CogWatch – Cognitive Rehabilitation of Apraxia and Action Disorganisation Syndrome

D3.3.1 Report on predictive models I

Deliverable No.		D3.3.1	
Workpackage No.	**WP3**	Workpackage Title	**Action Recognition and Prediction**
Task No.	**T3.3**	Activity Title	**Action prediction**
Authors (per company, if more than one company provide it together)		**Author (s) Name (s) (Company)** **Martin Russell, Manish Parekh, Emilie Jean-Baptiste, Chris Baber, Hedieh Ekhlasi, and Maryam Najafian (UOB-EECE),** **Marta Bienkiewicz (TUM).**	
Status (F: final; D: draft; RD: revised draft):		**F**	
File Name:		**Cogwatch_D3 3 1_Prediction_I_final.doc**	
Project start date and duration		**01 November 2011, 36 Months**	

EXECUTIVE SUMMARY

The goal of the CogWatch system is to monitor automatically the progress of an individual executing an everyday task, and to intervene with an appropriate cue if an error occurs or if the system believes that an error is imminent. The application is rehabilitation for stroke patients. In order to achieve this goal, the objects involved in the task are instrumented so that their use can be measured. In the first prototype system, which focuses on tea-making, the kettle, milk container and mug are each fitted with a CogWatch instrumented coaster (CIC) containing an accelerometer and three force sensitive resistors (FSRs). This information may be supplemented with information about the participant's hand position, for example, using the Microsoft Kinect system. An automatic activity recognition (AAR) system takes integrated data from these sensors and uses it to recognise component sub-goals of the task, plus additional information such as the time taken to complete the sub-goal. This information is passed to the task model (TM), which uses it to infer the participant's state in the task and to accumulate a cost, based on the time taken to reach this state, and whether the participant's strategy deviates significantly from an optimal strategy. The cost function is used to infer likely failure.

This report describes the approaches to AAR and TM in the first CogWatch prototype. A TM based on a Markov Decision Process (MDP) has been implemented, and is described in Section 2.2. The model has been tested and verified using synthetic AAR data. The notions of optimal strategy and optimal plan, which are used in Section 3 to define cost functions, are introduced. A shortcoming of the MDP-based approach is that it is not well-suited to dealing with ambiguity. Ambiguity arises because the AAR is imperfect and makes classification errors. A potential solution is described where the MDP is replaced by a Partially Observable MDP. The basic theory of POMDPs is presented in Section 2.3.

Section 3 is concerned with failure prediction, and in particular the definitions of the online cost functions which will be used to detect problems in the implementation of a task.

The rationale for choosing parallel, HMM-based sub-goal detectors for activity recognition in CogWatch prototype 1 (originally given in deliverable D3.1), is reviewed in Section 4. Section 5 presents the results of a preliminary evaluation of sub-goal detection. The problem of detecting the "pour milk into jug" sub-goal is investigated, using only the outputs from the CIC attached to the base of the jug. The system uses a multiple state "left-right" HMM to capture the sequential structure of the "pour" activity, and single state HMMs with a multiple component GMM state for the "toy" and "rest" activities.

The experiments investigate approaches to pre-processing the CIC sensor data. Simple thresholding is applied to the outputs of the FSRs, to overcome variability in the FSR data between different instances of an FSR-augmented object at rest. The effects of smoothing and differentiation of the FSR outputs are also investigated, and a number of additional types of pre-processing are suggested. So far, the recognition accuracy achieved is approximately 70%. However there are good reasons to believe that this is an underestimate of the potential performance. For example, the "toying" data is unconstrained and contains actions very similar to pouring, which would be unlikely in real data where the jug which is being "toyed" with contains milk. Also, the classifier uses only information from the CIC attached to the jug, whereas in practice the data from the CIC attached to the 'receiving' mug would also be used.

Finally, the implementation of a real-time AAR is discussed in Section 6.

Contents

Table of Figures

Table of tables

REVISION HISTORY

Revision no.	Date of Issue	Author(s)	Brief Description of Change
0	24/10/12	Martin Russell Emilie Jean-Baptiste	First draft
1	30/10/12	Martin Russell Emilie Jean-Baptiste	Add section on cost functions. Update author list
2	2/11/12	Martin Russell	Add table of tables. Revised Executive Summary
final	9/11/12	Alan Wing, Christos Giachritsis	Text figure referencing updated; minor typographical corrections; text font size equalised; contents updated.

LIST OF ABBREVIATIONS AND DEFINITIONS

Abbreviation	Abbreviation
AAR	Automatic activity recognition/recogniser
CIC	CogWatch instrumented coaster
FSR	Force sensitive resistor
GMM	Gaussian mixture model
HMM	Hidden Markov model
MDP	Markov decision process
POMDP	Partially Observable MDP
SDP	Spoken Dialogue Processing
TM	Task Model

1. INTRODUCTION

1.1 Purpose

This deliverable describes the current status of action prediction algorithms in the CogWatch project, leading up to Prototype 1.

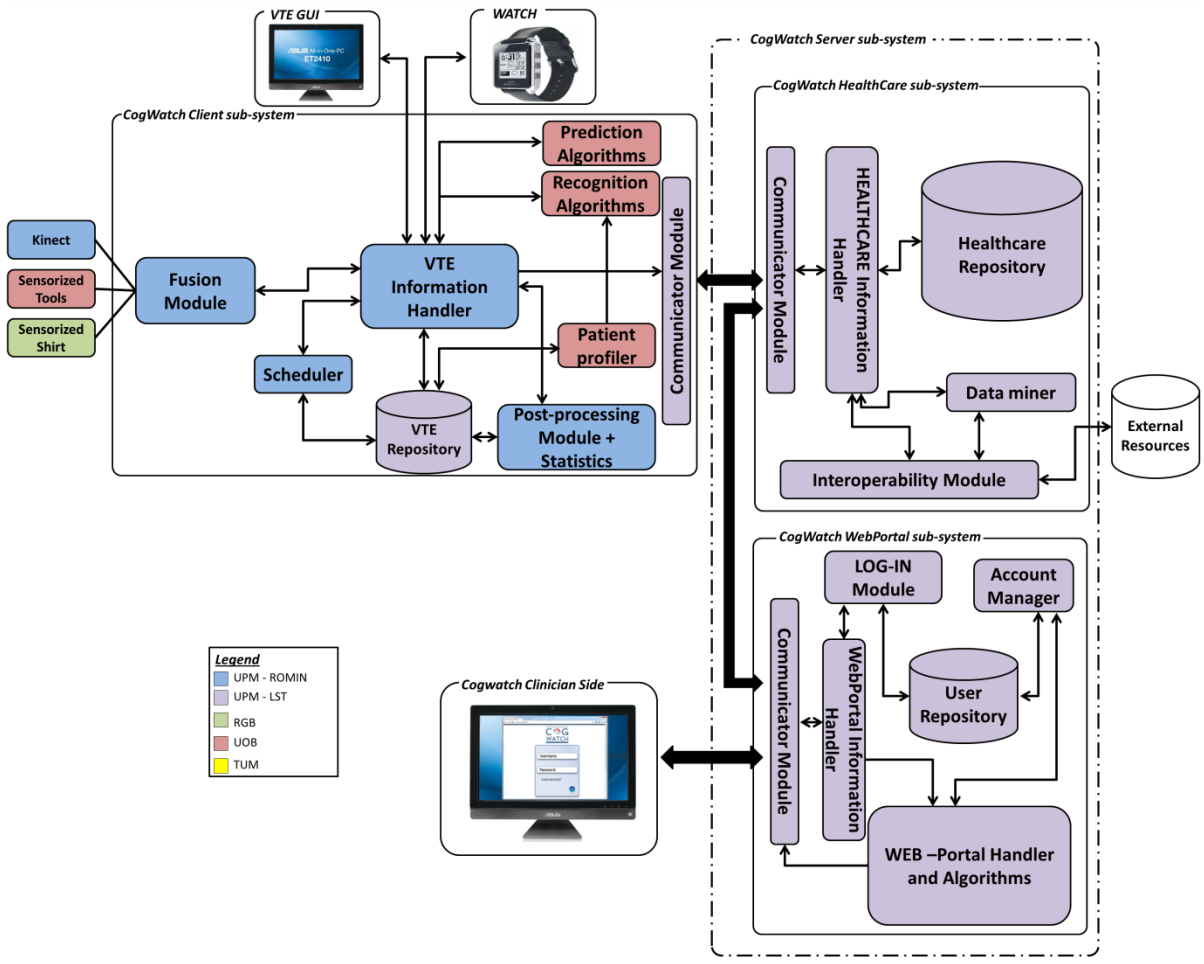

Figure 1: General architecture of the CogWatch system (taken from deliverable D2.2.1)

1.2 Action prediction

'Action Prediction' is a slight misnomer. The purpose of the techniques described in this deliverable is to take the data from the sensors attached to the task objects which are being used by a participant engaged in the task, to interpret this data in terms of the sub-goals that the participant is trying to achieve, and to detect when the sequence of sub-goals that is being executed by the participant is unlikely to

result in successful task completion. The term 'action prediction' refers to the ability of the system to compare the participant's actions with those predicted by the system, in order to predict or detect errors, or to increment a Cost Function. In the current phase of the project the task is tea making, the objects are the kettle, milk jug, mug and other items involved in tea-making, and the sensors are the accelerometers and force sensitive resistors in the CogWatch instrumented coaster (CIC) attached to the objects.

In terms of Figure 1, the sensor outputs are delivered via the Fusion Module to the VTE Information Handler, from where they are accessed by the Automatic Activity Recognition (AAR) algorithm. The outputs from the AAR algorithm, namely sub-goals of the tea-making task together with timing information, are passed back to the VTE Information Handler, from where they are accessed, in turn, by the Prediction Algorithm/Task Model (TM). The outputs of the TM are data structures which indicate that the task has been completed or, in the event of an error, the type of error that has occurred and the identity of the cue that should be presented to the participant. The TM includes methods for Failure Prediction, based on costs incurred for excessive sub-goal completion times, choice of non-optimal strategies at different stages of the task, or deviation from a task execution plan defined by, for example, a clinician

The report is organized as follows:

Section 2 (Action Prediction) is concerned with the CogWatch Prototype 1 Task Model. The current implementation of the TM as a Markov Decision Process (MDP) is described in section 2.2.1. The notions of a cost function (2.2.1.2), an optimal strategy (2.2.1.3) and an optimal plan are defined. A problem with this type of TM is that it is not well suited to dealing with uncertainty in its inputs, due in this case to classification errors in the AAR. More precisely, the information passed to the TM is the output a of the AAR due to the sub-goal A performed by the participant. The identity of the true sub-goal A is not available to the TM. Therefore an extension of MDPs, known as Partially Observable MPDs (POMDPs) is introduced. The theory and proposed implementation of POMDPs is presented in section 2.3.

Section 3 is concerned with Failure Prediction. This refers to the use of cost functions and the accumulation of cost, to anticipate user errors.

Section 4 (Recognition Algorithms) reviews the approach to sub-goal detection based on Hidden Markov Models (HMMs) that was presented in deliverable D.3.2.1.

Section 5 presents the results of preliminary experiments on sub-goal detection using HMMs applied to the output of the CIC. The specific problem that is addressed is detection of the "pour milk into mug" sub-goal. Initial recognition accuracies of up to 70% are achieved.

Section 6 describes the status of the implementation of the AAR.

Section 7 presents our conclusions.

2. ACTION PREDICTION

2.1 Action prediction in the first CogWatch prototype

Action prediction in the first CogWatch prototype is the responsibility of the Task Model (TM). The inputs to the TM are sub-goal labels, plus sub-goal execution time information, that are output from the automatic action recognition (AAR) system.

This information is passed to the TM via the VTE. The sub-goals correspond to the second level of the tea making task tree from D1.1 "Report on scenarios". The outputs from the TM are "cue prompts" that are passed to the VTE Information Handler in the event that the participant makes an error or an error is anticipated by failure prediction, or an indication that the task has been completed successfully.

Options for the TM and the AAR systems were discussed in detail in D3.1 "Report on action recognition techniques" and the rationale was given for the choices of the TM and AAR system for the first prototype.

2.2 The Task Model

As explained in D3.1, the TM in CogWatch Prototype 1 is based on a Markov Decision Process (MDP). This is motivated by the use of MDPs in spoken dialogue processing (SDP, see, for example, [3]).

2.2.1 Markov decision processes (MDPs)

2.2.1.1 Formal definition

Formally, a MDP comprises:

- A finite set $\boldsymbol{S}$ of N states. In SDP these are often referred to as belief states,
- A finite set $\boldsymbol{A}$ of actions,
- For each pair of states $\boldsymbol{s_1}$ and $\boldsymbol{s_2}$ in $\boldsymbol{S}$ and action $\boldsymbol{a}$,
  - $P_{\boldsymbol{a}}(\boldsymbol{s_1},\boldsymbol{s_2})$ is the probability of being in state $\boldsymbol{s2}$ at time t+1 given state $\boldsymbol{s1}$ at time t and that action $\boldsymbol{a}$ was taken
  - $R_{\boldsymbol{a}}(\boldsymbol{s_1},\boldsymbol{s_2})$ is the corresponding reward/cost

2.2.1.2 The Cost Function

The cost function $R_a(s_1,s_2)$ provides a mechanism through which human judgements about the importance of different types of behaviour can be incorporated into the MDP. In this way the cost function becomes the means whereby the MDP can be transformed into a more psychologically plausible model. The details of the cost function are yet to be decided, however it is envisaged that it will include:

- A penalty based on the time taken to complete the sub-goal.

- A penalty for non-fatal deviations the optimal strategy (see 2.2.1.3 below). This is a penalty which is incurred each time the subject executes a sub-goal in a particular state, which is different from the optimal strategy at that state.
- A penalty for repeating a sub-goal (where repetition is not a fatal error). For example if the subject executes a sub-goal "add milk", then executes one or more other sub-goals, and then executes a second "add milk", this might incur a penalty even though it is legal.

Cost functions are discussed in the context of Failure Prediction in Section 3.

2.2.1.3 The Optimal Strategy

In the context of MDPs, a *strategy* is a function $\pi: \boldsymbol{S} \rightarrow \boldsymbol{A}$. In other words, for each state s, $\pi(s)$ is an action (sub-goal).

Given a strategy π a function V_π can be defined on the state space $\boldsymbol{S}$ such that for any state s, $V_\pi(s)$ is the minimum accumulated cost of completing the task given that the participant is currently in state s and follows the strategy π.

The *optimal strategy*, denoted by π^*, is the strategy π for which the cost function V_{π^*} is minimised.

Example algorithms for computing the optimal strategy are described in [3]. Our current implementation of the MDP-based TM uses the Monte Carlo Algorithm with exploring starts described in [3]. An important consideration is that once the cost function has been defined the optimal strategy can be computed before the MDP is used and the optimal action at each state s, $\pi^*(s)$, can be stored as part of the state s.

The cost functions that are used to compute the optimal strategy are based on training material and on human intuition. The objective is that the cost function should have the property that large costs are indicative of task failure, so that errors can be anticipated and cues can be provided in a timely manner.

2.2.1.4 The Optimal Plan

The optimal plan is a strategy defined by a human expert, for example a clinician that prescribes the order in which the sub-goals should be executed.

The optimal plan could be entered by hand. Alternatively the clinician could demonstrate the task to the participant in the presence of the CogWatch system. In this case the AAR and TM would run in a special mode, where the sequence of states and sub-goals would be recognised and recorded. Once the clinician had verified that the interpretation of the task was correct, the participant would be required to adhere to the optimal plan.

2.2.2 Interpretation and operation of the MDP

In the tea-making task, the states of the MDP correspond to sequences of sub-goals that can be extended to a complete, successful instantiation of the tea making task. Associated with each state is a set of potential state transitions, corresponding to sub-goals which are valid extensions of that state. For example, in the case of the extended tea-making task (tea with milk and sugar) if the current state corresponds to ["fill kettle", "boil water", "add water to cup"] then potential transitions would be to the states corresponding to:

- ["fill kettle", "boil water", "add water to cup", "add teabag to cup"]
- ["fill kettle", "boil water", "add water to cup", "add sugar to cup"]
- ["fill kettle", "boil water", "add water to cup", "add milk to cup"]
- ["fill kettle", "boil water", "add water to cup", "add water to cup"]

The final transition is possible because it has been decided that it is only if the patient repeats "add water to cup" more than twice that an error is detected and a cue is raised.

Suppose that the system is in state $\boldsymbol{s_1}$ and a sub-goal $\boldsymbol{a}$ is output from the AAR and passed to the TM by the VTE. The TM looks to see if $\boldsymbol{a}$ is a valid extension of s_1. If this is the case, and if s_2 denotes the state obtained by extending s_1 with the action $\boldsymbol{a}$, then $P_a(s_1,s_2) = 1$, the transition is taken and the system moves into the corresponding new state s_2.

If the sub-goal that is output from the AAR is not a valid extension of the current state, or the cost function exceeds a given value (for example, because of the delay in completing the next sub-goal, see section 3) then the faulty current state is passed to the TM's error recognizer which identifies the type of error made, and the sequence of cue prompts that should be passed to the VTE Information Handler. The data structure of the cue is a vector containing the ID of the cue that should be sent to the subject, its priority (Fatal, non Fatal…or Level_1, Level_2…), its number (CA#1, CA#2…), and the next legal action predicted by the algorithm, if applicable.

The error types, the corresponding cues, the time constraint within the latter, and the impact of the use of specific participant's inputs during particular moment during the task, have been agreed with psychologists in the project.

2.2.3 Implementation

An MDP-based TM has been implemented in Python and C#. The TM has been tested successfully using outputs from a simulated AAR.

2.3 Coping with uncertainty – Partially Observable MDPs

A major problem with an MDP-based TM is that it is not well equipped to accommodate errors in the output of the AAR system. If the system is in state s_1

then in a conventional MDP the probability $P(s_2 \mid s_1, A)$ of moving to state s_2 depends on s_1 and the sub-goal A that has been executed. In practice, A is unknown and the TM must be satisfied with a, the output the AAR system when sub-goal A was performed by the participant. In this sense the sub-goal is only *partially observable*, since the true sub-goal can only be inferred, and not known, from the recognised sub-goal. This requires an extension of a MDP called a Partially Observable MDP or POMDP.

2.3.1 POMDPs

Given a sub-goal a output from the AAR, we need to compute $P(s_2|s_1,a)$. This can be achieved by factorization according to A, the true sub-goal executed by the participant:

$$P(s_2 \mid s_1, a) = \sum_A P(s_2, A \mid s_1, a) = \sum_A P(s_2 \mid A, s_1, a) P(A \mid a, s_1)$$
$$\approx \sum_A P(s_2 \mid s_1, A) P(A \mid a, s_1) \qquad (1)$$

The final approximation assumes that $P(s_2|s_1, A, a) = P(s_2|s_1, A)$, in other words if the true sub-goal A is known then the output a of the AAR does not exert any additional influence on the probabilities of state transitions. The probability $P(s_2|s_1,A)$ is known, and the probability $P(A|a,s_1)$ can be computed from Bayes' rule:

$$P(A \mid a, s_1) = \frac{P(a \mid A, s_1) P(A \mid s_1)}{P(a \mid s_1)}$$

The probabilities on the right-hand side of the equation can be computed as follows:

- If it is assumed that the output a of the AAR depends only on the input A to the AAR and not on the current state, then $P(a|A,s_1) = P(a|A)$. This probability can be obtained from the AAR confusion matrix, which can be determined empirically from activity classification experiments.
- The probability $P(A|s_1)$ is the probability that the participant executes a particular sub-goal when in a particular state. In other words, given that the participant has already completed the sequence of sub-goals corresponding to state s_1, what is the probability that the participant's next sub-goal is A. This will be estimated from the clinical trials that are currently being conducted.
- Finally, $P(a \mid s_1) = \sum_A P(a \mid A, s_1) P(A \mid s_1)$.

Note that all of these probabilities can be pre-computed.

2.3.2 POMDP computations

In the MDP-based TM, the state of the TM is unique and known after each input from the AAR. However, this is not the case for the POMDP-based TM. Instead the state of the TM after the n^{th} input from the AAR is a set of probabilities $\{P_n(s): s \in S\}$ over the states of the underlying MDP.

Suppose that a sub-goal a is the n^{th} output from the AAR. For each state s_2, the probability $P_n(s_2)$ is updated according to:

$$P_n(s_2) = \sum_{s_1} P_{n-1}(s_1) P(s_2 \mid s_1, a),$$

where $P(s_2|s_1,a)$ is calculated from equation (1). In this way the state of the TM (i.e. the accumulated probabilities for each of the states of the underlying MDP) is updated after each input from the AAR.

2.3.3 Errors and task completion in the POMDP-based TM

In the case of the MDP-based TM, errors are detected when the cost function exceeds a particular value or the current input from the AAR is not a valid extension of the current state. For a POMDP-based TM, the case where there is no valid extension of the current state after the n^{th} input corresponds to $P_n(s)$ being small (perhaps below some threshold $\varepsilon > 0$) for all states s.

Similarly, for a MDP-based TM the task is complete when the model reaches the final 'end' state. For a POMDP-based TM, task completion will occur when the probability of the final 'end' state is sufficiently high compared to that of other states.

Experiments will need to be conducted using simulated data with known levels of AAR system error to determine how sensitive the POMDP is to AAR error.

2.3.4 Confidence

For a POMDP-based TM it is also possible to define notions of confidence. For example, if after the n^{th} AAR input there is a state s for which $P_n(s)$ is much greater than $P_n(r)$, for all states $r \neq s$, then we can be confident that the system is in state s, but if $P_n(s)$ is similar to $P_n(r)$ it is not possible to be confident that the system is in state s.

Intuitively, confidence will depend on the accuracy of the AAR. If the AAR is very inaccurate then after each sub-goal input to the TM the probability will be distributed between different states. In contrast, if the AAR error rate is small then these probabilities will focus on particular states. In the limit, if the AAR error rate is zero then there is no longer any ambiguity between the true sub-goal A and the recognised sub-goal a, and the POMDP becomes an MDP

2.3.5 Implementation and testing

A POMDP-based TM will be implemented and tested in the next phase of the project.

Initial testing will be done using a simulated AAR, driven by a sub-goal confusion matrix corresponding to a known AAR error rate. In this way it will be possible to measure how uncertainty in the AAR output corresponds to uncertainty in the TM, and to determine how much AAR error the POMDP-based TM can accommodate.

2.1 Summary of section 2

This section has described an approach to task modelling based on Markov Decision Processes (MDPs), which with be used in the first CogWatch prototype. A MDP-based TM has been implemented and verified.

A limitation of the MDP-based approach is that it is not well-suited to coping with ambiguity in its inputs. In this application ambiguity arises as a consequence of classification errors in the AAR. Partially Observable MDPs are proposed as a solution to this problem. The theory and practice of POMDP-based TM is discussed.

3. FAILURE PREDICTION

3.1 Failure prediction in future CogWatch prototypes

Another requirement of the Task Model is failure prediction. For the purpose of the project, we would like the TM to be able to detect when the participant is unlikely to achieve the goal successfully. Human faults give rise to errors that can be manifested in two kinds of ways: well known “factual errors” that are implemented in the TM to be systematically cued when they happen (e.g., perseveration error, anticipation error, omission error, etc.); and faults that will appear as deviations from an expected behaviour during the trial.

Currently, the TM is an “after-the-fact” system, which means that a factual error has to be made in order for the TM to react and provide specific cues. But as written in [4]:

> “With this kind of reactive (or “after-the-fact”) fault-recovery, the impact of the fault is not necessarily averted. This approach also fails to take advantage of any pre-failure indicators or symptoms that might be present in the system. While analyzing these indicators might not avert the fault, the results of this analysis might have allowed recovery to be initiated faster or proactively, thereby mitigating the impact of the fault on the application/system.”

Thus, in our case, the failure prediction’s aim will be to collect the information in advance on any participant’s abnormal behaviour that could lead to a system fatal failure, so that the failure can be predicted.

3.2 Approaches

3.2.1 Methods based on machine learning

Measures to detect anomalous system behaviour can be implemented using different machine learning or pattern processing techniques such as: Support Vector Machine (SVM) [5], Similar Events Predictions (SEP) [6], or genetic-based machine learning systems [7].

All of these methods evaluate the system’s current state in a different way, but share the same ability to learn from previous experience in order to predict future events. This ability is based on the fact that all those techniques use supervised machine-learning methods, which implies the necessity to use a training set of labelled data in order to build the model they rely on. For example, in [4], the model is trained with labelled commercial telecommunication data. Then, during the online test, the system is able to base its evaluation of the testing data on the special patterns leading to failures that it learned during the offline training.

Training data are currently not available in the CogWatch Project, consequently none of those methods can be applied for the moment. Nevertheless, when a

sufficient amount of data has been collected, it will be interesting to find how those methods can be used in the context that we are working in.

Fulp et al. [5] and Salfner et al. ([6], [8]) used system log files as input data in their system. These log files contain messages about the system's state, and the analysis of the latter permitted them to predict future failures. In our case, the input data will include the actions made by the participant, and the time taken to achieve those actions. Characteristic patterns in time-labelled sequences of actions may make it possible to predict participants' failures. Weiss wrote in [7] that it is often important to be able to predict future behaviour based on past data. The ability of the TM to keep the history of the participants' past actions should be exploited to predict a potential future failure.

For one of those methods to be used in a future prototype, its ability to predict failures online in real-time will need to be tested, as well as its reliability.

These issues will be raised again and developed later as training data becomes available.

3.2.2 Knowledge-driven failure prediction

In the absence of sufficient data, the first prototype TM will base its failure prediction on parameters supplied using expert knowledge. While the performance of the AAR can be measured relatively easily, the values of parameters such as cost functions and their implications for error prediction will need informed estimation. For example, we could base our approach on the hypothesis that a fault manifests as increasingly unstable behaviour before escalating into a failure [6]. This hypothesis could be added to an adaptation of the assumption Salfner used in his approach in [8]:

> "The fundamental assumption in my approach is, that the occurrence of failures can be predicted by identifying special patterns of errors the system is experiencing. This assumption is based on the fact, that dependencies among the components of systems exist."

In the case of the CogWatch prototype, cost could be incurred by deviation from the normal time taken to perform an action, or deviation from the optimal strategy. Failure would be predicted when these deviations go above specific levels, defined in consultation with psychologists.

Thus, without the training data that would permit us to test failure prediction based on machine-learning techniques, the idea is to use a network of cost functions that will allow monitoring of symptoms that could lead to future participants' failures, and alert or cue them before the failures actually happen. The interpretation of this cost will have to be discussed with psychologists: the decision to consider an abnormal behaviour as an error that should be cued will be linked to exceeding psychologically plausible thresholds.

3.3 Online Cost Function Network

As explained in 2.2.1.3 the TM already computes a cost function offline, and the Monte Carlo Algorithm uses this to find the optimal strategy (action), $\pi^*(s)$, that the participant should execute at state s in order to complete the task.

In online failure prediction, a network of cost functions will be used; taking into account all of the inputs that the TM will have access to online during the trials. At the same time, the online cost function clearly needs to be closely related to the offline function for consistency.

3.3.1 Definition

In [9], Li wrote:

> "The construction of cost functions should rely on one's knowledge of the system and common sense. There is no absolute right or wrong for cost functions. One can only say if it is reasonable or not."

The online cost functions are functions that return the "price" of the participant's actual decisions and actions during the task, compared to what she or he is expected to do.

Denote the TM input space (i.e., the data that the TM has access to during the task), by I. In the current implementation I consists of pairs $i=(a,d)$ where a is the sub-goal output by the AAR and d is its duration. Then the online cost function network could be defined as an association of three partial cost functions, related to the time taken to achieve an action $\zeta(i)$, the participant's choice of action $\rho(i)$ compared with the action suggested by the optimal strategy, or the ability of the participant to follow a given plan p, $\gamma(i)$, plus an additional cost that refers to the ability of the participant to respect hidden and optimal rules $\delta(r)$. In other words:

- $\zeta : I \to [0, MC_{time}], \zeta(i) = \zeta(a,t) = c$, depends on the time t taken to execute sub-goal a
- $\rho : I \to [0, MC_{optStrategy}], \rho(i) = \rho(a,t) = c$, depends on whether the sub-goal a output by the AAR matches the optimal strategy π^*,
- $\gamma : I \to [0, MC_{optPlan}], \gamma(i) = \gamma(a,t) = c$, depends on whether the sub-goal a output by the AAR matches the optimal plan p, specified by the participant or a clinician, and
- $\delta : I \to [0, MC_{optRule}], \delta(i) = \delta(a,t) = c$,

where MC_{time}, $MC_{optStrategy}$, $MC_{optPlan}$ and $MC_{optRule}$ denote the maximum values taken by ζ, ρ, γ and δ respectively. Before explaining the partial cost functions and the associated variables, let us consider that we also have a Global Cost Function

(GCF) Γ taking all these parameters into account. The latter could be defined as followed:

- $\Gamma: I \to R^+, \Gamma(i) = \Gamma(\zeta(i), \rho(i), \gamma(i), \delta(i))$. In other words $\Gamma(i)$ is a combination of the previous four cost functions.

3.3.2 Explanations

3.3.2.1 $\zeta(i)$: Ability to perform the task in a short amount of time

The output $i = (a,t)$ of the AAR includes the recognised sub-goal a, and t - the time the participant takes to achieve that sub-goal. Shared online with the TM, this data will permit analysis of the participant's ability to complete a task in a reasonable amount of time. The 'local' cost associated with sub-goal duration t is $\zeta(i)$, where $i=(a,t)$.

The accumulated 'duration cost' up to state s, denoted by $\bar{\zeta}(s)$, can then be calculated by $\bar{\zeta}(s) = \bar{\zeta}(s') + \zeta(i)$, where s' is the previous state $i = (a,t)$ is the output from the AAR at state s', and s is obtained from s' by addition of the sub-goal a.

The decision about whether or not to indicate an error and output cue prompt will depend on Max_{dur} and Min_{dur}, the maximum time allowed for the task and minimum time to complete the task, respectively. These are defined in consultation with psychologists.

3.3.2.2 $\rho(i)$: Ability to follow the optimal strategy

Suppose that $i=(a,t)$ is the output of the AAR while the system is in state s. During the task, the TM will have access to this observation and to the optimal strategy $\pi^*(s)$ computed offline by the Monte Carlo Algorithm. Thus the TM will be able to compare a and $\pi^*(s)$ and modify the GCF Γ and the online local cost function $\rho(i)$ accordingly. The cost incurred for executing a sub-goal a when the optimal strategy is $\pi^*(s) \neq a$ will be defined in consultation with psychologists.

The more the patient deviates from the optimal strategy during the task, even if the action taken doesn't raise any error, the more this accumulated value of this online partial cost function will tend to return a high partial cost value c. Let the accumulated cost function due to deviation from the optimal strategy at state s be denoted by $\bar{\rho}(s)$, then by analogy with 3.3.2.1:

$$\bar{\rho}(s) = \bar{\rho}(s') + \rho(i).$$

The decision about whether or not to indicate an error and output a cue prompt will depend on $T_{\pi Dev}$ the maximum number of deviations from the optimal strategy that is allowed. This parameter will be set in consultation with psychologists.

3.3.2.3 $\gamma(i)$: Ability to follow an imposed plan

This partial cost function relates to Patient Profiling and corresponds to a special case of deviating from the optimal strategy. The 'Patient Profiler' will permit the clinician to record the plan (i.e., sequence of actions) that he or she would like the patient to follow (this is referred to as the 'optimal pan').

Intuitively, deviation from the optimal plan p should incur cost. By analogy with 3.3.2.2, the cost $\gamma(i)$ incurred for executing a sub-goal a in state s when the optimal plan is $p(s) \neq a$ will be defined in consultation with psychologists.

The accumulated cost due to deviation from the optimal plan at state s is denoted by $\bar{\gamma}(s)$, and defined by $\bar{\gamma}(s) = \bar{\gamma}(s') + \gamma(i)$.

The decision about whether or not to indicate an error and output a cue prompt will depend on T_{pDev} the maximum number of deviations from the optimal path that is allowed. This parameter will be set in consultation with psychologists.

3.3.2.4 $\delta(r)$: Ability to respect hidden rules

Hidden rules are all of the rules that the clinicians or the psychologists would like the TM to implement, and which will have a negative effect on the GCF and the online partial cost function if the participant does not respect them. These rules are called hidden because the participant will not necessarily be aware of their existence.

For example, a hidden rule may be linked to the inability of the patient to focus on one specific sub-task before beginning another one. In practice, if one imagines that the participant is performing the low level actions related to a specific subtask (e.g., the patient is manipulating the kettle in order to heat the water), but the participant then stops before the completion of the highest level action (i.e., heat water) in order to begin another sub-task (e.g., add sugar), $\delta(i)$ and the GCF Γ may return a higher partial cost than they would have done if the participant had finished the manipulation of the kettle before manipulate the spoon to add sugar.

At present this type of rule breaking will not be detected because the output of the AAR for the first CogWatch prototype is at the sub-goal level. However, in future versions of the system which include models of lower-level features, this type of cost function will become increasingly relevant.

3.3.2.5 Γ: The Global Cost Function

While all the other partial cost functions are directly linked to parameters such as time, deviation from the optimal strategy, deviation from the optimal plan, or the non-respect of hidden rules, and return partial cost exclusively associated to them; the GCF would return a cost representing how "globally" the participant deviates from a

common or optimal scheme. A cue could be raised if this cost goes above a specific threshold T_{global}.

To do so, the cost of the GCF could be a simple weighted mean, such as:

$$\Gamma(s) = \frac{w_1\bar{\zeta}(s) + w_2\bar{\rho}(s) + w_3\bar{\gamma}(s) + w_4\bar{\delta}(s)}{w_1 + w_2 + w_3 + w_4}$$

Here, w_i for $i = 1,2,3,4$ are the weights chosen by the designer in order to give a specific importance to each parameter.

Other non-linear combinations of the individual costs could also be considered.

3.3.2.6 The cost function

Rather than simply comparing the various accumulated cost values with thresholds, a cost function could be computed based on the TM's interpretation of $\bar{\zeta}$, $\bar{\rho}$, $\bar{\gamma}$ and $\bar{\delta}$, which are related to the participant's efficiency during the trial. Ideally, the cost function should be psychologically plausible, which means that it should return a low cost value to sequences of actions that a control user (or a clinician) would consider to be a successful achievement of the goal, and a high one to sequences which are unlikely to result in successful goal completion.

3.3.3 Implementation and testing

The utility of online cost functions based on no training data, and using handcrafted thresholds for failure prediction is uncertain. However, as training data becomes available it will be possible to tune and adapt these online cost functions to the statistics learnt with the data.

3.4 Summary of section 3

This section has discussed approaches to online failure prediction using the information that will be available in a MDP or POMDP-based TM. Failure prediction is based on whether the values of various cost functions exceed pre-defined thresholds. The proposed cost functions measure the time taken by the participant to complete a task, the number of times that the participant's actions deviate from the optimal strategy, the number of times that the participant's actions deviate from a prescribed plan (if one has been defined) and the number of times that the participant breaks one of a set of 'hidden' rules.

As training data become available from the trials that are being conducted with controls and patients, it is hoped that it will be possible to use these measures as the basis of robust and psychologically plausible approaches to failure prediction.

4. RECOGNITION ALGORITHMS

4.1 Automatic activity recognition (AAR)

Options for the AAR system were discussed in detail in D3.1 “Report on action recognition techniques” and the rationale was given for the choices of the AAR system for the first prototype. The AAR system is described in detail here and also summarised in deliverable D3.2.1 “Report on predictive models 1”.

The inputs to the AAR system are the outputs of the Fusion Module (**Figure 1**). The Fusion Module synchronises the outputs from the instrumented devices (for example, the CogWatch instrumented coasters (CICs)) attached to the objects involved in the task, and synchronises this with the outputs of other sensors, such as hand-position data detected by Kinect. It collates all of this data into a single feature vector, at a typical frame rate of 200 Hz.

The outputs from the AAR system are sub-goals corresponding to the second level of the tea making task tree from D1.1 “Report on scenarios”. These are passed to the TM via the VTE.

4.2 Hidden Markov Model (HMM) based sub-goal detection

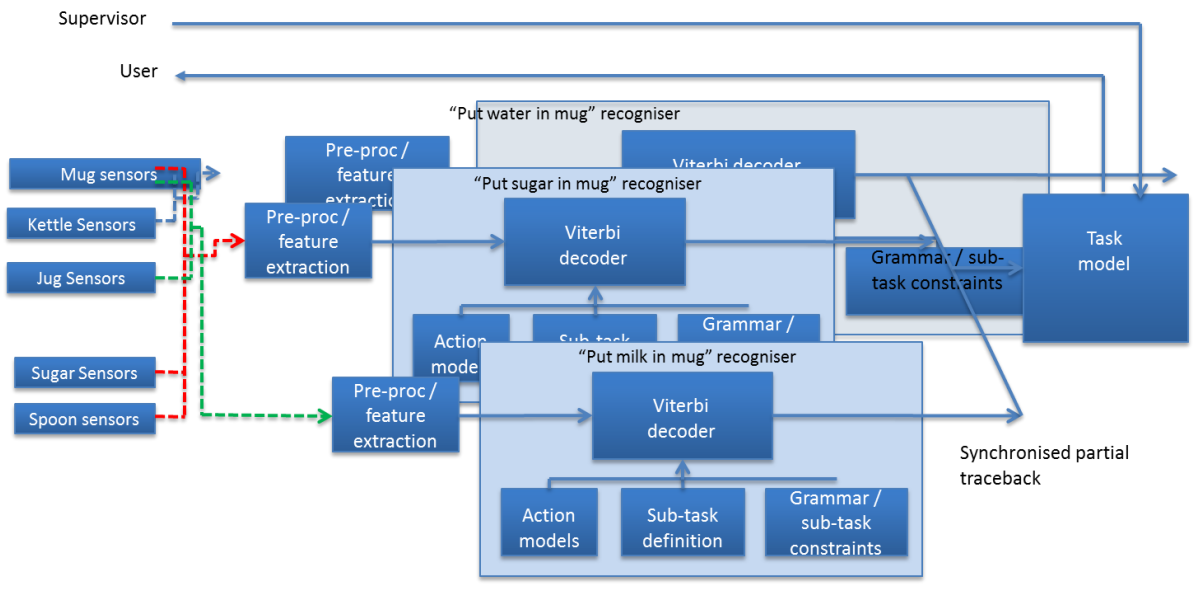

Figure 2: Schematic diagram of HMM-based sub-goal detection (taken from CogWatch deliverable D3.1).

The arguments for using an HMM-based activity detection system in the first CogWatch prototype were presented in D.3.1 “Report on action recognition techniques”.

A schematic diagram of the sub-goal detector is shown in **Figure 2**. The figure shows a bank of detectors, each responsible for detecting a particular sub-goal.

Prior to the start of the project it was intended that the first prototype would exploit an existing real-time HMM-based speech recognition system. However, there are fundamental differences between speech recognition and activity recognition. For example, in the former it can be assumed that the signal corresponds to a well-ordered sequence of words, but this is not necessarily the case in activity recognition, where sub-goals can overlap or co-occur. In a simple speech recognition system, all valid word sequences are compiled into a single integrated grammar network. Recognition is then the process of finding the route through this network that achieves the most probable match with the acoustic data.

The need to accommodate the partially-ordered structure of sub-goals, is the reason why the CogWatch AAR is being implemented as a set of parallel, HMM-based sub-goal detectors (this was described in D3.1). The input to a sub-goal detector is the sub-vector from the Fusion Module corresponding to sensors associated with the objects that are relevant to the sub-goal. For example, the system responsible for detecting the “pour milk into mug” sub-goal requires sensor data from the instrumented coasters attached to the milk-jug and the mug, plus the Kinect hand-coordinate data. The detector includes the relevant sub-goal model (in this case a HMM of the “pour milk into mug” sub-goal) and a “background” model to accommodate any activity involving the jug and mug which is not the “pouring” sub-goal (for example ‘toying’ with the jug, the jug at rest). The sub-goal detector continuously compares the input sequence with the “sub-goal” and “background” models and the sub-goal is detected when its model has a higher probability than the background model.

4.3 Summary of section 4

The purpose of this section is to remind the reader of the ‘parallel bank of sub-goal detectors’ architecture chosen for activity recognition. The rationale for the choice of this architecture is given in D3.1.

5. PRELIMINARY EXPERIMENTS ON SUB-GOAL DETECTION

This section reports the results of preliminary experiments to assess the performance of HMM-based sub-goal detection. The experiment used data collected using a prototype CogWatch instrumented coaster (CIC) attached to the base of a jug. This is an electronic coaster (or drink mat) which contains a 3 axis accelerometer, three force sensitive resistors (FSRs), a microcontroller and a Bluetooth module. Full details of the CIC are included in CogWatch report D2.1. The CIC communicates with a host computer via Bluetooth, where its outputs are sampled at 200Hz and stored in a file. A CIC data file comprises a sequence of six dimensional feature vectors (x, y and z accelerometer values and the outputs of the three FSRs).

5.1 Definition of the data set

Naive subjects were asked to perform four types of activity with the instrumented jug:

- "pour" – subjects were asked to pour liquid from the jug into a mug. They were told that the jug should be at rest on the desk surface at the start and end of the activity.
- "toy" - subjects were asked to move the jug arbitrarily, whilst avoiding the "pour" activity. Again they were told that the jug should be at rest on the desk surface at the start and end of the activity.
- "pour-toy" - subjects were asked to pour liquid from the jug into a mug and then to toy with the jug. They were told that the jug should be at rest on the desk surface at the start and end of the activity. This data was used only for testing the detector.
- "rest" – subjects were asked to leave the jug at rest on the desk top.

A total of 96 files were recorded. These were partitioned into a training set of 63 files and a test set of 33 files. The numbers of files representing each activity are indicated in table 1.

	"pour"	"toy"	"pour–toy"	"rest"	Total
Test set	7	8	7	11	33
Training set	20	20	-	23	63
Total	27	28	7	34	**96**

Table 1: The number of files corresponding to each activity in the training and test sets.

For each data file a label file was created indicating the sequence of actions represented by the data. For example, an instance of the "pour" activity would be associated with the following label sequence:

rest
pour
rest

No timing information was included in the label file.

5.2 Visualisation of the data

In our current implementation, the CIC produces a 6 dimensional feature vector

$o = (o_1, .. , o_6)$

200 times per second. The coordinates of the feature vector are as follows:

- o_1 (acc 1), o_2 (acc 2) and o_3 (acc 3) are the acceleration in the x, y and z directions, respectively. The (x,y) plane is parallel with the plane of the coaster and the z axis is orthogonal to the plane of the poster,
- y_4 (FSR 1), y_5 (FSR 2) and y_6 (FSR 3) are the outputs of the three FSRs.

5.2.1 Example of CIC outputs for the different activities

Figure 3 shows CIC data for an example of the "pour" activity lasting approximately 9.75s.

The times at which the jug is raised and put back on the surface are evident from the FSR graphs in the figure. Before the lift, the three FSRs have different values, due, presumably, to the characteristics of the individual FSRs and the precise orientation of the mug. However, after the lift all of the FSRs give a value of approximately 4100 until the jug is put down again after approximately 1200 samples. The FSRs then take between 200 and 800 samples to return, approximately, to their original values.

The accelerometer values are more difficult to interpret and appear to be noisier than the FSR values when the jug is at rest. The third accelerometer reading (acc 3) corresponds to the vertical axis (assuming that the cup is upright), therefore it is measuring gravity as 1g accelerating upwards. When the mug is tipped most of the structure in the accelerometer graphs is the gravity component moving across the axes. The gravity component across the three axes always adds up to 1g but it is difficult to distinguish between the effects of gravity and more general motion when the item is rotated

Figure 4 shows CIC data for an example of the "toy" activity lasting approximately 11.05s. The plots for the FSR values are similar to those for "pour", although the

values for FSR 1 and FSR 2 do not return to their original values after the jug is put down on the surface.

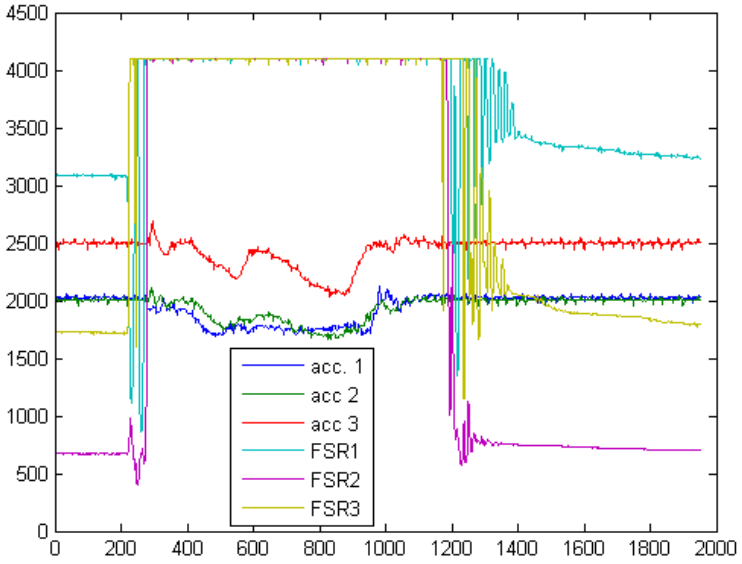

Figure 3: CIC outputs for an example of the "pour" activity lasting approximately 9.75s.

.

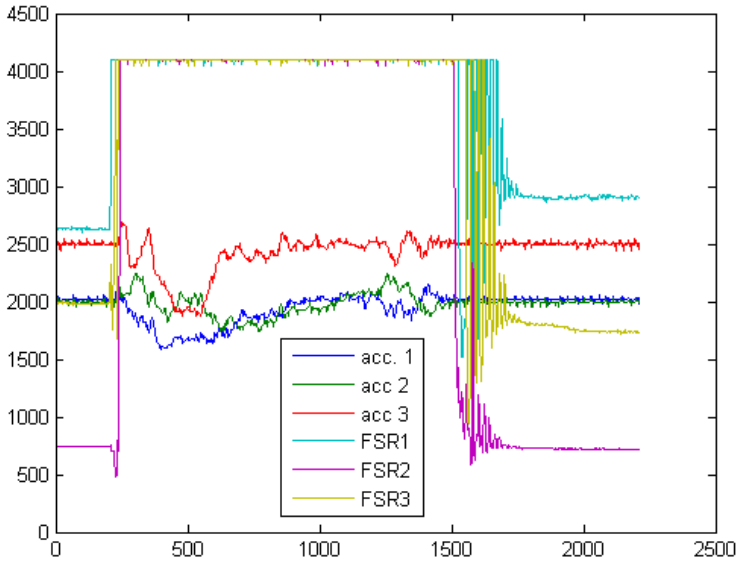

Figure 4: CIC outputs for an example of the "toy" activity (approximate duration 11.05s).

Figure 5 shows CIC data for an example of the “rest” activity lasting approximately 5.45s. The six outputs are as one would expect, given the discussion of the figures for “pour” and “toy”. Each output is a noisy, stationary signal, with the accelerometer values exhibiting a higher level of noise than the FSRs.

Figure 6 shows CIC data for an example of the “pour-toy” activity lasting approximately 18.45s. The first part of the figure, corresponding to the “pour” activities, shows some similarities with **Figure 3**, though the event between 600 and 1000 samples, which presumably corresponds to the actual pour, is much clearer in **Figure 6**.

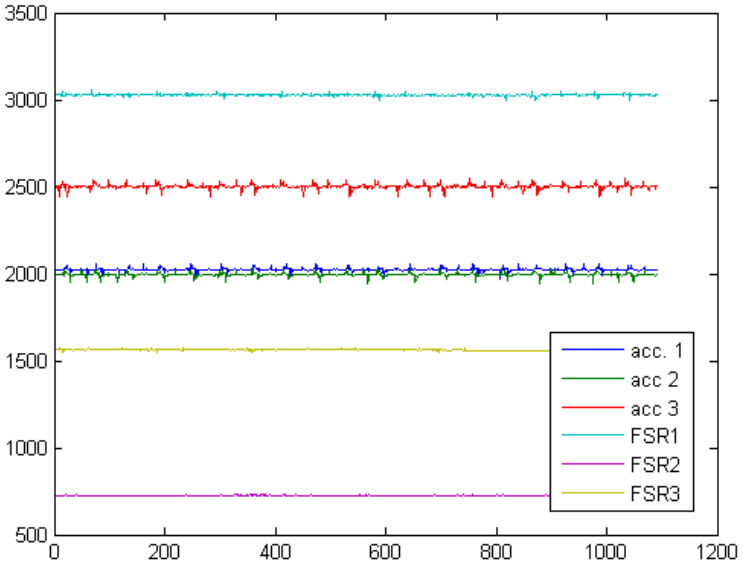

Figure 5: CIC outputs for an example of the "rest" activity (approximate duration 5.45s).

5.2.2 HMM structure

A HMM is specified by:

- The number of states, N.
- For each state n, a state output probability density function (PDF) b_n such that for any feature vector o, $b_n(o) = p(o \mid n)$ is the probability (density) of the vector o given state n, ($n = 1,..., N$). The type of PDF is restricted to those which are compatible with automatic parameter estimation algorithms. For the current experiments it will be assumed that each b_n is a Gaussian Mixture Model (GMM). In other words,

$b_n(o) = \sum_{i=1}^{M} w_i b_{n,i}(o)$, where $b_{n,i}$ is a multivariate Gaussian PDF and $\sum_{i=1}^{M} w_i = 1,\ 0 \le w_i \le 1$

- An N dimensional vector π such that $\pi(n) = P(\text{state } n \text{ at time } t{=}0)$. π is called the initial state probability vector
- An N x N matrix A such that $a_{ij} = P(\text{state } m \text{ at time } t \mid \text{state } n \text{ at time } t\text{-}1)$. A is called the state transition probability matrix.

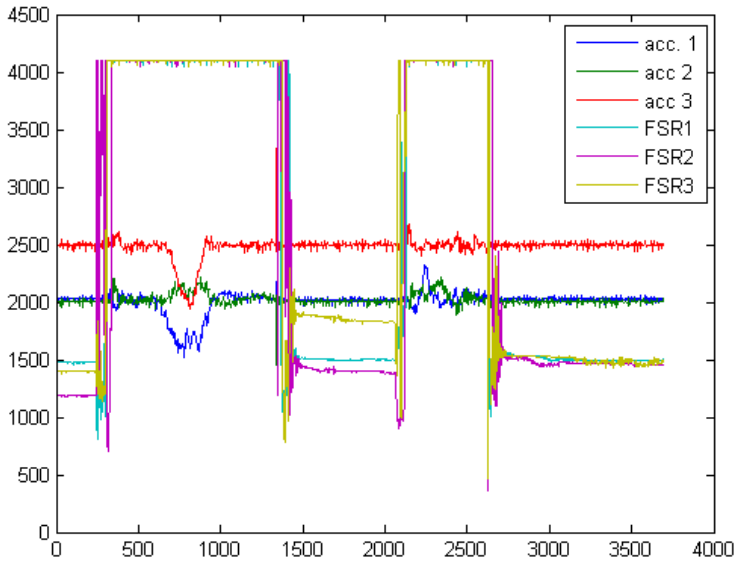

Figure 6: CIC outputs for an example of the "pour-toy" activity (approximate duration 18.45s).

5.3 HMM system specification

5.3.1 HMM parameter estimation

The experiments described here were conducted using the Cambridge University Engineering Department's HMM Toolkit, HTK [1]. This is has become the *de facto* standard for experiments in HMM based automatic speech recognition.

Basic parameters, such as the number of states, the number of components in the GMM state output PDFS (M), and the structure of the initial state probability vector and state transition probability matrix, needed to be chosen manually. Once these have been set, the remaining parameters are estimated from data.

5.3.1.1 Initial parameter choices

A HMM models a signal as if it were piecewise constant, where the constant segments correspond to the states.

The “pour” activity was modelled as a 9 state left-right HMM (Figure 7). This was chosen to try to capture the sequential structure of the “pour” action. Intuitively, the ‘segments’ of the “pour” activity are (1) the jug at rest, (2) the jug is raised, (3) the jug is moved, (4) the jug is tipped, (5) the jug is moved, (6) the jug is lowered, and (7) the jug is at rest. Two additional states were included to obtain a better piecewise constant approximation to the complex tipping movement.

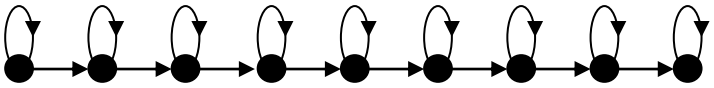

Figure 7: "left-right" structure of the "pour" HMM

As indicated in the figure, only transitions from a state to itself or the next state were permitted, to reinforce the sequential structure of the activity. Initially, each state of the “pour” HMM was associated with a single Gaussian state output PDF ($M=1$).

The “toy” and “rest” actions were modelled as single state HMMs with multiple-component GMM states output PDFs. It was envisaged that different GMM components would capture different positions of the object during toying.

5.3.1.2 Model optimisation

5.3.1.2.1 Parameter initialisation

Initially, all states are associated with single Gaussian state output PDFs. Each of the training files was then segmented uniformly into N segments, where N is the number of states in the corresponding model (thus $N=9$ for “pour” and $N=1$ for “toy” and “rest”). The (vector) means and variances of each of these segments were chosen as the initial mean and variance of the multivariate Gaussian PDFs associated with the corresponding state. This process is implemented using the HInit tool in HTK [1].

5.3.1.2.2 Parameter optimisation

These initial HMM parameters were then refined using the HTK implementation of embedded training with the Baum-Welch algorithm (HERest). Ten iterations of Baum-Welch training were applied.

5.3.1.2.3 Increasing the number of GMM components

Each GMM component PDF was split into two PDFs along its direction of maximum variation using the HTK HHed tool, and a further two iterations of Baum-Welch training were applied. This resulted in:

- Two 9 state “pour” models, one with Gaussian state output PDFs, and one with two-component GMM state output PDFs
- Seven single state “toy” models and seven single state “rest” models, with 1, 2, 4, 8, 16, 32 and 64 component GMM state output PDFs.

5.3.2 Activity detection

5.3.2.1 The recognition grammar

The ‘recognition grammar’ is represented in Figure 8. Intuitively, this grammar sets up a competition between the “pour”, “toy” and “rest” models in terms of which one of them can provide the best explanation of the data up to any point. During recognition, each of the solid square boxes is replaced by the corresponding HMM. Given a sequence of test vectors o corresponding to a particular movement of the jug, the recognition algorithm, known as the Viterbi decoder [1] finds the sequence of states s such that the joint probability of o and s is maximised. From this state sequence the optimal explanation of the data as a sequence of “pour”, “toy” and “rest” activities can be recovered.

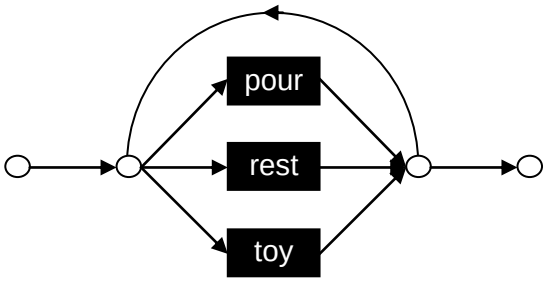

Figure 8: Recognition network for “pour” detection

5.4 Experimental results

5.4.1 Activity recognition using the ‘raw’ CIC data

Figure 9 shows activity recognition results, for the “pour”, “toy” and “rest” activities, using the raw CIC output. The graphs show “% Correct” and “% Accuracy” on the training and test data. These are standard performance measure in automatic speech recognition, defined by:

$$\%\mathrm{Corr.} = \frac{N-S-D}{N}\times 100,\ \ \%\mathrm{Acc} = \frac{N-S-D-I}{N}\times 100,$$

where N is the number of test tokens, S is the number of substitution errors, D is the number of deletion errors, and I is the number of insertion errors. These are calculated using dynamic programming to compute an optimal alignment between

the output of the recognizer and the correct transcription of the data. They can be calculated using the HTK tool HResults [1]. Clearly the % correct figure is always greater than or equal to % accuracy.

Figure 9 shows typical results for this type of experiment. On the training data performance improves as the complexity of the model is increased by increasing the number of GMM components. By contrast, on the test data, performance peaks for GMMs with 32 components. In the case of the system with 64 component GMMs the model is capturing detail in the training data that does not generalize to the test data, so that performance on the test data is compromised. This is a limitation of the training set size.

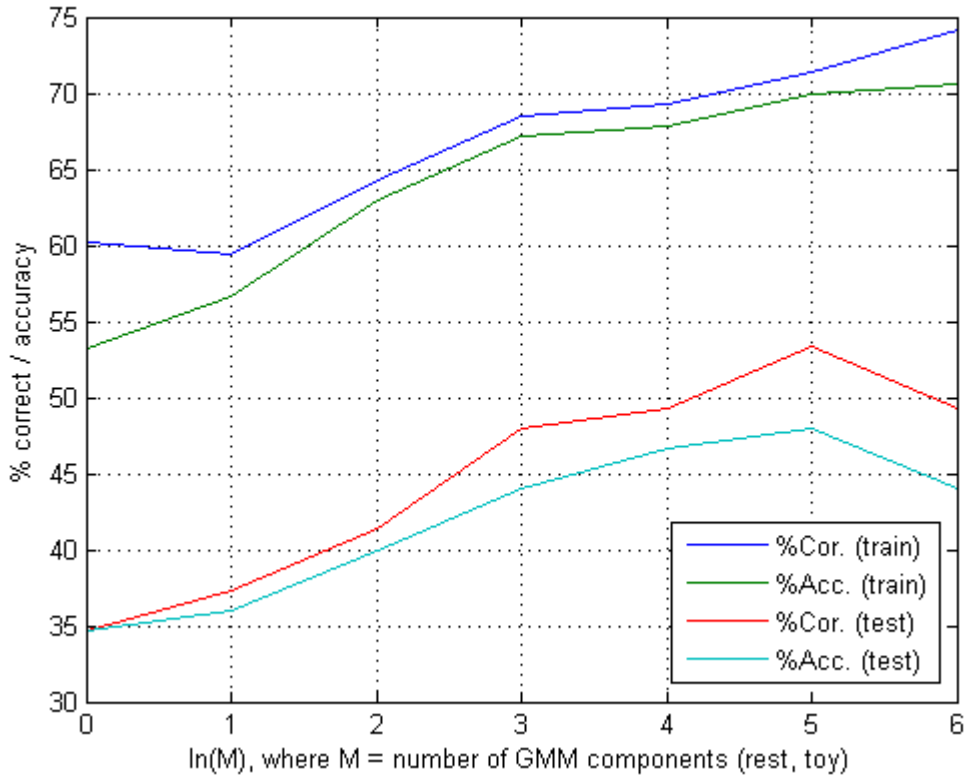

Figure 9: Activity recognition results using the raw CIC data.

The best test set performance is 48% accuracy. A closer inspection of the results shows that “rest” is consistently misrecognised as “toy”. This is due to the variations in the FSR values between different placements of the jug on the surface. There are also frequent confusions between “pour” and “toy”.

The HTK parameters ‘word-insertion penalty’ and ‘minimum variance’ were set empirically to -350 and 0.1, respectively.

5.4.2 Pre-processing of the FSR data

The most significant problem with the CIC data is the inconsistency between FSR values for different instances of the object at rest. By contrast, it has already been seen that the FSR values are consistent when the object has been lifted off the surface (in this case all three FSRs give values close to 4096). Despite their inconsistency, the FSR values for an object at rest are all significantly less than 4000. Therefore a simple solution is to threshold the FSR values. Define:

$$D(FSR) = \begin{cases} 1 \text{ if } FSR \geq 4000 \\ 0 \text{ otherwise} \end{cases}.$$

Figure 10 shows activity recognition performance for pre-processed CIV data comprising the raw accelerometer data and thresholded FSR data. Focusing on % accuracy on the test set, the best result, 64%, is again obtained with a system based on 32 component GMMs. The improvement relative to the system using the raw CIC data is due mainly to correct recognition of "rest". The remaining confusions are between "toy" and "pour".

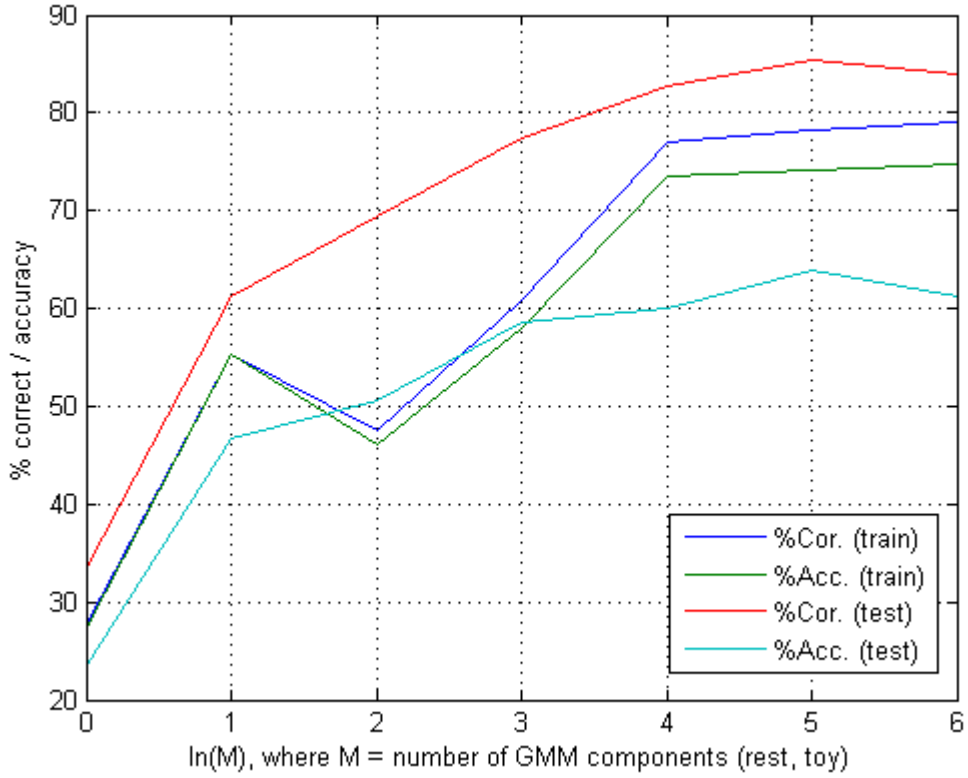

Figure 10: Activity recognition results using the raw accelerometer data and thresholded FSR data.

5.4.3 An alternative model optimization scheme

Section 5.3.1.2.3 describes an incremental approach to increasing the number of GMM components in the "toy" and "rest" model, where the GMM with 2^M components is obtained by splitting the components of the GMM with 2^{M-1} components and then applying Baum-Welch parameter training. An alternative is to create a GMM with 2^M components directly from repeated splitting of the corresponding 2 component GMM, and then to apply Baum-Welch parameter estimation. This will be referred to as "one step" GMM expansion. Intuitively one might expect this approach to lead to problems with local optima. However, for the current task, the second approach gives better results.

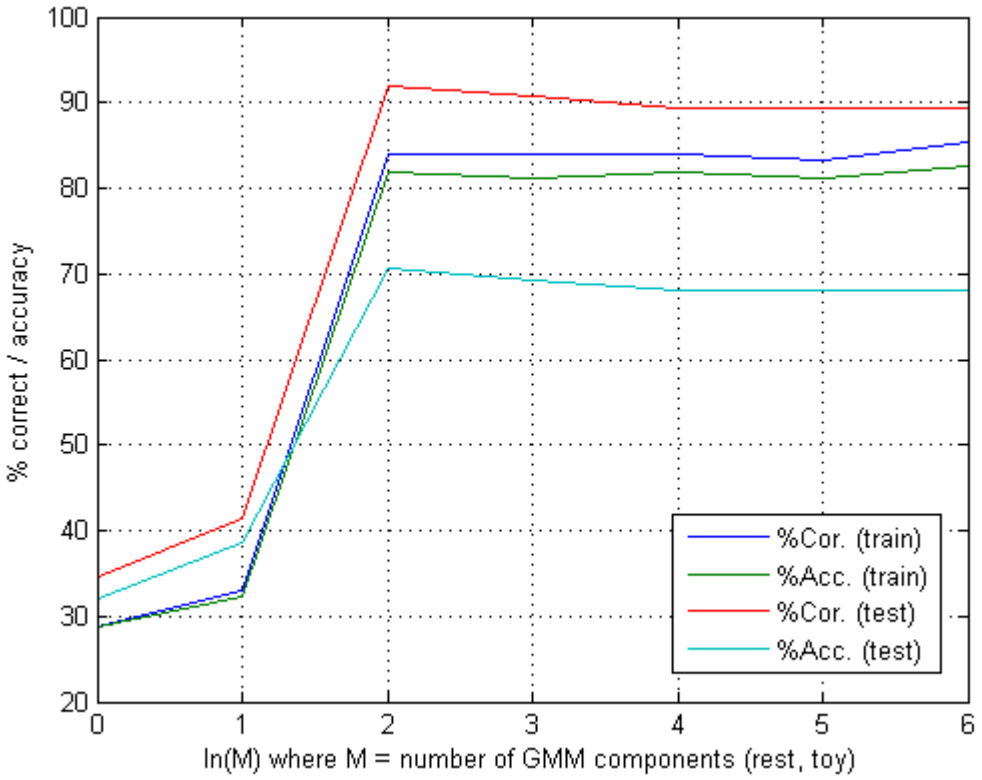

Figure 11: Activity recognition results using the raw accelerometer data and thresholded FSR data with M component GMMs derived directly from 2 component GMMs

Figure 11 shows performance on the training and test sets as a function of the number of GMM components in the "toy" and "rest" models, for one-step GMM expansion. The test set recognition accuracies of around 70% for 4 and 8 component GMMs are the best results that have been achieved to date.

5.4.4 "Delta" FSR parameters

The simple thresholding method described above is a simple and effective solution to the problem of variability in the FSR outputs when the jug is resting on a surface. However it has limited utility.

The results of CIC calibration presented in D.2.3.1 show that the FSRs in the CIC can detect changes in weight of as little as 5 grams. This is less than the weight of a typical portion of milk that is added to tea. This suggests that more robust "pour" detection could be achieved by considering not just the outputs of the CIC attached to the jug, but also the outputs of the CIC attached to the 'receiving' mug. However, without the thresholding method described in 5.4.2 the effects of these weight changes will be masked by the variability in the FSR outputs for an object at rest, but applying thresholding will remove any differences in the FSR outputs due to such a weight change.

A potential solution is to use the time derivatives of the FSR outputs as additional features. In principle these derivatives should be equal to zero when an instrumented object is at rest, but should register the weight change when, for example, milk is added to a jug.

In automatic speech recognition, the time derivatives of features are estimated using "delta" parameters. For a time varying feature y_t the corresponding delta feature Δy_t is defined by:

$$\Delta y_t = \frac{\sum_{w=1}^{W} w\left(y_{t+w} - y_{t-w}\right)}{\sum_{w=1}^{W} w^2}$$

where the derivative is estimated over the interval $y_{t-W}, \ldots, y_{t+W}$ and W is referred to as the analysis window size. Increasing the value of W will compensate for noise in the FSR parameters and lead to smoother delta FSR parameters. It is evident from, for example, **Figure 11**, that the FSR outputs are noisy.

Figure 12 shows activity recognition results for features comprising the three raw accelerometer outputs, the three thresholded FSR outputs, and three FSR derivatives (computed from the raw FSR data), for values of W between 2 and 30. The figure shows the results obtained using "toy" and "rest" models with 8, 16, 32 and 64 GMM components, plus the average result. The figure indicates that increasing W from 2 to 8 results in decreases in recognition accuracy, but that increasing W beyond 8 leads to performance improvements, leveling out for values of W over 20. The value $W=20$ corresponds to estimating derivatives over a 0.2s window.

Figure 13 shows the sequence of augmented CIC vectors for an example of the "pour" activity. The augmented feature set comprises the 3 raw accelerometer outputs, 3 thresholded FSR outputs (these have been scaled and offset so that they

are visible in the figure), and the three delta-FSR parameters. The latter were computed with W=20. The delta-FSR parameters are approximately zero when the jug is either on the surface or raised above the surface, but non-zero values are clearly visible when the jug is lifter or placed back on the surface.

The highest accuracy achieved using the FSR derivative is 69.33%, which is slightly poorer than the best performance achieved without these parameters. However, in the current test set there are no cases where one would expect delta FSR parameters to give an advantage over the simple thresholded FSR parameters. Further experiments, in which, for example, the jug CIC data is augmented with CIC data from the mug receiving milk during a "pour" activity, are needed to properly judge the utility of the delta FSR parameters.

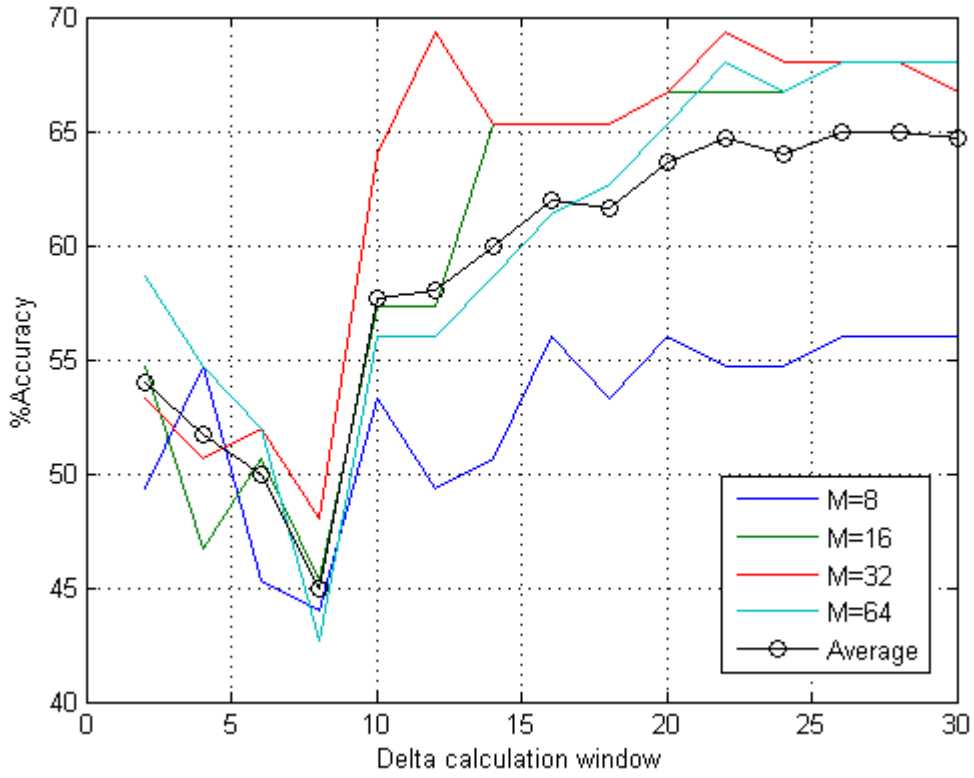

Figure 12: Activity recognition results using the raw accelerometer data, thresholded FSR data and derivatives calculated from the original FSR data over different analysis windows. M component GMMs derived directly from 2 component GMMs.

5.4.5 Further pre-processing of the CIC parameters

There are a number of opportunities for pre-processing the CIC parameters which we expect to result in improvements in performance.

5.4.5.1 Filtering of the FSR data

Figure 12 shows the effect on recognition performance of the size W of the analysis window over which delta-FSR parameters are computed. Increasing W is equivalent to smoothing the FSR parameters during delta-FSR calculation. An alternative approach would be to low-pass filter the FSR signals and then calculate delta-FSR parameters with a smaller analysis window. We are currently conducting experiments to identify suitable filters.

Filtering of the accelerometer outputs should also be advantageous.

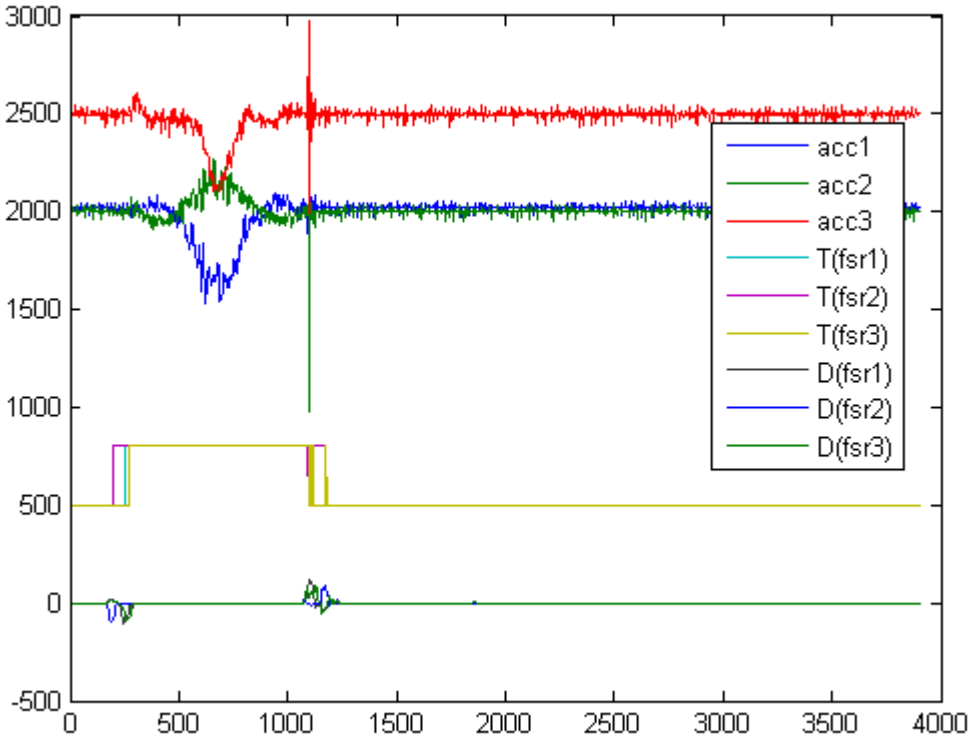

Figure 13: Raw accelerometer data, thresholded FSR data (scaled for visualisation) and FSR derivative data for an example of the "pour" activity. Derivatives calculated over 20 samples.

5.4.5.2 Integration of the accelerometer values

In order to know whether zero-valued accelerometer data corresponds to the CIC moving at non-zero constant velocity or being stationary, it is necessary to know about past accelerometer output values. This 'history' is not available during HMM processing because of the Markov property (this is particularly significant in the case of the single stage "toy" and "rest" models). Therefore, intuitively it would be appear to be advantageous to integrate the accelerometer data to recover velocity.

Synchronized data will be collected from a CIC and from a marker-based 3D motion tracking system for an instrumented mug in motion. Experiments will then be conducted to measure the extent to which the mug's trajectory can be reconstructed from the CIC accelerometer parameter. These are similar to previous experiments in which accelerometer data was used to augment a marker-based 3D human body motion tracking system in order to maintain trajectories during occlusions [2].

The effect of velocity and position parameters recovered from the accelerometer data, on recognition accuracy will be tested.

5.4.5.3 Further pre-processing of the accelerometer data

A further source of variability in the data output by the accelerometers in the CIC is the direction of travel. If the initial positions of the objects involved in tea-making are known, and can be maintained throughout a tea-making session (for example, by tracking using the accelerometer data, as described in 5.4.5.2) then the direction of travel of an object is useful information and needs to be incorporated explicitly in the recognition process. However, if these locations are not known then the direction of travel of an object is effectively noise. In this case it might be beneficial to combine the different velocity parameters recovered from the accelerometer data. For example, the x and y velocity parameters could be combined to give the speed of horizontal motion.

5.5 Summary of Section 5

This section has presented the results of preliminary experiments in automatic activity recognition using the outputs of a single CIC attached to a jug. So far, the best performance achieved is a recognition accuracy of approximately 70%.

Additional types of pre-processing of the CIC parameters have been discussed which may lead to improved performance.

Further experiments are needed, for example to measure the benefits of attaching several CICs to different objects which interact in an activity (for example, the milk jug and the 'receiving' mug).

6. IMPLEMENTATION OF THE AAR

6.1 Features of the CogWatch AAR

The AAR is being implemented in C#. It has the following features:

- It uses the standard format for HMMs from Cambridge University Engineering Department's HMM Toolkit (HTK) [1]. This means that models developed and evaluated offline using HTK can be immediately ported to the CogWatch AAR.
- It uses standard HTK format recognition networks (these are normally referred to as "wordnets" in HTK [1]). A separate network is applied to each sub-goal detector. Typically this network sets up a 'competition' between the relevant sub-goal model, a model of 'toying' with the objects involved in the sub-goal, and a 'rest' model corresponding to the objects at rest.
- Recognition is performed using a standard, continuous time-synchronous Viterbi decoding algorithm. A minor difference from standard speech recognition is that rather than searching a single integrated network, the CogWatch decoder searches a set of separate sub-goal networks in parallel.
- Beam search is applied to reduce computational load. In Viterbi decoding, at each time t the system computes a vector of probabilities, where the indices of the vector components correspond to the states of the model, and the value in component i corresponds to the joint probability of the data up to time t and the 'optimal' partial state sequence ending in state i at time t. The beam search algorithm 'prunes out' paths whose probabilities are less than the maximum probability at time t by more than a given margin, referred to as the 'beam width'. Consequently, at time $t+1$, states that only connect to 'pruned out' states at time t need not be considered.
- In order to be able to recover the best explanation of the data, the Viterbi decoder must maintain the histories of the best state sequences up to particular states. Because the recogniser runs continuously, with no 'end', a technique called 'Partial Traceback' is used to free the memory used to store these histories. At regular intervals the separate histories are traced-back through time to see if they converge at some point in the past (for example, all of the alternative explanations may agree that all objects were at rest at a particular point in the past). Once such a point of convergence has been found, the best explanation of the data up to that point can be output and the memory used to store alternative histories up to that point can be freed. If no point of convergence is found, the best guess of the optimal explanation of the data is output before memory is exhausted.

7. CONCLUSIONS

7.1 Action prediction

This report has described the approach to automatic activity recognition (AAR) and task modelling (TM) in the first CogWatch prototype. When combined, these two processes are referred to as Action Prediction.

7.2 The Task Model (TM)

A TM based on a Markov Decision Process (MDP) has been implemented, and this was described in Section 2.2. This model has been tested and verified using synthetic participant data. However, a shortcoming of the MDP-based approach is that it is not well-suited to dealing with ambiguity. In the CogWatch system, ambiguity arises because the AAR is imperfect and makes classification errors.

A potential solution to this problem is described, in which the MDP is replaced by a Partially Observable MDP. The basic theory of the type of POMDP that we propose to use is presented in Section 2.3.

7.3 Failure prediction

Approaches to online failure prediction that use the information that is available in a MDP or POMDP-based TM were described in Section 3. The proposed method is based on cost functions that measure (i) the time taken by the participant to complete a task, (ii) the number of times that the participant's actions deviate from the optimal strategy, (iv) the number of times that the participant's actions deviate from a prescribed plan (if one has been defined) and (v) the number of times that the participant breaks one of a set of 'hidden' rules. As representative data becomes available it is hoped that it will be possible to use these measures as the basis of robust and psychologically plausible approaches to failure prediction.

7.4 Recognition algorithms

The rationale for choosing parallel, HMM-based sub-goal detectors for activity recognition in CogWatch prototype 1, which was first presented in deliverable D3.1, is reviewed for completeness in Section 4.

7.5 Recognition algorithm performance

Section 5 present results of a preliminary evaluation of HMM-based sub-goal detection. Specifically the problem of detecting the "pour milk into jug" sub-goal, using only the outputs from the accelerometers and force sensitive resistors in a CIC attached to the base of the jug, is investigated. The data used to train and evaluate the system was collected from non-patient subjects using the jug. The system uses a multiple state "left-right" HMM to capture the sequential structure of the "pour"

activity, and single state multiple component GMM HMMs for the “toy” and “rest” activities.

The experiments investigate the effect of changing the number of states in the “pour” model and the number of GMM components in the “toy” and “rest” models. Pre-processing of the CIC sensor data is also investigated. In particular simple thresholding is applied to the outputs of the FSRs, to overcome variability in the FSR data when the object is at rest on the work surface. The effects of low-pass filtering and differentiation of the FSR outputs are also investigated. A number of additional types of pre-processing are suggested.

So far, recognition accuracy achieved in the experiments is approximately 70%. However, this accuracy is expected to increase as the modelling improves.

7.6 Implementation of the AAR

Finally, the implementation of a real-time AAR is discussed.

REFERENCES

[1] S. Young, G Evermann, M J F Gales, T. Hain, D. Kershaw, G.Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev and P. Woodland, "The HTK book, version 3.4" , Cambridge University Engineering Department, 2006.

[2] Nicholas Roach, "The effect of multimodal data sources in parallel on the accuracy and reliability of optical motion capture of human subjects", PhD thesis, University of Birmingham, Birmingham, UK, 2011.

[3] Levin, E., Pieraccini, R., Eckert, W., (2000), "A stochastic model of human-machine interaction for learning dialogue strategies", *IEEE Transactions on Speech and Audio Processing*, Vol. 8, No. 1, January 2000, pp 11-23.

[4] Andrew W. Williams, Soila M. Pertet, and Priya Narasimhan. "Tiresias: Black-Box Failure Prediction in Distributed Systems". In: IPDPSIEEE (2007) , p. 1-8.

[5] Errin W. Fulp, Glenn A. Fink, Jereme N. Haack. "Predicting Computer System Failures Using Support Vector Machines". Proceedings of The First USENIX conference On Analysis Of System Logs. p.5-5, December 07, 2008, San Diego, California.

[6] Salfner.F, M. Schieschke, and M.Malek. "Predicting Failures Of Computer Systems: A Case Study For A Telecommunication System". In Proceedings of IEEE International Parallel And Distributed Processing Symposium (Ipdps 2006), Dpdns Workshop, Rhodes Island, Greece, Apr. 2006.

[7] Gary M. Weiss. "Timeweaver: a Genetic Algorithm for Identifying Predictive Patterns in Sequences of Events". In proceedings of the Genetic and Evolutionary Computation Conference, 718-725 Morgan Kaufmann, San Francisco, CA, 1999.

[8] Felix Salfner. Predicting Failures with Hidden Markov Models, 2002.

[9] Li, Q. Observer-Based Fault Detection for Nuclear Reactor. Massachusetts Institute of Technology, 2001.

APPENDICES

Page left intentionally blank