Action Recognition using Instrumented Objects for Stroke Rehabilitation

ROOZBEH NABIEI

School of Electronic, Electrical and System Engineering

University of Birmingham

A thesis submitted to the University of Birmingham for the degree of

DOCTOR OF PHILOSOPHY

May 2016

Abstract

Assisting patients to perform activities of daily living (ADLs) is a challenging task for both human and machine. Hence, developing a computer-based rehabilitation system to re-train patients to carry out daily activities is an essential step towards facilitating rehabilitation of stroke patients with apraxia and action disorganization syndrome (AADS). This thesis presents a real-time Hidden Markov Model (HMM) based human activity recognizer, and proposes a technique to reduce the time delay occurred during the decoding stage. Results are reported for complete tea-making trials. In this study, the input features are recorded using sensors attached to the objects involved in the tea making task, plus hand coordinate data captured using Kinect^{TM} sensor. A coaster of sensors, comprising an accelerometer and three force-sensitive resistors, are packaged in a unit which can be easily attached to the base of an object. A parallel asynchronous set of detectors, each responsible for the detection of one sub-goal in the tea-making task, are used to address challenges arising from overlaps between human actions. In this work HMMs are used to exploit temporal dependencies between actions and emission distributions are modelled by two generative and discriminative modelling techniques namely Gaussian Mixture Models (GMMs) and Deep Neural Networks (DNNs). Our experimental results show that HMM-DNN based systems outperform the GMM-HMM based systems by 18%. The proposed activity recognition system with the modified HMM topology provides a practical solution to the action recognition problem and reduces the time-delay by 64% with no loss in accuracy.

TO MY BELOVED PARENTS AND SISTER

Who taught me how to love unconditionally, to forgive and to forget always

Acknowledgements

I would like to express my sincere gratitude towards my supervisors Professor Martin Russell and Dr. Peter Jancovic for their excellent support, invaluable guidance, and kind encouragement throughout my PhD research. This research would never have been possible without their great assistance and support.

Many thanks also goes to members of the Cogwacth group, in particular alan Wing, Winnie Chua, Manish Parekh, chris bieber and Pia Rotshtein for their kind support during the project.

I am also very grateful to be surrounded by great friends who have provided moral support during this time, so another big thanks goes to my friends and colleagues from the University of Birmingham Maryam, Borna, Stevie, Masoud, Emilie, Hamid, Linxue, Saeid, Sina, Chloe, Mengjie, Eva, Xizi and Farzad for a great time we spent together.

Last but not least, special thanks to my parents, my sister Roja and his husband Mohammadreza, my brothers Khashayar and Mahyar, my uncles and aunts for their emotional support and constant encouragement. This thesis is especially dedicated to my wonderful parents.

Table of contents

List of figures

List of tables

List of Abbreviations

APM	Action Policy Module
ARS	Action Recognition System
ADL	Activity of Daily Living
ADC	Analogue to Digital Converter
AADS	Apraxia and Action Disorganization Syndrome
ANN	Artificial Neural Network
AAR	Automatic Action Recognition
BT	Black Tea
BTS	Black Tea with Sugar
CVA	Cerebrovascular Accident
CIC	CogWatch Instrumented Coasters
DBN	Deep Belief Network
DBN	Deep Belief Networks
DNN	Deep Neural Networks
DFT	Discrete Fourier Transform
DBN	Dynamic Bayesian Networks
DTW	Dynamic Time Warping
ERM	Error Recognition Module
EF	Errorfull
EL	Errorless
FSR	Force Sensitive Resistors
GMM	Gaussian Mixture Models
HMM	Hidden Markov Model
HTA	Hierarchical Task Analysis
HOG	Histogram of Oriented Gradient
HAR	Human Activity Recognition
HCI	Human Computer Interaction
IO	Instrumented Object
IOT	Internet OF Things
IR	Infra-Red
K-NN	K Nearest Neighbour
MDP	Markov Decision Process
MDP	Markov Decision Process
IMU	N Inertial Measurements Unit
NNS	Nearest Neighbour Search
OUAR	Object Usage based Activity Recognition

POMDP	Partially Observable Markov Decision Process
PHS	Personal Healthcare System
PSAR	Physical Sensor For Activity Recognition
PDF	Probability Density Function
RFID	Radio-Frequency Identification
RVM	Relevant Vector Machine
SIFT	Scale-Invariant Feature Transform
STV	Space-Time Volume
SDS	Spoken Dialogue System
SGD	Stochastic Gradient Descent
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TM	Task Manager
UPM	Technical University of Madrid
TUM	Technical University of Munich
UART	Universal Asynchronous Receiver/Transmitter
UOB	University of Birmingham
VSAR	Video Sensor based Activity Recognition
WSAR	Wearable Sensor based Activity Recognition
WT	White Tea
WTS	White Tea with Sugar

CHAPTER 1

INTRODUCTION

Strokes, also known as a cerebrovascular accident (CVA), occur approximately 152,000 times a year in the UK; that is one every 3 minutes 27 seconds [4]. Recent studies have found that 68% of stroke survivors in the UK suffer from Apraxia or Action Disorganization Syndrome (AADS), which leads to impairment of cognitive abilities to complete activities of daily living (ADLs) [5, 6]. Patients with AADS, might perform a sequence of actions in the wrong order, skip mandatory steps towards completion of the task, and misuse objects with possible safety implications. It also has been observed that they could make errors such as hesitating a long period of time trying to decide what to do next, repeatedly verifying if the action is completed, moving object without knowing their use, or undo a part of the task that has already been completed. These errors can relate to the defective use of real tools and objects [7], the inability to correctly select appropriate tools for a task [8], or the inability to complete sequences of actions [9]. These impairments are typically associated with loss of action knowledge [10], attention and executive function deficits [11], and/or loss of object knowledge [12]. Such acquired neurological deficits have a direct impact on stroke survivors' daily life, relatives, caregivers and society as a whole.

For patients who aspire an independent living, existing methods of rehabilitation, which are hugely dependent on the help of caregivers, are generally unsatisfactory and very expensive. A cost-effective rehabilitation system for strokes requires maximising the independence of the patient with minimising the hospital admissions, through provision of home-based (community) services.

The objective of the CogWatch project [13] is to develop an intelligent human computer interaction (HCI) system, which is affordable, user-friendly, portable and non-restrictive. The CogWatch system is a long-term solution for stroke patients to recover their cognitive abilities in performing ADLs, without going to hospital. The system works by monitoring the patient's progress throughout the completion of ADL and providing appropriate guiding cues or feedbacks when an error is detected or anticipated. Recognising the individual actions that make up the ADL, and planning the patient's optimal strategy to successfully finishing the task are critical.

This study proposes an action recognition system, based on state-of-the-art pattern recognition algorithms. In this work temporal dependencies between actions are modelled using Hidden Markov Models (HMMs) and emission distributions are modelled by a generative model using Gaussian Mixture Models (GMMs) and a discriminative model using Deep Neural Networks (DNNs). In the next section, the technical implementation of the CogWatch project is explained.

1.1 THE COGWATCH PROJECT

CogWatch[1] is a personal healthcare system (PHS) that provides a cognitive rehabilitation to stroke patients with Apraxia or Action Disorganization Syndrome (AADS) in a non-clinical environment. The rehabilitation in CogWatch is accomplished by providing appropriate

[1] http://www.cogwatch.eu/

guidance and feedback to the patients during activities of daily living (ADL). In this thesis, the focus is on the tea-making task and it's variants which is the first and main ADL addressed in the CogWatch project.

CogWatch is a European commissioned project that is co-ordinated by the University of Birmingham (UOB) in collaboration with with the Technical University of Madrid (UPM) and Technical University of Munich (TUM); other partners involved in the project are BMT Group Ltd, RGB Medical Devices, the Stroke Association and Headwise Ltd. Partners in the CogWatch project and their main contributions are as follow:

- University of Birmingham
  - Department of Electronic, Electrical and Systems Engineering: responsible for the design of the CogWatch instrumented coasters (CICs), development of the action recognition system (ARS), and implementation of the task manager.
  - School of Psychology: undertaking researches about the psychology and health-care aspect of the project, and providing information about the patient's behaviour for development of the technical part (task manger). Recording video and sound cues.
- Technical University of Madrid: Development of the interface of the system including the patient's and clinician's screens; patients could choose the type of the task and receive cues on their screen, and clinicians have the ability to control the session on theirs (for example, manually sending actions to the task manager instead of waiting for ARS).
- Technical University of Munich (TUM): carrying out further psychology and health related researches.

- Other partners helped in commercialization of the system and adding more patient monitoring devices (e.g. blood pressure). The full list of the partners can be find in the CogWatch website [2].

Except for the action recognition system (ARS) module the rest of the system were designed and managed by other researchers and colleagues who have been involved in the project. In the following sections, the different parts of the CogWatch system and the error types are explained.

1.1.1 TASK DEFINITION

Stroke patients with AADS are more likely to make errors when performing ADL. For instance, in [12], the authors observed the behaviour of a patient with AADS while making a coffee and brushing teeth. The authors reported that the patient made several errors, such as not opening the milk container before pouring, or adding coffee grinds into the bowl of oatmeal.

In the first stages of CogWatch development, ADLs suitable for home scenarios (activities such as drink and meal preparation, dressing, tooth-brushing, and personal hygiene) were considered for creation of the first prototype. From possible alternatives, the tea-making task was selected as the first task; this task was chosen because it is a generally known activity relevant to everyday life, should be familiar to the majority of participants, at an appropriate level of complexity to make sure a substantial number of apraxic patients are included. Furthermore, drink prepration (coffee or tea making) has been thoroughly studied in the psychology literatures, thus providing a starting point for the patient results comparison.

In CogWatch, four types of tea are considered: Black tea (BT), Black tea with sugar (BTS), White tea (WT) and White tea with sugar (WTS). Once the tea-making task had

[2] http://www.cogwatch.eu/

been chosen Hierarchical Task Analysis (HTA) was applied to derive a hierarchical task tree. Higher levels of the tree describe the task in greater detail. The first level above the root of the tree decomposes the task into sub-goals. These are the basic units that were used to describe the task in the CogWatch project. The sub-goals of tea making are listed below. More detail about HTA is given in Section 3.1.

- Filling the kettle with water
- Boiling the water
- Pouring boiled water from the kettle to the mug
- Adding milk
- Adding teabag
- Adding sugar
- Removing teabag
- Stirring

1.1.2 ERROR TYPES

Two methods of learning could be practised when it comes to rehabilitation of stroke patients. For prompting a cognitively impaired persons, clinicians can follow an errorless (EL) or errorfull (EF) technique. In the case of an EL technique, prompts are given to the patients at each step of the task to lead them complete the task with no errors. This technique was promoted by Werd et al. in the case of dementia, which is an umbrella term for symptoms such as loss of memory, language or other intellectual capacities [14]. However, when the EF technique is applied, patients take all the initiatives during the task, and receive cues only when they make mistakes or clinicians decide to do so. Middleton et al. and van Heugten promoted this technique in the case of stroke [15, 16]. Research in psychology has

shown that with intensive rehabilitation therapy a stroke survivor with AADS can re-learn to complete ADLs. Therefore for AADS the emphasis is on rehabilitation for re-learning and hence EF. In the case of dementia the emphasis is on assistance to enable the patient to function independently rather than rehabilitation. For an assistive technology EL is the appropriate approach.

Schwartz et al. [12] categorised errors made by apraxic patients during the coffee making task in six types: (1) place substitutions (transporting objects to the wrong destination), (2) object substitution (misuse of objects; for example, adding oatmeal or orange juice to the mug contents), (3) drinking anticipation (start drinking before the task is finished; for example, drink prior to ground coffee being poured), (4) omissions (Situations in which an action could not be completed because of failing a necessary prior action; e.g., failure to open the cream container before pouring), (5) instrumented substitutions (using the wrong equipment), and (6) faulty execution (an action was attempted but not finished appropriately; for example, the sugar pack was not completely opened). In the case of CogWatch and the tea-making task, the errors that the system should be able to detect are defined and categorised as follow:

1. Addition: An addition error occurs when the user carries out an action that belongs to another task. For example, adding sugar in to the cup when the user specifically decided to make a "Black tea".

2. Perseveration: A perseveration error happens when the user repeats any action that has already been performed during the task.

3. Anticipation: Some actions in tea-making task are sequentially restricted, meaning one should happen before the other. For example, the user must fill the kettle with water before turning it on. An anticipation error happens when the user performs such actions in wrong order.

4. Perplexity: A perplexity error occurs when the user does not perform any action during a specific amount of time T. A counter is reset after each action performed by the user, and the system is programmed to contemplate that the user needs help if no appropriate action is received after T seconds.

5. Omission: An omission error occurs when the user has missed an action but considers the task is completed. For example, the user considers that the task is over, but he or she has not put teabag in the cup.

6. Fatal Error: A fatal error is an error from which the patient cannot recover (e.g., adding milk to a black tea), or when one of the errors described above are repeated too many times.

7. Quantity Error: A quantity error occurs when the amount of ingredients added to the cup is either too much or too little (e.g., adding too much sugar).

Detection of errors can be performed directly by the action recognition system (ARS) by building explicit models of specific errors. Alternatively, errors can be detected by the Task Model using the outputs of the ARS, however this would require the ARS to send the TM a richer description of the subject actions than the current sub-goals (for example, to detect a tea bag being put into the kettle or using the outputs of more sensitive FSRs to detect quantity errors). The CogWatch system used both of these approaches. Errors that are potentially dangerous, such as toying with a kettle, are modelled explicitly and detected by the ARS, while other, less serious errors are inferred by the Task Model. In Table 1.1, one can see the specific types of errors that the CogWatch can automatically detect during a trial.

1.1.3 OVERALL COGWATCH ARCHITECTURE

Structure of the CogWatch system is shown in Figure 1.1. It consists of sets of sensors (monitoring module), an automatic action recognition module, a task manager, and a cue selector (cue generation module).

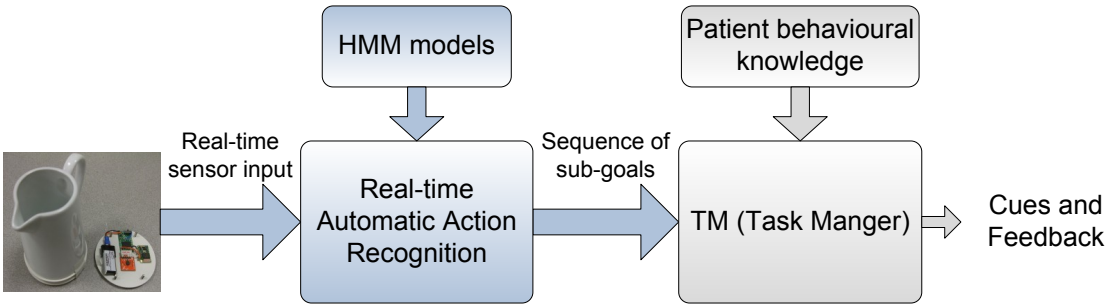


Fig. 1.1 *Diagram of the CogWatch rehabilitaion system*

Initially, the User interacts with the system through a touch-screen tablet to choose the type of tea (e.g., white tea with sugar) intend to receive training for. After choosing the task, the user begins the process of making the desired type of tea at their own pace. During the completion of the task, the position of the user's hand on the table is tracked by a Kinect camera, while sensors in some of the object indicate the type of motion that the object is subject to, whether the object is resting on the surface and, to some extent, the weight of the object. Data from the Kinect and sensorized objects are passed to the ARS module to recognize the completed action. The output of the automatic action recognition system is a sequence of actions carried out by the user. Task manager uses the output of the action recognition system to estimate the user state, which is a simplified model of what user have achieved. A user state in the task model corresponds to a sequence of sub-goals that can be extended to achieve successful tea-making. Considering the user state, the task manger can output the optimal next recommendation (feedback) to the user.

The system depicted in Figure 1.1 resembles a Spoken Dialogue System (SDS). In an SDS the inputs are from an acoustic sensor (a microphone) rather than motion sensors. The action recognition system is replaced by a speech recognition system and the task manager is replaced by the dialogue manager. This similarity led us to adopt technologies from SDS, such as the use of hidden Markov models for acti n recognition and a (Partially Observable) Markov Decison Process based Task Model, in the CogWatch system.

1.1.4 ACTION RECOGNITION SYSTEM

The development and evaluation of the action recognition system (ARS) is the main focus of this research. The sensorized objects and Kinect^{TM} monitor the progress of the task by continuously transmitting data to the ARS. The aim of the ARS is to analyse these data to automatically spot what actions are being performed by the user during the task, and to send its observations to the Task Manager. The output of the ARS is a sequence of actions performed by the user. In CogWatch's ARS, actions are defined to be binary whereas they could be continuous; For example, when the user adds some milk into the mug the ARS will send a flag to the task manager to determine that the "Add milk" action is done, instead of identifying how much milk is added. The continuous recognition of actions is not possible due to the drop in accuracy of relevant weight sensors during the process of data recording, because the coaster of sensors are a prototype design and they are not waterproof, so they are easily worn off. However, the binary recognition of actions is sufficient for the rehabilitation purpose and CogWatch application.

The ARS can output nine different observations as far as the user's behaviour is concerned. Seven out of ten observations correspond to the actions defined in Section 1.1.1. The other observations correspond to erroneous actions that can be inferred from the way objects are moved or used: (1) "toying with the milk container", (2) "Toying with boiling water" - users

with AADS may have such hazardous behaviours that need to be detected early in the process in order to send them appropriate alerts..

1.1.5 TASK MANAGER

In CogWatch, the task manger's overall job is to track the user's history of actions during the tea making task, and to provide related assistance when the user needs help or an error is detected in the sequence of actions. To do this, the task manager process observations in three main modules: the State Modeler, the Action Policy Module (APM) and the Error Recognition Module (ERM). Task manager receives recognised actions from the action recognition system (ARS) in real time, in other words, the task manager (TM) takes the output of the ASR as its input. The State Modeler employs this information to build a representation r_s of the user's state, which is an abstract model of what the user has achieved so far during the task. This abstract state r_s is then passed to the APM and ERM, to decide what are the optimal prompts to be sent to the user during the task. The APM is responsible to work out what are the optimal next instructions to send to the user, if required; and the ERM is responsible for detection of potential errors in the user's behaviour. Together, they select the most appropriate cue to give to the user.

In the prototype and final version of the CogWatch's task manger, a Markov decision process (MDP) and partially observable Markov decision process (POMDP) based planing system is implemented for decision making, respectively. The POMDP was employed in the task manager to accommodate uncertainties in received input from the ARS [17]. In the MDP a user state corresponds to a sequence of actions that can be extended to successful tea-making. A POMDP acknowledges the fact that, due to errors in the output of the ARS, the user state cannot be known precisely. Therefore it utilises the idea of "belief state" which is a probability distribution over the set of all MDP user states. When the POMDP-based

Task Model receives an action from the ARS it uses this action, plus its knowledge of the probabilities of the different errors that the ARS might make, to update the probabilities in the belief state. In a POMDP the belief state space is infinite. The POMDP can only learn optimal strategies for the belief states that it encounters during training. When it is being used, belief states will occur that were not seen during training. The problem therefore arises of what action the user should take when he or she is in such an unseen state. The solution is to use a nearest neighbour search (NNS) technique to find the closest belief state to the current state that was observed during training and therefore is associated with an optimal action.

The accuracy of the prompts being generated by the POMPD-based APM is influenced by three parameters:

1. Task complexity: The accuracy of the POMDP-based TM is related to the complexity of the task. As an example, for the task of making a "White tea with sugar", each belief state consists of probability distribution over 1539 states; On the other hand, this number is reduced to 33 states for making a "Black tea". In CogWatch

2. Nearest Neighbour Search technique: Variety of NNS techniques has been implemented in the POMDP-based TM, such as, Euclidean, support vector machines (SVM), K-D trees, Manhattan, Correlation and metric-based algorithms. In [17], author claims that the novel metric-based "SciMK" technique outperformed the others.

3. ARS error rate: As the ARS error rate increases so does the number of wrong observations being sent to the task manager. Even though, the latest version of the task manager is designed to cope with such uncertainties, eventually the number of erroneous prompts output by the system would increase when the input's ambiguity rises. POMDP-based TM employs the ARS confusion matrix together with the recognised action to update the probabilities in all of its belief states.

In [17], authors have generated observations from a simulated ARS with different overall error rates of 10%, 20% and 30%. The quality of prompt generation is measured in terms of the percentage of times that the ADL is successfully completed if the user adheres to the prompts generated by the TM. The evaluation is repeated for simulated results for "Black tea", "Black tea with sugar" and "white tea with sugar" tasks. In the case of the "Black tea" task, the POMPD-based TM has managed to achieve 100% accuracy and deal with all ARS error rates. During the "black tea with sugar" task, the accuracy drops from 100% to 83% when the ARS error rates increased from 20% to 30%. However, when the task gets more complex, for example, in the case of the "White tea with sugar", the performance of the task manger is even more effected by the ambiguity of the ARS observations; In the case of the ARS with error rate of 10%, 20% and 30%, simulated user adhering to the TM prompts achieves successful task completion in 90%, 72% and 46% of the trials, respectively. Consequently, with the assumption that the patients fully comply with the prompts and the ARS error rate is set at 10% , minimum 90% of trials could successfully be completed by the user.

1.1.6 EFFICACY STUDY

In [18], twelve patients performed 96 trials of tea making, both with and without interacting with the CogWatch; the patients made less errors and successfully completed the task 95.8% of the time with the help of the CogWatch, compare to only 67.7% without. Also, at early stages of the project patients used the system in a longer time intervals and in some cases in hospital to investigate its efficiency for rehabilitation. At these stages, the system was equipped with an early version of the ARS capable of recognising two "Add Milk" and "Pour boiling water", and a task manager with MDP implementation.

1.1.7 CONCLUSION

In this section, the architecture and the main purpose of the intelligent part of the CogWatch system was described. The detail and goal of the two intelligent system components (action recognition system and task model), and the relation between them was explained.

The focus in rest of this thesis is the development and evaluation of the action recognition system and its impact on the ability of CogWatch to be an automated home-based rehabilitation system.

1.2 RESEARCH QUESTIONS

Human action recognition is a very important and challenging research topic. In this growing area, most researches are focused on the recognition of the high-level human activities like, sitting, standing, walking and cooking. However, in order to use the recognition system in more specific tasks like rehabilitation, detection of low-level actions is needed.

In this section, the main research questions and challenges of designing an action recognition system are outlined. These questions define the direction and the boundary of the research. The aim of the thesis, is to answer and discuss each research question in relevant chapters. The discussions can be found in the conclusion sections of the chapters.

1. Can techniques from automatic speech recognition(ASR), specifically hidden Markov models (HMMs), be applied to the outputs of sensorized objects and Kinect^{TM}, and deliver sufficiently high-accuracy recognition of human actions during activities of daily living (ADLs), specifically tea-making? Here the sufficient accuracy can be defined with respect to the discussion in Section 1.1.5.

2. Hierarchical Task Analysis models a task in terms of a set of sub-goals (for example "add milk" is a sub-goal of "make white tea"). Are these sub-goals appropriate units for HMM-based modelling of human actions?

3. ASR techniques and tools are normally used to detect temporally sequential units of spoken language. But, can methods like HMMs be employed and designed in away to accommodate recognition of concurrent human actions?

4. The output of the ARS is used by the task model. Can detection of actions during an ADL happen in real time?

5. In recent years, HMMs that use Deep Neural Networks (DNN-HMMs) have been shown to outperform conventional HMMs for automatic speech recognition. Do DNN-HMMs have similar advantages for human action recognition?

1.3 THESIS OUTLINE

This thesis consists of the following chapters:

- **CHAPTER 2 - LITERATURE REVIEW AND TECHNICAL BACKGROUND:**
  This chapter provides a literature review of related works carried out for capturing ADLs using different categories of sensing approaches such as wearable, pervasive and pervasive-wearable. Furthermore, the chapter includes background review of techniques used for development of the action recognition systems such as overview of the HMM system, conventional Viterbi algorithm and the training process of a DNN-HMM system.

- **CHAPTER 3 - TEA-MAKING DATABASE:**
  In this chapter, the high-level tea-making task, which is the first ADL used in CogWatch,

is analysed and broken down into lower-level objectives of sub-goals. Also, the tea-making table setup with location of the objects on the table is introduced. Finally, the tea-making database is created by recording the execution of individual sub-goals and complete trials of making a cup of tea. The database is later employed in the experiments described in Chapter 7.

- **CHAPTER 4 - SENSORS AND FEATURE EXTRACTION:**

  In this chapter, different sensors , such as the CogWatch instrumented coasters (CIC) and Kinect^{TM}, that were used to record and monitor the user's activities throughout the completion of the ADL are introduced. Furthermore, some basic signal processing and feature extraction methods are applied to the data captured from the sensors. Processed data are used in the action recognition system to detect the sub-goals that make up the tea-making task.

- **CHAPTER 5 - REAL-TIME ACTION RECOGNITION OVERVIEW:**

  In this chapter, the challenges for the real-time recognition of concurrent activities are described and addressed by introducing the parallel architecture of detectors. The AR system comprises six independent real-time HMM-based detectors which together can identify occurrences of all sub-goals at any time during completion of the task. The different parts and the overall structure of a detector are presented. In addition, the proposed real-time Viterbi algorithm is introduced and explained in this chapter.

- **CHAPTER 6 - SUB-GOAL DETECTORS' CONFIGURATIONS**

  In this chapter, the target sub-goals, HMM models and feature-vector for each individual detector are presented. In addition, different types of HMM topology and a model-loop decoding network, which are shared amongst all detectors, are described.

- **CHAPTER 7 - EXPERIMENTAL DESIGN:**

  In this chapter, the process of training an optimal set of HMM models for the real-time

system, is explained. The process consists of two sets of experiments: verification of the isolated sub-goal recordings and spotting of sub-goals in the full-trial recordings of making a cup of tea. The sub-goal detection appears in the output of the real-time decoder with a time-delay; a modification to the HMM-sets is proposed to reduce the time of the output delay.

- **CHAPTER 8 - GMM-HMM EXPERIMENT RESULTS:**

  The experiments that ere conducted in Chapter 7 are repeated, to optimise the parameters of the GMM-HMM models and features for each individual detector. In this chapter, results of these experiments, are reported and discussed. Also, for each detector the optimum HMM-set is selected to be used in the real-time action recognition. In the last section of the chapter, the output delay of the real-time sub-goal spotting, using the original and modified versions of the optimum HMM-sets, are measured and compared.

- **CHAPTER 9 - DNN-HMM SYSTEM:**

  In this chapter, results of the sub-goal spotting, using a DNN-HMM based action recognition system, are reported and compared with the results achieved from a baseline GMM-HMM system. The comparison demonstrates that employing a DNN-HMM system can be beneficial for action recognition using the monitory sensors.

- **CHAPTER 10 - CONCLUSION:**

  The final chapter summarises the major results and conclusions of the thesis and suggests future directions for research.

1.4 KEY CONTRIBUTIONS OF THE THESIS

The research described in this thesis provides original contributions to the fields of automatic action recognition, sensing technology and healthcare. The major contributions can be summarised as follows:

- Exploitation of analogies between a spoken dialogue system and task monitoring for stroke rehabilitation.
- Application of a parallel asynchronous architecture of the HMM-based action detectors, to deal with the concurrent sub-goals and the benefit from the customised selection of features in each detector.
- Proposal of real-time and memory efficient implementation of the Viterbi algorithm, employing the trace-back algorithm.
- Integration of the real-time action recognition into the CogWatch rehabilitation system, which has been presented to hospitals and conferences. The CogWatch system has been shown to have a positive effect on the rehabilitation process for patients who suffer from Apraxia.
- Collection of a tea-making database, specifically for training and optimisation of the HMM models. The dataset consists of recordings from a Kinect^{TM}, an inertial measurements unit (IMU) like accelerometer sensors, and force sensitive resistors (FSRs). The dataset will be made available on the Internet.
- Improvement of the sub-goal spotting performance by using a DNN-HMM recognition system instead of GMM-HMM.

1.5 PUBLICATIONS RESULTING FROM THIS THESIS

Some of the ideas and results in this thesis have been published in reviewed conference papers as follows:

- "Delay reduction in real-time recognition of human activity for stroke rehabilitation," R. Nabiei, M. Najafian, M. Parekh, P. Jancovic, and M. Russell, in Sensing, Processing and Learning for Intelligent Machines (SPLINE), 2016, IEEE.
- "Object-centred recognition of human activity," R. Nabiei, M. Parekh, E. Jean-Baptiste, P. Jancovic, and M. Russell, in Healthcare Informatics (ICHI), 2015, IEEE.
- "Intelligent assistive system using real-time action recognition for stroke survivors," E. Jean-Baptiste, R. Nabiei, M. Parekh, E. Fringi, B. Drozdowska, C. Baber, P. Jancovic, P. Rotshein, and M. Russell, in Healthcare Informatics (ICHI), 2014, IEEE .

Table 1.1 *Types of user's behaviors that the TM can detect. BT = Black tea, BTS = Black tea with sugar, WT = White tea, WTS = White tea with sugar, "all" = [BT, BTS, WT, WTS]. AD = Addition error, OM = Omission error, FE = Fatal error, PE = Perplexity error, PsE = Perseveration error, AN = Anticipation error, QT = Quantity error, BTr = button trigger, NE = Not an error. [3]*

ID	Error type	Task	Description of user interaction with CogWatch
E01	BTr	all	Presses "Help Button"
E02	PE	all	Makes long pause during task
E03	FE	all	Toys
E04	BTr	all	Presses "Finish Button" when not required
E05	OM	all	Omits to press "Finish Button" when required
E06	PsE	all	Adds water to kettle multiple times
E07	OM	all	Omits to add water to kettle
E08	OM	all	Omits to boil water
E09	PsE	all	Boils water multiple times
E10	OM	all	Omits to add teabag to cup
E11	PsE	all	Adds teabag multiple times
E12	AN	all	Pours water to cup before boiling water
E13	PsE	all	Pours water to cup multiple times after boiling water
E14	OM	all	Omits to add boiled water to cup
E15	OM	all	Omits to remove teabag from cup when required
E16	AN	all	Stirs while no water is in the cup
E17	OM	BTS, WTS	Omits to stir
E18	PsE	BT	Stirs multiple times
E19	PsE	WT, BTS	Stirs multiple times
E20	PsE	WTS	Stirs multiple times
E21	AD	BT, WT	Adds sugar when not required (based on type of task)
E22	QT	BTS, WTS	Adds too much sugar
E23	OM	BTS, WTS	Omits to add sugar
E24	AD	BT, BTS	Adds milk when not required (based on type of task)
E25	QT	WT, WTS	Adds too much milk
E26	OM	WT, WTS	Omits to add milk
E27	AN	all	Boils water before adding water to kettle
E28	AN	all	Removes teabag before adding boiled water to cup
C01	NE	all	Task successfully completed

CHAPTER 2

LITERATURE REVIEW AND TECHNICAL BACKGROUND

2.1 INTRODUCTION

Research on human activity recognition has made significant progress in the last decade and is attracting growing attention in a number of application domains, such as surveillance and entertainment environments, and healthcare systems. Despite much progress made in human action recognition, achieving high recognition rates under realistic conditions still remains a challenge. Within the context of intelligent human assistive systems a number of interfaces has been proposed in the literature.

We start this chapter by reviewing related work on action recognition and then we present the background for algorithms used in this work, namely Hidden Markov Models (HMMs), Gausssian Mixture Models (GMMs), Deep Neural Networks (DNNs), their training criteria, and their parameter estimation process. Then, we describe Deep Belief Networks (DBNs) and their application in initialising DNNs model parameters (pre-training).

2.2 RELATED WORK

Human activity recognition (HAR) has become a hot research topic due to its strength in providing personalized support for many diverse applications. These application can be categorized as, (1) security and surveillance (in airports, train stations and banks), (2) intelligent environment (smart homes, offices, hospitals and cars), (3) healthcare (rehabilitation, assistive living, well-being monitoring) and (4) military (soldier monitoring in battlefields). Due to its importance, a considerable amount of AR systems have been developed. In general, these systems utilize diverse sensors to obtain the activity related information, which are then used by machine learning techniques to infer human's ongoing activity.

According to the sensing techniques, existing AR can be roughly divided into two categories: video sensor based activity recognition (VSAR) and physical sensor for activity recognition (PSAR).

2.2.1 VIDEO SENSOR BASED ACTIVITY RECOGNITION (VSAR)

This is the classical activity recognition approach which attempts to capture human's activity information by appropriately placed cameras. Video analysis is the most important and challenging part in VSAR. Generally speaking in video analysis of an AR systems, three main processing stages are considered: (1) object segmentation, (2) feature extraction and representation, and (3) activity detection and classification algorithms.

1. Depending on the mobility of cameras, the object segmentation can be categorized as two types of static or moving camera segmentation. The most popular method for static camera segmentation is background subtraction [19–21] due to its simplicity and efficiency. The background subtraction is very sensitive to illumination changes. On the other hand, for moving camera segmentation, the camera is moving with

a photographer or a robot. The moving camera segmentation is more challenging because, it needs to consider the motion of the camera and the change of background. The most common method for moving camera segmentation is the temporal difference [22, 23], i.e., the difference between consecutive image frames.

2. Characteristics of the segmented objects such as shape, silhouette, colors and motions are extracted and represented in some form of features. Generally speaking, the features can be categorized as four groups, space-time information, frequency transform, local descriptors and body modeling. The space-time information is first considered. The space-time volume (STV) [24–27] is built as the image features by concatenating the consecutive silhouette of objects along the time axis. Compared to spatial-temporal domain image, the frequency domain information can also be exploited. More specifically, the discrete Fourier transform (DFT) [28], which has been widely used to represent information about the geometric structure of the object. The STV and DFT features belong to global features which consider the whole image so that they are limited on viewpoint changes and occlusion. Hence, some local descriptors are considered. The local descriptors [29–34], such as scale-invariant feature transform (SIFT) [35, 36] and histogram of oriented gradient (HOG) [37] capture the characteristics of an image patch. They are ideally invariant to background clutters, appearance and occlusions, and also invariant to rotation and scale in some cases. However, the above mentioned feature representations do not fully capture the whole body actions. Therefore, some human modelling methods [19, 38–47] are also proposed to model the human body including simple blobs, 2D body modelling and 3D body modelling.

3. The activity detection and classification algorithms are used to recognize various human activities based on the represented features. They can generally be categorized as dynamic time warping (DTW), generative models, discriminative models and others. The dynamic time warping (DTW) [31, 48], a method for measuring

similarity between two temporal sequences, which may vary in time or speed, is one of the most common temporal classification algorithms due to its simplicity; however, DTW is not appropriate for a large number of classes with many variations. Some probability-based methods by generative models (dynamic classifiers) are proposed such as Hidden Markov Models (HMM) [49–53] and Dynamic Bayesian Networks (DBN) [54–56]. On the other hand, discriminative models (static classifiers) such as Support Vector Machine (SVM) [57–59], Relevant Vector Machine (RVM) [60, 61] and Artificial Neural Network (ANN) [62, 63], can also be used in this stage. For both of the probability model-based algorithms, including generative models and discriminative models, their performance relies on extensive training dataset. Therefore, other methods are proposed, such as Kalman filter [64, 65], binary tree [66, 67], multidimensional indexing [68], and K nearest neighbor (K-NN) [28]. Different classification algorithms usually require different sets of suitable feature representations.

APPLICATIONS OF VSAR

Human activities can be single-person actions, such as walking, waving, ballet dancing; the interactions between persons, such as hand shaking, hugging; or the interaction between humans and object, such as preparing a meal, brushing teeth. One or more of these activities put together in different scenarios, can be suitable to be used in each specific domain of applications. In this part, we focus on two dominant applications of the VSAR, surveillance environments and healthcare systems.

The application of human activity recognition in surveillance systems mainly focus on automatically tracking individuals and crowds, so as to support security personnel to observe and understand activities, resulting in recognition of the criminal and detecting suspicious activities. People detecting and tracking is one of the first objectives of a security surveillance

system [40]. the tracking results are further exploited to detect suspicious behaviours [63], such as loitering [69, 70], entering a "secured area" and moving against traffic [64]. Besides this, some researchers perform detection of various kinds of violent behaviours such as fighting, punching and stalking [48, 66, 71–74].

The applications for activity recognition in healthcare systems facilitate health workers to diagnose, treat and care for patients, resulting in improving the reliability of diagnosis, decreasing the working load for the medical personnel, shortening the hospital stay for patients, and improving patients' quality of life [49, 75–79, 62].

One way to use VSAR in health care application is to monitor the daily life activity of seniors, this help to recognise any abnormalities. For example, automated respiration monitoring is performed by Kuo et al. [75], and results are used to detect health issues, such as, obstructive sleep-apnea syndrome, cardiovascular disease, and stroke. The Falling is a major health risk of elderly as it is known to be the leading cause of injury and deaths among seniors. Foroughi et al. [62, 77, 78] have used different feature extraction techniques and classification methods in various studies to detect the fall activity.

Rehabilitation systems are another form of action recognition application in health care. Very similar to the Cogwatch project, Ghali et al. [79] design a system to provide real-time feedback to stroke patients performing the daily activity of making cup of coffee. The position and movement of the patient's hands and the objects he/she manipulates are captured by overhead cameras and monitored using histogram-based recognition methods. Objects are of very different colours that are also distinct from the background. Shadows cast by objects and hands reduce the recognition performance, so dark background is used. The key events (e.g., picking up a cup) are recognized and interpreted in the context of a model of the coffee-making task.

In order to objectively evaluate the improvement of motor functions of the elders at home, as well as to reduce burden on fitness instructors, Ryuichi et al. [80] propose a "multimedia fitness exercise progress notes" system, where the video capturing exercise movements of the elders are sent to an analysis center. Snapshots of the captured videos are used to semi-automatically measure many kinds of exercise parameters, such as lap time, distances and angles.

2.2.2 PHYSICAL SENSOR BASED ACTIVITY RECOGNITION (PSAR)

More recent technology gain attention for action recognition purposes are physical sensors. Generally, physical sensor can be attached either to the human body (e.g., hand, leg, hand and waist) or objects (e.g., cooking tools, home or office objects). Therefore, PSAR systems are further divided into two sub-categories: wearable sensor based activity recognition (WSAR) and object usage based activity recognition (OUAR).

2.2.3 WEARABLE SENSOR BASED ACTIVITY RECOGNITION (WSAR)

Wearable sensors are positioned on different parts of human body, and they generate signals when the user performs activities. They monitor features that are descriptive of the user's physiological state or movement. Wearable sensors can be embedded into clothes, eyeglasses, belts, shoes, wristwatches, mobile devices, or positioned directly on the body. They can be used to collect information such as body position and movement, heartbeat, and skin temperature.

VARIOUS SENSORS IN WSAR

In VSAR, only video sensor is involved. However, WSAR involves many different kinds of sensors. Each type of sensor has its own strengths and suitable applications. In the following, we will review the sensors used in WSAR.

The types of sensors used in WSAR range from relatively simple sensors with discrete output, such as ball switches [81], to sensors with continuous output such as inertial sensors like accelerometers and gyroscopes [82, 83], to more complex sensing methods such as audio processing [84, 84]. Among all sensors, accelerometers are probably the most commonly used type of sensor for WSAR. It mainly because they are more accurate at detecting movements, and usually lead to good results in recognition of physical activities. In addition to above sensors, there are some sensors which are less common used. Fiber-optical sensors are used in [85] to measure posture. Foam pressure sensors are used to measure respiration rate [86]. Force sensitive resistors are used to measure muscle contractions [87]. Moreover, various kinds of physiological sensors such as oximetry sensors [88], skin conductivity sensors [89], electrocardiographs [90], and body temperature sensors have also been used. Depending on the type of activity, recognition performance can be improved by using the same type of sensor at multiple body locations [81, 82, 91].

APPLICATIONS OF WSAR

Possibility of using sensors and microprocessors smaller than a coin, enables WSAR systems being regularly used in our daily lives in forms of different applications, such as entertainment and gaming [92, 93], industrial [94, 95] and healthcare[96–99, 108]. A significant class of applications is for healthcare and assisted living.

WSAR can help elderly people to live more independently by monitoring their home environment while they are remotely monitored for safety and for the purpose of facilitating

the implementation of clinical interventions. Accordingly, extensive research efforts have been made to assess the accuracy of wearable sensors in classifying activities of daily living (ADL). Mathie et al [96] showed the feasibility of using accelerometers to identify the performance of ADL by older adults monitored in the home environment. Sazonov et al [97] developed an in-shoe pressure and acceleration sensor system that was used to classify activities including sitting, standing, and walking with the ability of detecting whether subjects were simultaneously performing arm reaching movements. Giansanti et al [98] developed an accelerometer-based device designed for step counting in patients with Parkinson's disease.

By detecting the activities of elder people, we can implicitly detect potentially dangerous situations in an elder person's life. The typical functions of these systems are to detect the fall down of a person [99, 100], or detect the vital body signs [101, 102]

Some diseases can be reflected from activity information before they actually happen. For example, Alzheimer can be detected by some early symptoms related to human's activity. These applications accumulate and summarize statistics about daily activities [103] or perform continuous recordings of physiological parameters [101, 104–106]. These applications can help physicians and caregivers to estimate the physical well-being of a person.

There are applications which aims to promote elderly or disabled people in performing everyday activities and thus generates more healthy lifestyle. With the increasing intelligence and process power of mobile phone, many such applications use mobile phone to remind human to perform certain activities [85, 107, 108].

2.2.4 OBJECT USAGE BASED ACTIVITY RECOGNITION (OUAR)

In our daily life we usually perform an activity by interacting with a series of objects. For example, for bathing we may interact with a door, light, exhaust fan, shower faucet, etc. The strategy of object usage based activity recognition (OUAR) is to attach sensors on these objects such that it is possible to determine the state of that object should a person interact with it.

Object usage based activity recognition (OUAR) [109–113] can be particularly useful in domains such as cooking, which involve a relatively small number of repeated actions such as chopping, pouring, spreading, etc. Object use information can help discriminate between activities such as making toast and making a sandwich, which may be similar from the standpoint of the activity alone. Such distinctions can be important for application domains such as health monitoring or memory aids.

2.2.5 COMPARISON BETWEEN VSAR, WSAR AND OUAR

In this section, further discussion about strength and limitation of video based action recognition (VSAR) and physical sensor based action recognition (PSAR) is made. Based on the comparisons made between VSAR and PSAR, fusing the two approaches together seems to be a promising solution for complex activity recognition applications.

VSAR

VSAR is the most traditional activity recognition method. If this method is implemented with use of appropriately placed cameras, it could provide richest information and facilitate the most detailed analysis. The signal processing involved in extracting this information however can be computationally intensive which is an open problem. Generally, VSAR often

works well in a laboratory or well-controlled environment. However, it fails in achieving the same level of accuracy under a real home setting due to the variable lighting and occlusion. VSAR is complex to implement because it requires processing of highly multidimensional data. The biggest challenge to use VSAR in home environment is the possible violation to user's privacy.

WSAR

Using these types of wearable sensors, sensing studies have successfully recognized such activities as walking, bicycling, brushing teeth, speaking and laughing, and workshop activities such as sawing and drilling that have characteristic motions and/or sounds. An advantage of this approach is that it does not require environment embedded sensors. Also, users can easily turn off their wearable devices when they want to preserve their privacy. They are well-suited to collecting data on daily activity patterns over an extended period of time as they can be integrated into clothing, or worn as wearable devices. Since they are attached to the subjects they are monitoring, wearable sensors can therefore measure physiological parameters which may not be measurable using ambient sensors. Although body attached sensors is promising to identify primitive sequences of movements such as walking and running, it is difficult to identify goal-oriented activities (e.g. making tea and taking medicine). In addition, the design of wearable systems is complicated by size, weight, and power consumption requirements.

OUAR

OUAR is the third type of activity recognition approach. It recognise the person's activity by analyzing the person's interactions with instrumented objects. Relatively, this approach is newer than VSAR and WSAR. It is proposed to overcome the limitations of VSAR and WSAR. The fact is many people are uncomfortable living with cameras. Moreover, we find

that people are often unwilling or forget to wear sensors. However, OUAR also has some drawbacks. For extensive and detailed recognition, OUAR requires a large number of objects to be attached with sensors. Sometimes it is either infeasible or too expensive. Cost of sensors and sensor acceptance are pivotal issues, especially in the home.

Table 2.1 *Main strenghts and limitations of activity recognition sensors based on the deployed sensing technology*

Sensor	Strengths	Limitations
VSAR	1) Includes richest inforamtion 2) Environmental setting up is easy	1) Violating privacy 2) Sensitive to light condition and viewpoint variation 3) Computationally intensive
PSAR		
WSAR	1) Environmental setting up is not required 2) Good for application which require explicit motion analysis	1) Wearing sensors is a burden 2) Unable to separate similar actions (e.g. making tea)
OUAR	1) Do not violate privacy 2) Good at recognize goal-oriented activities	1) Recognizable activities need to be related with objects 2) Environmental setting uo needs more effort than VSAR and WSAR

2.2.6 SUMMARY

In this section, applications and researches that employed activity recognition systems (ARSs) are reviewed. Based on the sensing technology, an ARS is divided in three categories, namely, video sensor based AR (VSAR), wearable sensors based AR (WSAR) and object usage based AR (OUAR). Strengths and limitations of ARSs with different sensing technologies are discussed and compared, then they are summarized in Table 2.1. To summarise, it seems like a lot of progresses have been made in achieving a similar goal with a lot of success in employing different approaches, methodologies and implementations. Smart home systems are widely being used by elderly and seems to be an success driven industry. The research into rehabilitation is something that still remains demanding. It seems like employing previous

successful research methods and integrate them into one could potentially help towards the industry and further researches to come.

2.3 AN OVERVIEW OF HMM BASED SYSTEMS

HMMs offer a great solution when there is a need to describe a set of time-varying feature sequences where there is no one to one correspondence between states and the system outputs. An observation sequence $O = \{o_1, o_2, ..., o_T\}$ is produced by an HMM, through a hidden state sequence of form $Q = \{q_1, q_2, ..., q_T\}$ at each given point in time t.

An example HMM with 5 states including non-emitting entry and the exit states is shown in Figure 2.1. Here, between every two states i and j the corresponding transition probabilities is denoted by a_{ij}, and output probability for state i is denoted by $b_i(o_t)$.

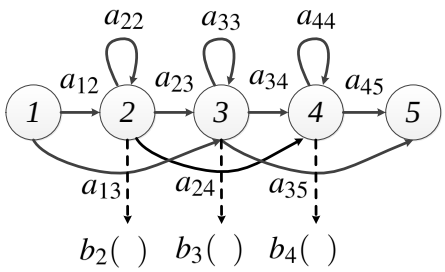


Fig. 2.1 *A 5-state HMM with three emitting and two non-emitting states [1]*

(1) A state transition probability matrix $A = \{a_{ij}\}$, (2) initial state probability vector π, and (3) an observation Probability Density Function (PDF) for each HMM state, altogether form the HMM parameters for a given HMM λ [114, 53, 115, 116]. In a system with N-state HMMs $S = \{s_1, ..., s_N\}$, an initial state occupation probability vector for state i corresponds to the probability that state i is the first state in a set of states (Equation 2.1). A transition probability matrix, A, describes the probability of a transition from state i to state j where

$i, j \in \{1,...,N\}$ (Equation 2.2). Output PDFs of form $b_i(o_t)$ describe the observation PDF for a given HMM state i where $i \in \{1,...,N\}$. To describe an output PDF either a Gaussian mixture distribution or a Deep neural network can be used among other techniques.

$$\pi_i = p(q_0 = s_i) \quad 1 \leq i \leq N \tag{2.1}$$

$$a_{ij} = p(q_t = s_j | q_{t-1} = s_i) \tag{2.2}$$

2.4 HMM BASED MODELLING USING GMMS

In GMM-HMM based systems each different state is modelled by a different GMM, and the output PDFs are denoted as shown by Equation 2.3, where $m = 1,...,M$ represents the number of Gaussian mixture components, the weight, mean vector, and covariance matrix for each mixture component, are shown by $c_{i,m}$, $\mu_{i,m}$, and $\Sigma_{i,m}$ respectively, where $\sum_{m=1}^{M} c_{i,m} = 1$.

$$\begin{aligned} b_i(o_t) &= \sum_{m=1}^{M} c_{i,m} \mathcal{N}(o_t | \mu_{i,m}, \Sigma_{i,m}) \\ \mathcal{N}(o_t | \mu_{i,m}, \Sigma_{i,m}) &= \frac{1}{(2\pi)^{D/2} |\Sigma_{i,m}|^{1/2}} exp[-\frac{1}{2}(o_t - \mu_{i,m})^T \Sigma_{i,m}^{-1} (o_t - \mu_{i,m})] \end{aligned} \tag{2.3}$$

In the GMM-HMM systems the Viterbi algorithm is used to provide a state level forced alignment of the training data.

2.5 HMM BASED MODELLING USING DNNS

In DNN based systems a single DNN can be trained to provide an estimate for all the state output probabilities [117]. In order to train a DNN-HMM based system, firstly a GMM-

HMM system is trained. It provides the HMM transition probabilities for the DNN based system. Then, the DNN parameters are initialized during the pre-training stage. Next, all the previously initialized DNN parameters are fine-tuned and updated during the training stage [118]. To achieve state-frame alignments in DNN based systems, each state, can be mapped to a state ID. The features and each state ID can form a pair hich can be used to train the DNN system. After this stage the DNN-HMM system parameters are updated During each system re-training phase an updated state-frame alignment is produced from the fine-tuned DNN and the transition probabilities are updated correspondingly. The DNN-HMM system is re-trained till no more improvement is observed on the accuracy of recognition for a set of held out data.

2.5.1 DEEP NEURAL NETWORKS BACKGROUND

A simple form of a neural network is a multi-layer perceptron. It consists of an input layer containing the input features, an output layer which sends the output, and at least one hidden layer in between the input and output layers. There are a set of weighted connections between the input layer, hidden layers, and the output layer as shown in Figure 2.2.

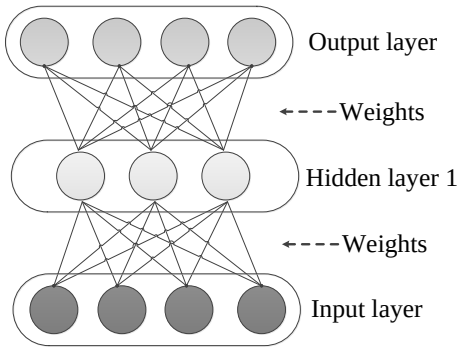


Fig. 2.2 *A multi-layer perceptron with one hidden layer*

During neural network's training phase, the inputs are propagated to the next consecutive layer, such that the outputs of the units in each layer will be the input of the next layer's units.

For each hidden unit j the weighted summation of its inputs x_j is calculated as shown in Equation 2.4, and then passed through an activation function.

$$x_j = b_j + \sum_i y_j w_{ij} \tag{2.4}$$

Here the bias of unit j, is denoted by b_j, and the weight of the connection between unit j from the current layer and unit i from the layer below is shown by w_{ij}.

A sigmoid function, $\sigma(.)$, is one of the most popular action functions; because it is continuous and differentiable, its derivative is very fast to compute (as opposed to other functions with similar properties like tanh), and has a limited range (from 0 to 1, exclusive). sigmoid function maps input of the unit j from the layer below, x_j, to the scalar state, y_j that it sends to the layer above as shown in Equation 2.5.

$$y_j = \sigma(x_j) = \frac{1}{1 + exp(-x_j)} \tag{2.5}$$

The hidden layers enable the network to successfully address the non-linearly separable classification problems [119] through bringing higher order of complexity to the system's architecture. In order to solve a multi-class classification problem, where k is an index over all classes the class probability values, p_j, are computed by applying a 'softmax' function to the total input of the output unit, x_j, as shown in Equation 2.6. Here, p_j represents the probability that the network's input feature vector belongs to the certain class [2, 117].

$$p_j = softmax(x_j) = \frac{exp(x_j)}{\sum_k exp(x_k)} \tag{2.6}$$

2.5.2 FINE-TUNING THE DEEP NEURAL NETWORKS

DNN's can be discriminatively trained (fine-tuned) by applying the back-propagation algorithm [117, 53]. During the back-propagation stage the derivatives of the cost function are computed and the weights are updated through minimizing the error between the predicted outputs and the target outputs of the system [53]. Backward propagation of errors is based on the Stochastic Gradient Descent (SGD) which is an standard approach for minimizing the network's error (fine-tuning) [120]. The cost function, C, is coupled with a softmax output layer and it is the cross-entropy between the outputs of the softmax, p, and target probabilities, d, as represented by Equation 2.7. To train the DNN classifier the target probabilities, $d \in \{1,0\}$, are required from a labelled training data [2, 121, 117].

$$C = -\sum_{j} d_j log(p_j) \tag{2.7}$$

Here j is the number of neurones in the output layer. To be able to do error back-propagation on large training sets, the model parameters are adjusted using the mini-batch SGD that estimates the gradient based on a small batch (rather than the whole training set) of randomly selected training samples [120, 2, 121, 117].

The update formula for weights using mini-batch SGD algorithm is denoted by Equation 2.8. The update formula for biases can be computed in a similar manner by treating biases as weights on connections coming from units with a state of 1. Here, the momentum and the learning rate are denoted by α and ε respectively. A momentum, $0 < \alpha < 1$, smooths the gradient computed for mini-batch t, and thereby helps with getting out of local optima due to its noisy estimation of gradient.

$$\Delta W_{ij}(t) = \alpha \Delta W_{ij}(t-1) - \varepsilon \frac{\partial C}{\partial w_{ij}(t)} \tag{2.8}$$

2.5.3 PRE-TRAINING OF DEEP NEURAL NETWORKS

Pre-training helps with optimization of the cost function during the fine-tuning phase and results in generalization error reduction [122–124]. Restricted Boltzmann Machines (RBMs) are a type of a neural network. During the pre-training phase the DNN parameters are initialised using the RBMs.

An RBM consists of a hidden layer comprising hidden units which take binary values , $h \in \{0,1\}$, and a visible layer comprising visible units , $v \in \{R\}$, which are modelled with a Gaussian distribution and take real values. A set of weights W are mapped to the connections between the hidden units and visible units. A joint configuration of the units across visible and hidden layers , (v,h), of an RBM has an energy associated to them, $E(v,h)$, as shown by Equation 2.9.

$$E(v,h) = -\sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{2.9}$$

Here v_i, h_j are the binary states of visible unit i and hidden unit j, and ai, b_j represent their corresponding bias values and w_{ij} represents the weight between units i and j.

A probability is allocated to all possible visible and a hidden units pair via the energy function described in Equation 2.9. The probability that the network assigns to a joint configuration of the units across visible and hidden layers, is given by Equation 2.10, where Z is referred as the partition function.

$$p(v,h) = \frac{e^{-E(v,h)}}{Z} \tag{2.10}$$

$$Z = \sum_{v,h} e^{-E(v,h)} \tag{2.11}$$

Pre-training aims to design an architecture which the hidden layer of each RBM acts as the visible layer for the next RBM, and this enables the final system's structure to be good at modeling the input data's structure.

After the pre-training stage, the stacked RBMs can be used as a much better initialization point for the fine-tuning stage during which the weights found in pre-training stage are adjusted during the back-propagation stage.

The fine-tuning and pre-training procedure for a DNN system with a softmax output layer can be summarised as shown in in Figure 2.3.

This process starts by pre-training of the DNN through training a stack of RBMs on the input features such that the first visible layer v is initialised with input observation data, and training of RBM continues such that the output layer of each RBM acts as input layer of the next RBM (part a). Part (b), shows how a stack of RBMs form a network. The weight and bias values learnt during the pre-training phase, will be used for DNN parameter initialisation. Part (c), shows a softmax layer is added on top of the stacked RBMs. The softmax layer contains all the training targets, and it enables the DNN discriminative training. In the fine-tuning stage the network weight and bias values are adjusted in a layer-by-layer manner by applying the back-propagation algorithm.

2.6 DECODING HMM STATE SEQUENCES (VITERBI ALGORITHM)

For an observation sequence of form $O = \{o_1, ..., o_T\}$, the Viterbi algorithm computes the probability of observation given the model, $p(O|\lambda)$ through finding the single most likely sequence of hidden states, $Q = \{q_1, ..., q_T\}$. Here, the maximum probability over all partial state sequences ending in state i at time t cabn be computed by Equation 2.12. In order to

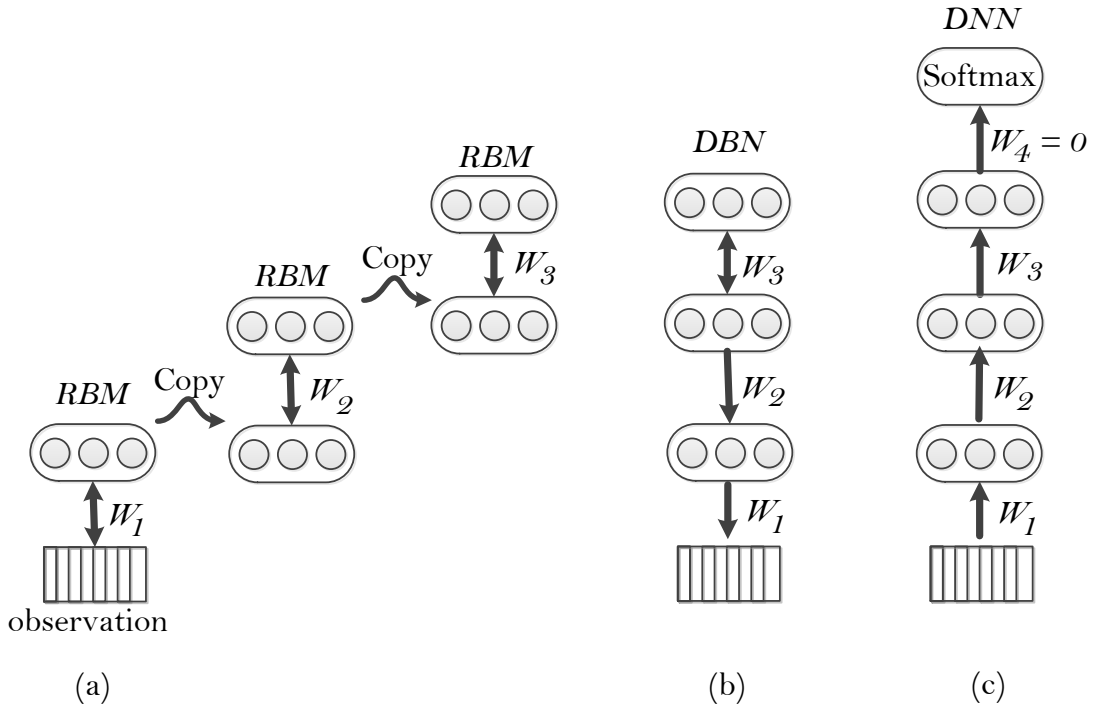


Fig. 2.3 *A layer-wise pre-training using a stack of RBMs [2]*

keeps track of the previous state with the highest probability values a place holder, $\psi_t(j)$, is used. The best state sequence can be found after the initialization stage, recursively through a sequence of recursive stages [114, 53].

$$\delta_t(i) = max_{q_1,q_2,...,q_{t-1}} p(q_1,q_2,...,q_t;q_t = s_i;o_1,o_2,...,o_t|\lambda) \tag{2.12}$$

Viterbi algorithm for decoding, which can be applied to the GMM based systems. (1) Firstly, an initial value is given to system variables $\delta_t(i)$ and $\psi_1(i)$ at state i and time $t = 1$ as shown by Equation 2.13. (2) Then array $\psi_t(j)$ is recursively estimated which keeps track of the most likely previous state with highest probability as shown by Equation 5.1. (3) At the end of the observation sequence $O = o_1,...,o_T$, the best score p^* and the probability of state q_T^* is computed as shown by Equation 2.15. (4) At the end of the observation sequence,

after we finished backtracking through the most likely set of states q_t^*, the optimal HMM state sequence, Q^*, is delivered as denoted by Equation 2.16.

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(o_1), \;\; 1 \leq i \leq N \\ \psi_1(i) &= 0 \end{aligned} \tag{2.13}$$

$$\begin{aligned} \delta_t(j) &= max_{1 \leq i \leq N}[\delta_{t-1}(i)a_{ij}]b_j(o_t), \;\; 1 \leq j \leq N, 2 \leq t \leq T \\ \psi_t(j) &= argmax_{1 \leq i \leq N}[\delta_{t-1}(i)a_{ij}], \;\; 1 \leq j \leq N, 2 \leq t \leq T \end{aligned} \tag{2.14}$$

$$\begin{aligned} p^* &= max_{1 \leq i \leq N}[\delta_T(i)] \\ q_T^* &= argmax_{1 \leq i \leq N}[\delta_T(i)] \end{aligned} \tag{2.15}$$

$$\begin{aligned} q_t^* &= \psi_{t+1}[q_{t+1}^*], \;\; t = T-1, T-2, ..., 1 \\ The \;\; best \;\; state \;\; sequence: \;\; Q^* &= q_1^*, q_2^*, ..., q_T^* \end{aligned} \tag{2.16}$$

In order to carry out decoding in DNN-HMM systems, the likelihood, $p(o_t|q_t)$ has to be computed from the posterior probability produced by a DNN, $p(q_t = s|o_t)$, as shown by Equation 2.17. The posterior probability $p(q_t|o_t)$ can be estimated from a DNN directly. Here, $p(s)$represents prior probability of the state [117, 125].

$$p(o_t|q_t = s) = p(q_t = s|o_t)p(o_t)/p(s) \tag{2.17}$$

CHAPTER 3

TEA-MAKING DATABASE

3.1 INTRODUCTION

The objective of this work is to provide an automatic action recognition(AAR) system to detect the patient's progress through performing an activity of daily living (ADL). The initial ADL task was chosen to be "making a cup of tea".

A formal analysis of tea-making task, using Hierarchical Task Analysis (HTA) is presented in [126]. The analysed task in [126] is "making two cups of tea", one with lemon and sugar and one with milk and sweetener. This is slightly different from CogWatch tea-making tasks (Section 1.1.1) in which only milk and sugar are used. The analysis from [126] was modified by the psychologists working on CogWatch in response to practical experience observing stroke survivors making tea. Also, two additional sub-goals, corresponding to commonly occurring errors of toying with the kettle and toying with the milk container, were added. The final tea-making task tree, which was used in the CogWatch system, is shown in figure 3.1. In this figure, it is illustrated that the task of "making a cup of tea" is composed of seven basic actions (they are referred as sub-goals in the thesis) that need to be completed for

the cup of tea to be made (coloured in green). Each of these basic actions (sub-goals) can be defined by more detailed sub-actions, which will also have to be properly completed for their corresponding higher-level actions to be succeeded. The definition of all sub-goals are explained in Section 3.2 of this chapter.

In order to build statistical models for the real-time HMM-based AAR system, the tea-making data base is collected by recording data from sensors (Chapter 4) while a control group of participants perform tea-making sub-goals. Recordings are made at three sites: the Technical University of Munich (TUM), Technical University of Madrid (UPM), and University of Birmingham (UOB). A total of 38 participants, completed multiple individual sub-goals and full tea-making trials. The trial instructions and tea-making table setup that were used to collect the recordings are explained in section 3.3. Information about the participants and the details of the recordings can be found in section 3.4.

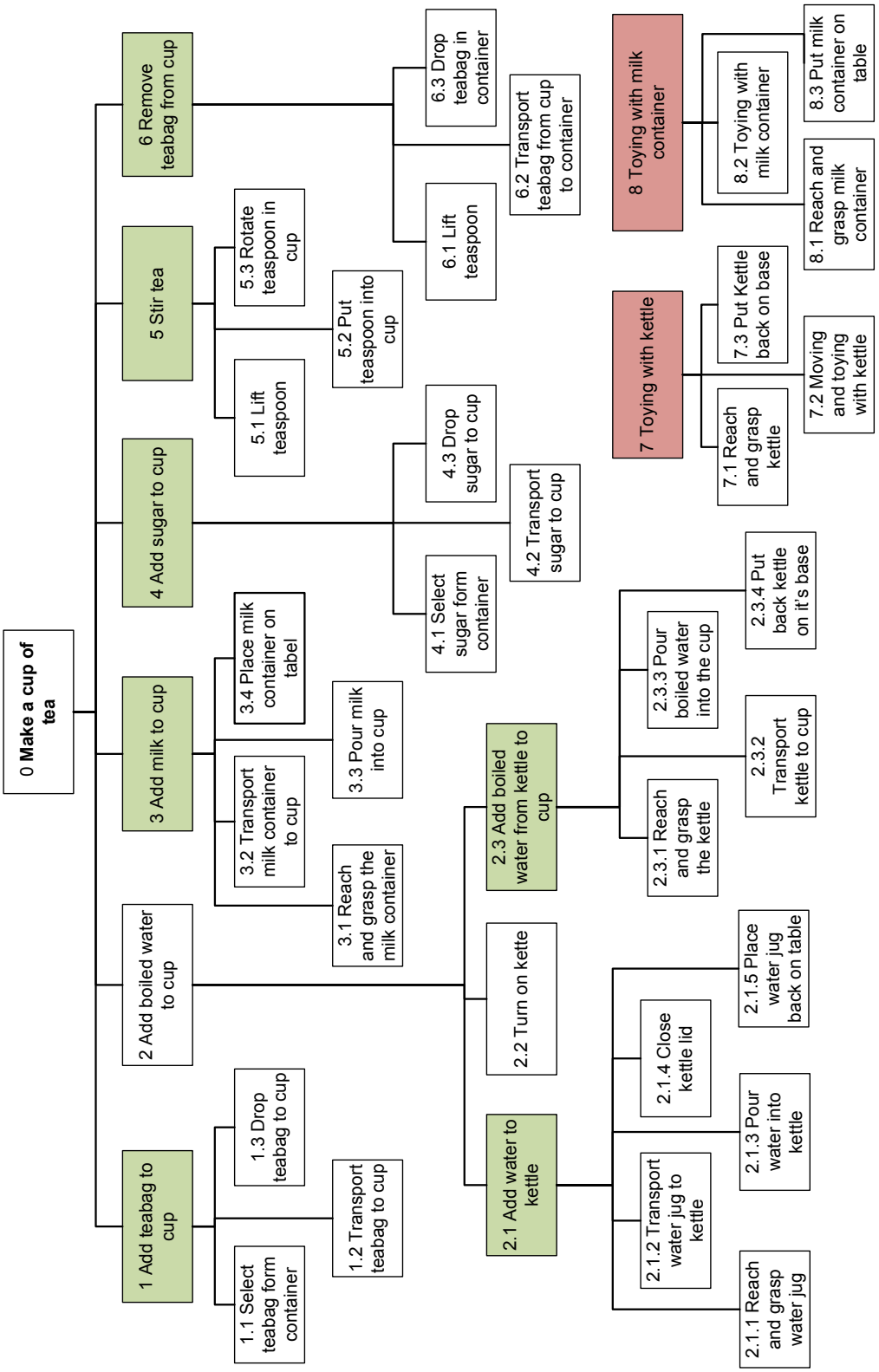


Fig. 3.1 *Hirichical tree Analysis of the tea-making*

3.2 ANALYSIS OF TEA-MAKING

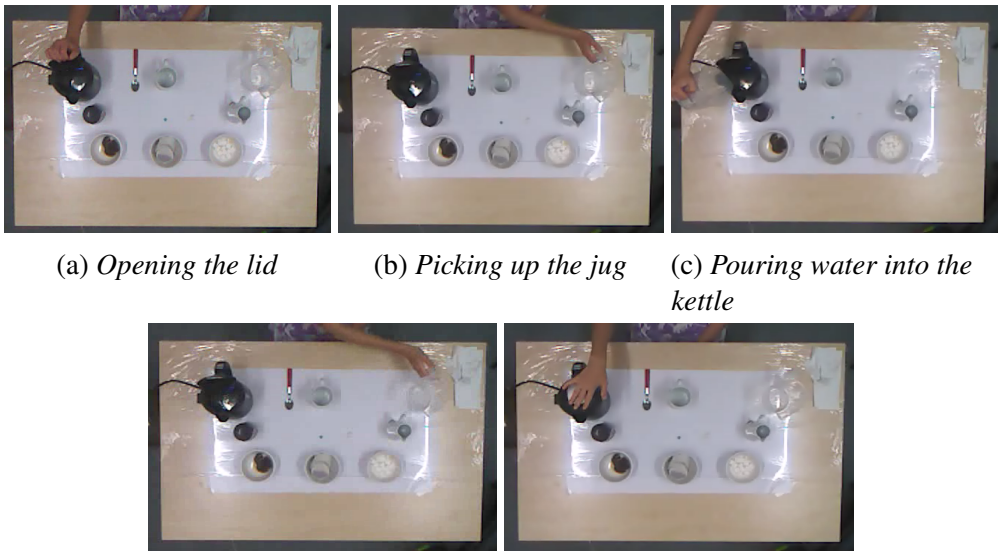
The tea-making task is defined with eight sub-goals in total, including six primary and two optional sub-goals. Six sub-goals that are obligatory to successfully making a complete cup of tea are: filling kettle with water, boiling the water, pouring the boiled water into the mug, adding teabag to cup, stirring (mixing the boiled water with tea) and removing the used teabag. Two extra optional sub-goals are adding milk and adding sugar into the mug. Executing a different combination of these sub-goals can provide four types of tea; namely "black tea", "black tea with sugar", "tea with milk" and "white tea with sugar". The definition of the sub-goals are as follow:

1. **Fill kettle with water**

   In this thesis, the process of filling the kettle with water from the provided pre-filled jug, is called the "Fill Kettle" sub-goal. This sub-goal needs to be executed before the participant can boil the water inside the kettle. Filling the kettle can be further broken down into sub-actions such as, opening the kettle lid, picking up the jug, pouring water into the kettle, putting down the jug and closing the kettle lid. An example of completing this sub-goal has been illustrated in Figure 3.2.

2. **Boiling Water**

   Boiling water inside the kettle, which we call the "Boil Water" sub-goal in this thesis, can be accomplished by pressing the button, which is placed on the handle of the kettle, and waiting for the alarm that goes off when the pre-filled water has reached fully boiled. This is a relatively long period of waiting; which is why, often participants tend to perform other sub-goals during this time, for example adding sugar or tea bag.



(a) *Opening the lid* (b) *Picking up the jug* (c) *Pouring water into the kettle*

(d) *Putting down the jug* (e) *Closing the lid*

Fig. 3.2 *"Fill Kettle" sub-goal - Process of filling kettle with water*

3. Pour boiled water into the mug

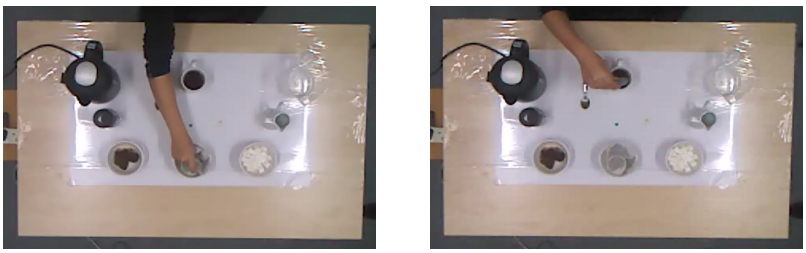
The "Pour Kettle" sub-goal is the short name chosen for the process of pouring boiled water from the kettle into the mug. This sub-goal consists of consecutive low-level sub-actions, such as picking up the kettle, placing it above the mug, tilting it to pour water and putting down the kettle. These sub-actions are displayed in Figure 3.3.

(a) *Picking up the kettle* (b) *Tilting and pouring water into the mug* (c) *Putting down the kettle*

Fig. 3.3 *"Pour Kettle" sub-goal - Sub-actions invloved in pouring water from kettle into the mug*

4. Adding a tea bag to the mug

The “Add Teabag” sub-goal is the process of adding a tea bag to the mug. This sub-goal can be performed before or after “Pour Kettle”; thus, the mug can be empty or filled with boiled water when adding the tea bag to it. In order to perform the “add teabag” sub-goal, the participant needs to pick a tea bag from the specific container filled with tea bags and place it into the mug. These two steps have been projected in Figure 3.4.



(a) *Selecting teabags from container*

(b) *Placing teabags inside the mug*

Fig. 3.4 *“Add Teabag” sub-goal - Proccess of adding tea bags into the mug*

5. Remove tea bag to rubbish

Removing the used teabag from the mug to the rubbish container is called the “Remove Teabag” sub-goal and will be executed after boiled water is mixed with the teabag. The “Remove Teabag” sub-goal starts by taking out the used tea bag from the mug, using a spoon or the string attached to the tea bag. In the case of using the spoon, optional stirring is possible, while removing the tea bag. “Remove Teabag” sub-goal finishes by placing the used teabag inside the rubbish container on the table. These steps are demonstrated in Figure 3.5, using images from one of the recording sessions.

6. Adding milk into the mug

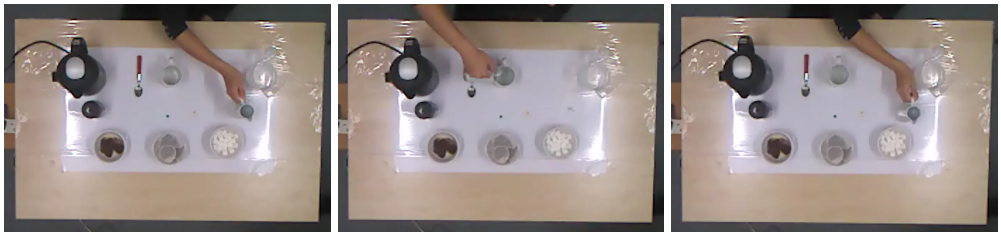
Adding milk into the mug is a complimentary sub-goal for making teas with milk, and it is called ”Add Milk”. Participants can complete this sub-goal by picking up the pre-filled milk container, moving it over the mug, tilting the container to pour the milk

(a) *Picking up the spoon* (b) *Removing the tea bag from the mug* (c) *Put used tea bag into the bin*

(d) Putting down the spoon

Fig. 3.5 *"Remove Tea bag" sub-goal - the process of removing tea bag from the mug*

and putting down the milk container. The photographs in Figure 3.6 are snapshots from a recording session, and illustrate the required steps.



(a) *Picking up the milk container* (b) *Tilting the milk container, and pouring milk into the mug* (c) *Putting down the milk container*

Fig. 3.6 *"Add Milk" sub-goal - Process of adding milk into the mug*

7. Adding sugar cubes into the mug

"Add Sugar" is an optional sub-goal and can happen at any time during the tea-making task. This sub-goal consists of two sub-actions, selecting a sugar cube from the container and putting it inside the mug. Figure 3.7 illustrates the performing of these sub-actions.

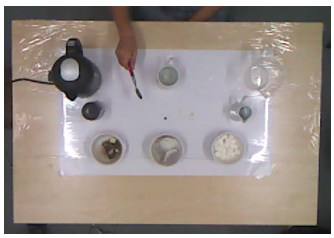
Fig. 3.7 *"Add Sugar" sub-goal - Proccess of adding sugar cubes into the mug*

(a) *Selecting sugar cubes*

(b) *Placing sugar cubes inside the mug*

8. Stir

This sub-goal is the process of stirring the contents of the mug with a spoon. Stirring can happen multiple times during the tea-making task, in order to mix the tea, milk and sugar with boiled water. For completion of this sub-goal the participant needs to pick up the spoon, place it inside the mug, stir the contents of the mug and put down the spoon (Figure 3.8).



(a) *Picking up the spoon*

(b) *Stirring*

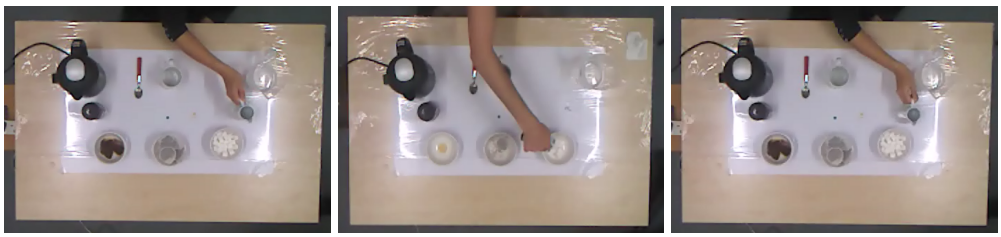
(c) *Putting down the spoon*

Fig. 3.8 *"Stir" sub-goal - Process of stirring contents of the mug*

For better recognition of the "Add Milk" and "Pour Kettle" sub-goals, two auxiliary sub-goals of toying with the milk container and kettle are proposed, respectively. Also, detection of these sub-goal is used as an error feedback to patients, especially toying with the kettle which is a prompt for a dangerous mistake. The definition of these sub-goals is as follow:

- **Toying with the milk container**

  The "Toy Milk" sub-goal is defined as using the milk container (picking up and putting down) and not pouring milk into the mug; this can include tilting the milk container and spilling the milk on the table, pouring milk into objects other than the mug (e.g. water jug, sugar and tea bag container) or simply fiddling with the milk container in random moves. An example of the "Toy Milk" sub-goal is when the milk is poured into sugar container by mistake as shown in Figure 3.9.



(a) *Picking up the milk container* (b) *Pouring milk into the sugar container* (c) *Putting down the milk container*

Fig. 3.9 *"Toy Milk" sub-goal - Pouring milk into the sugar container by mistake*

- **Toying with the kettle**

  Toying with kettle can be very dangerous, especially when it is filled with boiled water and can be spilt on the person. The "Toy Kettle" sub-goal is defined as any fiddling with the kettle apart from pouring water into the mug. An example of pouring water from the kettle into the tea bag container is shown in Figure 3.10.

(a) *Picking up the kettle* (b) *Pouring water into the tea bag container* (c) *Putting down the kettle*

Fig. 3.10 *Example of the "Toy Kettle" sub-goal where boiled water is poured from the kettle into the tea bag container*

3.3 DATA COLLECTION PROCEDURE AND TABLE SETUP

In all recording sites, a C# program is used to record the data from all the sensors with time labels to be used for synchronisation. Data is collected from four coasters attached to the mug, milk container, kettle base and body; also data from the Kinect camera and depth sensor is captured. The software associates a special ID with each recording, including the name of the session and the information about the participant.

3.3.1 TEA-MAKING TABLE SETUP

The Kinect must be located above the table, pointing down at the table, and so that the line between the centre of the camera lens and the centre of the table is perpendicular to the plane of the table. The distance between the Kinect and the centre of the table should not exceed 1.8 metres. Although the 2D resolution increases as the Kinect approaches the surface of the table, the depth image has an optimal resolution at 1.8 metres, with an accuracy of approximately 2 millimetres at that distance [127]. All edges of the table must fit in the captured pictures from the Kinect camera. The size of the table is not important as long as it fits in the picture. However, the position of the objects on the table and the distance

between them are important. Thus, a template with the position of all the objects drawn on it, is printed to be used at all sites. The template of the table setup is illustrated in Figure 3.11.

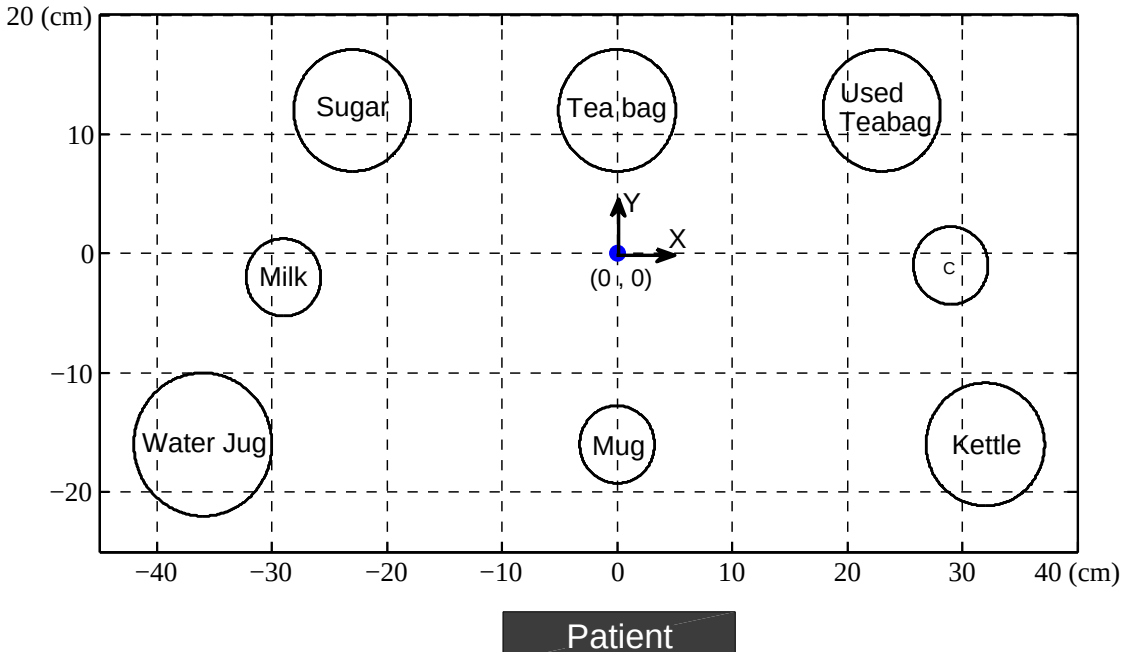
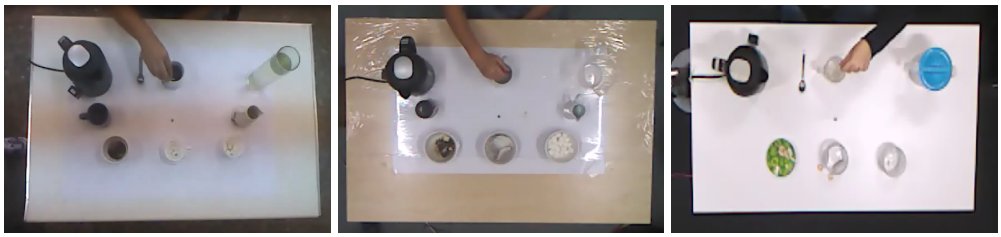


Fig. 3.11 *Tea-making setup - Distances are in centimetres and the reference point is marked in the centre of the template (0,0)*

For calibration of the table, the middle point of the template (marked as (0,0)) must be aligned with the middle of the picture captured from the Kinect camera, which has been marked in the recording software. Objects are arranged for convenience of access for participants when they are seated, while ensuring a good separation between each. Figure 3.12, shows photographs of the tea-making table in three sites captured from the Kinect camera video.



(a) Table setup in UPM (b) Table setup in UOB (c) Table setup in TUM

Fig. 3.12 photographs of the table setups in three sites, captured from the Kinect camera

3.3.2 ISOLATED AND FULL-TRIAL RECORDINGS' INSTRUCTIONS

The recording session for each participant is divided into two parts: recording trials of performing sub-goals alone in isolation and trials of making a complete cup of tea.

- **Isolated recordings of sub-goal**

  Each participant was asked to perform a sub-goal described in section 3.2, after the recording is started. The trial finishes when the sub-goal is successfully executed. So, the isolated recording of a sub-goal, starts and ends with a possible silence with the execution of that specific sub-goal in-between. This process is repeated for all sub-goals between 2-5 times.

- **Full-trial recordings of making a cup of tea**

  In order to collect full-trial recordings, participants are asked to make a cup of tea, as they are used to making tea in a normal situation in their house. Participants can choose between four types of tea which are described in section 3.2. A full-trial recording consists of some primary and optional sub-goals; the start and finish time of the sub-goals are manually labelled using the trial video captured from the Kinect camera as guidance.

3.4 DATABASE DETAILS

The control groups were composed of a total of 38 participants, aged between 20 and 50 years, attended recording sessions of the tea-making data set. Among all participants, 25 of them are right-handed and the remaining are left-handed. In total, there were 1510 trials of isolated sub-goals consisting of 4.14 hours of recordings from the sensors and 100 full trials of making a cup of tea comprising 2.15 hours of sensor data recording. The number of trials, the total duration of the recordings, and the average execution time of each sub-goal, are presented in Table 3.1. The "Fill Kettle" and "Add Teabag" are the longest and shortest sub-goals with an average execution time of 14.23 and 6.32 seconds, respectively.

Table 3.1 *Detailed information about the total number and duration of traials in the tea-making dataset*

Recordings	Trials	Dur. (minutes)	Avg. (Second)	Variance
Pour Kettle	144	29.74	12.39	12.3
Add Milk	91	17.03	11.23	11.2
Fill Kettle	184	43.66	14.23	14.2
Add Sugar	207	21.89	6.34	5.4
Add Teabag	223	23.52	6.32	6.0
Remove Teabag	154	21.81	8.49	13.2
Stir	187	32.07	10.28	10.2
Toy with Kettle	186	36.13	11.65	8.6
Toy with milk	134	22.78	10.20	4.0
Full trial	100	129.55	77.73	256.3

The execution time of an individual sub-goal in a full-trial recording, is manually labelled using the video recording of the session captured by the Kinect RGB camera. Unfortunately,it was not possible to label the "Boil Water" sub-goals from the videos. The total number of times that each sub-goal is performed in all full-trial recordings is different and it has been broken down in Table 3.2.

Table 3.2 Total number of sub-goals in all 100 full-trial recordings

Sub-goal	Trials	Sub-goal	Trials
Pour kettle	95	Stir	134
Add milk	56	Remove teabag	95
Add sugar	90	Fill kettle	95
Add teabag	94	Boil Water	0

3.4.1 NAME OF RECORDINGS IN THE DATA-SET

Recordings in the dataset are named in this format:

"recording name - site name - C - L\R - participant ID - recording number"

The "recording name" can be the name of any sub-goal or "FullTrial". The "site name" is TUM, UPM or UOB depending where the trial is recorded; C stands for control or F can be used for future trials recorded from patients; "L" and "R" are used to specify whether the participant is left or right-handed. For example, "AddSugar-TUM-C-L-06-01" trial is an isolated recording of the "Add Sugar" sub-goal collected from the sixth participant, who is a left-handed control, in TUM university .

3.5 SUMMARY

In this chapter, the high-level tea-making ADL was broken down to low-level sub-goals. The process of collecting the tea-making database was explained. The recording sessions are carried out by giving structures about how to perform individual sub-goals, and asking 38 participants to perform each sub-goal in isolation, and make complete cups of tea (full-trials).

Approximate 4 hours of the isolated sub-goals and 2 hours of the full-trials were recorded. The full-trials were annotated using the video recordings of sessions. The tea-making dataset is used in the rest of the thesis for training the models and producing detection results.

CHAPTER 4

SENSORS AND FEATURE EXTRACTION

4.1 INTRODUCTION

The primary use of the developed system in this work is for cognitive rehabilitation of patients with Apraxia or Action Disorganization Syndrome (AADS). Due to the high level of stress for some of these patients, it is important that the system operates as hidden as possible to the eye of the patients. One way to achieve this, is to use sensors installed in the environment and attached to the objects, rather than wearable on-body sensors. Attaching sensors to the objects without changing their appearances and functions is useful for everyday use in real home setup, and it is more comfortable for patients to work with as they are used to. Also instrumented objects (IOs) can be considered as components of an internet of things (IOT). The IOT will become more important in the future, therefore potentially more support of IOs via the IOT (e.g. better sensors, better wireless communications). It may be that IOs become commonplace in the IOT and the technologies developed in this project will exploit this opportunity.

In this chapter, different sensors that were used to record and monitor the user's activities throughout the completion of the ADL are introduced, such as the CogWatch instrumented coasters (CIC) and Kinect^{TM}. The CIC can be attached to the object and transfer data to the main computer wireless; and it consists of three force sensitive resistors (FSRs), a 3D accelerometer, a Bluetooth module and a microprocessor.

Furthermore, in the last part of the chapter, some basic signal processing and feature extraction methods are applied to the data captured from the sensors. Processed data are used in the action recognition system to detect the sub-goals that make up the tea-making task.

4.2 SENSORIZED OBJECTS

The instrumented coaster is designed to provide data that can be used to monitor and recognise interactions with objects which are used in daily tasks; such as mugs, jugs and kettles. The aim of the design is to integrate sensors with the objects themselves without interfering with their use or significantly modifying their appearance; and without requiring any device to be fitted to the person using them. To achieve this aim, a wireless device that fits discreetly underneath the object was created. This device is called the CogWatch Instrumented Coaster (CIC). The sensors contained in a CIC include a 3-axis accelerometer and three force sensitive resistors, together with a Bluetooth transmitter to send the data to the host computer. The entire circuit is powered by a 260mAh 3.7V rechargeable LiPo battery, which provides a minimum of 2 hours operation. The battery is charged using a separate circuit that can be USB or mains powered (Figure 4.1(a)).

For the rest of the thesis, the two CICs placed underneath the mug and milk container are called Mug-CIC and Milk-CIC, respectively. Also, as the kettle body is separated from the base, two separate circuits are designed to transmit data from accelerometer sensors attached

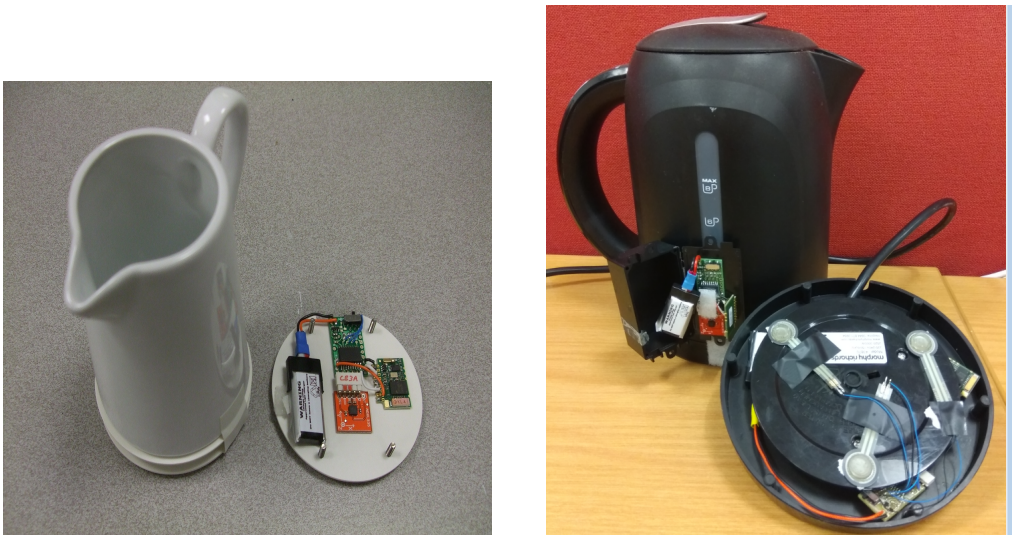
to the kettle body and FSRs sensors under the kettle base (Figure 4.1(b)). This choice of sensors was made on the basis of three criteria:

- **Capture of data relevant to the modelling of ADL**: the models of tea-making ADL (both in terms of predictive models and in terms of experimental design) involve the movement of objects and their change in state. Changes in state are defined in terms of the addition or removal of materials, such as water and milk. Thus, the minimal set of sensors required for these models would record motion (hence the accelerometers) and the gravitational force to the object (hence the force sensitive resistors). For detection of sub-goals in other ADLs (e.g. tooth brushing), additional sensing capability might be required for the modelling of the sub-goals.

- **Size of the CogWatch Coaster**: although there are many other sensors that could be added, doing so would increase the size of the device making it obtrusive to use. The form factor for the original CogWatch Coaster was defined by the circumference of a coffee mug; the design fits exactly under the base of a mug. A further design was made to fit under the kettle.

- **Power and Battery Life Considerations**: the selection of the sensors was based on reasonable power efficiency relative to the cost of the sensors and their processing capability.

The instrumented coaster in this research is controlled by a Microchip dsPIC30F3012 microcontroller. This microcontroller has an integrated 12 bit analogue to digital converter (ADC) that is used for digitising the sensor readings. The ARF7044 Bluetooth module is used, to transmit the sensor data to a computer via a Bluetooth wireless connection. This module complies with the V 2.0 Bluetooth ® standard and use universal asynchronous receiver/transmitter (UART) serial data communication; it has up to 20 metres communication range in the worldwide 2.45 GHz frequency band. The microcontroller is programmed to



(a) (b)

Fig. 4.1 *(a) milk-container fitted with a CogWatch Instrumented Coaster (CIC) and an 'open' CIC, showing the accelerometer, PIC, Bluetooth module and battery (b) Accelerometer circuit attached to the kettle-body and FSR sensors planted under kettle-base*

digitise, compress and prepare the sensor data and manage the transmission of the data via the Bluetooth module. The data are buffered on the microcontroller so that they can be re-transmitted (avoiding data loss) if the wireless connection is interrupted for a short period.

No gyroscopic sensors is included in our instrumented object, because for recognition of sub-goals in the tea-making task, the accelerometer sensor data is informative enough, especially they are very suitable for tilt sensing. On the other hand, high battery life is crucial for instrumented objects, and more sensors means more power consumption.

4.2.1 FORCE SENSITIVE RESISTORS

Force sensitive resistors (FSRs) are used to measure the gravity force applied to the instrumented object when placed on the table, if the object is lifted the force is zero. As the force is applied to an FSR the electrical resistance decreases. Using a voltage divider circuit the resistance is changed into a voltage that can be read by the microcontroller ADC. The FSRs

are less than 20 mm wide and less than 1 mm thick so they are well suited to the size of the instrumented coaster. The raw sensor readings cannot directly be added up to obtain a single unified reading from the three sensors due to the non-linear nature of the voltage divider and the resistance output of the sensors. The non-linear effect of the voltage divider can be removed using the voltage divider - Equation 4.1 which results in the resistance of the an FSR (R_{FSR}). R_{fixed} is the resistance of the resistor used in series with the FSR sensor in the voltage divider circuit, V_{in} is the input voltage of the microcontroller and V_{out} is the analogue reading from the FSR sensor output.

$$R_{FSR} = \frac{R_{fixed}}{(\frac{V_{in}}{V_{out}}) - 1} \tag{4.1}$$

Three sensors is the minimum number that can be used to support the full weight of an object with stability. All the objects' weight must go through the sensors otherwise inconsistent weight distribution would create unreliable readings. Measurements read from changes in the weight of an object, can be used by the computer to detect when liquids are added or removed from a mug or a milk container for example. When the sensors are not in contact with any surface their weight reading becomes zero, allowing them to be used to help detect when an item is picked up.

There are many factors that affect the accuracy and repeatability of measurements from the FSR sensors. The contents of the instrumented object, the slope and the evenness of the surface on which the object is placed, and the distribution of weight between the three force sensors can all affect the readings. This can cause large changes in the three sensors' readings when an object is moved, even without any changes in the actual weight of the

object. However, once the object is stationary and stable, readings from FSRs are expected to be reliable in measuring the real weight changes.

In order to evaluate the capability of the FSR sensors to measure the weight change while stationary, an experiment was carried out by comparing measurements calculated from three different instrumented mugs with accurately measured weights using a very precise digital scale. This experiment was carried out using the following steps:

- place the empty mug with the coaster attached to it on the digital scale.
- Start recording data from the sensors.
- Pour water into the mug to increase the weight into a set of values including 30, 140 and 250 grams to replicate real scenarios of adding milk and adding water to the half and full capacity of the mug, respectively. Also add a sugar cube which weights around 5 grams.
- Stop the recording.
- Calculate the weight of the mug using FSRs' readings at the beginning and end of the recording. Conduct the end and beginning weights.

The process was repeated ten times for each value. The mean and variance of the readings are shown in Figure 4.2. Focusing on half of the mug (140 g), the experiment suggests that for each CIC the variance of the measurements is less than 20. In this case, from the central limit theorem, with 10 samples we can be 95% confident that the true mean lies within 2% of the measured mean. These measurement shows that the change in the weight of an object can be extracted from FSRs reading under the object when it is stationary. However, weight of the milk added to the mug is considered to be 30 gram, which in real life scenarios this weight can reduce to less than 10 gram (depending on the taste of the user); for lower weights,

detection of the adding milk activity is more challenging, and impossible to do so with only looking at the FSR readings.

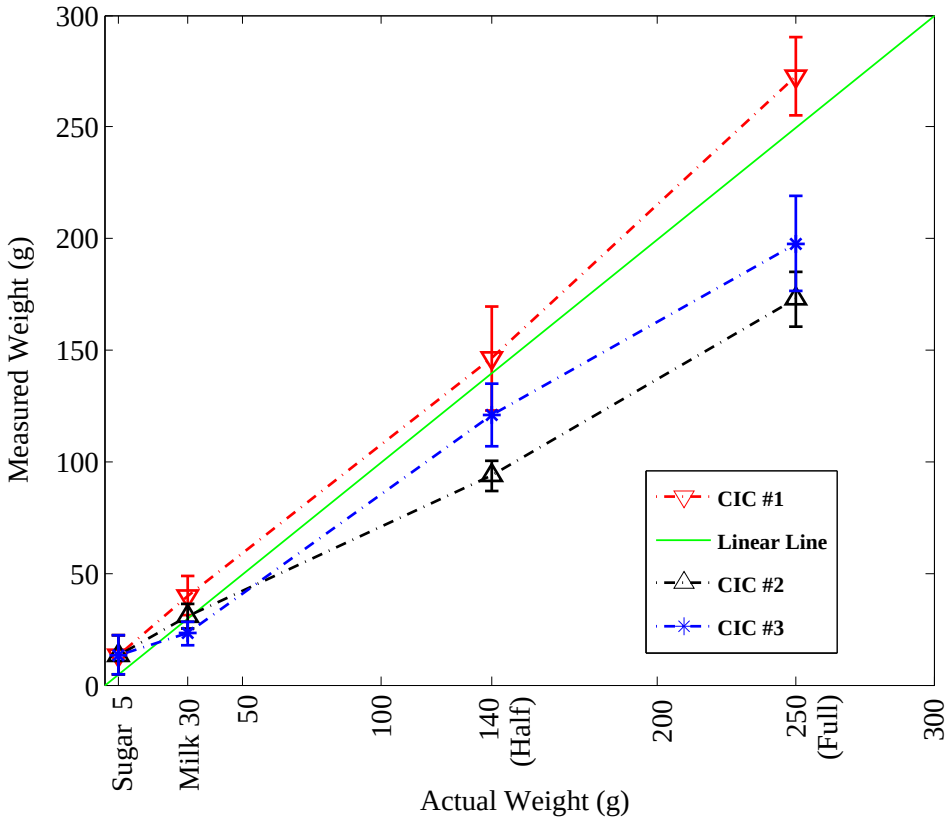


Fig. 4.2 *The object's weight is measured by three different CogWatch instrumented coasters (CICs) versus the actual weighs of the object measured by an accurate scale*

4.2.2 ACCELEROMETERS

The accelerometer used in the coaster is the Analog Devices ADXL335. It provides acceleration measurements on 3 axes in a range of $\pm 3g$. This accelerometer provides an analogue output signal that is read by the ADC of the microcontroller. There are multiple kinds of interaction with instrumented devices that the accelerometer is well suited to detect:

- when an instrumented object is moved, the accelerometer measures the changes in motion.
- the accelerometer is able to detect very slight movement of boiling water when it is directly attached to the body of the kettle
- the accelerometer can be used to detect the orientation of an object; accelerometers interpret gravity as continuous upwards acceleration, the distribution of this force between the three axes of the accelerometer can be used to discern the orientation of the device. This effect is useful in detecting actions such as pouring with a kettle, or detecting whether an object is being held in an incorrect orientation.
- accelerometers are well suited to detecting the shock of an object being dropped; Which in this case the trial could be stopped by prompting the patient with a warning.

Figure 4.3 shows the output of the sensors during an example of performing the pouring boiled water from kettle to the mug.

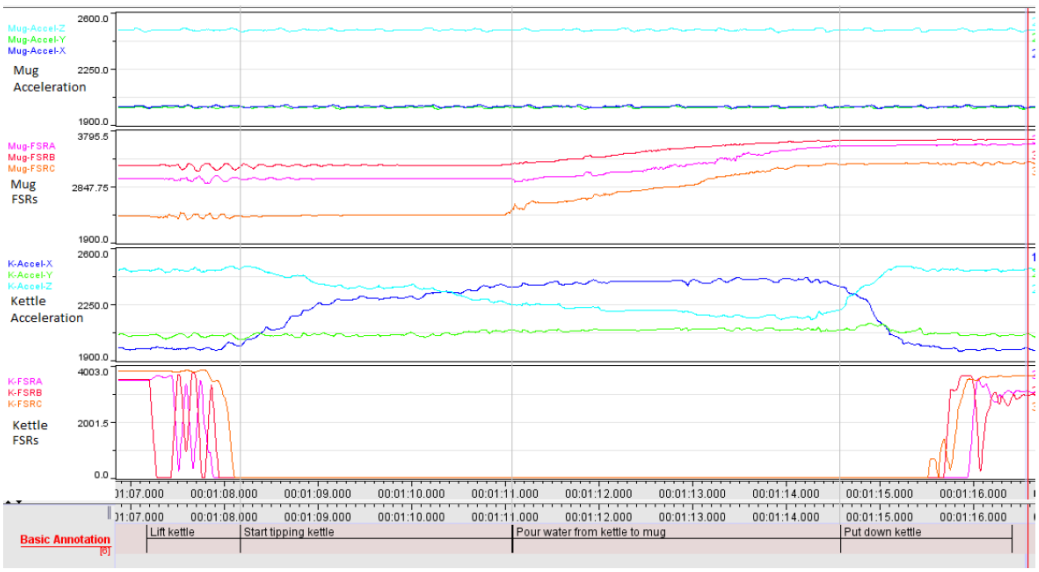


Fig. 4.3 *From top to bottom, output of mug accelerometers, mug FSRs, kettle accelerometers and kettle FSR sensors, during pouring water from the kettle into the mug*

4.3 KINECT^{TM} SENSOR

In addition to outputs from CICs, the system uses hand coordinates data, captured using a non-invasive Kinect^{TM} sensor, developed by Microsoft. The main reason to add a video sensor to the CogWatch system is the inability to add any physical sensor to teabags and sugar cubes. As it is pointed out in literature review , the video analysis is computationally intensive and challenging to run in real-time. Using the KinectArms toolkit [127], is a fast and easy to use alternative to often computationally expensive, prone to error (particularly if colour is used for separation), and dependent on good lighting conditions [128]. In recent years, Kinect^{TM} has been used widely in physical and cognitive rehabilitation applications for patients with neurological disorders (e.g. stroke and Parkinson's) [129–131]. The Kinect sensor consists of a RGB camera, two 3D depth (infra-red (IR)) sensors and a multi-array microphone, which is shown in Figure 4.4. Using Kinect to track hand movements on the table can be beneficial in the detection of sub-goals involving items which can not be instrumented using sensors, e.g., sugar cubes, teabags and spoon.

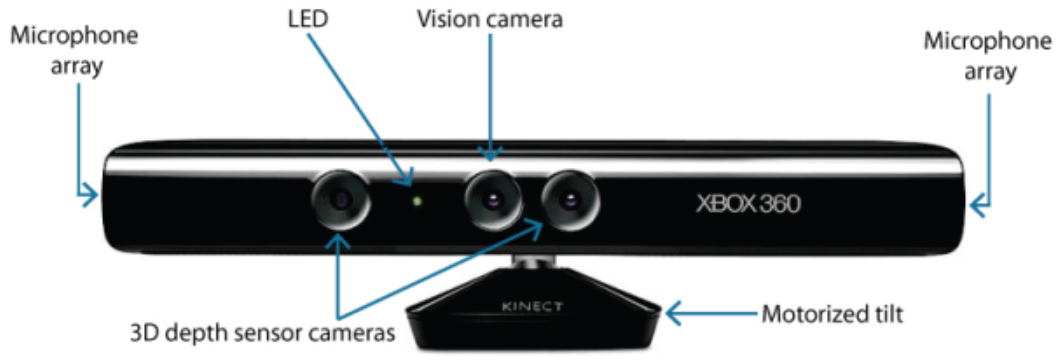


Fig. 4.4 *Kinect^{TM} sensor showing its parts including, a RGB (vision) camera, two IR sensors, and multi-array microphone*

4.3.1 HAND TRACKING USING KINECT

The Kinect™ depth sensor is capable of detecting human body joints (e.g., head, neck, shoulders, and hands) using Kinect™ SDK development software, when it is located in front of a person. The problem with placing Kinect in front of the tea-making table is the detection of hands (on the table) is not accurate enough for the recognition of sub-goals in the tea-making task. Also, grabbing items located on the table cause discontinuity in tracking the hand. For the recognition of sub-goals, from all the body joints which the Kinect is able to track, only the hands' locations are used. Thus, hand movements on the table can be tracked more accurately by locating the Kinect sensor perpendicular and pointing down to the centre of the table. Also, this ensures continuity while detecting and tracking hands over a table surface when working with objects.

In order to detect and track hand movements over the table in real time, a software written in C# \C++ and compiled using Microsoft Visual Studio is used. This software employs image processing libraries and toolkits, such as Microsoft Kinect™ SDK development software, KinectArms toolkit [127], OpenNI and OpenCV libraries.

Kinect SDK is used to capture images from the Kinect sensors and align the colour and depth images before any further processing. Subsequently the hand tracking process is applied through stages of table detection (calibration) and arm detection.

TABLE DETECTION (CALIBRATION)

The height and size of the operating table are determined during the calibration process; this process is carried out at the beginning of every trial. The calibration process needs to be repeated every time the table or Kinect is moved. In order to do the calibration, first, the middle of the tea-making set-up and the table are aligned. The middle of the table is

signed on the video from the Kinect camera, and the middle of the set-up is marked on the sketch paper, which has a mark for the location of items on the table. Then, by clicking the calibration button on the software GUI, the following steps will be applied automatically, until a table with four corners is detected:

1. A Laplacian filter is a second order derivative filter, and it is applied over the Kinect's depth image data to find areas of rapid changes. Then, sharp changes in depth (which are edges of the table in Cogwatch setup, because the table is closer to the Kinect than its surrounding background) are detected by thresholding the result of the filter.

2. To ensure continuity depth edges are dilated.

3. The region in the centre of the image, which is enclosed by the continuous edges represents the table area.

4. To find the table corners, a K-curvature [132] with k-value of 30 and threshold of 70°, is applied. If the number of detected corners at this stage, were not equal to four, the process of detecting the table, starts over from step one, And, in the case of finding four corners next step will the final steps towards recognising the table.

5. The height of the table, is set as the lowest value among the table pixels.

ARMS DETECTION

The detection of arm movement on the table is carried out by applying the following steps:

1. By removing all background pixels around the table from the depth image, the remaining pixels present the table area, and can be arm candidates.

2. The edges of the arms have been found by applying a Canny edge detector, with maximum and minimum threshold of 200 and 150, and a Sobel filter of order 5.

3. Contours of each arm candidate are found using OpenCV's Suzuki85 algorithm. All small contours (< 40 pixels in length) are discarded as noise. The remaining contours are considered as arms.

4. The base of each arm is determined by finding the intersection of the arm with the edge of the table.

5. The location of the palm on the table is calculated using a Euclidean distance transform of the arm's boundary points.

4.4 LIMITATION OF THE SYSTEM

In this section, the limitations of the system are discussed. similar to any other project where obstacles could distance you from reaching your optimal goal, there were few constrains associated with this experiment. Most of the limitations in this experiment relates to sensors, video setup and method used.

One of the main sensor related constraint is the inability to attach any sort of sensor to teabags and sugar cubes in order to monitor whether there were picked up from container or not. Another hardware limitation which should be taken to account is the maximum number of bluetooth connections supported by a single device. The computer could only support 7 bluetooth nodes/connections which means implementing more sensors could by itself create problems. The accuracy level of FSR sensors does have a huge impact on the number and magnitude of actions which could be monitored. By implementing more accurate FSR sensors for example, the presence of tea bag or sugar cube in a cup could be detected as well as quantity errors.

Kinect video calibration process would need optimisation to increase employability. A small adjustment in placement of the table and Kinect requires recalibration which is a

lengthy process itself and also could give unwanted results if happens during the experiment. Lights scatters and different light conditions could have negative impacts on the performance of the Kinect device. The hand detection algorithm works by finding the hand and table intersected area, so any object placed on the edge of the table would be recognized as an extra hand (this includes patient's head). In addition to above, the system is only able to detect one hand on the table.

The selection of features for each detector is manual rather than use of a formal systematic approach to data-driven feature extraction techniques. Another limitation pf the algorithm is that each object has its own dedicated location meaning if a hand is to be placed in this location, the program will consider it as using that object.

Finding solutions to overcome these limitations requires a lot of effort in terms of hardware and software algorithms which could be considered in future improvements before it is implemented for consumer usage.

4.5 FEATURE EXTRACTION

The data captured from the sensors in their raw form are often confusing for the sub-goal detection task. Therefore, some basic features are extracted from the raw captured data, which accommodate more distinctive information about each sub-goal. In this section, all these features are explained.

The output of all the sensors synchronised at the sampling frequency rate of 50 Hz, which means the time between two consecutive samples transferred from a sensor to the system is 20 ms. The Kinect's hand location data was obtained at 20 Hz, in order to upsample this data to 50 Hz, the linear interpolation method is used to construct new data points between two consequent hand locations. Also, the recognition of sub-goals in the system needs to happen

in real-time, which means features must be calculated over the limited and short window of samples. due to this restrictions, the calculation of frequency-based features, which are very popular to use in applications with IMU sensors and speech recognition systems, is impossible because of the low temporal or frequency resolution.

Feature extraction is performed by applying a sliding window of a 21-sample length (420 ms) and a 95% overlap (one sample frame length) on the output of each sensor. The first observation is calculated and sent to the action recognition after the first 20 samples are captured. After capturing the first 20 samples, observations (features) are sent to the AR system in real-time with a 10 sample (200 ms) delay. The raw (primary) sensor data is the value of the middle sample in a window. In this thesis, the name of an object followed by the feature abbreviation, is used to point to that specific feature calculated from sensors attached to that object, for example, “Milk-FSR-SmoothDer” means the “FSR-SmoothDer” feature for sensors attached to the milk container.

4.5.1 DERIVATIVE OF SMOOTHED WEIGHT (FSR-SMOOTHDER)

There are three FSR sensors located under each instrumented object (milk container, kettle and mug). Using data from these sensors the weight of each object can be calculated. however, the change in the weight of an object is important for recognition of the sub-goals, for example during the sub-goal of "adding milk in to the cup" the increase in the weight of the mug indicate the milk was added. So, one of the features used in AR is the derivative of the weight after applying a low-pass filer. Smoothing is done to eliminate small changes in FSR reading because of the table’s vibration.

To calculate the “FSR-SmoothDer” feature, first, the linearised resistance of a FSR (R_{FSR}) is calculated using Equation 4.1. The weight of the object is the sum of the weights on three

individual FSRs ($\frac{1}{R_{FSR}}$). Then, a low-pass Butterworth IIR-filter with 10 poles, 2 zeros, and very low ($0.002 \times f_s$) cut-off frequency is applied to achieve the smoothed derivative.

Finally, the derivative of the smoothed weight is calculated for the middle point of the window using the following regression formula

$$d_m = \frac{\sum_{\theta=1}^{\Theta} \theta(s_{m+\theta} - s_{m+\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \tag{4.2}$$

where d_m is the derivative at middle point of the window computed in terms of the previous and following samples of smoothed weight $s_{m+\Theta}$ to $s_{m-\Theta}$ in the window when Θ is equal to 10.

4.5.2 THRESHOLD OF FSR SENSORS (FSR-THRESH)

The "FSR-Thresh" feature is calculated using only the mid-point of the sliding window, to indicate whether or not the objects is picked up. It is done by thresholding the sum of the raw output of the three FSRs' sensors under an object. The raw recording of a FSR sensor appears in a 12-bit ADC output of the microcontroller varying between 0 and 4095 to represent the maximum (infinity) and minimum zero weight applied on the sensor, respectively. Theoretically, the value of a FSR sensor output when the object is picked up must be 4095, but in many recordings this is not the case, especially for the FSR sensors used under the kettle-base and old sensors that have been used for a longer period of time. Therefore, the threshold of 6000, for the sum of all three readings from the FSR sensors under an object, is empirically chosen to indicate that the object is picked up.

The "Mug-FSR-Thresh", "Milk-FSR-Thresh" and "Kettle-FSR-Thresh" feature are calculated for the coasters used under the mug, milk container and kettle-base, respectively.

4.5.3 WEIGHT AND ACCELERATION VARIANCE (FSR/ACC-VAR)

The variance of the weight and magnitude of the acceleration of an object, is calculated using Equation 4.3. For an accelerometer module, with acceleration of a_x, a_y and a_z in X, Y and Z axes, respectively, the magnitude of the acceleration is equal $\sqrt{a_x^2 + a_y^2 + a_z^2}$.

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{4.3}$$

In Equation 4.3, N is the number of samples in the window and μ is the mean value of the samples (weight or acceleration magnitude). This feature is named in the format of "object name-FSR/ACC-Var", e.g. "Kettle-FSR-Var" is the variance of the kettle's weight.

4.5.4 DERIVATIVE OF HAND'S LOCATION (KINECT-DER-X/Y)

To understand the speed which the hand is moving over the table, The derivative of the Kinect data (the location of the hand on the table), is calculated in X and Y directions, using the regression Equation 4.2 with order Θ of three. In next chapters, The feature is called "Kinect-Der-X" for derivative in X-axis and "Kinect-Der-Y" for derivative in Y-axes.

4.5.5 KINECT NEIGHBOURHOOD (HARD/SOFT)

In section 4.3.1, how to detect the location of the hands on the tea-making table was explained. Also, Figure 3.11 illustrates the template of the table and location of the objects in two-dimensional axes with the origin point (0,0) marked in the middle. The location of the hands, which are detected by the Kinect sensor, is presented in the same coordinate system.

In this section, two features for the hands' locations on the table when they are stationary, are introduced, namely, hard and soft neighbourhoods. The hand is assumed to be stationary

if the hand's movement between two successive samples is less than one centimetre. The distance that the hand has travelled between times t and $t+1$ is the Euclidean distance:

$$d(h_t, h_{t+1}) = \sqrt{(x_{t+1} - x_t)^2 + (y_{t+1} - y_t)^2}. \tag{4.4}$$

Where $h_t = (x_t, y_t)$ is the position of the hand at time t.

The idea behind employing such a feature is to detect when the hand is located in the selected neighbourhood area of the table (empirically around a specific object). There are two benefits to using such a feature; first, the trajectory of the hand reaching the areas is not important and so will not affect the feature. Also, it is not important which hand is present in that area, i.e., the features produce the same values for both left-right handed people using the system. The general two-dimensional Gaussian function (Equation 4.5) is used to calculate soft and hard elliptical neighbourhood features.

$$f(x,y) = \exp\ (\ -\ (\ a(x-x_0)^2\ -\ 2b(x-x_0)(y-y_0)\ +\ c(y-y_0)^2)) \tag{4.5}$$

Where x and y are the location of the hand on the table; x_0 and y_0 are the centre (mean) of the area. Also, in order to rotate the area by a clockwise angle θ, a,b and c are defined as follows:

$$a = \frac{\cos^2\theta}{2\sigma_x^2} + \frac{\sin^2\theta}{2\sigma_y^2} \tag{4.6a}$$

$$b = -\frac{\sin 2\theta}{4\sigma_x^2} + \frac{\sin 2\theta}{4\sigma_y^2} \tag{4.6b}$$

$$c = \frac{\sin^2\theta}{2\sigma_x^2} + \frac{\cos^2\theta}{2\sigma_y^2} \tag{4.6c}$$

Where σ_x and σ_y are the variances (the spread) of the area in x and y directions, respectively. Using Equation 4.5, the Gaussian neighbourhood feature of the hand when located at coordinate (x,y) is calculated from Equation 4.7.

$$G(x,y) = \begin{cases} 0, & \text{if } f(x,y) < 0.1 \\ f(x,y), & \text{otherwise} \end{cases} \tag{4.7}$$

The $f(x,y) = 0.1$ is an ellipse equation with the centre point of (x_0, y_0), radius of ($r_x = 2.146 \times \sigma_x$) and ($r_y = 2.146 \times \sigma_y$) in x and y directions. The Gaussian neighbourhood value is at the maximum when the hand is on the centre of (x_0, y_0) of the area and it's value reduces exponentially as the hand moves towards the edges of the area.

Using the same logic, the hard neighbourhood feature is calculated by applying the following equation:

$$E(x,y) = \begin{cases} 0, & \text{if } f(x,y) < 0.1 \\ 1, & \text{otherwise} \end{cases} \tag{4.8}$$

Hard neighbourhoods for "Front Actions" detector

For detection of the front sub-goals ("Add Sugar", "Add Teabag" and "Remove Teabag"), four hard-neighbourhood features around the sugar container, tea bag container, empty bin and mug, are used. The size and position of these ellipses are manually chosen, using hands' locations in isolated recordings of the front sub-goals in tea-making database (see section 3.4). Figure 4.5, shows the locations of the hands (when they are stationary over the table) at all times during the recordings of the front actions. The ellipse around an object is chosen in

such a way that covers as much hand's locations as possible (when it is interacting with that object), without overlapping with any other ellipses.

Location of the hands in "Fill Kettle" sub-goals

Figure 4.6, shows locations of the hands (when they are stationary over the table) at all times during the isolated recordings of "Fill Kettle" sub-goal. The information in this figure is used in Chapter 8 to explain the behaviour of the results while changing size of the neighbourhood features around the jug and kettle.

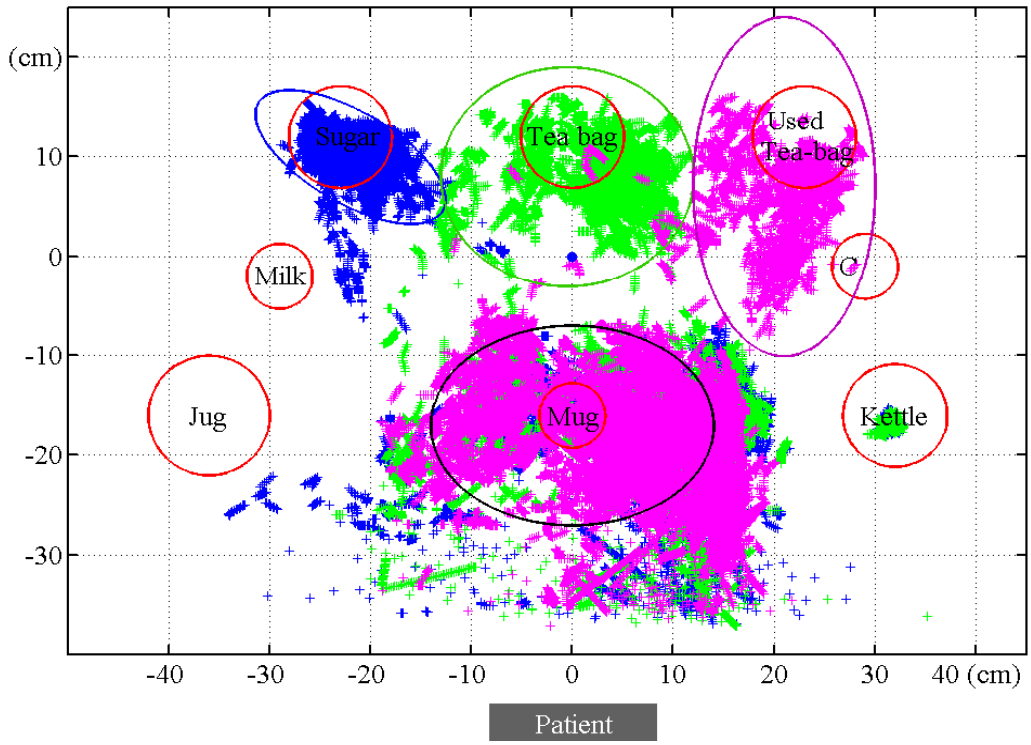


Fig. 4.5 *Hand locations and neighbourhood area for "Add Sugar" (blue), "Add Teabag" (green) and "Remove Teabag" (pink) sub-goals*

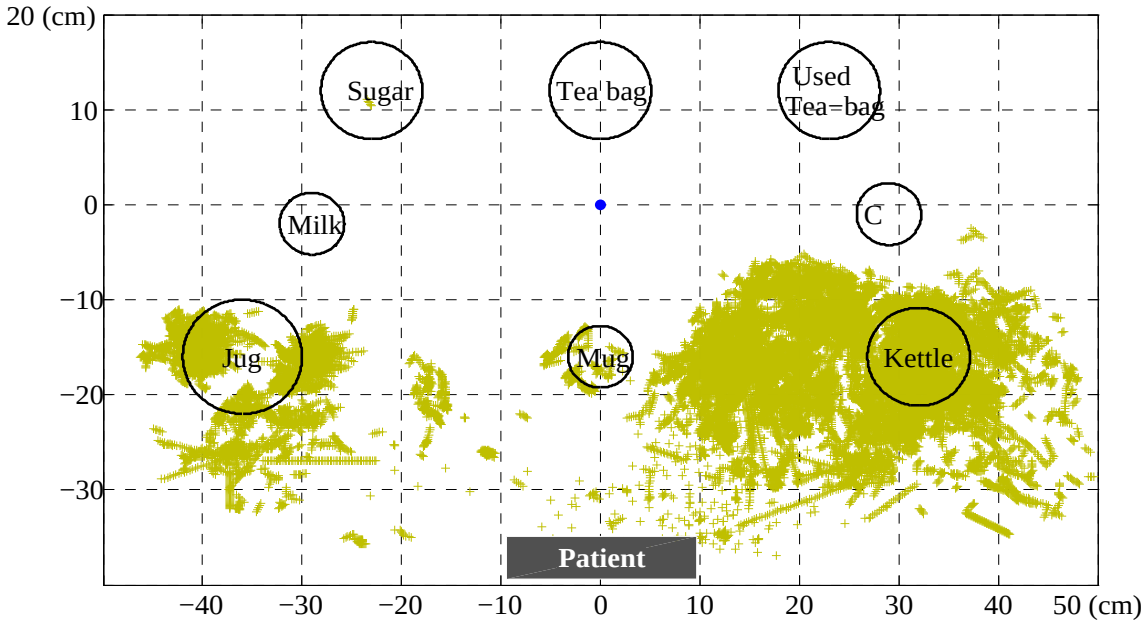


Fig. 4.6 *Hand locations in all isolated recordings of "Fill Kettle" sub-goals*

4.6 SUMMARY

In this chapter different types of sensors that are used in the CogWatch system are introduced, such as CogWatch instrumented coaster (CIC) consists of three force sensitive resistors (FSRs), a 3D accelerometer and Kinect sensor. Also, the process of table calibration and hand tracking for the Kinect sensor are explained. Furthermore, the limitations of the system discussed, in terms of attaching sensors to objects, Kinect calibration, hand tracking, and feature extraction methods. Finally, The calculation of the five types features from sensors' raw-data, have been explained.

When the system is up and running, all sensors' raw data plus extracted features, are being stored on computer's memory and being sent to the ARS, in a form of a 56-dimensional array. This feature vector (array) is fully shown in Table 4.1. The last column of the table includes the mapping of feature vector elements to the sub-goal detectors.

Table 4.1 *The feature vector in Cogawtach. third column shows which detectors use that specific feature. PK = "Pour Kettle", FK = "Fill Kettle", AD = "Add Milk", FA = "Front Actions" and ST = "Stir".*

Index	Description	Detector
1-3	Mug Acceleration in 3 directions: X-Y-Z	-
4-6	Raw values of the three FSR sensors under Mug	-
7-9	Milk Acceleration in 3 directions: X-Y-Z	AM
10-12	Raw values of the three FSR sensors under milk container	-
13-15	Kettle Acceleration in 3 directions: X-Y-Z	PK
16-18	Raw values of the three FSR sensors under kettle	-
19-21	Primary calibrated hand's location on the table in 3 direction of X-Y-Z	-
22-24	First hand's locations raw data in three dimensions	-
25-27	Second hand's locations raw data in three dimensions	-
28	Threshold of the data from FSRs under the mug	-
29	Threshold of the data from FSRs under the milk container	AM, FK, FA, ST
30	Threshold of the data from FSRs under the kettle	PK, FK, FA, ST
31	Weight of the mug calculated using features # 4-6	-
32	Weight of the milk container calculated using features # 10-12	-
33	Weight of the kettle calculated using features # 16-18	-
34	Weight of the mug (Feature #31) is smoothed and its derivative is taken	PK, AM
35	Weight of the milk container (Feature #32) is smoothed and its derivative is taken	-
36	Weight of the kettle (Feature #33) is smoothed and its derivative is taken	-
37	Variance of the mug's weight (feature # 31)	FA, ST
38	Variance of the Milk container's weight (feature # 32)	-
39	Variance of the kettle's weight (feature # 33)	FK
40	Variance of the acceleration's magnitude for mug	FA, ST
41	Variance of the acceleration's magnitude for milk container	-
42	Variance of the acceleration's magnitude for kettle	FK
43-44	Derivative of hand's location in X and Y directions (features # 19 and 20)	FK, FA
45-46	Hard and soft neighbourhoods around the mug area	FA, ST
47-48	Hard and soft neighbourhoods around the sugar container	FA
49-50	Hard and soft neighbourhoods around the teabag container	FA
51-52	Hard and soft neighbourhood around the used teabag container	FA
53-54	Hard and soft neighbourhoods around the kettle	FK
55-56	Hard and soft neighbourhoods around the jug	FK

CHAPTER 5

REAL-TIME ACTION RECOGNITION OVERVIEW

5.1 INTRODUCTION

The objective of the CogWatch project [13] is to provide an intelligent computer-based rehabilitation system to help patients recover their cognitive ability to carry out ADLs. To achieve this, the system must be able to monitor the patient's progress through performing the ADL and provide appropriate guiding cues or feedback when an error is detected or anticipated. The automatic recognition of the executed actions during the rehabilitation process is the main focus of this chapter.

The challenge is to recognise executed sub-goals from sequential sensor data. This is analogous to the speech recognition problem, where the objective is to recognise words from a sequence of outputs from an acoustic sensor (a microphone). HMMs have become established as a successful approach to automatic speech recognition (ASR), and they have been very popular in activity recognition researches and applications to exploit temporal

dependencies. Therefore it is reasonable to use them as a starting point for a sensor based action recognition.

In Section 5.2 introduces the parallel architecture of HMM action recognition that consists of an independent set of detectors, in order to overcome the problem of concurrent activities (sub-goals). In section 5.3, the structure of a sub-goal detector is explained. Finally in the last section, the real-time decoding algorithm of the CogWatch action recognition system is explained.

5.2 OVERVIEW OF THE ACTION RECOGNITION SYSTEM

Hidden Markov models (HMMs) are a generic framework for statistical sequential pattern processing, but they have received most attention in the area of automatic speech recognition (ASR) (for example, see [133]). However, there are a number of important differences between action and speech which determine the design of our HMM-based AR system:

- In ASR, words occur one-after-another whereas in AR, actions can occur in overlapping time, so that the natural structure is a partially-ordered lattice rather than a sequence. Overlap may occur, for example, if the subject uses both hands, or executes one or more sub-goals while the kettle is boiling. Therefore a conventional ASR decoder, which will compute the most probable sequence of actions given the data, is not appropriate for AR. This partially-ordered structure also complicates the inclusion of sequential constraints in the decoder.

- In ASR, the same features are used by all HMMs whereas in AR, different subsets of features are appropriate for recognising different sub-goals.

- In AR there is no universally accepted equivalent to a ‘phone set’.

The key process in a typical HMM-based ASR system is a Viterbi decoder [134] explained in Chapter 2. In speech recognition, the language model and the individual HMMs are compiled into a single network and the most probable path through this network is found using Viterbi decoding. However, because the execution of sub-goals is realised as a partially-ordered lattice, an alternative architecture is needed for AR.

The proposed solution in this thesis involves using independent real-time HMM-based detectors to work in parallel asynchronously. In this way the recognition task is broken down to the activity spotting of different sub-goals which together can monitor the process of the patient during completion of the tea-making task.

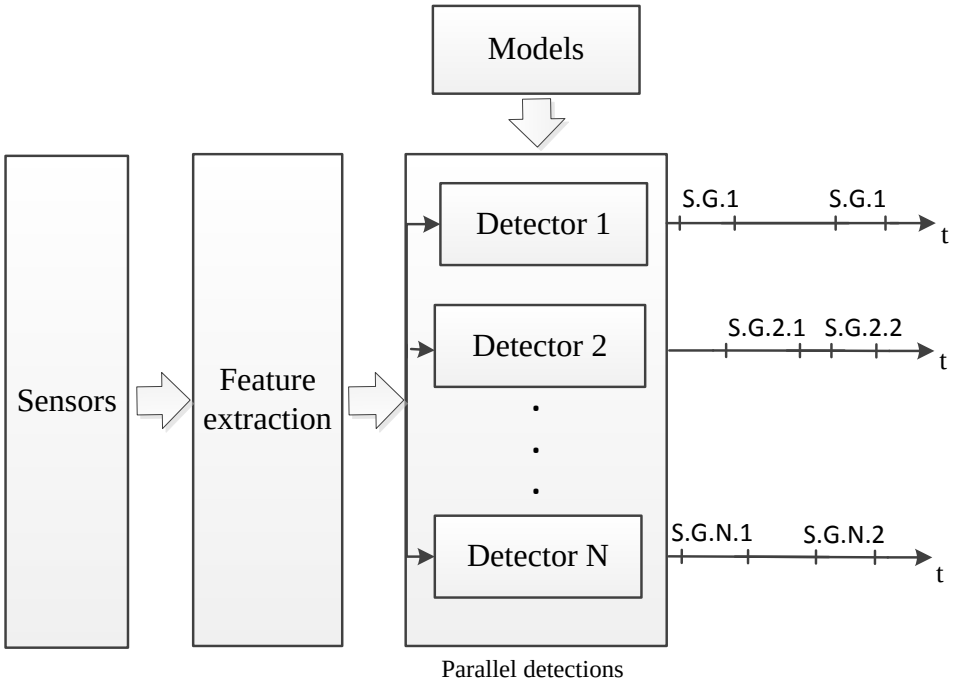


Fig. 5.1 *Cog-watch parallel action recognition system*

The input to the action recognition system is a feature vector, all extracted from sensors explained in Chapter 4. Instead of having the optimum sequence of models that represent the input data (like in speech recognition tasks), the output of the action recognition consists of

the time boundaries for any detected sub-goals in each detector. The digram of the action recognition system is illustrated in Figure 5.1.

5.3 OVERVIEW OF THE SUB-GOAL DETECTOR

The structure of the sub-goal detectors in this thesis, is largely used in the continuous speech key-word spotting systems [135]. A detector consists of a feature selector, HMM-set, decoding network and decoder unit, is responsible for detection of the sub-goal(s) in tea-making task.

Features for each detector are logically chosen depending on the involvement of their corresponding sensors in the process of performing the target sub-goal. For example, for detection of pouring water from kettle to the mug, it is sensible to choose features extracted from sensors attached to the kettle and mug.

The basic idea behind the spotting systems, is to train a relatively large "background" (Filler) model, which is able to represent all activities in the training data with an average likelihood probability; also, train specific models ("sub-goal") for the target activities in order to achieve high likelihood score in the decoder when they are performed. In this work, in order to further discriminate the toying with the kettle and milk container and correct use of these object to perform the "Pour Kettle" and "Add Milk" sub-goals, auxiliary recordings of "Toy Kettle" and "Toy Milk" (Chapter 3) are used to build separate non-target sub-goal models.

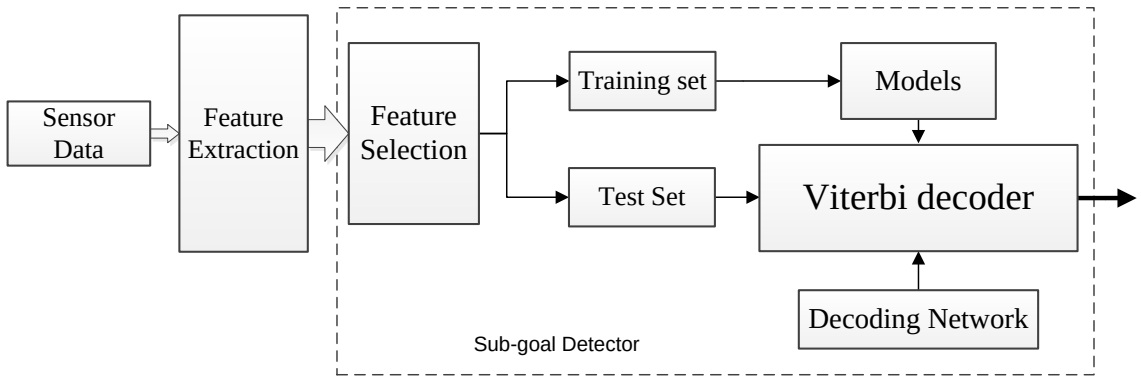


Fig. 5.2 *HMM-base sub-goal detector structure*

The conventional Viterbi decoder is used to find the most likely sequence of hidden states in HMMs that produce the input data. Furthermore, a real-time version of the Viterbi algorithm is implemented to be used in the CogWatch system. The implementation of the real-time Viterbi is explained in next section. Figure 5.2 shows the overview of a sub-goal detector.

5.4 REAL-TIME VITERBI DECODING

The conventional Viterbi algorithm is describe in section 2.6. In the basic Viterbi decoding, at the end of the observation sequence, the search path with the maximum probability is backtracked to find the optimum state sequence. Consequently, the best explanation of the data is not recovered until the final time T. However, in a real life situations the input to the recogniser is an unbounded sequence of observations $O = \{o_1, ..., o_t, ...\}$, so the memory required to store the $\psi_t(i)$ and $\delta_t(i)$ (section 2.6) will increase and no output will be produced. The solution to overcome these problems is to apply the "partial trace-back" algorithm [136, 137] and prune out the search paths which are unlikely to succeed.

The use of partial traceback algorithm allows the decoder to:

1. output the partial results as the decoder accumulates the likelihood probabilities of the new observations.

2. reduce the memory requirements by removing the values of $\psi_t(i)$ and $\delta_t(i)$ for old partial results and inactive search paths. The latter is done by beam pruning the unlikely search paths.

5.4.1 PARTIAL TRACEBACK IMPLEMENTATION IN REAL-TIME VITERBI

The objective of the partial traceback algorithm is to find a time t_0 in the past that optimum sequence of HMMs up to t_0 is not effected by the future observations. This can be achieved by tracing back all optimum paths to all states in the Viterbi algorithm. Hence, new arrays able to store trace-back information at time t, are introduced.

The entry and exit states of a HMM (q_0 and q_e, respectively) are non-emitting (figure 6.1). Non-emitting states connect HMMs together and don't have any output probability distributions associated with them. Also, any transition to/from a non-emitting state happens instantaneously. Calculation of the forward probability array $\delta_t(i)$, for state i at time t, explained in section 2.6. Also, a_{ij} is the transition probability from state i to state j. For each state i at time t, the array $\beta_t(i)$ stores the time in which the optimum path first entered to the model that state i belongs to. The entry time to the model is propagated through the optimal path, using the following formulas:

$$\begin{aligned} \beta_t(q_0) &= t \\ \beta_t(i) &= \beta_{t-1}\left(\underset{j}{\operatorname{argmax}}\left[\delta_{t-1}(j)a_{ji}\right]\right) \end{aligned} \tag{5.1}$$

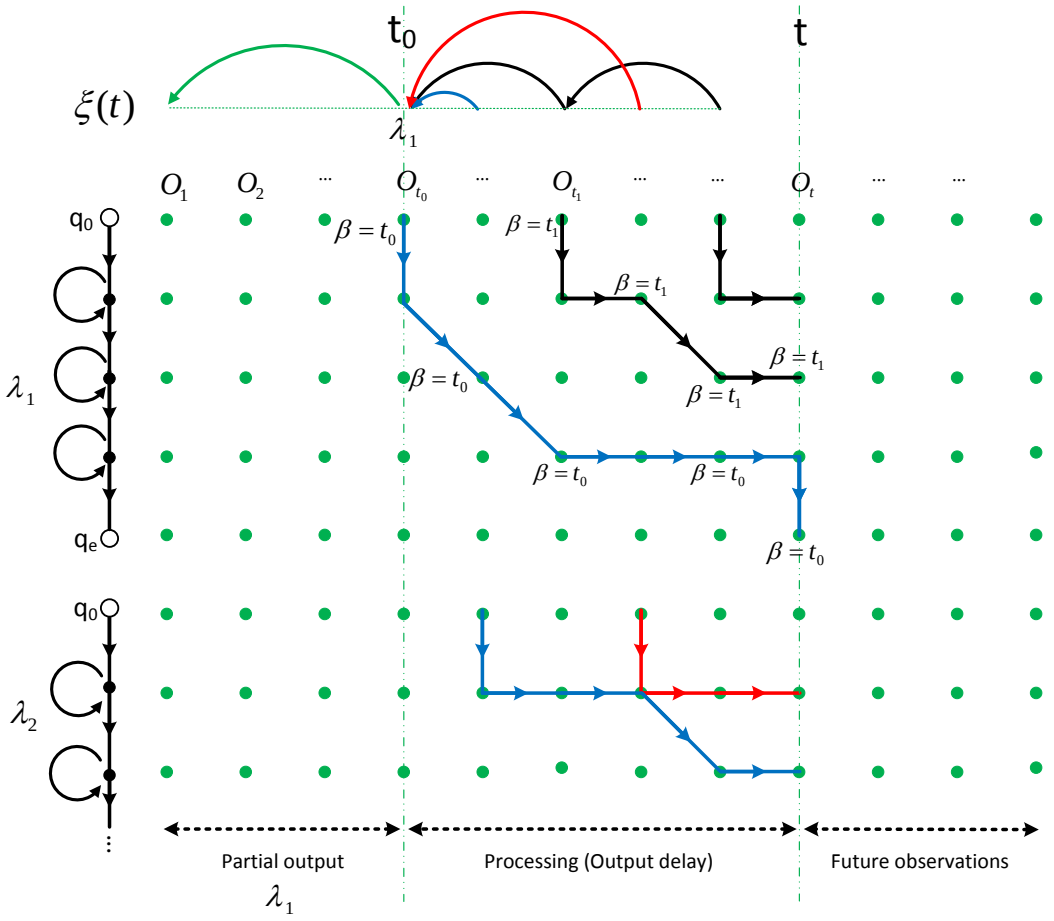
In the CogWatch rehabilitation system, the task model unit and user, are not interested in details of the optimum hidden state sequence $\hat{Q}$, but only in corresponding sequence of models $\hat{\lambda}$. The model link record array $\xi(t)$ is defined to store two values: index of the best model λ_m (with highest probability) that finishes at time t which is denoted by $\xi_m(t)$, and the start time (entry point) of the model λ_m which is denoted by $\xi_{tm}(t)$. Backtracking the $\xi(t)$ array, results in the optimal sequence of models up to time t, i.e., the $\xi(t)$ array keeps track of the most likely sequence of HMMs up to time t. These two values are calculated from the following equations:

$$\begin{aligned} \xi_m(t) &= \underset{i \in q_e}{\operatorname{argmax}} \left[\delta_t(i) \right] \\ \xi_{tm}(t) &= \beta_t \left(\xi_m(t) \right) \end{aligned} \tag{5.2}$$

Here, the q_e includes end states of all models, so the $\xi_m(t)$ is the index of the model with the highest end-state forward probability at time t.

All active search paths can be traced back, by Using $\beta_t(i)$ and $\xi(t)$ arrays together (like $\xi\left(\beta_t(i)\right)$) to find the most likely sequence of HMMs (with their entry times) ending in state i at time t. The partial trace-back is done by tracing back active paths from all states at time t. If the optimal sequence of HMMs from all states converge at a time t_0 in the past, partial results can be output. The partial result is the trace-back of $\xi(t_0)$. After, outputting the partial result, all arrays are removed from the memory up to time t_0.

Fig. 5.3 *Trellis illustration of partial trace-back algorithm, trace-back from all states at time t (current time) converge at time* t_0



The duration between the current time t and the convergence time t_0 is the detection output delay. The output delay is furthore discussed in section 7.4.2. The trellis illustration of the partial trace-back algorithm is shown in Figure 5.3. The optimum sequence of models from all states at time t (current time) converge at time t_0.

5.5 SUMMARY

In this chapter the automatic action recognition difficulties were discussed, and compared with the automatic speech recognition. in order to address these issues, a novel parallel architecture of HMM-based action recognition were presented.

The real-time and memory efficient implementation of the trace-back algorithm in Viterbi decoder was explained in this chapter.

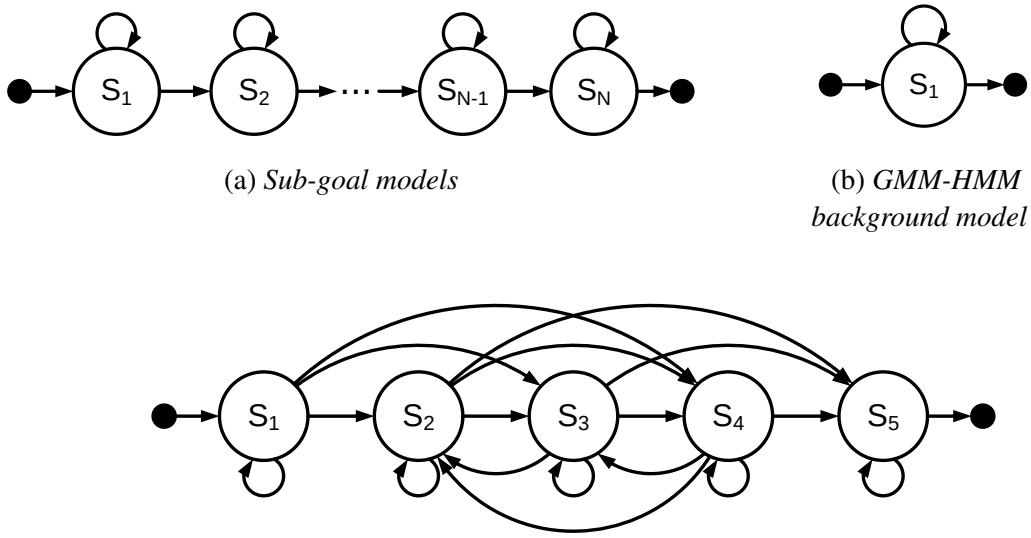
CHAPTER 6

SUB-GOAL DETECTORS

6.1 INTRODUCTION

The parallel action recognition system in this thesis, comprises five independent real-time HMM-based sub-goal detectors, namely “Pour Kettle”, “Add Milk”, “Fill Kettle”, “Front Actions” and “Stir”. These mutually independent detectors run parallel to each other, in order to recognise occurrences of all sub-goals during the completion of the tea-making ADL task. The layout of a sub-goal detector is illustrated in Figure 5.2. It consists of a feature selector unit, trained HMM models, a Viterbi decoder and a model network. Among these units, the conventional and real-time Viterbi decoding algorithms are explained in Sections 2.6 and 5.4.

In this chapter, first, different types of HMMs used for sub-goals and background activities, are introduced in Section 6.2. Then the Model-loop decoder network in general, which has the same structure for all detectors, is explained in Section 5. Finally in Section 6.4, the HMM models, sensors and feature-vector used for each specific detector is explained. The most important aspect of the chapter, is the understanding of the sensors and features



(a) *Sub-goal models*

(b) *GMM-HMM background model*

(c) *DNN-HMM background model*

Fig. 6.1 *Schematic representaion of HMM models: (a) sub-goal models, (b) GMM-HMM background model, and (c) DNN-HMM background model*

deployed in each detector; to understand the chapter, knowing the abbreviation name of the features is essential (please see section 4.5).

6.2 STRUCTURE OF HMMS IN SUB-GOAL DETECTOR

In each detector, two types of HMMs, namely, sub-goal and background are trained; any sub-goal model in a HMM-set is optimised to present one specific sub-goal (target or non-target). Also, the background model is optimised to present all activities which are not covered by the sub-goal models in a detector, including silence.

Sub-goal HMM models, take the form of the left-to-right linear HMMs [138] (LR-HMM) with N number of states, and each state is associated with a single component Gaussian probability density function (PDF). In this work, Linear HMMs are considered with one-step transitions (i.e. from state i to $i+1$) (Figure 6.1a). To achieve this, the transition matrix has been initialised in such a away that, only the main diagonal and the first upper diagonal

components are non-zero. On the other hand, two different HMM structures are used for the background ("toying") model in GMM- and DNN-HMM systems. The background model in GMM-HMM system, is a single-state HMM with a loop to the same state; and the background model in the DNN-HMM system, is a five-state HMM with state connections which illustrated in Figure 6.1c.

6.3 DECODER MODEL NETWORK

The Viterbi decoder unit in the sub-goal detector uses the model network to describe the sequence of models that can be recognised. A sub-goal level network is represented by a model-loop which simply puts all models of a detector, including all sub-goals and background models, in a loop and therefore allows any model to follow any other model. Figure 6.2, shows the structure of the recognition network used in the system. The insertion penalty is applied on the transition from the end of one model to the start of the next.

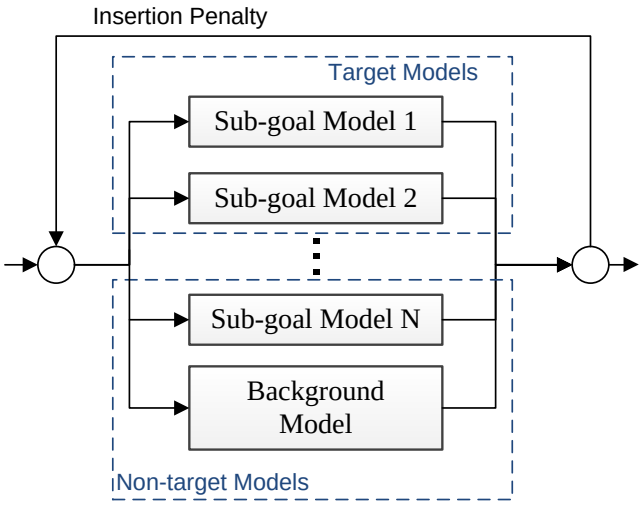


Fig. 6.2 *Model-loop decoder network with an insertion penalty for transition between models*

6.4 SUB-GOAL DETECTORS SPECIFICATIONS

In this section, the target and non-target sub-goal models are named. Also, the selected feature-vector in each detector, which is used for training and decoding purposes, is defined. The names of the sensors and features used in this section, are expressed in an abbreviated form and are comprehensively explained in Section 4.5. The preliminary sensors used in each detector is summarized in Table 6.1. The Cogwatch instrumented object is presented as CIC in the table, i.e., as an example Mug-CIC is the the name of the coaster mounted under the mug.

Table 6.1 *List of main sensors used in each sub-goal detector*

Detector	Main Sensors
Pour Kettle	1.Mug-CIC 2.Kettle-CIC
Add milk	1.Mug-CIC 2.Milk-CIC
Fill Kettle	1.Kettle-CIC 2.Kinect™
Front Actions	1.Mug-CIC 2.Kinect™
Stir	1.Mug-CIC 2.Kinect™

6.4.1 "POUR KETTLE" SUB-GOAL DETECTOR

The objective of the "Pour Kettle" detector is to recognise the action of pouring water out of the kettle. Also, the detector must be able to figure out if the water was poured correctly into the mug, or outside the mug by mistake.

HMM-SET

The HMM-set of the "Pour Kettle" detector consists of three HMMs, two sub-goal models, the "Pour Kettle" as a target sub-goal and the "Toy Kettle" as a non-target sub-goal, and

one background ("toying") model. The "Toy Kettle" model corresponds to any activity that results in picking up the kettle but not pouring water into the mug.

FEATURE-VECTOR

The "Pour Kettle" sub-goal detector selects features calculated from kettle-body accelerometer, kettle-base FSRs, and Mug-CIC FSRs sensors to form a five-dimensional feature vector in total. The five features are described below:

1–3: Raw output data captured from the kettle-body accelerometer in X, Y and Z axis is used to detect movements of the kettle, especially the tilting action.

4: The "Mug-FSR-SmoothDer" feature, which indicates any changes in weight of the mug, i.e, the weight of the mug increases when boiled water is added.

5: The "Kettle-FSR-Thresh" feature, which specifies whether or not the kettle is lifted at any given time t in a form of a binary value.

For recognition of the "Pour Kettle" sub-goal, output-data from other sensors are of no interest, e.g. Kinect or Milk-CIC data.

6.4.2 "ADD MILK" SUB-GOAL DETECTOR

The objective of the "Add Milk" detector is to detect the action of pouring milk out of the milk-container, and be able to verify if the milk was poured correctly into the mug, or poured outside the mug by mistake. Sensors which are directly involved in detection of this sub-goal, are accelerometers and FSRs inside the Milk-CIC and Mug-CIC (Section 4.2).

HMM-SET

In order to discriminate between pouring milk inside the mug and fiddling with milk container, two sub-goal models, are trained to present the "Add Milk" as a target sub-goal and the "Toy Milk" as a non-target sub-goal. The "Toy Milk" model corresponds to picking up the Milk-Container but not pouring water into the mug. Another non-target model in this detector is the background "toying" model.

FEATURE VECTOR

The "Add Milk" sub-goal detector selects features derived from Milk-CIC FSRs,accelerometer sensors, and Mug-CIC FSRs sensors to form a five-dimensional feature vector. The five selected feature are as follow:

1–3: Raw output data is collected from the CIC-Milk accelerometer in X, Y and Z axis respectively. Movement acceleration of the object in three dimensions, plus, the effect of gravity acceleration (g-force), which always points towards the ground and its projection on axis changes when the object is tilt, is a perfect demonstration of any kind of movements, especially tilting the milk container.

4: "Mug-FSR-SmoothDer" feature indicates any changes in weight of the Mug including changes happening while pouring milk.

5: "Milk-FSR-Thresh" feature specifies whether or not the milk container is lifted at any given time in the form of a binary signal.

For recognition of the "Add Milk" sub-goal, output-data from other sensors, for example, sensors attached to kettle and kinect, are of no interest.

6.4.3 "FILL KETTLE" SUB-GOAL DETECTOR

The "Fill Kettle" detector recognizes the process of adding water from the pre-filled jug into the kettle. There are no sensors attached to the jug, thus, picking up and putting down the jug can be predicted by following the position of the hand on the table, which is captured using the Kinect sensor. Sensors that primarily have been involved in detection of this sub-goal, are accelerometers attached on kettle body, FSRs under the kettle base and Kinect (Chapter 4).

HMM-SET

The HMM-set used in 'Fill Kettle" detector consists of a sub-goal model and a background model, to present the "Fill Kettle" sub-goal and the remaining sub-goals, respectively.

FEATURE VECTOR

Input to the "Fill Kettle" sub-goal detector is an eight-dimensional feature-vector which has been calculated from the Kinect, kettle-base, kettle-body and CIC-Milk FSRs sensors output data. The eight features are listed below:

1: "Kettle-Acc-Var" feature benefits from super sensitive accelerometer sensors attached on the Kettle-body to detect very small movement of the kettle caused during pouring water into the kettle.

2: "Kettle-FSRs-Var" feature senses the pressure applied on the kettle during opening and closing the lid.

3: "Jug-Hard" or "Jug-Soft" neighbourhood feature, recognises when the hand is located in the vicinity of the jug. The effect of using hard versus soft elliptical neighbourhoods with different radius sizes are investigated in the next chapters.

4: "Kettle-Hard" or "Kettle-Soft" neighbourhood feature, recognises when the hand is located in the vicinity of the Kettle. The same neighbourhood feature as for the Jug is used for the Kettle.

5–6: "Kinect-Der-X" and "Kinect-Der-Y" features indicate the speed of the hand moving on the table in X and Y directions, respectively.

7–8: "Milk-FSR-Thresh" and "Kettle-FSR-Thresh" features, which indicate weather or not the milk container and kettle are lifted, respectively.

6.4.4 "FRONT ACTIONS" DETECTOR

"Front actions" detector is responsible to detect any occurrences of "Add Sugar", "Add Teabag" and "Remove Teabag" sub-goals, during the completion of the tea-making task. On the tea-making table, There are three containers placed on top side of the table, two of them are filled with the sugar cubes and teabags, and one is an empty container. At some point during the completion of these sub-goals, the participant's hand would reach the vicinity of these containers, either after taking out the used-teabag from the mug, or before placing the sugar cube and teabag inside the mug. Thus, the detector is primarily influenced by features calculated from Kinect and Mug-CIC sensors.

HMM-SET

In "Front action" detector, three target sub-goal models are trained, namely, "Add Sugar", "Add Teabag" and "Remove Teabag", which present adding sugar into the mug, adding a teabag to the mug and, emptying the used teabag from the mug sub-goals, respectively. Also, a single-state HMM model, is trained to present all other non-target background ("toying") activities.

FEATURE VECTOR

For the model training and recognition of the sub-goals in the "Front actions" detector, a ten-dimensional feature vector is selected. All features with their intuitive meanings in term of actions in tea-making task, are listed below.

1–4: "Sugar-Hard", "Teabag-Hard", "UsedTeaebag-Hard" and "Mug-Hard" features, are used to determine when the hand is placed in vicinity of the sugar container, the teabag container, the empty container for used-teabags and the mug, respectively. The size of ellipses use for these features, are determined from the location of the hand during all isolated recordings of "Add Sugar", "Add Teabag" and "Used Teabag" sub-goals. This is explained in Section 4.5.5.

5–6: "Mug-FSR-Var" and "Mug-Acc-Var" features are used for monitoring any minor movement of the mug including those caused by removing teabag from the mug, or dropping sugar cube and teabag into the mug.

7–8: "Kinect-Der-X" and "Kinect-Der-Y" features indicate the speed of the hand moving on the table in X and Y directions, respectively.

9–10: "Milk-FSR-Thresh" and "Kettle-FSR-Thresh" features indicate whether or not the milk container and kettle are lifted, respectively.

6.4.5 "STIR" SUB-GOAL DETECTOR

"Stir" detector is responsible for detection of the sub-goal in which the participant stirs contents inside the mug with a spoon. The feature vector that is used to train the models are mainly focused on the output data captured from CIC-Mug sensors and Kinect.

HMM-SET

Two models are trained for this detector, the “Stir” sub-goal model as a target, which present the action of stirring, and the “toying” background model, which present all other activities involved in tea-making task but stirring.

FEATURE VECTOR

The data used for training and recognition tasks in the ”Stir” sub-goal detector is a five-dimensional feature vector, described as follow:

1–2: “Mug-FSR-Var” and “Mug-Acc-Var” features are used for monitoring any minor movements of the mug including those caused by removing teabag from the mug, or dropping sugar cube and teabag into the mug.

3: “Mug-Hard” or “Mug-Soft” features recognise when the hand is located in the vicinity of the mug. The effect of using hard versus soft elliptical neighbourhoods with different radius sizes are investigated in the verification experiment, which is explained in next chapter.

4–5: “Milk-FSR-Thresh” and “Kettle-FSR-Thresh” features, which indicate weather or not the milk container and kettle are lifted, respectively.

6.5 SUMMARY

In this chapter the configuration of all five “Pour Kettle”, “Add Milk”, “Fill Kettle”, “Front Actions” and “Stir” sub-goal detectors are introduced. These configurations are used for all experiment in this thesis.

CHAPTER 7

EXPERIMENTAL DESIGN

7.1 INTRODUCTION

In previous chapters, the parallel action recognition system and the structure of individual detectors have been explained. Each detector needs a set of Hidden Markov Models to be able to decode the new input data into different classes. The main goal of this chapter is to build an optimal set of HMMs for each real-time detector in the system. In order to do that, three experiments will be carried out for each detector; namely isolated sub-goal verification, full-trial sub-goal spotting (detection) and real-time sub-goal spotting. To accomplish the "isolated sub-goal verification" experiment, the isolated recordings of sub-goals are used for training the HMM models and evaluation of the system's performance (testing phase). Similarly, in the "full-trial sub-goal spotting" and "real-time sub-goal spotting" experiments the isolated recordings are used for training the HMM model, but, during the testing (evaluation) phase of these experiments the full-trials of tea making are utilized instead of the isolated recordings.

Collected tea-making data-set is not perfect in the sense that, participants were asked to perform sub-goals and make cups of tea as they desire with the speed they choose, and

move or play with objects around the table. Also, two separate sets of recordings have been specifically collected for the "toying with kettle" and "toying with milk container" sub-goals. Although the testing database used for evaluation of the action recognition is not perfect and include misuse of objects, the only way to truthfully evaluate the system, is to collect data from stroke patients while using it in real-world environments.

The action recognition in this work is designed to work as a sub-goal detection system. It means that, detectors generate an output only when a complete execution of a sub-goal is detected. In each detector, real-time stream of data from relevant sensors are compared with the "target sub-goals" and "background" models. In order to build a reach "background" model, all available data including toying, imperfect, and other sub-goals recordings are used for training. Because of this, ideally any incomplete execution of sub-goals or toying with objects expected to be ignored by detectors.

The first two experiments (described in section 7.2 and 7.3) are designed to optimise parameters of HMMs for detection of sub-goals in isolation and in the context of tea-making, respectively. The last experiment is designed to assess the real-time performance of the system in the detection of each sub-goal. It also investigates the effect of manual fixes to reduce the real-time output delay, which is a time delay between the actual finishing time of the sub-goal and the time it appears on the output of the AR system. These experiments, must be repeated if the feature-set used in a detector has changed.

7.2 SUB-GOAL VERIFICATION SYSTEM

Given an isolated recording of a sub-goal, the goal of the verification system is to determine whether or not it contains the specific sub-goal. To achieve this, an isolated recording can be decoded through parallel detectors, each responsible for verification of pre-defined sub-goals.

The performance of the verification system for each sub-goal detector, can be evaluated and optimised by applying the following three stages:

- **Training**: for each sub-goal detector, various sets of Hidden Markov Models are trained, each with a different number of states and Gaussian mixture components.
- **Development**: for each set of models, verification performance of the individual detector on the development set has been evaluated.
- **Performance evaluation (testing)**: using the optimal HMM sets, the final performance of the system has been evaluated.

These stages have been repeated for each detector, when put together they will be able to carry out the verification task for all of the eight sub-goals that are included in the tea-making task tree (Section 3.2).

7.2.1 DATA SETS

Isolated recordings of sub-goals in the CogWatch tea-making data-set (Chapter 3.4), are used for the training, development and testing stages of the verification experiments in all detectors.

For each sub-goal detector, in order to have balanced data for training and performance evaluation of models, isolated recordings relevant to each model have been written into a separate list. For example, the "Pour Kettle" detector consists of three models, namely "PourKettle", "ToyKettle" and "toying"; so isolated recordings are divided into three lists: the first list includes names of all 144 isolated recordings of adding water from the kettle into the mug, the second list includes all 186 isolated recordings of toying with the kettle; and the remaining 1180 isolated recordings of the other sub-goals are kept in the last list. The

full description of the tea-making data-set, plus the number of trials and total duration of recordings for each sub-goal are available in Section 3.2.

In order to have independent sets of data for each stage, five-fold cross-validation is applied. For each fold, 60%, 20%, and 20% of recordings from each list are randomly selected, and put into three complementary subsets of training, development, and test respectively. This procedure is repeated five times in such a way that each isolated recording is used for testing and development exactly once. As a result of this process, the recording included in the test and development set of each fold are unseen to the training set in that fold, and different from the test and development set of other folds.

7.2.2 TRAINING HMM MODELS WITH DIFFERENT PARAMETERS

Generally the HMM-set of each detector consists of two types of models, namely sub-goals and background. Any sub-goal model is trained to represent one specific sub-goal (target or non-target) and the background model is optimised to fit any other activity apart from those covered by sub-goal models. In Chapter 6, for each detector, the target and non-target sub-goals and name of the models are described. For each detector, HMMs with different number of parameters are trained using the train-set in each fold of the cross-validation.

An HMM assumes that a signal comprises a sequence of stationary segments. In the case of sub-goals it is not clear how many states are appropriate. AS the number of states is increased the model has the potential to fit the data better, but at the expense of a large number of parameters and therefore a need for a large amount of training data, For these reasons experiments are conducted to estimate the optimal number of states.

For the initialization of a N-state sub-goal model in the HMM-set, parts of the isolated recordings that have been time labelled as sub-goals are divided into N equal segments, and the data in the n^{th} segment is used to calculate the mean and diagonal covariance matrix of the

n^{th} HMM state. Further more, iterative Viterbi training is applied on the same time-labelled segments of the training set , to initially optimise HMM parameters of sub-goal models.

In order to initialize the single-state background ("toying") model in the HMM-set of detector, the rest of the recordings in the training-set, which have not been used in training of the sub-goal models, are used to estimate the mean and diagonal covariance matrix of a single Gaussian PDF. Then, the single Gaussian HMM is repeatedly divided into $M \in \{$ 1, 2, 8, 16, 32, 64, 128, 256, 512 $\}$ number of Gaussian mixture components, and optimised using the E-M algorithm.

After the initialisation of the models and between each iteration of splitting the mixture components, the parameters of all the models are optimised using the Baum-Welch algorithm 5.1. the training process is repeated to build sub-goal models with different number of states $N \in \{$ 5, 10, 20, 30, 40, 50, 60, 70 $\}$.

7.2.3 EVALUATING VERIFICATION RESULTS

Isolated recordings of sub-goals in the development sets, are decoded into a sequence of models using the Viterbi algorithm [139]. the decoding process for each detector is carried out by matching the input sub-goal recordings against the network of HMMs, which have been optimised using training-set data in the same cross-validation fold.

To evaluate the result of the decoding, first, depending on which detector is operating, each isolated recording of sub-goals can be categorised as a target or non-target trial (e.g. a recording of pouring water from the kettle into the mug, is a target trial for the "Pour Kettle" detector, but a non-target trial for the "Front Actions" detector). For any target trial in a development set, if the detector output obtains the corresponding sub-goal model, it counts as a correct detection; otherwise it is a deletion (False Rejection) error. Furthermore, for any non-target trial, if the output of the detector contains any target model, it is an insertion

(False Alarm) error. It is also important to note that each isolated recording contains the execution of only one sub-goal, so any repetition of target sub-goal models in the result is an extra insertion error. Using the total number of correct detections, deletions and insertions, the performance of the system (in terms of accuracy percent or error rate percent %ER) can be measured from Equation 7.1.

$$\%Accuracy = 100 - \%ER = \frac{Corrects - Insertions}{Corrects + Deletions} \times 100 \tag{7.1}$$

The relative levels of insertion and deletion errors can be controlled by adding a fixed penalty to any transition between models, in the network. In each sub-goal detector, the performance of the system on the development set using each set of models with a different number of parameters, has been calculated and illustrated in figures in Chapter 8.

7.2.4 FINAL PERFORMANCE OF THE VERIFICATION SYSTEM

During the development stage, the most suitable number of parameters for HMM models which achieve the highest verification performance are found for each sub-goal detector. These results are biased to the development set data. In order to find the real performance of the verification system for each sub-goal detector, test-set trials, which have been kept unseen to the development and training sets, are decoded using the optimal HMM-set with the best verification performance on the development set. After counting the number of correct detections, deletions and insertions that happened during detection of trials in the testset, the performance of the each detector is calculated using Equation 7.1. It can be said that the overall performance of the verification system is the average performance of all sub-goal detectors.

7.3 SUB-GOAL SPOTTING SYSTEM

Given a full-trial recording of making a cup of tea, the sub-goal spotting system's task is to extract time boundaries in which any specific sub-goal has happened. As it was mentioned in Chapter 5, this can be completed by assigning separate detectors to explicitly spot different target sub-goals. This would allow each detector to be able to operate, by concentrating only on sensors that were involved in the execution of target sub-goals in that detector. Selected features for each detector have been explained in Chapter 6. The structure of the models used for the spotting task in each detector are the same as the verification system, but the models have been trained with more data.

After the spotting system has been completed the optimal hidden Markov models are presented to run the real time action recognition system.

7.3.1 DATA SETS

All isolated sub-goal and full-trial recordings from the collected tea-making database, are used for training models and spotting sub-goals in each detector, respectively. The full description of the database can be found in Section 3.4.

Each isolated sub-goal recording starts with an optional silence (toying), followed by the execution of one specific sub-goal and finishing with another optional silence. For each recording, the start and finish time of the sub-goal section are automatically labelled using the expected characteristics of its feature, and kept in a transcription file. For example, for the "Pour Kettle" and "Add Milk" isolated recordings, the FSR outputs were used to define the start and finish times as the moments of picking up and putting down the kettle and milk container, respectively. Isolated recordings along with their transcriptions, are used to train sub-goals and background models in each detector.

For the testing (evaluation) part of the sub-goal spotting experiments in each detector, all 100 full trial recordings of making a cup of tea are used. The execution time of each individual sub-goal in a full-trial recording, is manually labelled using the video data from the Kinect camera. These time labels will be matched against the output of the detectors to evaluate the performance of the system. The total number of times that each sub-goal is performed in all full-trial recordings is different and it has been broken down in Table 3.2.

7.3.2 MODEL TRAINING

As with the verification system, two types of Hidden Markov Models are trained to present sub-goals and background data in each detector. The number of states used for a sub-goal LR-HHM is taken from the HMM-set with the highest performance in the verification experiments; however for GMM background models, a different number of mixture componenrs $M \in \{ 1, 2, 8, 16, 32, 64, 128, 256, 512 \}$ has been used in the experiment. The only difference between the models in the spotting and verification systems, is the amount of data available for training each model. Due to the left to right structure of sub-goal models, it seems that the optimal number of states corresponds to the characteristic of the sub-goal (e.g. average length of the sub-goal's execution time), not the amount of data. On the other hand, because more data is available for training background models, the optimal number of the Gaussian mixture model components, which can best fit the background data needs to be discovered.

The same process of training as for the verification system in Section 7.2.2, is applied in these experiments. The time labelled data corresponding to any sub-goal model was used for initialisation of that sub-goal model. The remainder of the recordings are used to estimate the mean and diagonal covariance matrix of a single Gaussian PDF. The iteration of splitting

Gaussian mixture components and Baum-Welch training are executed, until the probability of the training data given models is locally maximised.

7.3.3 EVALUATING THE SPOTTING SYSTEM PERFORMANCE

In this stage of the experiment, full-trial recordings of making a cup of tea are decoded into a sequence of models in each detector, using Viterbi decoding Section 2.6. The decoder network of models is defined with a model-loop structure; this mean all sequences of models are allowed in the output of the Viterbi decoder (see Section 6.3). Note in this case, insertion penalties placed on the transition between two consecutive model can be used to gain fine control over the false-alarm rate. The output from the detector, in addition to the sequence of models, includes the start and finish times of each model.

To evaluate the result of the decoding, the time boundary of each target sub-goal label W in the output, is compared with the reference transcription. If the mid-point of the W lies between the start and end time of an identical label in the reference transcript, then that label represents a correct detection, otherwise it is an insertion (False Alarm). Any more than one detection for the same label in the transcript is an extra insertion. Using the total number of target sub-goals from Table 3.2, correct detections, and insertions, the performance of the spotting system (in terms of accuracy percent or error rate percent) can be calculated from Equation 7.2.

$$\%Accuracy = 100 - \%ER = \frac{Correct - Insertions}{Total} \times 100 \tag{7.2}$$

7.4 REAL-TIME SUB-GOAL SPOTTING

The action recognition system developed in this thesis is used as part of the CogWatch system, which aims to monitor the patient's progress through the ADL and provide the appropriate guidance. For the system to be able to prompt the patient with appropriate cues on time, the process of recognising the individual action that makes up the ADL, needs to happen in real time. Due to the nature of any real-time system and the inevitable processing time, there is always a time delay between the completion of the action and the appearance of the feedback in the output. The goal of this experiment is to calculate and minimize the output delay, using the HMM models trained in the full-trial sub-goal spotting experiment described in section 7.3.

Data set

The full-trial recordings of making a cup of tea, from the tea-making data set (section 3.4) are used as an input of detectors. In order to simulate the tea making task in real time, full-trial recordings are sent to the detector one frame at a time. Readings from all the sensors are synchronised at a 50 Hz sampling frequency, and the features are calculated over the observation window of 400 ms (20 samples) length with 20 ms (50 Hz frame rate) steps.

7.4.1 DETECTION OF SUB-GOALS IN REAL TIME

For real-time decoding of the input data into a sequence of models, the modified real-time Viterbi algorithm (explained in section 5.4) is used instead of the conventional Viterbi algorithm. In contrast to the conventional Viterbi algorithm where the length of the observations in the input is known, in real-time decoding the detector doesn't have any prior information about the possible length of the input trial, or time boundaries in which any sub-goal may

start or finish in that trial. In order to make decisions about the progress of the trial, the detector applies a partial trace-back algorithm every 20 frames (400 ms).

The partial trace-back is successful at time t if all active paths in the Viterbi decoder up to time t converge at some time t_0 ($0 > t_0 > t$). this means that anything happening after time t will not have any effect on the recognition decision up to time t_0; so the most likely sequence of models up to time t_0 can be output. Between the time t_0 when a sub-goal is finished and the time t when the sub-goal is detected and appeared on the output of the system, there is an inevitable delay.

7.4.2 OUTPUT DELAY IN REA-TIME DECTOR

Full-trial recordings are decoded in each detector, using a real-time Viterbi algorithm and HMM models trained during the full-trial sub-goal spotting experiment 7.3. For any detection of a sub-goal, the delay between the recognition and output time is calculated.As described in previous section, the output delay is the extra number of frames that takes for the real-time Viterbi decoder to detect the end-time of the sub-goal execution, due to the nature of partial trace-back algorithm when then end of the file is unknown. This delay for a single detection of a sub-goal is a multiple of the sampling interval (which is 20 ms in the case of CogWacth system) .The mean and variance of delays for each sub-goal are presented in Table 8.4. From this table, the average delay among all detection of sub-goals is 2.621 seconds with a variance of 1.77 using original trained HMMs (Section 7.4.3). It needs to be mentioned that, the amount of delay does not depend upon the processing power of the computer or speed of the memory, but it does depend on the similarity of the data and HMMs, and may depend on path pruning. The reason behind the long delay could be the similarity between the models; this makes the decision harder for the decoder. To address this issue, the following methods are used in different stages of the action recognition system:

- **Feature selection**: using a collective choice of sensor data and features based on the target sub-goal in each detector, helps sub-goal and background models to be more discriminative. The aim was also to calculate features which have a distinct characteristic at the end of the sub-goal, to highlight the difference during the transition between the sub-goal and the background data.

- **Training**: initializing parameters of sub-goal models prior to embedded re-estimation, using a time-labelled transcript of isolated recordings. Each isolated recording starts and finishes with silent ("toying") data with the execution of a sub-goal in-between. Labelling the time of the sub-goal execution and using it for initializing the sub-goal model, results in more accurate alignment of models and training data, between iteration of embedded training.

- **Decoding**: in real-time Viterbi algorithm, the use of beam pruning forces paths with a low probability, into an inactive state. Paths with very low probabilities can delay the detection output for a sub-goal severely, to the point that the detection of a sub-goal would not appear in the output until the start of the sub-goal's execution for the second time.

The average delay of 2.621 seconds in output was more than expected for this task. After debugging the process of the sub-goal's detection in the C# code of the real-time Viterbi algorithm, it was realized that decoding paths that end up in the final states of a sub-goal model can remain in that state long after the sub-goal execution is finished. Looking into sug-goals and background HMMs, it is clear that the Gaussian PDFs forming late states of sub-goal models are similar to those in background ("toying") models.

7.4.3 MODIFIED SUB-GOAL HMMS

Execution of a sub-goal finishes by fading into a sensor's inactivity state; thus, the similarity between the end states of a sub-goal model with a "toying" action is inevitable, especially during the iteration of embedded training where parameters of HMMs are optimised based on the alignment of the training data with the model's states.

The manual modification of the sub-goal models is done when the training is finished. Modification is achieved by changing the state transition probabilities for states that caused the delay. Sub-goal models are left to right HMMs, with a state network including a self-loop transition for all states and a one-step transitions between two consecutive states; i.e. from time $t-1$ to t, a model can decide to either stay in the same state i or move to the next state $i+1$. Deleting the self-loop transition for states similar to background data, will stop the model from staying too long in these states; so the recognition path in the Viterbi algorithm will exit the sub-goal model as soon as it reaches the final stage of the sub-goal. The state network of a sub-goal model before and after the modification is illustrated in Figure 7.1.

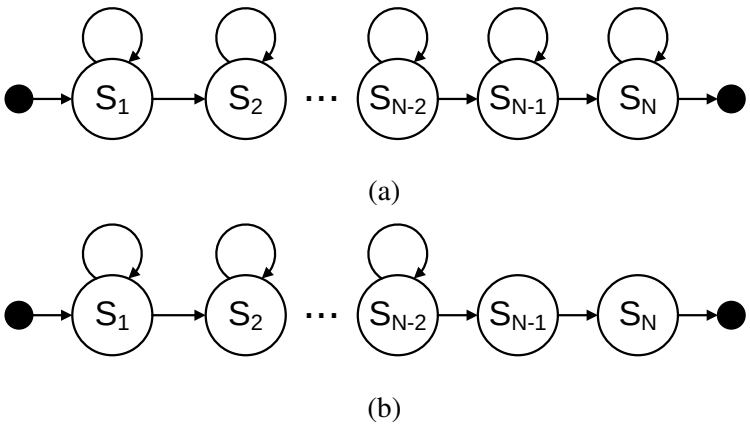


Fig. 7.1 *State network of sub-goal models (a) before and (b) after the modification*

After applying changes into the state transition matrix of the sub-goal models, full-trial recordings are decoded using the real-time Viterbi algorithm and modified HMMs, to

make sure the spotting performance of the system is maintained after modification. The performance of the sub-goal spotting in all the detectors remained the same except for the "AddMilk" detector, in which one instance of previously correct detection of adding milk into the mug was missed (extra deletion error). The reason for this may be due to the shorter length of the "AddMilk" model (only five states), compared to other sub-goal models.

The calculated time of delays in the real-time detection of each sub-goal is summarized in Table 8.4 and can be compared with those calculated from using the original models. Applying the modification to sub-goal models, reduced the average time delay among all detected sub-goals down to less than one second.

7.5 SUMMARY

In this chapter, the process of building a set of HMM models to be used in a real-time sub-goal detector, was explained through completion of three experiments, namely sub-goal verification, sub-goal spotting and real-time sub-goal spotting.

The isolated recordings of sub-goals from the tea-making data-set, are used in the verification experiment to find the optimal number of states to be used for sub-goal models of a detector. In the sub-goal spotting experiment, all recordings in the tea-making data-set and optimal number of states found in the verification experiment are used to train and optimise the final HMM-set for each detector.

In the last section, the reason for having a detection delay when using the real-time Viterbi algorithm, is introduced. Also, a manual modification for HMMs is proposed, which can reduce the detection delay-time of the system.

CHAPTER 8

GMM-HMM EXPERIMENT RESULTS

8.1 INTRODUCTION

In this chapter, the results of the experiments described in Chapter 7, is reported for each individual detector. Each experiment was carried out for all the detectors by using the feature-vector and HMM-set defined in Chapter 6. The HTK speech recognition toolkit [140] is used for training HMMs in this chapter.

In section 8.2, the effect of using HMM-sets with different numbers of parameters on the detection of sub-goals and a summary of the results for the sub-goal verification experiments for each detector are investigated. From these results, the optimal HMM parameters and suitable sensor data features, to be used in sub-goal spotting experiments are obtained.

In section 8.3, by using the obtained parameters from the verification experiments, new models are trained to evaluate the performance of the system in detecting a sub-goal during the full-trial of making a cup of tea (spotting task).

Finally in the last section, the original trained models in the sub-goal spotting experiments and the modified version of them, are put to test in the real-time detector to measure the output delay for each sub-goal detection.

8.2 SUB-GOAL VERIFICATION RESULTS

In this section, first the results of the development stage of the verification experiment for each detector are discussed in sections 8.2.1 to 8.2.5. Then the final results of the verification task (test stage of the experiment) are summarized in section 8.2.6 .

8.2.1 "POUR KETTLE" DETECTOR

Figure 8.1 shows the effect of using a various number of states in the HMM sub-goal models ("Pour Kettle" and "Toy Kettle") and an increasing number of GMM components in the "toying" model, on the verification performance of the "Pour Kettle" detector. The results are expressed in terms of the recognition error-rate that was calculated using Equation 7.1. In this detector, there is no insertion penalty for transition between the models.

The results are obtained from the recognition of trials in the development set of a five-fold cross-validation, which all together consists of 144 recordings of pouring water from a kettle into the mug; 186 trials of toying with the kettle; and 1305 recordings of other sub-goals. The challenge in this detector is to differentiate between sub-goals in which the kettle is used to pour water into the mug and the complimentary sub-goal recordings in which the kettle is used, but water is poured intentionally outside of the mug. These recordings produce very similar sensor data; the only difference is the weight of the mug which increases when the water is poured into the mug.

The highest accuracy (with no error) was achieved using 60 states for the sub-goal models and 64 GMM components for the "toying" model. Although, sub-goal models with 70 states can achieve the same performance, 60 states were chosen to be used in the remaining experiments, due to performance consistency while increasing GMM components from 8 to 128.

A plot of the sensor outputs, during pouring water from the kettle into the mug (an example of "Pour Kettle") is shown in Figure 4.3.

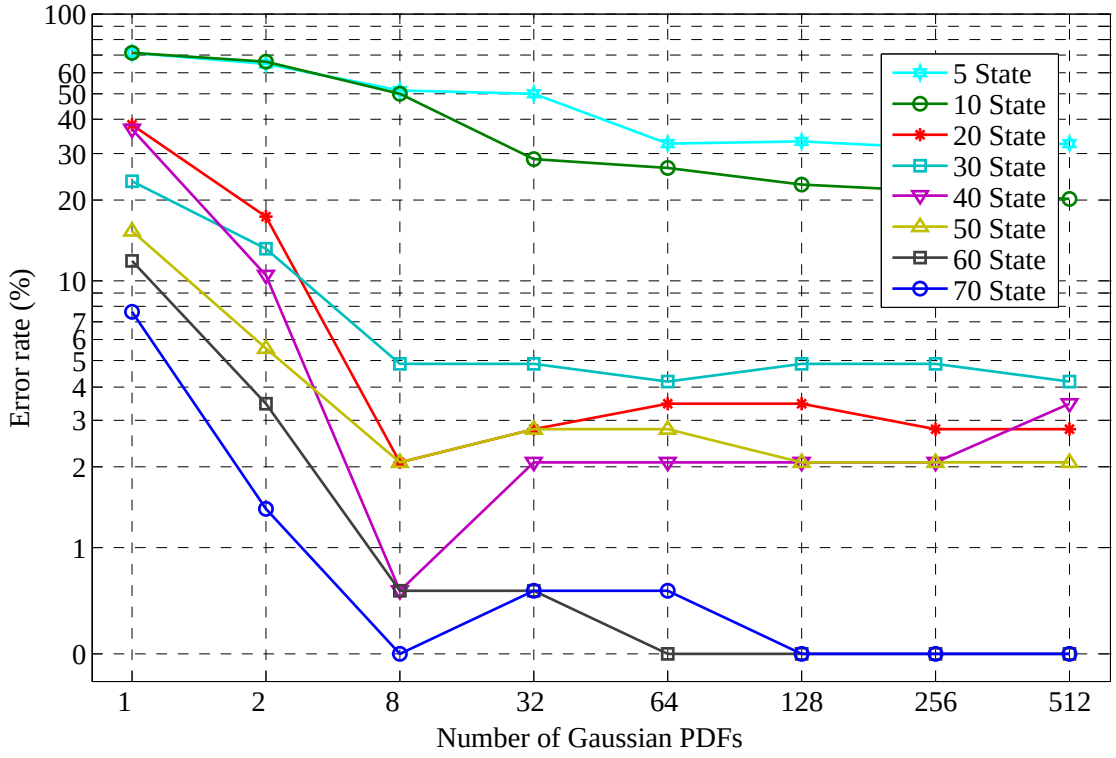


Fig. 8.1 *"Pour Kettle" sub-goal verification system performance*

8.2.2 "ADD MILK" DETECTOR

In this section, the results from the development stage of the sub-goal verification experiment (see section 7.2) for the "Add Milk" detector, are discussed. As explained in section 7.2.1,

each isolated recording of a sub-goal appears only once in a development set of a five-fold cross-validation, which all together consists of 91 recordings of adding milk from the container into the mug ("Add Milk" sub-goal); 134 recordings of toying with the milk container ("Toy Milk" sub-goal); and 1508 recordings of the remaining sub-goals in the tea-making task.

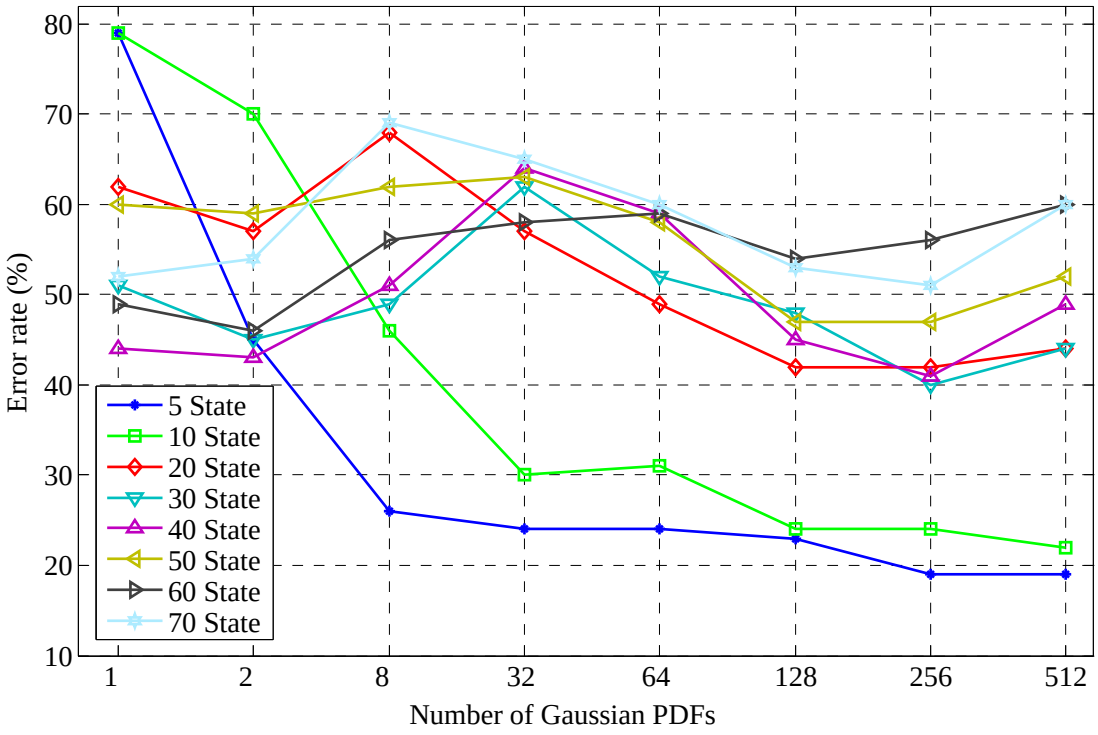


Fig. 8.2 *"Add Milk" sub-goal verification performance*

Figure 8.2, presents the verification accuracy of "Add Milk" trials in the development set, which was calculated using Equation 7.1. Detection of sub-goals in the development set is repeated using HMM sets with a different number of states and GMM components for the sub-goal models ("AddMilk" and "ToyMilk") and the background ("toying") model, respectively. Also, for each set of models, the value of the sub-goal insertion penalty p ($0 \leq p \leq 1000$) is tuned in the decoder.

The difference between "Add Milk" and "Toy Milk" sub-goals' data, is the increase in the weight of the mug when milk is added. The weight of the milk added to the mug, which in some trials is too small to be distinguished, refers to the results of the FSRs, sensor sensitivity test in Figure 4.2. Due to this similarity, increasing the number of states in sub-goal models, seems to only intensify the confusion and reduce the accuracy. The lowest error rate of 19% is achieved by using an HMM-set with 5-states sub-goal models and a background model consisting of 256 GMM components.

Figure 8.3, shows the effect of the insertion penalty on the verification error rate, using fixed 256 GMM components for the background model. From the figure, it is clear that the insertion penalty has more effect when the sub-goal model is shorter. The probability of generating a sequence of observation for given model λ ($P(O|\lambda)$) is calculated by accumulating the state output probabilities, and the insertion penalty is a fix log probability added to P, therefore for shorter models (with a fewer number of states) the difference between these two probabilities is less.

8.2.3 "FRONT ACTIONS" DETECTOR

The "Front Action" detector is responsible for detecting isolated recordings of three target sub-goals ("Add Sugar", "Add Teabag" and "Remove Teabag") among all trials in the development sets of the verification experiment. The development sets consists of: 207 trials of adding sugar into the cup ("Add Sugar" sub-goal), 223 trials of adding a teabag into the mug ("Add Teabag" sub-goal), 154 trials of removing the used teabag from the mug into the empty container ("Remove Teabag" sub-goal), and 926 trials from the remaining sub-goals in the tea-making task.

In a HMM-set used in the "Front Actions" detector, the same number of states $N \in \{ 5, 10, 20, 30, 40, 50, 60 \}$ is used for all three sub-goal models; while increasing the number of

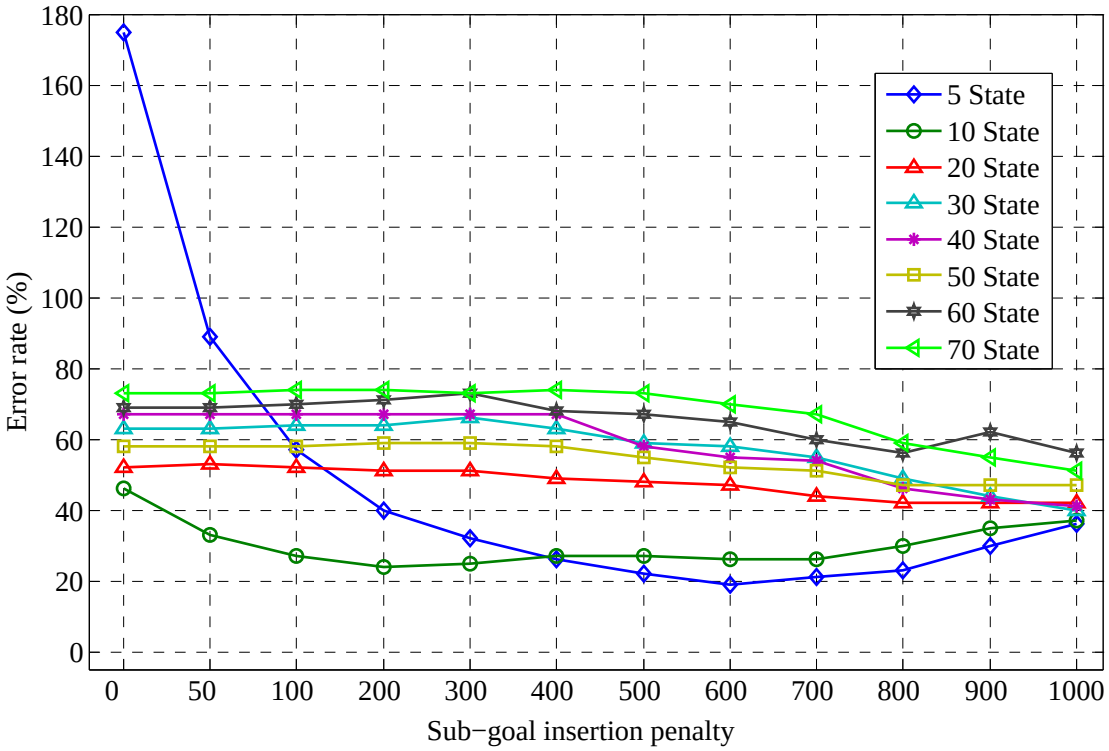


Fig. 8.3 *Effect of changing sub-goal insertion penalty on "Add Milk" verification performance with fixed 256 GMM components for "toying" model*

GMM components $M \in \{ 1, 2, 8, 32, 64, 128, 256, 512 \}$ for the background model. Figure 8.4, shows the error rate of the sub-goal verification in the "Front Actions" detector, using HMM-sets with different numbers of parameters. The lowest error rate of approximately 4% is achieved using 30-states sub-goal models and a background model consisting of 8 GMM components.

With a fixed 8-GMM background model in a HMM-set, increasing the number of states in the sub-goal model from 5 to 30 leads to an error rate reduction; using more than 30 states results in a performance decrease. This trend can be explained by the very short time that the hands appear in the vicinity of the sugar, tea bag and rubbish container, for execution of three "front" sub-goals; thus it's believed that the sub-goal models in the "Front Action" detector need to be shorter compared to other sub-goals.

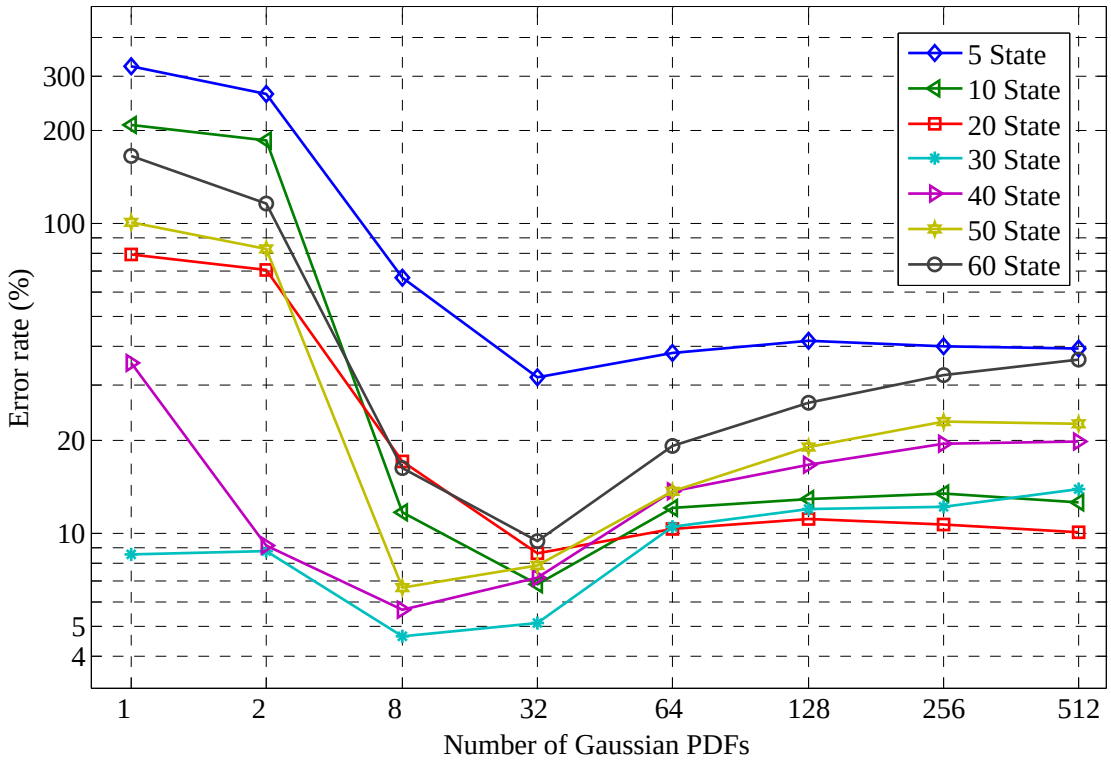


Fig. 8.4 *"Front Action" sub-goal's verification performance*

8.2.4 "STIR" DETECTOR

As it mentioned in section 5.2, the "Stir" detector uses recorded data from the Kinect and Mug-CIC sensors for detection of the stirring action. From the Kinect data, two similar features (hard and soft elliptical neighbourhoods around the mug) are calculated and explained in section 4.5. In this section, the effect of using hard and soft neighbourhoods on the verification error-rate, while changing the number of states in the "Stir" sub-goal model, is discussed. All results presented in this section are maximised by optimising the number of GMM components $M \in \{ 1, 2, 8, 32, 64, 128, 256, 512 \}$ in the background model and the insertion penalty p ($0 \leq p \leq 1000$) in the decoder network, during the model training and recognition process, respectively.

Hard neighbourhood feature

Figure 8.5, shows the effect of the size of the hard neighbourhood feature on the verification of the stirring sub-goal, using a various numbers of states $N \in \{ 10, 20, 30, 40, 50, 60, 70 \}$ in the "Stir" sub-goal model.

From looking at the graph it can be seen that the first locally minimum error-rate of 23.6% occurrs for the neighbourhood feature with the radius of 10.7 centimetres using a 60-state sub-goal model. The lowest verification error rate of 18.6% is achieved for the neighbourhood feature with a radius of 21.4 centimetres, using a 40-state sub-goal model, a background model consisting of 512 GMM components, and the insertion penalty of 700. As mentioned previously, the stirring sub-goal is executed using a spoon that is located on the table, approximately 10 centimetres away from the mug. The position of the mug and spoon on the table justifies the two troughs of the graph at 10.7 centimetres radius, which is large enough to cover the mug but not the spoon; and 21.4 centimetres radius, which covers both the mug and spoon.

Soft neighbourhood feature

Figure 8.6 illustrates the stirring verification performance using the soft neighbourhood feature around the mug with different radius sizes. From looking at the graph, it's clear that use of sub-goal models with a larger number of states (e.g. 60 and 70) results in a lower error rate. The lowest verification error rate of 23.3% is achieved using a 60- and 70-state sub-goal model; these results were obtained for features with 17.1, 19.2, 51.3 and 55.6 centimetres radius.

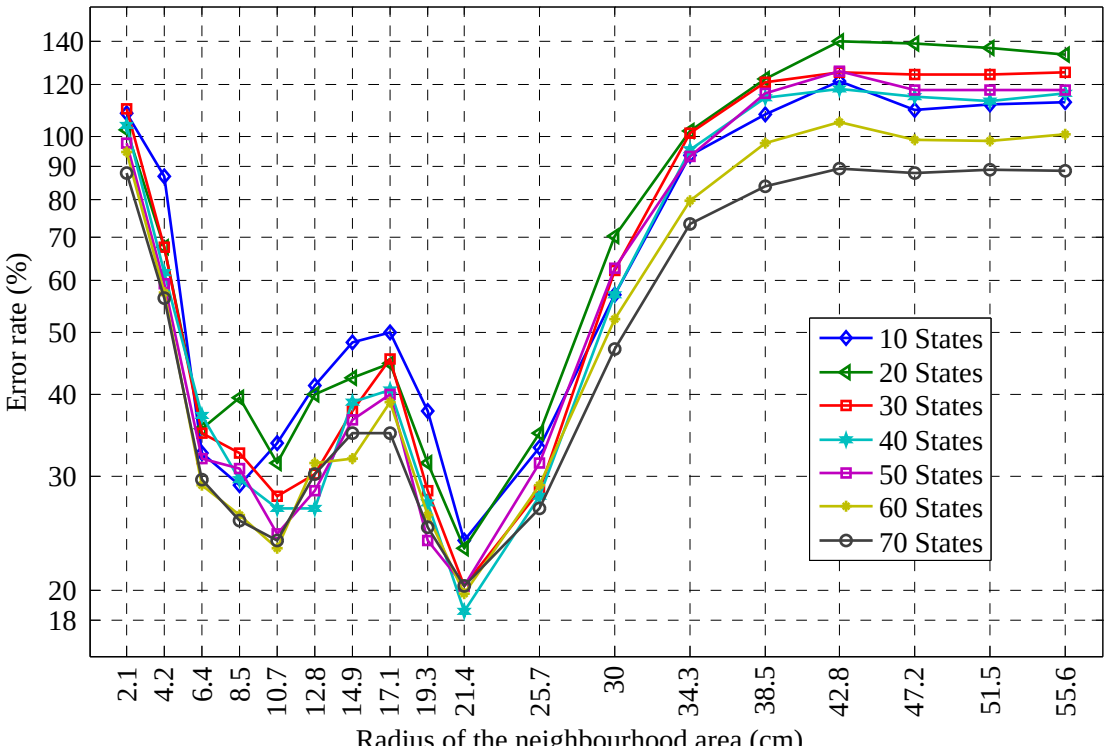


Fig. 8.5 *"Stir" sub-goal verification performance (%ER), using various radius sizes for hard neighbourhood feature*

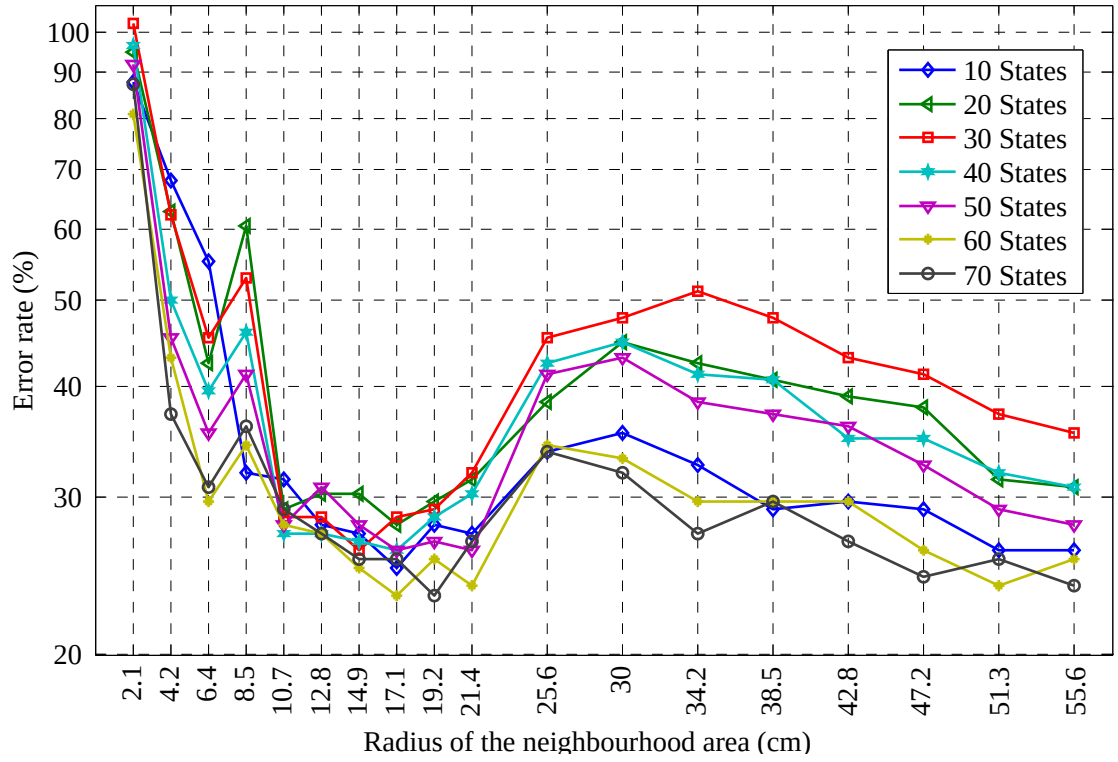


Fig. 8.6 *"Stir" sub-goal verification performance (%ER), using various radius sizes for soft neighbourhood feature*

Hard versus soft neighbourhood

The lowest error rate for both soft and hard neighbourhood features with different radius sizes are compared in Figure 8.7. By referring to the graph it can be seen that performance for the hard neighbourhood method is maximised with a radius of 21.4 centimetres. For a very large radius, where the neighbourhood covers most of the table, and a very small radius, the error rate is increased to approximately 90%. On the other hand, performance for the soft neighbourhood is less sensitive to changes in the radius. This is due to the extra information about the distance between the hand and the mug in the soft feature. From the Kinect data in recordings of the "stir" sub-goal, the location of the hand while stirring was anticipated to be 18 to 25 centimetres around the mug; the error rate trough is well placed in that region.

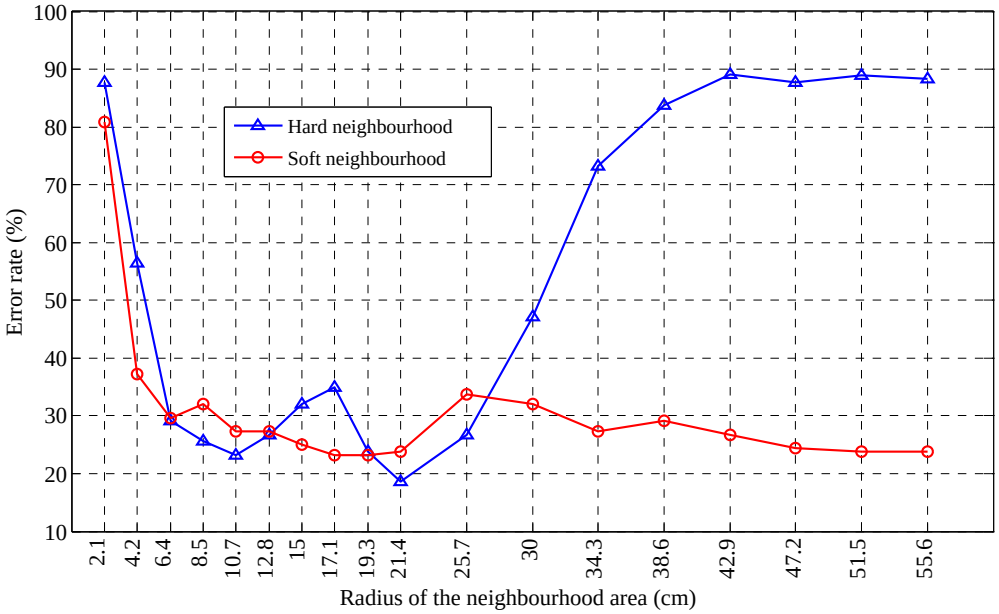


Fig. 8.7 *Comparison of the "Stir" sub-goal verification performance for hard and soft neighbourhood features with various radius sizes*

Also, from this experiment can be concluded that if there is no prior information about the possible hand positions and exhaustive experiments are not viable, a soft neighbourhood with a large enough area can be used to achieve a base-line system for the verification task.

8.2.5 "FILL KETTLE" DETECTOR

In the process of filling the kettle with water from the jug, only the kettle is instrumented with sensors. Thus, the part of the "Fill Kettle" sub-goal that incorporates picking up and putting down the jug, is detected solely by using the Kinect sensor. From the Kinect data, the hard or soft neighbourhood feature (see section 4.5) can be used to indicate when the hand is located in the vicinity of the jug and kettle.

For each set of features with various neighbourhood radius sizes and the number of states $N \in \{10, 20, 30, 40, 50, 60, 70, 80\}$ that are used in the detector, the number of GMM $M \in \{1, 2, 8, 32, 64, 128, 256, 512\}$ in the background model and insertion penalty value p ($0 \leq p \leq 1000$) in the decoder, are fine-tuned.

In this section, the radius of the circle area around the jug r is shown on the horizontal axes of the graphs; however, the neighbourhood area around the kettle is an elliptical shape with a vertical radius of r and horizontal radius of $1.5 \times r$. From looking at the Kinect data for all "Fill Kettle" recordings in Figure 4.6, it is clear that the hand location around the kettle has a larger variability in the horizontal direction.

Hard neighbourhood feature

Figure 8.8, illustrates the verification performance of the "Fill Kettle" sub-goal, using the hard neighbourhood feature with different radius sizes and a various number of states in the "Fill Kettle" sub-goal model. The lowest error rate of 1.6% is achieved for the hard neighbourhood with a radius of 8.5 centimetres; use of a larger radius size leads to a higher error rate for all numbers of states. By referring to the graph, it is clear that the use of sub-goal models with a high number of states (e.g. 60 and 70 states) is beneficial for reducing the errors in the detection of the "Fill Kettle" sub-goal. This was expected, as the "Fill Kettle" sub-goal is the longest of all (Table 3.1).

On looking at the location of the hands around the jug during all ”Fill Kettle” isolated recordings in Figure 4.6, a circle with a radius of 13 centimetres covers all of the hand's data around the jug. However, one explanation for achieving a better performance at an 8.5 centimetres radius, is that the bigger area may overlap with the hand locations in the “Add Milk” sub-goal.

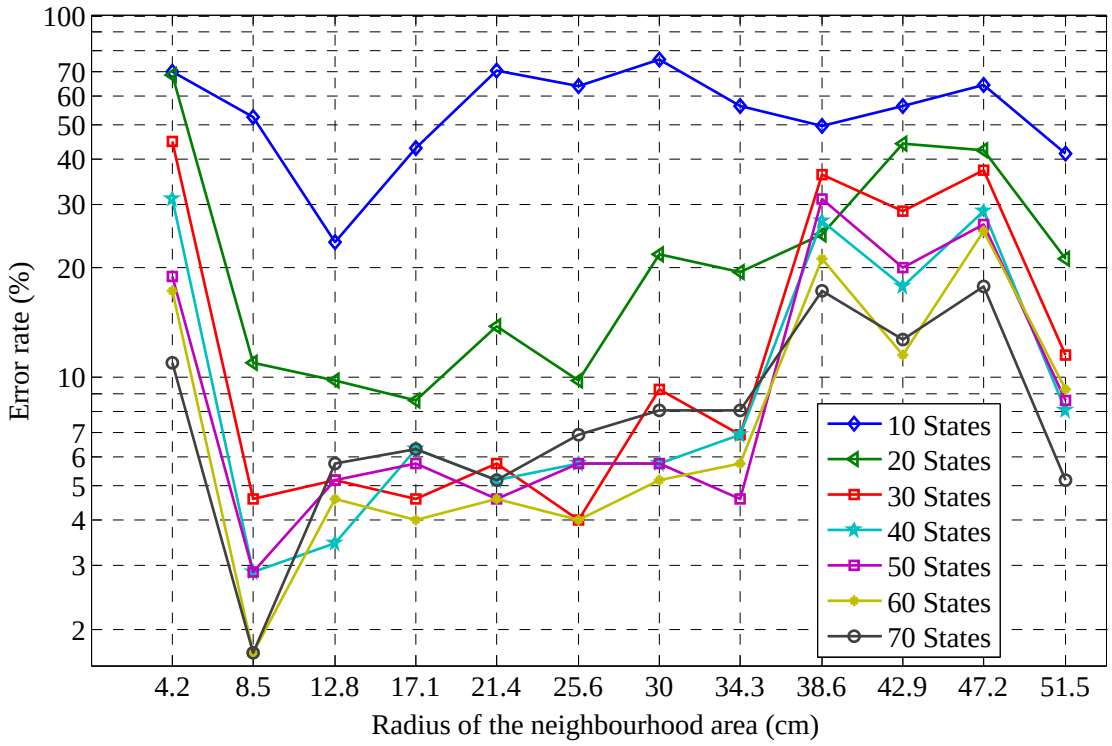


Fig. 8.8 *“Fill Kettle” sub-goal verification performance (%ER), using various radius sizes for the hard neighbourhood feature*

Soft neighbourhood feature

Figure 8.9, illustrates the verification performance of the “Fill Kettle” sub-goal, using the soft neighbourhood feature with different radius sizes, and various number of states in the “Fill Kettle” sub-goal model. The lowest error rate of less than 1% is achieved for the hard neighbourhood with a radius of 38.53 centimetres when the neighbourhood feature of the

kettle reaches to the cluster of the hand's data around the jug (Figure 4.6). In contrast with the hard neighbourhood feature, use of a large radius size did not lead to a higher error rate. Similarly to the hard neighbourhood, there is a local minimum at 8.5 centimetres radius.

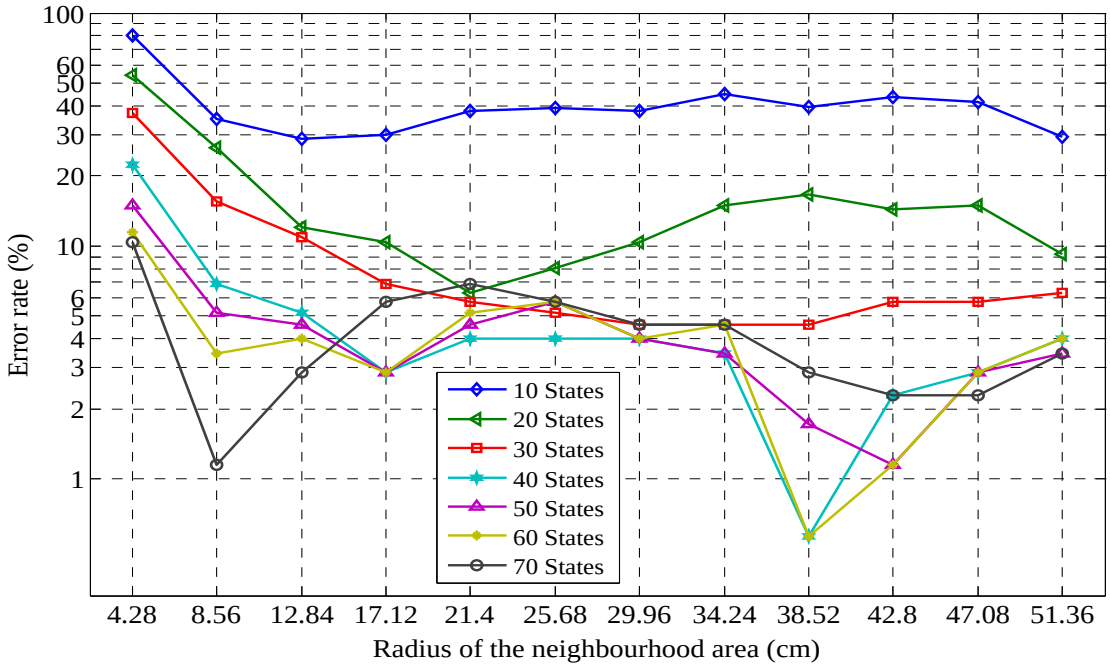


Fig. 8.9 *"Fill Kettle" sub-goal verification performance (%ER), using various radius sizes for the soft neighbourhood feature*

Soft versus hard neighbourhood

Figure 8.10 presents the lowest verification error rate of "Fill Kettle" sub-goals, for each radius size of the hard and soft neighbourhood features. The soft neighbourhood feature achieved a lower performance for all radius sizes.

From Figure 4.6, it is expected that a radius of around 12-15 centimetres is a desirable choice, as it would perfectly fit the hand data when stationary around the jug and kettle; however the experiments proves otherwise. The lowest error rate achieved for the soft

neighbourhood with a radius of 38.5 centimetres, using the HMM-set, consists of a 60-state sub-goal model and a 32-GMM background model.

Although, there are performance boosts near some of the expected radius sizes, the difference in error rate is not much (around 4%); thus, the better performance might not have been caused solely due to the explanations given.

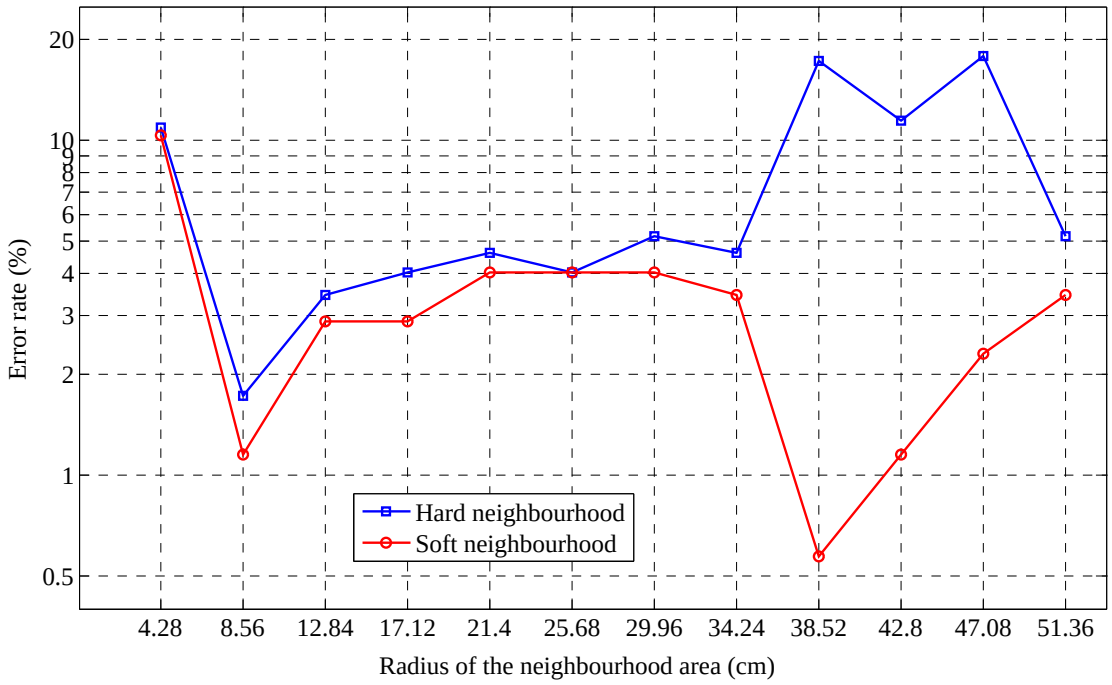


Fig. 8.10 *Comparison of the "Fill Kettle" sub-goal verification performance for hard and soft neighbourhood features with various radius sizes*

8.2.6 FINAL PERFORMANCE OF THE VERIFICATION SYSTEM

In this section, for all sub-goals in the tea-making task-tree, the results of the isolated recording detection are presented and discussed. Isolated recordings of the sub-goals in the test sets, are decoded using the HMM-sets which achieved the highest detection performance on development-set. The recordings in the test set of a five-fold cross-validation, were kept

unseen during the training and development stages of the verification experiment. The total recordings in five test sets of each detector, include all isolated trials explained in section 3.4.

Table 8.1 summarizes detection accuracies of all sub-goals, using five parallel detectors. The "Add Sugar", "Add Teabag", and " Remove Teabag" sub-goals are detected, using the same detector ("Front Action"); the rest of the sub-goals are detected in a detector that is specifically assigned for detection of that sub-goal. For each sub-goal the number of parameters used in the HMM-set of the detector is reported as well.

The best performance, with only one insertion, is achieved for the "Pour Kettle" sub-goal, due to the use of the instrumented coasters for all objects involved in the execution of the sub-goal. The poorest performance with 19.78% error rate belongs to the "Add Milk" sub-goal; the majority of the insertions occur for the recording of the auxiliary sub-goal "Toy Milk" due to the high similarity.

Table 8.1 *Summary of Isolated recordings verification results for all tea-making sub-goals using optimal models*

Sub-goals	HMMs parameters		Detection details			%Error rate
	States	GMMs	Correct	Deletion	Insertion	
Pour Kettle	60	64	114	0	1	0.90
Add Milk	5	256	86	5	13	19.78
Fill Kettle	70	32	168	8	2	5.88
Add Sugar	30	8	204	3	9	5.63
Add Teabag	30	8	217	5	1	2.75
Remove Teabag	30	8	143	10	4	9.52
Stir	40	512	152	20	12	19.52
					Average	9.14

For all sub-goals, very similar detection accuracies are achieved for test sets and development sets; this suggests that the process of training and optimisation of HMM-sets was not biased to the data in development sets. The average error rate of 9.14% is claimed to

be the performance of the verification system for sub-goals in the tea-making task. Based on the discussion in Section 1.1.5, by using a POMPD-based task manger, In the case of the ARS with error rate of 10%, the simulated user adhering to the TM prompts achieves successful task completion in 90% of the trials. Recognition performance with less than 10% is an acceptable result for a rehabilitation system like CogWatch project.

8.3 SUB-GOAL SPOTTING RESULTS

In this section, results of the sub-goal spotting experiment (section 7.3) for all detectors, are presented and discussed. The training data for each detector consists of all isolated recordings of sub-goals; the details of the recordings are available in section 3.4. The number of states for the sub-goal models in each detector is taken from the verification experiments. On the other hand, the number of GMM components for the background model and the value of the insertion penalty in the decoder, are optimised to achieve the lowest error rate possible.

During the verification experiment for the "Fill Kettle" and "Stir" detectors, the size of the neighbourhood features were adjusted; however the results were not conclusive as to which radius size is optimal to use in the spotting task. Hence, the spotting task for these detectors are repeated with different neighbourhood sizes.

8.3.1 "STIR" DETECTOR

Figure 8.11 and 8.12, show the effect of changing the size of the soft and hard neighbourhood features, respectively. The number of states in the sub-goal model is fixed at 40 (taken from the verification experiment), and the number of GMM components are varied.

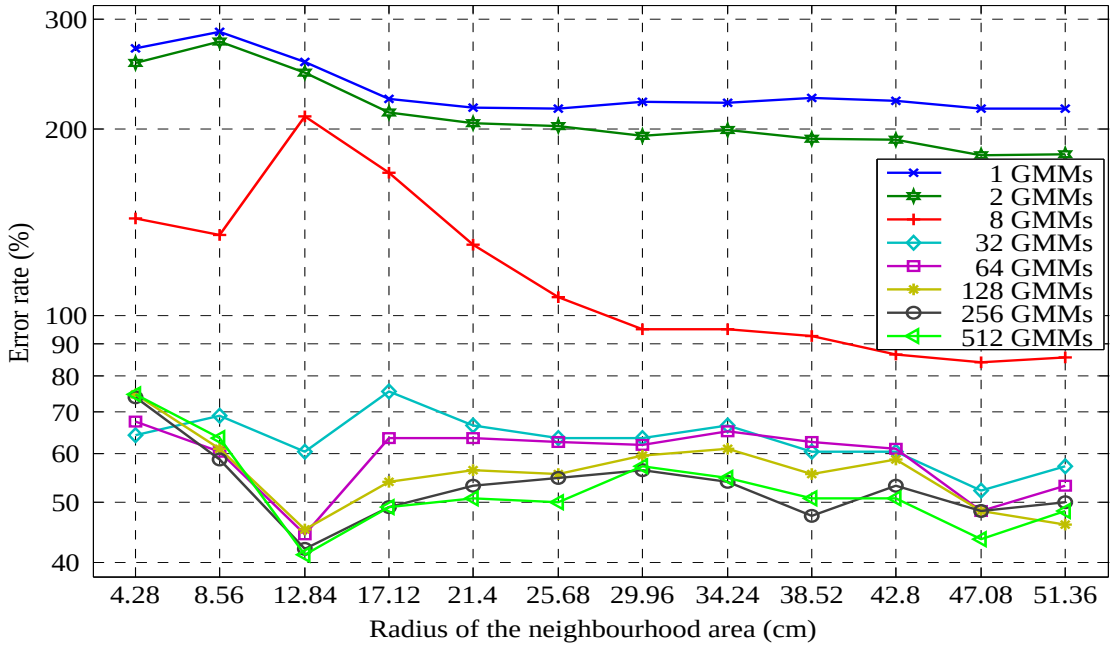


Fig. 8.11 *"Stir" sub-goal spotting performance (%ER), using various radius sizes for the soft neighbourhood feature*

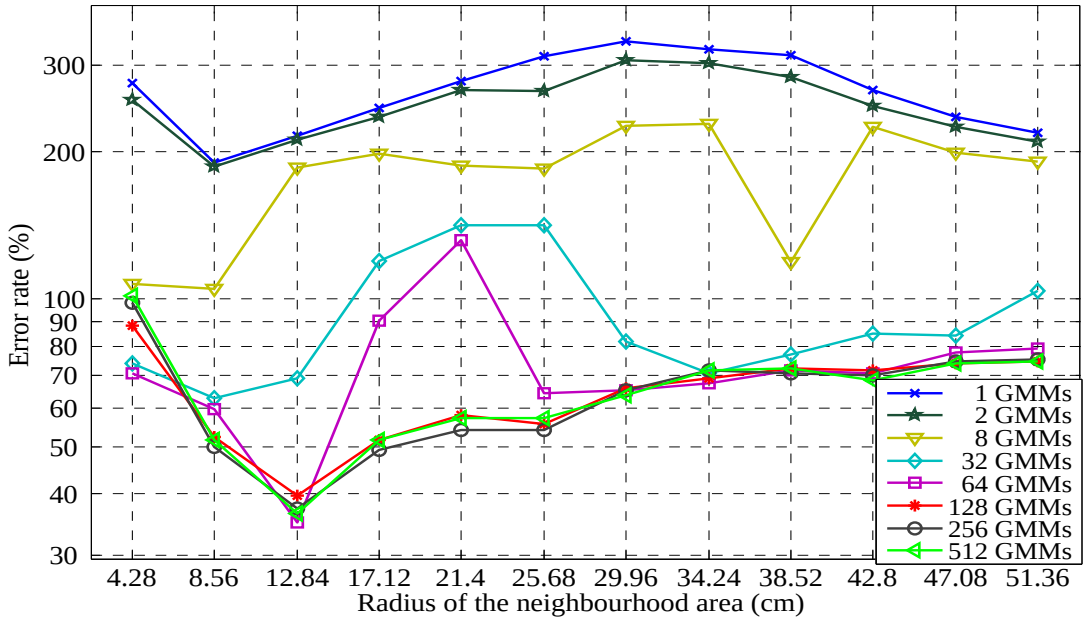


Fig. 8.12 *"Stir" sub-goal spotting performance (%ER), using various radius sizes for the hard neighbourhood feature*

For both features the minimum error rate occurred at a radius size of 12.8 centimetres. The lowest error rates for the soft and hard neighbourhoods are approximately 41% and 35%, respectively. Comparing the hard neighbourhood results with Figure 8.5 from the verification experiment, it can be seen that no performance improvement is gained by including the spoon in the mug neighbourhood area at a radius of 21.4 centimetres.

8.3.2 "FILL KETTLE" DETECTOR

Figures 8.13 and 8.14, show the effect of changing the size of the soft and hard neighbourhood features, respectively. The number of states in the sub-goal model is fixed at 70 (taken from the verification experiment), and the number of GMM components are varied.

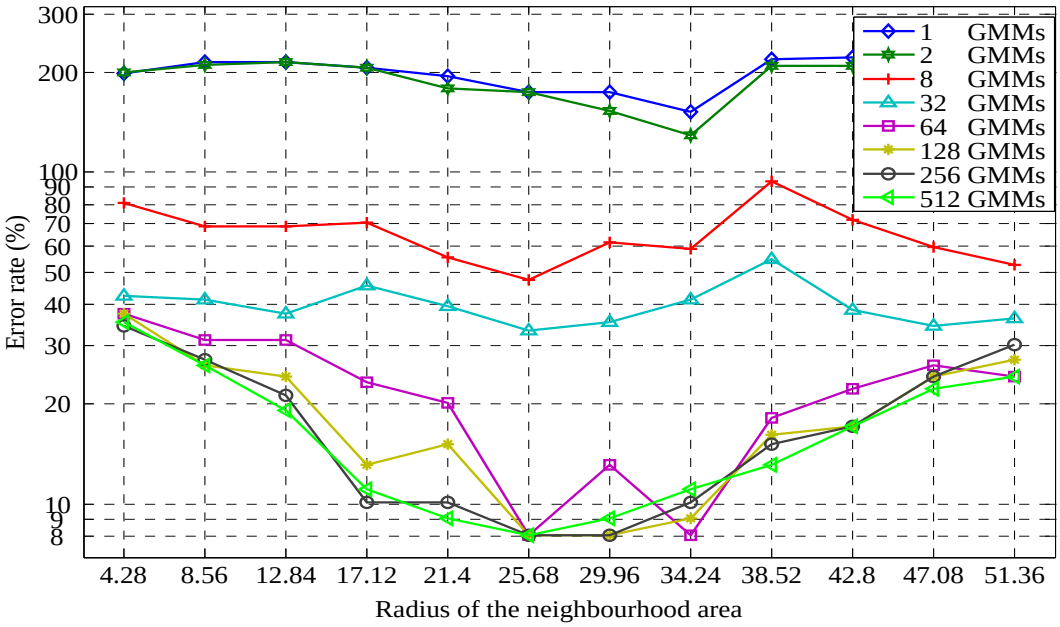


Fig. 8.13 *"Fill Kettle" sub-goal spotting performance (%ER), using various radius sizes for the soft neighbourhood feature*

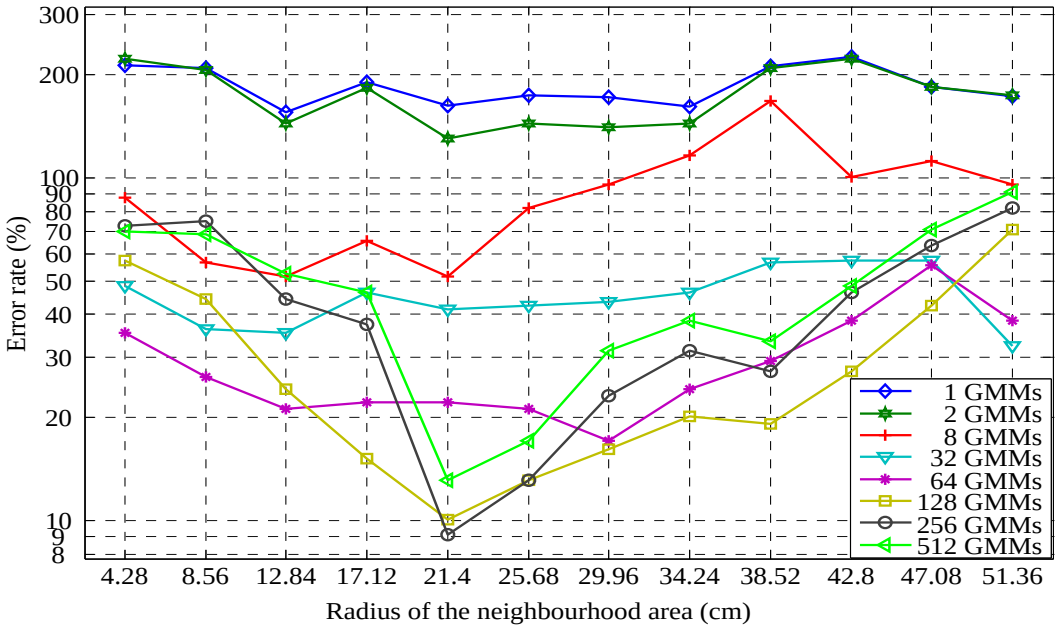


Fig. 8.14 *"Fill Kettle" sub-goal spotting performance (%ER), using various radius sizes for the hard neighbourhood feature*

Unlike the verification system the minimum error rate didn't occur at 8.5 nor 38.5 centimetres radius. The best performances belong to the features with the radius size between 21.4 and 34 centimetres. The lowest error rate of 8.43% is achieved using a soft neighbourhood with the radius of 25.6 centimetres and the background model consists of 512 Gaussian mixture components.

8.4 FINAL PERFORMANCE OF THE SPOTTING SYSTEM

Full-trials of making a cup of tea, are decoded into sequence of models in each detector and spotting performance are evaluated by comparing decoded time boundaries with the reference transcript of the full-trials. The best spotting performance for all sub-goals are summarised in Table 8.2. Furthermore, the Table contains information about the number of parameters in the models that used to achieve these results. Similar to the verification

system, the lowest error rate of 5.27% is achieved for the "Pour Kettle" sub-goal. The poorest performance is for the "Stir" sub-goal, with 34.93% ER.

Table 8.2 *Summary of sub-goal spotting results for all tea-making sub-goals using optimum HMMs*

Sub-goals	HMMs parameters		Detection details			%Error rate
	States	GMMs	Correct	Deletion	Insertion	
Pour Kettle	60	32/64	92	3	2	5.27
Add Milk	5	256	50	6	4	17.86
Fill Kettle	70	512	88	7	1	8.43
Add Sugar	30	32	79	11	6	18.89
Add Teabag	30	32	83	11	2	13.83
Remove Teabag	30	32	89	6	10	16.17
Stir	40	64	108	18	26	34.93
					Average	16.58

Comparing Table 8.1 and 8.2, The number of GMM components is higher in spotting experiments, due to the increase of training data for optimization of the background model. Also, it is evident that the absence of information about sub-goal start and end times makes action recognition much more challenging, which is a relevant problem in real applications. Because of this, the detection error rate is increased for all sub-goals, apart from the "Add Milk" sub-goal that gained 2% ER, because, there are no instances of confusing the action of toying with the milk-container with the sub goal in full-trial recordings. The average accuracy of 83.42% (16.58 % ER) is claimed to be the performance of the spotting system for all sub-goals in the tea-making task.

Table 8.3 *Full detail of sub-goal detection during the full-trial recordings in form of a confusion matrix (SIL refers to the sensors' silent state)*

Sub-goal	P.K	A.M	F.K	A.T	R.T	A.S	Stir	SIL	INS
Pour Kettle	92	0	0	0	0	0	1	1	0
Add Milk	1	50	0	0	0	0	0	0	3
Fill Kettle	1	0	88	0	0	0	0	1	0
Add Teabag	0	0	0	83	1	0	0	1	0
Remove Teabag	0	0	8	1	89	0	0	1	0
Add Sugar	0	0	0	1	0	79	2	2	1
Stir	1	0	0	0	15	2	108	8	0
Deletion	3	6	7	11	6	11	18	-	-

Table 8.3 shows the details of deletions and insertions for all sub-goals on the full-trial recordings, in the form of a confusion matrix. The column labelled as SIL, indicates the insertion of the sub-goal when the data was a "silence". Also, the column labelled as INS, shows the actual insertion of the sub-goal, i.e., one execution of a sub-goal in a recording is detected two times (one of the detections is correct and the other one is an insertion).

The poor performance of the "Stir" sub-goal detection, can be explained by the 57% (15 out of 26 insertions) confusion with the beginnings of the "Remove Teabag" sub-goal. Since removing a tea bag from the mug include putting the spoon into the mug and moving it to pick up the teabag, the outputs of the CIC-Mug and the Kinect hand coordinates behave very similarly to those for stirring. It needs to be noted that, when the final goal of the trial was making a cup of tea, participants tend to move and stir the teabag inside the boiled water more, in order to get more flavour out of the tea, where as in isolated trials of "Remove Teabag" they simply took the teabag out of the mug and placed it in rubbish container.

8.5 REAL-TIME OUTPUT DELAY

As mentioned in Section 7.4, real-time detections of sub-goals appear in the output of the AR system with a large time delay, when the HMM-sets trained in the spotting experiments are used. In the same Section, a manual fix is proposed that can be applied to the transition probability matrix of HMMs in order to reduce the output delay time. The same sub-goal spotting performance as Table 8.2 is achieved using the modified HMM-sets, which assure that the modification will not effect the detection performance of the detectors.

Table 8.4 shows the mean and variance of the output delays among all detections of a sub-goal in the full-trial recordings, using the real-time Viterbi algorithm. The detections of sub-goals are repeated using the models trained in the spotting experiment and the modified models. Modification to the HMM-sets reduced the average output delay of detections for all sub-goals. The maximum 0.16 and minimum 3.74 seconds improvement, was achieved for detection of the "Remove Teabag" and "Pour Kettle" sub-goals, respectively. Also, the modification to the HMMs reduced the average delay-time of outputs from 2.6 seconds to an acceptable delay of less than one second.

8.6 SUMMARY

In this chapter the verification and spotting task were repeated for all sub-goals. For the verification task, the average error rate (ER) of 9.15% is achieved amongst all sub-goals. The lowest and highest ER of 0.9% and 19.78% are belong to the "Pour Kettle" and "Add Milk" sub-goals, respectively.

Table 8.4 *Detection's output delay for each sub-goal using the original and modified HMMs*

Models	Sub-goal Output Delay Time (seconds)			
	Before Fix		After Fix	
	Mean	Variance	Mean	Variance
Pour Kettle	4.553	1.836	0.804	0.459
Add milk	2.025	0.939	0.314	0.321
Fill Kettle	3.940	3.054	1.695	1.695
Add Sugar	2.206	2.474	1.059	0.837
Add Teabag	2.663	1.808	0.264	0.435
Remove Teabag	0.737	0.784	0.572	0.339
Stir	2.834	1.543	0.986	0.724
Average	2.647	1.776	0.813	0.583

For spotting task, the average ER of 16.58% is achieved amongst all sub-goals. The lowest and highest ER of 5.27% and 34.92% are achieved for the "Pour Kettle" and "Stir" sub-goals, respectively.

The spotting task was repeated using the original and modified HMMs (section 7.4.2) in real-time decoder; the average output delay for all detections of sub-goals reduced from 2.64 to 0.81 seconds. The maximum 0.16 and minimum 3.74 seconds improvement, was achieved for detection of the "Remove Teabag" and "Pour Kettle" sub-goals, respectively.

CHAPTER 9

DNN-HMM SYSTEM

9.1 INTRODUCTION

In recent years, deep neural networks (DNNs) have gained the attention of researchers with their recognition performance in pattern recognition applications. In this research, for the detection of the sub-goals in the tea-making task, hidden Markov models (HMMs) are used to represent the temporal variability of different activities.

In this part of the research, instead of using Gaussian mixture models (GMMs), DNNs are employed to represent the emission distribution of HMMs. However, for fine-tuning the DNNs to fit the target HMM states, a baseline GMM-HMM system is used to produce the forced alignment of the training data. The Kaldi open source toolkit [141] is used to build the baseline GMM-HMM and DNN-HMM sub-goal spotting systems.

In this chapter, first the procedures of building the GMM- and DNN-HMM sub-goal spotting systems are discussed in section 9.2. In addition, important differences between the Kaldi and HTK toolkits' training methods are explained. In section 9.3, the results of the baseline GMM-HMM sub-goal spotting system are reported and compared with the results

from section 8.3. Lastly, the final performance of the DNN-HMM system is reported and compared with the GMM-HMM system.

9.2 PROCEDURE OF BUILDING THE DNN-HMM SYSTEM

The DNN-HMM sub-goal spotting system for each detector, is trained and tested on the isolated sub-goal and full-trial recordings in the tea-making database, respectively (section 3.4). The feature-vectors, HMMs' structure and decoding network of each detector, are introduced in Chapter 6. These specifications of the detectors are used, to train the GMM- and DNN-HMM models in the following sections.

9.2.1 GMM-HMM system

The GMM-HMM models are trained, using the optimal number of states for the sub-goal models and the most favourable radius size for the neighbourhood features, which has been established in section 8.3. The process of training GMM-HMM models in the Kaldi toolkit is principally similar to the HTK, with some differences which are listed below:

- Instead of the Baum-Welch (forward-backward) algorithm, the Viterbi training is used in Kaldi, which is computationally cheaper and it has shown to achieve competitive results [142].
- Unlike the HTK toolkit, the number of GMMs per state cannot be defined explicitly. However, the total number of GMMs for all states of HMMs can be specified; the Gaussian mixture components are allocated to states according to the occupation count during the Viterbi training iterations. For this reason, a different HMM topology is employed for the background models in the DNN-HMM system (see Section 6.2).

The optimal number of GMMs in each detector is empirically chosen from a set of values between 100 and 1500.

9.2.2 DNN-HMM training

The DNN training is carried out in two steps, the unsupervised pre-training and the fine-tuning of the DNNs.

The pre-training of the DBN (i.e. Deep Belief Network, stack of RBMs) is completed, using 20% of the training set and contrastive divergence algorithm with 1-step of Markov chain Monte Carlo sampling [143]. The first layer and the following layers of RBMs are composed of Gaussian-Bernoulli and Bernoulli-Bernouli units, respectively. Also, the number of epochs in the pre-training process is set to be five.

In order to fine-tune the DNNs, per-frame cross-entropy between the HMM state target posterior probabilities and network output is minimised, using mini-batch stochastic gradient descent (SGD). The mini-batch size and learning rate are set to be 256 and 0.008, respectively. In this stage, the baseline GMM-HMM system is used, to generate the state level alignment of the training data, which will provide a frame level label for training the DNN-HMM models. The DNN system consists of 4 hidden layers with total n number of hidden units (neurones); different values of $n \in \{ 16, 32, 64, 128, 256, 512 \}$ are tested for each detector.

9.3 RESULTS OF THE BASELINE GMM-HMM SYSTEM

The sub-goal spotting experiment is repeated using the GMM-HMM models trained by the Kaldi toolkit. The spotting performance of each sub-goal is evaluated by comparing the time at which the sub-goal is detected with the reference transcription of the full-trial recordings (section 3.6).

The lowest spotting error-rate for each sub-goal is presented in Table 9.1. Also, the total number of GMMs in each detector and details of the detected sub-goals are included in the table. The ”Pour Kettle” detector achieved the best performance amongst all the sub-goals with 4.22% error rate, using the total number of 500, 600, 1000 or 1300 GMMs in the HMM-set.

The two poorest performances were for the “Stir” and “Add Milk” sub-goals with a 34.13% and 19.65% error rate, respectively. The average error-rate of 15.95% is achieved for detection of all sub-goals in the full-trial recordings using the GMM-HMM models.

Table 9.1 *Sub-goal spotting performance for each detector using the Baseline GMM-HMM system. Three "Add Sugar", "Add Teabag" and " Remove Teabag" sub-goals are recognised in "Front Action" detector, so they share the total number of GMMs*

Sub-goals	Total number of GMMs	Detection details			%Error rate
		Correct	Deletion	Insertion	
Pour Kettle	500/600/1000/1300	92	3	1	4.22
Add Milk	200	47	9	2	19.65
Fill Kettle	900	90	5	1	6.32
Add Sugar		78	12	3	16.67
Add Teabag	200	86	8	6	14.90
Remove Teabag		89	6	9	15.79
Stir	600	107	19	24	34.13
				Average	15.95

Figure 9.1 compares the results of the sub-goal spotting using the GMM-HMM systems trained with the Kaldi and HTK toolkits. The HTK system achieved a better performance (lower error rate) in the detection of the “Add Milk” and “Add Teabag” sub-goals; for the remaining sub-goals, the Kaldi system obtained the lower error rate. The spotting performances of the two systems are comparable for all sub-goals; with the maximum error rate difference of 2.21%, which occurred for the “Add Sugar” sub-goal, where the Kaldi

system performed better. These similar results between the two systems, justifies that the trained baseline system is reliable for generating the force alignment.

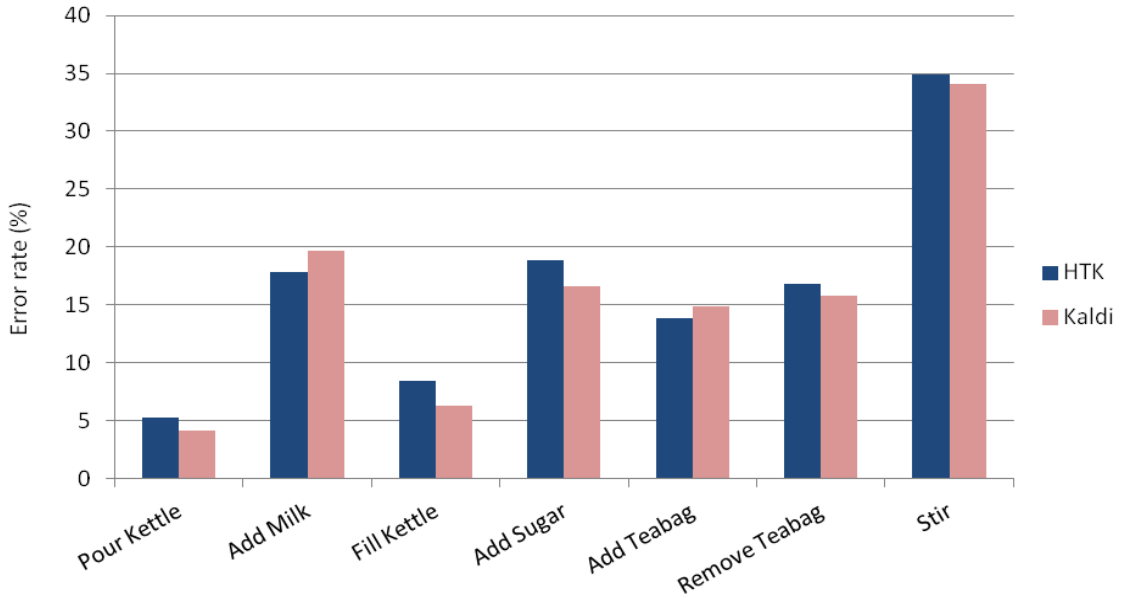


Fig. 9.1 *Sub-goal spotting performance of the HTK versus Kaldi GMM-HMM models*

9.4 RESULTS OF THE DNN-HMM SYSTEM

Table 9.2 summarises the performance of the spotting task for each individual sub-goal, using the DNN-HMM system. The lowest error rates of 2.11% and 3.34% are achieved for the “Pour Kettle” and “AddSugar” sub-goals, using a total number of 128 and 256 hidden units (neurons) in the “Pour Kettle” and “Front Actions” detectors, respectively. Same as the GMM-HMM system, the poorest spotting performance with a 29.3% error rate belongs to the “Stir” sub-goal with 15 deletion and 22 insertion errors; where most of the insertions happened during the beginning of the “Remove Teabag“ sub-goal.

The average error rate of 13% is claimed to be the final performance of the DNN-HMM system for spotting the sub-goals in the tea-making task. This result is a 18% relative error rate reduction as compared to the GMM-HMM system's result.

The minimum amount of training data (sample size) required to train the parameters of a neural network is an open research topic and dependant on many different aspects of the system and experiment. These include the number of output classes and how different they are from each other, the statistical characteristics of the data (and in particular the similarity of the test and training data) and features, and the possible use of pre-trained weights to initialise the lower layers. In this research, five DNN detectors are used to detect all sub-goals of the tea-making task. To empirically find the optimal DNN size for each detector, the number of hidden units per hidden layer was varied between 16 and 512 in DNNs with 4 hidden layers. Initially the recognition performance increases as the DNN size gets bigger, reaches a maximum, and then starts decreasing. In Table 9.2, the optimal DNN size for each detector is illustrated; the biggest network (with 256 neurons in each layer) belongs to the "Front Actions" detector, that is responsible for detection of "Add Teabag", "Add Sugar" and "Remove Teabag" sub-goals. In these experiments, 248 minutes of sub-goals' isolated recordings (with 50Hz sampling rate) are used for training the DNNs, so in total there are around 744,000 data points (samples) in the training set. The complexity of a neural network can be expressed through the number of parameters (weights) in the network. In a DNN with 4 hidden layers and 512 neurons per layer, there are approximately 800,000 parameters in the network to be learned (plus couple of thousand for input and output layers). This number goes down to 196,000 and 50,000 when 256 and 128 neurons are used in each layer, respectively. For all detectors increasing the number of neurons per layer to 512 has caused the performance to fall. This may be due to over-fitting when there are more parameters in the DNN than training samples in the training data. This experiment suggests that the

minimum number of training samples required per parameter in a DNN is approximately four, as in the "Front Action" detector.

Table 9.2 *Sub-goal spotting performance for each detector using the DNN-HMM system*

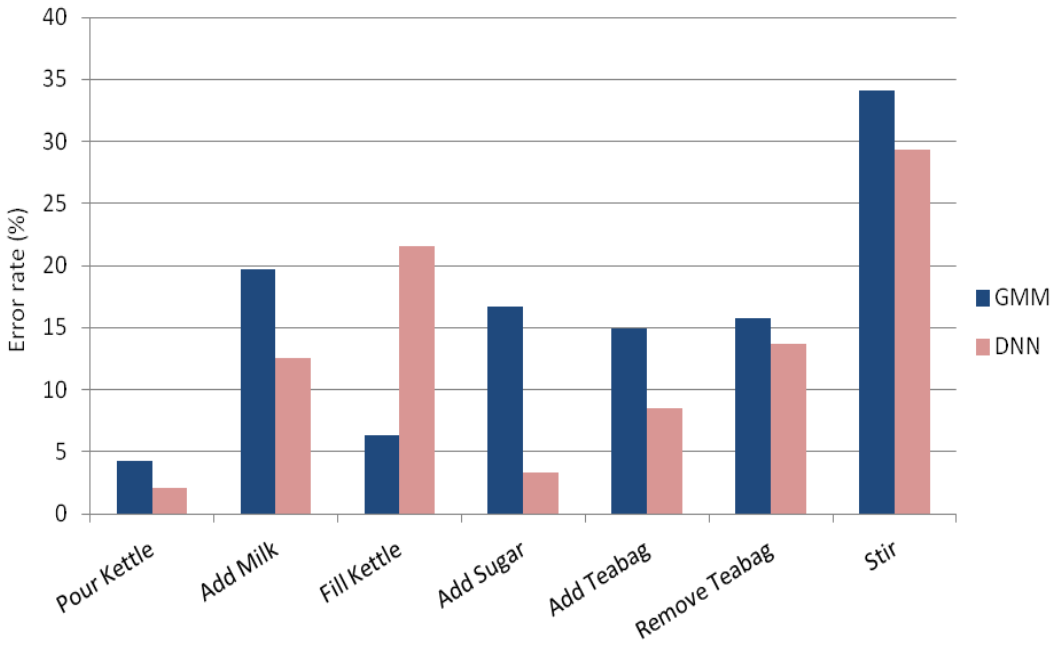
Sub-goals	Total number of neurons	Detection details			%Error rate
		Correct	Deletion	Insertion	
Pour Kettle	128	94	1	1	2.11
Add Milk	16	52	4	3	12.5
Fill Kettle	64	90	5	15	21.6
Add Sugar		89	1	2	3.34
Add Teabag	256	92	2	6	8.52
Remove Teabag		90	5	8	13.68
Stir	128	111	15	22	29.37
				Average	13.01

Figure 9.1 compares the results of the sub-goal spotting using the baseline GMM-HMM and the DNN-HMM systems.

DNN-HMM system outperforms the GMM-HMM system in spotting of all sub-goals but the “Fill Kettle” . Relative error rate reduction of 79%, 50%, 42%, 36%, 13% and 13% were achieved in detection of “Add Sugar”, “Pour Kettle”, “Add Teabag”, “Add Milk”, “Remove Teabag” and “Stir” sub-goals, respectively.

The spotting error rate of the “Fill Kettle” sub-goal increased by 15% due to the extra 14 insertion errors in the DNN system (Table 9.1 and 9.2). The time boundaries in which these errors happened were investigated; 12 out of 15 “Fill Kettle” insertions started at the beginning of the “Boil Water” sub-goal and finished at the beginning of the “Pour Kettle” sub-goal. Comparing the way these sub-goals are performed (section 3.2) and the effect of each sub-action on the features calculated from relevant sensors involved in the “Fill Kettle”

Fig. 9.2 *Performance of the DNN-HMM versus baseline GMM-HMM system for all sub-goals in the tea-making task*



detector (kettle sensors and Kinect), the similarity between these sub-goals can be explained as follows:

- Pressing the key on top of the kettle's handle and opening the lid of the Kettle, generates similar force on the FSRs under the kettle.
- The back and forth placement of the hand in the vicinity of the kettle causes the neighbourhood features to react similarly.
- Water boiling inside the kettle and adding water from the jug into the kettle generate same vibrations on the accelerometer that is attached to the kettle body. The vibration of the accelerometer (attached to the kettle body) is similar from water getting boiled and adding water from the jug into the kettle.

- Closing the lid and picking up the kettle, both apply big force on the FSRs under the kettle.

Considering these similarities and lack of training data for the boiling water sub-goal in the dataset, we can conclude that the DNN models generate more distinctive representation of the sub-goal; and the GMM models might be more suitable to handle unseen "toying" data.

9.5 SUMMARY

In this chapter, the effect of using a DNN-HMM system on sub-goal spotting performance for all sub-goals were investigated. The average error rate of 13.01% is achieved amongst detection of all sub-goals, which is an 18% relative ER reduction compare to the GMM-HMM system. This shows that deploying DNN-HMM system in activity recognition applications can be beneficial.

For all sub-goals but the "Fill Kettle" the performance were improved by using the DNN-HMM system. Relative ER reduction of 79%, 50%, 42%, 36%, 13% and 13% were achieved for detection of "Add Sugar", "Pour Kettle", "Add Teabag", "Add Milk", "Remove Teabag" and "Stir" sub-goals, respectively.

From the results in this chapter, it can be concluded that the DNN-HMM outperforms the GMM-HMM system in our activity recognition task. Also, depite all difference in the way that the GMM-HMM system is built in two HTK and Kaldi toolkits, very comparable result achieved for all sub-goals.

CHAPTER 10

CONCLUSION

This thesis evaluated the feasibility of employing HMM-based action recognition system for rehabilitation of stroke patients, using pervasive sensing technologies.

10.1 PRACTICAL CONTRIBUTIONS

In Section 1.4 we summarised the key contributions of this thesis. In this section we aim to elaborate on these key contributions by reviewing the important results and achievements in each chapter.

- **Chapter 3: collection of tea-making dataset**

  The tea-making ADL was broken down to low-level sub-goals. Given structures about performing an individual sub-goal, 38 participants were asked to perform each sub-goal in isolation, and make complete cups of tea (full-trials). Approximate 4 hours of the isolated sub-goals and 2 hours of the full-trials were recorded. The full-trials were annotated using the video recordings of sessions. The tea-making dataset will be made available on the internet.

- **Chapter 4: development of CogWatch instumented coaster**

  Pervasive and wireless CogWatch instrumented coaster (CIC) is developed, which consists of three force sensitive resistors (FSRs), a 3D accelerometer, a Bluetooth module and a microprocessor. Three CICs attached to the cup, Milk container and Kettle, plus the Kinect(TM) sensor, are employed in data collection.

- **Chapte 5: implementation of the partial trace-back algorithm**

  A novel parallel architecture of HMM-based action recognition is presented to address the issue caused by concurrent execution of sub-goals. The AR consists of five detectors that together can accommodate the lattice of sub-goals that describes an ADL, such as tea-making. Customised set of features is employed in each detector. The real-time and memory efficient implementation of the trace-back algorithm in Viterbi decoder is explained and used in CogWatch AR detectors.

- **Chapter 6: sub-goal detectors configurations**

  The configuration for all five "Pour Kettle", "Add Milk", "Fill Kettle", "Front Actions" and "Stir" sub-goal detectors are explained.

- **Chapter 7: real-time output delay**

  Two sets of experiments are defined for each detector, namely, sub-goal verification and spotting. The optimum number states for sub-goal HMMs are discovered from the verification experiment, and used in the spotting experiment. The optimum HMM-set that trained in the spotting experiment are used in CogWatch action recognition with some modification.

  The proposed modification to the HMM's state transition probabilities, lead to an average 64% output delay reduction for all detections of sub-goals in the full-trial recordings.

- **Chapter 8: GMM-HMM Results for verification and spotting tasks**

  For the verification task, the average error rate (ER) of 9.15% is achieved amongst all sub-goals. The lowest and highest ER of 0.9% and %19.78 are belong to the "Pour Kettle" and "Add Milk" sub-goals, respectively.

  For spotting task, the average ER of 16.58% is achieved amongst all sub-goals. The lowest and highest ER of 5.27% and %34.92 are achieved for the "Pour Kettle" and "Stir" sub-goals, respectively.

  By applying the modification to the HMMs in spotting task, the average output delay for all detections of sub-goals reduced from 2.64 to 0.81 seconds. The maximum 0.16 and minimum 3.74 seconds improvement, was achieved for detection of the "Remove Teabag" and "Pour Kettle" sub-goals, respectively.

- **Chapter 9: DNN-HMM results for spotting task**

  The effect of using a DNN-HMM system on detection performance of the spotting task for all sub-goals, is investigated. The average error rate of 13.01% is achieved amongst detection of all sub-goals, which is an 18% relative ER reduction compare to the GMM-HMM system.

  For all sub-goals but the "Fill Kettle" the performance were improved by using the DNN-HMM system. Relative ER reduction of 79%, 50%, 42%, 36%, 13% and 13% were achieved for detection of "Add Sugar", "Pour Kettle", "Add Teabag", "Add Milk", "Remove Teabag" and "Stir" sub-goals, respectively.

10.2 FUTURE WORK

The work presented in this thesis can be developed in several ways. In the following, some of the interesting future works are pointed out.

- The robustness and deployability of the system can be improved by removing the Kinect sensor from the system and instrumenting all objects, e.g. the spoon and jug. Also, new sensors can be added to the system, such as audio, temperature and Radio-frequency identification (RFID).

- The focus of this thesis was on the recognition of the tea-making ADL, but now that the concept of using a HMM based system with pervasive sensors are approved. The same real-time AR system can be deployed to generalised the work to other ADLs, e.g., tooth brushing and cooking.

- The real-time Viterbi decoder in the system works for GMM-HMM models. With some adjustments to the current decoder, one can develop a real-time AR based on DNN-HMMs. The adjustment would be adding a C# class to the program that calculates the output (observation) probabilities of a state from trained DNN.

- The current architecture of action recognition assumes all sub-goals can happen at once, which is a positive point in case of generalizing the same system to other ADLs. However, focusing on explicit constraint of the tea-making task, the user can perform a maximum of two sub-goals simultaneously (with two hands) and Kettle can boil water itself. It is interesting to develop a new architecture for the AR system which focuses on recognition of sub-goals performed by the motors (e.g. the Kettle and hands in the tea-making task) in the system.

- Breaking down the sub-goals to "atomic" levels of sub-actions, e.g. the "pour kettle" sub-goal can be broken down to sub-actions of picking up the kettle, tilting the kettle and putting down the kettle. If sub-goals in the action recognition represent words in speech recognition, the sub-actions are phonemes. So, context-dependant sub-actions models can be trained.

REFERENCES

[1] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. A. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, *et al.*, "The htk book (for htk version 3.4)," 2006.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[3] E. Jean-Baptiste, *Statistical Task Modeling of Activities of Daily Living for Rehabilitation*. PhD thesis, School of Engineering, Univeristy of Birmingham, 2 2016.

[4] Stroke association, "State of the Nation Stroke statistics," *https://www.stroke.org.uk/resources/state-nation-stroke-statistics*, 2016.

[5] C. Barbieri and E. De Renzi, "The executive and ideational components of apraxia," *Cortex*, vol. 24, no. 4, pp. 535–543, 1988.

[6] W.-L. Bickerton, M. J. Riddoch, D. Samson, A. B. Balani, B. Mistry, and G. W. Humphreys, "Systematic assessment of apraxia and functional predictions from the birmingham cognitive screen," *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 83, no. 5, pp. 513–521, 2012.

[7] B. Petreska, M. Adriani, O. Blanke, and A. G. Billard, "Apraxia: a review," *Progress in brain research*, vol. 164, pp. 61–83, 2007.

[8] G. Goldenberg, M. Daumüller, and S. Hagmann, "Assessment and therapy of complex activities of daily living in apraxia," *Neuropsychological Rehabilitation*, vol. 11, no. 2, pp. 147–169, 2001.

[9] K. Morady and G. W. Humphreys, "Comparing action disorganization syndrome and dual-task load on normal performance in everyday action tasks," *Neurocase*, vol. 15, no. 1, pp. 1–12, 2009.

[10] G. Goldenberg, *Apraxia: The cognitive side of motor control*. 2013.

[11] D. Hyndman and A. Ashburn, "People with stroke living in the community: Attention deficits, balance, adl ability and falls," *Disability and rehabilitation*, vol. 25, no. 15, pp. 817–822, 2003.

[12] M. F. Schwartz, E. S. Reed, M. Montgomery, C. Palmer, and N. H. Mayer, "The quantitative description of action disorganisation after brain damage: A case study," *Cognitive Neuropsychology*, vol. 8, no. 5, pp. 381–414, 1991.

[13] M. Pastorino, A. Fioravanti, M. T. Arredondo, J. M. Cogollor, J. Rojo, M. Ferre, and A. M. Wing, "Cogwatch: A web based platform for cognitive tele-rehabilitation and follow up of apraxia and action disorganisation syndrome patients," in *Proceedings of IEEE-EMBS International Conference on Biomedical and Health Informatics, BHI 2014, Valencia, Spain, June 1-4, 2014*, pp. 133–136, 2014.

[14] M. M. de Werd, D. Boelen, M. G. O. Rikkert, and R. P. Kessels, "Errorless learning of everyday tasks in people with dementia," *Clinical interventions in aging*, vol. 8, p. 1177, 2013.

[15] E. L. Middleton and M. F. Schwartz, "Errorless learning in cognitive rehabilitation: A critical review," *Neuropsychological Rehabilitation*, vol. 22, no. 2, pp. 138–168, 2012.

[16] C. M. van Heugten, J. Dekker, B. Deelman, A. Van Dijk, J. Stehmann-Saris, and A. Kinebanian, "Outcome of strategy training in stroke patients with apraxia: a phase ii study," *Clinical Rehabilitation*, vol. 12, no. 3, pp. 216–225, 1998.

[17] E. M. D. Jean-Baptiste, P. Rotshtein, and M. J. Russell, "Cogwatch: Automatic prompting system for stroke survivors during activities of daily living," *J. Innovation in Digital Ecosystems*, 2016.

[18] E. M. Jean-Baptiste, M. Russell, J. Howe, and P. Rotshtein, "Cogwatch: Intelligent agent-based system to assist stroke survivors during tea-making," in *SAI Intelligent Systems Conference (IntelliSys), 2015*, pp. 452–456, 2015.

[19] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[20] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *IEEE transactions on pattern analysis and machine intelligence*, vol. 25, no. 10, pp. 1337–1342, 2003.

[21] M. Seki, H. Fujiwara, and K. Sumi, "A robust background subtraction method for changing background," in *Applications of Computer Vision, 2000, Fifth IEEE Workshop on.*, pp. 207–213, 2000.

[22] D. Murray and A. Basu, "Motion tracking with an active camera," *IEEE transactions on pattern analysis and machine intelligence*, vol. 16, no. 5, pp. 449–459, 1994.

[23] K. K. Kim, S. H. Cho, H. J. Kim, and J. Y. Lee, "Detecting and tracking moving object using an active camera," in *Advanced Communication Technology, 2005, ICACT 2005. The 7th International Conference on*, vol. 2, pp. 817–820, 2005.

[24] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, pp. 1395–1402, 2005.

[25] Y. Ke, R. Sukthankar, and M. Hebert, "Spatio-temporal shape and flow correlation for action recognition," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8, 2007.

[26] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pp. 65–72, 2005.

[27] E. Shechtman and M. Irani, "Space-time behavior based correlation," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 405–412, 2005.

[28] S. Kumari and S. K. Mitra, "Human action recognition using dft," in *Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2011 Third National Conference on*, pp. 239–242, 2011.

[29] W.-L. Lu and J. J. Little, "Simultaneous tracking and action recognition using the pca-hog descriptor," in *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*, pp. 6–6, 2006.

[30] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th ACM international conference on Multimedia*, pp. 357–360, 2007.

[31] A. Veeraraghavan, A. K. Roy-Chowdhury, and R. Chellappa, "Matching shape sequences in video with applications in human movement analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1896–1909, 2005.

[32] D. Ramanan, D. A. Forsyth, and A. Zisserman, "Tracking people by learning their appearance," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 1, pp. 65–81, 2007.

[33] K. Schindler and L. Van Gool, "Action snippets: How many frames does human action recognition require?," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, 2008.

[34] S. Danafar and N. Gheissari, "Action recognition for surveillance applications using optic flow and svm," in *Asian Conference on Computer Vision*, pp. 457–466, 2007.

[35] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, pp. 1150–1157, 1999.

[36] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[37] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, 2005.

[38] F. Huo, E. Hendriks, P. Paclik, and A. H. Oomes, "Markerless human motion capture and pose recognition," in *Image Analysis for Multimedia Interactive Services, 2009. WIAMIS'09. 10th Workshop on*, pp. 13–16, 2009.

[39] S. Sedai, M. Bennamoun, and D. Huynh, "Context-based appearance descriptor for 3d human pose estimation from monocular images," in *Digital Image Computing: Techniques and Applications, 2009. DICTA'09.*, pp. 484–491, 2009.

[40] A. Nakazawa, H. Kato, and S. Inokuchi, "Human tracking using distributed vision systems," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, vol. 1, pp. 593–596, 1998.

[41] S. Iwasawa, K. Ebihara, J. Ohya, and S. Morishima, "Real-time estimation of human body posture from monocular thermal images," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pp. 15–20, 1997.

[42] M. K. Leung and Y.-H. Yang, "First sight: A human body outline labeling system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 4, pp. 359–377, 1995.

[43] I.-F. Leong, J.-J. Fang, and M.-J. Tsai, "Automatic body feature extraction from a marker-less scanned human body," *Computer-Aided Design*, vol. 39, no. 7, pp. 568–582, 2007.

[44] M. W. Lee and R. Nevatia, "Body part detection for human pose estimation and tracking," in *Motion and Video Computing, 2007. WMVC'07. IEEE Workshop on*, pp. 23–23, 2007.

[45] M. W. Lee and R. Nevatia, "Human pose tracking in monocular sequence using multilevel structured models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 1, pp. 27–38, 2009.

[46] S.-R. Ke, L. Zhu, J.-N. Hwang, H.-I. Pai, K.-M. Lan, and C.-P. Liao, "Real-time 3d human pose estimation from monocular view with applications to event detection and video gaming," in *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, pp. 489–496, 2010.

[47] S.-R. Ke, J.-N. Hwang, K.-M. Lan, and S.-Z. Wang, "View-invariant 3d human body pose reconstruction using a monocular video camera," in *Distributed Smart Cameras (ICDSC), 2011 Fifth ACM/IEEE International Conference on*, pp. 1–6, 2011.

[48] S. Sempena, N. U. Maulidevi, and P. R. Aryan, "Human action recognition using dynamic time warping," in *Electrical Engineering and Informatics (ICEEI), 2011 International Conference on*, pp. 1–5, 2011.

[49] T. V. Duong, H. H. Bui, D. Q. Phung, and S. Venkatesh, "Activity recognition and abnormality detection with the switching hidden semi-markov model," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 838–845, 2005.

[50] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pp. 379–385, 1992.

[51] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden markov models for complex action recognition," in *Computer vision and pattern recognition, 1997. proceedings., 1997 ieee computer society conference on*, pp. 994–999, 1997.

[52] P. Natarajan and R. Nevatia, "Online, real-time tracking and recognition of human actions," in *Motion and video Computing, 2008. WMVC 2008. IEEE Workshop on*, pp. 1–8, 2008.

[53] X. Huang, A. Acero, H.-W. Hon, and R. Foreword By-Reddy, *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice Hall PTR, 2001.

[54] Y. Luo, T.-D. Wu, and J.-N. Hwang, "Object-based analysis and interpretation of human motion in sports video sequences by dynamic bayesian networks," *Computer Vision and Image Understanding*, vol. 92, no. 2, pp. 196–216, 2003.

[55] Y. Du, F. Chen, and W. Xu, "Human interaction representation and recognition through motion decomposition," *IEEE Signal Processing Letters*, vol. 14, no. 12, pp. 952–955, 2007.

[56] K. P. Murphy and S. Russell, "Dynamic bayesian networks: representation, inference and learning," 2002.

[57] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3, pp. 32–36, 2004.

[58] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, 2008.

[59] V. Vapnik, S. E. Golowich, and A. J. Smola, "Support vector method for function approximation, regression estimation and signal processing," in *Advances in neural information processing systems*, pp. 281–287, 1997.

[60] A. Agarwal and B. Triggs, "Recovering 3d human pose from monocular images," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 1, pp. 44–58, 2006.

[61] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of machine learning research*, vol. 1, no. Jun, pp. 211–244, 2001.

[62] H. Foroughi, A. Naseri, A. Saberi, and H. S. Yazdi, "An eigenspace-based approach for human fall detection using integrated time motion image and neural network," in *Signal Processing, 2008. ICSP 2008. 9th International Conference on*, pp. 1499–1503, 2008.

[63] M. Fiaz and B. Ijaz, "Vision based human activity tracking using artificial neural networks," in *Intelligent and Advanced Systems (ICIAS), 2010 International Conference on*, pp. 1–5, 2010.

[64] R. Bodor, B. Jackson, and N. Papanikolopoulos, "Vision-based human tracking and activity recognition," in *Proc. of the 11th Mediterranean Conf. on Control and Automation*, vol. 1, 2003.

[65] G. Bishop and G. Welch, "An introduction to the kalman filter," *Proc of SIGGRAPH, Course*, vol. 8, no. 27599-23175, p. 41, 2001.

[66] P. C. Ribeiro, J. Santos-Victor, and P. Lisboa, "Human activity recognition from video: modeling, feature selection and classification architecture," in *Proceedings of International Workshop on Human Activity Recognition and Modelling*, pp. 61–78, 2005.

[67] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 747–757, 2000.

[68] J. Ben-Arie, Z. Wang, P. Pandit, and S. Rajaram, "Human activity recognition using multidimensional indexing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, pp. 1091–1104, 2002.

[69] N. D. Bird, O. Masoud, N. P. Papanikolopoulos, and A. Isaacs, "Detection of loitering individuals in public transportation areas," *IEEE Transactions on intelligent transportation systems*, vol. 6, no. 2, pp. 167–177, 2005.

[70] T. T. Zin, P. Tin, T. Toriu, and H. Hama, "A markov random walk model for loitering people detection," in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on*, pp. 680–683, 2010.

[71] W. Niu, J. Long, D. Han, and Y.-F. Wang, "Human activity detection and recognition for video surveillance," in *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, vol. 1, pp. 719–722, 2004.

[72] N. Moënne-Loccoz, F. Brémond, and M. Thonnat, "Recurrent bayesian network for the recognition of human behaviors from video," in *ICVS*, pp. 68–77, 2003.

[73] W. Lin, M. T. Sun, R. Poovandran, and Z. Zhang, "Human activity recognition for video surveillance," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pp. 2737–2740, IEEE, 2008.

[74] C. Chen, K. Liu, and N. Kehtarnavaz, "Real-time human action recognition based on depth motion maps," *Journal of Real-Time Image Processing*, pp. 1–9, 2013.

[75] Y.-M. Kuo, J.-S. Lee, and P.-C. Chung, "A visual context-awareness-based sleeping-respiration measurement system," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 2, pp. 255–265, 2010.

[76] J. Gao, A. G. Hauptmann, A. Bharucha, and H. D. Wactlar, "Dining activity analysis using a hidden markov model," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 2, pp. 915–918, 2004.

[77] H. Foroughi, A. Rezvanian, and A. Paziraee, "Robust fall detection using human shape and multi-class support vector machine," in *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*, pp. 413–420, 2008.

[78] H. Foroughi, B. S. Aski, and H. Pourreza, "Intelligent video surveillance for monitoring fall detection of elderly in home environments," in *Computer and Information Technology, 2008. ICCIT 2008. 11th International Conference on*, pp. 219–224, 2008.

[79] A. Ghali, A. S. Cunningham, and T. P. Pridmore, "Object and event recognition for stroke rehabilitation.," in *VCIP*, pp. 980–989, 2003.

[80] R. Ayase, T. Higashi, S. Takayama, S. Sagawa, and N. Ashida, "A method for supporting at-home fitness exercise guidance and at-home nursing care for the elders, video-based simple measurement system," in *e-health Networking, Applications and Services, 2008. HealthCom 2008. 10th International Conference on*, pp. 182–186, 2008.

[81] K. Van Laerhoven and H.-W. Gellersen, "Spine versus porcupine: A study in distributed wearable activity recognition," in *Wearable Computers, 2004. ISWC 2004. Eighth International Symposium on*, vol. 1, pp. 142–149, 2004.

[82] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive computing*, pp. 1–17, Springer, 2004.

[83] K. Van Laerhoven and O. Cakmakci, "What shall we teach our pants?," in *Wearable Computers, The Fourth International Symposium on*, pp. 77–83, 2000.

[84] T. Choudhury and A. Pentland, "Sensing and modeling human networks using the sociometer," in *null*, p. 216, 2003.

[85] L. E. Dunne, P. Walsh, B. Smyth, and B. Caulfield, "Design and evaluation of a wearable optical sensor for monitoring seated spinal posture," in *Wearable Computers, 2006 10th IEEE International Symposium on*, pp. 65–68, 2006.

[86] S. Brady, L. E. Dunne, R. Tynan, D. Diamond, B. Smyth, and G. M. O'Hare, "Garment-based monitoring of respiration rate using a foam pressure sensor," in *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pp. 214–215, 2005.

[87] P. Lukowicz, F. Hanser, C. Szubski, and W. Schobersberger, "Detecting and interpreting muscle activity with wearable force sensors," *Lecture Notes in Computer Science*, vol. 3968, pp. 101–116, 2006.

[88] N. Oliver and F. Flores-Mangas, "Healthgear: a real-time wearable system for monitoring and analyzing physiological signals," in *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, pp. 4–pp, 2006.

[89] T. Westeyn, P. Presti, and T. Starner, "Actiongsr: A combination galvanic skin response-accelerometer for physiological measurements in active environments," in *Wearable Computers, 2006 10th IEEE International Symposium on*, pp. 129–130, 2006.

[90] T. Linz, C. Kallmayer, R. Aschenbrenner, and H. Reichl, "Fully untegrated ekg shirt based on embroidered electrical interconnections with conductive yarn and miniaturized flexible electronics," in *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, pp. 4–pp, 2006.

[91] T. Huynh, U. Blanke, and B. Schiele, "Scalable recognition of daily activities with wearable sensors," *Location-and context-awareness*, pp. 50–67, 2007.

[92] E. A. Heinz, K. S. Kunze, M. Gruber, D. Bannach, and P. Lukowicz, "Using wearable sensors for real-time recognition tasks in games of martial arts-an initial experiment," in *Computational Intelligence and Games, 2006 IEEE Symposium on*, pp. 98–102, 2006.

[93] R. P. Aylward, *Sensemble: a wireless inertial sensor system for the interactive dance and collective motion analysis*. PhD thesis, Massachusetts Institute of Technology, School of Architecture and Planning, Program in Media Arts and Sciences, 2006.

[94] P. Lukowicz, A. Timm-Giel, M. Lawo, and O. Herzog, "Wearit@ work: Toward real-world industrial wearable computing," *IEEE Pervasive Computing*, vol. 6, no. 4, 2007.

[95] T. Stiefmeier, D. Roggen, G. Ogris, P. Lukowicz, and G. Tröster, "Wearable activity tracking in car manufacturing," *IEEE Pervasive Computing*, vol. 7, no. 2, 2008.

[96] M. J. Mathie, A. C. Coster, N. H. Lovell, B. G. Celler, S. R. Lord, and A. Tiedemann, "A pilot study of long-term monitoring of human movements in the home using accelerometry," *Journal of telemedicine and telecare*, vol. 10, no. 3, pp. 144–151, 2004.

[97] E. S. Sazonov, G. Fulk, N. Sazonova, and S. Schuckers, "Automatic recognition of postures and activities in stroke patients," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pp. 2200–2203, 2009.

[98] D. Giansanti, G. Maccioni, and S. Morelli, "An experience of health technology assessment in new models of care for subjects with parkinson's disease by means of a new wearable device," *TELEMEDICINE and e-HEALTH*, vol. 14, no. 5, pp. 467–472, 2008.

[99] R. Jafari, W. Li, R. Bajcsy, S. Glaser, and S. Sastry, "Physical activity monitoring for assisted living at home," in *4th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2007)*, pp. 213–219, 2007.

[100] A. Bourke, J. O'brien, and G. Lyons, "Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm," *Gait & posture*, vol. 26, no. 2, pp. 194–199, 2007.

[101] K. J. Liszka, M. A. Mackin, M. J. Lichter, D. W. York, D. Pillai, and D. S. Rosenbaum, "Keeping a beat on the heart," *IEEE Pervasive Computing*, vol. 3, no. 4, pp. 42–49, 2004.

[102] E. Villalba, M. Ottaviano, M. T. Arredondo, A. Martinez, and S. Guillen, "Wearable monitoring system for heart failure assessment in a mobile environment," in *Computers in Cardiology, 2006*, pp. 237–240, 2006.

[103] T. Choudhury, M. Philipose, D. Wyatt, and J. Lester, "Towards activity databases: Using sensors and statistical models to summarize people's lives.," *IEEE Data Eng. Bull.*, vol. 29, no. 1, pp. 49–58, 2006.

[104] K. Van Laerhoven, B. P. Lo, J. W. Ng, S. Thiemjarus, R. King, S. Kwan, H.-W. Gellersen, M. Sloman, O. Wells, P. Needham, *et al.*, "Medical healthcare monitoring with wearable and implantable sensors," in *Proc. of the 3rd International Workshop on Ubiquitous Computing for Healthcare Applications*, 2004.

[105] U. Anliker, J. A. Ward, P. Lukowicz, G. Troster, F. Dolveck, M. Baer, F. Keita, E. B. Schenker, F. Catarsi, L. Coluccini, *et al.*, "Amon: a wearable multiparameter medical monitoring and alert system," *IEEE Transactions on information technology in Biomedicine*, vol. 8, no. 4, pp. 415–427, 2004.

[106] R. Paradiso, G. Loriga, and N. Taccini, "A wearable health care system based on knitted integrated sensors," *IEEE transactions on Information Technology in biomedicine*, vol. 9, no. 3, pp. 337–344, 2005.

[107] J. Maitland, S. Sherwood, L. Barkhuus, I. Anderson, M. Hall, B. Brown, M. Chalmers, and H. Muller, "Increasing the awareness of daily activity levels with pervasive computing," in *Pervasive Health Conference and Workshops, 2006*, pp. 1–9, 2006.

[108] S. Consolvo, D. W. McDonald, T. Toscos, M. Y. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, *et al.*, "Activity sensing in the wild: a field trial of ubifit garden," in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1797–1806, 2008.

[109] J. R. Smith, K. P. Fishkin, B. Jiang, A. Mamishev, M. Philipose, A. D. Rea, S. Roy, and K. Sundara-Rajan, "Rfid-based techniques for human-activity detection," *Communications of the ACM*, vol. 48, no. 9, pp. 39–44, 2005.

[110] M. Philipose, K. P. Fishkin, M. Perkowitz, D. J. Patterson, D. Fox, H. Kautz, and D. Hahnel, "Inferring activities from interactions with objects," *IEEE pervasive computing*, vol. 3, no. 4, pp. 50–57, 2004.

[111] D. H. Wilson and C. Atkeson, "Simultaneous tracking and activity recognition (star) using many anonymous, binary sensors," in *International Conference on Pervasive Computing*, pp. 62–79, 2005.

[112] R. Aipperspach, E. Cohen, and J. Canny, "Modeling human behavior from simple sensors in the home," in *International Conference on Pervasive Computing*, pp. 337–348, 2006.

[113] X. Hong, C. Nugent, M. Mulvenna, S. McClean, B. Scotney, and S. Devlin, "Evidential fusion of sensor data for activity recognition in smart homes," *Pervasive and Mobile Computing*, vol. 5, no. 3, pp. 236–252, 2009.

[114] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[115] Z. Ghahramani, "An introduction to hidden Markov models and Bayesian networks," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 15, no. 01, pp. 9–42, 2001.

[116] A. Poritz, "Hidden Markov models: A guided tour," in *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pp. 7–13, IEEE, 1988.

[117] D. Yu and L. Deng, *Automatic speech recognition: a deep learning approach*. Springer Publishing Company, Incorporated, 2014.

[118] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[119] G. E. Hinton, "Connectionist learning procedures," *Artificial intelligence*, vol. 40, no. 1, pp. 185–234, 1989.

[120] L. Bottou, "Online learning and stochastic approximations," *Online learning in neural networks*, vol. 17, no. 9, p. 25, 1998.

[121] D. Yu, L. Deng, and G. Dahl, "Roles of pre-training and fine-tuning in context-dependent DBN-HMMs for real-world speech recognition," in *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[122] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[123] A. Mohamed, T. N. Sainath, G. E. Dahl, B. Ramabhadran, G. E. Hinton, and M. A. Picheny, "Deep belief networks using discriminative features for phone recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, pp. 5060–5063, 2011.

[124] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novák, and A. Mohamed, "Making deep belief networks effective for large vocabulary continuous speech recognition," in *2011 IEEE Workshop on Automatic Speech Recognition & Understanding, ASRU*, pp. 30–35, 2011.

[125] N. Morgan, H. Bourlard, *et al.*, "Neural networks for statistical recognition of continuous speech," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 742–772, 1995.

[126] C. M. L. Hughes, C. Baber, M. Bienkiewicz, and J. Hermsdörfer, "Application of human error identification (HEI) techniques to cognitive rehabilitation in stroke patients with limb apraxia," in *Universal Access in Human-Computer Interaction. Applications and Services for Quality of Life - 7th International Conference, UAHCI 2013, Held as*

Part of HCI International 2013, Las Vegas, NV, USA, July 21-26, 2013, Proceedings, Part III, pp. 463–471, 2013.

[127] A. M. Genest, C. Gutwin, A. Tang, M. Kalyn, and Z. Ivkovic, "Kinectarms: a toolkit for capturing and displaying arm embodiments in distributed tabletop groupware," in *Proceedings of the 2013 conference on Computer supported cooperative work*, pp. 157–166, 2013.

[128] A. D. Wilson, "Using a depth camera as a touch sensor," in *ACM international conference on interactive tabletops and surfaces*, pp. 69–72, ACM, 2010.

[129] N. Kitsunezaki, E. Adachi, T. Masuda, and J. Mizusawa, "KINECT applications for the physical rehabilitation," in *IEEE International Symposium on Medical Measurements and Applications, MeMeA 2013, Gatineau, QC, Canada, May 4-5, 2013, Proceedings*, pp. 294–299, 2013.

[130] H. Mousavi Hondori and M. Khademi, "A review on technical and clinical impact of microsoft kinect on physical therapy and rehabilitation," *Journal of Medical Engineering*, vol. 2014, 2014.

[131] C.-J. Su, "Personal rehabilitation exercise assistant with kinect and dynamic time warping," *International Journal of Information and Education Technology*, vol. 3, no. 4, p. 448, 2013.

[132] J. Segen and S. Kumar, "Human-computer interaction using gesture recognition and 3d hand tracking," in *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, pp. 188–192, 1998.

[133] M. Gales and S. Young, "The application of hidden markov models in speech recognition," *Foundations and trends in signal processing*, vol. 1, no. 3, pp. 195–304, 2008.

[134] M. J. F. Gales and S. J. Young, "The application of hidden markov models in speech recognition," *Foundations and Trends in Signal Processing*, 2007.

[135] R. C. Rose and D. B. Paul, "A hidden markov model based keyword recognition system," in *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pp. 129–132, IEEE, 1990.

[136] P. F. Brown, J. Spohrer, P. H. Hochschild, and J. K. Baker, "Partial traceback and dynamic programming," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, vol. 7, pp. 1629–1632, IEEE, 1982.

[137] J. C. Spohrer, P. F. Brown, P. H. Hochschild, and J. K. Baker, "Partial traceback in continuous speech recognition," in *IEEE International Conference on Cybernetics and Society*, 1980.

[138] G. A. Fink, *Markov Models for Pattern Recognition: From Theory to Applications*, ch. Configuration of Hidden Markov Models, pp. 127–136. Springer Berlin Heidelberg, 2008.

[139] G. D. Forney Jr, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[140] S. Young, P. Woodland, G. Evermann, and M. Gales, "The HTK toolkit 3.4. 1," 2013.

[141] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584, 2011.

[142] L. J. Rodríguez and I. Torres, "Comparative study of the Baum-Welch and Viterbi training algorithms applied to read and spontaneous speech recognition," in *Pattern Recognition and Image Analysis*, pp. 847–857, Springer, 2003.

[143] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," in *Neural Networks: Tricks of the Trade - Second Edition*, pp. 599–619, 2012.