OBSERVATION OF MULTIPLE FEATURES FOR RECOGNITION OF DAILY ACTIVITIES

June 2012

Francesco Lilli
Master of Science in Computer Science

OBSERVATION OF MULTIPLE FEATURES FOR RECOGNITION OF DAILY ACTIVITIES

Francesco Lilli
Master of Science
Computer Science
June 2012
School of Computer Science
Reykjavík University

M.Sc. PROJECT REPORT

ISSN 1670-8539

Observation of Multiple Features for Recognition of Daily Activities

by

Francesco Lilli

Project report submitted to the School of Computer Science
at Reykjavík University in partial fulfillment of
the requirements for the degree of
Master of Science in **Computer Science**

June 2012

The undersigned hereby certify that they recommend to the School of Computer Science at Reykjavík University for acceptance this project report entitled **Observation of Multiple Features for Recognition of Daily Activities** submitted by **Francesco Lilli** in partial fulfillment of the requirements for the degree of **Master of Science** in **Computer Science**.

Date

Hannes Högni Vilhjálmsson, Supervisor
Associate Professor, Reykjavík University

Emanuela Merelli, Supervisor
Professor, University of Camerino

Jón Guðnason
Assistant Professor, Reykjavík University

Observation of Multiple Features for Recognition of Daily Activities

Francesco Lilli

June 2012

Abstract

In this document we present a research study on the simultaneous observation of several features as a means for the recognition of human activities; in particular we refer to high-level daily activities performed in a home environment. We abstract from techniques and modalities for observations retrieval, assuming each feature to be easy to track from the domain environment. No mechanisms for further probabilistic recognition of lower-level activities should be used: we recognize activities directly from information extracted from the features. In our work we introduce the concept of “observational channels”, each channel representing an abstract feature, as organized sources of information from the environment. We refer to real-world scenarios in which by definition not all information is accessible, meaning that we deal with partial knowledge: only information from plausible sources should be considered. In our study we use several Hidden Markov Models, a standard tool being widely used in such applications, for a feature-based recognition; we compare the recognition rates resulting from partial or total observation of available features. In our setting we focus on a typical domain from daily life, proving that the use of multiple features could be beneficial for the recognition; however the observation of certain combinations of features tends to be confusing for recognizers, which in some cases do not offer a minimally acceptable recognition rate. We conclude with a discussion about the results we obtained, possible improvements and refinements to the approach, and future works.

To Marina, Maurizio and Natalina.

Acknowledgements

It is a pleasure for me to thank those who have contributed in any way to make the fulfilment of this report possible.

I am very thankful to my supervisor in Reykjavík University, Hannes Högni Vilhjálmsson, for his precious guidance and encouragement from the very first to the last stages of the project. I also thank Jón Guðnason for his precious comments and for his availability to help me when needed.

I am deeply indebted to many of my professors in Italy, especially to my supervisor Emanuela Merelli, for offering me the possibility to move my first steps in the world of academic research, and to work on such an interesting topic. I am obliged to both UNICAM and RU for giving me the invaluable opportunity to study in Reykjavík with the Double Degree Programme between universities.

I would like to thank my fellow students David, Andrea and Alfredo; besides being simply awesome friends, they have helped and supported me in all the difficult moments I have experienced during my stay in Iceland. I will never be able to thank them enough.

A special thanks goes to Lucia for the patience she showed every time I was sad or unhappy with something (an occurrence that happened too many times), for always making me smile, and for giving me the motivation to carry on, no matter what. The moral support she gave me is priceless; she would actually deserve an entire chapter of this document. I especially thank her for being such a wonderful girl.

I owe my deepest gratitude and love to my mother Marina, my father Maurizio and my grandmother Natalina for all their dedication and encouragement, and for always believing in me. They supported me during my studies, thus providing the foundation for this work. I dedicate this report to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The automated analysis of human activities has been, and still is, of great interest for scientists from different fields. Systems for modelling and automatically recognizing human activities have a wide range of applications, such as surveillance, sign language recognition, intelligent environments. Starting from the last decades many research studies have been devoted to representing and recognizing varied kinds of activities performed by human subjects. In the last few years innovative technologies have been used more specifically in activity recognition in indoor spaces, especially home environments: the miniaturization and the increasing precision of sensing devices became the basis for pervasive computing-based approaches. In this chapter we present a concise introduction to the study that has been carried out and fully described in this document.

1.1 Brief elucidation

Before diving into the matters we explored in our study, we shall briefly cast light on the concepts mentioned in the title of this document.
With the term “feature” we refer to any particular parameter that can be observed:

- in a person performing a certain task, or
- in the environment surrounding the person.

For example we could consider a feature, called “Near”, from which we somehow get information about which objects a person is near to at a certain moment. Hence, the “content” of a feature changes every time the person is near to a different set of objects (with respect to the set of objects he/she was near to in the previous moment considered). Consider the example in Fig.1.1: the person is close to the mug and to the fridge, but not to the bowl; the “Near” feature would then report the value “Mug, Fridge” at the given instant of time.

With “daily activities” we refer to high-level tasks we normally accomplish every day, or at least rather frequently, in a home environment: examples of daily activities performed in a kitchen could be cooking, eating, washing-up. Often it is possible to categorize daily activities: certain tasks are assumed to be performed only in specific areas (e.g., one should brush his/her teeth in the bathroom, and not in the kitchen; one could sleep in the bedroom or in the living room, and not in the bathroom).

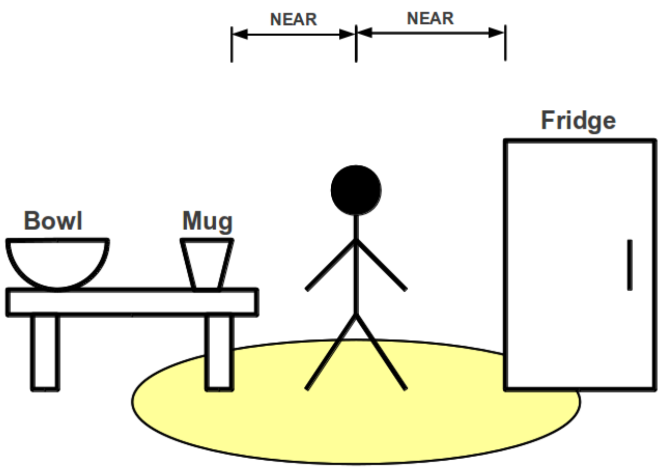


Figure 1.1: Example scenario: the person is close to the mug and to the fridge. The content of the “Near” feature excludes the bowl.

1.2 Motivation

In a few words, the main aim of our study is to verify “how good” a machine can be in recognizing daily activities, relying on information extracted from different features. As a simple example we shall consider any mechanism, for the recognition of two different activities (namely “Having Lunch” and “Washing-up”), which can take input data from two features (namely “Near” and “Hold”). The mechanism monitors the actions a person is performing inside a kitchen. At time t the mechanism takes input data:

- the “Near” feature says that the user is close to the oven and to a shelf where he/she usually stores food;
- the “Hold” feature says that the user is holding a bottle of washing-up liquid.

The recognition mechanism processes this data (following some kind of reasoning on prior knowledge about human actions) and comes up with the following response:

- according to the “Near” feature, the mechanism would state that the user is performing the activity “Having Lunch” at time t;
- according to the “Hold” feature, the mechanism would state that the user is performing the activity “Washing-up” at time t.

Of course, one of the previous statements if false: the user is either having lunch or doing the washing-up (in the assumption that he/she cannot perform more than one activity at the same time). Which feature is more often the most suitable for the correct recognition of the user’s activities then?
When considering just one time instant as in this example, this question does not really make much sense. It is obvious that, for every feature, the mechanism would just guess the most probable activity implied by the input actions (i.e., being close to the oven itself implies that the user is most probably having lunch according to the “Near” feature, like holding a bottle of washing-up liquid itself implies that the user is most probably doing the washing-up according to the “Hold” feature). Instead, the interesting part comes as we consider sequences of actions, where each action takes place at different times. For each feature, the mechanism can in fact take into account whole sequences of actions instead of single ones. By making use of certain techniques, the mechanism can revisit and adjust the choices it made when considering shorter sequences: this makes it able to make intelligent choices when guessing the activities of the user.

1.3 Research scope

Ideally we could consider any kind of feature: an indefinite number of parameters is “extractable” from a home environment where a subject is moving and performing actions. Examples of possible features could be the posture, or gestures of the subject; nonetheless, for a combination of reasons, it could be hard to extract such kind of information from the domain environment. Fortunately there also exists information conceptually easy to retrieve: as an example, there are multiple ways to get the position of a subject moving inside an indoor space. Many of these ways are extremely easy to implement, and the risk of picking the wrong position at any moment is almost null (i.e., high precision, low noise).
In order to have an idea about (or, in the case of machines, to recognize) what a subject is doing at a given lapse of time, it is often sufficient to observe a single feature. It is also true that in theory, if we could consider multiple features all together at the same instant, the recognition would benefit from it. Or, rather surprisingly, this may not be the case as

several combined features would end up being confusing for a recognizer. This and other matters are being explored in our work and in this document.
The question we want to address with this study is the following:

To what extent multiple types of easily trackable features contribute to the direct recognition of daily activities in a real-world scenario?

1.4 Overall project

This work was conceived as part of a larger project in *Assisted Living* (technologies providing supervision or assistance with activities of daily living, coordination of services by outside health care providers, and monitoring of resident activities to help ensure their health, safety, and well-being*). The main project idea is a real-time tracking and simulation of a home environment to identify and avert potentially dangerous situations.

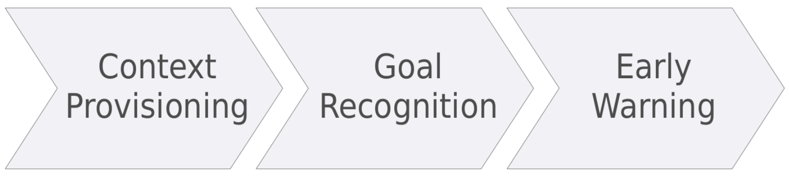


Figure 1.2: Project overview

The general approach is to embed a range of sensors in objects and appliances, in order to allow context awareness. By exploiting the context a state of the real world is described in an abstract language.
Eventually, a simulation, that simulates both the physics of objects and the behaviour of the human, runs a number of steps into the future, and watches for possible dangerous states (as defined by given rules). If the current behaviour of the human seems to be leading towards a disaster (according to the simulation), then the human is alerted through voice, sounds or projected visuals.
For this purpose, the goal of the user needs to be defined; conjoint use of activity recognition and prediction is exploited to achieve this.

* Website: http://www.wikipedia.org. Accessed May 2012

1.5 Thesis Overview

After the brief introduction to the problem we presented in this chapter, the rest of this document is organised as follows. Chapter 2 provides a discussion on activity recognition techniques, with particular focus on research being done in human activities recognition; general approaches taken in some recent works are briefly presented. We then provide a section about Hidden Markov Models, statistical models we make use of in our work. In Chapter 3 we present our approach, from the main elements to the differences with generally used approaches, basing our choices upon the research question we want to address; we also explore the concept of observational channels, which is of primary importance. In Chapter 4 we describe the implemented prototype: this part consists of a brief introduction about the files we use as input, the procedure for building the models for recognition of activities, the problems arising with our choices, a high-level description of the main functions provided, and output testing. Chapter 5 explores the modalities with which our experiments were conducted. We first mention the data collection method, then we describe the experimental setup, and lastly we discuss the results we obtained. In Chapter 6 we report our conclusions, and some ideas for future work to be done.

Chapter 2

Background

In this chapter we explore some of the concepts the reader would need to know about for a complete comprehension of this document; furthermore we refer to research being done in the field of human activities recognition, mentioning some works and existent approaches for handling problems somehow related to the one we want to address.

2.1 Activity Recognition in Assistive Technology

Most of the work being done for the automated activity recognition in home domains can be found in the field of Assistive Technology: people with some sort of disabilities (or inability to perform certain tasks) need particular care, nowadays not only given directly by humans. Assistive Technology touches across very broad fields. Health Science research in Assistive Technology is mostly on scoring how well people with disabilities can perform Activities of Daily Living (i.e., daily self-care activities), and helping them to avoid/reduce the effects of cognitive errors; Electrical and Electronical Engineering researchers work on new kinds of sensors and networking protocols; Computer Science on activity and plan recognition. Assistive Technology can refer to terms such as Ambient Intelligence (the use of sensors capable of working together to support people with disabilities and caregivers), Telemedicine (the transfer of medical information over telecommunications technologies), Telehealth (the delivery of health-related services and information) and Telecare (remote care in smart homes with ambient intelligence, using devices, networks, activity and plan recognition). State-of-the-art devices, networks, activity and plan recognition for physical health are presented in [PRA$^+$10], which also explores ideas for developing mental training, such as social networking for social health. In the last few years, superlative examples in the literature of human activity recognition

for dementia assistance are been developed. Phua et al. [PFB$^+$09] propose an erroneous-plan recognition system (EPR) aiming to detect defects or faults in the execution of correct plans by dementia patients, and to send timely audio and visual prompts to caregivers in order to correct these faults. [BGB07] introduce a keyhole plan recognition model to address the problem of recognizing the behaviour of Alzheimer's patients, transforming the recognition problem into a classification issue.
Assistive Technology applies not only to people with disabilities, but also to the elderly without dementia. More needs to be done in the field, which is a relatively new subject of study, to improve mental and social health of people with disabilities; furthermore, assistive software is not highly accurate and persuasive yet [PRA$^+$10].
Even though the research question we want to address does not specifically have to do with Assistive Technology and people with dementia/disabilities, the huge amount of work being done in this field should give an idea about the importance of human activities recognition and its applications.

2.2 Concepts in Recognition

In this section we present some aspects related to activity recognition: namely, input gathering through sensing devices and approaches generally used for low-level management of activities.

2.2.1 Sensing devices for input retrieval

Different kinds of devices can be configured to work together as sensing means for human behaviour. Various devices are widely available and relatively cheap [PRA$^+$10]: pressure sensors measure resistance changes after physical pressure is applied (for example, they make possible the detection of a person sitting on a chair); Pyroelectric InfraRed sensors measure infra-red light radiating from objects moving in their field of view (for example, they make possible the detection of a person walking into the dining area); Reed switch sensors are operated by an applied magnetic field (for example, for detecting when a person has opened the cupboard).
Radio-Frequency IDentification antennas and tags constitute a particularly interesting case of cheap devices with which it is possible to identify and track objects using radio waves, from different distance ranges (for example, near-field RFID tags can be used for detecting when a person has placed a glass on the table; mid-range RFID antennas and tags can be used to warn the user when he/she is getting close to a dangerous object

in certain conditions, for the prevention of unsafe situations). Recent works in activity recognition were done by making use of these devices, showing that the use of RFID sensor networks is a promising approach for indoor activity recognition and monitoring. Wu et al. [WOC+07] propose a dynamic Bayesian network model using RFID and video data, to jointly infer object labels and recognize low-level activities. Buettner et al. [BPPW09] instrumented everyday objects with UHF RFID tags equipped with accelerometers; RFID readers can detect whether objects are being used by examining sensor data, and daily activities are then inferred via a HMM from the traces of object uses.
More expensive devices that can be found in the market are accelerometers (which for example can detect hand movements), video sensors and Ultra-wideband transmitters (which for example could detect the number of people in a certain area, and annotate/audit their plans), and so on. Many other interesting (real) applications using some of these devices are also listed in [PRA+10].

2.2.2 Micro-context for activity recognition

“Micro-context” is any kind of information about actions performed by humans and about objects nearby; in some applications, micro-context was proved to be useful to help understanding the current activity/plan in progress. As an example, we could consider an accelerometer to get the micro-context needed to determine whether a mug is being drunk from or not. Biswas et al. [BSH+10] propose a sensor-based approach for the recognition of activities of mild dementia patients, showing that the knowledge of micro-contexts is useful for both activity recognition and prompted error correction. In their approach for the development of an activity recognition system, the recognition itself is made possible under certain conditions. A smart space is decomposed into a number of stations where activities are likely to occur; next, for each station, what is taken into consideration is the list of actions which could take place in there, and the list of objects that are likely to be used.
At least one major limitation exists in Biswas’ approach. All the stations need to be modelled, and for each of them all possible key actions are to be specified, as well as all key objects which are likely to be used in there. This information is likely to be very different from house to house, from person to person; engineering work needs to be done by hand every time the system is put to work under different conditions.
We will not use the concept of micro-context to be derived from the sensors, as it is not appropriate to what we want to build; however, we still need to take into account any kind of intermediate layer (instead of using direct communication with bare sensors) from which

it would be possible to get interactions with objects, or simple and general actions the user is performing at a certain moment.

2.2.3 A glimpse of general approaches

Assistive Technology is not the only field in which activity recognition can be useful; instead, activity recognition is applied to many real-life, human-centric problems. The recognition of simple human activities has so far been a successful research field, as optimal results have been achieved in many applications; yet complex activities recognition remains a challenging and active area of research. Aggarwal & Ryoo [AR11] provide an outstanding review on the state-of-the-art for human activity analysis from video data; even though the discussed approaches are specifically designed for computer vision applications, the review is still a good reference for the research being done in human activity recognition. All recognition methodologies are first classified into two categories:

1. *sigle-layered approaches*, in which the recognition is directly based on input sequences, and
2. *hierarchical approaches*, in which high-level activities are described in terms of simpler activities (sub-events).

Single-layered approaches are, due to their nature, largely used for the recognition of simple actions with sequential characteristics; instead, hierarchical approaches are mostly suitable for the analysis of complex activities (at least when using video tracking as a mean to get input data). Hierarchical approaches, as the name suggests, make use of several layers of abstraction: they assume the impossibility of guessing high-level activities without first guessing low-level ones.
Among single-layered approaches, state model-based approaches are probably the most widely and successfully used. State model-based approaches are sequential approaches, which represent a human activity as a model composed of a set of states. Normally each activity has its own model; when an input sequence is given, there is a mechanism which chooses between the available models to get the most probable activity the user is performing at a certain time interval. In the simplest case, where it is possible to automatically segment the input data to contain just one execution of a human activity, the only difficulty is to correctly classify the activity among all the models. In more general cases, things are not that easy though: many applications have to deal with continuous streams of data, which must be segmented before being classified. Sometimes this is not very hard work, as starting and ending times of all occurring activities can be somehow detected without problems (in computer vision, most single-layered approaches have adopted slid-

ing windows techniques to classify all possible subsequences [AR11]); other times such difficulties cannot be easily solved.
Hidden Markov Models (abbreviated HMMs) have by far been the most widely used probabilistic framework in activity recognition for state model-based approaches; they offer several advantages, such as clear Bayesian semantics, efficient learning and inferencing algorithms [NN07]. An activity is usually represented as a set of hidden states; humans are assumed to be in one state at time t, and each state generates an observation (also known as emission). At time $t + 1$ the system transits to another state (which could possibly be the same as before) considering the transition probability between states, as well as the probability of a given observation to be emitted by the states. Activities are commonly recognized by solving the evaluation problem., i.e., to calculate the probability of a given sequence generated by a particular state-model. In the next section an overview on HMMs is provided.

2.3 Hidden Markov Models

Various definitions for HMMs exist. As defined in [NN07], HMMs are basically "*a class of Dynamic Bayesian Networks (DBN) where there is a temporal evolution of nodes*". The reader not familiar with these concepts could refer to [RN10] for an extensive and detailed description of the concepts of Bayesian Networks and Probabilistic Reasoning in general.
Rabiner and Juang [RJ86] define a Hidden Markov Model as a "*doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols*". The authors propose effective examples to illustrate what HMMs can handle.
Probably the most "digestible" definition of HMMs is provided in [War96]:
"*The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multidimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. It is only the outcome, not the state visible to an external observer, therefore states are hidden to the outside; hence the name Hidden Markov Model.*"
In the rest of this section we are going to introduce the essential notation and concepts regarding HMMs, which will be useful for the comprehension the rest of this work. For a full understanding of HMMs, the reader can refer to [RJ86, Rab90].

2.3.1 Notation

According to Rabiner's notation [RJ86, Rab90], an HMM is characterized by these elements:

- N, the number of states in the model. Although the states are hidden, it is important to note that in the model there is some kind of physical significance attached to them. We denote individual states as

  $$S = \{S_1, S_2, \dots, S_N\}$$

  and the state at time *t* as q_t.

- M, the number of distinct observation symbols, i.e., the discrete alphabet size. The observation symbols correspond to the physical output of the system being modelled. We denote individual symbols as

  $$V = \{v_1, v_2, \dots, v_M\}$$

- $A = a_{ij}$, the state transition probability distribution, where

  $$a_{ij} = P[q_{t+1} = S_j \mid q_t = S_i]$$

  with $1 \leq i, j \leq N$.

- $B = b_j(k)$, the observation symbol probability distribution in state j, where

  $$b_j(k) = P[v_k \text{ at } t \mid q_t = S_j]$$

  with $1 \leq j \leq N$ and $1 \leq k \leq M$.

- $\pi = \{\pi_i\}$, the initial state distribution, where

  $$\pi_i = P[q_1 = S_i]$$

  with $1 \leq i \leq N$.

Given appropriate values for all of these elements (N, M, A, B, π), the HMM can be used both as a generator and as a recognizer for an observation sequence $O = O_1 O_2 \dots O_T$, where each observation O_t (also called "emission") is a symbol from V, and T is the number of observations (the length of the "emission sequence"). For convenience, we use the following compact notation to indicate a Hidden Markov Model λ and its complete parameter set:

$$\lambda = (A, B, \pi)$$

In the next paragraph we will consider a well-known example, to fix ideas and to set the described notation straight.

2.3.2 A simple example

Durbin et al. [DEKM98] provide an as excellent as straightforward example throughout the understanding of HMMs: the "Unfair Casino" problem. Here a dealer in a casino occasionally exchanges a fair die with a loaded one. We are only able to observe the throws of the die, but we cannot see when these exchanges happen; we do not even know whether they actually happen.

This example can be modeled with two states, as the die is either fair or loaded. Therefore $N = 2$, and

$$S = \{S_F, S_L\}.$$

The observation symbols are the possible results coming out after the dealer throws the die: obviously $M = 6$ and

$$V = \{1, 2, 3, 4, 5, 6\}.$$

We should now define the probabilities in the model. Please note that all following probabilities are arbitrary, and there are no particular reasons for their choice: this is just for the sake of argument.

Since the dealer can unpredictably switch the die whenever he wants, we assume the transition probability distribution A to be:

$$\boxed{\begin{array}{l} P[q_{t+1} = S_F|q_t = S_L] = P[q_{t+1} = S_L|q_t = S_F] = 0.05 \\ P[q_{t+1} = S_F|q_t = S_F] = P[q_{t+1} = S_L|q_t = S_L] = 0.95 \end{array}}$$

The emission probability distribution B can be described as:

$$\boxed{\begin{array}{l} P(1|S_F) = P(2|S_F) = P(3|S_F) = P(4|S_F) = P(5|S_F) = P(6|S_F) = \frac{1}{6} \\ P(1|S_L) = P(2|S_L) = P(3|S_L) = P(4|S_L) = P(5|S_L) = \frac{1}{10}\ ; \quad P(6|S_L) = \frac{1}{2} \end{array}}$$

The initial state distribution π_i has no preference for either state, meaning that the dealer could start with throwing the fair die with the same probability he could start with the loaded one.

$$\boxed{P[q_1 = S_F] = P[q_1 = S_L] = \frac{1}{2}}$$

Our model is graphically represented in Fig. 2.1.

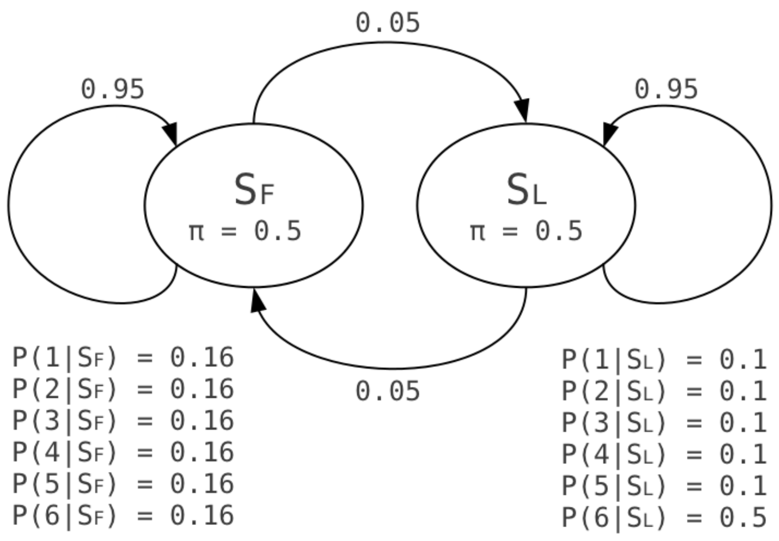


Figure 2.1: Hidden Markov Model for the Unfair Casino problem

2.3.3 The three basic problems for HMMs

In real-world applications of Hidden Markov Models, there are three fundamental problems of interest to be solved [RJ86, Rab90]. Simple applications would possibly require just a subset of them to be solved, but in general these three problems are the heart of the whole theory behind HMMs. Here we briefly present them:

- *Evaluation Problem:* Given the observation sequence $O = O_1 O_2 \dots O_T$ and a model $\lambda = (A, B, \pi)$, how do we efficiently compute the probability of the observation sequence $P(O|\lambda)$?
- *Decoding Problem:* Given the observation sequence $O = O_1 O_2 \dots O_T$ and a model $\lambda = (A, B, \pi)$, how do we choose a corresponding state sequence $Q = q_1, q_2, \dots, q_T$ which is optimal in some meaningful sense (i.e., best "explains" the observations)?
- *Learning Problem:* How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?

In the Evaluation Problem we want to know how to compute the probability of occurrence of the observed sequence. This is especially useful when we have to choose among several competing models, so that we can pick the model which best matches the observations; nevertheless, even when we are talking about just one model, the solution to this problem is very useful in many applications.
In the Decoding Problem we try to guess the correct state sequence, "uncovering" the hidden part of the model. In our work we put special focus on this task, as the solution to it is particularly useful for the kind of recognition we are interested in.

The Learning Problem has to do with the optimization of the parameters in HMMs, so as to best describe how a certain observation sequence comes about; a new sequence is used to “train” the model, and this is called a “training sequence”. The solution to this problem is crucial for most applications of HMMs as it allows to optimally adapt model parameters to observed data.

2.3.4 Solutions to the three problems of HMMs

The solutions to the three basic problems of Hidden Markov Models are briefly presented in this section. All the presented material is based on the tutorial [Rab90]; the reader might want to refer to that document for further details and clear explanations of formulae.

Evaluation Problem: the Forward-Backward Procedure

The problem is to calculate the probability of the observation sequence $O = O_1 O_2 \dots O_T$, given the model $\lambda = (A, B, \pi)$. The most straightforward way of doing this is through enumerating all possible state sequences of length T; however, the calculation of $P(O|\lambda)$ would require in this case on the order of $2T \cdot N^T$ calculations, since at every $t = 1, 2, \dots, T$ there are N possible states that can be reached, and for each such state sequence about $2T$ calculations are required. Even for small values of N and T, such calculation is computationally unfeasible: for example, 5 states and 100 observations would lead to $2 \cdot 100 \cdot 5^{100} \approx 10^{72}$ computations. A more efficient way to solve the Evaluation problem is the Forward-Backward procedure, which we now present (n.b.: strictly speaking, the Evaluation problem can be solved with the forward part of the procedure; the backward part is useful to help solve the Learning problem).
Consider the forward variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(O_1 O_2 \dots O_t, q_t = S_i|\lambda)$$

i.e., the probability of the partial observation sequence (until time t) and state S_i at time t, given the model λ. The value of $\alpha_t(i)$ can be solved inductively, as follows:

1. Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1),\ 1 \leq i \leq N.$$

2. Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}),\ 1 \leq t \leq T-1,\ 1 \leq j \leq N.$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

If we examine the computation involved in the calculation of $\alpha_t(j), 1 \leq t \leq T, 1 \leq j \leq N$, we see that it requires on the order of N^2T calculations, much better than $2T \cdot N^T$ as required by the direct calculation. The key is that since there are only N states, all the possible state sequences will remerge into these N nodes, no matter how long the observation sequence.
In a similar manner we can consider a backward variable $\beta_t(i)$ defined as

$$\beta_t(i) = P(O_{t+1}O_{t+2}\dots O_T|q_t = S_i, \lambda)$$

i.e., the probability of the partial observation sequence from $t+1$ to the end, given state S_i at time t and the model λ. The value of $\beta_t(i)$ can also be solved inductively, as follows:

1. Initialization:

$$\beta_T(i) = 1,\ 1 \leq i \leq N.$$

2. Induction:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}b_j(O_{t+1})\beta_{t+1}(j),\ t = T-1, T-2, \dots, 1,\ 1 \leq i \leq N.$$

The backward, as well as the forward calculations are extensively used to help solve the other problems.

Decoding Problem: Viterbi Algorithm

There are several possible ways of solving the Decoding Problem, which consists in finding the optimal state sequence associated with the given observation sequence. The difficulty lies in the definition of the optimal state sequence, as there are multiple possible optimality criteria. The most widely used criterion is to maximize $P(Q|O,\lambda)$, which is equivalent to maximizing $P(Q,O|\lambda)$; the Viterbi Algorithm is used as a formal technique for solving this problem.
Given the observation sequence $O = \{O_1O_1\dots O_T\}$, the procedure to find the single best state sequence $Q = \{q_1q_1\dots q_T\}$ can be stated as follows:

1. Initialization:

$$\delta_1(i) = \pi_i b_i(O_1),\ 1 \leq i \leq N,$$

$$\psi_1(i) = 0.$$

2. Recursion:

$$\delta_t(j) = \max_{1\leq i\leq N}[\delta_{t-1}(i)a_{ij}]b_j(O_t),\ 2 \leq t \leq T, 1 \leq j \leq N,$$
$$\psi_t(j) = \operatorname*{argmax}_{1\leq i\leq N}[\delta_{t-1}(i)a_{ij}],\ 2 \leq t \leq T, 1 \leq j \leq N.$$

3. Termination:

$$P^* = \max_{1\leq i\leq N}[\delta_T(i)],$$
$$q_T^* = \operatorname*{argmax}_{1\leq i\leq N}[\delta_T(i)].$$

4. State sequence backtracking:

$$q_T^* = \psi_{t+1}(q_{t+1}^*),\ t = T-1, T-2, \ldots, 1.$$

Learning Problem: Baum-Welch Reestimation

The last problem is by far the most difficult problem of HMMs: determining a method to adjust the model parameters (A, B, π) to maximize the probability of the observation sequence, given the model λ. Given any finite observation sequence as training data, no optimal ways of estimating the parameters can be found. However, it is possible to choose $\lambda = (A, B, \pi)$ such that $P(O|\lambda)$ is locally maximized; this is done through the Baum-Welch method (note that other methods exist, but Baum-Welch re-estimation is the "standard de facto" procedure in most applications of HMMs).
A complete explanation of this method requires quite a lot of space. As the Learning Problem is not treated as a fundamental part of this work, we are not going to explore the algorithm in detail: we restrict ourself to say that a set of reasonable re-estimation formulae for π, A and B are, at a high-level point of view,

$$\overline{\pi}_i = \textit{expected frequency (number of times) in state } S_i \textit{ at time t=1}$$

$$\overline{a}_{ij} = \frac{\textit{expected number of transitions from state } S_i \textit{ to state } S_j}{\textit{expected number of transitions from state } S_i}$$

$$\overline{b}_j(k) = \frac{\textit{expected number of times in state } j \textit{ and observing symbol } v_k}{\textit{expected number of times in state } j}$$

Again, we invite the reader to refer to [Rab90] for further details.

Chapter 3

Approach

Our main focus in this work is on studying how the utilization of multiple features could provide effective means for the direct recognition of high-level human activities. We recall our research question from the first chapter:

To what extent multiple types of *easily trackable* features contribute to the *direct recognition* of *daily activities* in a *real-world scenario*?

3.1 Main aspects to address

In particular, the following aspects concerning the research question should be satisfied:

- *Easily Trackable*: the features to deal with (e.g., “Near”, “Position”, etc.) should not be too hard for a machine to retrieve.
- *Direct*: Data representing instances of the features should not be obtained from additional recognition mechanisms, and fine-grained activities should not be considered at all.
- *Recognition*: A “standard” recognition tool should be used; the focus should not be on comparing different techniques for activity recognition.
- *Daily Activities*: Special focus should be put on a “typical” set of high-level activities which can be performed in a home environment, on the assumption that these activities could be good representatives for the given domain.

- *Real-world Scenario*: Special focus should also be put on a "typical" domain from daily life, on the assumption that users' behaviour inside the domain could be a good representative for similar domains. Furthermore, input data for the recognition process should refer to plausible sources: associating sensors to certain objects could be awkward, even if it would help enhancing the recognition rates.

In this chapter, as well as in the next one, we provide further details about the above-mentioned aspects as addressed in our work.

3.2 Basic elements

In this section we explore the elements we base our approach upon: activities, channels, alphabets. We briefly present all of them in the following paragraphs.

3.2.1 Activities

We said that all activities we refer to in this thesis are high-level activities. It is not easy to define exactly how complex an activity is in order to be considered a "high-level" one; in general, we refer to activities that need a series of steps to be performed completely. Again, as we said, our approach does not deal with these "steps", or at least not directly.
MedicineNet[*], an online healthcare media publishing company, defines the "Activities of Daily Living" (ADLs) as *"The things we normally do in daily living, including any daily activity we perform for self-care (such as feeding ourselves, bathing, dressing, grooming), work, homemaking, and leisure"*. In the field of Medicine, the ability/inability of a person to perform specific ADLs is widely used as a very practical measure of disability in many disorders (e.g., Alzheimer's disease and other types of dementia). Although our study is not focused specifically on people with diseases, and the domain of activities to take into account is not restricted to ADLs, we still refer to activities that are regularly performed ("daily activities"), and that need a variable amount of steps for their completion. We could possibly consider any kind of activity to be performed in a home environment; to restrict our field, it is probably a good idea to consider only activities being performed in a specific part of a home environment, such as a kitchen (e.g., "Cooking", "Eating", "Washing-Up") or a bathroom (e.g., "Cleaning", "Showering", "Grooming").

[*] Website: http://www.medicinenet.com. Accessed May 2012

3.2.2 Channels

The use of what we call “observational channels” is primary in our work. An observational channel (channel for short) is a parameter being observed in a person performing a certain task; in principle, it represents what the user does under a certain point of view. The concept of channel is strictly related to the concept of feature. In fact, every channel represents a space where information about a certain feature is stored, extracted from the domain environment. The specification of a channel does not follow a particular scheme: we could specify a channel in whichever way we prefer, as long as it makes sense. Examples of channels include the following:

- “Hold”, a channel in which we can store data about which objects the user is holding at a certain moment. We assume that the user can hold a number of objects all together at a given time instant; we also assume the set of objects to be taken into consideration to be well defined.
- “Position”, a channel that stores the position of the user at a certain moment. In general, positions could be specified in many different ways; ideally we could even describe a position as a point in a (x,y,z) system (3-dimensional), but for simplicity we can specify it as a cell in a mono-dimensional system. Here the assumptions are that the user can only be in one cell at a given time, and of course the set of cells to be taken into consideration is well defined and finite.

Multiple channels can also be considered in conjunction, in order to get several kinds of information all together. The use of multiple channels is further deepened later on in this document.

3.2.3 Alphabets

As explained later in this chapter, we make use of Hidden Markov Models for the recognition of activities; it is then a good idea to use the proper vocabulary for the concepts we present in this and in the next chapters. As in its most general definition, an alphabet is simply a standard set of symbols of whichever nature; in the specific case of discrete HMMs, an alphabet represents the distinct emissions (observation symbols) which could be observed by a model. What is needed is a schematic way for representing the content of the channels; every possible configuration of a channel (e.g., in the case of the “Hold” channel, which objects the user is currently holding, and which ones are not held) should be correctly specified. In this way, when a certain configuration of a channel is observed, the model being used can evaluate it and directly recognize which activity is probably going on.

Every channel has its own alphabet. As a simple example we shall consider the alphabet for the channel “Position”; we can specify all the possible symbols that can be observed in the way we want, as long as it is well-defined. Intuitively, assuming that we are uniquely interested in tracking areas of the floor the user could walk on, what we need to do is to enumerate all possible cells in the domain environment (e.g., a room). If we consider twelve cells in a room, the alphabet for the “Position” channel could be specified as:

$$V_{Position} = \{ \text{C01, C02, } \ldots \text{, C12} \}$$

with $M = |V| = 12$, the number of observation symbols, using the notation from HMMs literature. The general rule is that only one symbol can be observed at each time; as a consequence, in case there are doubts about the user’s position (i.e., the user stands with one feet in a cell, and with the other feet in a different cell), a choice has to be made.

When multiple channels are considered, multiple alphabets consequently need to be taken into account. To fix ideas, consider the channels “Hold” and “Position”. The joint alphabet “Hold, Position” should contain all possible cases that could occur when considering these two channels; if the user is holding a “Mug” in cell “04” at a certain time, a symbol that is able to define the given situation must exist.

$$V_{Hold,Position} = \{ \text{Plate/C01, Plate/C02, } \ldots \text{, Mug/C04, } \ldots \text{, Book/C11, Book/C12} \}$$

3.3 Main differences to generally used approaches

A large part of the literature about recognition of high-level human activities makes use of multiple layers of abstraction, because high-level activities are seen as particular sequences of low-level ones. We believe that observational channels can help avoiding the modelling of these sequences, which could follow very different orders even when a single subject is considered. In fact, taking the activity “Cooking” as an example, there are hundreds of ways to perform it (also considering that we could cook many different dishes). Our system should be able to execute a continuous recognition of complex human activities, and this recognition should uniquely be based on input observations, nothing else. What we use as input observations are easily trackable configurations depicting a situation in the environment, under certain aspects (described by what we call “observational channels”). Hence we could say that what we want to achieve is a high-level recognition, but skipping any intermediate layer of abstraction for the activities (i.e., sub-events).

3.3.1 Input retrieval

Devices like sensors, tags, antennas and transmitters can be used in several kinds of applications; sometimes the question even becomes whether a lot of sensing devices are really necessary to fully recognize a significant set of basic activities [LCB06]. In our recognition system we need any kind of devices to retrieve simple information about the user (i.e., anything that is directly correlated to what the user is doing). Instead, we do not need any information about the environment: for example we do not need any device such as thermometers or humidity sensors, as they would not give us meaningful information about what the user is interacting with, or in general what the user is doing.
Since we are exclusively interested in understanding high-level human activities (e.g., "Preparing Lunch") we first need to get the exact actions the user is performing at a certain moment (e.g., "Hold a pot"). Every class of actions (e.g., the actions in which the user is holding an object) can be derived through different devices; in this work we do not focus on how to get the data, under the assumption that the information we need can be always and easily derived by using devices such as the ones described in the previous chapter, or higher-level structures such as semantic networks.
A semantic network is "*a graphic notation for representing knowledge in patterns of interconnected nodes and arcs*"*. These networks were first used in the areas of artificial intelligence and machine translation, but they have been recently applied to many other fields. A semantic network can provide all information about a certain context at a certain time; it is effective even on continuous updating, and it could be used to give information about all features in a given domain [Sor12].

3.3.2 Modelling activities with HMMs

Hidden Markov Models represent a standard framework being used for multiple types of recognition in many fields. As they offer several advantages (as we discussed in the previous chapter), and as they have been widely used in activity recognition, we decided to base our approach on this well-tested framework.
Normally, approaches making use of HMMs for high-level and low-level activity recognition have at least one thing in common. Each activity is represented by a HMM, and each state in the model is a "step" towards the correct completion of the activity. HMMs with such a structure are said to have a left-right topology (i.e., the underlying state sequence associated with the model has the property that as time increases the state index increases or stays the same). However, we said that no sub-events should be considered: our recog-

* Website: http://www.jfsowa.com/pubs/semnet.htm. Accessed May 2012

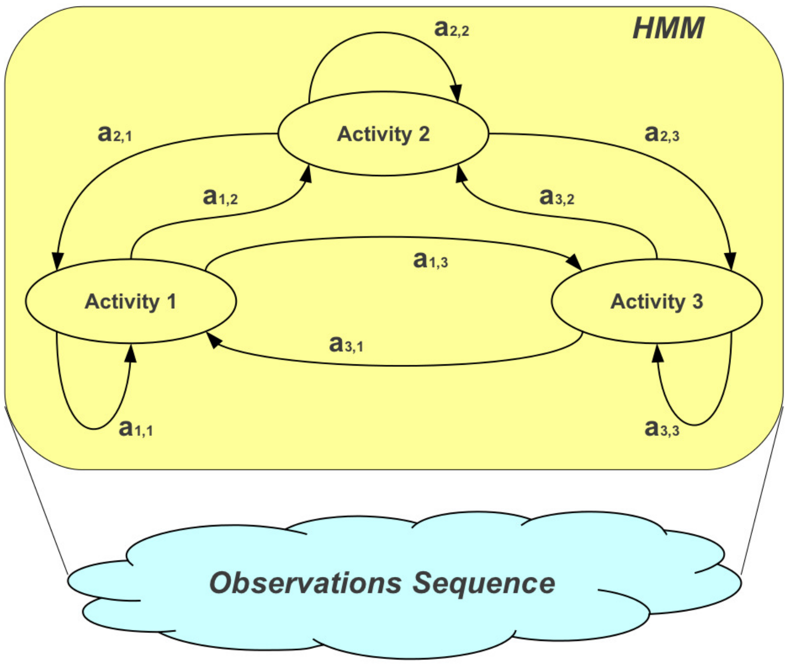


Figure 3.1: States in HMMs represent high-level activities; a subset of observations is considered by each HMM

nition should be direct, as no additional recognition mechanism should be necessary. In other words, a hypothetical high-level activity (like "Eating Lunch") should not be seen as a sequence of sub-events (such as "Taking Fork", "Eat with Fork", "Drink from Cup", etc.); the system should not even handle such elements. We do not want to model simpler activities along with more complex ones, as the latter are the only kind of activities we should be aware of; clearly an alternative approach has to be used.

The topology of the HMMs we consider is different from the generally used left-right models: indeed, as in [BPPW09], we use an ergodic topology (i.e., fully connected topology in which every state of the model can be reached, in a single step, from every other state of the model) for all the models we use. Fig. 3.1 shows a 3-states HMM with an ergodic topology. The reason for using this kind of models will be clear to the reader later on in this chapter.

Bad initial estimates of parameters, as well as insufficient training data, are major problems for HMMs. In theory, the algorithm solving the Learning Problem (e.g., Baum-Welch) should provide values of HMM parameters corresponding to a local maximum of the likelihood function; a key question is how to choose good initial estimates of parameters, so that such algorithm could work properly. There is no simple answer to this question. Sometimes the π and A parameters can even be taken either randomly or uniformly, giving an adequate basis for the re-estimation process; however, experience has shown that good initial estimates at least for the B parameters are helpful in the case of

discrete HMMs.
As our focus is on studying the effectiveness of observational channels in the real-world, and as we have no efficacious instruments for retrieving a huge quantity of data from different people, we decided to keep our approach certainly suitable for the particular case in which we consider only one user: we model our HMMs to be effective with data from a single subject, by estimating the A and B parameters on the basis of subject's habits.

3.3.3 Recognition tailored to a single subject

We said that our approach cannot be proven to be a general approach for an effective subject-independent recognition. In particular, for simplicity matters, our design of the models assures that the recognition would work well with a single subject, but we cannot say anything about how it would work with multiple subjects. By mixing information about multiple users, and by considering the users' habits (by means of transition and emission probabilities), it is possible that our models would not fit anyone very well.
Since we want to perform a direct high-level recognition, and as we chose not to make any assumption about how users perform certain tasks (e.g., what objects are likely to be used when performing an activity, where a subject is likely to be during an activity, etc.), information coming from "samples" of a single user's behaviour is the only help to rely on for the subsequent recognition. A single user has his/her own "behaviour patterns" when performing activities; a behaviour pattern is defined as "*a recurrent way of acting by an individual or group toward a given object or in a given situation*"[*]. It is not very hard to believe that different people generally have different behaviour patterns; there are multiple factors that could influence one's way of performing tasks, e.g., personality, age, sex, environment, etc. As an example we shall consider the activity "Relaxing": one person could be used to relaxing by reading a book at certain positions in a room (which is the "normal" case), others could find it relaxing to walk forward and backward in the room ("strange" case, but still plausible), and so on. Moreover, some people perform several tasks once inside a room, others could use the room only for certain tasks, and do some other tasks very rarely. All of these variables can contribute to define a "user profile", which somehow describes how a single subject behaves.
Recent works have shown that a simple HMM (with activities represented by states, as in our case) can be used for subject-independent activity recognition based on "object movements" (which can be considered as a feature) [BPPW09]; in some respects, our approach is rather similar to the mentioned work. The main difference is that we deal

[*] Website: http://dictionary.reference.com/browse/behavior+pattern. Accessed May 2012

with a real-world scenario. Unlike [BPPW09], where each activity can be performed in only few different ways, in our case all activities can be performed in a high number of possible ways. This is why we cannot assure that our approach would be suitable for subject-independent recognition. Taking into account all possible behaviours of users is an impossible task; the fact that our approach is subject-dependent does not represent a serious problem though. We explain what we mean by that later in this chapter.

3.3.4 Interactions with objects

In the previous chapter we have shown approaches, adopted in activity recognition, making explicit use of interaction with objects as a mean for the recognition process; some of them can be considered fairly related to our work. In [BSH$^+$10] activities taken into account are actually activity primitives, considered as micro-contexts (“Open cupboard”, “Bring out cutting board”, etc). An important aspect has to be pointed out though: the level of abstraction of the considered activities is lower than the level we are interested in. Wu et al. [WOC$^+$07] propose an approach based on the detection and analysis of the sequence of objects being manipulated by the user, making use of RFID-tagged objects along with video data and common-sense knowledge about which objects are likely to be used during a given activity. In our case we have no support of video cameras (which would imply some kind or additional recognition mechanism itself). Furthermore, we decided not to rely on common-sense knowledge, as we believe that it would be difficult to correctly recognize activities performed in “strange” ways; engineering is required for each considered channel.
Buettner et al. [BPPW09] use RFID tags equipped with accelerometers to detect object movements for helping the recognition process. RFID tags are placed on several objects, even cereal boxes and toothpaste. This approach is ideal for showing how the recognition rates could reach excellent results, but we believe that in a real-world scenario all of these clues would hardly be available: tags can surely be put on mugs and pots (which we use every day), but when it comes to elements we throw away after few days, it becomes unlikely. Furthermore, a set of precise tasks can be recognized (e.g., “Make sandwich”, “Make Coffee”, etc.), and of course the recognition of a diverse task (e.g., the preparation of different meals) could easily fail. Besides, the recognition of food preparation activities is proven to be perfectly possible and effective with Buettner’s approach, but no experiments on eating activities were performed (probably because it is hard to distinguish between preparation and eating only with the help of RFID tags).
The use of multiple features could be beneficial when not all possible information about objects is available. When a subject uses a set of objects that he/she also uses in the

preparation of different meals, the mechanism should be able to correctly categorize the high-level task; furthermore, a multi-channel system can be used to tell between similar tasks (such as “Eating” and “Preparing Food”), thanks to the additional information coming from different extracted features.
In Buettner’s approach the recognition is solely based on objects’ movements and proximity with the user; we want to study how effective other kinds of information can be in helping the recognition process. For example, an interesting aspect would be to determine whether information about On/Off states of household appliances could be misleading for such process. Indeed, another reason why we chose not to use any common-sense knowledge is the lack of generality for some channels: a subject could be used to eat sandwiches while the toaster is turned on, which means that the activity is “Eating” while the “Appliances” channel reports that the toaster is on. The use of common-sense knowledge would probably suggest that the user is “PreparingSandwich”, unless a multi-channel (and very complex) common-sense database is used.

3.4 Observational channels for Recognition

The series of steps to be followed when performing an activity is not always the same and can vary a lot: a person that is “Cooking” (high-level activity) does not always behave in the same way, especially because the dishes being cooked can be very different each day. One day a person wants to cook pasta, the day after he/she wants to have a toasted ham and cheese sandwich; the steps the person would perform in the first case (e.g., taking a pot, turn on the stove, put the pot on the stove, get a pan, preparing pasta sauce, etc.) are completely different with respect to the steps required to prepare a sandwich (e.g., turn on the toaster, get the bread, get the ham, etc.). Furthermore, even when considering a single dish, the person could follow different orders of steps obtaining the same result; some steps could even be missing from time to time, as we are not always willing to obtain a perfect dish. All of these factors make it hard for a recognizer to correctly categorize the actions being performed, and the models should necessarily be relatively complex for a precise analysis. By using observational channels, we address the problem of incomplete knowledge about all possible ways to perform certain tasks.

3.4.1 Starting and Ending times

It is not always obvious to tell when the moment of switching activity is occurring; in other words, determining starting and ending times for the activities is a problem. Since

we take simple configurations as input observations, it would be very hard to tell both when a certain activity is starting and when it is ending.
An external classifier could be used to guess the switching instant between activities; however, this classifier should be trained on all possible ways to start and finish an activity. Each channel is assumed to have a high number of configurations, and it would be necessary to consider multiple channels for the classification issue; furthermore, the user is assumed to be able to stop doing something and continue at a later time. For these reasons, this solution would not be precise (probably not even with a large training set).
In our view, this is not a severe problem. We believe that it is not extremely necessary to understand the exact moment in which a new activity is starting, or when an old one is ending; the important thing would be, instead, to guess the exact sequence of high-level activities (obviously the earlier we can recognize the activity, the better is in terms of reliability). This is made perfectly possible with the ergodic topology of HMMs we chose in our approach. In fact, when a recogniser "thinks" that an activity-switching is occurring, it would just change state accordingly. With a left-right topology we would have needed several HMMs (one for each activity), and it would have been hard to decide when to start/stop considering a model.

3.4.2 Observations

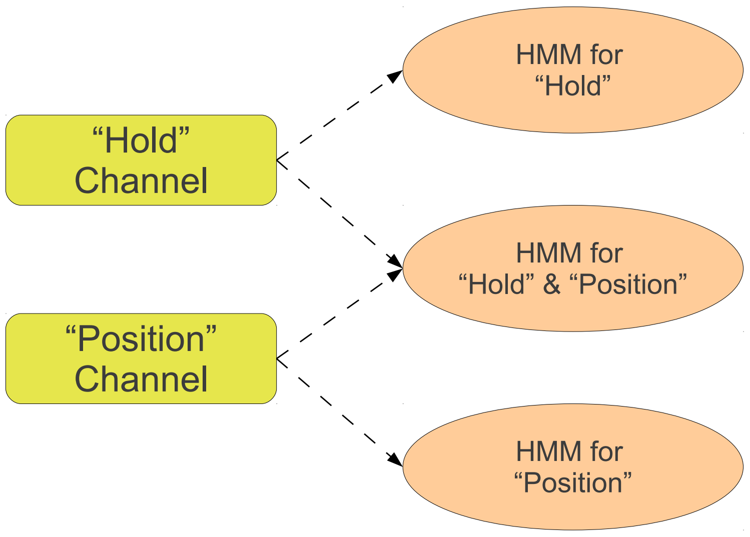


Figure 3.2: Example of relation between channels and models

Previously we said that every channel to be considered has its own alphabet. There is another important point to highlight now: each Hidden Markov Model is associated with a certain set of channels, i.e., every model takes into account different types of input at a given time (Figure 3.2); we can say that every HMM is actually a recognizer, which works in a completely independent way from the other recognizers. As a result it is possible to study how each model behaves, giving us an idea about "how good" a well-specified channel (or set of channels) could be in recognizing activities. This is especially good for making experiments with real-world data; before experimenting, we could not be extremely sure that considering all the channels is better than considering only a subset of them (in terms of recognition rates). Before claiming any result for this work, it would be necessary to understand whether the usage of all resources at our disposal is really necessary.

A single, entire observation is composed of information about all channels. Every time there is a change in the content of a channel, a new observation is generated; in other words the content of a given channel does not get refreshed at regular intervals of time, but only when the user performs an action that actually changes somehow the "configuration" of the world (i.e., the content of one of the channels). In other words, we chose an event-driven approach. We believe that the (correct) usage of observational channels' information would allow the recognition of human activities without considering observations at regular time intervals. Imagine the user playing a game on a mobile phone, holding a mug with his/her free hand, and not walking around the room: as long as nothing changes with this situation, no changes in the channels "Hold" and "Position" should occur. The channels would get refreshed accordingly as soon as the user stops playing and puts the mobile phone back in his/her pocket, or when he/she release the mug, or takes another object, or alternatively when he/she walks onto a different cell. Figure 3.3 shows a schematic view of channels "Hold" and "Position" for this example: the user is at first holding his/her mobile phone and the mug labelled with '02', and he/she is on the cell labelled with '04'; eventually the user puts the mobile phone back in the pockets, making the content of channel "Hold" change immediately and accordingly. Afterwards the user starts walking, changing position to cell '03' but still holding the mug.

3.4.3 Supervised specification of activity sequences

In the literature, many approaches for any kind of probabilistic recognition make use of existing models, trained upon common-sense knowledge about which objects are likely to be used during given activities (in the case of low-level activity recognition), or upon predefined sequences of actions to be performed for the completion of a task (in the case of high-level recognition). These works consider structured sequences of activities, but in real-world scenarios people can have unpredictable behaviours; as a consequence, in

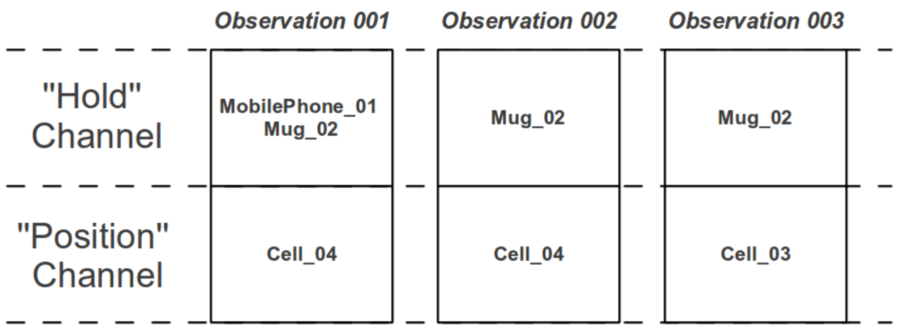


Figure 3.3: Example of observation sequence of length T = 3, considering channels "Hold" and "Position"

these cases recognition rates would degrade.

We do not make any assumption based on the common-sense; we believe that people can act in very different ways when performing the same task, and could also stop doing something and continue at a later time. This is why we chose to use a supervised specification of activities, a task that cannot be performed automatically by machines, but that should be somehow monitored by an external human. After few samples are processed, the models should be ready for the automatic recognition of activities. In general we expect these "few samples" to be around a dozen, depending on how long they are.

The samples are, intuitively, normal input sequences on which a human "supervisor" applies a specification (label) of the activities going on. This means that each sample is composed by multiple-channel observations with an additional field stating what activity is being performed at that instant; Fig. 3.4 shows the difference between input observations and sample observations.

In a real-world situation it is unlikely that an external human could "spy" the user for the whole training period: it would certainly be annoying. Instead of making an external human label the user's performed activities, an additional system (yet to be implemented) could be used for the supervised training of the models, thanks to user's labelling suggestions. The system would provide buttons the user can push once the corresponding activity (or part of it) is performed. To fix ideas, consider the following scenario in a kitchen:

- one day during the training period, the user starts performing the activity "PreparingFood";
- the user has to wait for his/her sandwich (in the toaster) to be ready: the user pushes the button "I have performed: Preparing Food";

- the user starts playing with the mobile phone; after a while, he/she realizes that one side of the sandwich is ready, but the other side is not. The button "I have performed: Relax" is pushed;
- the user changes side of the sandwich in the toaster, and after a while he/she pushes the button "I have performed: Preparing Food";
- the user starts thinking about his/her job and family, until the sandwich is ready. The button "I have performed: Relax" is pushed;
- the user gets the sandwich and eats it; once done, he/she pushes the button "I have performed: Eating" and leaves the room.

This additional system is perfectly feasible: a tailored, single-user approach is not a serious limitation, specially considering that it could handle real-world situations (unlike lab-specific recognizers).

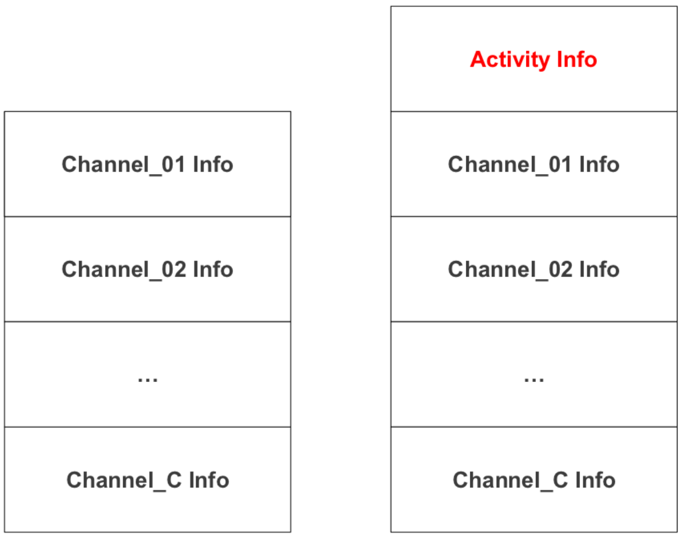


Figure 3.4: Input observation (left) and Sample observation (right).

3.5 Adding durations

As we said, our approach is event-driven. We hypothesize that, by using observational channels, there is no need to know what is happening in the environment at regular time intervals; what is important is to be always updated about new configurations of the channels, as soon as a change in the previous configuration occurs. Besides, the models we use for the recognition are by design not suitable for a time-driven approach. In particular, transition probabilities of the Hidden Markov Models we use would be largely affected by

continuous streams of (sample) observations of the same activity. If a person takes a lot of time when performing tasks like "Cooking" (which is very likely), the transition probability of such states/activities to go back in the same state/activity would be of course extremely high. This could negatively affect the recognition, because the probability of changing state would be, by consequence, extremely low.
Although a completely time-driven approach would probably be difficult to implement with our current view, two mechanisms for considering activity durations can be used: duration channel and duration extensions (Fig. 3.5). We briefly introduce these concepts in the following paragraphs, even though they are not to be considered a fundamental part of our approach: duration support for enhancing the recognition rates could be added in other ways we did not explore in this thesis.

3.5.1 Duration Channel

Durations can be taken into account by considering a new channel whose content is a discrete time interval, measured in seconds. A time interval takes the form $(time_i, time_j)$, with $time_k \in \mathbb{R}, i, j, k \in \mathbb{N}$, to be chosen from a set of possible time intervals:

$$(time_0, time_1) \mid (time_1, time_2) \mid ... \mid (time_{x-1}, time_x)$$

where $time_k \in \mathbb{R}, k \in \mathbb{N}$. For example, assume the set of possible intervals to be $(0, 5)|(5, 10)|(10, 500)$; assume also that an observation (i.e., all channels) remained unchanged for 6.3 seconds. The duration channel of this observation would store the interval $(5, 10)$, as 6.3 is a number between 5 and 10 seconds. Hence, the content of the duration channel represents the amount of time the current observation was "active" (i.e., all channels remained unchanged in their content).
The duration in this case is treated exactly as an additional channel, which brings information about how long the user did not trigger any event (e.g., changes in position).

3.5.2 Duration Extensions

Instead of considering a new channel, durations can be added by extending the alphabets of the channels. Defining the durations as the discrete time intervals described above, it is possible to append a duration to each channel. For example, assume the "Near" channel to contain the value "Mug, Book" at a certain time, without considering any durations. By adding a duration of 6.3 seconds, the channel would contain the value "Mug, Book; Duration: $(5, 10)$". In the case of extensions, the duration is not referred to the whole observation: every channel would contain a different duration, which is referred only to that channel.

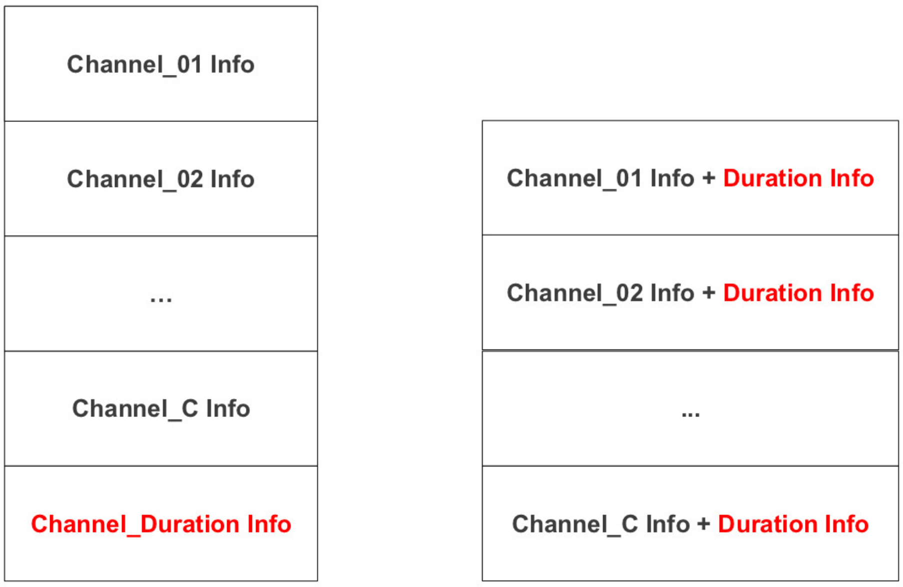

Channel_01 Info
Channel_02 Info
...
Channel_C Info
Channel_Duration Info
Channel_01 Info + Duration Info
Channel_02 Info + Duration Info
...
Channel_C Info + Duration Info

Figure 3.5: Durations support: structure of an observation considering the Duration Channel (left) and the Duration Extensions (right).

Chapter 4

Implementation

We developed our prototype in Python[*], a programming language which combines high power with clear syntax. As from its official website: *"Python is a remarkably powerful dynamic programming language that is used in a wide variety of application domains."* Among the key features it provides, we chose Python mostly for its clear and readable syntax, very high-level dynamic data types, extensive standard libraries and third-party modules which can be written/found in a variety of other programming languages (e.g., C/C++). As our approach is based on Hidden Markov Models, we used a freely available C library: the General Hidden Markov Model library (GHMM)[†] [SRSG04]. This library implements efficient data structures and algorithms for HMMs with discrete and continuous emissions, besides making the generation of synthetic observation sequences very easy; it comes with additional Python bindings which provide added functionality and a much more intuitive interface.

4.1 External files

As we said in the previous chapters, we disregard the type of sensing device for getting environmental information. Depending on the channel, there could be multiple ways to get the same information. The only important thing is that this information should not be biased, and the risk of getting noisy observations should be minimized; in particular, no additional recognition process should be allowed (e.g., video cameras for recognizing the users' movements). Photosensors could be used for getting the position of the user; RFID tags equipped with accelerometers, along with an RFID reader (e.g., to be used as

[*] Website: http://www.python.org. Accessed May 2012
[†] Website: http://www.ghmm.org. Accessed May 2012

a bracelet, or placed on the ceiling), could be used for getting what the user is holding; a central system could provide information about which household appliances are switched on in a room. All of these devices can operate with minimal possibility of failure.
In actual fact, our implementation assumes the input information to come from a common source (Fig. 4.1). As soon as the content of a channel mutates, a new observation has to be generated from such source (even if the other channels did not undergo any change in their content); this restriction can be managed through mechanisms we abstract from. As we said, ontologies, semantic networks or other structural frameworks for representing knowledge about the world (or part of it) can be easily used as a basis for dealing with such issues.

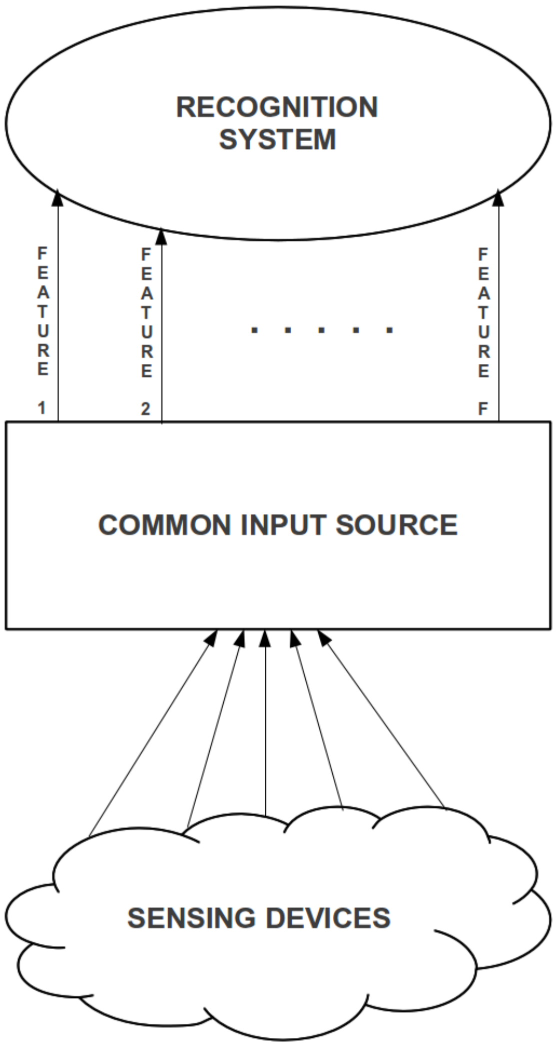


Figure 4.1: High-level scheme representing input abstraction: all sensing devices of whichever nature send data to a common input module, which in turn acts as an input source for the recognition system. Input is grouped by features.

Our preliminary implementation makes use of different kinds of files, in order to get the necessary information being used during the execution. These files can be categorized into the following: specification file, sample files and input files. We now briefly present an overview of them.

- *Specification file:* an XML file stores the specification of all activities and channels to take into account. For the activities, only the list of names is required (e.g., "Bathing", "Grooming", etc.). As for the channels, we not only need to provide their names, but we must also specify all configurations of their possible values (note that individual values represent emission symbols, and the set containing all symbols for a channel represent the channel's alphabet). The shape of these configurations depends on what we want to represent with them. For example, in a hypothetical "Position" channel we could specify all the possible cells the subject could be placed on; the best practice would be to assume that the user cannot be in more than one cell at a given time. In this case the emission symbols (configurations) of the "Position" channel would correspond to the cell names (e.g., "Cell1", "Cell2", "Cell3"). Instead, a "Hold" channel should represent the list of objects the user holds at a certain moment. The configurations of the "Hold" channel are logically all the possible situations which can arise. For example, considering the two objects "Pot" and "Pan", there are four possible situations: no objects held; pot is held; pan is held; pot and pan are both held.
- *Sample files:* the sample files, written in XML, represent the "supervised" observation sequences. In other words, each file defines a sequence of observations, and each observation carries additional information (i.e., a label) about the activity being performed at that time.
- *Input files:* the input files representing the input observation sequences. Of course no specification of activities has to be provided, as the (guessed) activity being performed should be the output coming out from the recognition process. These files are also written in XML format for this prototype, but in the future these informations could directly be taken from the common source we discussed previously.

4.2 Building models

In this section we explain the preliminary steps our prototype follows in order to build all Hidden Markov Models it uses for the recognition. As we have no initial parameter estimates for the models, and as we want to study the recognition under different points of view (e.g., different approaches for adding durations), as well as different experimental setups (besides the one we present in the next chapter), we decided to construct our models from scratch for this implementation. Future versions of this prototype will possibly count on pre-defined models, ready for being used and tested.

4.2.1 Initial probabilities calculus

We said we would not take into account any common-sense knowledge about users' ways to perform tasks; neither do we have a clue about how to set the initial parameters (A, B, π) for all the Hidden Markov Models we consider. There are several ways to obtain such estimation of parameters [Rab90]; to keep this process simple, one solution consists of including "manual" segmentation of observations sequences into states, with averaging of observations within states. As this method of supervised training has been proved to be effective in most of the cases (with a sufficient amount of sample observations), and as our approach assumes a set of these manually segmented observations to be present, we have chosen to implement this solution.

Transition probabilities

The state transition probability distribution A contributes to build a "user profile", which is like a specification of user's habits; in particular, A defines how likely the user is to pass from a certain activity to another.
The sample files contain observations with an additional field stating the activity which was going on at that moment. For the estimation of transition probabilities A we analyse all sample sequences, with no regard of observations themselves, and distinctly count all occurrences of activities performed; we then arithmetically calculate the probabilities of going from one state to another (possibly the same). Transition probabilities can be easily calculated with the following formula:

$$a_{ij} = \frac{A_{ij}}{\sum_k A_{ik}}$$

where i, j, k are states, and A_{ij} is the number of transitions from state i to state j. The calculation of transition probabilities is easy and fast, as all observations sequences have to be analysed only once.
As we said, we do not limit ourself to building a recognizer based on all available channels' information; we also want to study how effective the subsets of information can be in recognizing activities. All HMMs being used share the same A though: it is not important to know which channels are considered by a HMM, because the probabilities of going

from one state to another state only depend on the high-level activity being performed (as all HMMs share the same ergodic topology, with the same states).

Emission probabilities

The observation symbol probability distribution B defines the probabilities with which certain configurations of the channels occur, for each considered activity; with B the user profile becomes complete. Emission probabilities are also easy to calculate, with the following formula:

$$b_i(v) = \frac{B_i(v)}{\sum_w B_i(w)}$$

where i is a state, v, w emission symbols, and $B_i(v)$ the number of emissions of symbol v at state i.
Unlike transition probabilities, the estimation of emission probabilities B is quite time-consuming. In this case we need to take into account all observation sequences, and for each channel we analyse how often a certain emission occurs under a certain activity; then we proceed with calculating arithmetically all probabilities.
Emission probabilities must be calculated for each HMM. Obviously, all HMMs have a different set of emission probabilities B which depends on the channels considered: the more channels are considered (and the wider the alphabets are for each channel), the more time-consuming is the estimation process for the emission probabilities of a HMM.

Initial State probabilities

The initial state distribution π defines the probabilities of all states in a HMM to be a start state. This parameter could be set by means of simple arithmetical calculations as we do for the A and B parameters; this choice could lead to errors in case of scarce sample data. Assume our domain to be a kitchen, and all the times (in the sample observation sequences) the user has started cooking right after accessing the room. When a new observation sequence is given in input, and the first observation cannot be recognized as "Cooking" (i.e., that specific configuration has never been observed while the user was cooking, but only when performing other activities), an error occurs: in fact the model would assume $\pi_{Cooking} = 1$, and each observation sequence should necessarily start with a symbol which can be attributable to the "Cooking" activity.
To prevent a potential lack of sample data to spoil the recognition, we decided to use uniform initial estimates (e.g., if $N = 3$, $\pi = \{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$). In this way we make no assumptions on the starting state.

4.2.2 Unidentified emission symbols

In any set of sample observations, especially when the data set is small and the associated alphabet is wide, there is the possibility of certain events (i.e., emission symbols) not occurring. The observed frequency of that emission symbols should therefore be zero, implying a zero-probability; this situation is an inaccurate oversimplification in most of probability-based models. To avoid the zero-frequency problem it is possible to "manually" adjust the probability of (apparently) rare, but not impossible, events.
In order to allow "unidentified" emission symbols to have a non-zero probability to occur, several approaches have been proposed. The simplest approach is to add 1 (a pseudo-count) to each observed number of events, including the zero-frequency possibilities; this technique is called *Laplace's rule of succession*. By applying Laplace's rule, the formulae for calculating initial transition and emission probabilities become:

$$a_{ij} = \frac{A_{ij} + 1}{\sum_k \left(A_{ik} + 1 \right)}$$

where i, j, k states, A_{ij} number of transitions from i to j, and

$$b_i(v) = \frac{B_i(v) + 1}{\sum_w \left(B_i(w) + 1 \right)}$$

where i state, v, w emission symbols, $B_i(v)$ number of emissions of v at i.
This mechanism works fine until parameters re-estimation (via unsupervised learning mechanisms) is performed. In case re-estimation is used there could exist parameters for which the associated probability falls to zero (because they never occurred in the training sequence, no matter how large this sequence is).
One possible way of handling the effects of insufficient training data would be to add a constraint to the models, to ensure that no parameter estimates would fall below the value δ (i.e., $b_j(k) \geq \delta$). This constraint can be applied as a postprocessor to the re-estimation equations: when a constraint is violated, the parameter is manually corrected, and all other parameters are rescaled (in such a way that all densities obey the required constraints) [Rab90]. This and similar techniques have been applied to several problems (specially in speech recognition) obtaining good results [RJS83]: a future implementation could consider this hypothesis to deal with the problem of zero-probabilities.

4.3 Main functions

Our work is totally focused on the recognition of activities, and of course the recognition itself is the main feature provided by our prototype. We implemented an additional basic mechanism for the prediction of activities, and we added a small module, currently under construction, for the online re-estimation of parameters. We present a high-level description of all of these functions.

4.3.1 Recognition

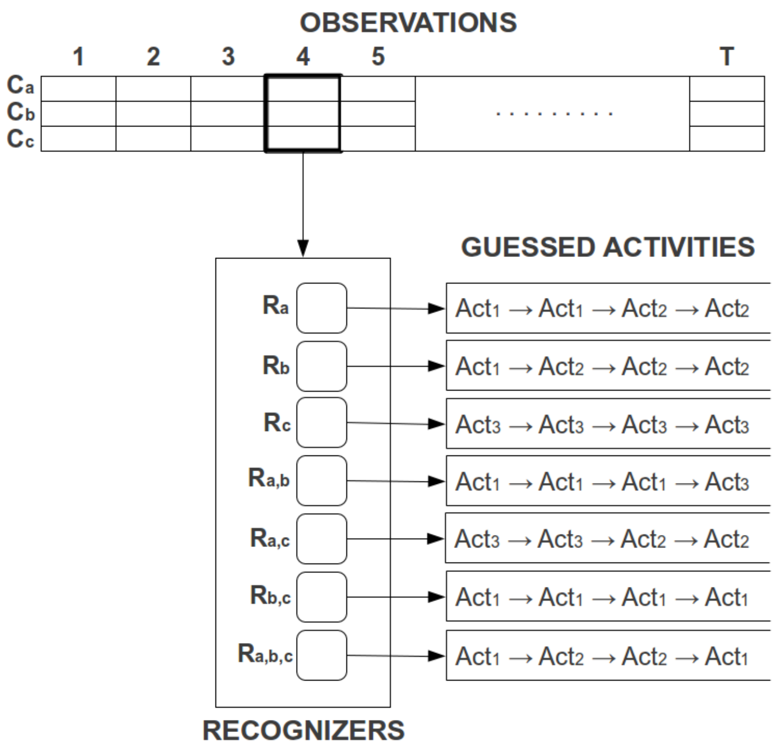

OBSERVATIONS
1
2
3
4
5
T
Ca
Cb
Cc
GUESSED ACTIVITIES
Ra
Rb
Rc
Ra,b
Ra,c
Rb,c
Ra,b,c
Act1 → Act1 → Act2 → Act2
Act1 → Act2 → Act2 → Act2
Act3 → Act3 → Act3 → Act3
Act1 → Act1 → Act1 → Act3
Act3 → Act3 → Act2 → Act2
Act1 → Act1 → Act1 → Act1
Act1 → Act2 → Act2 → Act1
RECOGNIZERS

Figure 4.2: Scheme of recognition mechanism with $|C| = 3$, and examples of recognized activities

Our input sequences are currently given as XML files. This means that the whole observation sequences (for all observational channels) are completely specified, but we only consider one observation at a time: every time we click a button on the keyboard, a new observation is processed. This is to simulate the real-world scenario, in which new observations are created every time a change in the environment occurs; in fact, our approach being event-driven, our prototype is also based on events. In other words, events are by now triggered through button pressure instead of through changes in the environment: the reader should understand the equivalence of these triggering events, the concept being

exactly the same.
A new observation is created if any channel changes state; for example, if “Channel1” changes configuration and “Channel2” does not, the new observation will just hold the current configurations of all channels (being the content of the first channel changed, and the content of the second channel unchanged).
Consider the example in Figure 4.2. Assume the set of available channels C to be composed by three elements, $C = \{C_a, C_b, C_c\}$. The number of recognizers we have at our disposal corresponds to the number of possible subsets of the set C (except the empty set), which is $2^C - 1 = 2^3 - 1 = 7$. In fact our set of recognizers is $R = \{R_a, R_b, R_c, R_{a,b}, R_{a,c}, R_{b,c}, R_{a,b,c}\}$, whose length is 7. Assume now that we have already analysed three input observations, and a new event was triggered: the current observation to be considered is the fourth observation. All recognizers have already tried to guess the activities being performed during the first three observations. We not only consider the last (i.e., fourth) observation, but all the sequence of observations until the last one. We do this by means of the Viterbi algorithm, so that we can find the most likely sequence of activities (also called Viterbi path) resulting in the partial sequence of observations. In other words, our problem is not limited to discovering the single most likely explanation for the last observation; we need to consider all the available observations, so that we can find the sequence of activities (being performed by the subject) that best explains what has been observed in a given set of channels (according to the model parameters).
When Viterbi is applied to the new sequence of four observations, the choices each recognizer made before (i.e., when applied to only three observations) can possibly change. In fact, Viterbi algorithm looks at the whole available sequence before deciding on the most likely final state, and then “back-tracking” to indicate how it might have arisen. When considering a new observation, Viterbi could even decide that the last observation was actually of the same class as the new one, according to probabilities. In the end, when all the observations are being analysed, the whole activity sequence which Viterbi tried to “explain” is given as a result.

4.3.2 Prediction

Our prototype provides an extremely simple implementation for predicting the possible next activity to be performed. As we put our focus entirely on the recognition, this feature is actually just a tiny additional module, not to be considered as relevant as the recognition part.
The prediction is not based on the observations, but on the activities. In fact, we assume the activity being performed at a certain time to be known, and the output should specify

which activity will be performed next. The prediction is based specifically on parameter A of whichever HMM we have, as all HMMs share the same transition probabilities. What is done is very simple: given the current activity $Act\_current$, the predictor just looks for the state (i.e., activity) for which the transition probability $a_{Act\_current,Act\_other}$ is maximized, without considering the same activity $Act\_current$. For example, assume the set of activities to be

$$S = \{Bathing, Grooming, WashingTeeth\}$$

and

$$a_{Bathing,Bathing} = 0.7, \quad a_{Bathing,Grooming} = 0.1, \quad a_{Bathing,WashingTeeth} = 0.2.$$

If the current activity is “Bathing”, then “WashingTeeth” will be predicted.
Ideally, in the subject-dependent case, a good prediction of activities could be jointly used with a good recognition: the input for the prediction would be the recognized activity, instead of just a given activity.

4.3.3 Online re-estimation of parameters

Assume an hypothetical subject using the system for a long time. After a sufficient amount of non-labelled observations is available, it would be possible to use unsupervised learning techniques for adjusting the initial parameters of the models. At this stage of our project we do not have much data about users; yet we are interested in the (future) possibility of training our models in an unsupervised way.
The fact that observation sequences used for training are finite (for necessity) constitutes one of the major problems associated with training HMM parameters via re-estimation methods. Often, an insufficient number of symbol occurrences within states leads to impossibility to give good estimates of model parameters. Possible solutions to this problem are the following:

1. Increasing the size of the training observation set: in our case this is impractical, as in a real-world scenario there may be a lot of possible configurations for the channels (especially when considering multiple channels), and good estimations could be achieved only having a huge quantity of (real) training data.

2. Reduce the number of states of the model: this is not possible, as our models’ states correspond to the activities to be recognized. Reducing the size of the model would mean being able to recognize a restricted set of activities.

3. Interpolate a set of parameter estimates with another set of parameter estimates from a model for which an adequate amount of training data exist: as we do not have other examples of models making use of observational channels for the recognition, this solution is not possible either.

For these reasons we chose to use synthetic data, for showing that unsupervised training can be applied. This is just to show that, in theory, our models can learn how a user behaves inside his/her environment, adjusting their parameters accordingly.
The GHMM library provides simple methods for the automatic generation of observation sequences, based on the initial parameters of a given HMM. Also, whatever length of synthetic observation sequences can be specified. Good results can be achieved with a sufficiently long sequence (e.g., 10000 episodes); all HMMs we use can be easily trained in this way. In a future implementation, real data would be possible to be used for unsupervised learning: it is only necessary to store real input data, and wait until sufficiently long sequences are observed. Again, we remind the reader that the zero-probability problem should be somehow avoided.
As discussed in the background part there are several ways to re-estimate the parameters of a given HMM, but the most widely used method is the Baum-Welch procedure. Our synthetic learning is implemented by making use of this method, as GHMM provides intuitive and very high-level functions for this re-estimation algorithm.

4.4 Testing output data

We provide a testing mechanism for both recognition and prediction (again, we remind the reader that the prediction part is just an additional tiny module). In the recognition case, given a certain input observation, the testing process requires the knowledge of the activity being performed at the time the observation was triggered. In the prediction case, given a certain activity (performed during a certain sequence of observations), the process requires the knowledge of the next activity which will be performed.
In general, for each input file (without specification of activities) to be used as input for recognition and prediction, we assume that the same files include additional specifications of activities (for each observation) to be used for testing purposes. In reality, there is no difference between these “testing files” and the sample files.

4.4.1 Recognition

The recognition tester takes into account the real activities that are performed, so that it can compare them with the guessed activities and draw conclusions about the quality of the recognition. We implemented two methods for recognition testing: the “single observations recognition testing” and the “sequence similarity testing”. Each method has different aims, but they can be jointly used for quality assessment.

Single observations recognition testing

This testing method deals with the recognition of single activities (one by one) in the input sequence. In particular, the following three possibilities could occur for each observation:

- RECOGNITION OK: the guessed activity corresponds to the really performed activity at time t.
- SOFT ERROR: the guessed activity does not correspond to the really performed activity at time t, but the guessed activity will occur (in the future) at most after a given amount of steps, or it occurred (in the past) at most a given amount of steps before.
- HARD ERROR: the guessed activity does not correspond to the really performed activity at time t, and it cannot be related to any activity which will occur, or that has occurred.

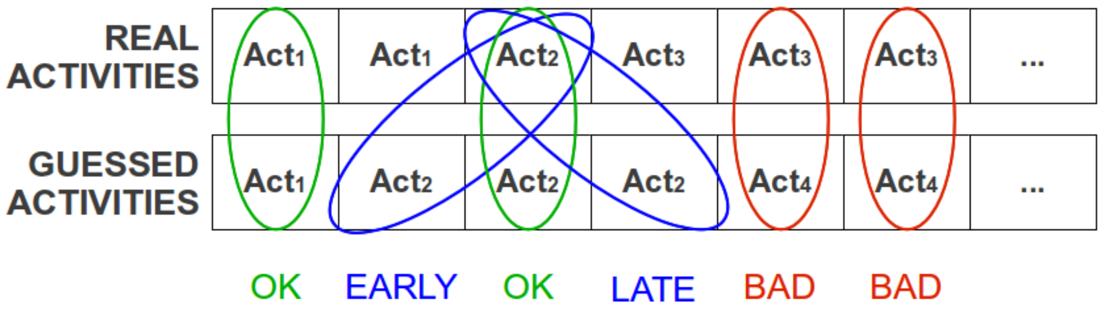


Figure 4.3: Single observations recognition testing: an example

Figure 4.3 shows an example of the result coming out by using this testing method. The “given amount of steps” we mentioned is a tolerance value: we can tolerate recognition errors as long as they occur inside the tolerance range $[t - tolerance, t + tolerance]$, and such errors are defined as “soft” (please note that the tolerance value in Figure 4.3 is equal

to 1). Soft errors are in turn classified into “Early recognition” (in the case an activity gets recognized at most $tolerance$ steps before it actually starts) and “Late recognition” (in the case an activity gets recognized at most $tolerance$ steps after it actually ends).
We consider the $tolerance$ value because it is hard to decide when activities start/end, even for a human observer. If a subject has just finished preparing some food and is going to do the washing-up, how do we decide when exactly the activity-switching occurs? How can we tell whether a configuration like “Hold a mug, Position in cell03” is still under the activity “Preparing Food”, or already under “Doing Washing-Up”? Even in the sample files we basically can just approximate our choices. This is why an early or late recognition is sometimes not seen as a hard recognition error. By tuning the value of $tolerance$ we decide how tolerant we are to errors.

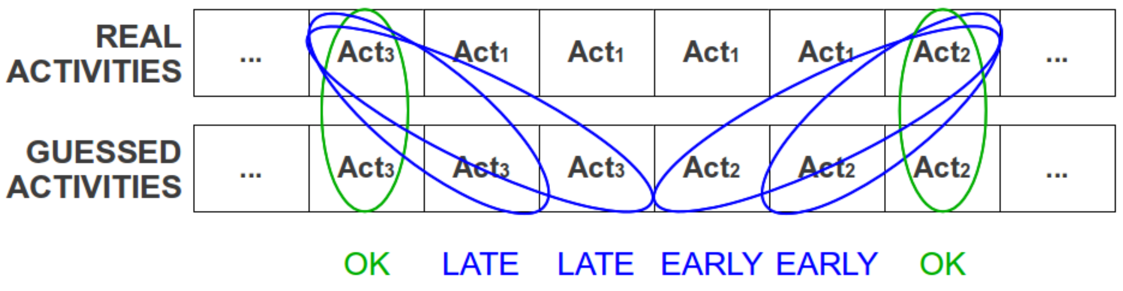


Figure 4.4: Single observations recognition testing can fail in recognizing correct sequences

At every step, we can calculate the error rates: we can count how many single activities were recognized, how many soft errors were made, how many hard errors, and then calculate the percentages.
As this method can give us an idea about how many observations are correctly classified, it is normally good for testing our recognizers. However, sometimes it could fail in understanding whether the right sequence of activities was correctly guessed. Consider the example in Fig. 4.4. Here the activity Act_1 is not recognized at all, and this situation should be categorized as a clear case of bad recognition (i.e., hard error). Instead, by setting $tolerance = 2$ (which is generally a good value for our tolerance), the testing phase results in just soft errors. Clearly, a solution to this problem should be found.

Sequence similarity testing

The second method we propose deals with the similarity of the real and guessed activity sequences (instead of recognition of single observations as the first method does). Note

that this method is related to the recognition, but it is not to be considered a method for testing the recognition percentage; this method uses the Levenshtein distance between two strings.
The Levenshtein distance is a metric for measuring “how different” two sequences are; it is defined as *“the minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character”*[*]. As a simple example, consider the strings “kitten” and “sitting”: the distance between these two strings is 3, since three is the minimum amount of edits that are necessary to transform one string into the other.

1. *kitten* $\longrightarrow$ *sitten*
   (substitution of ’k’ with ’s’)
2. *sitten* $\longrightarrow$ *sittin*
   (substitution of ’e’ with ’i’)
3. *sittin* $\longrightarrow$ *sitting*
   (insertion of ’g’ at the end)

It can be proven that there is no way to transform the source string (kitten) into the destination string (sitting) with a smaller amount of steps. Of course, the lower the Levenshtein distance, the more similar the strings are: perfectly identical strings have distance zero, while completely different strings have distance equivalent to the length of the longest string. On the website of Merriam Park Software[†], Michael Gilleland provides an outstanding explanation of the Levenshtein algorithm; the reader can refer to that for further details.
Coming back to the sequence similarity testing: in this case we do not consider all observations with their specified activity; instead, we only consider which activities are actually performed, and we put them in the temporal sequence (path) in which they are executed. For example, assume the sequence of real activities to be Act_1, Act_1, Act_1, Act_3, Act_3, Act_2, Act_2, Act_2, Act_4, Act_4: the real activities’ path is then Act_1, Act_3, Act_2, Act_4. Assume also the guessed activities’ path to be Act_1, Act_2, Act_4 (as in Fig. 4.5). In order to check how good a certain model is in recognizing, we simply compare the two paths by means of the Levenshtein distance. In this example the distance between the two paths is 1, as Act_3 was not recognized (this is a case of deletion of one element in the path).

[*] Website: http://rosettacode.org/wiki/Levenshtein_distance. Accessed May 2012
[†] Website: http://www.merriampark.com/ld.htm. Accessed May 2012

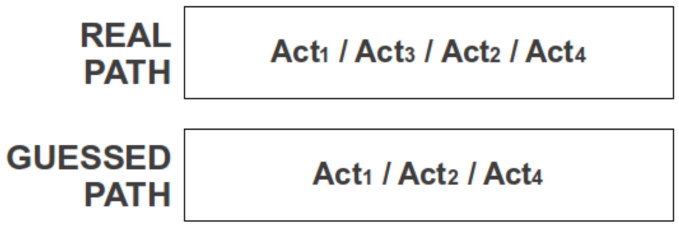


Figure 4.5: Sequence similarity testing: an example

Even in this case we can calculate a recognition rate, this time referred to activity path recognition. There are different ways to calculate a percentage out of a calculation of Levenshtein distance; after doing some research, we found the following formula to be widely used in measuring the percentage of similarity between two strings:

$$R = 100 * \left[1 - \frac{levenshtein}{max\Big(length(path_r),\ length(path_g)\Big)} \right]$$

where R is the path recognition rate, $levenshtein$ is the Levenshtein distance between the two paths, $path_r$ is the real path, and $path_g$ is the guessed path. Again, other possible formulae can be used in place of the just proposed one; no substantial changes in the obtained percentages would be obtained though.

4.4.2 Prediction

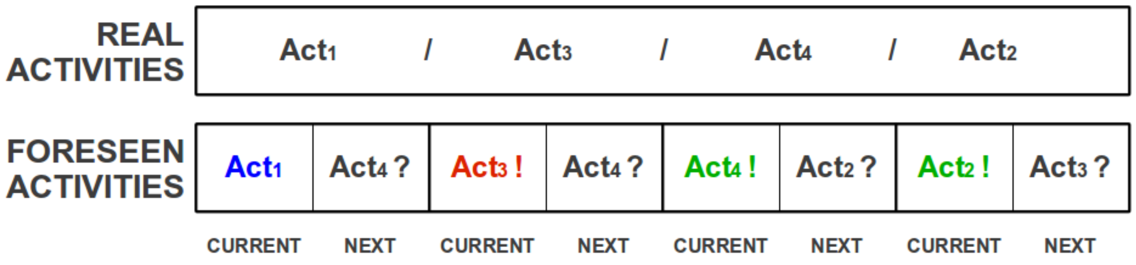


Figure 4.6: Prediction testing: an example

The testing phase for prediction is extremely simple: the only thing to test is whether the predictor has correctly foreseen the next activity the user is performing. The only two possibilities are then “Good Prediction” and “Bad Prediction”. To fix ideas, consider the example in Fig. 4.6: the path of real activities is $Act_1, Act_3, Act_4, Act_2$. We assume the first activity to be known; every time there is a switching in the (real) activity, the predictor tries to foresee the next activity. If the result is right we add a “Good Prediction” to the list of successfully foreseen activities, otherwise we add a “Bad Prediction” to the list of unsuccessfully foreseen activities. By counting the occurrences of good and bad predictions step by step (still by means of paths instead of observation sequences), prediction rates can be calculated.

Chapter 5

Experiments

In this chapter we present the modalities with which our experiments were performed. We start with a discussion about the procedure we followed for real data collection, given that no sensing device was at our disposal. We then define the basic experimental setup we used, and briefly discuss how we changed parameters for different experiments. Lastly we present the results obtained though the experiments.

5.1 Videos for real data

In this work we did not focus on how to get the data, as we assumed the information we need to be easily obtainable from sensing devices (even very simple ones). In particular, each feature (i.e., observational channel) is easily trackable: there is no need for further probabilistic recognition systems to derive data from. However, in order to try our prototype in a real-world scenario, we could not count on an existent system/structure to get real data from. We decided to solve this problem by observing a real subject (Andrea) performing activities inside a kitchen, and annotating his actions by hand according to the basic elements of our approach: activities, channels and alphabets.
For the observation of Andrea we first made 18 videos of him performing activities, by means of a video camera (Figure 5.1). For each video, we filmed the entire scene from the moment he entered the room until the moment he left. The average scene lasted for approximately 9:18 minutes, with the shortest being 3:38 minutes and the longest being 14:14 minutes; the average number of observations was approximately 51, with the highest number of observations in a scene being 108 and the lowest being 22. We let Andrea choose the activities he wanted to perform each time, without any restrictions on the number of activities: we just wanted to make sure he would not feel under pressure, and we

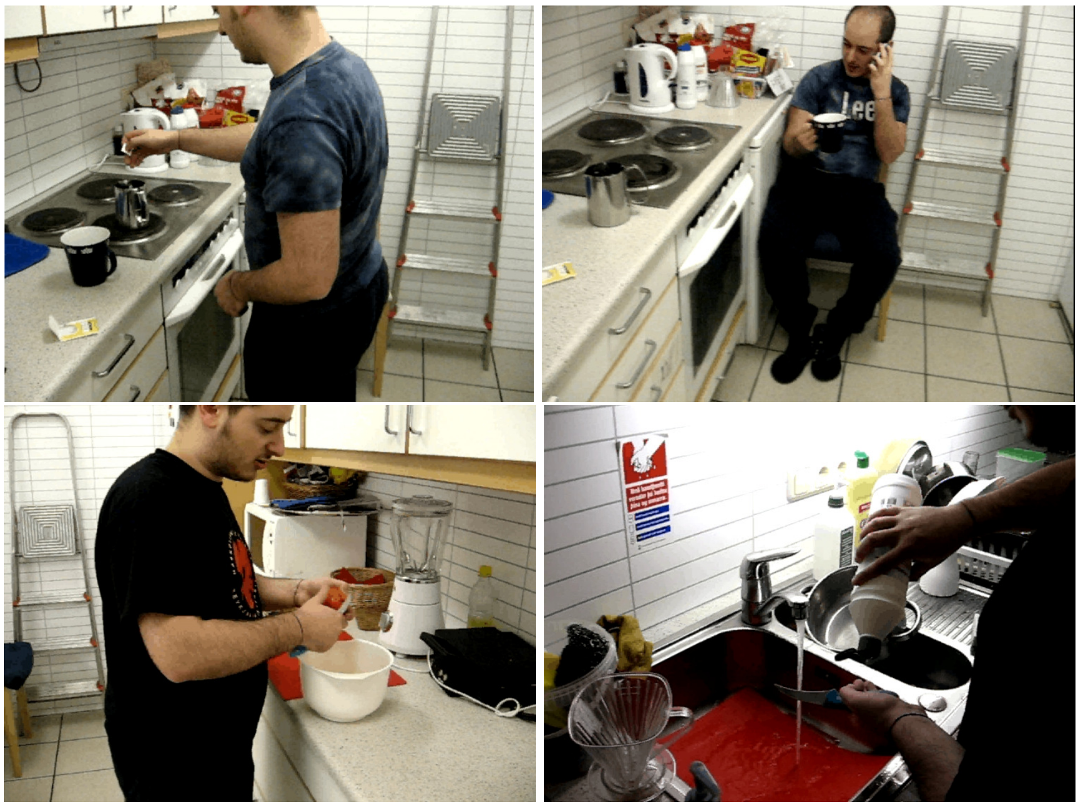


Figure 5.1: Screenshots extracted by videos of Andrea

wanted him to do what he would normally do when accessing the kitchen. We only asked him to respect the list of possible activities, and we also informed him about what objects and appliances we were going to track for the recognition. Note that he could also use other objects/appliances in order to perform his tasks: we wanted to simulate a scenario in which not all information is provided. In fact, in a real-world scenario it would be improbable to have sensors (e.g., RFID tags) everywhere and for each possible object; similarly, it would be unlikely to have a central system that controls every single appliance in the house, even the smaller and simpler ones.

Once all the videos were ready, we manually annotated them. We made use of ANVIL* [Kip01], a free video annotation tool mostly used in research. It was originally developed for gesture research, but it is now being used in many areas (e.g., Human-Computer Interaction, anthropology, psychotherapy, embodied agents, etc.). ANVIL, among its several features, offers multi-layered annotation: we used this important functionality for a very straightforward annotation of Andrea's observations (i.e., configurations) for each channel, as well as his performed activities.

ANVIL offers the generation of XML files out of annotated videos. The structure of these files can be defined, to a certain extent, according to a given specification: this is one of

* Website: http://www.anvil-software.de. Accessed May 2012

the reasons why, in our implementation, we are using a specification file (besides being useful to get basic information from). These XML files we are talking about can be directly taken and used as sample files: we obtained 18 sample files in this way (i.e., each video was converted into information, and considered as sample data).

5.2 Experimental setup

In the experiment we performed with Andrea, the basic elements (i.e., activities, channels, alphabets) were chosen as described in the following paragraphs.

5.2.1 Activities

We considered the following four activities:

- *Preparing Food*,
- *Eating*,
- *Washing-Up*,
- *Relaxing*.

The decision of taking this restricted number of activities has two reasons. First of all, the listed activities are good representatives of the tasks a person is likely to perform in a kitchen. Other activities could have been, for example, "Watching TV" (we did not have one in the real kitchen we have considered though) and "Clean Kitchen" (but this task is usually carried out pretty rarely, way more rarely than the other activities we consider). Second, as we use an ergodic topology for the HMMs, the less states we consider, the lower the complexity of the models.

5.2.2 Channels

We considered the following three channels:

- *Hold*,
- *Use*,
- *Position*.

The “Hold” channel represents which objects the user is holding at a certain moment. Speaking in terms of sensing devices, this information could be easily retrieved by means of RFID tags equipped with accelerometers on the objects as in [BPPW09], or perhaps in conjunction with RFID reader bracelets worn on the user’s wrists (which indicate whenever the user’s hands are close to tagged objects) as in [WOC$^+$07]. A certain amount of noise could result in the readings, but in general there is no need for further recognition mechanism, and this is what really matters; besides, as for all channels, other ways to get the same data can be used (again, we remind the reader that all possible modalities for retrieving data are out of the scope of this work).
The “Use” channel represents the appliances “being used” or turned on at a certain moment, or alternatively the status of the “important” parts of the kitchen. As for the appliances, we consider for example when the hob is turned on, but also when the door of the fridge is open (note that information about the internal status of the fridge would not be useful for the recognition, because this kind of appliance is likely to be always turned on). An important part of the kitchen could be the sink, which is not an appliance, but can be considered as an “agent” with an internal status (tap on/off). These information could be provided by a central system which monitors/manages some parts of the home environment: the status of each agent is constantly monitored by the system, which can also provide functionalities for sending information to other systems.
The “Position” channel represents the 2-dimensional cell the user is located on at a certain moment. This information could be provided by means of simple photocells, or alternatively by means of a grid of passive RFID tags placed on the ceiling or on the floor, as in [CCC10]. Again, specially in the case of RFID tags, a certain amount of noise could result in the readings; however giant steps were made in this relatively new technology during the last years, and good results have been achieved in indoor activity recognition.
By considering a restricted number of channels the (manual) annotation phase is easier and faster than it would be with more channels. Furthermore we believe that the chosen channels are good representatives of features to extract from an environment, and especially good to get useful information for the recognition from.

5.2.3 Alphabets

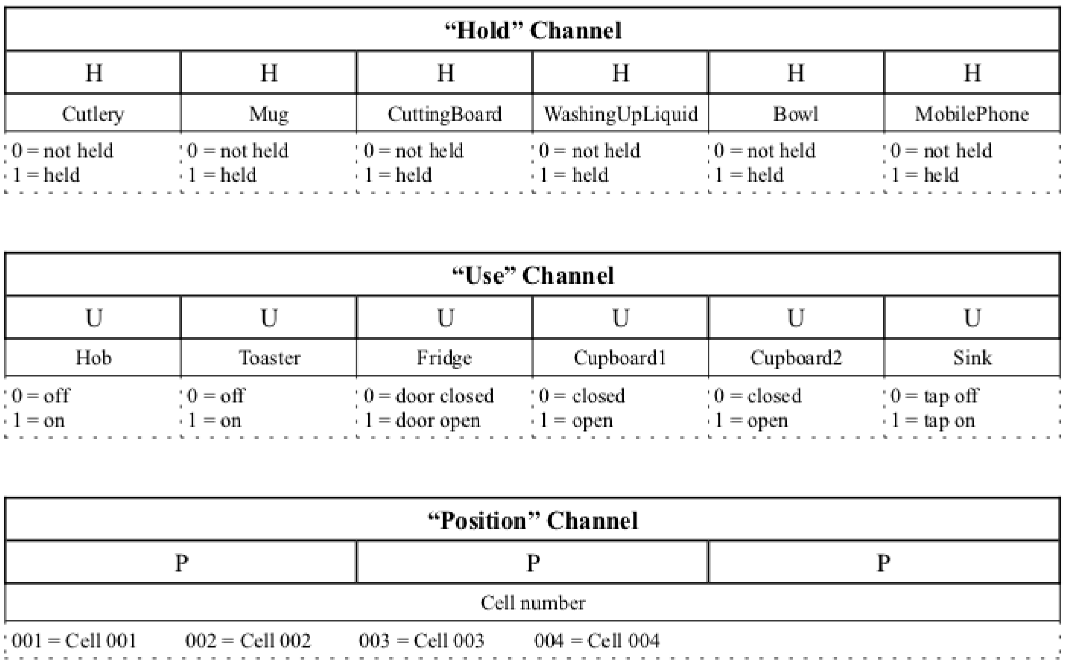
The alphabets for the channels we are considering are taken as configurations:

- “Hold” channel: configuration of the “hands” of the user (what is being held, and what is not);

- "Use" channel: configuration of the kitchen's appliances and main parts (what is on/open/used, and what is off/closed/unused);
- "Position" channel: configuration of the kitchen's places with respect to the user (where the user is located, and where the user is not located). In the assumption that the user can only be in one cell at a time, we can consider this not as a configuration, but as an ordinary value.

As for the durations, in these experiments we used only two intervals for simplicity matters:

- from 0 to 20 seconds (short duration for a configuration), and
- 20 or more seconds (long duration for a configuration).



"Hold" Channel					
H	H	H	H	H	H
Cutlery	Mug	CuttingBoard	WashingUpLiquid	Bowl	MobilePhone
0 = not held 1 = held	0 = not held 1 = held	0 = not held 1 = held	0 = not held 1 = held	0 = not held 1 = held	0 = not held 1 = held

"Use" Channel					
U	U	U	U	U	U
Hob	Toaster	Fridge	Cupboard1	Cupboard2	Sink
0 = off 1 = on	0 = off 1 = on	0 = door closed 1 = door open	0 = closed 1 = open	0 = closed 1 = open	0 = tap off 1 = tap on

"Position" Channel		
P	P	P
Cell number		

001 = Cell 001 002 = Cell 002 003 = Cell 003 004 = Cell 004

Figure 5.2: Alphabetic configuration meanings

The configurations we use are outlined in Fig. 5.2, along with all the objects we consider in the "Hold" channel, the appliances/parts of the "Use" channel and the cells of the "Position" channel. Figure 5.3 shows a screenshot of an example of ANVIL annotation: some of the considered symbols of the alphabets (configurations) are visible in the lower part of the figure.

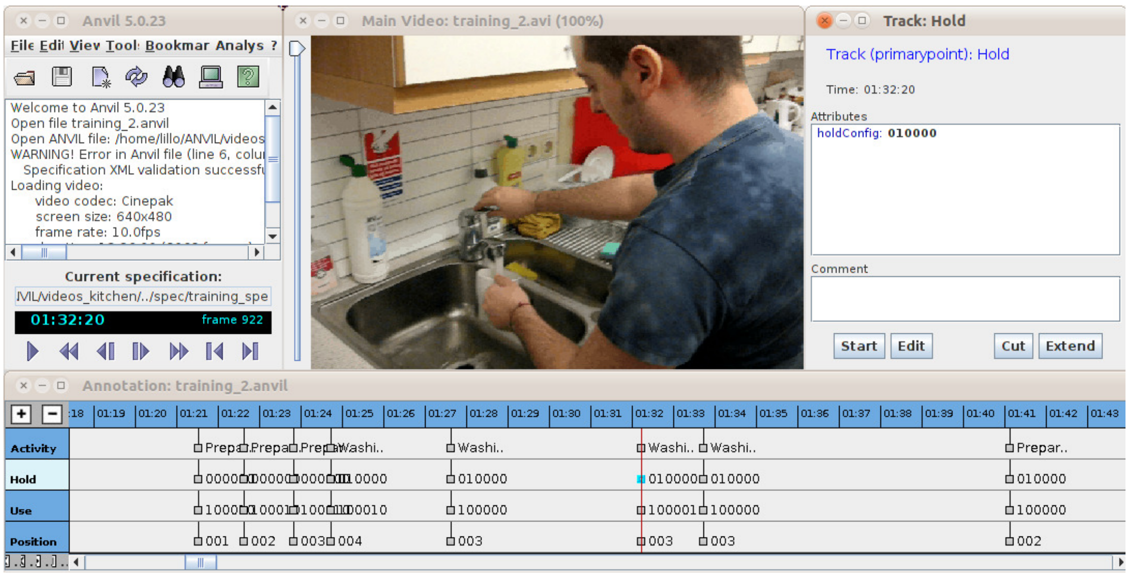
The "Hold" alphabet is composed by symbols of length six (as 6 "special" objects can be held), describing which objects are being held (marked with 1) and which ones are not (marked with 0). As all the configurations are assumed to be possible, the cardinality of this alphabet (taken individually) is $2^6 = 64$ symbols. The "Use" alphabet is also

composed by symbols of length six, describing on-off/open-closed states according to Fig. 5.2. As all the configurations are assumed to be possible, the cardinality of this alphabet (taken individually) is, also in this case, $2^6 = 64$ symbols. The "Position" alphabet is composed by symbols of length three; they could even be of length one though, as we are considering only 4 cells. The cardinality of this alphabet (taken individually) is 4 symbols indeed.
As the reader may imagine, a large number of symbols is necessary when considering multiple channels. The cardinality of the "Hold, Use" alphabet is

$$|V_{Hold,Use}| = |V_{Hold}| * |V_{Use}| = 64 * 64 = 4096,$$

while the cardinality of the "Hold, Use, Position" alphabet is

$$|V_{Hold,Use,Position}| = |V_{Hold}| * |V_{Use}| * |V_{Position}| = 64 * 64 * 4 = 16384.$$


Anvil 5.0.23
File Edit View Tools Bookmark Analysis ?
Main Video: training_2.avi (100%)
Track: Hold
Start
Edit
Cut
Extend
Annotation: training_2.anvil
Activity
Hold
Use
Position

```
Welcome to Anvil 5.0.23
Open file training_2.anvil
Open ANVIL file: /home/lillo/ANVIL/videos
WARNING! Error in Anvil file (line 6, colu
  Specification XML validation successfu
Loading video:
    video codec: Cinepak
    screen size: 640x480
    frame rate: 10.0fps
```

Current specification:
/VIL/videos_kitchen/../spec/training_spe
01:32:20 frame 922

Track (primarypoint): Hold
Time: 01:32:20
Attributes
holdConfig: **010000**
Comment

```
:18 01:19 01:20 01:21 01:22 01:23 01:24 01:25 01:26 01:27 01:28 01:29 01:30 01:31 01:32 01:33 01:34 01:35 01:36 01:37 01:38 01:39 01:40 01:41 01:42 01:43
Activity  Prepa Prepa Prep Washi..  Washi..  Washi.. Washi..  Prepar..
Hold      00000 00000 0000 010000   010000   010000  010000   010000
Use       10000 10001 100  100010   100000   100001  100000   100000
Position  001   002   003  004      003      003     003      002
```

Figure 5.3: Screenshot of ANVIL annotation. Channels, activities and configurations are visible in the lower frame

5.3 Configuration file

For the experiments we used an additional file along with the existing specification, sample and input files. A configuration (.ini) file is used to test the recognition with different parameters and under different modalities. Most importantly it is possible to specify:

1. whether to consider durations, and if so:

- with which modality (either duration channel or duration extension can be used);
- with which time intervals (being each interval a symbol in the "Duration" alphabet);

2. which input file to take for the recognition.

The first point is intuitive and should not need further clarification; we shall briefly explain the second point instead. Having taken 18 videos of Andrea, we have converted all of them into sample files through annotations. Every sample file was, in turn, converted into an input file (through a simple Python program, written by us, which builds new XML files leaving out the specification of activities).
We used a variant of the "K-fold cross validation" technique[*] to test the performance of our models. With this technique, the data set is divided into k subsets; each time, one of the k subsets is used as test set and the remaining $k-1$ subsets are put together to form a set of samples. When initial parameters estimation is completed, the test set can be used to test the performance of the models on "new" data. In our case $k = 18$, the test set is an input file, and the set of samples are the sample files which do not correspond to the input file. Having 18 sample files and as many input files, every time we take one of the input files as input for the recognition, and we leave the corresponding sample file out of the initial parameters' estimations process: we use the 17 sample files which have nothing to do with the specified input file. This process is repeated 17 times, this being the number of possible combinations between the test subset (of length 1) and the sample subsets (of length 17).

5.4 Results

For the results of our experiments we consider average recognition/prediction rates, calculated on the basis of single-file rates. Every time a single file is used as input file (both for recognition and prediction), testing results are stored into an external file and used to calculate average values. The graphs we show in this section were generated automatically: the ReportLab Toolkit open-source library[†] was used to programatically create charts in PDF format.

[*] Website: http://www.cs.cmu.edu/schneide/tut5/node42.html. Accessed May 2012
[†] Website: http://www.reportlab.com/software/opensource/rl-toolkit. Accessed May 2012

5.4.1 Prediction

For each input file, as we said, we count the occurrences of both “good” and “bad” prediction and we extract percentages out of it. Fig. 5.4 shows the average rates for the prediction module. Note that these values are valid for every settable parameter/modality in the configuration file: the prediction process does not deal with emission probabilities (which change a lot depending on the considered channels, parameters and modalities), but only with transition probabilities (which are always the same for all models).

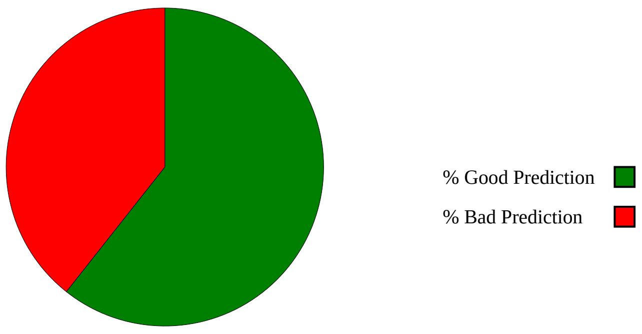


Figure 5.4: Prediction meanvalues

We obtained a 60.7% of “good prediction” and a 39.3% of “bad prediction”. In this work we did not focus much on the prediction; however with this preliminary result we showed that the observation of a subject’s behaviour patterns, described by transition probabilities in whichever HMM we consider, could be a good means for the prediction of his/her future activities. Focused work should be done in this sense, by using observational channels for joint application of recognition and prediction.

5.4.2 Recognition

As for the operation we focused on, we said that two testing methods were implemented:

- single observations recognition testing, and
- sequence similarity testing.

In these experiments we consider both methods, showing the relative percentages for each model (i.e., for each channel and combination of channels). A *tolerance* of 2 observations was used for single observations recognition testing.
When no durations were considered (Tab. 5.1 and Fig. 5.5) we obtained the highest rates

		min	max	average
Hold	Observations Recognition	47.05%	91.42%	66.97%
	Sequence Similarity	11.11%	66.67%	32.79%
Use	Observations Recognition	39.82%	97.14%	74.71%
	Sequence Similarity	14.29%	100.00%	44.42%
Position	Observations Recognition	36.37%	100.00%	73.05%
	Sequence Similarity	20.00%	100.00%	46.37%
Hold_Use	Observations Recognition	32.40%	92.15%	71.02%
	Sequence Similarity	11.11%	75.00%	33.48%
Hold_Position	Observations Recognition	46.42%	100.00%	78.35%
	Sequence Similarity	12.50%	100.00%	42.92%
Use_Position	Observations Recognition	48.57%	100.00%	80.67%
	Sequence Similarity	20.00%	100.00%	54.90%
Hold_Use_Position	Observations Recognition	59.09%	100.00%	78.76%
	Sequence Similarity	11.11%	100.00%	42.58%

Table 5.1: Resulting percentages when no durations are considered. Minimum/maximum (relative to one particular observation sequence) and average (relative to all observation sequences) percentages are presented.

for the "Use_Position" channel (80.7% for single observations recognition, and 54.9% for sequence similarity); we obtained roughly the same rates for "Hold_Position" and "Hold_Use_Position" (78.3% and 78.7 for single observations recognition, 42.9% and 42.6 for sequence similarity). "Hold" (66.9%, 32.8%) and "Hold_Use" (71.0%, 33.5%) are the worst in recognizing.

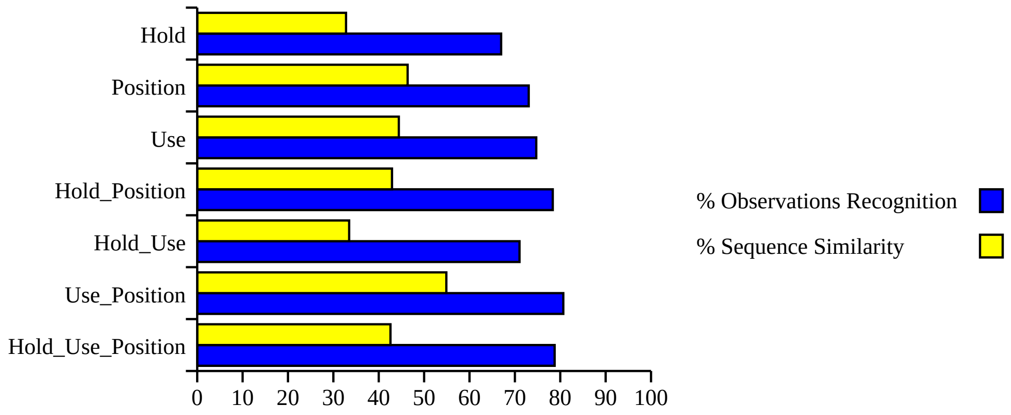


Figure 5.5: Average results when no durations are considered.

When duration extension was considered (Tab. 5.2 and Fig. 5.6), as when no durations were considered, we obtained the highest rates for "Use_Position" (84.9% for single observations recognition, and 54.5% for sequence similarity). The "Hold_Position" channel is also good for single observations recognition (78.9%); however it is not as good for

		min	max	average
Hold	Observations Recognition	40.91%	91.42%	67.60%
	Sequence Similarity	11.11%	57.14%	29.83%
Use	Observations Recognition	40.91%	89.29%	77.40%
	Sequence Similarity	20.00%	71.43%	48.14%
Position	Observations Recognition	31.82%	100.00%	73.75%
	Sequence Similarity	20.00%	100.00%	48.58%
Hold_Use	Observations Recognition	32.40%	96.07%	70.52%
	Sequence Similarity	11.11%	66.67%	29.76%
Hold_Position	Observations Recognition	46.42%	100.00%	78.92%
	Sequence Similarity	12.50%	100.00%	43.82%
Use_Position	Observations Recognition	68.16%	100.00%	84.91%
	Sequence Similarity	20.00%	100.00%	54.52%
Hold_Use_Position	Observations Recognition	59.09%	100.00%	75.25%
	Sequence Similarity	11.11%	100.00%	34.64%

Table 5.2: Resulting percentages when durations extension is considered. Minimum/-maximum (relative to one particular observation sequence) and average (relative to all observation sequences) percentages are presented.

sequence similarity (43.8%). Again, "Hold" (67.6%, 29.8%) and "Hold_Use" (70.5%, 29.8%) are the worst in recognizing.

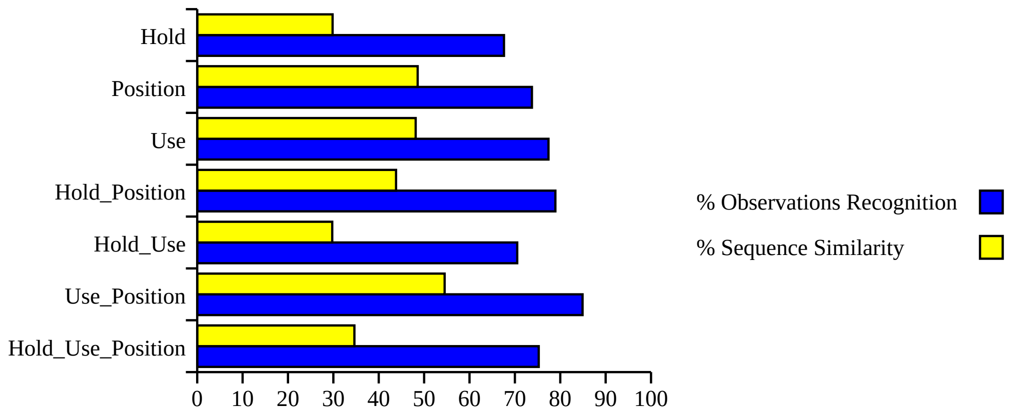


Figure 5.6: Average results when durations extension is considered.

When adding duration channel information (Tab. 5.3 and Fig. 5.7) we logically obtained the same rates we got when no durations were used, for the channels not considering "Duration". When information about duration is used, we found that "Use_Position_Duration" is slightly better than "Use_Position" for single observations recognition (81.4% and 80.7% respectively), but worse in sequence similarity (52.4% and 54.9% respectively). For the "Hold" channel we got higher single observations recognition when duration in-

		min	max	average
Duration	Observations Recognition	39.82%	91.42%	63.72%
	Sequence Similarity	11.11%	50.00%	19.97%
Hold_Duration	Observations Recognition	46.42%	97.83%	69.65%
	Sequence Similarity	11.11%	66.67%	33.39%
Use_Duration	Observations Recognition	39.82%	97.14%	74.30%
	Sequence Similarity	11.11%	100.00%	43.81%
Position_Duration	Observations Recognition	31.82%	100.00%	69.54%
	Sequence Similarity	20.00%	100.00%	46.23%
Hold_Use_Duration	Observations Recognition	32.40%	97.82%	70.91%
	Sequence Similarity	11.11%	62.50%	32.07%
Hold_Position_Duration	Observations Recognition	46.42%	100.00%	78.46%
	Sequence Similarity	11.11%	100.00%	42.31%
Use_Position_Duration	Observations Recognition	52.78%	100.00%	81.38%
	Sequence Similarity	20.00%	100.00%	52.48%
Hold_Use_Position_Duration	Observations Recognition	59.09%	100.00%	77.73%
	Sequence Similarity	11.11%	100.00%	38.73%

Table 5.3: Resulting percentages when durations channel is considered. Minimum/maximum (relative to one particular observation sequence) and average (relative to all observation sequences) percentages are presented.

formation is used (69.7%), but for the “Position_Duration” channel we got lower single observations recognition (69.5%) with respect to “Position” (73.1%).

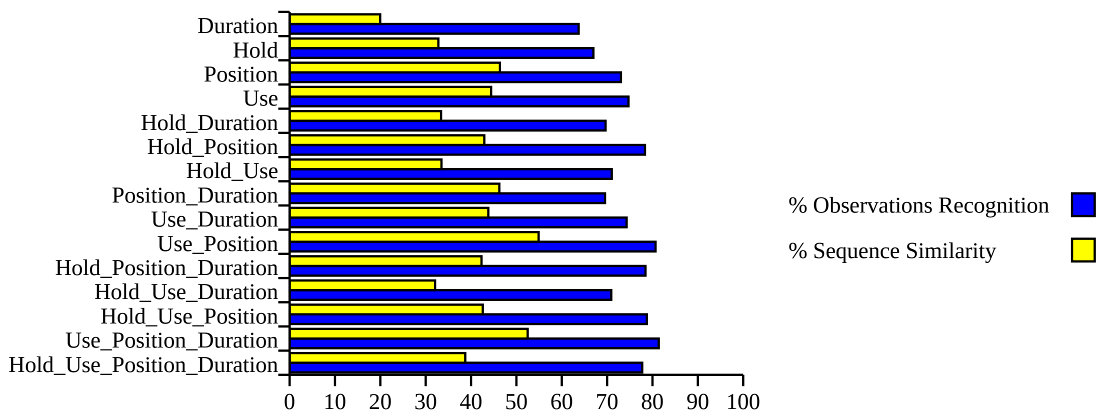


Figure 5.7: Average results when durations channel is considered.

Discussion

Results have shown that generally, for these experiments, the “Use” channel (taken individually) contributes the most to the recognition, followed by “Position”, “Hold” and “Duration”; as for the sequence similarity, the best is “Position” followed by “Use”, "Hold" and "Duration". We were actually expecting the "Hold" channel to be very important for the recognition; while this ended up not being true when the channel is taken individually, it often contributes when in conjunction with the "Position" channel. Joint information about the "Hold" and "Use" channels does not tend to constitute good information for the recognition.

The 3-channels models are generally not better than 2-channels models. The use of more than 2 channels is in general not recommendable, as a decrease in recognition rates was observed in most of the cases. However, this is probably due to the lack of enough sample data. The more channels are considered, and the more symbols are in each alphabet, the more the lack of sample data would affect the recognition. The "Use_Position" channel generally showed the best rates among all other channels; Fig. 5.8 summarizes how rate changes for this channel.

We remind the reader that the results we obtained are not absolute, as they refer to these very experiments we carried out; however we believe that similar results could be obtained when repeating the experiment with different parameters (e.g., different alphabets for the channels, unless activity-specific objects are tracked, such as a cereal box for the hypothetical activity "PreparingBreakfast").

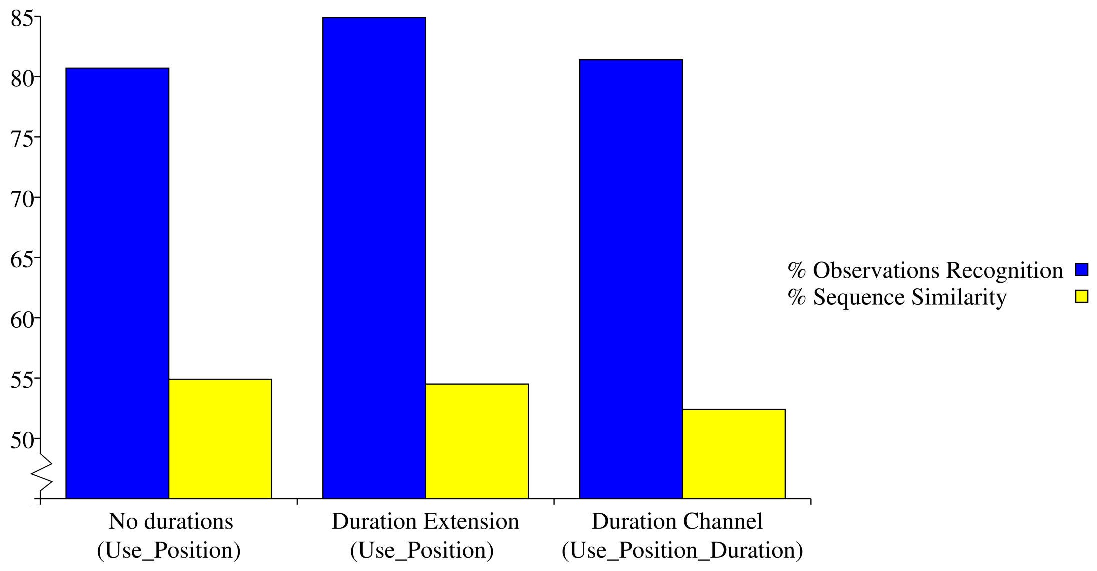


Figure 5.8: Summary of "Use_Position" channel obtained rates (single observation recognition and sequence similarity).

Chapter 6

Conclusions

We presented an approach, based on what we call "observational channels", for studying how different combinations of easily trackable features (extracted from a home environment) contribute to the recognition of human activities in a real-world scenario. We refer to features as parameters describing, under different points of view, actions of the subject (e.g., holding objects) or state changes of components in the environment (e.g., appliances in use). Features are easily trackable, as there should exist devices (or techniques) for data retrieval which do not make use of further recognition mechanisms. The recognition is direct, as there is no need to consider activity sub-sequences for the recognition of high-level daily activities (e.g., cooking, bathing, etc.); indeed data coming from extracted features constitute the only information about the world to be aware of. A complete activity is not modelled as a sequence of sub-activities to be performed in a certain order; our models can recognize (to some extent) what activity the subject is performing by only looking at the observational channels they consider. Our approach uses the Hidden Markov Models framework, as HMMs (and their extensions) constitute the most widely used framework for many kinds of recognition; recently, they have successfully been used in many applications of activity recognition in indoor environments as well. As we want our approach to work in a real-work scenario, many aspects should be taken under careful consideration for a good initial estimation of HMMs parameters. For each feature (and combinations of them), we should know how people tend to behave when performing certain (general) tasks such as cooking and relaxing; as a social study would have been rather time-consuming (even considering a small sample of people), we limited our study to a single subject and to a typical domain from daily life. We assumed that subject's behaviour inside this domain may be a good representative for similar domains; yet, transfer of results to different domains still needs to be verified.

6.1 Final considerations

Our approach for the recognition is based on single-user's behaviour; this is mainly to contrast our lack of extensive knowledge about generalized behaviour when performing activities, specially when considering multiple features together. Furthermore, as we consider a real-world scenario, each activity can be performed in a high number of possible ways; activities cannot be inferred very easily, as features can have a lot of possible combinations. The recognition is made possible only by considering common behaviour patterns.
The fact that our approach is user-specific does not represent a serious problem. An additional system, yet to be implemented, could be used for the supervised training of the models, thanks to user's labelling suggestions. The training period could be for example one week, the user performing activities twice a day.
We proved our perhaps simplistic approach to be functional, even though the recognition rates for entire input sequences are not close to 100%. The reasons why this happens are mainly the following:

- Sometimes it is good to consider multiple factors (ideally it is helpful to know that the user is holding a cutting board while the toaster is on, to understand that he/she is probably preparing food), but in some cases it would be rather confusing (many appliances turned on and a fork in the user's hand could not easily lead to the recognition of activity "Relaxing"; in two episodes during the experiments the subject behaved exactly like that, and none of the models was able to make a correct recognition). Considering a single channel is not good though; even if experiments proved that the single feature "MovedObject" can be extremely meaningful for the recognition [BPPW09], we believe that in a real-world scenario it is not feasible to put tags in every kind of object (e.g., RFID tags on the cereal box for recognition of "Having Breakfast", or RFID tags on a book for "Reading Book"); objects to take information from should be the ones we always use (e.g., our favourite mug/mugs, our cutlery, our mobile phone).

- The lack of training data; by considering multiple channels it is normal to miss some possible (multiple) emissions when data is insufficient. Of course much more sample data, even before the training, would help rising the rates; alternatively, a different approach for a better estimation of parameters should be used.

- Our approach, in theory, should be feasible for activity recognition even when activities are not completed all at once. In our experiments the subject often performed other activities while waiting for his lunch to be ready; with certain combinations of channels it was possible to recognize sequences like "PreparingFood $\longrightarrow$ Relaxing $\longrightarrow$ PreparingFood" with no problems. Other times, instead, the fact that a user could stop doing an activity and continue it at a later time drastically decreased the recognition rates; by adding the constraint of not performing multiple activities all at once, we surely could have achieved better results.

- Our approach does not take into account starting and ending times of activities. A mechanism for recognizing activity boundaries would be of great help for the recognition, as only one activity is performed between starting and ending time; different topologies of HMMs could have been used in this case.

We believe that a good use of different observational channels can constitute a significant help for indoor activity recognition; our results show that generally a joint usage of channels (e.g., "Use" and "Position" with our configuration/parameters) works better than taking the same channels singularly. Further investigation about which activities are difficult to recognize with these channels are necessary. In this work we just focused on activity sequences recognition, but other aspects are yet to be explored; other experiments can be done with our prototype, perhaps trying with different domains, channels, alphabets and activities.
Observational channels need to be "fed" by means of a mechanism dealing with sensors. This is not to be considered a limitation of our approach, as a simple semantic network (or ontology, or even a database) can be used for it. Our approach relies on a noise-free extraction of data: the devices used to get any kind of data (e.g., RFID readers and accelerometers for the "Hold" channel) should be as trustworthy as possible, minimizing the ever possible noise in readings.
A major limitation of our approach is the high complexity of the models when considering too many channels (in particular when alphabets for each channel are large). Fortunately, sometimes it would be possible to consider a subset of all possible configurations (emission symbols). The simplest example is for the "Hold" channel: in our experiment we considered the user to be able to hold even all objects at a time (i.e., the configuration "111111" was considered to be possible even if, of course, it could have never happened). Some configurations can be safely left out, so that the alphabet for a channel (and consequently for all multiple channels considering this channel) can be made smaller.
Another major limitation is the fact that, with our approach, it is not possible to count how many times a certain activity was recognized; we can only look at the real and guessed sequences of activities. We tried to build a recognition system without relying on mechanisms being able to tell the starting and ending times of activities; some changes in the approach would probably be necessary, because the recognition is penalised if no help of any kind is provided.

6.2 Future works

The following points represent future work to be done.

1. Further experiments should be done with different domains and activities, to confirm our findings about how the considered channels contribute to the recognition.
2. New channels (different from “Hold”, “Use” and “Position”) should be tried with our prototype, to discover new “good sources” of information for human activity recognition.
3. A better method for considering durations should be developed.
4. Check whether different alphabets for the considered channels could lead to higher rates in the recognition of the same activities.
5. A conjoint use of prediction and recognition (so that the input for the prediction would be the recognized activity) should be implemented.
6. Try the system in the real-world, interfacing real sensors with the system to get input automatically, and developing an additional training system so that the user could label his/her own activities.
7. It could be interesting to use our approach with different groups of people (e.g., male and female individuals), for studies on which features are the most effective (in average) for the recognition of the same tasks performed by the different groups.
8. Starting and ending times of activities should be defined (e.g., probabilistically) by an additional component, to enhance the recognition; slight changes in the approach should therefore be taken into consideration.

Bibliography

[AR11] J.K. Aggarwal and M.S. Ryoo. Human activity analysis: A review. *ACM Comput. Surv.*, 2011.

[BGB07] B. Bouchard, S. Giroux, and A. Bouzouane. A keyhole plan recognition model for alzheimer's patients: First results. *Applied Artificial Intelligence*, 2007.

[BPPW09] M. Buettner, R. Prasad, M. Philipose, and D. Wetherall. Recognizing daily activities with rfid-based sensors. In *Proceedings of the 11th international conference on Ubiquitous computing*, Ubicomp '09, New York, NY, USA, 2009. ACM.

[BSH$^+$10] J. Biswas, K. Sim, W. Huang, A. Tolstikov, A. Aung, M. Jayachandran, V. Foo, and P. Yap. Sensor based micro context for mild dementia assistance. In *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '10, pages 20:1–20:4, New York, NY, USA, 2010. ACM.

[CCC10] D. Cacciagrano, F. Corradini, and R. Culmone. Resourcehome: An rfid-based architecture and a flexible model for ambient intelligence. In *Proceedings of the 2010 Fifth International Conference on Systems*, ICONS '10. IEEE Computer Society, 2010.

[DEKM98] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. Biological sequence analysis: probabilistic models of proteins and nucleic acids, 1998.

[Kip01] M. Kipp. Anvil - a generic annotation tool for multimodal dialogue, 2001.

[LCB06] J. Lester, T. Choudhury, and G. Borriello. A practical approach to recognizing physical activities. In *In Proc. of Pervasive*, 2006.

[NN07] P. Natarajan and R. Nevatia. Coupled hidden semi markov models for activity recognition. In *Proceedings of the IEEE Workshop on Motion and Video*

Computing, WMVC '07, Washington, DC, USA, 2007. IEEE Computer Society.

[PFB$^+$09] C. Phua, V. S. F. Foo, J. Biswas, A. Tolstikov, A. P. W. Aung, J. Maniyeri, W. Huang, M. H. That, D. Xu, and A. K. W. Chu. 2-layer erroneous-plan recognition for dementia patients in smart homes. In *Proceedings of the 11th international conference on e-Health networking, applications and services*, Healthcom'09, Piscataway, NJ, USA, 2009. IEEE Press.

[PRA$^+$10] C. Phua, P. C. Roy, H. Aloulou, J. Biswas, A. Tolstikov, S. Foo, V. Fook, W. Huang, M.A. Feki, J. Maniyeri, and A. K. W. Chu. State-of-the-art assistive technology for people with dementia. IGI Global, IGI Global, 2010.

[Rab90] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[RJ86] L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, Jan 1986.

[RJS83] L. R. Rabiner, B. H. Juang, and M. M Sondhi. On the application of vector quantization and hidden markov models to speaker-independent isolated word recognition. *Bell Syst. Tech. J., vol. 62*, 1983.

[RN10] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2010.

[Sor12] D. Sorrentino. Integrating semantic networks and object-oriented model to represent and manage context. 2012.

[SRSG04] A. Schliep, W. Rungsarityotin, A. Schonhuth, and B. Georgi. The general hidden markov model library: Analyzing systems with unobservable states, 2004.

[War96] N. Warakagoda. Hidden markov models. *Budapest University of Technology and Economics Physics Department Student Page*, 1996.

[WOC$^+$07] J. Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J. M. Rehg. A scalable approach to activity recognition based on object use. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2007.

School of Computer Science
Reykjavík University
Menntavegi 1
101 Reykjavík, Iceland
Tel. +354 599 6200
Fax +354 599 6201
www.reykjavikuniversity.is
ISSN 1670-8539