside	Corridor
	Halt	Inside	Lift upstairs
	Walking	Inside	Corridor
	Halt	Inside	Lift downstairs
	Walking	Inside	Corridor
	Device in hand	Inside	Office room
	Device on table	Inside	Office room

Table 4.8: Descriptions of user scenarios. From: (Mantyjarvi et al., 2004)

The benchmark data is pre-processed by processing the raw sensor signals into context atoms, i.e., extracting features from the raw signals and then quantising the range of feature values into the range [0, 1] with fuzzy sets. For example, the raw illumination signals are processed into four context atoms describing the level of illumination calculated using the mean value within a sliding window. Context atoms are assigned by quantising the dynamic range of the calculated mean value with fuzzy sets: *Bright(x), Normal(x), Dark(x), Total-darkness(x) [0, 1]*. Since the raw data of the benchmark dataset is not released for public use, we cannot apply our own steps of data pre-processing. For filtering, they have applied an *Infinite Impulse Response Butterworth* low pass filter on both accelerometer and illumination signals. For variable normalisation, they have used fuzzy sets to quantise the dynamic range of the calculated feature values. Various feature extraction methods have also been applied to the raw data of each sensor. For segmentation, each repeated user scenario in this benchmark data has already been segmented based on the SID and RID values. We know the start and end time for each repeated user scenario.

However, it is not easy to segment this benchmark data automatically, since there is no ‘‘no activity’’ region as can be found in gesture sequences. More details on the processing of sensor data into context atoms are available in Mantyjarvi (2003). Hence, each user scenario is described by a more complex multimodal sequence, which has 29 dimensions (mix of both discrete and continuous variables) of feature values per second over durations of 1 to 5 mins. More detailed information on this dataset is available in Mantyjarvi et al.(2003; 2004).

From the SID and RID values in the datasets, we know the start and end time for each repeated user scenario. In Section 4.3.1, we use the segmented multimodal sequences to evaluate the performance of the proposed system for the offline temporal *feature in–decision out* fusion process. We also compare the results to the HMM based system.

In Section 4.3.2, we use two test cases to evaluate the performance of the system for the online temporal *feature in–decision out* fusion process. In the first test case, we use the benchmark data as it is in the ASCII flat file to test our online recognition system. There are a total of 240 repetitions of the five user scenarios and each repetition is followed by another repetition without any other data in between. Since the feature vectors in the ASCII file are annotated with picture sequences (Mantyjarvi et al., 2004), we can identify the feature vectors while the mobile device is on the table and doing nothing. To make the online recognition task more realistic, we created the second test dataset by rearranging the order of the 240 repetitions of five user scenarios randomly and also inserting 1–3 min of ‘‘on the table’’ feature vectors in between each repetition. The total length of the second test data becomes 75,687 vectors, corresponding to approximately 21 hours of data. The influence of different selection methods for the class templates (ref. Section 3.4) on both accuracy rate and local minimum deviation are studied. The values of AR achieved by using both normal and average selection methods are not shown, because they produced similar results as the minimum selection method. Furthermore, the influences of using different values of *K* (in the K-nearest neighbour decision rule ref. Section 3.6) and different options of normalisation factors (ref. Section 3.5.3) on the value of AR are also analysed. As before, we cannot compare the proposed system to the HMM based system for online temporal fusion, because a solution to the precise endpoint detection is necessary for the HMM based system. However, even if we have a robust implementation of endpoint detection for the HMM recogniser, the HMM recogniser can only achieved results for ARs that are less than the best results obtained during offline temporal fusion i.e., 88.19% (ref. Table 4.11).

4.3.1 Offline temporal fusion

To train the DTW recogniser, minimum selection is again used to select the best class template for each of the five user scenarios. The DTW recogniser then performs offline classification by measuring the similarity between each of the 235 instances and the five selected class templates and determining which class each instance belongs to. Note that there should be 250 repeated user scenarios in total, however, some are missing from the original data and five of them are selected to represent the class templates. The classification rates are shown as a confusion matrix in Table 4.9 and Table 4.10. The DTW recogniser with both local distance measures achieve good results (92.08% and 97.5%) and spent only 6–9 seconds to classify all 235 instances of the five user scenarios.

Scenario 1	44	0	0	0	0
Scenario 2	1	40	7	0	0
Scenario 3	0	0	41	0	7
Scenario 4	0	0	0	50	0
Scenario 5	0	0	0	4	46

Table 4.9: Confusion matrix produced by the system that uses Euclidean distance as local distance measure for the DTW recogniser. The classification rate is 92.08%, and the average running time is 6.45 s.

Scenario 1	44	0	0	0	0
Scenario 2	2	44	2	0	0
Scenario 3	0	0	47	0	1
Scenario 4	0	0	0	50	0
Scenario 5	0	0	0	1	49

Table 4.10: Confusion matrix produced by the system that uses cosine correlation coefficient as local distance measure for the DTW recogniser. The classification rate is 97.50%, and the average running time is 8.89 s.

Each feature vector of the benchmark dataset has 29 dimensions and is a mix of both discrete and continuous variables. This causes another problem for the HMM based system when estimating the emitting probability of observations. A more complex HMM with a mixture of discrete and continuous outputs is suggested e.g., (Xue & Govindaraju, 2006). Since we do not have such an implementation, we again use the HMM with the mixture-of-Gaussians outputs. To demonstrate the problem of insufficient training data and to compare it with the DTW recogniser, we use two training datasets: **T1** – 20 instances are randomly selected from each class and **T2 –** 1 training instance for each class which are the class templates selected by minimum selection and used in Table 4.10. The training procedure is the same as described in Section 4.2.3, and the classification rates of Table 4.11 are the average of five classification rates achieved by five different model initializations.

The HMM recogniser uses the *Forward–Backward* Procedure (Rabiner, 1989) to measure the similarity between each of the 235 instances and the five trained HMM models by calculating the log-likelihood. The HMM recogniser then passes the results to the decision making module to make the correct class decisions. First, the average classification rates for T2 and different values of G and S are shown in the second row of Table 4.11. The best result is 78.00% when $G = 3$ and $S = 3$, and took 22 seconds to classify all 235 instances of the five user scenarios. The DTW recogniser used T2 as class templates and achieved a much better result of 97.5% (and took only 8.89 seconds) as shown in Table 4.10. Second, the HMM models are trained with T1 which has 20 training instances per class (nearly half of the entire dataset). By comparing the first row of Table 4.11 to the second row, it can be seen that the HMM recogniser trained with T1 performs better than T2. The best result is 88.19% when $G = 2$ and $S = 9$ (took 34 seconds) which is still worse than the DTW recogniser.

The benchmark dataset has also been analysed using the *Symbol Clustering Map* (SCM) technique (Flanagan, Himberg, & Mantyjarvi, 2003; Flanagan, Mantyjarvi, & Himberg, 2002;

Himberg et al., 2003). This method is similar to *Self-Organizing Maps* (Kohonen, 2000). Two levels of SCM are used to classify the five scenarios. The first level clusters the instantaneous patterns and then the second level finds the temporal relationships. The results presented in Himberg et al. (2003) use two levels of SCM and can only identify four clusters from the five classes of user scenarios. The method cannot distinguish well between scenarios 1 and 2. However, SCM has the advantage of performing unsupervised clustering of the user scenarios. Although we have to train and find the best class template for each of the five user scenarios for the DTW recogniser, the classification results are better.

	$G=1$ $S=3$	$G=1$ $S=5$	$G=1$ $S=7$	$G=1$ $S=9$	$G=2$ $S=3$	$G=2$ $S=5$	$G=2$ $S=7$	$G=2$ $S=9$	$G=3$ $S=3$	$G=3$ $S=5$	$G=3$ $S=7$	$G=3$ $S=9$
T1	75.58%	80.25%	75.16%	78.66%	83.41%	87.08%	86.08%	88.19%	85.25%	86.25%	86.75%	86.24%
T2	74.91%	72.58%	74.66%	73.50%	74.58%	76.83%	75.16%	76.83%	78.00%	75.50%	76.41%	75.83%
Time (s)	16	19	21	23	19	22	27	34	22	28	35	43

Table 4.11: G – number of the mixture-of-Gaussians per hidden state, S – number of hidden states, T1 – first training dataset, T2 – second training dataset. This table shows the average classification rates achieved by the HMM recogniser for different training datasets and different values of G and S.

	T1-Min-5	T1-Min-1	T2-Min-5	T2-RD-5	T2-Min-1	T2-RD-1
$K=1$, $NF=1$	71.25%	67.92%	70.83%	69.58%	67.92%	71.67%
$K=1$, $NF=2$	93.75%	91.25%	80.83%	80.00%	77.50%	72.92%
$K=1$, $NF=3$	98.33%	96.67%	94.58%	92.50%	93.33%	88.75%
$K=1$, $NF=4$	98.33%	98.33%	92.50%	94.58%	92.08%	90.42%
$K=1$, $NF=5$	98.33%	97.92%	94.17%	93.33%	92.50%	88.75%
$K=2$, $NF=1$	70.83%		70.42%	66.25%		
$K=3$, $NF=2$	92.50%		77.50%	71.67%		
$K=3$, $NF=3$	97.92%		94.58%	91.67%		
$K=3$, $NF=4$	98.33%		92.92%	92.92%		
$K=3$, $NF=5$	98.33%		94.17%	92.92%		
$K=5$, $NF=1$	68.33%		67.50%	60.83%		
$K=5$, $NF=2$	92.92%		77.92%	69.58%		
$K=5$, $NF=3$	97.92%		94.58%	92.50%		
$K=5$, $NF=4$	97.92%		94.17%	93.33%		
$K=5$, $NF=5$	98.33%		94.17%	92.50%		
Average time (s)	13,334	2564	23,312	23,241	5545	5844

Table 4.12: The ARs for both T1 and T2 achieved by using different sets of class templates, K values, and NF options.

	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
T1-Min-5, $K=1$, $NF=3$	0.77%	0.35%	0.30%	1.07%	0.78%
T1-Min-1, $K=1$, $NF=4$	2.80%	0.43%	0.41%	2.64%	0.79%
T2-Min-5, $K=1$, $NF=3$	3.60%	0.49%	0.35%	0.83%	2.63%
T2-RD-5, $K=1$, $NF=4$	2.63%	0.46%	0.36%	0.82%	3.50%
T2-Min-1, $K=1$, $NF=3$	0.60%	0.58%	1.70%	1.11%	1.10%
T2-RD-1, $K=1$, $NF=5$	5.83%	0.70%	5.08%	3.40%	2.66%

Table 4.13: Some LMDs for the five scenarios

4.3.2 Online temporal fusion

To perform the online temporal feature in–decision out fusion process for the five user scenarios, the DTW recogniser has to deal with more complex multimodal sequences that have about 80–300 feature vectors (1–5 mins) and larger dimensions (29 dimensions with a mix of both discrete and continuous variables). To train the DTW recogniser, we selected four different sets of class templates from the benchmark data to represent each of the five user scenarios, and we also derived the end region ($E1$ and $E2$) and the rejection threshold respectively for each user scenario as described in Section 3.4.

In Table 4.12, the 2nd and 3rd columns are the values of AR for the first test data set (T1), and from the 4th to 7th columns are the values of AR for the second test data set (T2). The values of AR were obtained for four different sets of class templates: Min-5 (use five class templates to represent each scenario), RD-5, Min-1, and RD-1, different values of K, and different options of normalisation factors from $NF = 1 \ to \ 5$. The best value of AR on T1 is 98.33%, and the best value of AR on T2 is 94.58%. Comparing the same settings of class templates, K and NF, most values of AR obtained for T1 are better than for T2. This is because the scenarios in T1 are not preceded and followed by any noise, whereas the scenarios in T2 are separated by 1–3 mins of ‘‘on the table’’ feature vectors. In a real situation, the multimodal sequences will usually be preceded and followed by noise, and the value of AR obtained will also be reduced by noise. Second, we can see that applying normalisation on DTW distances improves the value of AR dramatically (e.g., comparing the values of AR from $NF = 3, 4, or\ 5$ to $NF = 1$). This is because the average length of all repetitions of the 4th user scenario is only 100 seconds compared to the average length of the other user scenarios (238, 206, 217, and 205 seconds for scenarios 1, 2, 3 and 5 respectively), for which the differences are high. From observation of the confusion matrices of all ARs, most misclassifications occurred when the 4th user scenarios are misclassified as a 5th user scenario. Therefore, normalisation on the DTW distance is required if the lengths of the multimodal sequences vary greatly between the classes. The options of $NF = 3, 4, and\ 5$ always have similar values of AR and are much higher than $NF = 1\ and\ 2$.

Table 4.13 shows some of the local minimum deviations (LMDs) of all five user scenarios. Most of the LMDs are less than 3%, except T2-RD-1 that has few values of LMD that are greater than 3%, which again shows that the randomly selected class templates are not good representatives. The values of LMD calculated from these two test data sets (T1 and T2) are low, corresponding to the high values of AR obtained.

For T1, the online temporal fusion took 13,334 seconds and achieved the highest value of AR of 98.33%. When the DTW recogniser receives each new feature vector, it only takes 0.29 seconds (13,334/46,045 = 0.29) to calculate new warping paths and classify the data currently buffered in the sliding window. For T2, the recognition process took 23,241–23,312 seconds to achieve the best AR value of 94.58%. To classify each updated sliding window, only takes 0.31 seconds. Compared to the 0.016 seconds required in the last case study (Section 4.2.2), the online recognition of this benchmark takes 19 times longer (0.31/0.016 = 19), which is expected as the multimodal sequences to be recognised have more feature vectors and larger dimensions. The results again demonstrate that the DTW recogniser is both fast and accurate for online temporal fusion of multimodal sequences.

4.4 Chapter Summary

The first case study investigated the performance of the DTW temporal fusion agent for temporal fusion with six channels of continuous data acquired from two accelerometers. First, the gesture sequences were segmented from the continuous data streams to evaluate the performance of the DTW agent for *offline temporal DAI-DEO* and *FEI-DEO* fusion processes, and the best result of 96.79% were achieved. Second, we used a second set of gesture sequences to evaluate the performance of the DTW temporal fusion agent for *online temporal FEI-DEO* fusion, and the highest values of AR of 100% were achieved in 44 seconds. Third, the classification rates achieved by the HMM recogniser for offline temporal fusion were compared to those from the DTW recogniser, and the DTW recogniser outperformed the HMMs for both efficiency and accuracy.

The second case study aimed at demonstrating that the system can achieve good results on more complex multimodal sequences, which has 29 dimensions (mix of both discrete and continuous variables) of feature values per second over durations of 1 to 5 mins. In the first offline temporal fusion experiment, the DTW recogniser achieved a much better result of 97.5% than the classification rate of 88.19% achieved by the HMM recogniser. To compare the performance of both systems for online temporal fusion, the HMM recogniser must have a reliable solution for the endpoint detection of multimodal sequences embedded in continuous data streams. Since we did not have a reliable implementation for this endpoint detection, we only compared their results for offline temporal fusion.

The experimental results from both real-world datasets demonstrated that the proposed DTW temporal fusion agent outperforms HMM based systems, and has the capability to perform online temporal fusion efficiently and accurately in real–time. In all experiments, we have only explored the performance of the centralized fusion architecture for one central DTW fusion agent, since there is no real-world dataset available to discover the hierarchical fusion and distributed fusion architectures. This will be one of the suggested future research directions.

Chapter 5 Conclusions

This chapter summarises the work carried out within this thesis and discusses possible future directions for the research. Section 5.1 is devoted to the summary. Section 5.2 suggests some possible future work to extend our multi-agent temporal fusion framework and the DTW temporal fusion agent.

5.1 Summary

Fusion is a fundamental human process that occurs in some form at all levels of sense organs such as visual and sound information received from eyes and ears respectively, to the highest levels of decision making such as in our brain that fuses visual and sound information to make decisions. *Multi-sensor data fusion* is a way of processing, extracting, combining, and integration multi-sensor data and information content about the world and the environment that is relevant to a human and the decisions that must be made. Chapter 2 surveyed the related work in the fields of formal frameworks for multi-sensor data fusion systems and pattern recognition techniques for temporal sequences:

- First, the sections on frameworks presented some popular data fusion frameworks in the literature that are widely used for designing multi-sensor data fusion systems. However, many real world situations change over time. When the multi-sensor system is used for situation awareness, it is useful not only to know the state or event of the situation at a point in time, but also more important to understand the causalities of those states or events changing over time. The reviewed frameworks lack representation of temporal fusion processes.

- Second, the other sections presented the pattern recognition techniques for temporal sequences. Temporal fusion over complex sequences is a challenge. First, the complex sequences acquired are multi–dimensional, multimodal, interacting, time–varying, and are typically a mix of binary, discrete, and continuous variables. Second, the same class of multimodal sequences might vary in length locally or globally. These variations make temporal fusion a difficult task. Moreover, performing temporal fusion efficiently in real–time is another challenge due to the large amounts of multi–sensory information.

Thus, chapter 3 describes our major contributions that address the above-mentioned issues and challenges:

- We proposed a multi-agent framework for temporal fusion, with emphasis on the time dimension of the fusion process, that is, fusion of the multi-sensor data or events derived from multi-sensor data over a period of time. Three major layers: *hardware*, *agents*, and *users* of the proposed fusion framework were described. The core of the proposed temporal fusion framework is the DTW temporal fusion agent. Three fusion architectures: *centralized*, *hierarchical*, and *distributed*, for organising the group of agents were presented, and the temporal fusion process described in detail by using an *information graph*.

- We proposed the DTW temporal fusion agent that includes four major modules: *data pre-processing*, *DTW recogniser*, *class templates*, and *decision making*. The DTW recogniser is extended in various ways to deal with the variability of multimodal sequences acquired from multiple heterogeneous sensors, the problem of unknown start and end points, the problem of multimodal sequences for the same class having different lengths locally or globally, and the challenges of online temporal fusion.

Chapter 4 then presented the experimental results on evaluating the performance of the proposed DTW temporal fusion agent. Two real-world datasets were used in the evaluation:

1. Accelerometer data acquired from performing two hand gestures (Chambers et al., 2004). The gesture dataset contains six channels of continuous data that are acquired from two accelerometers.

2. A benchmark dataset acquired from carrying a mobile device that has six sensors, and performing the pre-defined user scenarios (Mantyjarvi et al., 2004). This dataset is a complex multimodal sequence that has 29 dimensions.

Performance results of the DTW temporal fusion agent were compared with those of a Hidden Markov Model (HMM) based system. The experimental results from both datasets demonstrate that the proposed DTW temporal fusion agent outperforms HMM based systems, and has the capability to perform online temporal fusion efficiently and accurately in real–time.

5.2 Future work

There are following possible future extensions to the thesis work presented here:

1. Bowman has generalized a conclusion that the main functions of a data fusion systems are the encompassing duality between *data fusion* (i.e., estimate) and *resource management* (i.e., control) (Bowman, 1994; Bowman et al., 1999; Llinas et al., 2004; Steinberg & Bowman, 2004). In this thesis, the agent layer of the multi-agent temporal fusion framework consists of a group of the *DTW temporal fusion agents* and *control agents*, which also corresponds to this duality between estimate and control. This thesis has explored the use of the DTW temporal fusion agent for data fusion, but not much on the control agent for resource management. Hence, future work should investigate the use of *sensor selection* (Zhang & Ji, 2005), *sensor management* (Xiong & Svensson, 2002), and *resource management* (Bowman, 2004), in control agents.

2. Extending the proposed multi-agent temporal fusion framework with self-improvement potential through the use of *reinforcement learning* (Sutton & Barto, 1998) between the control agent and the *Tuner* module of the DTW temporal fusion agent as depicted in Figure 3.5. The control agents not only performs sensor and resource management, but also sent the feedback to the *Tuner* module to improve the performance of the DTW temporal fusion agent over time. The performance of the DTW temporal fusion agent is improved by the *Tuner* taking the feedback to fine-tune both the decision making and DTW recogniser modules as needed to reinforce their correct decisions and punishing their wrong decisions so their decision making ability is improved.

3. One of the modules of DTW temporal fusion agent is the *database of class templates*. Successful temporal fusion processing depends on the existence of this prior information (i.e., a database of class templates), without which it is difficult to perform appropriate resource management. Hence, one of the future directions is to apply the data mining

techniques (Chudova & Smyth, 2002) to discover the previously unrecognised temporal sequences (i.e., class templates) in data that are important for the data fusion systems to know to perform appropriate resource management.

4. We have compared the performances of the DTW temporal fusion agent with those of a hidden Markov model (HMM) based system. The Self-organizing Map (SOM) for temporal sequences (refer to section 2.2.3) is another enabling technique for temporal fusion. Future work can compare the performance of the SOMSD based system to our DTW temporal fusion agent.

5. Future work can explore the dynamic time warping recogniser for classifying more complex multimodal sequences such as: *interleaved sequences*, *sequences with gaps*, and those sequences with *missing sub-sequences*.

In addition, there are still challenges in the information fusion domain such as robust estimation and prediction of states of entities, learning of the interrelations among such entities, learning of the interrelations between entities and situations, combining no commensurate information (e.g., image data, text, and signals), and maintaining and manipulating the enormous number of alternative ways of associating and interpreting large numbers of observations of multiple entities (Liggins, Hall, & Llinas, 2009).

References

Abdulla, W. H., Chow, D., & Sin, G. (2003). *Cross-words reference template for DTW based speech recognition systems.* Paper presented at the IEEE TENCON 2003, Bangalore, India.

Aziz, A. M., Tummala, M., & Cristi, R. (1999). Fuzzy logic data correlation approach in multisensor-multitarget tracking systems. *Signal Processing, 76*(2), 195-209.

Bedworth, M., & Brien, J. O. (1999). *The Omnibus model: a new model of data fusion?* Paper presented at the Proceedings of the Second International Conference on Information Fusion, Sunnyvale, CA.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Bellot, D., Boyer, A., & Charpillet, F. (2002). *A new definition of qualified gain in a data fusion process: application to telemedicine.* Paper presented at the Proceedings Fifth International Conference on Information Fusion, Annapolis, Maryland.

Bernardin, K., Ogawara, K., Katsushi Ikeuchi, & Dillmann, R. (2003, January 2003). *A Hidden Markov Model Based Sensor Fusion Approach for Recognizing Continuous Human Grasping Sequences.* Paper presented at the Proc. 3rd IEEE International Conference on Humanoid Robots.

Bowman, C. L. (1994). *The data fusion tree paradigm and its dual.* Paper presented at the Proceedings of the Seventh National Symposium on Sensor Fusion.

Bowman, C. L. (2004, September 20-22). *The dual node network (DNN) data fusion & resource management (DF&RM) architecture.* Paper presented at the AIAA Intelligent Systems Conference, Chicago.

Bowman, C. L., Steinberg, A. N., & White, F. E. (1999). Revisions to the JDL Model. *Sensor Fusion: Architectures, Algorithms, and Applications, Proceedings of the SPIE, 3719.*

Boyd., J. R. (1987). *A Discourse on Winning and Losing.* . Paper presented at the Air University Library, Maxwell AFB, AL, USA.

Bruckner, D., Sallans, B., & Russ, G. (2007, July 23-27, 2007). *Hidden Markov Models for Traffic Observation.* Paper presented at the 5th IEEE Intl. Conference on Industrial Informatics, Vienna, Austria.

Castellanos, J. A., & Tardós, J. D. (1999). *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*. Boston: Kluwer Academic Publishers.

Chambers, G., Venkatesh, S., West, G., & Bui, H. (2004). *Segmentation of intentional gestures for sports video annotation.* Paper presented at the 10th IEEE International Multimedia Modelling Conference, Brisbane, Australia.

Chappell, G., & Taylor, J. (1993). The temporal Kohonen map. *Neural Networks, 4,* 441-445.

Chudova, D., & Smyth, P. (2002). *Pattern Discovery in Sequences under a Markov Assumption.* Paper presented at the SIGKDD 2002, Edmonton, Alberta, Canada.

Crochiere, R. E., Tribolet, J. M., & Rabiner, L. R. (1981). An Improved Endpoint Detector for Isolated Word Recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing, COM-29*(5), 621-659.

Damarla, T. (2008, June 30). *Hidden Markov Model as a Framework for Situational Awareness.* Paper presented at the The 11th international conference on information fusion, Cologne, Germany.

Dasarathy, B. V. (1994). *Decision Fusion*. Washington, DC: IEEE Computer Society Press.

Dasarathy, B. V. (1997). Sensor fusion potential exploitation-innovative architectures andillustrative applications. *Proceedings of the IEEE, 85*(1), 24-38.

Dasarathy, B. V. (1998, 12/16/1998). *Intelligent learning techniques for multi-source information fusion environments.* Paper presented at the 37th IEEE Conference on Decision and Control, Huntsville, AL, USA.

Dasarathy, B. V. (2000, 22 Jan. 2000). *Industrial applications of multi-sensor multi-source information fusion.* Paper presented at the IEEE International Conference on Industrial Technology 2000, Huntsville, AL, USA.

DeVaul, R., Sung, M., Gips, J., & Pentland, A. (2003). *MIThril 2003: applications and architecture.* Paper presented at the 7th IEEE International Symposium on Wearable Computers, New York.

Flanagan, J. A., Himberg, J., & Mantyjarvi, J. (2003). *A Hierarchical Approach to Learning Context and Facilitating User Interaction in Mobile Devices.* Paper presented at the Artificial Intelligence in Mobile System 2003 (AIMS 2003 in conjunction with Ubicomp 2003), Seattle,USA.

Flanagan, J. A., Mantyjarvi, J., & Himberg, J. (2002). *Unsupervised clustering of symbol strings and context recognition.* Paper presented at the Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), Maebashi, Japan.

Frey, C. W. (2008, Aug. 2008). *Process diagnosis and monitoring of field bus based automation systems using self-organizing maps and watershed transformations.* Paper presented at the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems.

Hagenbuchner, M., Sperduti, A., & Tsoi., A. C. (2003). A Self-Organizing Map for Adaptive Processing of Structured Data. *IEEE Transactions on Neural Networks, 14*(3), 491-505.

Hagenbuchner, M., Tsoi, A. C., & Sperduti, A. (Eds.). (2001). *A supervised self-organizing map for structured data*: Springer.

Hall, D. L., & Llinas, J. (1997). An introduction to multisensor data fusion. *Proceedings of the IEEE, 85,* 6-23.

Himberg, J., Flanagan, J. A., & Mantyjarvi, J. (2003). *Towards Context Awareness Using Symbol Clustering Map.* Paper presented at the Proc. Workshop for Self-Organizing Maps 2003 (WSOM2003), Kitakyushu, Japan.

Honkela, T. (1997). *SELF-ORGANIZING MAPS IN NATURAL LANGUAGE PROCESSING.* Thesis. Helsinki University of Technology.

Hossain, M., & Jenkin, M. (2005). *Recognizing hand–raising gestures using HMM.* Paper presented at the Second Canadian Conference on Computer and Robot Vision (CRV 2005).

Itakura, F. (1975). Minimum prediction residual principle applied to speech recognition. *IEEE Trans Acoustics Speech Signal Process ASSP, 23,* 52-72.

Jouseau, E., & Dorizzi, B. (1999). Neural network and fuzzy data fusion: application to an on-line and real-time vehicle detection system. *Pattern Recognition Letters, 20,* 97-107.

Kallio, S., Kela, J., & Mantyjarvi, J. (2003). *Online Gesture Recognition System for Mobile Interaction.* Paper presented at the IEEE International Conference on Systems, Man and Cybernetics, Washington D.C., USA.

Keogh, E. (1997). *A fast and robust method for pattern matching in time series databases.* Paper presented at the Proceedings of 9th International Conference on Tools with Artificial Intelligence (TAI '97), Newport Beach, CA.

Keogh, E., & Ratanamahatana, C. A. (2004). Exact Indexing of Dynamic Time Warping. *Knowledge and Information Systems: An International Journal (KAIS)*, 358-386.

Ko, M. H., West, G., Venkatesh, S., & Kumar, M. (2005a, December, 2005). *Online Context Recognition in Multisensor Systems using Dynamic Time Warping.* Paper presented at the Proceedings of the second International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2005), Melbourne, Australia.

Ko, M. H., West, G., Venkatesh, S., & Kumar, M. (2005b, July, 2005). *Temporal Data Fusion in Multisensor Systems Using Dynamic Time Warping.* Paper presented at the Workshop on Information Fusion and Dissemination in Wireless Sensor Networks (SENSORFUSION 2005) co-located with WICON 2005, Budapest, Hungary.

Ko, M. H., West, G., Venkatesh, S., & Kumar, M. (2008). Using dynamic time warping for online temporal fusion in multisensor systems. *Information Fusion, 9*(3), 370-388.

Kohonen, T. (2000). *Self-Organizating Maps* (2nd ed.): Springer-Verlag.

Koskela, T., Varsta, M., Heikkonen, J., & Kaski, K. (1998). Time series prediction using recurrent SOM with local linear models. *International Journal of Knowledge-based Intelligent Engineering Systens, 2*(1), 60-68.

Krause, A., Siewiorek, D. P., Smailagic, A., & Farringdon, J. (2003). *Unsupervised, dynamic identification of physiological and activity context in wearable computing.* Paper presented at the 7th IEEE International Symposium on Wearable Computers, New York.

Kuhn, M. H., & Tomaschewski, H. H. (1983). Improvements in isolated word recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing, ASSP-31*(1), 157-167.

Lai, H. C., Yang, R., & Ng, G. W. (2007, July 2007). *Enhanced self-organizing map for passive sonar tracking to improve situation awareness.* Paper presented at the 10th International Conference on Information Fusion, Quebec, Que.

Lee, J.-M., Yoo, C., & Lee, I.-B. (2003). On-line batch process monitoring using a consecutively updated multiway principal component analysis model. *Computers & Chemical Engineering, 27*(12), 1903-1912.

Liggins, M., Chong, C.-Y., Kadar, I., Alford, M. G., Vannicola, V., & Thomopoulos, S. (1997). Distributed fusion architectures and algorithms for target tracking. *Proceedings of the IEEE, 85*(1), 95-107.

Liggins, M., Hall, D., & Llinas, J. (2009). *Handbook of Multisensor Data Fusion: Theory and Practice*. New York: CRC Press.

Lin, J., Vlachos, M., Keogh, E., & Gunopulos, D. (2004). *Iterative Incremental Clustering of Time Series.* Paper presented at the proceedings of the IX Conference on Extending Database Technology, Crete, Greece.

Llinas, J., Bowman, C. L., Rogova, G., Steinberg, A. N., Waltz, E., & White, F. (2004). *Revisiting the JDL data fusion model II.* Paper presented at the Seventh International Conference on Information Fusion, Stockholm.

Mantyjarvi, J. (2003). *Sensor-based Context Recognition for Mobile Applications.* Ph.D. Thesis Thesis. Ph.D. Thesis, VTT Publications.

Mantyjarvi, J., Himberg, J., Kangas, P., Tuomela, U., & Huuskonen, P. (2004). *Sensor Signal Data Set for Exploring Context Recognition of Mobile Devices.* Paper presented at the Workshop "Benchmarks and a database for context recognition" of 2nd International conference on Pervasive Computing (PERVASIVE 2004), Linz/Vienna, Austria.

Markin, M., Harris, C., Bernhardt, M., Austin, J., Bedworth, M., Greenway, P., et al. (1997). Technology Foresight on Data Fusion and Data Processing. *The Royal Aeronautical Society*.

Mitchell, H. B. (2007). *Multi-Sensor Data Fusion:An Introduction*. Berlin, Heidelberg: Springer.

Morriss, S. B. (1994). *Automated Manufacturing Systems - Actuators, Controls, Sensors and Robotics*: McGraw-Hill.

Murphy, K. (2001). The Bayes Net Toolbox for Matlab. *Computing Science and Statistics, 33*, 257-286.

Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning.* Thesis. University of California, Berkeley.

Myers, C. S., & Rabiner, L. R. (1981). A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing, ASSP-29*(2), 284-297.

Myers, C. S., Rabiner, L. R., & Rosenberg, A. E. (1980a). *An Investigation of the Use of Dynamic Time Warping for Word Spotting and Connected Speech Recognition.* Paper presented at the IEEE International Conference on Acoustics, Speech, and Signal Processing.

Myers, C. S., Rabiner, L. R., & Rosenberg, A. E. (1980b). Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing, ASSP-28*(6), 623-635.

Opitz, F., Henrich, W., & Kausch, T. (2004). *Data fusion development concepts within complex surveillance systems.* Paper presented at the The 7th International Conference on Information Fusion, Stockholm, Sweden.

Rabiner, L. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of IEEE, 77*(2), 257-286.

Rabiner, L. R. (1985). Some properties of continuous hidden Markov model representations. *AT&T Tech. J., 64*(6), 1251-1270.

Rabiner, L. R., & Myers, C. S. (1981). Connected Digit Recognition Using a Level Building DTW Algorithm. *IEEE Trans. on Acoustics, Speech, and Signal Processing, ASSP-29*(3), 351-363.

Rabiner, L. R., Rosenberg, A. E., & Levinson, S. E. (1978). Considerations in Dynamic Time Warping Algorithms for Discrete Word Recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing, ASSP-26*(4), 575-582.

Rabiner, N., & Juang, L. Y. (1993). *Fundamentals of speech recognition*. Englewood Cliffs, NJ: Prentice.

Ratanamahatana, C. A., & Keogh, E. (2005). *Three Myths about Dynamic Time Warping.* Paper presented at the SIAM 2005 Data Mining Conference, CA.

Ratanamahatana, C. A., & Keogh, E. J. (2004a). *Everything you know about Dynamic Time Warping is Wrong.* Paper presented at the Third Workshop on Mining Temporal and Sequential Data, in conjunction with KDD-2004, Seattle, WA.

Ratanamahatana, C. A., & Keogh, E. J. (2004b). *Making Time-Series Classifiation More Accurate Using Learned Constraints.* Paper presented at the In Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, FL, USA.

Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics,Speech, and Signal Proc., 26*, 43-49.

Schmidt, A. (2002). *Ubiquitous Computing–Computing in Context.* Ph.D. Thesis Thesis. Ph.D. Thesis, Lancaster University.

Shahbazian, E., ve, D. B., & Labbe, P. (2001). *The Extended OODA Model for Data Fusion Systems.* Paper presented at the 4th International Conf. on Information Fusion, Montreal, Kanada

Shepherd, D. (2005). BBC sport academy cricket umpire signals. (http://news.bbc.co.uk/sportacademy/bsp/hi/cricket/rules/umpires_signals/html/).

Steinberg, A. N., & Bowman, C. L. (2004). *Rethinking the JDL data fusion levels.* Paper presented at the Proceedings of MSS National Symposium on Sensor and Data Fusion, Laurel, Maryland.

Steinberg, A. N., Bowman, C. L., & White, F. E. (1999). Revisions to the JDL Model. *Sensor Fusion: Architectures, Algorithms, and Applications, Proceedings of the SPIE, 3719.*

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Varsta, M., Milan, J. d. R., & Heikkonen, J. (1997). *A recurrent self-organizing map for temporal sequence processing.* Paper presented at the In Proc. ICANN'97.

Venkatasubramaniam, V. (2003). A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. *Computers and Chemical Engineering, 27*(15), 293-311.

Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., & Keogh, E. (2003). *Indexing Multi-Dimensional Time-Series with Support for Multiple Distance Measures.* Paper presented at the In Proc. of 9th International Conf. on Knowledge Discovery & Data Mining (SIGKDD), Washington, DC.

Wilpon, J. G., & Rabiner, L. R. (1987). Application of Hidden Markov Models to Automatic Speech Endpoint Detection. *Computer Speech and Language, 2*(3/4), 947-954.

Wilson, A., & Bobick, A. (2000). *Realtime Online Adaptive Gesture Recognition.* Paper presented at the Proceedings of the International Conference on Pattern Recognition, Barcelona, Spain.

Xiong, N., & Svensson, P. (2002). Multi-sensor management for information fusion:issues and approaches. *Information Fusion, 3*, 163-186.

Xue, H., & Govindaraju, V. (2006). Hidden Markov Models Combining Discrete Symbols and Continuous Attributes in Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28*(3), 458-462.

Zhang, Y., & Ji, Q. (2005, July, 2005). *Sensor Selection for Active Information Fusion.* Paper presented at the The Twentieth National Conference on Artificial Intelligence (AAAI-05), Pittsburgh, Pennsylvania.