Learning to Understand Multimodal Commands and Feedback for Human-Robot Interaction

Anja Austermann

DOCTOR OF PHILOSOPHY

Department of Informatics,

School of Multidisciplinary Sciences,

The Graduate University for Advanced Studies (SOKENDAI)

2010 (School Year)

September 2010

A dissertation submitted to
the Department of Informatics,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies (SOKENDAI)
In partial fulfillment of the requirements for
the degree of Doctor of Philosophy

PhD Committee:

Seiji Yamada	National Institute of Informatics, SOKENDAI Tokyo Institute of Technology
Helmut Prendinger	National Institute of Informatics, SOKENDAI
Ken Satoh	National Institute of Informatics, SOKENDAI
Tetsunari Inamura	National Institute of Informatics, SOKENDAI
Ryutaro Ichise	National Institute of Informatics, SOKENDAI

Abstract

Understanding a user's natural interaction is a challenge that needs to be addressed in order to enable novice users to use robots smoothly and intuitively. While using a set of hard-coded commands to control a robot is usually rather reliable and easy to implement, it is troublesome for the user, because it requires him/her to learn and remember special commands in order to interact with the robot and does not allow the user to use his or her natural interaction style. Understanding natural, unrestricted spoken language and multi-modal user behavior would be desirable but is still an unsolved problem.

Therefore, this dissertation proposes a domain-specific approach to enable a robot to learn to understand its user's natural way of giving commands and feedback through natural interaction in special virtual training tasks. The user teaches the robot to understand his/her individual way of expressing approval, disapproval and a limited number of commands using speech, prosody and touch.

In order to enable the robot to pro-actively explore how the user gives commands and provoke approving and disapproving reactions, the system uses special training tasks. During the training, the robot cannot actually understand its user. In order to enable the robot to react appropriately anyway, the training tasks are designed in such a way that the robot can anticipate the user's commands and feedback – e.g. by using games which allow the user to judge easily whether a move of the robot was good or bad and give appropriate feedback, so that the robot can accurately guess whether to expect positive or negative feedback and even provoke the feedback it wants to learn by deliberately making good or bad moves.

In this work, "virtual" training tasks are used to avoid time-consuming walking motion and to enable the robot to access all properties of the task instantly. The task-scene is shown on a screen and the robot visualizes its actions by motion, sounds and its LEDs. A first experiment for learning positive and negative feedback, uses easy games, like "Connect Four" and "Pairs" in which the robot could explore the user's feedback behavior by making good or bad moves. In a follow-up study, which was conducted with a child-sized humanoid robot as well as pet-robot AIBO, this work has been extended for learning simple commands. The experiments used a "virtual living room", a simplified living room scene, in which the user can ask the robot to fulfill tasks such as switching on the TV or serving a coffee.

After learning the names of the different objects in the room by pointing at them and asking the user to name them, the robot requests from the task server to show a situation that requires a certain action to be performed by the robot: E.g. the light is switched off so

that the room is too dark. The user responds to this situation by giving the appropriate command to the robot: "Hey robot, can you switch the light on?" or "It's too dark here!". By correct/incorrect performance, the robot can provoke positive/negative feedback from the user. One of the benefits of "virtual" training tasks is that the robot can learn commands, that the user cannot teach by demonstration, but which seem to be necessary for a service or entertainment robot, like showing the battery status, recharging, shutting down, etc.
The robot learns by a two-staged algorithm based on Hidden Markov Models and classical conditioning, which is inspired by associative learning in humans and animals. In the first stage, which corresponds to the stimulus encoding in natural learning, unsupervised training of HMMs is used to model the incoming speech and prosody stimuli. Touch stimuli are represented using a simple duration-based model. Unsupervised training of HMMs allows the system to cluster similar perceptions without depending on explicit transcriptions of what the user has said or done, which are not available when learning through natural interaction.
Utterances and meanings can usually not be mapped one-to-one, because the same meaning can be expressed by multiple utterances, and utterances can have different meanings. This is handled by the associative learning stage. It associates the trained HMMs with meanings and integrates perceptions from different modalities, using an implementation of classical conditioning. The meanings are inferred from the robot's situation. E.g. If the robot just requested the task server to show a dirty spot on the carpet, the robot assumes, the following utterance means clean(carpet), so the system first searches for a match of any of the HMMs, associated with the meaning "carpet". Then, the remainder of the utterance is used to train a HMM sequence to be associated with the meaning "to clean". The positions of the detected parameters are used to insert appropriate placeholders in the recognition grammar.
In a first study, based on game-like tasks, the robot learned to discriminate between positive and negative feedback based on speech, prosody and touch with an average accuracy of 95.97%. The performance in the more complex command learning task is 84.45% for distinguishing eight commands with 16 possible parameters.

Acknowledgments

I thank my advisor, Seiji Yamada, for his valuable guidance, advice and feedback and the fruitful discussions on my PhD research. Thank you for always being accessible and for giving me the freedom to develop my own ideas. I could not have imagined a better advisor for my doctoral research.

I am honored to have Helmut Prendinger, Ken Satoh, Tetsunari Inamura, and Ryutaro Ichise as committee members for my thesis. Thank you very much for taking the time to study my work and examine my thesis.

I am very thankful for the great opportunity to conduct experiments at Honda Research Institute. Many thanks to Hiroshi Tsujino, Mikio Nakano and Kotaro Funakoshi, not only for making this possible but also for giving me a great deal of support for my study and very valuable discussions and feedback on my work.

I would like to thank the National Institute of Informatics for the NII scholarship and for providing a great environment for study and research.

I would also like to thank Yoshiki Chubachi and Takao Izumi and my current and former colleagues at Elektrobit Nippon. I am really grateful for their support, understanding and flexibility, especially during the last phase of writing my thesis.

Last but definitely not least, thanks to my family and friends, especially to my parents and my brother in Germany. Without their constant support, advice and guidance, this thesis would not have been possible. Thank you for always encouraging me to follow my dreams and for letting me go to Japan although it was not easy for you. Thank you for always being there for me.

... Finally, I'd like to say a big thank you to my "robotic collaborators". I know, I tortured you a lot, made you work on weekends and late evenings and play the same games over and over again – but you were as patient, as only robots can be, and never complained.

Contents

List of Tables

List of Figures

"The true delight is in the finding out rather than in the knowing."
Isaac Asimov (1920 - 1992)

1 Introduction

1.1 Overview

In recent years, robot engineering has made significant steps forward toward creating robots that can coexist in a household together with humans and assist them in performing their household chores, help caring for the elderly, entertain and serve as a companion or provide an universal interface to electronic devices in a smart home. First entertainment robots like AIBO [2] or Pleo [62] as well as simple household robots like the robotic vacuum cleaner Roomba [38] or the robotic mop Scooba [39] have already found their ways into millions of households worldwide. Some examples of current robots, used in research, homes and industry are shown in Fig. 1.1.

Typical fields in which robots will be applied in the not-too-distant future, such as health care and care for the elderly, will require robots to be able to interact with naive users in a natural and socially acceptable way. An important factor that needs to be considered to make robots intuitively usable for non-experts, is their ability to understand natural instruction and feedback. Therefore roboticists are striving to create "manual-free robots", that is, robots which can be operated intuitively by everyone without having to study and remember the contents of a handbook.

This thesis concerns enabling a robot to learn to understand its user's natural, multimodal interaction, in order to make a step forward toward the implementation of a "manual-free robot".

The main goal of this work was to implement a method by which a robot can learn parameterized commands, object names and feedback through situated interaction with a user without requiring transcriptions of the user's utterances or remote-controlling of the robot. The proposed approach comprises a learning algorithm and special training tasks, which can be used for situated learning with an actual robot. The two-staged learning algorithm is based on a combination of unsupervised clustering of utterances as well as prosody and touch stimuli using Hidden Markov Models and supervised learning of associations between the clustered stimuli and symbolic representations of their meanings

Figure 1.1: Different types of robots used in research, homes and industry

using classical conditioning. Knowledge about the meanings, that the trained Hidden Markov Models are associated with, is obtained by using special training tasks, that allow the robot to accurately guess the meaning of the user's next command or feedback and even actively provoke behavior, such as utterances with a certain meaning from the user. Using the training tasks, the robot can actively select commands, feedback and object names for training and also repeat the training of selected commands or feedbacks for which it has not gathered sufficient training data yet.

A series of experiments has been conducted in order to look into how users give commands and feedback to robots. The aim of the user studies was to answer three questions which are the basis for this research:

- *Is there a benefit in having a robot learn multimodal feedback from its user?* This is only true, if different people give reward in different ways which are difficult to handle by using hard-coded patterns for recognizing commands and feedback.
- *Is it possible to learn multimodal user feedback in a training phase in a reasonable amount of time?* Only if commands and reward behavior, used by a single person,

do not vary excessively and are similar between different tasks, they can be learned effectively in a training phase

- *Which features of the training tasks are important for learning natural user feedback?*

Based on the experiments, the consistency of feedback, the use of multimodality as well as factors that influence the amount of feedback, given by a user, were analyzed and commands and feedback given to a humanoid and a pet-robot were compared. Based on the findings from the user studies, this thesis evaluates and discusses the feasibility and usefulness of learning commands and feedback from a user through the proposed training tasks and gives details on the implementation of the learning algorithm to enable a robot to learn to understand its user's feedback and commands.

1.2 Motivation

Current service robots, such as AIBO [2] or Roomba [38] can only react to hard-coded commands, which the user has to learn from a handbook. This means, that every user who wants to interact with or use the robot, first needs to memorize, which commands to utter or which buttons to press to make the robot perform the desired actions. In case of Roomba, which cannot be controlled by speech but is operated by pressing buttons either on a remote control or on the robot itself, controlling the robot actually requires close proximity to either the robot or its remote control. That means, that in many cases the user would have to stand up and physically manipulate the robot or fetch the remote control in order to give commands to the robot.

While learning commands from a handbook or walking to and physically interacting with a robot may be a mere inconvenience for average, adult, non-handicapped users with normal memory capabilities, this way of using robots may become increasingly difficult for elderly, memory-impaired or handicapped persons. Moreover, having to read and understand a complicated handbook or interact using non-natural modalities typically increases the inhibition level against using an electronic device in non-tech-savvy individuals.

Having a robot learn commands from its user instead of forcing the user to learn commands to control the robot, has different advantages: From the system designer's point of view, hard-coding multimodal input and speech dialogues in a way, that they allow commands and feedback to be given in a user's natural interaction style, is a difficult task, because it requires foreseeing how users will try to interact with the robot. The system designer not only needs to deal with personal, as well as language- or culture-dependent differences in user behavior but also with differences in interaction, depending on the type, the shape and the size of the robot. Even a carefully designed multimodal dialog is unlikely to work well for every user without reading a handbook and memorizing commands. Therefore, learning through situated training tasks, how a certain user actually interacts

with a robot and automatically adapting to the user's preferences instead of requiring pre-modeled dialogs etc. facilitates the conceptual work of the system designer.

One particularly difficult task in processing naturally spoken commands is handling implicit commands, such as "It is too dark here." or "I'd like to drink a coffee." which require world knowledge as well as reasoning capabilities to interpret them correctly as "Switch the light on!" or "Bring me a coffee!". By learning these commands from the user in a training tasks and mapping them to their meanings without analyzing the grammatical structure word-by word, the algorithm, proposed in this thesis can handle implicit commands in exactly the same way as explicit commands.

As the system learns speech commands directly from its user and trains user-dependent Hidden Markov Models, it is also robust against common causes of speech recognition problems, such as regional accents or unusual voices.

Having the robot learn to understand commands and feedback through a training task also has different benefits from the user's perspective: When a robot is able to learn its user's way of giving commands and feedback, the user can give commands and feedback to the robot naturally and intuitively in the same way as he or she would interact with a human or a pet. As the proposed approach, which adapts a robot to its user's natural way of giving commands and feedback, shifts the learning effort from the user to the robot, it would be especially desirable for elderly people with memory deficits.

One could argue, that the easiest way to control a robot and to avoid the difficulties of speech-based interaction, is by using a remote control with buttons and easily understandable icons, so that no time is needed for the robot to learn and no processing of natural speech data is necessary. However, using a remote control may become complicated if the number of commands increases or if commands have parameters etc. Moreover, a remote control needs to be carried around or fetched every time, the user wants to control the robot, and can easily be misplaced. Icons may be difficult to see for people with reduced eyesight, when there is a large number of buttons or the user is operating the system in the dark. As opposed to this, natural human modalities like speech are readily usable at any time without any preparation.

A downside of user adaptation based on the proposed technique is that it takes some time and effort until the robot can be used productively. However, as household robots are expected to interact with only a small number of individuals over a long time of several years, adaptation to specific users is still desirable.

Moreover, the virtual training tasks have been designed in a way, that they can not only be used for adapting the robot to its user but also to allow the user to familiarize with the robot within a controlled scenario where incorrect operation does not have any negative consequences.

The main design decisions for the training tasks and the learning algorithm and their motivation are summarized in the following paragraphs:

1.2.1 Learning user feedback

The starting point of this work was enabling a robot to learn one aspect of natural human teaching behavior: approving or disapproving feedback given in response to the robot's moves. Understanding approval and disapproval is important and a highly reliable recognition of these two kinds of feedback is desirable, because humans use approving and disapproving feedback in almost any kind of teaching situation. When teaching a robot by natural interaction, one important task for the human teacher is to evaluate the robot's performance and give positive and negative reward. This is especially true in case of reinforcement learning with a human teacher [79] [80], where positive and negative reward is the only means for the user to guide the behavior of the robot, but feedback is also used frequently in general teaching tasks.

Overall, positive and negative feedback are the smallest useful set of commands, that can be used to teach a robot, for example by reinforcement learning. Therefore, learning to understand positive and negative feedback was chosen as a first step toward learning more complex utterances.

1.2.2 Progressing from feedback to commands

While positive and negative feedback can be used for simple teaching situations, a robot, that can perform useful work in a household, needs to be able to understand more detailed instructions. Therefore the long-term-goal of this research was to develop a method for not only learning to understand positive and negative rewards but also parameterized commands, that can be used for controlling the service and entertainment functions of service-robots and pet-robots through the interaction with a user. The original approach for learning user feedback was extended to handle such commands.

Extensions were made to the training tasks as well as the learning algorithm for feedback learning to enable the system to learn parameterized commands, such as "Please put *the ball* into *the box*!" or "Please clean *the table*." through interaction with a user.

1.2.3 Processing multimodal user input

Humans express commands, as well as approval and disapproval toward a robot through different channels, such as spoken words, prosody, gestures, facial expressions and touch. Most work on understanding approval and disapproval has been done with single-modal approaches based on prosodic information from speech signals such as intonation, pitch, tempo, loudness and rhythm [13][55]. However, it can be assumed that integrating additional modalities improves the reliability of the recognition and allows the system to adapt to the preferences of the users. Therefore, the proposed approach integrates speech, prosody and touch for a reliable recognition of feedback and commands.

1.2.4 Using a biologically-inspired learning approach

The learning algorithm is inspired by the processes which are present in biological learning and speech understanding. Human speech acquisition has several features, which are very desirable for speech learning in a robot. Most important, humans can learn speech without transcriptions of the utterances and learn to segment utterances into relevant parts.

However, the goal of the thesis was not implementing a computational model of human speech acquisition but developing a method that can be integrated efficiently into a robot to learn natural human commands and feedback. While the proposed learning method does not try to model all processes accurately, that occur in nature, it uses the findings from biology and psychology to determine mechanisms that can be implemented efficiently and are suitable for the given task.

Some of the mechanisms, that have been designed after processes found in nature are the interplay of bottom-up and top-down processing, the separation of the learning process into a stimulus encoding phase and an associative learning phase, the use of a mathematical model of classical conditioning, which was created in psychological research to model human associative learning etc. Details are given in the literature review in chapter 2 as well as in the description of the learning algorithm in chapters 4 and 5.

1.2.5 Using "virtual" training tasks for user adaptation

The learning algorithm is, in principle, independent from the tasks used for training the robot. It only needs input data pairs consisting of meanings, such as "positive feedback"' or "MOVE(BALL, BOX)" and perceptions of the user's behavior, such as audio data of utterances or data from the touch sensors.

In this thesis, these data pairs are provided by virtual training tasks, which the robot performs in front of a screen and which allow it to accurately guess the meaning of the user's behavior and persuade the user to utter certain commands or feedback. They are described in detail in chapter 3.

The main motivation for using virtual training tasks was, that they are independent from the physical capabilities of the robot, so that the same tasks can be used for different robots, including very small ones, like AIBO, which have very limited physical capabilities. Moreover, they do not require time-consuming walking and other complex motions of the robot and permit focusing on the implementation of the interaction learning without having to handle perception of the environment and task execution.

In a real application with a service robot, the virtual training tasks would be performed by the user and the robot in front of the TV or a PC before actually using the robot. All commands and feedback, that the robot needs to understand, would first be trained using

the virtual tasks. After finishing the training, the learned commands and feedback could be used to control the robot in real-world tasks.

1.3 Outline of the training method

The proposed approach for training a robot through interaction with its user consists of special, "virtual" training tasks and a two-staged learning algorithm. This section intends to give a very brief overview of the training phase as well as the learning algorithm. Details are explained in chapters 3, 4 and 5.

1.3.1 The training tasks

In the studies, different, "virtual" training tasks were used, in which the robot interacted with the user in front of a screen. Scenes from the experiments are shown in Figure 1.2. Game-like tasks were used for enabling the robot to learn feedback from its user and a simplified living room scene was employed for learning object names and commands. During the experiments for learning feedback and commands, the actions, shown on the screen, and the motions of the robot were synchronized to give the user the impression that the robot actually manipulates objects on the screen.

Figure 1.2: Scenes from the experiments on feedback and command learning

The training tasks were designed to enable the robot to automatically record and learn users' feedback. The aim of the tasks is allowing the robot to provoke and learn natural feedback from the user without any prior understanding of the user's utterances. Therefore, they need to be designed in a way that they provide enough background knowledge to the robot to allow it to anticipate the commands or feedback that the user utters in response to an action of the robot or a change in the scenario. Moreover, the training tasks should be designed in such a way, that they can be performed without boring or stressing the user.

As the user should be able to train the robot independently, the robot needs to be able to explore its user's commands and feedback behavior autonomously without remote control. This is ensured by using training tasks, in which the robot can anticipate its users commands and feedback with very high confidence. For example, in the experiment for learning feedback, game-tasks were used, in which the robot could easily judge whether a move was good or bad. Moreover, the tasks were designed in a way that it was also obvious for the user, whether the robot had made a good or bad move and give appropriate feedback. This allowed the robot to rely on the heuristic, that a good move results in positive feedback and a bad move results in negative feedback. Based on this heuristic, it could associate stimuli, that occurred, when positive feedback was expected, with positive feedback, and associate stimuli, that occurred in a situation when negative feedback was expected, with negative feedback. In a similar way, the meaning of object names and commands was provided by the training task through vizualisations in the "'virtual living room"' scenario. For example, when the carpet in the "'virtual living room"' got visibly dirty, the robot expected a command from the user, telling it to clean the carpet. Details on the training tasks, used for learning to understand feedback and parameterized commands, can be found in chapters 3, 4 and 5

1.3.2 The learning algorithm

The learning algorithm was designed with the goal to allow the robot to learn a certain user's natural way of giving commands and feedback using speech, prosody and touch without posing any restrictions on users' expressions, such as having to use certain utterances or a simplified grammar when interacting with the robot and without requiring transcriptions or manual processing of the user's interaction.

As interaction behavior typically varies between users and even for the same user, the robot needs to learn how to deal with multiple stimuli that refer to the same meaning and multiple meanings for the same user behavior.

A combination of Hidden Markov Models [89] and classical conditioning [67] is used for enabling the robot to learn its user's preferred ways of giving reward and instruction, integrating multiple modalities and handling ambiguous utterances and multiple utterances with the same meaning. Details of the learning algorithm are given in chapters 4 and 5.

In the first learning stage, which is modeled after the stimulus encoding, that occurs in human perception, the different individual stimuli are clustered in an unsupervised way using Hidden Markov Models. Similar stimuli are clustered to train a model for their recognition. The clustering for speech utterances uses user-independent phoneme models as a basis for creating and training user-specific utterance models. For clustering prosody information, the k-means algorithm is used with global prosody information, such as mean pitch or energy, calculated from each utterance. Touch is classified, based on its duration and the sensor of the robot, that was touched by the user.

The second stage is based on the idea of associative learning, which uses the trained models, generated in the stimulus encoding phase and associates them with a meaning. In this stage the meanings, such as positive or negative feedback, which are provided through the training tasks, are used to train an association matrix storing the associative strengths between the stimuli and their meanings. Using the trained association matrix, multiple models can easily be associated with the same meaning and also multiple meanings can be associated with the same stimulus, e.g. calling the robot's name can be associated with both, positive and negative feedback. Depending on how often certain stimuli, such as utterances and touches are given by a user, when a certain meaning is expected, their association with that meaning becomes stronger or weaker.

1.4 Contributions of this work

The goal of this work was to enable a robot to learn to understand commands and feedback from a human through natural, situated interaction. The main contributions are:

- A two-staged learning algorithm based on Hidden Markov Models and classical conditioning to learn multimodal commands, object names and feedback from spontaneous, unrestricted, non-transcribed utterances.
- A method for teaching a robot by natural interaction in "'Virtual Training Tasks"'
- User studies on how users employ different modalities to teach robots.
- An implementation of a client-server based framework for learning and recording audio and video data based on virtual tasks, which was created with a focus on extensibility and reusability and can easily be adapted to new robots as well as new tasks

1.5 Organization of the thesis

This thesis is structured as follows: Chapter 1 gives a general introduction to the topic and the motivation for this work.

Chapter 2 presents an overview of existing work on robot learning and human-robot interaction that this thesis is built upon. It also gives a brief introduction into learning and speech acquisition in humans, which has inspired the learning method, presented in this thesis.

Chapter 3 outlines three user studies that were conducted to explore users' ways of giving feedback and commands to robots: The first exploratory study investigated the effect of restrictions in giving feedback and the dependence of user feedback on the robot's behavior. The second study quantitatively analyzed the way in which users give feedback

to a robot and the final study compared feedback and commands given to a pet-robot a humanoid robot.

Chapter 4 presents the learning algorithm for learning positive and negative feedback. The chapter outlines the requirements for such an algorithm, based on the experiments, presents basic techniques, used for learning, and explains the implemented learning algorithm and the results of the learning process in detail.

Chapter 5 presents the extensions of the learning algorithm for the more general task of learning commands with parameters. It analyses requirements for learning commands and outlines similarities and differences between the problem of symbol grounding and the proposed method for command learning. Based on this, it explains the modifications, made to the original algorithm, such as the extensions for learning the correct order and positions of parameters in a command, and presents the results of the learning process.

Chapter 6 gives details on the client-server based implementation of the experimental framework and the implementation of the learning method. It shows the different programs, which were developed for the experiments, as well as the usage of external tools which were used for implementing the learning method.

Finally, in chapter 7 the results of this work are discussed and conclusions are drawn, identifying possible areas for improvements and future work.

"Le langage est source de malentendus."
("Language is the source of misunderstandings.")
Antoine de Saint-Exupéry (1900 - 1944)

2

Multimodal perception and learning in humans and robots

2.1 Overview

This chapter discusses related work on speech acquisition and multimodal perception in humans and robots and gives an overview over previous research, that has influenced this work. The first section gives a brief overview over findings in human understanding of multimodal interaction and human language acquisition, outlining the biological foundations that have inspired this work. The rest of the chapter discusses related work. The proposed method for training a robot and learning multimodal commands and feedback is linked to different fields of computer science and robotics:

Systems, that use multimodal fusion, such as the integration of speech and prosody, to improve the recognition accuracy in speech-centered communication have been researched in depth in the field of intelligent user interfaces as well as human-robot interaction.

In recent years, different approaches toward learning through user feedback, symbol grounding and speech acquisition in robots, as well as training speech models without requiring transcriptions have been implemented by various researchers. There has also been a great deal of research investigating how users interact with robots and teach robots and on how the appearance of a robot influences the interaction with the robot and the impression of the user.

Taking into account these works from various fields this chapter concludes with a discussion of the features that differentiate the proposed method from previous work.

2.2 Human speech acquisition

The ability to speak and to understand spoken utterances is one of the most unique capabilities of human beings. It starts to develop in the first months in the life of an infant and

progresses very quickly during the first few years. However, speech acquisition does not only occur in children, but humans actually continue to learn new words along with their meanings and usage throughout adulthood [21].

The learning algorithm, implemented in this work, is inspired by findings from language and speech acquisition in infants. However, it does not claim to implement an accurate model of all processes which occur in natural associative learning and understanding of elementary utterances. Instead, it focuses on the concepts which appear most relevant to the research objective of learning to understand human commands and feedback for a robot and which can be implemented effectively, such as the interplay of bottom-up processes and top-down processes of speech understanding as well as the different functional layers of stimulus encoding and associative learning, which exist in human speech acquisition.

2.2.1 Speech bootstrapping in infants

Children's first words are usually the names of objects in their environment [28][9]. In a study on infant word learning, Huttenlocher et al. [36] found that the first words, learned by a child are names of small, mobile objects, such as "ball", "shoe", or "apple". They assume that the most probable reason for this is, that these are objects that they can take and manipulate with their hands and use the visual and tactile information to create a mental model of these objects. The learning of words, referring to actions, such as "walk", "eat" or "play" usually starts after the learning of object references. An overview the processes involved in infants' early speech acquisition is given by Roy in [70].

Jusczyk [45] has proposed a model of recognizing words in continuous speech utterances. He assumes that an infant is born with language independent abilities of distinguishing phonemes. Based on the frequencies of phonemes in its native language, the child learns to precisely distinguish phonemes, actually occurring in the native language while losing the ability to distinguish other phonemes. Werker et al. [86] outline the processes that are involved in children's bootstrapping of speech perception and the development of infants' perceptual systems. According to Werker, speech perception in infants starts from being able to distinguish speech sounds from non-speech sounds and continues with adapting the perception of phonemes to the native language. Next, the infants learn word segmentation and associate the words with actual objects.

The method, presented in this thesis, is modeled in such a way that its behavior resembles the findings on speech bootstrapping in children. It learns to understand object names based on an initial set of known phonemes, which is used for creating word models and is adapted to the actually perceived utterances. The robot first learns object names and then uses the learned object names to segment commands and their parameters when learning commands based on continuous speech from its user.

2.2.2 Stimulus encoding for associative learning

Before a child can establish an association between a stimulus and its meaning, the physical stimulus needs to be converted into a representation that the brain can deal with. This process is called *stimulus encoding* [21]. Stimulus encoding also enables the brain to abstract from the concrete individual stimuli – which always differ to some extend – to attain a common representation.

For speech, the process of phonological encoding develops and refines in the first months of an infant's life. Experiments found, that infants' speech acquisition starts from acquiring a proper way of encoding speech-based stimuli [86] several months before they are actually able to learn the meaning of words by associative learning.

Evidence of these two stages of stimulus encoding and learning associations with word meanings has been found in experiments on classical conditioning as well as infant word learning [21] [86].

The proposed system adopts this separation between the stimulus encoding and the learning of associations between stimuli and their meanings for the implementation of the learning algorithm. It combines a stimulus encoding phase based on unsupervised clustering of similar perceptions and an associative learning phase using classical conditioning as a supervised learning method. This way the system can learn the meaning of feedback from the user during natural interaction, because the learning algorithm does not require any explicit information, such as transcriptions of the user's utterances for stimulus encoding. It only needs the information about the meaning of a stimulus, such as positive or negative feedback or commands, to associate the HMMs with their correct meanings.

Classical Conditioning and Associative Learning

Associative learning is the process of learning connections between stimuli, ideas and percepts, which are repeatedly experienced together. The theory of classical conditioning explains how humans and animals learn associations and was first described by I. Pavlov [61]. It originates from behavioral research in animals. It models the learning of associations in animals as well as in humans. In classical conditioning, an association between a new, motivationally neutral stimulus, the *conditioned stimulus (CS)*, and a motivationally meaningful stimulus, the *unconditioned stimulus (US)*, is learned [7]. The unconditioned stimulus produces an unconditioned reaction *(UR)* as a natural behavior. After completing training, which is done by repeatedly presenting the conditioned stimulus just before the occurrence of the unconditioned stimulus, the conditioned stimulus is able to evoke the same reaction, when it is presented alone. This reaction is called the conditioned reaction.

Pavlov found this relationship while he was doing experiments investigating the gastric function of dogs and measuring the amount of their salivation in response to food.

At first the dog did not show any reaction to the tone of a bell (CS) but when the dog was given food (US), it salivated (UR). After repeatedly ringing the bell just before feeding the dog, the tone of the bell alone was able to make the dog salivate.

This example serves as a model for the implementation of classical conditioning for the associative learning phase of the proposed learning algorithm. In the proposed system, the symbolic representations of meanings are used as US. The models of the user's utterances, prosody patterns and touches are CS that get associated with approval or disapproval during the feedback association learning phase.

The relation of classical conditioning to the phase of learning word meanings in human speech acquisition has been postulated in the book "Verbal Behavior" by B. F. Skinner [73] and has been adopted and modified by researchers in the field of behavior analysis. In [74] Staats et al. describe an early approach to psychologically explain the learning word meanings by classical conditioning. An explanation of more complex phenomena in learning word meanings by conditioning is described by B. Lowenkron in [56].

For the task of learning multimodal commands and feedback patterns, the most relevant properties of classical conditioning are blocking, extinction and second-order-conditioning as well as sensory preconditioning:

Blocking occurs, when a CS_1 is paired with a US, and then conditioning is performed for the CS_1 and a new CS_2 to the same US [7]. In this case, the existing association between the CS_1 and the US blocks the learning of the association between the CS_2 and the US as the CS_2 does not provide additional information to predict the occurrence of the US. The strength of the blocking is proportional to the strength of the existing association between the CS_1 and the US.

For the learning of multimodal interaction patterns, blocking is helpful, as it allows the system to emphasize the stimuli that are most relevant.

Extinction refers to the situation, where a CS that has been associated with a US, is presented without the US. In that case, the association between the CS and the US is weakened. [7] This capability is necessary to deal with changes in user behavior and mistakes, made during the training phase, such as misunderstandings of the situation resulting in incorrect feedback.

Sensory preconditioning and second-order conditioning describe the learning of an association between a CS_1 and a CS_2, so that if the CS_1 occurs together with the US, the association of the CS_2 towards the US is strengthened, too. [7] In sensory preconditioning, learning the association between CS_1 and CS_2 is established before learning the association towards the US, in second-order conditioning, the association between the US and CS_1 is learned beforehand, and the association between CS_1 and CS_2 is

learned later. Secondary preconditioning and second-order conditioning are important for the proposed learning method, as they enable the system to learn connections between stimuli in different modalities.

Top-Down and Bottom-Up-Processes in Speech Understanding

Human perception is not an unidirectional process, that converts physical stimuli into meanings, but involves bottom-up as well as top-down processes. [21]. The *bottom-up processes* are triggered by the physical stimuli, such as audio signals received by the inner ear. The *top-down processes*, on the other hand, are based on the context in which a specific stimulus occurs. The context is used to generate expectations about which perceptions are likely to occur. Both, bottom-up and top-down processes, work together in human perception of audio-visual signals to determine the best explanation of the available data.

The interplay of bottom-up processes and top-down processes in speech perception has been investigated in detail by psychologists [21]. W. F. Ganong [27] found, that if a person heard a spoken word with an ambiguous phoneme, such as a mixture between "d" and "t", and one of the possible phonemes resulted in a correct word, while the other one did not, such as "drash"/"trash", the participants were more likely to identify the ambiguous phoneme as the one, that belonged to a correct word. C.M. Connine [18] found that the meaning of a sentence, that an ambiguous phoneme is presented in, has an influence on its identification. These findings suggested, that perception is not only driven by the physical stimulus but also depends on expectations generated from the context. Figure 2.1 shows an overview of bottom-up and top-down processes in human speech perception.

In the proposed system, top-down processes are used to improve the selection accuracy when choosing an HMM for retraining. They generate an expectation, which utterances or prosodic patterns are likely to occur, based on context information. The context information is calculated from the state of the training task, which provides the knowledge, which command or feedback is expected, using the previously learned associations between HMMs and the symbolic representations of commands or positive/negative feedback. This way HMMs, that have previously been associated with positive/negative reward or a certain command, become more likely to be selected, when another positive/negative reward or the same command is expected.

The implementation of top-down processes in the proposed learning algorithm is shown in section 4.8.

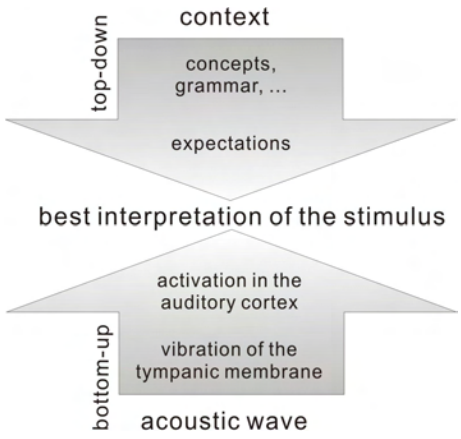

Figure 2.1: Bottom-up and top-down processes in speech perception.

2.3 Related Work

This work is related to research from different fields in human-computer interaction and human-robot interaction, such as multimodality, speech acquisition, learning through human guidance and studies on factors that influence users' interaction with robots.

2.3.1 Multimodality

The method, proposed in this thesis, integrates information from different modalities to reliably learn and recognize users' commands and feedback. Using multimodal information to improve the reliability of speech or object recognition has been an active field of research since a quite early stage of computing history. The first predecessor of current multimodal user interfaces was the "'Put that there!"'-system, which was proposed by R. Bolt already in 1980 [11]. It integrated speech and gestures to manipulate objects in a virtual space.

Multimodality and especially the integration of speech and pointing gestures is also applied in human-robot interaction. While many robots, that are designed for interacting with people, use speech and gestures when communicating with the users, only some of them are also able to process multimodal input from the user.

In [85] a system for multimodal interaction with a factory robot is outlined. Wallhoff et al. developed the system, which can solve an assembly task in cooperation with the user. The factory robot uses video data, 3D-data, speech, hand detection, detection of the storage boxes, used for the task as well as eye gaze and projected buttons as multimodal input for the robot.

A multimodal dialog system for the robot BIRON, that combines the recognition of speech and accompanying gestures, was developed by Haasch et al. [33]. It is used to teach the robot in a home-task scenario. More details on the implementation of the multimodal dialog system as well as the BIRON robot are given in [81] and [22].

Giuliani et al. [32] created a multimodal recognition system for artificial agents, such as computers and robots, that was designed with the focus on adaptability to various situations and integrates facial expression, head movements and speech. They presented an application of their system in a kitchen scenario with a mobile robot.

Stiefelhagen et al. [77] [78] have developed a system for multimodal interaction using speech, head pose and hand gestures for their humanoid robot ARMAR III. The system is able to recognize known users and aims at conducting natural multimodal dialogs with users. It can follow dialogs with multiple persons and automatically recognizie the addressee of dialog acts etc.. The system was evaluated in various different scenarios.

However, previous approaches to multimodal interaction with a robot typically use a predefined dialog system and do not learn multimodal communication through interaction with a user.

Various studies have been conducted to investigate how users apply different modalities when interacting with a computer system. In [60] Oviatt gives an overview of some major results of recent research in multimodality and points out ten of the most common misconceptions about how people interact with multimodal voice- and gesture-based systems. However, most quantitative analyses on multimodal user behavior have been conducted using GUI-based systems. Although several implementations of multimodal systems for Human-Robot-Interaction can be found in literature, such as the ones outlined above, little knowledge exists on user preferences for natural, unrestricted multimodal interaction with robots. As embodiment needs to be considered as an important factor in Human-Robot-Interaction [84], it is unclear in how far results from the interaction with GUI-based systems can be transferred to Human-Robot-Interaction.

2.3.2 Human-robot interaction

In the early days of robotics, which started with the first industry robot, Unimate, built in 1961 [68], most robots were huge machines used in industry and users of robots were usually experts. Since robots are becoming more and more a part of our everyday lives, the requirements for robots, concerning usability and the ability to interact with non-expert users increase.

In response to these new requirements, the research area of human-robot interaction has developed within the field of human-computer interaction in recent years to investigate on safe, comfortable and socially acceptable interaction between humans and robots. Human-

robot interaction strives to understand how people like to interact with robots and how natural interaction between a robot and a human can be realized.

Human-robot interaction is a very diverse, interdisciplinary field, which is influenced not only by computer science and robotics but also by psychology and social sciences. The questions that are most relevant for this thesis are how robots can acquire speech understanding through interaction with a user, how users can intuitively teach robots and which factors affect users' ways of interacting with robots.

Speech acquisition in robots

There has been a great deal of research on adapting robots to their users, speech acquisition and understanding natural language [40] [48] and recognizing affect and emotions for human-robot-interaction [13][55][59].

Learning the connections between words and their meanings through natural interaction with a user has been researched in the field of language acquisition. [15][40][48]. In speech acquisition a system, such as a robot, learns to understand the meaning of a user's utterances from non-transcribed speech data using visual or other cues, that provide information on the meanings of the utterances.

One of the most comprehensive systems for robot speech acquisition was developed by Iwahashi et al. [40] [41] [42] [43]. It allows the active and unsupervised acquisition of new words, simple grammar as well as pragmatic and communicative capabilities for the multimodal interface of a robot. Iwahashi et al. applied Hidden Markov Models to learn verbal representations of objects and motions, perceived by a camera. The robot learns to handle users' utterances following a simple grammar without any function words, such as "Red box Kermit move-over".

In the first step of the learning process, the system learns speech-unit HMMs, similar to phoneme models, in an unsupervised way from one minute of speech from the user. These speech-unit HMMs are then used to learn words from three different word classes using speech and visual information: Words, that refer to perceptual characteristics of objects, such as "box", "big" or "red", words that refer to abstract meanings, such as "tool" or "food", which describe classes of objects, that are characterized by the same functions, and words, that refer to motions, which are characterized by trajectories of a trajector (one of the previously learned objects, that is moved), relative to a landmark (another object, which is not moved). The user has to show the objects and demonstrate the motion verbs like move-onto, move-over, move-around etc. to the robot.

Based on these known words, the system can infer a simple grammar by statistically modeling the sequences of words, spoken by the user. E.g. if the user says "Kermit box move-onto", while putting the Kermit puppet on the box, then the sequence {trajector, landmark, motion} would be learned as a possible grammatically correct utterance.

Using the grammar, the system learns to actually execute commands. In this stage, the user utters a command, such as "Big box red box move-over" and the robot executes the command. If it performed incorrectly, the user slaps the robot arm for negative feedback. The robot maintains and updates a belief system, consisting of own, learned beliefs, that are used for understanding the user's utterances. e.g. "an object with certain visual features is called "Kermit"". It also maintains beliefs about the user's belief system, that allow the robot to estimate, whether the user would consider a certain utterance, action or pointing gesture from the robot as correct. The belief system is a statistical model, which allows the robot to calculate the most likely interpretation of an utterance given the current situation, the speech utterance, the learned words, etc. It also estimates a confidence value and calculates an appropriate reaction to the user's command.

The pragmatic capabilities enable the robot to estimate, which object the user refers to, in case of ambiguities, or understand that it cannot execute a command well enough without asking back to the user. E.g. if there are multiple boxes of different colors, and the user says "Kermit box move-onto", then the robot can ask back, which box the user refers to, if its estimate of the certainty of understanding the user's believes, is below a threshold.

Finally, the system learns to understand question words and to answer to its user's questions. In the experiments, the robot learned the word "What", which was used to make the robot utter the name of an object and "which" which was used to make the robot point to an object. For example, when the user moves the Kermit puppet over the box and asks "Box what move-over", then the robot would have to answer "Kermit", if the user says "Kermit Which", then the robot would have to point to the Kermit puppet.

The approach, proposed in this thesis, only tries to solve a small subset of the problems, that are handled by Iwahashi's system. The main drawback of his system is, that it can only learn strongly simplified utterances. Learning to understand non-simplified, naturally spoken utterances without restricting the usage of words and grammar was one of the main targets of this research.

Kayikci et al. [48] utilized Hidden Markov Models and a neural associative memory for learning to understand short speech commands in a three-staged recognition procedure.

The system has three different modules, a HMM-based triphone recognizer, a word recognizer and a sentence recognizer. Each of the modules generates hypotheses, which are then passed to the next module. E.g. the phoneme recognizer estimates the most likely sequence of triphones and then passes it to the word recognizer, which returns the most likely sequence of words. Ambiguities, that cannot be solved in one module, can be passed forward to the next module, which may be able to decide for one of the possibilities.

When an utterance is recorded, such as "Bot put red plum yellow lemon", the system first passes it to the phoneme recognizer, which returns a recognized sequence of triphones using standard HMMs. Then the phoneme sequences are translated into a sequence of the most likely words, spoken by the user, using a neural associative memory. This step allows

in many cases to correct phonemes, that were incorrectly recognized by the HMM based recognizer. Ambiguities may be carried over to the next stage, if there are multiple, possible phonemes, which would all result in a correct english word e.g. the word "bwall", which may either mean "ball" or "wall". Finally, a neural associative memory is used to obtain the most likely semantic representation of the utterance. The sentence recognizer tries to disambiguate ambiguous word hypotheses using the grammatical context of the words, learned associations between words or information from outside the speech recognizer, such as visual input. The system can be used to teach object names and simplified sentences to a robot.

Roy [69] proposed a model of cross-channel early lexical learning to segment speech and learn the names of objects, which are recorded by a camera. He used insights from infants' word learning and recorded the speech samples for training the robot through experiments with mothers playing with their infants.

He used models of long term memory and short time memory to find and learn recurring auditory patterns, which are likely to be object names. The short-term memory stores the recently perceived audio and video information. Speech is represented as a phoneme sequence. The short-term memory searches for recurring phoneme sequences occurring in the same visual context. The long term memory searches for phoneme sequences that often occur together with similar visual contexts and rarely occur in other contexts over time.

The basic concept of the learning method is discovering words by segmenting utterance parts, that reliably predict the occurrence of a certain visual stimulus. If the word "ball" occurs is many utterances of the user, while the user is showing a ball to the robot, the robot identifies this recurring speech pattern and learns to associate it with the visual representation of the ball, perceived by its camera.

The most remarkable capability of the system is, that it does not need object names to be uttered as single words, but is able to segment word names from continuous utterances by relying on the co-occurrence of speech and visual perceptions. However, it does not learn names of actions or other word categories.

Steels and Kaplan [72] developed a system to teach the names of three different objects to an AIBO pet robot. They used so-called "language games" for teaching the connections between visual perceptions of an object and the name of the object through social learning with a human instructor.

They conducted experiments with three different games. In the first game, the robot performed social learning and learned through intensive interaction with the user. The user drew the attention of the robot to a certain object and taught it the name of the object. In the second game, the user created a situation for the robot in a way, that the robot could learn by supervised learning from observations. The user uttered the names of objects, when the robot looked at them. In the third game, the robot performed unsupervised

learning and tried to cluster the visual information about the objects without any utterances from the user.

They found, that the recognition rate improved considerably through applying social learning. An overall classification rate of 82% for three visually presented objects and their names was reached for social learning and 58% for supervised observational learning. For unsupervised learning, the robot created nine clusters for the three different objects and four of the clusters contained views of two or more different objects. Their result shows, that language is important for learning to visually classify objects.

Gorin et al. [31] as well as Alshawi [3] proposed techniques for learning spoken language from training data without transcriptions. The focus of their work was different from the previous ones, as their methods were not used for robot speech acquisition but for mapping whole, short utterances to non-parameterized actions in a telephone dialog system. Both systems could learn without transcriptions but could not detect and handle parameters in parameterized utterances.

Learning through human instruction and feedback

Various researchers have investigated how robots can lean through human instruction, use human feedback and also learn to understand human feedback. One approach that is particularly related to this work is presented in [55]. Kim and Scassellati described an approach to recognize approval and disapproval in a Human-Robot teaching scenario and used it to refine the robot's waving movement by Q-Learning. They employed a single-modal approach to discriminate between approval and disapproval based on prosody.

Thomaz et al. described an experimental setting for assessing human reward behavior and its contingency [79][80]. The participants of the study could give positive as well as negative reward to teach the virtual character Sophie to bake a cake in the "Sophie's Kitchen" scenario. Reward could be given by an interactive reward interface that allowed the user to assign any reward on a scale from -1 to +1 either to a certain object or to the world state. The character learned from a human teacher by this kind of reinforcement. In their experiments they found a strong bias towards positive reward and discovered a phenomenon that they described as anticipatory rewards, positive rewards that were assigned to an object that the character has to use in a later step. This kind of reward can be interpreted as guidance for the character.

Blumberg et al. [10] showed how to teach a synthetic dog-like character by clicker training based on reinforcement learning. Clicker training was also used by Kaplan et al., who presented a method to train complex tasks to an AIBO robot by performing clicker training [47]. However, his study mainly focused on adapting reinforcement learning to work with a human teacher and on an adequate way to model the state- and action-space for reinforcement learning.

Yamada et al. described an approach to mutual adaptation between a human and an AIBO type robot based on classical conditioning using the Klopf neuron model [88]. While the robot learned to interpret the human's commands, given by touching its touch sensors, the human found out in the course of the experiments, which stimuli the robot understood in what way.

Ullerstam and Mizukawa [83] developed a method to teach complex behavior patterns to an AIBO pet robot based on reinforcement learning. In their system, reward is given through two predefined positive and negative utterances, as well as pressing the robot's head or back sensor.

Effect of the robot's appearance

Various studies [52] [49] [29] [35] have investigated the effect of a robot's appearance on the interaction with a user. However, most studies concerning the appearance of robots rather deal with the uncanny valley effect [19] and users' impression of robots than with the effect of a robot's appearance on its user's communicative behavior. The effects, that make users treat robots differently, depending on their appearance and behavior, need to be understood in order to design robots and robotic behavior in such a way, that they can be handled intuitively by naive users.

Kanda et al. [49] conducted a study with two different humanoid robots and showed that different appearances of the robots did not affect the participants' verbal behavior but did affect their non-verbal behavior such as distance and delay of response. They explain the observed differences by impressions, such as novelty, safety, familiarity and activity as well as attributions, such as whether the robot is respected as a conversation partner.

Kriz et al. [52] investigated users' conceptualizations of robots by analyzing the way the users talked to the robot. They compared features of robot-directed speech to how humans talk to infants or adult non-native speakers. They found that the participants spoke more loudly, raised their pitch, and hyperarticulated when they spoke to the robot. This behavior is typical when the conversation partner is assumed to have low linguistic competence. However, they did not speak in easier sentences, which suggests, that they believed that the robot has almost humanlike cognitive capabilities.

Goetz et al. [29] investigated users' attribution of capabilities depending on the appearance of a robot. They created images of more or less human-like looking robots and had participants judge their suitability for different tasks. They found that people systematically preferred robots for jobs when the robot's human-likeness matched the sociability required in those jobs. They also found in a second user study with a humanoid robot, that playful or serious demeanor of the robot affects the compliance of the participants. The participants performed a playful task longer, when the instructing robot showed a playful demeanor while the participants performed a serious task longer, when the robot behaved more seriously.

Similar results were obtained by Hegel et al. [35] who found that the appearance of robots affected users' attribution of possible applications. They conducted a user study in which the participants were asked to match videos of twelve robots to thirteen different categories of applications. Especially the perceived human-likeness or animal-likeness affected which tasks the participants considered suitable for each robot. While the participants considered human-like robots for fields like health care, personal assistance, security and business, they considered animal-like robots as companions, entertainers, toys, and robotic pets.

The experiment, presented in this thesis, built on the knowledge from these previous studies and tried to solve the question, in how far users' multimodal behavior as well as their way of uttering feedback and commands was influenced by an either human-like or pet-like appearance of a robot. The main research questions were in how far the usage of speech, touch and gestures varied for the two different robot types and whether certain features of speech utterances, such as their length, politeness or the amount of explanations given, are affected by the appearance of the robots. The study was based on the assumption that a more pet-like appearance of a robot might persuade the users to treat it more like a pet, while a more human-like appearance might make the interaction resemble more to the interaction between humans. Moreover, the experiments looked into the impression of the users when interacting with both robots in the same training tasks.

2.4 Unique characteristics of the proposed method

In the same way as the approaches, outlined in section 2.3.2, the proposed learning algorithm attempts at assigning a meaning to an observed auditory or touch pattern through Human-Robot Interaction using HMMs as a basis. However, the system is not trying to learn the meaning of individual words or symbols and a grammar, describing, how to connect them, but focuses on learning patterns expressing a feedback as a whole utterance or a command as a command pattern with placeholders for parameters. Moreover, the proposed approach is not limited to learning speech-utterances but tries to integrate perceptions from speech, prosody and touch.

In contrast to previous approaches in robotic language acquisition, such as the ones by Iwahashi [40] and Kayikci [48] the system tries to learn naturally spoken, domain-specific, parameterized commands and feedback that are not constrained by a restrictive grammar. The participants were instructed to utter commands to the robot in any way they consider natural, not restricting commands to simple utterances and even allowing implicit commands, such as saying "This room is too dark" to make the robot switch the light on. These natural utterances typically contain non-informative words such as "please" or "can you" as well as function words. There is usually no one-to-one mapping of utterances and their symbolic representations but one symbol may be represented by multiple utterances and one utterance can have multiple meanings. Most previous systems cannot handle

complex, natural utterances from a user but need a very restricted grammar. For example, in the system, proposed by Iwahashi [40], the user must not utter any function words or motion words that are divided into multiple parts, occurring at different positions in the utterance. So the system is not able to learn the utterance "Move the kermit onto the box, please", but the user has to utter the simplified sentence "Kermit box move-onto".

The previous systems for robotic speech acquisition typically learn object names, action names and grammar separately, as this facilitates the segmentation of these speech-units. Some of them only learn object names and do not learn actions or commands at all. For example, Iwahashi's system first learns object names and action names as separate words and then uses the known objects and actions to learn in the next training phase, how they can be put together to express a meaning. The method, proposed in this thesis, is able to learn command patterns and segment utterances into parameters and parts of the command during the learning process in one step from naturally spoken utterances. It still needs to learn object-names before learning to understand commands but does not require an additional step of learning action words separately. This is more similar to speech acquisition in infants, as parents often explicitly teach object names e.g. "This is a cat." but usually do not teach words for actions in this explicit way, e.g. "This is walking." but children typically learn action words from conversational utterances.

The focus of this work differs from typical symbol grounding for robots as it concentrates on learning how a certain user utters commands and feedback, but assumes that the robot already knows basic grounded symbolic representations of the set of actions, that it can perform, and is able to recognize objects and map them to symbols. That means, it can correctly interpret a written command in simple syntax like *MOVE(BALL, BOX)*. In order to react to naturally spoken commands, it needs to learn a mapping between these existing symbolic representations on the one hand and commands, object names and feedback, given through natural speech by the user, on the other hand. Having a set of known symbolic representations of objects and actions is a relatively strong requirement, but the required mapping of objects and actions to symbols can be performed without a human teacher or even be implemented based on markers or RFID tags attached to objects. Only the concrete command utterances and object names are really user-dependent. This was the main reason for deciding to focus on the mapping between simple symbolic representations and users' actual utterances.

The learning algorithm does not need any transcriptions of the users' utterances or the words and language to be used. There have been previous approaches for learning to classify utterances without transcriptions [3] [31]. However, while these approaches recognize the training utterances as a whole to classify them into actions, the proposed approach is able to segment utterances into commands and their parameters. Utterances which did not occur in the training data can be recognized if they are combinations of known parameters with known commands. This allows the system to avoid combinatorial explosion when learning actions which are combinations of commands with multiple different parameters.

2.4. UNIQUE CHARACTERISTICS OF THE PROPOSED METHOD

In the majority of systems for speech acquisition or robot leaning, understanding positive and negative feedback is preprogrammed and feedback needs to be given by fixed modalities. Only in the system by Kim et al. [55] understanding user feedback is actually learned based on prosody information. To the author's knowledge, the proposed system is the only one that learns feedback given by speech, prosody and touch through situated interaction with the user.

The algorithm combines HMMs, which are a typical method used in speech recognition and other fields which require the classification of time-series data, with concepts from biological and psychological research. For learning associations between commands, object names as well as feedback and the HMM representations of the observed user behavior, classical conditioning is used in the proposed system.

All proposed approaches for learning through feedback use either robots or virtual characters. The combination of a real robot with a virtual task is a novel approach proposed in this work. It allows the system to have different robots with different capabilities, shapes and sizes execute the exact same tasks while perceiving the interaction with the user through the robot's actual sensors and experiencing all the effects on the user behavior, that are caused by the appearance and behavior of the robot,

The main advantage of learning to understand users' interaction through virtual training tasks is, that the robot does not actually have to perform any time-consuming real-world tasks, just for learning to understand it's user's way of interacting, and that the robot can learn useful command utterances from a user, even when visual grounding is difficult or unnecessary. The existing systems for language acquisition and learning by demonstration usually use "toy tasks" for learning, which allow an easy visual grounding of symbols for objects and actions (e.g. all possible actions can be represented by a typical trajectory), and the methods cannot be easily applied to learn, for example, how a user utters actual commands for a household robot. For example, a household robot would probably be preprogrammed to know how to vacuum clean a room or wash the dishes and just needs to learn, how a certain user requests it to perform these tasks. In such cases, learning to understand utterances by demonstration would be unnecessarily time-consuming. Moreover, there are several commands, that are robot-specific and cannot be demonstrated by a human. For example, a human could not demonstrate, how to recharge, show the battery level, switch on/off etc. Visualizing and learning these commands in a virtual training tasks can therefore speed up and facilitate learning for a robot. Moreover, virtual tasks are easy to perform even for elderly and disabled persons, as they do not require much physical activity from the user.

"Brick walls are there for a reason. The brick walls are not there to keep us out. The brick walls are there to show how badly we want something."

Randy Pausch (1960 - 2008)

3

Characteristics of Users' Feedback and Commands in Human-Robot Interaction

3.1 Overview

This chapter discusses the design of the training tasks, used for command and feedback learning and summarizes the findings from three experiments concerning users' ways of giving commands and feedback in human-robot interaction tasks. While the experiments, described in this chapter, were used for analyzing the participants' behavior in a human-robot teaching scenario, they were also used to gather training data for learning to understand users' feedback and commands using the learning method proposed in chapters 4 and 5.

The first part of this chapter analyses the requirements for training tasks to be used with the proposed learning method. The main function of the training tasks is providing information on the meaning of the user's behavior, such as positive or negative feedback, to the learning algorithm, by giving the robot all necessary background information to enable it to non-intrusively persuade the user to give a certain command or feedback.

The following parts of this chapter describe three experiments on analyzing users' ways of giving feedback and commands to a robot:

- The first pilot study investigated how restrictions in the available ways of giving feedback affect the frequency and consistency of user feedback. It also looked into how users give feedback in response to different types of positive and negative actions of the robot.
- The second experiment looked into the modalities used for giving feedback as well as the typical variability and task-dependence of user feedback. This experiment also provided information on whether learning user feedback is useful and actually feasible. The data, recorded in this experiment, was used to learn user feedback as described in chapter 4.

- The third experiments dealt with users' commands and analyzed in how far commands and feedback differ for the pet-robot AIBO and a child-sized humanoid robot. The data, recorded in these experiments, was used to learn user commands and feedback as described in chapter 5

Based on the results of these three studies, the feasibility as well as the usefulness of learning to understand natural feedback and commands from a user instead of hard-coding sentences and other stimuli, that the robot can understand, are discussed.

3.2 Training tasks

In order to allow the robot to explore, how its user gives commands and feedback and in order to record positive and negative feedback for learning and further analysis, special training tasks are used. The tasks were designed based on the following requirements:

- The tasks must enable the robot to accurately guess the meaning of the user's commands or feedback and allow the robot to explore its user's way of giving commands and feedback
- The training should work without implementing a Wizard-of-Oz approach, that can only be used in a controlled experimental study. Instead the method should actually be implementable in a household robot and allow the user to train the robot independently.
- To allow independent training, the method should be easy to perform by an end-user within a home-scenario without any special equipment.

Adaptation of a robot to its user by the proposed method is done in a training phase before actually using the robot. During the training phase, the robot solves training tasks, in cooperation with the user. The training tasks are designed to allow the robot to anticipate and explore the user's commands and feedback.

The main difficulty, that has to be addressed when designing the training tasks for learning feedback and commands without actually understanding the user during the training and without using the Wizard-of-Oz principle, is that the robot needs to perform the training autonomously while not having any knowledge about its user and his or her way of interacting.

The training phase is designed to give the user the impression that the robot adequately reacts to his or her commands in a stage, where the robot actually does not understand the user. However, the training can be performed without remote controlling the robot, which would be infeasible for actual use with a newly bought service robot. Instead, the tasks are designed to ensure that the robot and the user share the same understanding of which command is expected or whether a move of the robot is good or bad. This way, the robot is

able to anticipate the user's feedback and commands and can explore its user's expressions of approval and disapproval as well as commands.

In order to learn positive and negative feedback, the robot uses its knowledge about the task to deliberately make good or bad moves to provoke positive and negative feedback. As a result, natural, situated feedback can be observed and learned.

When learning object names and commands, the training tasks are used to enable the robot to ask for object names and make the user utter commands with a predefined meaning.

In the experiments, different kinds of "virtual" training tasks are used. The robot executes these tasks in front of a big screen or a projection on a white surface, while interacting with the user. Different tasks were implemented to learn commands, object names and feedback. The experimental setting, that was used for the experiments on feedback learning can be seen in Figure 3.1.

Figure 3.1: AIBO performing training tasks.

3.2.1 Using "virtual" training tasks

An issue, the impact of which became obvious in the preparation and during the execution of preliminary experiments, is the very limited ability of the AIBO robot to physically manipulate its environment and to move precisely and quickly.

In the pilot study, described in section 3.3, this issue was addressed by equipping the robot with a shovel to facilitate moving objects around. However, in this experimental scenario it was still possible that the robot did not detect errors during task-execution, such as failing to pick up the correct object or moving imprecisely. This poses a risk for misinterpreting the current status of the task. Therefore the training tasks are implemented in a way that the robot can complete it without having to directly manipulate and observe its environment.

The tasks, that were used for the experiments, have been implemented as "virtual" training tasks. "Virtual" training tasks are tasks that do not require direct manipulation of the environment by the robot, but are computer-generated and visualized on a big screen and the robot and the user interact with each other in front of the screen. The design of the tasks allows the robot to manipulate objects on the screen and access all required information about the current situation of the robot through a direct connection to the task server. During the experiments, the image of a playfield or any other task is generated by the task server and shown on the screen, as seen in figure 3.1. The robot visualizes its moves and reacts to the user's behavior, such as utterances, touch or gestures, by motion, LEDs, gestures and sounds.

Using virtual training tasks as a basis for human-robot-communication has different benefits. One main advantage is the reduction of effort needed to implement perception and understanding of the environment, so that priority can be given to the system capabilities that are actually needed for interacting with a human.

Many commercially available robots, used in research, such as the AIBO [2] or Khepera [57] are quite small and have no or very simple actuators. So their ability to actually manipulate objects in their environment is often quite limited. AIBO, the robot, that the experiments were originally designed for, can only pick-up small cylindrical objects with its mouth and needs to approach them extremely precisely in order to be able to pick them up.

Another difficulty in real-world tasks is to detect errors during task-execution, such as failing to pick up an object, hitting any objects that are in the way etc. Failing to detect that an attempted action could not be performed successfully poses a risk for misinterpreting the current status of the task and misunderstanding the user's feedback.

Moreover, virtual tasks can be standardized easily and used with different robots that may have different sizes, shapes and capabilities for comparative user studies. One such study, comparing users' interaction with a humanoid and the dog-shaped pet robot AIBO is described in section 3.5.

For these reasons, the training tasks were designed in such a way that the robot can complete them without having to directly manipulate its environment. When using a computer-based task, the current situation of the robot can be assessed instantly and correctly by

the software at any time. It can be manipulated freely, e.g. to ensure exactly the same conditions for all participants in an experiment.

By reducing the risk, that either the robot or the user misinterprets the situation, the training tasks ensure that the robot is able to anticipate the user's next reward or instruction correctly. This is necessary for associating the observed behavior correctly with a command or positive/negative reward.

Using a robot simulation or a virtual agent can be an alternative to using a real robot in many cases. However, it has the disadvantage that interaction cannot be perceived through the actual sensors of the robot and does not occur in the same spatial context as with a real robot. Moreover, especially in case of gestures or touch, user behavior depends on inherent properties of the robot like its size and the location of its sensors and can be expected to differ significantly between interacting with a real robot and a computer simulation. In order to allow the user to train the robot in a virtual training task but use it in a real-world situation, it must be ensured, that the interaction is similar between training and actual interaction. Based on these considerations, it was decided to teach a real robot in a virtual training task in order to create an efficient teaching situation.

3.2.2 Prerequisites for designing training tasks

Deliberately provoking positive and negative rewards as well as command utterances with a predefined meaning from a user is only possible for the robot within a task where the human and the robot have the same understanding of which moves are desirable or undesirable or which commands need to be executed. If the task is sufficiently transparent so that the user and the robot have a common understanding of desirable and undesirable actions, the robot can anticipate the user's commands and explore his or her feedback behavior by performing compliant or non-compliant actions.

Considerations for learning user feedback

During the training tasks for feedback learning, the robot needs to be able to explore its user's feedback behavior autonomously without remote control. The system employs an approach for provoking users' commands and feedback that is inspired by the Wizard-of-Oz principle, which is widely used in research on Human-Robot Interaction [50]. In a Wizard-of-Oz scenario, the participant of an experiment has the impression that the robot adequately reacts to his or her interaction at a stage of development, where the robot actually cannot understand any utterances or other interaction from the user. This is realized by having a person - the wizard, who is hidden from the participant - watching the user and controlling the robot.

Instead of a person, who remote-controls the robot, the proposed learning method uses special tasks, that provide all necessary information to the robot, without the user being

aware of it. This allows the robot to robot act as if it understood the user's interaction. While the user thinks that he or she teaches the robot how to successfully perform a task, such as a simple game, by giving positive and negative feedback, the robot actually already knows how to perform the task, and therefore knows, which moves are good or bad, and can exploit this knowledge by deliberately making good or bad moves to learn to understand its user's positive and negative feedback. Because of its knowledge about the tasks and because of its ability to evaluate its own moves, the robot can accurately guess the meaning of the user's feedback and associate it with the observed utterances.

For example, in one of the tasks the user is asked to teach the robot to select an image which has the same contents as a sample image. In this task, the user gives feedback to the robot for correct and incorrect selections. Using the task server, the robot knows the position of the correct image and can therefore either select the correct image and expect positive feedback or select any of the other images and expect negative feedback. By selecting less correct images at the beginning of the task and more toward the end of the task, the robot can give the user the impression that it actually learns through his or her feedback.

Making the users believe, that the feedback is actually used by the robot to learn how to perform a task, is necessary to make sure that they give feedback in the way they would in a real-world teaching scenario. If a user is aware, that the robot actually learns to understand feedback, then he or she might try to help the robot by speaking slowly or use only simple sentences or repeat the same sentences over and over and try to accommodate the robot's learning process, which results in unnatural training examples.

In the experiments on feedback learning, four different game-based virtual training tasks were used for analyzing the feedback given by the user to the robot. The tasks resembled simple children's games and are described in detail in section 3.4.1. Two of the tasks allowed the user to give reward as well as instructions, while in the remaining two tasks, the users were asked to teach the robot by only giving positive and negative reward.

Tasks that only allow positive and negative feedback from the user for training the robot can be designed relatively freely as long as their goal is known to both, the user and the robot. However, tasks that allow the user to give additional instructions to the robot have to be designed carefully to ensure, that the user and the robot maintain the same understanding of whether an action of the robot is positive or negative. For example, when users feel, their current level of instruction is not interpreted correctly by the robot they tend to give more fine-grained instructions. E.g. when the robot did not understand "Put the red stone on the yellow field", the participants of the preliminary experiments typically changed their instructions to "go forward. stop. pick up the red stone, ...". In tasks which do not induce a total order on their subtasks, fine grained instruction can lead to a situation where the robot's actions do not comply with the user's instructions, although they lead to the correct goal state, which is known to the robot. In such cases the robot would expect positive feedback but the user would give negative feedback. Therefore, when

allowing instructions from the user, tasks for learning feedback have to be designed in a way that they are either atomic or that knowing the goal state imposes a total order on their subtasks.

However, the user's judgment of the situation remains a potential source of errors. If the users cannot fully understand, which moves are good or bad, they are likely to give feedback, that the robot does not expect. Therefore, the task must be designed in a way that the situation is easy to evaluate by the user. In three out of four tasks of the experiments for feedback learning, described in section 3.4, the goodness of the moves of the robot was fully transparent for the user. In the "Pairs" game and the "Same Image" game the user only had to evaluate whether two cards, chosen by the robot, show the same picture or not to decide, whether a move was good or bad. In the "Dog Training" game, the users had to evaluate, whether the motion of the robot corresponded to the command given. The fourth training task was used to assess changes in the participants' behavior depending on whether they are confident in the feedback, they provide, or not. For this task, the "connect four" game was used, which is a strategic game, where the evaluation of a single move is more difficult for the user and there are possible moves that are not clearly good or bad. Using these four different tasks, the task-dependence of user feedback and properties of the training tasks, that facilitate or hamper feedback learning, could be researched.

Considerations for learning object names and commands

When learning commands and object names, the training tasks need to ensure that the robot is able to ask for object names and provoke commands with a known meaning from the user. A simplified living room scenario, the "'virtual living room"', was used for this task, because this would be one of the potential working areas of a service robot.

One requirement from the learning algorithm was that the system needs to train a recognizer for object names before being able to use these object names as parameters when training commands. Therefore the task for learning commands had to consist of two successive phases. In the first one, the robot asked for object names, while in the second phase, the robot learned commands.

Designing a task for learning object names, that provides the meaning of the users' utterances to the learning algorithm, is relatively straight-forward. In order to learn an object name, the robot just points at an object and asks the user to name it. By using appropriate visualizations, it can be ensured, that the user and the robot focus their attention on the same object. This is necessary, because attaining shared attention is a prerequisite for learning and teaching object names.

However, it would have been unnatural to make the robot ask the user to teach it a certain command. Moreover the tasks needed to be designed in a way that the meanings of expected commands were provided to the user in a non-verbal way to avoid influencing the user's choice of words, sentence structure etc. when interacting with the robot.

To handle this problem, the system uses changes in the virtual living room, such as a carpet getting visibly dirty or the room getting dark, as well as thinking bubbles that visualize requests, such as watching TV or knowing the robot's battery status to make the user give appropriate commands to the robot. This enables the robot to accurately anticipate the meaning of the user's utterances and even request certain scenes from the task server, if it needed more training examples for a command.

Learning commands this way has the advantage, that even commands for actions that cannot be taught by demonstration, such as showing the battery status or switching to sleep mode, can be taught using the visualizations in the training task.

The sequence diagram in Figure 3.2 shows an example, how the robot uses the task server and the interaction with the user to learn the user's way of asking for the robot's battery status and giving negative feedback:

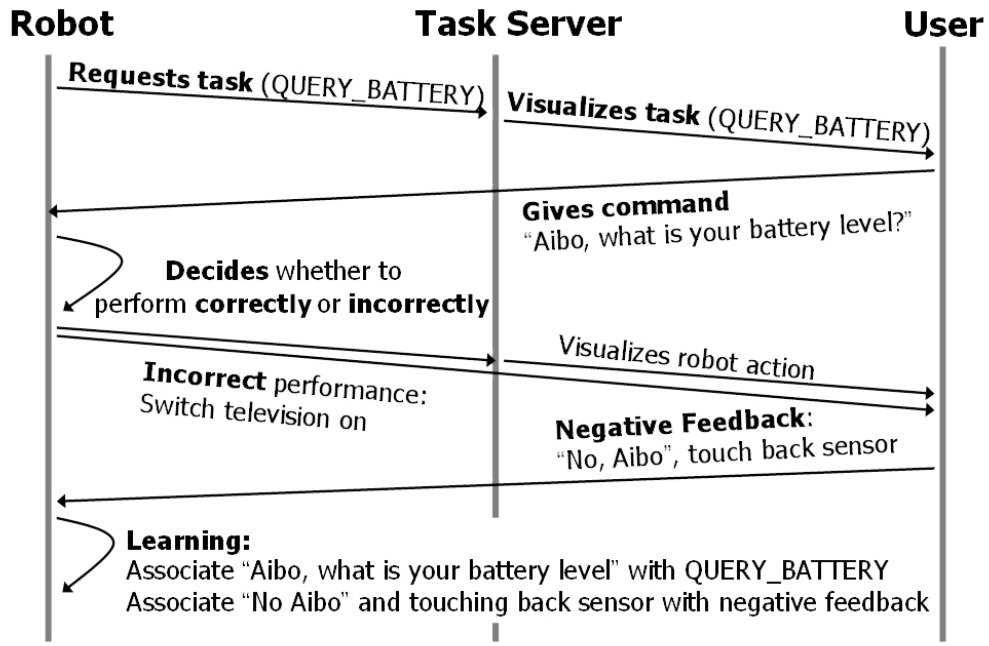

Figure 3.2: Sequence diagram of command and feedback learning

First, the robot requests the task "QUERYBATTERY" from the task server. The task server visualizes the task on the screen by showing a battery icon with a question mark. The user looks at the screen and gives the appropriate command to the robot "Aibo, what is your battery level?". The robot perceives the command, given by the user, and knows, that it means "QUERYBATTERY". It can now decide to either provoke positive feedback, by actually informing the user about its battery level or provoke negative feedback by

incorrectly doing something else, pretending it has misunderstood the user's command. In the example, shown in Figure 3.2, the robot decides to learn negative feedback, so it switches the television on instead of telling its battery level. It sends this information to the task server, which visualizes the robot's action by showing a paw icon moving to the television and switching it on. The robot makes appropriate movements and sounds to make its actions easily understandable for the user. As the robot did not execute the correct command, the user gives negative feedback, for example by saying "No Aibo!" and touching the back sensor.

Using the recorded data, consisting of expected commands and feedbacks on the one hand and perceived utterances and touch signals on the other hand, the robot leans, using the learning algorithm, proposed in this thesis, that "Aibo, what is your battery level?" means "QUERYBATTERY" and that saying "No Aibo!" in a certain tone and touching the back sensor of the robot means negative feedback.

3.3 Exploratory Study on fixed feedback vs. users' intuitive feedback

A preliminary experimental study was conducted to investigate, how the frequency and accuracy of user feedback as well as the user's impression of the interaction are affected by restrictions, such as using only fixed touch feedback or recorded utterances for giving feedback. Most previous studies, focusing on reinforcement learning with a human teacher [79] [80] [47], allow the user to only give restricted feedback, such as predefined utterances or feedback, given through a GUI interface. Throughout this pilot study the robot was fully remote-controlled and a Wizard-of-Oz-scenario was applied.

3.3.1 Experimental setting and instruction

The experimental setting can be seen in Fig. 3.3. The AIBO was equipped with a shovel attached to the front of its body, to enable it to move objects easily without having to pick them up with its mouth.

The participants were told to instruct the robot to approach one of four differently colored bricks and move it to its appropriate place. The places, that the robot had to deliver the objects to, were marked by four A3 sized sheets of paper with differently colored frames.

Each participant received a deck of cards showing which object to move and where it should be placed. One such card, representing the instruction to put the yellow object onto the red sheet of paper, is shown in Fig. 3.4.

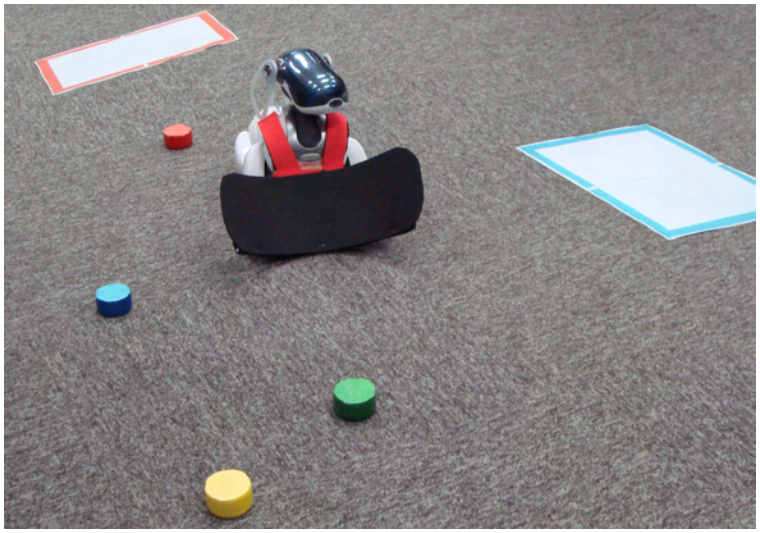

Figure 3.3: Experiment 1: Experimental Setting.

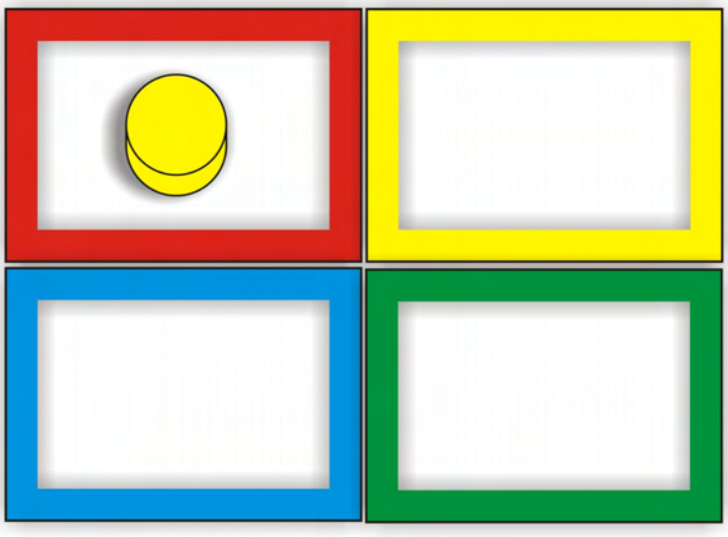

Figure 3.4: Experiment 1: Sample Instruction Card.

Every participant was given a short introduction on the concept of reinforcement and was told to give positive and negative reward to the robot, i.e. to praise and punish the robot, when necessary.

Three different reinforcement principles were introduced to each participant throughout the course of the experiment. The first reinforcement principle was introduced before the first trial, the following two reinforcement principles were introduced after three successful trials and after six successful trials. The order of reinforcement principles was changed for each user to avoid sequence effects.

The three different reinforcement principles will be referred to as "touch", "recorded" and "free" reward.

The following instructions were given to the participants describing the different reinforcement principles:

- *Touch:* If you want to give positive reward to the robot, touch its head sensor. If you want to give negative reward to the robot, touch its back sensor.
- *Recorded:* Please decide in which way you want to give positive/negative reward. You can choose any combination of spoken words, gestures as well as touching the head and back sensors of the robot. Before starting to instruct the robot, please have your choice of behavior for positive/negative reward recorded. Please stick to the chosen behavior whenever you want to give positive/negative reward to the robot. Please do not change your reward behavior after the beginning of this experiment
- *Free:* The choice of positive/ negative reward is up to you. Please give the kind of feedback you like, depending on the situation freely by voice, gestures or by touching the robot's head and back sensors.

The instructions, given to the participants as well as the questionnaire provided to them after the experiments, to evaluate their impression of the interaction, can be found in appendix 1.

3.3.2 Robot control in the experiment

During the experiments, AIBO was remote-controlled to perform the task of moving an object to its designated place, but make some mistakes in the fulfillment of the task in order to receive positive as well as negative reward from the user. The following actions were considered positive because they led to the correct fulfillment of the task. Therefore, the actions were likely to result in a positive reward from the user:

- picking up the correct object
- delivering the object to the correct target

- recovering from a mistake, e.g. changing direction when the robot was approaching the wrong target object

The robot made three different types of mistakes. Those actions were considered negative, as they hamper or slow down the fulfillment of the task. Therefore, negative reward was expected from the user:

- *"clumsy" mistake:* The robot tries to pick up an object with its shovel but misses it. The robot tries to deliver an object to a target but misses it.
- *"misunderstanding" mistake:* The robot walks into the wrong direction towards an object or target, simulating a misrecognition from the speech recognizer.
- *"lazy mistake":* The robot sits or lies down or walks very slowly.

As the different kinds of mistakes were expected to have an effect on the reward behavior of the participants, the three types of mistakes, committed by the robot, were balanced throughout the trials.

The reward from the user in the "touch" and "recorded" setting could be classified into one of four categories:

- *correct reward:* the designated reward was given. Repetition of correct reward, such as saying "good dog ... good dog" was still considered correct reward.
- plus-reward: the designated reward and some additional reward were given to the robot. (e.g. touching the robots' head sensor and saying "good robot", in the "touch" reward principle)
- *incorrect reward:* the designated reward was not given but replaced by some different reward
- *no reward:* Although one of the above described positive/negative actions was performed by the robot, no reward was given at all.

Only utterances/actions containing an implicit or explicit evaluation like "okay", "good", "bad", "no", etc. were counted as positive/negative rewards.

3.3.3 Results

The users' interaction with the robot was recorded on video and analyzed after the experiments. A still image from the recorded video is shown in Fig. 3.5.

In the experiments, a total of 109 minutes of video data were gathered from four participants, interacting with an AIBO pet robot. Altogether 141 rewards were given by the participants, 64 of which were positive, 77 negative.

All four participants were male graduate students aged 25-35 and experienced computer users but had no previous experience in interacting with entertainment robots or service robots.

Reward behavior

In this pilot study, the number of participants was rather small. Therefore, a thorough statistical analysis was not possible. However, the results can be understood as a rough direction of what effects have to be expected when developing more or less restricted reward modalities for a robot.

Figure 3.5: Experiment 1: Still image from the recorded video.

The most surprising result turned out to be that even though the participants were explicitly told to stick to the designated reward behavior and to give no different kind of reward in the "touch" and "recorded" reinforcement principle, none of the participants actually provided reward in the designated way only.

Out of 44 rewards, given by the participants with the "touch" reinforcement principle, there were 19 correct rewards, 17 plus-rewards, most of which adding a speech utterance

such as "very good!" to touching the sensors and 8 incorrect rewards, mostly providing a speech utterance only instead of touching the sensors.

Out of the 48 rewards given by participants with the "recorded" reinforcement principle, there were 23 correct re-wards, 18 plus-rewards and 7 incorrect rewards. The recorded preferred rewards differed between all four users, but all included verbal utterances. One of the users additionally decided to clap his hands for positive reward, and one participant decided to underline verbal positive reward by touching AIBO's head sensor.

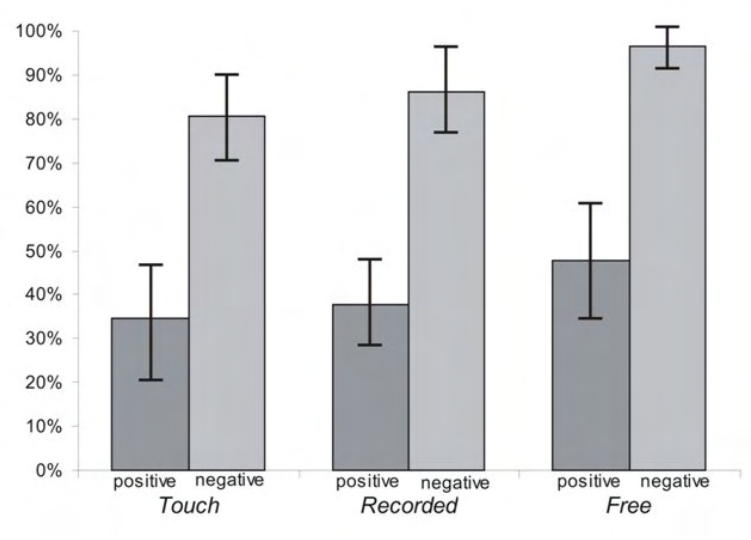

Figure 3.6: Experiment 1: Results as Bargraph.

The effect of the different reward principles on the frequency of giving rewards can be seen in Fig. 3.6. The diagram compares the percentage of positive and negative behaviors of the robot, like picking up an object, delivering an object or "misunderstanding" an instruction, in response to which a reward was given by the user, between the three reinforcement principles. The percentage of positive reward is lower because the users typically did not give positive reward to all of the positive actions performed by the AIBO. The only positive action that was always rewarded, was delivering the correct object to the correct place, but the amount of positive reward given for approaching the correct object and recovering from errors differed largely between the different participants. This is also the reason for the larger standard deviation in case of positive rewards.

In case of "touch" reward, there was reward given for 34.7% of the positive and 80.5% of the negative actions performed by the robot. In case of "recorded" reward, both reward percentages are slightly higher with 37.6% positive and 86.3% negative rewards. When it comes to "free" reward, the percentages both increase by roughly 10% compared to "recorded" reward. When reward could be given freely, 47.9% of the robot's positive and 96.4% of the robot's negative actions were rewarded by the users.

An additional finding from the experiments was that apart from giving explicit reward, the users tended to repeat their previous instructions in situations when either positive/negative reward was expected, either instead of giving a reward or in addition to a reward. This behavior was not noticeably affected by the choice of the reward principle.

The kind of mistake, committed by the robot, also affected the choice of reward behavior given by the participants. This was not only visible in the "free" reward principle but also became obvious in the "touch" and "recorded" reward principles, where especially "lazy" mistakes, which were probably rather unexpected to the participants, tended to cause punishment behavior different from the designated one.

Questionnaire

The participants were asked to fill a short questionnaire on their experience throughout the experiment and on their subjective rating of the different reinforcement principles.

Ratings from one to five could be given to rate the statements in the questionnaire, five meaning "absolutely agree", one meaning "absolutely disagree". The results can be found in Table 3.1. The values in each column are mean and standard deviation of the given responses.

In the free response part of the questionnaire, the participants could utter their comments concerning the experiments freely. Two participants remarked that, in real world, they would not want to give reward to a robot for just doing its job, especially, if it is just fulfilling a rather simple service task. This may be a cause for the bias towards negative rewards, found in the experiments. Therefore, in the following two experiments for learning feedback and commands, a teaching scenario was used and the participants were given the impression that the robot actually learned through their feedback to make them provide feedback to the robot more comfortably and avoid negative bias.

Participants also complained, that the task execution by the robot was quite slow, which was one reason for choosing "virtual" training tasks for the following experiments.

3.3.4 Discussion

The findings from the pilot study suggest that natural human feedback to a robot is relatively variable and that restricting the ways, that users can employ for giving feedback, not only affects the user's impression of the interaction but actually decreases the overall amount of feedback.

Moreover, it was found that users still give additional, "incorrect" positive and negative feedback even when only fixed feedback utterances are allowed. This may lead to misrecognitions and misinterpretations of feedback given to a robot with a fixed set of recognizable commands and feedback utterances.

Table 3.1: Experiment 1: Results of the questionnaire (standard deviations given in brackets)

	touch	**recorded**	**free**
I was able to instruct the robot in a natural way	1.65 (0.6)	2.75 (0.3)	4.75 (0.3)
I would like to give feedback to a real world service robot in the same way	1.5 (0.6)	4.0 (0)	3.5 (0.3)
Throughout the experiment, I was always sure about my next step to instruct the robot	3.5 (1.2)	4.25 (0.6)	4.5 (0.4)

3.4 User Study on feedback modalities and variability of user feedback

This section describes a user study which aimed at finding out, how users employ different modalities to give positive and negative feedback to a robot. The usage of speech, touch and gestures for giving reward was analyzed quantitatively. The experiments also assessed the variability of positive and negative feedback between users and between different tasks, that the robot performed. By analyzing and comparing users' feedback in different training tasks, it was investigated which features of a task encourage or discourage the user to give feedback in a natural way.

The experiment used virtual training tasks based on easy children's games. During the experiments, the image of the playfield was generated by a computer and projected from the back to the physical playfield, as seen in Figure 3.7. When the robot made a move, it pointed to the appropriate position on the screen and made a sound to ask the user for feedback. When playing together with the user against a computer opponent, it reacted to the moves of the computer by looking at the appropriate positions on the playfield. In order to make the moves of the robot easier to understand for the user, the game-server also visualized them on the playfield using appropriate animations. (e.g. showing a frame around the picture that AIBO is currently looking at)

In order to provoke positive and negative feedback from a user, the robot needs to know, which moves are good or bad, so that it can deliberately chose to make appropriate moves for getting positive or negative feedback. As the robot does not actually understand instructions from its user throughout the task, the user's instructions as well as positive and negative feedback need to be reliably predictable from the task-state. If the task fulfills this condition the robot can easily explore the user's reward behavior by performing in a good or bad way. The tasks and the behavior of the robot were designed with the objective to ensure that the user and the robot share the same understanding of the goal of the

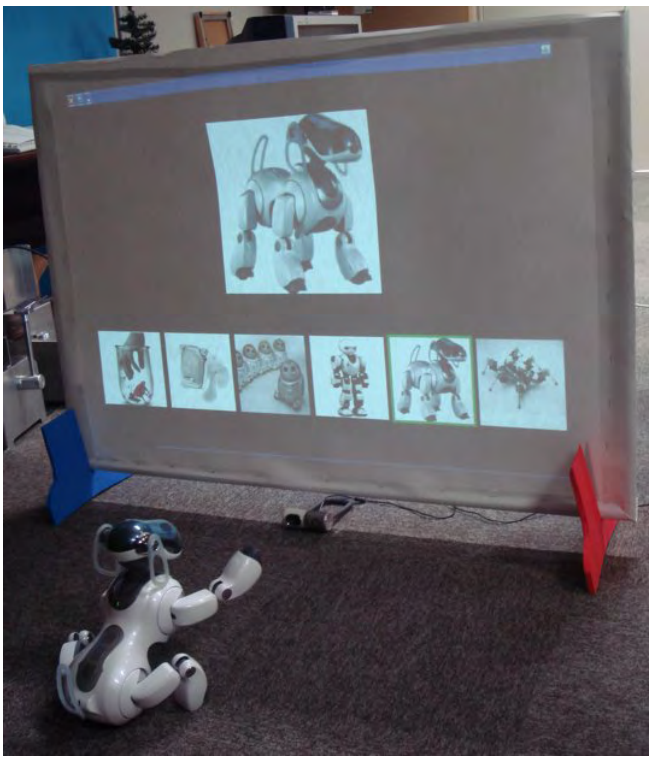

Figure 3.7: Experiment 2: Aibo playing one of the game tasks during the training phase.

task as well as the way to reach it. This is done in two ways: The rules of the game task as well as an evaluation function to calculate the quality of all possible moves are implemented in the robot, so that it knows whether a move is good or bad in a certain situation. On the other hand, tasks were selected, which are easy enough to allow the user an instant and correct evaluation of the moves of the robot and give feedback accordingly. So the robot does not need to actually understand the user's commands to know how to correctly perform the task and can deliberately provoke feedback from the user by good or bad performance.

Although the combination of Hidden Markov Models and classical conditioning is designed to be robust against occasional false training examples it is desirable to keep the number of errors as low as possible. In order to ensure that a good move of the robot results in positive reward and a bad move results in negative reward from the user, the games, used for training, must be designed in a way that the user can easily evaluate the situation. The suitability of the different training tasks has been analyzed in the user studies.

3.4.1 Selected game tasks

The following tasks were selected to be used in the experiments, because they are easy to understand and allow a user to evaluate every move instantly. Four different tasks were selected in order to see whether different properties of the task, such as the possibility to provide not only feedback but also instruction, the presence of an opponent or the game-based nature of the tasks influence the user's behavior. The tasks were implemented in such a way that the robot was not required to perform any time-consuming walking. The different training tasks were selected and implemented in a way, that they cover two dimensions which are assumed to have an impact on the interaction between the user and the robot.:

- *Easy - Difficult*: Training tasks can range from ones, that are very easy to understand and evaluate for the user, to tasks where the user has to think carefully to be able evaluate the moves of the robot correctly.
- *Constrained - Unconstrained*: In the most constrained form of interaction in the training tasks, the user is told to only give positive or negative feedback to the robot but not to give any instructions. In an unconstrained training task, the user is only informed about the goal of the task and asked to give instructions and reward to the robot freely.

The positions of the different tasks in the two dimensions can be seen in Figure 3.8. There is one task for each of the combinations "easy/constrained", "easy/unconstrained" and "difficult/constrained". The reason, why there is no task for the combination "difficult/unconstrained" is that that in such a situation, the user behavior becomes too hard to predict,

so that the robot cannot reliably anticipate positive or negative reward. Screenshots of the playfields can be seen in Figure 3.9

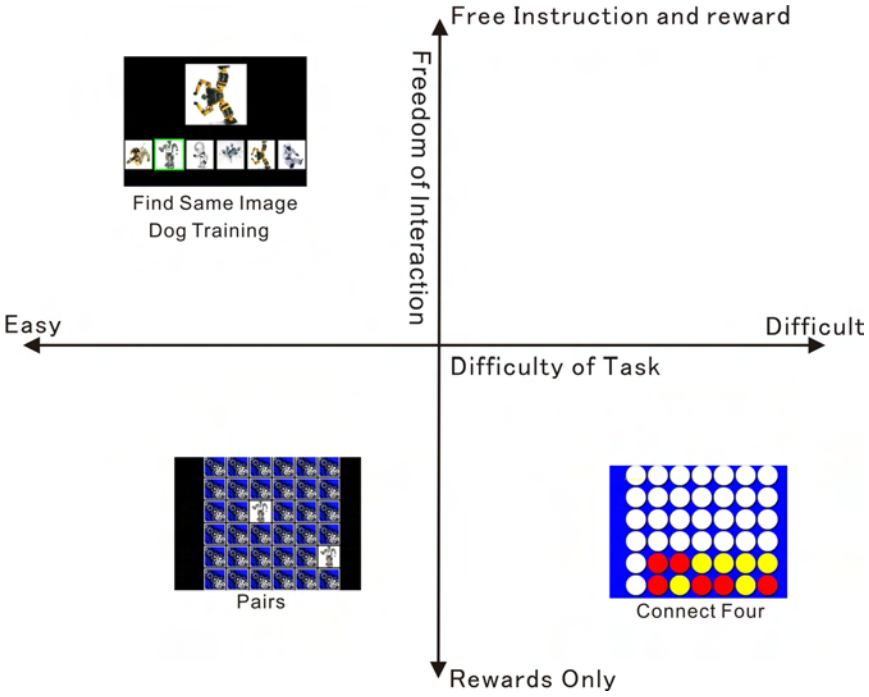

Figure 3.8: Experiment 2: Dimensions for the game tasks.

Find Same Images

On the easy/unconstrained end of the scale, there is the "Find Same Images" task. In this task, the robot has to be taught to chose the image, that corresponds to the one, shown in the center of the screen, from a row of six images. While playing, the image that the robot is currently looking or pointing at is marked with a green or red frame to make it easier for the user to understand the robot's viewing or pointing direction. By waving its tail and moving its head the robot indicates that it is waiting for feedback from its user. In this task the user can evaluate the move of the robot very easily by just looking at the sample image and the currently selected image.

The participants were asked to provide instruction as well as reward to the robot freely without any constraints to make it learn to perform the task correctly. The system was implemented in such a way that the rate of correct choices and the speed of finding the correct image increased over time.

Pairs

As an easy/constrained task, the "Pairs" game was chosen. In this task, the robot plays the classic children's game "Pairs": At the beginning of the game, all cards are displayed upside down on the playfield. The robot chooses two cards to turn around by looking and pointing at them. In case, they show the same image, the cards remain open on the playfield. Otherwise, they are turned upside down again. The goal of the game is to find all pairs of cards with same images in as little moves as possible. In this task the user can evaluate easily whether a move of the robot was good or bad by comparing the two selected images.

The participants were asked not to give instruction to the robot, which card to chose but to assist the robot in learning to play the game by giving positive and negative feedback only.

Connect Four

As a difficult/constrained task, the "Connect Four" task was selected. In the "Connect Four" game, the robot plays the game "Connect Four" against a computer player. Both players take turns to insert one stone into one of the rows in the playfield, which then drops to the lowest free space in that row. The goal of the game is, to align four stones of one's own color either vertically, horizontally or diagonally.

The participants were asked to not to give instructions to the robot but provide feedback for good and bad draws in order to make the robot learn how to win against the computer player. Judging whether a move is good or bad is considerably more difficult in the "Connect Four" task than in the three other tasks as it requires understanding the strategy of the robot and the computer player.

Dog training

The "Dog Training" task was implemented as a control task in order to detect possible differences in user behavior between the virtual tasks and "normal" Human-Robot Interaction. Like the "Find Same Images" task covers the dimensions easy/unconstrained. The user can easily evaluate the robot's behavior and use his/her way of giving instruction and reward freely without restrictions. In the "Dog Training" task, the participants were asked to teach the speech commands "forward", "back", "left", "right", "sit down" and "stand up" to the robot. The "Dog Training" task is the only task that is not game-like and does not use the "virtual playfield". Only in this task the robot was remote-controlled to ensure correct performance.

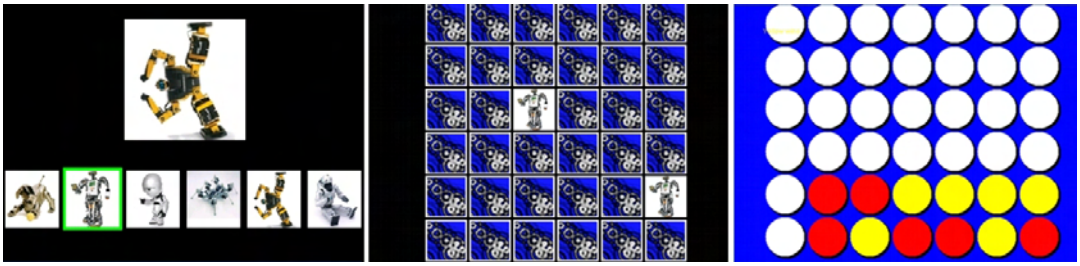

Figure 3.9: Experiment 2: Screenshots of the virtual game tasks.

3.4.2 Instruction and experimental setting

Ten persons, aged 23 to 47, participated in the study. All of them were Japanese graduate students or employees at the National Institute of Informatics in Tokyo. Five of them were females, five males. All participants had experience in using computers. Two participants had previous experience in interacting with entertainment robots. Seven participants have kept pets before, five of them have or had a pet dog. Interaction with the robot was done in Japanese. During the experiments, roughly 5.5 hours of audio and video data were recorded and analyzed. The instructions and questionnaires, given to the participants, can be found in appendix 2.

Figure 3.10 shows an overview of the experimental setting. Some videos of the interaction which were taken during the experiments are shown in Figure 3.11. During the experiments, the participants could sit or stand next to the AIBO robot in front of a white, semi-transparent screen. The playfield is projected onto the screen from the back.

The users received some general explanations on the experiment as well as a brief explanation of every game task as a written document. The participants were asked to teach the robot, how to correctly play the different games by giving instructions and positive/negative reward for the robot's moves. They were instructed to interact with the robot naturally in their preferred way by gesture, voice and/or by touching the robots touch sensors. They received the explanation that the robot adapts to their way of teaching and learns to play the different games through their instructions.

In order to record audio and video data and to endorse the impression, that the system actually processes and learns from gestures as well as speech data, a stereo camera was placed in 2.5 meters distance, facing the participant, and a microphone was attached to his/her clothes. The locations of the touch sensors on the back and the head of the robot were explained to the participants. In order to give the participants the impression, that the robot is acting independently, a third dummy camera was placed below the screen facing the robot. The participants were told that it is used by the game server for recognizing the moves of the robot.

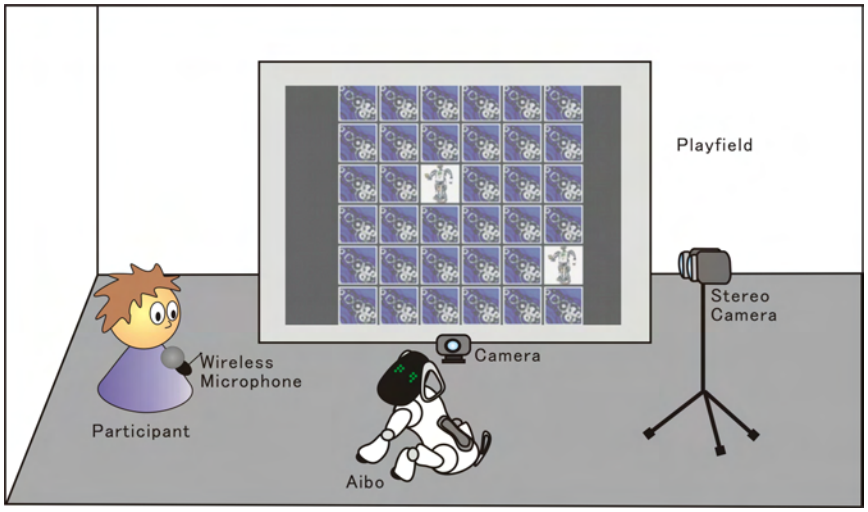

Figure 3.10: Experiment 2: Experimental setting

3.4.3 Results

As for the modalities used for giving reward, a strong preference for speech-based reward was found. Among 2409 stimuli used for giving reward, 1888 (78.37%, sd=10.51%) were given by speech, 504 (20.92%, sd=10.95%) were given by touching the robot and 17 (0.71%, sd=0.85%) were given by gestures. For the different users, the percentage of speech-based rewards ranged from 52.25% to 97.75%. Gestures were frequently employed by the participants for giving instructions, but gestures were almost never observed when users gave positive or negative reward.

Typically, multiple rewards were given for a single positive or negative behavior of the robot. Counting only the rewards given during the time, when the robot signaled that it was waiting for feedback after an action, 3.43 rewards were given for one action on average, usually including one touch reward and one to four utterances. One utterance was counted as one reward. Repetitions of an utterance were counted as multiple rewards. In case of touch reward, one or multiple contacts with the robot's touch sensors were counted as one reward, as long as the participant kept his/her hand close to the sensor.

The favorite verbal feedback differed between the users especially in case of positive reward. None of the utterances, used for positive feedback, appeared within the first six most frequently used utterances for all ten participants. On average, each person shared his/her overall most frequently used positive feedback with one other person. In case of negative reward, the feedback, given by the participants was more homogenous. The most frequently used feedback - "wrong" (chigau) - was preferred by eight out of ten persons. For the two remaining persons, it was the second and third most frequently used feedback utterance.

As for the variability of the feedback, given to the robot by an individual user: On average, participants used 12.3 (sd=6.26%) different verbal expressions to convey positive feedback and 13.4 (sd=6.65%) different expressions to express negative feedback. However, this number varies strongly between individuals: One person always used the same utterance for giving positive feedback and a second utterance for giving negative feedback while the person with the most variable feedback used 30 different expressions for giving positive and 28 different expressions for giving negative feedback. 55.61% (sd=25.03%) of all verbal feedback was given by the participants using their preferred feedback utterance. 88.73% (sd=8.17%) of a user's verbal feedback was given using one of his/her six most frequently used positive/negative utterances, so understanding a relatively small number of different utterances suffices to cover most of a participant's verbal feedback.

For positive feedback, four out of ten participants had one preferred utterance which did not vary between the four training tasks. In case of negative reward, this was true for five people. For eight out of ten participants in case of positive reward and six participants in case of negative reward, their overall most frequently used feedback utterance was among the top three feedback utterances in each individual task. In the cases, where the preferred feedback was not the same in all tasks, it typically differed for the "Connect Four" task, while in the three other tasks, including the "Dog Training" control task similar feedback was used as described above. As in the "Connect Four" task it was difficult for the users to judge, whether a move was good or bad in order to provide immediate reward, feedback tended to be very sparse and tentative like "not really good" (amari yokunai), "Is this good?" (ii kana?) or "good, isn't it" (ii deshou).

Participants' evaluation of the different tasks

The participants were asked to answer a questionnaire about their evaluation of the different tasks. They could rate their agreement with different statements concerning the interaction on a scale from one to five, where five meant "completely agree" while one meant "completely disagree". The results can be found in table 3.2. As can be seen from the table, the four tasks were considered almost equally enjoyable by the participants. For the "Find same Images" task and the "Dog Training" task, the participants' impression that the robot actually learned through their feedback and adapted to their way of teaching was strongest. Those two tasks allowed the participants to not only give feedback to the robot but also provide instructions. Moreover, they were designed in a way that the robot's performance improved over time. In the "Dog Training" task, the robot was remote-controlled to react to the user's commands and feedback in a typical Wizard of Oz-Scenario. However, in the "Find Same Images" task, which was judged almost equally positively by the participants, the user's instructions and feedback were not actually understood by the robot but anticipated from the state of the training task. This did not have a negative impact on the participants' impression that the robot understood their feedback, learned through it and adapted to their way of teaching. The lowest ratings were given for the "Connect Four" task. As

Figure 3.11: Experiment 2: Still images from the recorded video from different tasks.

Table 3.2: Experiment 2: Results of the questionnaire (standard deviations given in brackets)

	Same	Pairs	Four	Dog
Teaching the robot through the given task was enjoyable.	4.19 (1.04)	4.10 (0.83)	4.19 (0.89)	4.37 (0.81)
The robot understood my feedback.	4.73 (0.4)	4.19 (0.74)	3.10 (0.85)	4.19 (0.30)
The robot learned through my feedback.	4.64 (0.59)	3.19 (0.93)	2.55 (0.95)	4.46 (0.69)
The robot adapted to my way of teaching.	4.55 (0.66)	3.37 (1.05)	2.55 (1.04)	4.36 (0.58)
I was able to teach the robot in a natural way.	3.82 (0.96)	3.91 (0.86)	3.46 (1.12)	4.36 (0.69)
I always knew, which instruction or reward to give to the robot.	4.00 (0.72)	3.91 (0.86)	3.10 (1.02)	4.09 (0.83)

the robot's moves could not be evaluated as easily, as in the other tasks, the participants were unsure which rewards to give and therefore did not experience an effective teaching situation. This also becomes apparent in the overall low quantity of feedback given in this task which still included incorrect feedback.

3.4.4 Discussion

The user study was conducted to determine the modalities to implement in the system. It was found, that speech was by far the most frequently used modality, when giving feedback to an AIBO robot. 78.37% of all feedbacks were given by speech. It was followed by touch, which was used for 20.92% of the feedbacks. Gesture was applied for giving instructions, but did not play a significant role for giving feedback and was only used in 0.71% of the cases. Therefore, for learning and understanding users' feedback, emphasis should be placed on the modalities speech and touch, while gesture does not play a major role in conveying positive or negative reward.

Natural feedback given by different users can vary strongly, especially in case of positive rewards. Therefore, learning to understand the feedback that a certain user employs instead of using hard-wired and potentially unintuitive commands, which have to be learned by the user, helps to ensure natural interaction and a positive user experience.

Learning to understand feedback through a training task is only feasible and useful, if the feedback, given by one user is similar within different tasks. The results from the experiments suggest that this is actually the case, and that typically a limited number of utterances are used by an individual to convey positive and negative reward.

However, there are cases, where the contents of the utterances alone may not be understood as a reward. For instance, some of the users occasionally just repeated their previous command in a stricter tone before or instead of giving other negative feedback to the robot. In these cases, analyzing and learning the prosody, which determines the sentence melody of typical positive and negative feedback utterances, can be expected to improve the recognition accuracy.

Problems arise, if the user is not exactly sure, how to judge the robot's behavior, as in the "Connect Four" task. Therefore, for automatically learning rewards, the task has to be designed in a way that it is easy to understand for the user. Otherwise the amount and reliability of the given reward, as well as the user's motivation to complete the training, decrease.

The experiments have been conducted with a dog-shaped AIBO pet robot. This might have persuaded the users to interact with the robot in a similar way as with an actual pet. To ensure the transferability of the findings to general human-robot-interaction tasks, experiments with other types of robots are necessary. Because of this, a third study was

conducted, in which the differences in commands and feedback from a user for the pet robot AIBO and a humanoid robot have been assessed.

3.5 Users' commands for a humanoid vs. pet-robot

The third study targeted the issue that different robots may be treated differently by users when giving commands and feedback. While the two previous studies were conducted with the small pet-robot AIBO only, this study compares users' feedback between the previously used AIBO and a child-sized humanoid robot. Apart from feedback, the study also looked into users' ways of giving commands to the two robots.

When humans interact with other humans or with their pets they tend to adapt their way of speaking and interacting to their interaction partner. For example, people talk to adults in a more elaborated way than to small children, and they pet their dog as a reward while they would rather say "thank you" when their colleague has done them a favor. Moreover, they speak more slowly and clearly, when they assume their communication partner is not understanding them well.

It can be assumed that similar mechanisms also affect how people interact with robots. Especially the appearance of a robot and its resemblance to familiar creatures or objects can be an important factor, which helps a human to anticipate the capabilities of a robot and decide how to interact with it. The results from this research can help inform the design choices that roboticists make when considering what type of interaction they want with their robots.

3.5.1 Outline of the study

The user study investigated on how participants give commands and feedback to a pet-robot and a humanoid. As a pet-robot, the dog-shaped robot AIBO was used, which has roughly the size of a cat or a small dog. The humanoid robot, which was developed by Honda, is 1,20 m tall, which is about the size of a 8-year-old child. Both robots are shown in Fig. 3.13. The goal of the study was to find differences and similarities in user behavior when the participants give commands and feedback to the pet-robot and the humanoid.

Each participant interacted with either the humanoid or the pet-robot and instructed the robot to perform different typical household tasks like bringing a coffee, switching on the light or the TV, tidying up etc. and gave feedback to the robot for correct or incorrect performance.

Table 3.3: Experiment 3: Commands that were used in the training task.

Command	Parameters	Example sentence
move	object, place	Put the ball into the box.
bring	object	Bring me a coffee, please.
clean	object	Please clean up the carpet.
switch on	object	Robot, switch on the light.
switch off	object	Switch off the radio.
charge battery	-	Recharge your battery.
call	name	Please make a phone-call to Rita.
show status	-	What is your status?

The "Virtual Living Room"

In order to avoid time-consuming and error-prone task execution in the real world and because of the different physical capabilities of the two different robots, a "virtual living room" has been implemented. The tasks as well as the actions of the robot were visualized on a large screen. The robot was placed in front of the screen and used motion and speech to inform the user which action it is currently performing in the virtual living room. The robots' actions and pointing direction were also visualized in the living room scene with a hand or paw icon and the scene changed in response to the actions of the robot. Based on these cues the participants could easily understand the relation between the robot's motions and the changes happening in the scene. While the robots differed in shape and size all other parameters were kept as similar as possible, using the same synthesized speech utterances, similar gestures, same simulated learning rate, almost same position of the robot in relation to the user etc.

During the training, the user could figure out by looking at the scene, what command to give to the robot next. A graphical representation of the scene without any text was used in order to avoid influencing the participants' wording when giving commands to the robot.

Table 3.3 shows the list of all commands, that were used in the task along with sample utterances for each command. The users were not instructed in advance, which commands they had to teach to the robot but were asked to infer which commands were appropriate by looking at the virtual living room scene.

The training phases

One experiment with one participant comprised two successive training phases. In the first phase, the user had to teach the names of eighteen different objects to the robot. The robot

pointed at objects on the screen and a spotlight as well as a pointing arrow was shown in the living room scene to make it easier for the user to understand the robot's pointing direction. The robot then asked "What is that" ("kore ha nan desu ka?") to prompt the user for an object name. After the user uttered an object name, the robot continued with the next object.

The users were asked to only utter the object names without any additional, introductory words. This was a requirement for the learning algorithm. Because of this restriction, the speech, recorded in the first training phase was not evaluated.

In the second phase, a scene was shown on the screen. As described above, the scene visualized the task that had to be performed by the robot in a way that the user could understand which instruction would be suitable. The robot looked at the user while it was waiting for an instruction. After the user had uttered an instruction, the robot either performed correctly or incorrectly. For example when the user uttered a command like "Can you bring me a coffee" the robot pointed at the screen, made an appropriate gesture and the robot hand icon moved to the table to put the coffee cup there. Then the robot said "Here you are" ("douzo"). For a incorrect performance, it would, for example, switch the light off instead of bringing a coffee.

Then the robot looked at the user to wait for feedback. After receiving either positive or negative feedback, the robot confirmed by either thanking for positive feedback or confirming that it understood the negative feedback. As the robot could not actually understand the given feedback, it responded according to the expected feedback depending on whether it had performed correctly or incorrectly. The expectation almost always agreed with the actual feedback given by the user. After that, the next scene was shown on the screen, so that the user could continue teaching the next command.

When executing any of the commands, the robot performed a specific gesture. The gestures were selected so that they could be implemented in a similar way on the four-legged pet-robot and the humanoid.

Sample virtual living room scenes to prompt the user to give a command to the robot are shown in Fig. 3.12. In the first scene, the robot asks the user to name the "audioplayer" object. In the second scene, the user is expected to tell the robot to switch the light on. In the third scene, the user is expected to make the robot switch off the television. The white texts in the images show the internal representation of commands and were not shown to the user during the training task.

3.5.2 Assumptions

Based on the schema theory [14] in psychology, which suggests that people use schemata of familiar objects and situations to understand and handle unfamiliar situations, it was assumed that users are likely to interact with a pet-robot in a similar way as with a

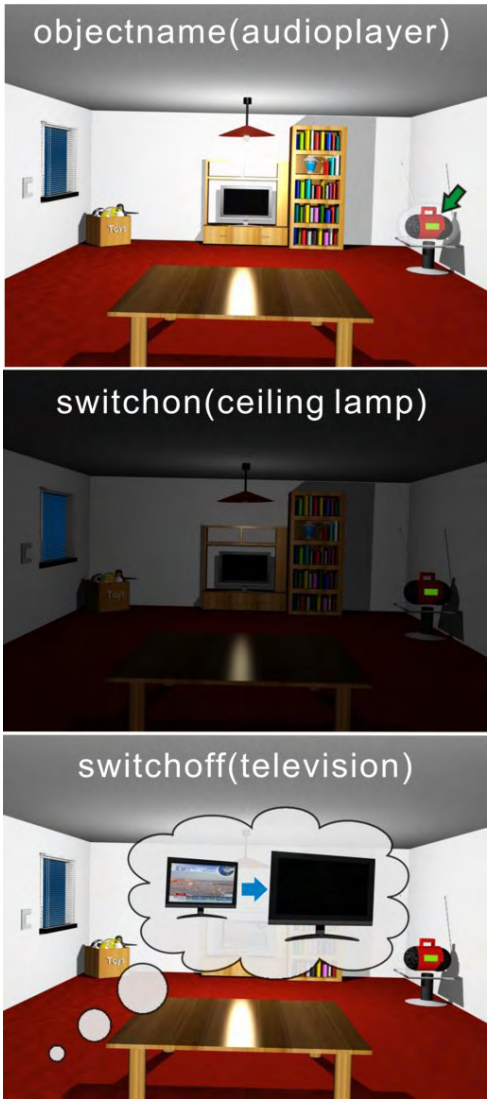

Figure 3.12: Experiment 3: Sample scenes from the "virtual living room"

real dog, while the interaction with a humanoid was expected to resemble more to the interaction with a human.

Moreover, it was assumed that the participants were likely to conclude that the humanoid is more intelligent than the pet-robot, based on its humanlike appearance. This might lead to higher expectations and to adaptations such as a more elaborated speaking style, more politeness, more explanations etc. when interacting with the humanoid. Details on the expectations as well as the actually observed interaction are given in the results section.

3.5.3 Experimental setting

Sixteen participants aged from 22 to 52 participated in the user study. Ten participants (7 males, 3 females) interacted with the humanoid and six participants (4 males, 2 females) interacted with the pet-robot for roughly 45 minutes. The language used in the experiments was Japanese. All participants were employees of the Honda Research Institute Japan. The instructions, given to the participants, as well as the questionnaire, the participants were asked to fill out after the experiment can be found in appendix 3.

Fig. 3.13 shows the experimental setting. The participants were asked to sit at a table in order to avoid excessive changes of position during the experiment. This was necessary because video data was recorded for later analysis and for gesture recognition. The robot was placed to the right of the participant, close enough that the user could easily reach it with its hand to touch it. As the pet-robot was a lot smaller than the humanoid, it was placed on the table, so that the participants could reach it easily.

The participants were equipped with a headset microphone to record audio data. Video data was recorded using a stereo camera which was placed above the screen.

The participants were given an explanation about the two training phases. In the first phase, they were asked to name the objects that the robot was pointing at. In the second phase, they were instructed to give commands to the robot and give positive feedback if the robot reacted correctly and negative feedback if the robot reacted incorrectly. They were instructed to give commands and feedback in any way they like by speech, gestures and touch. The participants had to teach each object name and each command ten times. As the duration of an experiment was relatively long and the users were required to talk a lot, there was five minute break between the learning of object names and the learning of commands.

3.5.4 Results

In the user study two different kinds of results have been obtained: The participants were asked to answer a questionnaire about their subjective impression of the interaction. Moreover the data, which was recorded during the interaction, was annotated and analyzed

Figure 3.13: Experiment 3: Experimental setting.

to find objective similarities and differences in the participants' behavior. The t-test was used to determine the statistical significance of the observed differences.

Questionnaire results

The results of the questionnaire, which are listed in table 3.4, show a slight tendency towards more positive ratings for the interaction with the pet-robot. However, none of the differences is statistically significant. There are multiple possible interpretations of the questionnaire results. As no statistical significance has been found, there actually may be no difference at all between the impressions of the participants who interacted with the pet-robot and who interacted with the humanoid.

Another possible explanation for the slightly better results for AIBO may be higher expectations for the humanoid robot, that are more difficult to satisfy. When the expectations toward a humanoid are higher than toward a pet-robot, the same behavior is perceived less satisfactory for a humanoid than for a pet-robot. As the robot was set to "learn" relatively slowly, in order to gather enough training data for the command learning algorithm, the higher expectations may have resulted in stronger dissatisfaction when interacting with the humanoid robot.

As the study was conducted with participants from Honda Research Institute, which did not all have previous experience in interacting with the humanoid robot but were more

Table 3.4: Experiment 3: Users' evaluation of the training task

Question (5: fully agree - 1: do not agree)	Humanoid	Pet-robot
I enjoyed teaching the robot through the given task	3.5 (0.8)	4 (0.8)
The robot understood my feedback	3.6 (0.9)	4.3 (1.1)
The robot learned through my feedback	3.2 (1.3)	4.3 (0.5)
The robot adapted to my way of teaching	3.2 (1.1)	3.8 (1.3)
I was able to instruct the robot in a natural way	3.6 (1.1)	3.5 (1.5)
The robot took too much time to learn	3.6 (1.4)	2.7 (0.9)
The robot is intelligent	2.7 (1.3)	2.8 (1.5)
The robot behaves autonomously	2.7 (1.4)	2.8 (0.9)
The robot behaves cooperatively	3.7 (0.8)	3.3 (0.7)

familiar with it, than they were with AIBO, the slight differences in the evaluation may also be attributed to the novelty effect.

User behavior

Different aspects of the users' commands and feedback, that were assumed to be related to the perceived intelligence and human-likeness of the robot, were analyzed. The analysis compared the speaking speed (in seconds per word) and the number of words per command/feedback, as it could be assumed that people talk slower and in simpler sentences, when they consider the robot less intelligent. However, it was found, that the length of commands was almost the same for both robots. An average command for the humanoid was 3.75 (sd=0.42) words long, while an average command for the pet-robot was 3.72 (sd=0.71) words long. The speaking speed was also similar for the pet-robot with 0.45 (sd=0.09) seconds per word, and the humanoid with 0.42 (sd=0.07) seconds per word. This is in line with the participants' subjective evaluation of the robots' intelligence, shown in Table 3.4.

Multimodality

During the interaction with both robots, no pointing gestures from any of the users were observed. A possible explanation is that all objects were very easy to distinguish verbally, so that pointing gestures would have been redundant. Moreover, the robot took the active role when asking for object names and pointed at the objects it wanted to learn. Touch-based rewards were only observed for one out of ten participants for the humanoid but for five out of the six participants who interacted with the pet-robot. As touch is frequently

Table 3.5: Experiment 3: Types of commands used in the interaction with the humanoid and the pet-robot (All values in percent, value in brackets is the standard deviation)

Type	Humanoid	Pet-Robot
Plain commands	75.01 (14.00)	60.83 (41.04)
Polite commands	9.86 (10.88)	26.23 (41.99)
Questions in commands	10.23 (3.51)	8.34 (6.73)
Implicit commands	3.40 (4.82)	4.10 (7.23)
Parameters left out	6.78 (2.25)	4.13 (4.77)
Explanations in commands	1.81 (3.90)	0.95 (2.32)

used with real dogs, it can be assumed that users considered touch to be appropriate for giving feedback to a pet-robot because of its dog-like appearance.

Verbal commands

The study also analyzed, how many commands had explanations or polite expressions and how many commands were phrased as a question. This was based on the estimation that users might be more polite, explain more and use more questions when talking to a humanoid robot, while they rather give plain commands to a dog-like robot. Commands that contain words like "kudasai", "kureru?", "moraeru?" etc., which are similar to the English word "please" were considered as polite commands. Moreover, it was analyzed, how many commands were implicit ones like saying "it is too dark here" to make the robot switch the light on, and in how many commands some expected parameters were left out like in "put away the toy car" instead of "put the toy car into the box", because this kind of verbal behavior might be related to the perceived intelligence of the robot.

The results can be found in Table 3.5. The values do not add up to 100% because not all types of commands are mutually exclusive (e.g. a polite command can have parameters left out). Quite different utterances were observed for different users and some of the findings actually contradicted the assumptions, such as more polite commands given to the pet-robot than to the humanoid. However, the differences seemed to be rather caused by personal preferences, than by the appearance of the robots. This assumption is supported by the high standard deviations between users. None of the observed differences was statistically significant. In case of polite expressions, it was found, that most users either used polite expressions for commands all the time or not at all. Therefore, the difference in the amount of polite commands for the pet-robot and the humanoid robot is most probably user dependent, rather than caused by the robot.

Verbal positive and negative feedback

Three different types of feedback were distinguished: Personal rewards like "Thank you" or "I like that better, now", which emphasize, that the robot has done something for the user and are given from the user's perspective, feedback which directly comments on the performance of the robot, like "Well done." or "That was wrong." and explanations used as rewards like "That is not a toy car, it is a ball." or "That is a toy car.". The usage of different rewards for the humanoid and the pet-robot is shown in table 3.6.

Table 3.6: Types of feedback used in the interaction with the humanoid and the pet-robot (All values in percent, value in brackets is the standard deviation)

Type	Humanoid	Pet-robot
Personal	52.78 (17.99)	24.83 (27.41)
Performance evaluation	38.39 (18.28)	70.02 (28.16)
Explanations	11.10 (14.29)	3.56 (3.90)

Statistically significant differences were found for the usage of personal rewards (df=14, t=2.48, p=0.026) and rewards, which comment on the robots' performance (df=14, t=2.75, p=0.016). While the users usually gave feedback like "well done (yoku dekimashita)" or "good (ii yo)" to the pet-robot, they used more personal rewards like "Thank you (arigatou)" for the humanoid, especially for positive reward. Fig. 3.14 shows the differences in user feedback, given to the humanoid and the pet-robot. While the participants gave more explanations when talking to the humanoid, especially for negative rewards, the difference between both robots was not significant.

Behavior changes over time

The experiments also investigated the change in user behavior over time by comparing the commands and feedback, the participants gave in the first five minutes of the command learning phase to the commands to the feedback given throughout the whole experiment and to the last five minutes of the experiment.

There were no significant changes in commands given to both robots over time and in the feedback given to the pet-robot. However, two marginally significant changes in the feedback given to the humanoid could be observed: The amount of explanations for negative feedback was marginally significantly lower (p=0.071, t=2.06, df=9) at the beginning of the experiments than it was throughout the whole experiment. While at the beginning of the experiment only 26.98% (sd=32.32%) of the negative feedback for the humanoid contained

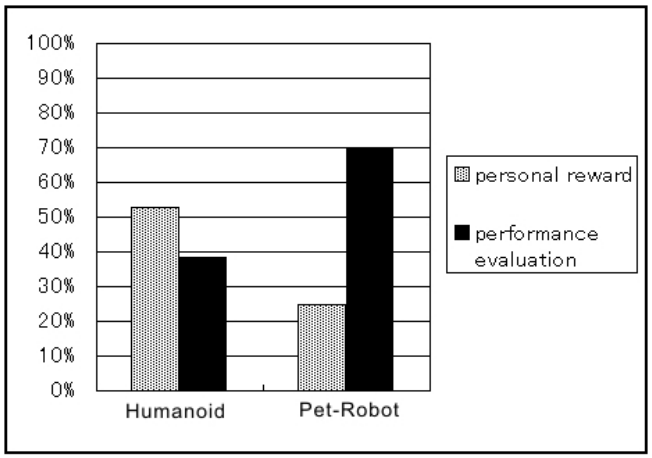

Figure 3.14: Experiment 3: Difference in feedback for the humanoid and the pet-robot.

an explanation, it was an average of 34.57% (sd=35.87%) during the whole experiment and went up to 75.00% (sd=35.36%) at the end of the experiment.

A marginally significant increase (p=0.091, t=1.90, df=9) in personal feedback comparing the first five minutes of the command learning to the whole command learning phase was found for the interaction with the humanoid. Overall the percentage of personal feedback increased from 34.85% (sd=22.62%) in the first five minutes to 61.92% (sd=24.60%) in the last five minutes, while the average was 52.78% (sd=17.99%).

Similar trends toward more personal feedback and more explanations for negative rewards were also found for the pet-robot. However, the statistical significance of these trends could not be confirmed. Fig. 3.15 compares the feedback given during the whole task to the feedback given during the first five and last five minutes of the task. For the interaction with the pet-robot no more explanations could be observed within the last five minutes of the training. This is because the analysis considered explanations accompanying negative feedback when the robot made mistakes. Due to the simulated learning, the robot stopped making mistakes towards the end of the training. As the amount of explanations was generally lower for the pet-robot, there was no negative feedback with explanations in the last five minutes of the experiments with the pet-robot.

3.5.5 Discussion

In the experiments, there were less than expected differences in users' behavior toward the pet-robot and the humanoid. While especially the way of uttering commands seems to depend rather on the personal preferences of the user, than on the appearance of the robot, robot-dependent differences were found concerning the feedback, given by the participants.

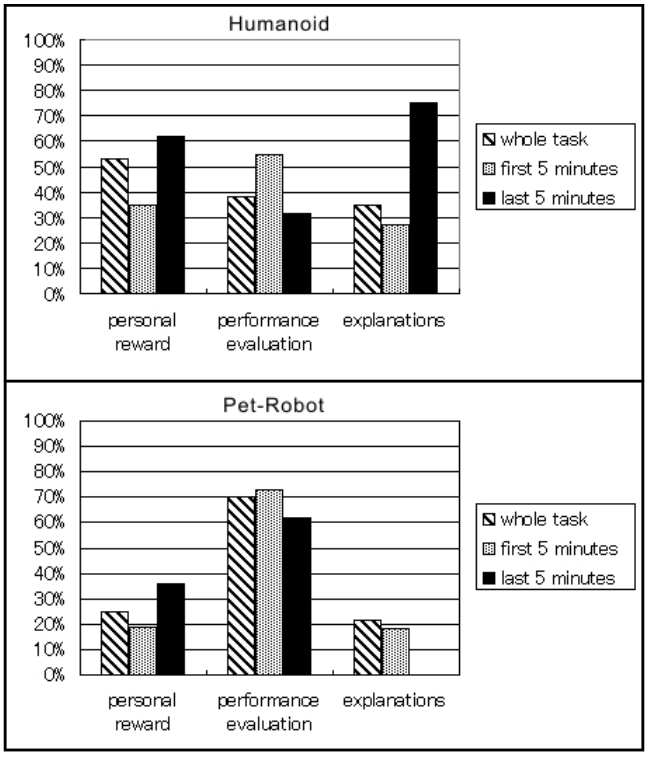

Figure 3.15: Experiment 3: User feedback changes over time.

The most obvious one was the frequent use of touch for giving feedback to the pet-robot, while touch was almost not used for the humanoid. Moreover, users tended to give personal feedback like "Thank you" to the humanoid, while they rather commented on the performance for giving feedback to the pet-robot. These findings suggest that people actually use their experience with real dogs as a guideline when giving feedback to the pet-robot.

When interacting longer with the humanoid, people started to give more explanations when the robot performed incorrectly and also gave more personal reward. While the results are only marginally significant and hard to interpret, one explanation may be, that the perception of the humanoid robot as an intelligent interaction partner increases when the robot shows learning capabilities and improves its performance during the experiment. Similar tendencies could be observed with the pet-robot. However, these tendencies were not statistically significant.

The users' subjective evaluation did not reveal significant differences between the humanoid and the pet-robot. As both robots were programmed to behave in the same way on the same task, it can be assumed that the users' impression of the robot's behavior on the given task depends rather on its actual performance than on its appearance.

There are different possible explanations, why no significant differences were observed for giving commands. One of them is that both robots used speech to communicate with the user. As speech is a typical human modality of interacting, differences might have been stronger, if the pet-robot had communicated with the user in a more dog-like non-verbal way. As there was no significant difference in users' evaluation of both robots' intelligence, users may have considered similar types of commands acceptable for both robots.

Further experiments would be necessary to confirm whether the trend that was found in the experiments with one particular humanoid and one particular pet-robot and a special training task can actually be generalized to other types of humanoids or pet-like robots and to more general tasks. As discussed in chapter 2, previous literature suggests, that depending on their appearance, user-behavior can vary for different types of humanoid robots and presumably the same is true for different pet-robots.

3.6 Conclusions from the experiments

Summarizing the findings from the three experimental studies, it can be concluded, that it is useful to enable a robot to understand its user's natural way of giving commands and feedback instead of restricting the user to predefined commands or other means of interaction and having him or her learn from a handbook, how to use the robot. As feedback and commands differ between users and also between different types of robots, adaptation to the concrete interaction situation between a user and a robot can be expected to improve the smoothness of the interaction and make the user feel comfortable about interacting with the robot.

3.6.1 Feasibility and usefulness of the proposed learning method

As the experiments confirmed, that feedback and commands do not vary excessively for a single user, learning them appears feasible, as long as there is only a limited number of commands and objects to be learned. If the number of commands and object grows large, the training should be conducted in multiple training sessions to avoid boring or stressing the user.

Feedback varies strongly between different users while feedback from one user is similar between different tasks. However, users do not use the same feedback all the time but have a limited number of utterances to express positive and negative feedback. The learning method needs to deal with this and allow multiple ways of expressing the same command or feedback. If this condition is fulfilled, learning feedback and commands through the proposed training tasks is feasible.

The findings from the first experiment suggest, that users react relatively sensitive to restrictions in allowed feedback behavior. If allowed feedback was restricted, then the

overall amount of feedback decreased. Moreover, it was difficult for users to actually stick to a designated feedback behavior while otherwise interacting naturally with the robot. Learning to understand natural feedback behavior from the user is therefore expected to increase the frequency as well as the quality of user feedback and make the user feel more comfortable with the teaching situation.

3.6.2 Design of the training tasks

The first experiment showed that training tasks, used for the experiments, in which the robot actually had to move around in its environment and had to pick up objects, resulted in relatively few feedback utterances throughout the course of one experiment, because the participants had to spend most of the time waiting for the robot to complete its tasks. Therefore training tasks, which do not require the robot to actually move around, are desirable to maximize the effectiveness of the training and minimize waiting times.

Training tasks need to be designed in a way that the user is sure which command, object name or feedback he or she should give next. As seen in the "Connect Four" task, the user's subjective uncertainness about his or her evaluation of the current state of the task results in less contingent and reliable reward being given by the user.

The results from the third experiment, which found changes in the amount of explanations and reward types toward the end of the experiment, suggest that the training must not be too short. As user behavior can change over time, when the user gets more acquainted with the robot, the training needs to be long enough to cover these possible changes in user behavior.

3.6.3 Multimodality

The results of the second experiment, which used virtual tasks for learning feedback, suggest that for learning and understanding users' feedback, emphasis should be placed on the modalities speech and touch, while gestures do not play a major role in conveying positive or negative reward. The same observation was made in the third experiment on learning commands and reward with a pet-robot and a humanoid. For commands, the by far most important modality was speech. Neither touch nor prosody conveyed information, necessary for interpreting the commands given by the user.

In case of ambiguous utterances, such as calling the name of the robot, which can mean positive or negative feedback, depending on the tone of the user's voice, or repeating a command with a stricter tone when the robot made a mistake, the prosody of the utterances should be taken into account, because the utterances cannot be disambiguated using only their semantic contents. Learning the prosody, that is the sentence melody, loudness etc.

of a user's typical positive and negative feedback utterances can be expected to have a positive effect on the recognition accuracy, compared with speech and touch only.

3.6.4 Robot Appearance

We could not confirm any dependence between the way, in which users express commands, and the appearance of the robot. However, we found, that the appearance of the robot influences, how users give feedback to it. In all experiments with the pet-robot, touch was used a lot for giving feedback, while no touch feedback was given to the humanoid. The experiments also showed that utterances for giving feedback were different for both robots. While the users commented on the actual performance of the robot when giving feedback to the pet-robot, using utterances like "Well done" or "That was wrong", they rather gave personal feedback to the humanoid, commenting on the effect of the robot's action for the user, such as saying "Thank you". Learning feedback through a training task allows to adapt to this kind of robot-dependent differences automatically without having to take them into account at design time.

"The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' but 'That's funny ...' "

Isaac Asimov (1920 - 1992)

4

Learning to understand multimodal user feedback

4.1 Overview

This chapter discusses the basic algorithm for learning user feedback given through speech, prosody and touch. Extensions of the algorithm to enable the robot to understand parameterized commands will be presented in the following chapter.

First, an overview of the method is presented and the basic techniques, such as Hidden Markov Models and classical conditioning, that the learning method is based on, are described.

In the following paragraphs the proposed learning method is described in detail: The first part explains the necessary preprocessing steps to convert audio signals to a representation that HMMs can deal with. Then the two phases of stimulus encoding and conditioning are explained.

Parallels to naturally occurring processes in human speech acquisition and how these processes are implemented in the system as well as several extensions to the first version of the feedback learning algorithm, that were implemented into the system to improve the learning and recognition accuracy, are explained in the last part of this chapter.

4.2 Outline of the learning method

In order to enable a robot to learn to interpret natural user feedback, a biologically-inspired two-staged learning method is proposed in this thesis. It is modeled after the stimulus encoding and the association processes, which are assumed to occur in human learning [15][56] [86] of associations and word meanings. Details about the biological background of this work are given in section 2.2.

The proposed approach combines unsupervised training of Hidden Markov Models (HMMs) with an implementation of associative learning using a mathematical model of classical conditioning. The Hidden Markov Models model the stimulus encoding occurring in natural learning. They are trained in a way that each HMM clusters similar observed user feedback. Classical conditioning is then used to associate the trained HMMs with either approval or disapproval and integrate the available information perceived through multiple modalities.

The combination of supervised and unsupervised learning as well as the special training tasks, described in chapter 3, allows the system to learn to understand and adapt to a user's natural way of giving feedback to a robot without requiring any transcriptions of training utterances and without any prior knowledge on the words, language or grammar to be used.

The robot was trained using the game-based training tasks described in section 3.4. As the tasks were designed in such a way, that they allow the robot to anticipate the feedback that the user is going to give, one input for the training algorithm corresponding to one feedback situation consists of three items:

- The symbol POSITIVE or NEGATIVE depending on whether positive or negative feedback was expected
- A sequence of speech data containing one or multiple rewards to be used for speech and prosody classification
- A sequence of data recorded from the touch sensors containing one or multiple rewards for touch classification

The training task makes it easy to distinguish between instructions and feedback. Instructions are only expected, while the robot looks at the screen before its next move. Feedback is expected after the robot made a move. The time between a good or bad move of the robot and its response to a feedback is considered as one feedback situation. All observations that are made during this time are combined and used for the training. The robot responds to the given feedback five seconds after either the voice activity detection or the touch sensors has detected the last activity. If no feedback is observed, the robot waits 15 seconds until proceeding with the next move. These values were selected based on previous experiments, in such a way, that the robot was unlikely to interrupt the user's feedback.

The term "multimodal feedback" is used hereafter to refer to a time sequence of observations that possesses the following properties:

- It consists of perceptions in one or more modalities
- It begins by an increase in activity in one of the modalities (e.g. voice onset)
- It ends by a period of inactivity in all modalities

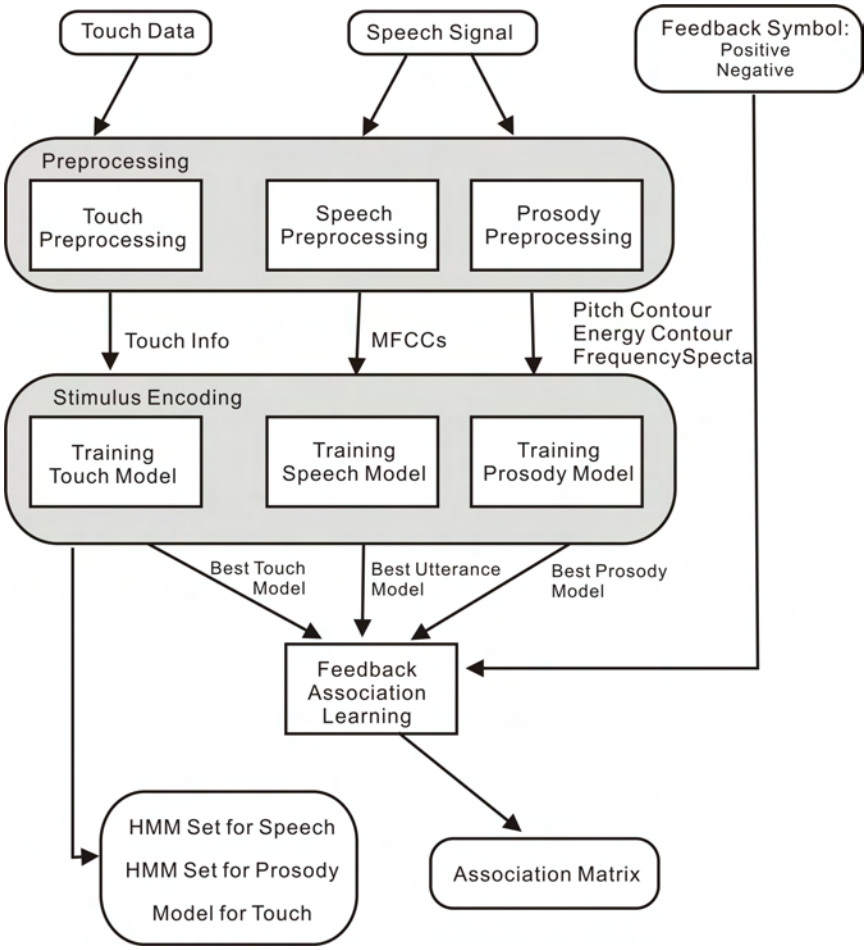

Figure 4.1: Overview of the learning method

- Actions in different modalities occur in close timely relation, that is, at the same time or in a sequence, quickly following one another.
- The perceptions follow a behavior of the robot that can be clearly attributed a positive or negative value

Figure 4.1 shows an overview of the proposed learning method. First the speech and touch signals are **preprocessed** to extract relevant features used for speech and prosody detection.

Based on the sequences of feature vectors, output by the preprocessing, the **stimulus encoding** stage trains a set of HMMs for both, speech and prosody. The models are trained in an unsupervised way and cluster similar perceptions, e.g. utterances that are likely to contain the same sequence of words or similar prosody. Touch is handled using a simple

duration based model, as the data returned by the AIBO remote framework does not suffice for HMM based modeling.

The **associative learning** stage is based on an implementation of classical conditioning. It associates the HMMs, trained in the first stage, with either approval or disapproval, integrating the data from different modalities. As users have different preferences for using speech, prosody and touch when communicating with a robot, the system has to weight the information, coming in through these different channels depending on the user's preferences. Classical conditioning can deal with this problem by emphasizing cues that frequently occur in connection with approving or disapproving feedback for a certain user. It allows the system to weight and combine user inputs in different modalities according to the strength of their association toward approving or disapproving feedback.

The information whether a certain HMM has to be associated with approval or disapproval is retrieved from the state of the training task. In the games, used for training, each move of the robot is either a good or a bad one. As the games were designed in a way that the participants could easily evaluate the moves, the system uses the heuristic that positive feedback is given for good moves while negative feedback is given for bad moves. Knowing whether the robot made a good move or bad move before getting a certain feedback, the system reinforces the association between the model of the observed utterance, touch or prosody pattern and either positive or negative feedback.

The output of the learning process are **HMM sets for speech and prosody as well as a classifier for touch** and an **association matrix** which stores the associative strengths between the different models and their meanings.

As a model of the top-down processes, which occur in human learning, the associations learned in the conditioning stage are used to integrate context information when selecting the best HMM for retraining. This is done by adding a bias on models, that are already associated with approval or disapproval depending on what feedback is expected based on the state of the training task. Based on this bias, models that are already associated with a certain type of feedback get more likely to be selected when the same feedback is expected again.

The audio and touch data, which was recorded in the experiment described in section 3.4 was used for the offline training and evaluation of a model of the user's preferred way of giving feedback. During the experiment, a log file was written, which contained all the moves, the robot made and an evaluation of the move as positive or negative. Based on the audio and touch data and the log file, the system created the mappings, as described above, between the expected feedback and the utterances and touch patterns produced by the user, which were used as an input to the learning algorithm.

As sample data structure, created during the learning process, is shown in Fig. 4.2. At the bottom, the HMMs as well as the touch models are shown, which are trained during the stimulus encoding phase. The symbolic representations of positive and negative feedback

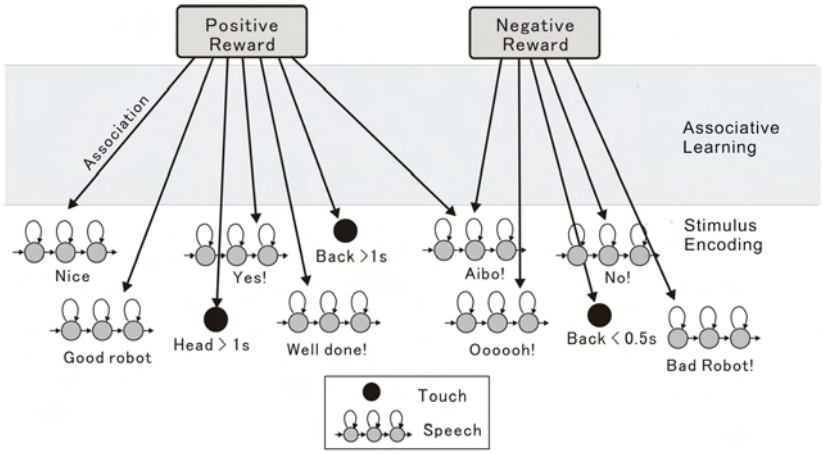

Figure 4.2: Data structure created by the learning method

are shown at the top. The connections between the symbols and the HMMs are learned during the associative learning phase.

4.3 Feedback modalities

Based on the findings from the studies on users' feedback behavior as well as literature research, the learning algorithm was designed to integrate multimodal information from the phonetic contents of speech utterances, prosody of speech utterances, as well as touch to recognize user feedback.

The implemented system also records video data and can track objects for gesture recognition. However, the experiments showed that the users did not employ gestures for giving feedback and commands to the robot in the given tasks. Therefore gestures were not used as a modality in the recognition algorithm.

Feedback given through **speech utterances** varies a lot between different users and also varies for one user. This was investigated in the experiments, described in section 3.4. An average user employs multiple different utterances for positive as well as for negative feedback.

Touch feedback, given by the users, could be classified into hitting and caressing the robot. Hitting the robot corresponded to short touches while caressing the robot corresponded to longer touches. Moreover, a relatively high number of participants only used touch for positive feedback.

As for **prosody**, previous studies by various researchers [13] [59] showed that certain prosodic features of an audio signal are correlated with positive or negative emotional valence of a speech utterance. The most important features are the pitch and the energy of the signal as well as the energy distribution in the frequency spectrum. Positive feedback is expected to have a high variability in the pitch and energy contours and a relatively high average pitch while negative feedback is correlated with low pitch and a low pitch variability and typically higher average energy but lower energy variability.

Based on these findings, appropriate features were selected for encoding speech, prosody and touch stimuli. For speech, standard MFCC feature vectors are used, which are common in speech recognition. For prosody the system uses feature vectors based on features calculated from the energy, pitch and frequency spectrum of the utterances. Touch is classified based on its duration and on the location of the touch sensor, touched by the user.

4.4 Basic techniques

The proposed learning method uses Hidden Markov Models as well as an implementation of the Rescorla-Wagner model of classical conditioning. This section gives a brief description of the basic ideas behind Hidden Markov Models and the development of mathematical models of classical conditioning. It also presents the main algorithms used in training and recognition with Hidden Markov Models and learning by classical conditioning.

4.4.1 Hidden Markov Models

Hidden Markov Models (HMMs) [89][66] are statistical models which are widely used in speech recognition and in various other fields, such as gesture recognition, handwriting recognition and genome analysis.

A HMM is essentially a Markov chain in which the current state is not observable. However, while the state of the HMM itself is not visible, a sequence of outputs $O = o_1, o_2, ..., o_n$ from the HMM can be observed. A HMM is defined by a number of states $X = 1...n$, transition probabilities $A = a_1, a_2, ...a_n$ between the different states, as well as output probabilities $B = b_1, b_2, ...b_n$. In every state i, the model produces a certain output vector o_i based on the probability density b_i. Fig 4.3 shows a HMM along with its transition probabilities and output probabilities.

The probability of an output sequence $o_1, ...o_6$ being produced by traversing a state sequence X in the model M, shown in Fig. 4.3, can be calculated as in equation 4.1.

$$P(O, X|M) = a_{12}b_2(o1)a_{22}b_2(o2)a_{23}b_3(o3)... \quad (4.1)$$

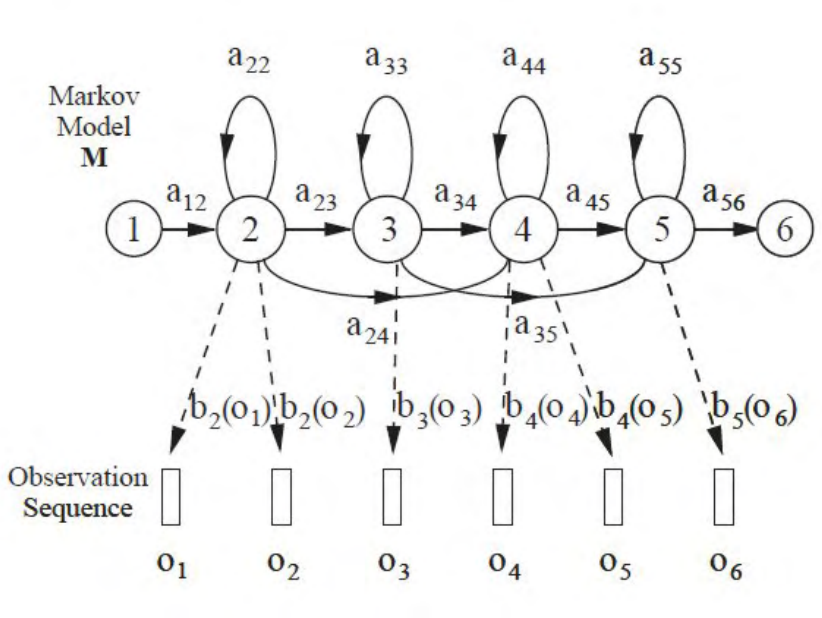

Figure 4.3: A Hidden Markov Model. (from HTKBook [89])

When HMMs are used for speech recognition, one HMM models a speech unit. This can be a word, a whole utterance but also a phoneme, diphone, triphone or any other unit deemed suitable for the task at hand. For speech recognition, the observation vectors are usually vectors of Mel Frequency Cepstrum Coefficients (MFCC), calculated from the speech signal. Details are given in section 4.5.2. Different types of observation vectors can be used for other applications like gesture recognition or prosody recognition. For the implementation of prosody recognition, described in this thesis, custom output vectors are created from relevant features of the speech signal like pitch, energy and the frequency spectrum.

In order to recognize an utterance from a recorded speech signal, the system needs to find the most likely HMM or sequence of HMMs that would produce the observed sequence of speech vectors, so it needs to determine the model M with the highest probability $P(O|M)$ which can be calculated as in equation 4.2.

$$P(O|M) = \sum_{X} a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(o_t) a_{x(t)x(t+1)} \tag{4.2}$$

Instead of summing up the probability over all possible state sequences, this probability can be approximated as follows by only taking into account the most likely state sequence:

$$\hat{P}(O|M) = \max_{X} \left\{ a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(o_t) a_{x(t)x(t+1)} \right\} \tag{4.3}$$

However, both equations cannot be used directly for efficiently calculating the probability of an output sequence. A computationally efficient method to calculate the probability $P(O|M)$ is the Viterbi algorithm which is described later in this section. The Viterbi algorithm can be used to recognize the meaning of a sequence of observations using a set of trained HMMs. The Baum-Welch algorithm, which is used for training HMMs from training data is described in the following paragraphs.

Training with the Baum-Welch Algorithm

In order to use HMMs for speech recognition, they need to be trained using existing speech data. Usually this speech data is labeled. That means, each audio file also has a label which specifies in detail which word or phoneme was uttered at what time. The timing information can be omitted in case of embedded training of HMMs for continuous data, such as continuous speech.

The Baum-Welch algorithm estimates the transition probabilities A and the output probabilities B of an HMM M in a way, that the maximum likelihood of producing a set of observation sequences, corresponding to the training data, when running through M is optimized.

The algorithm is based on the idea to first calculate the probability of an observation sequence based on an initial HMM and record how often which transitions and output symbols of the HMM were used. The ones, that are used more often, get assigned a higher probability while the probability of those, which are used less, is decreased. By iterating this procedure until the change in probabilities falls below a threshold, the HMM is adapted to the training examples.

The output distribution of a state is described by equation 4.4.

$$b_j(o_t) = \frac{1}{\sqrt{(2\pi)^n \left|\Sigma_j\right|}} e^{-\frac{1}{2}(o_t-\mu_j)' \sum_j^{-1} (o_t-\mu_j)} \tag{4.4}$$

The parameters, that need to be estimated by the Baum-Welch algorithm are the means μ and variations Σ that specify the output distribution. They can be estimated as the following weighted averages:

$$\hat{\mu}_j = \frac{\sum_{t=1}^{T} L_j(t) o_t}{\sum_{t=1}^{T} L_j(t)} \tag{4.5}$$

$$\hat{\Sigma}_j = \frac{\sum_{t=1}^{T} L_j(t)(o_t \mu_j)(o_t - \mu_j)'}{\sum_{t=1}^{T} L_j(t)} \tag{4.6}$$

$L_j(t)$ denotes the likelihood of being in state j at time t which is calculated using the forward and backward procedures of the Baum-Welch algorithm.

The forward procedure

The Baum-Welch algorithm, which is also named Forward-Backward algorithm consists of a forward procedure and a backward procedure The forward procedure calculates the joint probability $\alpha_j(t)$ of being in state j of model M at time t after seeing the observations $o_1 ... o_t$:

$$\alpha_j(t) = P(o_1, ..., o_t, x(t) = j|M) \tag{4.7}$$

The Markov condition defines that in a Markov chain the probability of a transition from a state i to a state j does not depend on how the state i was reached. Therefore the probability $\alpha_j(t)$ can be calculated recursively based on the probability of being in one of the predecessor states of α_j at time $t-1$. This recursive calculation is called "forward procedure", because the probability is calculated recursively, starting at the beginning of the model and proceeding forward until reaching the end of the model.

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1)a_{ij}\right] b_j(o_t) \tag{4.8}$$

The initial conditions for the recursion for $1 < \mathrm{j} < \mathrm{N}$ are $\alpha_1(1) = 1$ because the calculation always starts in the first state of the HMM and $\alpha_j(1) = a_{1j}b_j(o_1)$ because the first state of the HMM does not emit an output symbol. Thus, the total likelihood of observing the state sequence O in the model M can be calculated as

$$P(O|M) = \alpha_N(T) = \sum_{i=2}^{N-1} \alpha_i(T)a_{iN} \tag{4.9}$$

The backward procedure

The backward procedure calculates the probability $\beta_j(t)$ of being in state j of model M at time t and then seeing the outputs $o_{t+1} ... o_T$ in a similar way as the forward procedure but using a recursion that starts at the end of the HMM and goes backward from the end to the beginning of the model.

$$\beta_j(t) = P(o_{t+1}, ..., o_T|x(t) = j, M) \tag{4.10}$$

In a similar way as the forward probability α, the backward probability β can be calculated as follows:

$$\beta_j(t) = \sum_{j=2}^{N-1} a_{ij}b_j(o_{t+1})\beta_j(t+1) \tag{4.11}$$

The initial conditions become $\beta_i(T) = \alpha_i N$ and $\beta_j(t) = \sum_{j=2}^{N-1} a_{ij} bj(o_1) b_j(1)$ with $1 < i < N$. The final condition can be calculated as

$$\beta_1(1) = \sum_{j=2}^{N-1} a_{1j} bj(o_1) \beta_j(1) \tag{4.12}$$

Using the results of the forward and backward processes, the probability of state occupation $L_j(t)$, given the observation sequence O and the model M, can be calculated as follows:

$$\alpha_j(t)\beta_j(t) = P(O, x(t) = j|M) \tag{4.13}$$

$$L_j(t) = P(x(t) = j|O, M) = \frac{P(O, x(t) = j|M)}{P(O|M)} = \frac{1}{P(O|M)} \alpha_j(t)\beta_j(t) \tag{4.14}$$

Based on this formula, the Baum-Welch algorithm can be used as follows to train a HMM from a set of training utterances for that HMM:

1. Calculate the forward and backward probabilities $\alpha_j(t)$ and $\beta_j(t)$ for all states j and times t using the recursions explained above.
2. Calculate the probability $L_j(t)$ for each state using $\alpha_j(t)$ and $\beta_j(t)$
3. Iterate through all observation vectors o_t to calculate the numerators and denominators used in equations 4.5 and 4.6 for estimating the means and variances of the output distributions of each state. After processing each observation vector, store the current values in memory for the next iteration.
4. Calculate the new parameter values for the HMM from the stored values.
5. Repeat the steps one to four, using the new parameter values, until the value $P = P(O|M)$ does not exceed the value at the previous iteration

Training HMMs for Continuous Speech Recognition The algorithm, described above, can only be used to estimate parameters if labels exist, which provide exact timing information about which HMM is used for which part of the data. For embedded training of models for continuous speech recognition, some adaptations are necessary.
When training models for continuous speech recognition from data without time labels, there is usually a bootstrapping phase in which a small part of the data is time-labeled and used to calculate initial distributions for the parameters of the HMMs.

The main difference between the isolated word recognition, which was described above, and the continuous case is that the system trains all HMMs in parallel while the HMMs for isolated word recognition can be trained one after the other. Moreover, the Baum-Welch algorithm for continuous speech only updates the parameters of the HMMs after all utterances have been used for training.

For continuous speech recognition the steps to estimate the parameters of a set of HMMs are as follows:

1. Iterate the following steps over all utterances:
2. Create a sequence of HMMs according to the label of the utterance by concatenating the HMMs corresponding to the label of the utterance.
3. Calculate forward and backward probabilities $\alpha_j(t)$ and $\beta_j(t)$ as above
4. Iterate over all observations o_t to calculate the state occupation based on forward and backward probabilities and update and store the numerator and denominator of the equations 4.5 and 4.6.
5. After processing all utterances use the stored data to update HMM parameters

Recognition with the Viterbi Algorithm

When an utterance is recognized with a set of Hidden Markov Models, the recognition algorithm tries to find the HMM sequence that is most likely to have produced the utterance in question. While the maximum likelihood formula, outlined earlier, can be used for recognizing isolated words, corresponding to only one HMM, the usual case is that an utterance is modeled by a sequence of HMMs. In order to recognize such an utterance, the most commonly used technique is the Viterbi algorithm, which generalizes easily from isolated word recognition to continuous speech recognition.

The Viterbi algorithm tries to find the most likely HMM sequence and the most likely path through it corresponding to an observation sequence o. As the number of possible paths can grow very large, the algorithm prunes paths, which have a likelihood below a defined threshold.

For a model M the maximum likelihood $\phi_j(t)$ of being in state j and having observed the observation vectors o_1 to o_t can be calculated for $1 < j < N$ according to the following recursion:

$$\phi_j(t) = \max_i \left\{ \phi_i(t-1) a_{ij} \right\} b_j(o_t) \tag{4.15}$$

The start conditions are:

$$\phi_1(1) = 1 \tag{4.16}$$

$$\phi_j(1) = a_{1j} b_j(o_1) \tag{4.17}$$

The maximum likelihood $\hat{P}(O|M)$ is then given by:

$$\phi_N(T) = \max_i \left\{ \phi_i(T) a_{iN} \right\} \tag{4.18}$$

To avoid the likelihoods becoming too small when long HMM sequences of HMMs are used, log likelihoods are used in the calculations:

$$\varphi_j(t) = \max_i \left\{ \varphi_i(t-1) + \log(a_{ij}) \right\} + \log(b_j(o_t)) \tag{4.19}$$

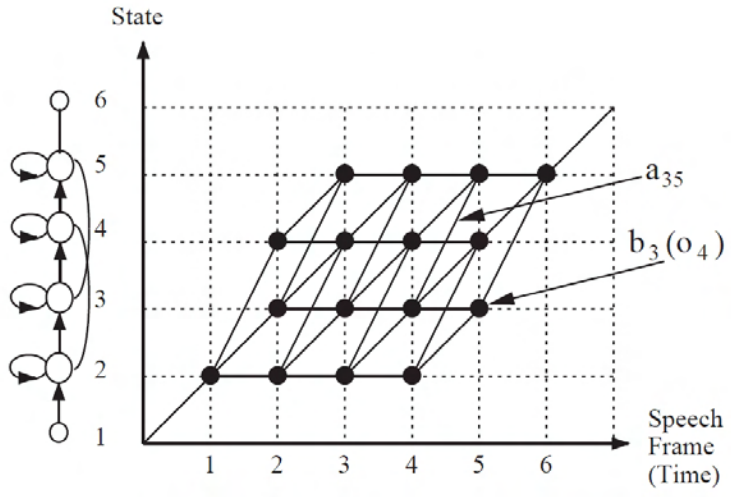

Figure 4.4: Visualization of the Viterbi algorithm. (from HTKBook [89])

The concept of the Viterbi algorithm is visualized in Figure 4.4. The dots symbolize the output probabilities while the arrows symbolize the transition probabilities of the HMM. Recognizing a sequence of observation vectors O can be understood as going through the network, starting at the lower left point and making a step to the adjacent points, summing up the probabilities, for every observation vector o_t. For continuous speech recognition some adaptations of the isolated word Viterbi algorithm are necessary in order to record the best path through the network of connected HMMs. The basic idea of the extension to continuous speech recognition is concatenating all possible HMMs and pruning the paths which get a too low probability.

4.4.2 Mathematical models of classical conditioning

In this work, a mathematical model of classical conditioning is employed to learn associations between speech, prosody and touch stimuli and their meanings. The theory of classical conditioning was first described by I. Pavlov and originates from behavioral research in animals. Details are given in section 2.2.2.

Since the 1950s, psychologists and mathematicians have tried to create mathematical models of learning processes in humans and animals. There are several mathematical theories, trying to model classical conditioning as well as the various effects that can be observed when training animals using the conditioning principle. The models describe how associations between unconditioned stimuli and conditioned stimuli are learned. One main topic of interest was creating a model of classical conditioning that can predict all the effects that occur when a human or animal learns through classical conditioning.

The first attempt at creating a mathematical model of classical conditioning was made 1950 by Estes, who described in his stimulus sampling theory [20], how associations can be predicted using a statistical model. The learning algorithm, described in this thesis, is based on the Rescorla-Wagner model, which was developed in 1972 and is described in detail in section 4.4.2. An overview over the various models of classical conditioning and the effects that can and cannot be explained by the different models can be found in [7].

Relevant features of classical conditioning

When humans or animals learn associations by classical conditioning, various effects can be observed that control learning and forgetting, interference of stimuli, etc. Chapter 2 provided an overview over the features of classical conditioning. This section gives some more details on how these properties actually affect the integration of multimodal data and the performance of the learning algorithm.

In the proposed learning method, the concepts of positive and negative feedback are modelled as unconditioned stimuli (US), which are known to the robot before starting the training. They get associated with the conditioned stimuli (CS), which are the trained models encoding a user's speech, prosody and touch stimuli.

For the task of learning multimodal feedback and command patterns, the most relevant properties of classical conditioning are blocking, extinction and second-order-conditioning as well as sensory preconditioning:

Blocking occurs, when a CS_1 is paired with a US, and then conditioning is performed for the CS_1 and a new CS_2 to the same US [7]. In this case, the existing association between the CS_1 and the US blocks the learning of the association between the CS_2 and the US as the CS_2 does not provide additional information to predict the occurrence of the US. The strength of the blocking is proportional to the strength of the existing association between the CS_1 and the US.

For the learning of multimodal interaction patterns, blocking is helpful, as it allows the system to emphasize the stimuli and modalities that are most relevant. If, for example, the user uses touch very reliably to distinguish positive and negative feedback - e.g. always touches AIBO's head sensor for positive reward and the back sensor for negative reward - then a strong association is learned quickly. If this reliable touch behavior is coupled with lots of different, maybe even unrelated speech utterances, the increasing associative strenghts for touch will more and more suppress the learning of the unnecessary speech utterances. The same would happen, if the user reliably says "yes" for positive feedback and "no" for negative feedback but gives some explanations after saying "yes" or "no". In that case, the increasing associations for the HMMs modeling "yes" or "no" would suppress the learning of associations for the HMMs modeling the varying, complicated explanations.

Extinction refers to the situation, where a CS, that has been associated with a US, is presented without the US. In that case, the association between the CS and the US is weakened. [7]

This capability is necessary to deal with changes in user behavior and mistakes, made during the training phase, such as misunderstandings of the situation resulting in incorrect feedback. When, for example, the user says "good", because he or she did not detect an error of the robot, the HMM modeling "good" is associated with negative feedback. However, when the HMM is observed multiple times later, but for positive instead of negative feedback, then the association to negative feedback gets weaker over time.

Sensory preconditioning and second-order conditioning describe the learning of an association between a CS_1 and a CS_2, so that if the CS_1 occurs together with the US, the association of the CS_2 towards the US is strengthened, too. [7] In sensory preconditioning, learning the association between CS_1 and CS_2 is established before learning the association towards the US, in second-order conditioning, the association between the US and CS_1 is learned beforehand, and the association between CS_1 and CS_2 is learned later.

Secondary preconditioning and second-order conditioning are important for the learning method, as they enable the system to learn connections between stimuli in different modalities. For example, a connection between the utterance "good" and touching the head sensor of the robot would be learned, if these stimuli regularly occurred together. While the effect of the sensory preconditioning is relatively minor during the training phase, where the meaning of utterances is given by the training task, it is interesting because it allows the learning to continue even after the training tasks have finished. If new utterances or other stimuli were paired with known utterances or stimuli during the actual use of the robot, secondary conditioning would enable the robot to associate the new stimuli with the same meanings as the trained stimuli.

The Rescorla-Wagner model

In this study, the Rescorla-Wagner model [7] is used for learning the associations between HMMs modeling speech utterances and the symbolic representations of feedback, object names and commands. The Rescorla-Wagner model was developed by Robert Rescorla and Allen Wagner in 1972 and most of the newer theories base on it. In the Rescorla-Wagner model, the change of associative strength $\Delta V_A(n)$ of the conditioned stimulus A to the unconditioned stimulus $US(n)$ present in trial n, is calculated as in (4.20).

$$\Delta V_A(n) = \alpha_A \beta_{US(n)} (\lambda_{US(n)} - V_{all(n)}) \tag{4.20}$$

α_A and $\beta_{US(n)}$ are the learning rates for the conditioned stimulus A and the unconditioned stimulus $US(n)$ respectively. $\lambda_{US(n)}$ is the maximum possible associative strength of the

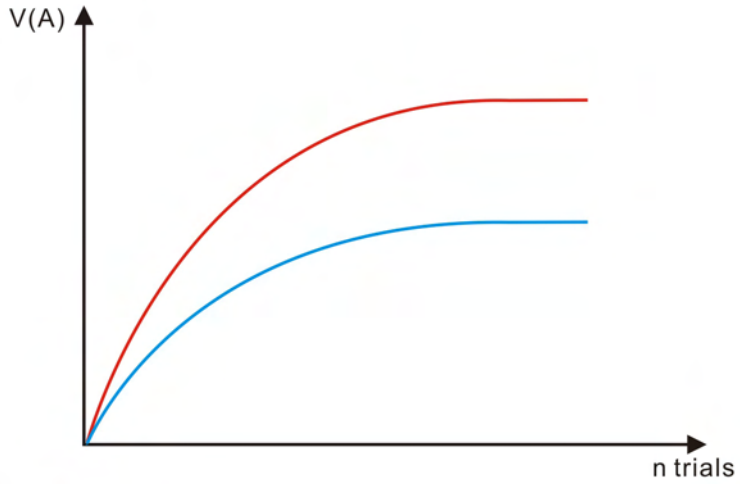

Figure 4.5: Learning Curves predicted by the Rescorla-Wagner model with two different learning rates α and β.

currently processed CS to the $US(n)$. $Vall(n)$ is the combined associative strength of all conditioned stimuli towards the currently processed unconditioned stimulus. The equation is updated on each occurrence of the unconditioned stimulus for all observed conditioned stimuli.

Modeling the effects occurring in classical conditioning with the Rescorla-Wagner model

Learning an association between two stimuli happens, when a CS is present just before the US occurs. In this case $\lambda_{US(n)}$ has a positive value. The actual value typically depends on the types of stimuli. For positive values of $\lambda_{US(n)}$, the change in associative strength $\Delta V_A(n)$ becomes positive, which means that the association between the US and the CS is strengthened. As the difference between $\lambda_{US(n)}$, the maximum possible associative strength for an US and $V_{all(n)}$, the already existing associative strength between the observed CS and the US, gets smaller with every iteration, the typical learning curve is generated, which is shown in Fig. 4.5.

Extinction occurs, when the CS is presented without the US. In this case, the maximum possible associative strength to the US, $\lambda_{US(n)}$ becomes zero, as the US is not present, and consequently $\Delta V_A(n)$ becomes negative, which leads to a decrease in the strength of the associative connection between the CS and the US. Thus, extinction of the previously learned association can be observed. The stronger the existing associative strength between the observed CS and a US the larger is the decrease when the CS occurs without the US.

Blocking is also modeled by the Rescorla-Wagner model. When there is an existing association between a CS_1 and an US and then a second CS_2 occurs together with the CS_1 before the US then learning the new connection is blocked. In this case the $V_{all(n)}$, that is the association of all currently observed CS to the US is already high because of the existing association between the CS_1 and the US, so the difference between $\lambda_{US(n)}$ and $V_{all(n)}$, which is used for learning CS_2 is smaller, than when the CS_2 were presented alone. Therefore learning is slower in this case. If the association between the CS_1 and the US is very strong, the learning of an additional CS_2, which always occurs together with the CS_1 and therefore does not provide additional information, can be almost completely blocked.

Limitations of the Rescorla Wagner model and overview over current theories

The Rescorla-Wagner model has a number of limitations, that is there are some processes which occur in classical conditioning in nature, but cannot be accurately predicted using the Rescorla-Wagner model.

The main limitation is, that the Rescorla-Wagner model is unable to correctly model secondary conditioning and sensory preconditioning. That means, connections among conditioned stimuli are not learned. The implementation, presented in this thesis, uses a second association matrix and a second pass of the Rescorla-Wagner model to model secondary conditioning and establish connections between simultaneously occurring multimodal stimuli. Details are explained in section 4.7.

Another limitation of the Rescorla-Wagner model is, that it is not able to model any time dependence of the conditioning process, so it does not matter, whether the conditioned stimulus and the unconditioned stimulus occur in very close temporal relation or whether they are further apart from each other. However, this is not relevant for the implementation of this learning algorithm, because only stimuli should be taken into account that occur as a response to and in very close temporal relation to one move of the robot and only these are passed to the learning algorithm.

Balkenius compared different advanced models of classical conditioning and their descriptive strengths and limitations in [7].

4.5 Preprocessing of the speech data

The first step of the implemented learning algorithm is the preprocessing of the speech data. Before the speech data can be used for training or recognition with HMMs, a number of preprocessing steps need to be performed. The preprocessing phase converts a continuous audio signal, containing speech and non-speech passages, into sequences of feature vectors. Each sequence corresponds to one speech utterance, which can be used to train HMMs.

A continuous speech signal first needs to be cut into a sequence of small, overlapping pieces, so-called frames. The proposed system uses a frame-length of 32 ms with an overlap of 16 ms. For every frame, the system needs to determine, whether it contains speech, or not and create frame sequences corresponding to utterances.

For each frame, that belongs to an utterance, relevant features are extracted and combined into a feature vector. The choice of appropriate feature vectors depends on the input data and the application. Appropriate feature-sets for representing speech and prosody are outlined in [89] [13],[55] and [59].

In the proposed system, Mel Frequency Cepstrum Coefficients (MFCC) are used for speech training and recognition. For prosody recognition a custom set of 14 features is calculated from the pitch-contour, energy-contour and frequency spectrum of the signal. The different steps involved in feature extraction and the feature-sets, that are used as an input for the HMM-based stimulus encoding, are described in sections 4.5.2 and 4.5.3.

4.5.1 Voice activity detection

In order to accurately and efficiently recognize speech utterances from a continuous stream of audio data the system first needs to determine, which of the frames of the signal actually contain speech and which frames only contain background noise. Frames that do not contain speech are considered irrelevant for the learning and recognition and can be discarded.

Therefore, the first step of the preprocessing phase is classifying, whether a frame contains speech, or not. This step is called voice activity detection (VAD). The approach for voice activity detection, that is used in this work, is described in [37].

The system uses two main cues to classify a frame of an audio signal into either speech or non-speech: The energy and the periodicity of the signal. While different methods for VAD exist, many of them are based on one or both of these features. An overview of different methods for voice activity detection is given in [30].

Energy

The energy of an audio signal is directly related to its observed loudness. It can be measured in decibel (DB). A frame, that contains a speech signal is typically louder than one which only contains background noise. This means, it has a higher energy. The energy of a whole speech signal or a frame of a speech signal can easily be calculated as:

$$E = \sum_{t=1}^{n} s(t)^2 \tag{4.21}$$

E is the energy of a signal s which contains n samples. If the energy is calculated for individual frames of the speech signal, the resulting value is called the short-time energy of a signal s in frame i. For a frame of N samples length and an offset o from the beginning of the signal the energy is calculated as follows:

$$E = \sum_{t=o}^{o+N} s(t)^2 \tag{4.22}$$

This equation is used for calculating the short-term energy of every frame for voice activity detection.

Periodicity

The periodicity of a speech frame mainly comes from the pitch of voiced speech. Noise as well as unvoiced speech usually has a low periodicity, while voiced speech, especially vowels, as well as music show a high periodicity. The more periodic a signal is, the more melodic it sounds. The less periodic a signal is, the more it sounds like noise. In order to estimate the periodicity of a frame of a speech signal the following approach is used:

In the Voice Activity Detector, the periodicity of the signal is calculated, based on the LSPE (Least Square Periodicity Estimation) technique, which was first described in 1992 by R. Tucker [82]. Every audio signal $s(t)$ can be separated into a periodic component $s_0(t)$ and a non-periodic component $n(t)$, so that it can be written as:

$$s(t) = s_0(t) + n(t) \tag{4.23}$$

for $t = 1, 2, ...N$, where N is the frame length. The method tries to determine the periodic component $s_0(t)$ in a way that it minimizes the mean squared difference between $s(t)$ and $s_0(t)$. The estimate $\hat{s}_0(t)$ can be calculated for $1 \leq t \leq P_0$ and $P_{min} \leq P_0 \leq P_{max}$. as:

$$\hat{s}_0(t) = \sum_{h=0}^{K_0} \frac{s(t + h\hat{P}_0)}{K_0} \tag{4.24}$$

P_{min} and P_{max} are the lower and upper boundaries for the period P_0. In [82] they are set to 2.75ms and 14 ms, which correspond to values between 71.43 Hz and 363.64 Hz, the typical pitch range in a human voice.

K_0 denotes the number of periods in a frame of N samples. It depends on the period length P_0 and the frame length N, and can be calculated as in equation 4.25.

$$K_0 = \frac{(N-t)}{P_0} + 1 \tag{4.25}$$

The algorithm tries to minimize the mean square error between the original signal and the estimate of the periodic component of the signal:

$$\sum_{t=1}^{N} [s(t) - \hat{s}_0(t)]^2 \tag{4.26}$$

The normalized periodicity measure R_1 was proposed by Friedmann in [24] and can be calculated as in equation 4.27.

$$R_1(\hat{P}_0) = \frac{I_0(\hat{P}_0) - I_1(\hat{P}_0)}{\sum_{t=1}^{N} s^2(t) - I_1(\hat{P}_0)} \tag{4.27}$$

With I_1 and I_0 defined as follows:

$$I_1(\hat{(}P)_0) = \sum_{t=1}^{P} \sum_{h=0}^{K_0} \frac{s(t + h\hat{P}_0)^2}{K_0} \tag{4.28}$$

$$I_0(\hat{P}_0) = \sum_{t=1}^{N} \hat{s}_0^2(t) = \sum_{t=1}^{\hat{P}_0} \frac{[\sum_{h=0}^{K_0} s(t + h\hat{P}_0]^2}{K_0} \tag{4.29}$$

By finding the maximum value for R_0 for period lengths $\hat{P}_0$ between the minimum period length P_{min} and the maximum period length P_{max} the system can estimate the periodicity and the main periodic component of the input signal.

The state machine

In order to determine, which frames of a signal contain speech or background noise, the VAD first calculates the energy and periodicity for each frame. A sample signal which contains only background noise is used to calculate initial energy and periodicity values. Based on these values a threshold is calculated to detect speech.

After initialization, the algorithm processes the signal frame by frame. It uses a simple state machine to decide whether a frame belongs to a non-speech passage, is a speech onset, is a frame within in a speech passage or is a speech ending. The state machine has four states and transitions between them, which are shown in Fig. 4.6:

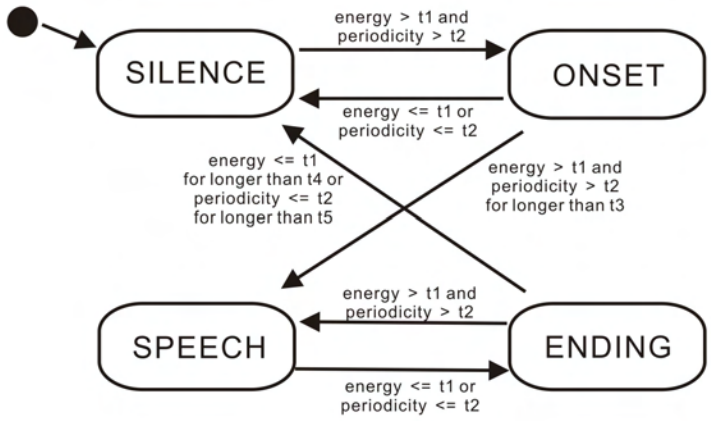

Figure 4.6: Statemachine for Voice Activity Detection. (t1 = energy threshold, t2 = periodicity threshold, t3 = minimum onset length, t4 = energy timeout, t5 = periodicity timeout)

- SILENCE: No speech is observed. The state machine starts in this state. When observing a new sample, the state machine stays in the state SILENCE, if either the energy or the periodicity does not exceed the computed thresholds. If both values exceed the thresholds it will go to the state ONSET.
- ONSET: A possible speech onset has been detected. From this state the state machine can go back to the state SILENCE, in case either the energy or periodicity thresholds have not been exceeded in a frame. This means that no actual speech was detected. It goes to the state SPEECH if both periodicity and energy thresholds have been exceeded for a given number of frames. If both energy and periodicity thresholds have been exceeded, but the set number of frames for detecting speech has not been reached, the system stays in the state ONSET.
- SPEECH: Speech is being observed. When observing a new frame, the system can either stay in the state SPEECH, if both energy and periodicity are above the threshold or go to the state ENDING if one of the values is below the threshold.
- ENDING: A possible speech ending has been found. If a timeout of 300 ms for energy or 500 ms for periodicity has occurred without detecting further speech, the system goes to the state SILENCE and returns the detected speech signal to be processed by the learning algorithm. If both thresholds are exceeded the system goes back to the state SPEECH. Otherwise it stays in the state ENDING to wait for either the timeouts to expire or new speech to occur.

4.5.2 Preprocessing for utterance recognition

The signal frames that have been classified by the voice activity detection as containing speech are passed to the subsequent preprocessing steps for utterance and prosody learning and recognition. To recognize utterances with HMMs they have to be transformed into a sequence of feature vectors. The proposed system uses feature vectors consisting of Mel-Frequency Cepstrum Coefficients as an input to the speech utterance based learning and recognition phases.

Extraction of Mel-Frequency Cepstrum Coefficients

The mel frequency scale is a psychophysiological scale to measure pitch, which is modeled after human aural perception. It has been developed in 1937 by Stevens et al. [76]. A tone that has twice the mel-value is perceived as twice as high. For tones up to 500 Hz the Mel scale corresponds to the frequency scale. That means 100 Hz correspond to 100 mel, 300 Hz correspond to 300 mel and so on. For tones that are higher than 500 Hz, the human perception of pitch increase and the actual increase in frequency diverge and the frequency intervals between tones that are perceived to have the same distance increase.

A frequency can be converted into a mel value as in equation 4.30 where m is the mel value and f a frequency in Hz.

$$m = 1124.01048 \log_e(1 + \frac{f}{700}) \tag{4.30}$$

Mel-Frequency Cepstrum Coefficients (MFCC) are based on the above described mel scale and represent the observed audio signal in a similar way as human aural perception. MFCCs are widely used in speech recognition applications because they emphasize the frequencies that differ depending on speech phonemes while suppressing the speaker-dependent influence of the speaker's vocal tract shape as well as the influence of prosody on the speech sounds.

To extract MFCCs from an audio signal, the signal is first transformed into the frequency domain by applying a Fourier transform. Then it is filtered using a special mel filter bank, consisting of 24 triangular shaped filters, which is shown in Fig. 4.7.

Next the mel filtered spectrum is converted into the cepstrum by first calculating the logarithm of each component of the filtered spectrum and then calculate the discrete cosine transformation of it. The result is called the "Mel Frequency Cepstrum" and each of the coefficients is called a "Mel Frequency Cepstrum Coefficient" and is used to construct the input vector to the speech learning and recognition.

In the proposed system, MFCCs are automatically extracted from the audio signals by HTK. [89].

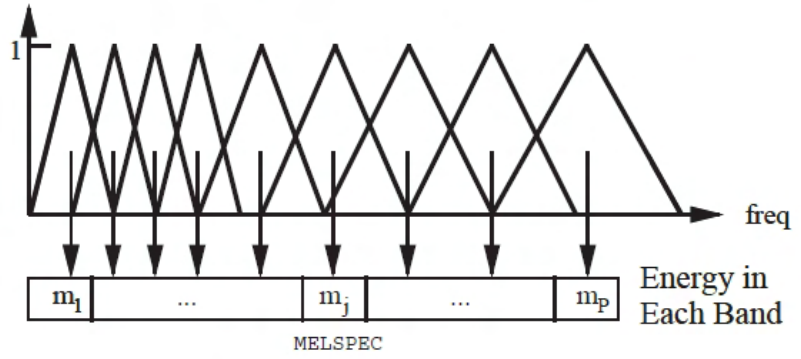

Figure 4.7: Filterbank for mel-filtering [89]

4.5.3 Preprocessing for prosody recognition

While MFCCs as a standard feature set are used for learning and recognizing the contents of speech utterances, a custom feature set is used for representing prosodic information. The system uses three different prosodic cues for learning and recognizing prosody, which have been described in literature [59] [13] as being related to the emotional valence of spoken utterances. These cues are the energy contour, pitch contour and the frequency spectrum of the utterances.

To capture this information, the system first cuts the signal into frames of 32 ms length with 16 ms overlap. The frame size must be selected in such a way that one frame is long enough to contain several pitch periods of the signal. On the other hand, a frame must not be too long, as a high resolution of pitch and energy contours is desirable. Moreover, the frame size must be selected small enough to avoid large variations in the length of fundamental periods within one frame, so that the signal can be assumed to be quasi-stationary within a frame. Typical frame sizes for speech processing are between 10 and 64 ms. To get smoother transitions between frames and increase the resolution of pitch and energy contours, frames typically overlap each other.

For each frame, the preprocessing phase extracts the short-time energy, pitch and short-time power spectrum. In the next step, feature vectors are calculated from this information, including the values of pitch, energy and the energy ratio in different frequency bands for the current frame as well as the changes in energy and pitch between the previous frame and the current frame.

Energy extraction

As explained in section 4.5.1, the energy of a frame corresponds to its perceived loudness. This is relevant for the prosody recognition, because the energy contour gives information on the positive or negative valence of a speech utterance. For example, an angry utterance is typically louder than an approving utterance. In the same way, as described in section 4.5.1, the energy of a frame of N samples length from a signal $s(t)$ with an offset of o samples from the beginning of the signal the energy is calculated as follows :

$$E = \sum_{t=o}^{o+N} s(t)^2 \tag{4.31}$$

The system calculates the energy as well as the difference in energy compared to the previous frame for each frame of the speech signal.

Pitch extraction

The most important cue to determine positive or negative valence of a speech signal is the pitch contour. The fundamental frequency (F_0) of a speech signal is determined by the resonant frequency of the vocal cords. It is the physical correlate of the perceived pitch of an audio signal. The fundamental frequency varies when a sentence is spoken. For example, it is raised at the end of a question while it falls at the end of a declarative sentence. The pitch also varies on syllable level. The relation between pitch and meaning or sentence types does, however, depend strongly on the language. For example, in tonal languages, pitch can actually change the meaning of a word. The variation of pitch in the course of a sentence is called pitch contour.

A sample of an audio signal and its pitch contour is shown in Fig. 4.8. Pitch only exists for voiced phonemes and it is visible in the power spectrum of a voiced frame as a peak in the range between 80 and 450 Hz, depending on the age and gender of the speaker.

In many cases the pitch can be seen with the bare eye when looking at the power spectrum of a frame of a speech signal. It is the first maximum in the spectrum while higher maxima typically mark the formants of the speech signal. However, determining the pitch only by peak-picking from the spectrum is rather unreliable.

To determine pitch, lots of methods have been proposed in literature over the years. Most algorithms for pitch tracking are based either on the cepstrum, LPC or autocorrelation of a signal. Each of them has its various advantages and disadvantages and a method that always yields optimal results is yet to be found, as explained in [75]. Comparisons of the different pitch tracking methods can be found in [23], [75] and [65]. The error rate of the different algorithms depends on the use-case as well as the speaker and the background noise.

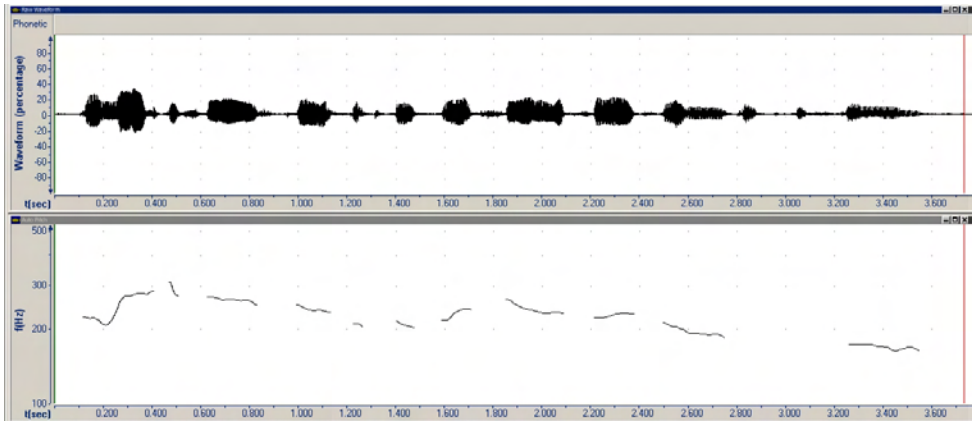

Figure 4.8: A speech signal and its pitch contour

In this work, the YIN algorithm [17] is used for extracting the pitch of an audio signal. The YIN algorithm is based on the autocorrelation idea, however it has various advantages:

- It is relatively easy to implement and well-documented in literature.
- It is robust against noise, which is important as it is used with data recorded by a moving robot.
- According to [17], the error rate of the YIN algorithm is lower than in most older methods.

The YIN algorithm

The YIN algorithm [17] was developed in 2002 and is based on the autocorrelation method. The autocorrelation method, which is not very reliable, when used alone has been extended and refined in six steps. According to [17] these steps lowered the average error rate on the authors' test database from 10% to 0.5% compared with the standard autocorrelation method.

The main difference between the autocorrelation method and the YIN algorithm is that the autocorrelation method tries to *maximize the product* of the original signal and its time-shifted version while the YIN algorithm tries to *minimize the difference* between the original signal and its time-shifted version.

The basic idea behind this method is, that if a perfectly periodic signal s is shifted against itself, then the difference between the signal and its shifted version is 0 when the shift distance is the same as the period T of the signal:

$$s_t - s_{t+T} = 0, \ \forall t \tag{4.32}$$

Therefore, the YIN algorithm can determine the most salient periodic component in a frame of the speech signal by minimizing the following difference function:

$$d_t(\tau) = \sum_{j=1}^{W} (s_j - s_{j+\tau})^2 \tag{4.33}$$

The main advantage of this approach over the autocorrelation is, that changes in amplitude are less likely to cause errors. If a signal gets louder over time, then the autocorrelation function might show higher peaks in later parts of the signal and therefore estimate a too low fundamental frequency. The difference function does not have this problem, because changes in amplitude always result in higher values of the difference function.

The next step is replacing the difference function by a cumulative mean normalized difference function. This means, that the original value of the difference function at the position τ is divided by the mean of the difference function for all possible time shifts smaller than τ.

$$d'(\tau) = \begin{cases} 1 & \text{for } \tau = 0; \\ \frac{d_t(\tau)}{\frac{1}{\tau}\sum_{j=1}^{\tau} d_t(j)} & \text{otherwise;} \end{cases} \tag{4.34}$$

Errors when determining pitch by searching the minimum of $d'(\tau)$ often come from picking later minima, which may be caused by the first formant and can be deeper than the minimum corresponding to the pitch frequency. A threshold is used to handle this problem. If a minimum is lower than the threshold then the search for later minima is stopped.

In the next step, the minimum, found in $d'(\tau)$, and its direct neighbors are interpolated by a parabolic function and the minimum of the parabolic function is used as the real minimum. This improves the results of the pitch extraction, if the sampling frequency, that was used when recording the signal, is not a multiple of the fundamental frequency. Without interpolation, errors of at most a half sampling period length are expected to occur.

The final step ensures that estimations of the fundamental frequency do not change too much from one frame to the next. When a large jump in pitch is detected, an error is likely, as the pitch in a natural speech signal typically only varies slightly from one frame to the next. The final step is based on the observation that erroneous estimations of the fundamental frequency are often accompanied by a high value of $d'_t(T_t)$ where T_t is the pitch period length determined by the algorithm. Slightly varying the parameter t often results in a lower minimum that corresponds to the actual fundamental frequency of the frame.

Therefore, for each t a minimum $d'_\theta(T_\theta)$ is searched in the interval $[\frac{t-T_{max}}{2}, \frac{t+T_{max}}{2}]$ where T_θ is the expected fundamental frequency at time θ and T_{max} the maximum expected period length. According to [17] a value of $T_{max} = 25ms$ has been used successfully.

The pitch, calculated by the YIN algorithm, as well as the pitch difference between the current frame and the previous frame are calculated for every frame of the speech signal and added as components in the feature vector for that frame.

Frequency spectrum

The frequency spectrum of a frame of an audio signal gives information about which frequency components a signal consists of. The lowest frequency in the spectrum is zero, the highest possible frequency is half the sampling rate of the audio signal, as the highest frequency, that can be represented in a sampled audio signal depends directly on the sampling rate. This frequency is called the Nyquist frequency. Fig. 4.9 shows a speech signal in the frequency domain.

To calculate the frequency spectrum of a frame, first a window function is applied to the samples in the frame. The purpose of windowing a frame is to suppress artifacts, resulting from the samples at the beginnings and ends of frames, by reducing the amplitudes of those samples. This phenomenon is called "spectral leakage". "Spectral leakage" results in additional maxima in the frequency spectrum, which are artifacts and do not have a correspondent in the actual signal.

Typical window types are Hamming-, Hanning- Blackman or rectangle windows [64]. All window types have various advantages and disadvantages and therefore have to be selected based on their purpose. In speech recognition the most frequently selected windowing functions are Hamming- and Hanning windows. In this work the Hamming window function is used, which can be calculated as in 4.35, where n are the samples and N is the frame length.

$$w(n) = 0.54 - 0.46\cos(\frac{2\pi n}{N}),\ -\frac{N}{2} \leq n \leq \frac{N}{2} \tag{4.35}$$

For each windowed frame the power spectrum is calculated. It can be determined from the sampled input signal with the help of a discrete fourier transform:

$$S_k = \sum_{t=0}^{N-1} s_t e^{\frac{-2\pi i t k}{N}} \tag{4.36}$$

S_k denotes the fourier coefficients of the signal s. N denotes the number of samples per frame. The power spectrum can be calculated from the fourier coefficients S_k by squaring them.

$$SL_k = |S_k|^2 \tag{4.37}$$

The effort for directly calculating the fourier transform using the DFT formula can be reduced by using a fast fourier transform (FFT). Calculating the FTT using the Cooley-Tukey-algorithm, which was used in this thesis, is a standard method that can be found in

literature, such as [25][64]. The FFT implementation used in this work is taken from the book "Numerical Recipes: The Art of Scientific Computing" [63].

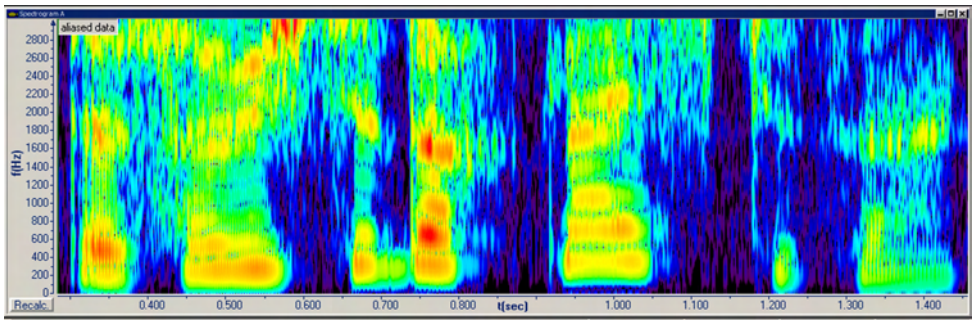

Figure 4.9: Speech signal in the frequency domain

From the power spectrum the system calculates the energy ratio in ten frequency bands, which are equally spaced between 0 Hz and 8000 Hz, that is the energy in each of the frequency bands normalized by the energy of the whole frame.

Creation of the prosody feature vector

Based on the data calculated from the energy, pitch and power spectrum the system generates feature vectors for prosody learning and recognition. As shown in Fig. 4.10, one feature vector consists of the pitch, the pitch difference to the previous frame, the energy, the energy difference to the previous frame as well as the energy ratio in the ten frequency bands as described above.

Pitch	Pitch Difference	Energy	Energy Difference	Energy Ratio FB1	...	Energy Ratio FB10

Figure 4.10: Prosody vector

The sequence of feature vectors for a speech utterance is written to a file in a HTK compatible format, which is described in section 6.3.2. Then it is used as input for the stimulus encoding stage of the algorithm.

4.6 Stimulus encoding

Based on the feature vectors, which were generated during the preprocessing phase, the stimulus encoding phase clusters the robot's perceptions of the user's feedback, creating a low-level model of the user's behavior. The stimulus encoding uses Hidden Markov Models for speech and prosody and a simple duration-based model for touch.

For each feedback, given by the user, the best matching speech, prosody and touch models are determined using the methods, described in 4.6.1 to 4.6.3 and new models are created, if necessary. Then, the best matching models are retrained with the data corresponding to the observed feedback and passed on to the feedback association learning stage where they are associated with either approval or disapproval based on the situation, that the robot was in, when perceiving the feedback.

4.6.1 Speech utterances

To model speech utterances, the system trains a user-dependent set of whole-utterance HMMs for the observed feedback. As the robot learns automatically through interaction, no transcription of the utterances is available. Therefore, unsupervised clustering of perceived feedbacks, that are likely to correspond to the same utterance, is necessary.

This is done by using two recognizers in parallel. **One recognizer tries to model the observed utterance as an arbitrary sequence of phonemes** using an existing set of monophone HMMs. **The other recognizer uses the already trained utterance models to calculate the best-matching known utterance**. It initially does not contain any HMMs and is filled with the trained models in the course of the training. When the system observes feedback from the user, it tries to recognize the utterance with both recognizers. Matching is done using HVite, an implementation of the Viterbi Algorithm, included in the Hidden Markov Model Toolkit (HTK) [89]. The recognizers return both, the best-matching phoneme sequence and the best-matching utterance out of the previously generated utterance models. A confidence level is output by the system for both recognition results.

The confidence levels, which are calculated by HVite as the log likelihoods per frame of both results, are compared to determine whether to generate a new model or retrain an existing one. Typically, for an unknown utterance, the phoneme-sequence based recognizer returns a result with a noticeably higher confidence, than the one of the best matching utterance model. For a known utterance, the confidence corresponding to the best-matching utterance model is either higher or similar to the best-matching phoneme-sequence. Therefore, if the confidence level of the best-fitting phoneme sequence is worse than the confidence level of the best-fitting utterance model or less than 10^{-5} better, then the best-fitting utterance model is retrained with the new utterance, assuming that the best-fitting model is actually a model of the observed utterance. If the confidence level of the best-matching phoneme sequence is more than 10^{-5} better than the one of the best-

fitting whole-utterance model, then a new utterance model is initialized for the utterance, assuming the utterance is unknown. The new model is created by concatenating the HMMs of the recognized most likely phoneme sequence. The new model is retrained with the observed utterance and added to the HMM-set of the whole-utterance recognizer. So it can be reused when a similar utterance is observed. The threshold of 10^{-5} was determined experimentally, using data that was recorded with the same audio equipment but not used for training or evaluation. An overview of the training for speech is shown in Figure 4.11.

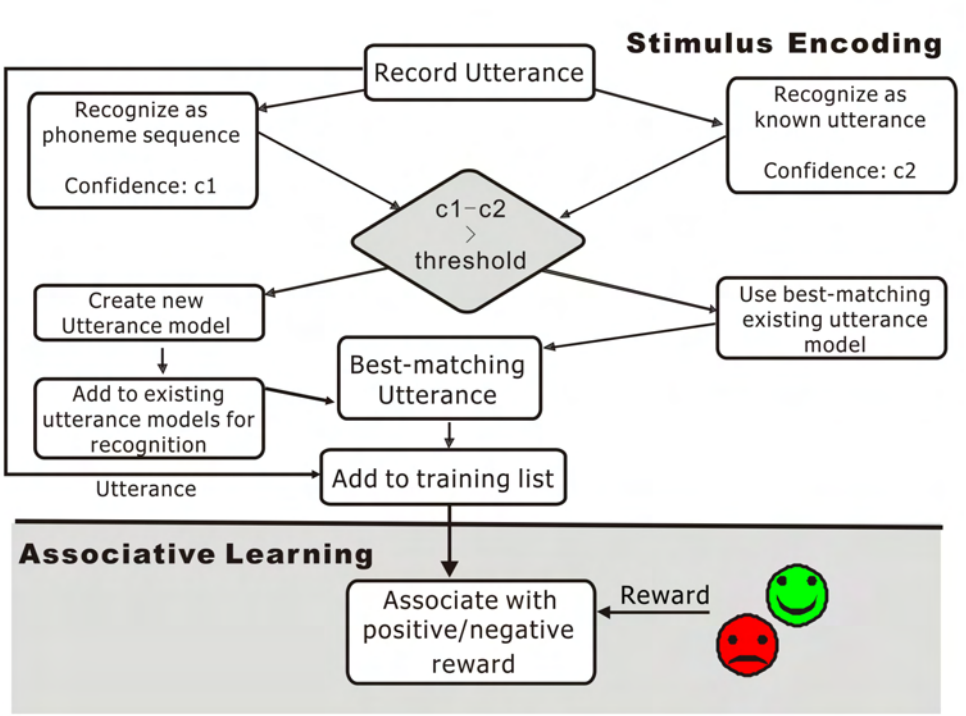

Figure 4.11: Algorithm for recognizing speech.

The phoneme-sequence recognizer

The HMM-set for the phoneme-sequence recognizer contains all Japanese monophones and is taken from the Julius Speech Recognition project [44].

The phoneme-sequence recognizer uses a simple grammar, which is shown in listing 4.1. It permits an arbitrary sequence of phonemes, not restricted by a language dependent

dictionary. A sequence of phonemes may have an optional beginning and ending silence and contain short pauses:

Listing 4.1: Grammar of the phoneme-sequence recognizer.

```
$phoneme=N|a|a:|b|by|ch|d|dy|e|e:|f|g|gy|h|hy|i|
i:|j|k|ky|m|my|n|ny|o|o:|p|py|q|r|ry|s|sh|sp|
t|ts|u|u:|w|y|z;

$feedback=[silB]{$phoneme}[silE]
```

Standard left-right HMMs are used for training and recognition. The HMMs are based on the MFCC feature-vectors calculated in the preprocessing phase. Monophone models are used instead of diphone and triphone models although the latter are more powerful and widely used in speech recognition, because of their smaller number and lower complexity. While the monophone set for Japanese contains only 43 models, 7946 HMMs are contained in the Julius triphone set for Japanese.

As the initial HMMs only form a basis for constructing word models and training them in a user-dependent way, perfect accuracy is not needed for the initial models. Moreover, the number of states of the word models directly depends on the number of states of the concatenated elementary models, which is significantly higher for triphone models. To keep the number of necessary training utterances low, the degrees of freedom, that is the number of states and transitions, used when training the models should not grow excessively large.

The utterance-based recognizer

The utterance based recognizer starts with an empty HMM set for recognition at the beginning of the training process. It is successively filled with models, generated from observed utterances. When an utterance occurs, that is classified as new by the above described algorithm, the system creates a new model by concatenating the HMMs of the phoneme sequence recognized by the phoneme-sequence based recognizer:

First, the beginning and ending silence models are discarded, as they do not belong to the utterance itself and the recognizer should be able to recognize similar utterances even when no silence is present at the beginning and the end. Optional beginning and ending silence models are added later using the recognition grammar.

During recognition with the phoneme-based recognizer, the Viterbi algorithm calculates the recognized phoneme sequence by concatenating phoneme models. Concatenation is done by unifying the non-emitting first and last states of the phoneme models. Details

are given in the description of the Viterbi algorithm at the beginning of this chapter. The same way of concatenating a sequence of phoneme models is applied by the learning algorithm for creating a new utterance model from the best sequence of phonemes, that the Viterbi algorithm has determined. Based on the phoneme models, a new state sequence and transition matrix are written by copying the states and transitions from the phoneme models to the new model, which is used as a basis for training with the user's actual utterances.

To be able to integrate the newly created model into the utterance-based recognizer, the HMM list, grammar and dictionary need to be updated to include the new model. The HMM list is simply a list of all existing HMMs in a HMM set. The dictionary can be used to map a sequence of HMMs to an output symbol - e.g. to map the phoneme sequence "i: z i:" to the word "easy". In the proposed system, the dictionary maps every model on itself, because every feedback utterance is modelled by one HMM. The grammar, which is shown in listing 4.2, describes all possible sequences of HMMs that are allowed as an output of the recognizer. In the utterance recognizer, exactly one of the learned utterance is allowed, with an optional beginning and ending silence:

Listing 4.2: Grammar of the utterance recognizer.

```
$possibleutterance=utterance1|utterance2|utterance3|...

$feedback=[silB]{$possibleutterance}[silE]
```

Determining the threshold for creating new models

In the description of the learning algorithm for speech utterances, a threshold of 1^{-5} is mentioned as a suitable cutoff value for the decision whether an utterance is already known, or not.

The threshold was determined experimentally using a set of utterances, not used in training or recognition. The utterances were first hand-classified into classes of same or very similar utterances. In the next step, each of the utterances was recognized with the phoneme recognizer to create the best-fitting phoneme sequence. For each utterance a grammar was created, allowing only the best matching phoneme sequence of that utterance. And all other utterances were "recognized" using that grammar, to obtain the confidences, which are represented as the log likelihoods per frame.

This way, typical differences of log likelihoods per frame within a class of same utterances and between classes could be determined. The threshold was selected in such a way, that it lies between the average in-class difference and the average between-classes difference.

However, it was found to be useful to select a threshold closer to the average in-class difference than to the between-classes difference, so that the system avoids training and associating the same model with utterances with different meanings, which would lead to misrecognitions. This may sometimes lead to creating two different models for utterances, that are actually the same, but this can be handled by the associative learning stage, which can associate both models with the same meaning.

4.6.2 Prosody

The approach, proposed in this dissertation, employs HMMs for recognizing the prosody of speech utterances. The HMMs for interpreting prosody are based on features extracted from the speech signal, as described in section 4.5.3. The signal is divided into frames of 32 ms length with 16 ms overlap. For every frame, the system calculates the pitch, the energy as well as the frequency spectrum.

Based on this data, a feature vector is calculated consisting of the pitch, the pitch difference to the previous frame, the energy, the energy difference to the previous frame and the energy in frequency bands 1-n. The sequence of feature vectors is written to a file in HTK format in order to be used for training the HMMs.

Additionally, the algorithm calculates some global information based on the sequence of feature vectors for all frames belonging to one utterance. These are the average, minimum and maximum pitch and energy, the range and standard deviation of pitch and energy as well as the average difference between two adjacent frames of pitch as well as energy. For determining, which HMM is trained with which utterances, the system relies on these global features which have proven effective for speech emotion and affect recognition [13][55][59].

The k-means [34] [71] algorithm is used for clustering utterances with similar global features. The cluster number is optimized between two and ten. This clustering approach using global features was chosen instead of directly working with the time-series data, because there were no available, previously trained initial models for clustering similar perceptions, like the phoneme HMMs for speech. Therefore, the approach of comparing the confidence values of two recognizers to decide whether to create a new cluster or not, could not be applied easily to learning prosody. After the clustering has finished, one HMM is trained for each of the clusters, generated by the k-means algorithm.

To associate the HMMs with approval or disapproval, every utterance is recognized using the trained HMMs to get the best matching model. This model is then passed to the feedback association learning stage. Figure 4.12 shows an overview of the prosody recognition.

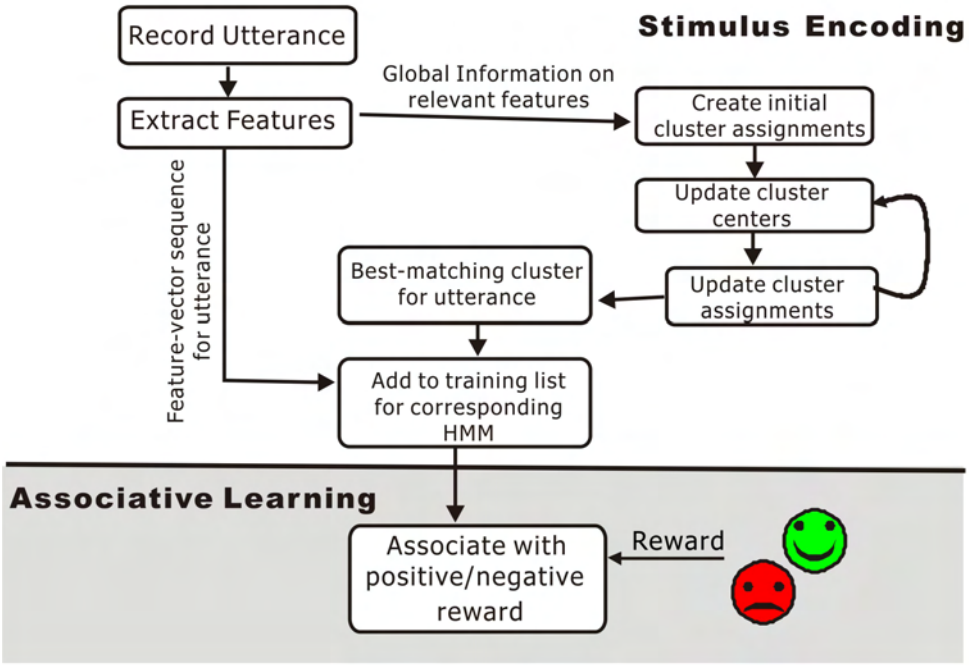

Figure 4.12: Algorithm for learning prosody.

Clustering prosody training samples

The input to the clustering are global feature vectors. One such feature vector is generated per utterance, containing global information on its pitch and energy contours as well as the frequency distribution, such as averages, variances and average differences. Before starting the clustering, the feature vectors are normalized. This is necessary to calculate the euclidian distance for the k-means algorithm, because the different features have different ranges of values. Otherwise, features with large ranges of values, such as energy variance, would have a higher influence on the distance than features with a small range of values, such as the mean pitch difference between frames.

The clustering, based on the k-means algorithm then works as follows: First, the prosody feature vectors are distributed randomly to k clusters. The averages of the feature vectors in each cluster are calculated and used as the centers of gravity of the clusters for the next step of the clustering algorithm.

Next, every feature vector is re-assigned to the cluster with the smallest euclidian distance between the feature-vector and the cluster center. After reassigning all feature vectors, new centers of gravity are calculated for each cluster.

The execution of the algorithm ends, when either a maximum number of 1000 iterations has been reached or the cluster centers do not change anymore.

The k-means algorithm assumes a fixed value for k. As the number of different prosody classes is not known, the system runs the k-means algorithm multiple times for cluster numbers between two and ten. The quality of the clustering is then calculated as the average distance between cluster centers divided by the average distances within the clusters between the prosody feature vectors and the cluster center. The best cluster partitioning is then used to decide which utterances are used to train which HMM. For every cluster one HMM is created and trained with the utterances corresponding to the feature vectors within that cluster.

4.6.3 Touch

The implementation of the touch recognition does not use HMMs to model touch but a simple duration based model. This is because the output of the touch sensors of the AIBO robot does not suffice for HMM-based modeling: It is binary and does not contain any information on the force applied when touching the sensors. Moreover, the refresh rate when using the AIBO remote framework is quite low. Therefore, the algorithm classifies touches of the head sensor and of the back sensor depending on their duration:

- short: less than 0.5 seconds
- medium: between 0.5 seconds and 1 second
- long: one second or longer

Typically, short touches were observed when the user was hitting the robot, while medium and long touches corresponded to stroking the robot. However, many participants in the user study employed touch only for expressing approval by stroking the robot.

4.7 Feedback association learning

In the feedback association learning phase, an association between the HMM or touch pattern, that matches the current observation and is obtained from the feedback recognition learning, and either approval or disapproval is created or reinforced. The information of whether an HMM should be associated with approval or with disapproval is obtained from the current state of the game and passed as an input to the learning algorithm. If the last move of the robot was a good one, the model, which represents the perceived user feedback, is associated with approval. If the last move was a bad one, it is associated with disapproval.

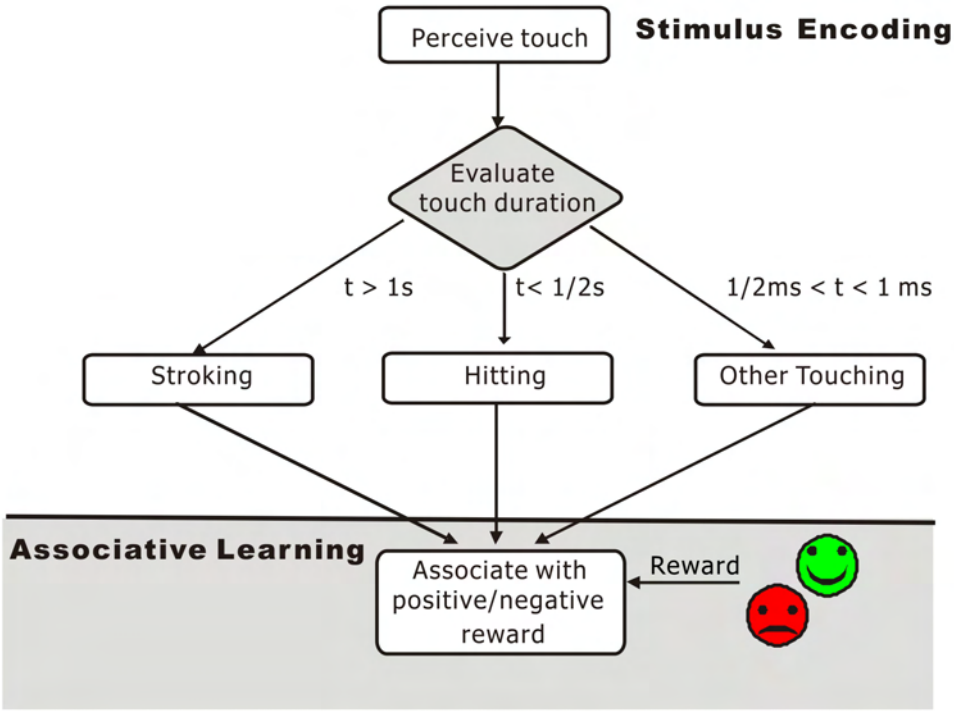

Figure 4.13: Algorithm for learning touch.

In the proposed system the associations between HMMs and stimuli are stored in an association matrix. The system uses the Rescorla-Wagner algorithm, which is described in more detail in section 4.4.2, to calculate the associations between positive and negative feedback on the one hand, and utterances, prosody patterns and touches on the other hand, and to update the association matrix accordingly. When a user feedback is observed, the association between the observed stimuli, encoded by the corresponding HMMs and touch models, and either the US "positive feedback" or "negative feedback" is strengthened. The change in associative strength, $\Delta V_A(n)$, is calculated as described in equation 4.38 and added to the corresponding value $V_{A,US(n)}$ in the matrix.

$$\Delta V_A(n) = \alpha_A \beta_{US(n)} (\lambda_{US(n)} - V_{all(n)}) \tag{4.38}$$

In the next step, the formula is used with the CS that have been observed and the US that did not occur (e.g. negative feedback in a positive feedback situation) to handle extinction, so that incorrect associations, resulting from misrecognitions or incorrect user feedback get weaker over time.

In this study, the learning rates α and β for conditioned and unconditioned stimuli are fixed values for each modality. They determine how quickly the robot learns and adapts to changes in feedback behavior. The maximum associative strength $\lambda_{US(n)}$ is set to one, if the corresponding CS is present, when the US occurs, zero otherwise. The combined associative strength of all conditioned stimuli towards the unconditioned stimulus $V_{all(n)}$ is calculated by summarizing the association values of all CS towards the US, that have been calculated in the previous runs of the feedback recognition learning.

The major drawback of the Rescorla-Wagner-Model is that it cannot model the effects of second-order-conditioning and sensory preconditioning directly. The algorithm deals with this issue by using a second association matrix storing the associations between different conditioned stimuli and running a second pass of the Rescorla-Wagner-algorithm to learn associations between simultaneously occurring CS. In this second pass, the CS_1 serves as the US for the conditioning of CS_2. In a third pass, the algorithm updates the relation between the US and all CS_2, that have an association to the observed CS_1, using a new learning rate $\alpha_{A_{second}}$, which is calculated as the product of the original learning rate α_A and the associative strength between the CS_1 and the corresponding CS_2.

4.8 Integration of top-down processes

Without top-down processes, all HMMs are equally likely to be selected for retraining in the feedback recognition learning phase. The selection of the best-matching model depends only on the perceived signal while the context is not taken into account.

Based on the findings on human learning, which are described in section 2.2, integrating context information using top-down processing is helpful to disambiguate ambiguous perceptions, deal with noisy signals and reduce the search space by pruning interpretations of perceptions that do not fit in the situation.

In order to improve the selection of the best-matching speech and prosody models for retraining, a model of top-down processes has been integrated into the system [21]. It uses the associations, learned in the feedback association learning phase, to generate expectations about which HMMs are most likely to occur in a given context.

Knowing through the state of the training task, whether positive or negative feedback is expected from the user, the system uses the learned association matrix to assign a positive or negative bias to each of the existing HMMs. The algorithm calculates the bias B_A for an HMM A from the difference of the associative strength V_A of the HMM A towards the expected feedback and the associative strength of it towards the opposite feedback. For positive feedback, the factor would be calculated as in (4.39).

$$B_A = aV_{A,positive} - bV_{A,negative} \tag{4.39}$$

The constants a and b, which can have values between 0 and 1, determine the impact of the excitatory and inhibitory influences on the calculated bias. A high value a makes the system reuse known HMMs, which are already associated to the present stimulus. A high value b makes the system avoid HMMs, which are already associated to a different stimulus. It was found that moderate values for a and high values for b produce best results. In the experiment, the learning algorithm used the values $a = 0.2$ and $b = 0.8$.

The bias B_A is used, if the feedback recognition learning determines that there is more than one HMM that models the stimulus well enough to be a candidate for retraining. In this case, the biases modify the confidence factors returned by the Viterbi algorithm. The biases B_A and the normalized confidence factors C_A are weighted as shown in equation 4.40 to select the best HMM for retraining.

$$D_A = cB_A + (1 - c)C_A \tag{4.40}$$

Using this method, HMMs, which are already associated with either positive or negative feedback, become more likely to be selected when a similar feedback is expected again. Depending on the constant c, associations of one HMM with both positive and negative reward are more or less likely. A value of $c = 0.8$ has turned out to increase the quality of the HMM selection, while still allowing HMMs for ambiguous utterances to be associated with both, positive and negative reward.

4.9 Extensions of the learning method

While the previously explained procedure has been designed to allow both, online learning during actual interaction as well as offline learning using pre-recorded data, some additional improvements can be made when online training is not needed and the system is trained offline using a full set of data, recorded during the training phase with a user.

4.9.1 Multiple passes through the training data

By exploiting the offline training, additional information can be extracted from the training data and initial misrecognitions and sub-optimal assignments of utterances to HMMs can be handled to improve the trained models by passing through all training data gathered from one participant multiple times:

When the system learns in one pass through the training data, applying the proposed algorithm can result in different qualities of trained HMMs depending on the order of the training samples. Utterances, that have been assigned to train a certain model at the beginning of the training might be a better fit for another model which is created later.

To overcome this problem and increase the accuracy of the clustering process, the algorithm can make up to three passes through the training data. In the **first pass** it only decides whether to create a HMM if an utterance is unlikely to have been observed before, or not to create a new HMM if an utterance is close to an existing model. However, in this phase, the system does not train the models with the user's utterances but just creates utterance HMMs by concatenating phoneme models.

In the **second pass** the algorithm does not create any new HMMs but uses the HMMs, created in the first pass and assigns each utterance to the best-fitting HMM. At the end of the second pass, the HMMs are retrained with the assigned utterances. This way, the system can make sure that all utterances, including the ones at the beginning of the training are always assigned to the best fitting model.

After the second pass the HMMs are trained with the assigned data. It is possible to use a **third pass** to again reassign the utterances. This especially improves the accuracy if HMMs with a low number of training examples have been pruned and the corresponding data has to be reassigned.

4.9.2 Treating "no feedback" as a special kind of feedback

One issue, that became clear during the evaluation of the study, is that in many cases there is a meaning or systematics when users do not give feedback or do not use a certain modality, when giving feedback. For example, the user only gives positive feedback by touch, while he does not touch the robot for negative feedback, or the user sometimes does not give any feedback at all when the robot makes good moves but reliably corrects bad moves.

Therefore treating "no feedback" in a situation, where feedback is expected, as a special kind of feedback helps increasing the recognition accuracy. The information, that no feedback has been given through a certain modality is exploited by the learning algorithm as follows:

If no speech or touch was observed after the robot has made a good or bad move, the symbol NO_TOUCH or NO_SPEECH is passed to the association learning phase as a result of the stimulus encoding instead of a HMM or touch model together with the other observed stimuli. The symbols NO_TOUCH or NO_SPEECH are handled by the associative learning in the same way as "normal" speech or touch stimuli and thus get associated with positive or negative reward and can be used for recognition in the same way as other stimuli.

4.9.3 Pruning and re-clustering samples

Training a Hidden Markov Model needs a minimum number of training samples. Otherwise the trained model overfits the training samples and does not generalize well. When the system creates models which only very few training examples then there are basically two options to avoid overfitting models: Either not to train these models but use the original phoneme sequence that was generated when the model was first created or to prune models with a too low number of training samples and delete them from the HMMSet. In [89] a minimum number of three training utterances per HMM is suggested. This value has been used for training and evaluation of the system.

If models are pruned and the amount of training data is small, it usually makes sense to re-allocate the training samples for these models to one of the existing models. In a third pass through the data after the first training, this is done by forbidding the creation of new models, reassigning each utterance to the best-fitting HMM and re-training the models with the data.

4.9.4 What parts of the HMMs should be re-trained?

Due to the small number of training examples available for each model, it is necessary to reduce the number of degrees of freedom of the HMMs to be trained. As described at the beginning of this chapter, an HMM is defined by its states, its output probabilities and its transition probabilities. In order to keep the degrees of freedem for training low, the system only trains the means of the gaussians for the output probabilities of the HMMs while keeping the transition probabilities as well as the variances of the original models.

Keeping the variances of the original models has a second benefit apart from reducing the degrees of freedom for the training. If the amount of training data is small, the variances can become too narrow, as a result of overfitting to the training data, which makes them unsuitable for recognizing utterances, that have not been part of the training data. In [89] it is recommended to set a floor for the variances to avoid this problem. If re-training from an existing set of HMMs, trained with a large amount of data, such as the Julius phonemeset, used in this thesis, keeping the variances results in the same effect.

4.10 Recognition using the trained models

The output from the training phase is a set of HMMs for speech utterances and prosody, models of touch as well as an association matrix holding the associations between the trained HMMs and the symbols “positive” and “negative”.

After the training, the trained models and association matrix can be used for recognizing feedback. Fig. 4.14 shows an overview of the recognition process. Like in the training

phase, the data is first **preprocessed** and converted into sequences of feature vectors for speech, prosody and touch. After the **preprocessing**, the speech, prosody and touch feature vectors are passed to the **stimulus encoding** stage, which uses recognizers based on the trained models created by the learning algorithm to determine the best-matching HMM for the observed utterance or prosody pattern or the best touch model. Then the **strongest association is determined** using the trained association matrix and the results of the recognizers as an input. The output of the recognition is a classification result of either "positive feedback" or "negative feedback".

All feedback given in one feedback situation, that is after one good or bad move of the robot, is combined for recognition. The combined associative strengths $V_{all,positive}$ and $V_{all,negative}$ for positive and negative feedback respectively are obtained by summing up the associative strengths between all recognized speech, prosody and touch models (CS) in feedback situation n and either positive or negative feedback. For $V_{all,positive}$ the combined associative strength is calculated as follows:

$$V_{all,positive}(n) = \sum_{A \in CS(n)} V_{A,positive} \tag{4.41}$$

The combined associative strength for negative feedback is calculated accordingly. The recognition result is the feedback with the highest combined associative strength to the recognized models. It is output as the result of the recognition process.

4.11 Experiments

The training method and the learning algorithm were evaluated experimentally. Ten persons, aged 23 to 47, participated in the study. All of them were Japanese graduate students or employees at the National Institute of Informatics in Tokyo. Five of them were females, five males. All participants had experience in using computers. Two participants had previous experience in interacting with entertainment robots. Interaction with the robot was done in Japanese. During the experiments, roughly 5.5 hours of audio and video data were recorded. Details on the experiments as well as a quantitative analysis of the observed user behavior were presented in section 3.4.

4.11.1 Results

The performance of the learning algorithm was evaluated offline with the data recorded within the above described setting. On average 24.73 (sd=16.50) HMMs were created per speaker for speech and 4.49 (sd=0.22) for prosody. On average, 65.13% (sd=18.37%) of the models for speech, prosody and touch were associated with positive and 34.87% (sd=18.37%) with negative reward.

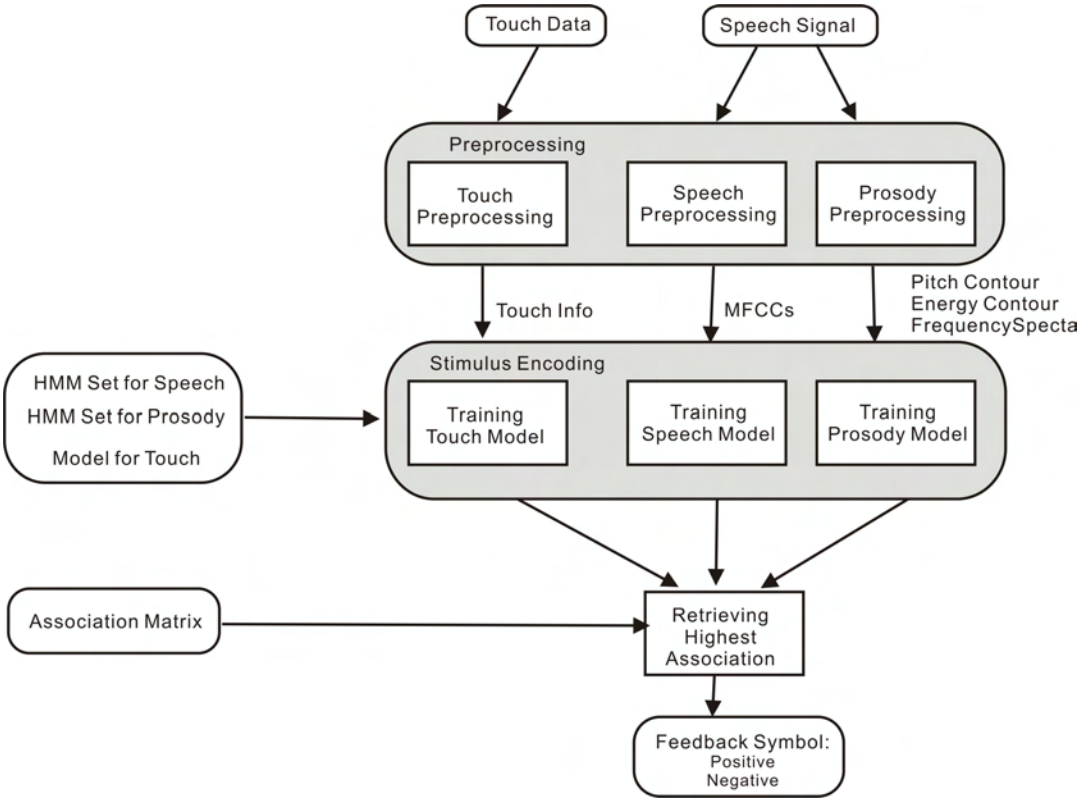

Figure 4.14: Overview of the recognition

The system was trained and evaluated in a user-dependent way using 10-fold cross evaluation. The recognition rate is calculated as the number of accurately recognized feedback situations divided by the overall number of feedback situations. A feedback situation corresponds to one good or bad move of the robot. All the feedback that occurs after a move of the robot is combined for recognition. The average accuracy of the proposed method for classifying between approving and disapproving feedback, given by one user, based on speech, prosody and touch was 95.97%. The standard deviation between users was 3.30%. As the feedback given by the participants showed a slight bias toward approval, the confusion matrix, shown in Table 4.1 gives a more detailed overview over the performance of the recognizer.

Without using top-down processes for speech, the recognition rate for speech as a single modality drops to 77.42% (sd= 13.12%). This degrades the overall recognition rate to 93.95% (sd= 5.33%).

Using speech only the recognition rate was 83.53% with a standard deviation of 8.30%. Using prosody only, the recognition rate was 84.27% with a standard deviation of 8.57%.

Table 4.1: Confusion matrix for feedback learning (in percent)

	Positive(actual)	Negative(actual)
Positive (recognized)	52.50	1.76
Negative (recognized)	2.27	43.47

For touch the recognition rate was 88.17% with a rather high standard deviation of 11.77% as the usage and frequency of touch varied strongly between users.

To calculate the single-modality recognition rates, all feedback given through other modalities was filtered out. However, all feedback given through the selected modality occurring in one "feedback situation", for example multiple speech rewards given for one move of the robot, was still combined. All single-modality recognition rates are considerably lower than the recognition rate for multimodal feedbacks shown above. This underlines that combining stimuli given through different modalities is crucial for reliable recognition.

A marginally significant improvement ($p=0.0514$, $t=1.816$, $df=9$) was found when comparing the combined multimodal recognition to the best single modality, touch, using a paired T-Test on the recognition rates for all participants. Significant improvements were obtained when comparing the combined multimodal results to the results from speech ($p=0.0015$, $t=4.014$, $df=9$) or prosody ($p=0.0010$, $t=4.315$, $df=9$) alone. The recognition rates for the different modalities and participants can be seen in Fig. 4.15.

When using a combination of speech and touch without prosody and without top-down processes, the recognition rate was 90.33%. The improvement reached by adding prosody and top-down processes was not statistically significant. Possible explanations for this are the relatively low number of participants and the fact that three of the ten participants already had 100% recognition rate with only speech and touch. Especially the results for the participants which had low recognition rates using only speech and touch benefitted from adding prosody and top-down processing.

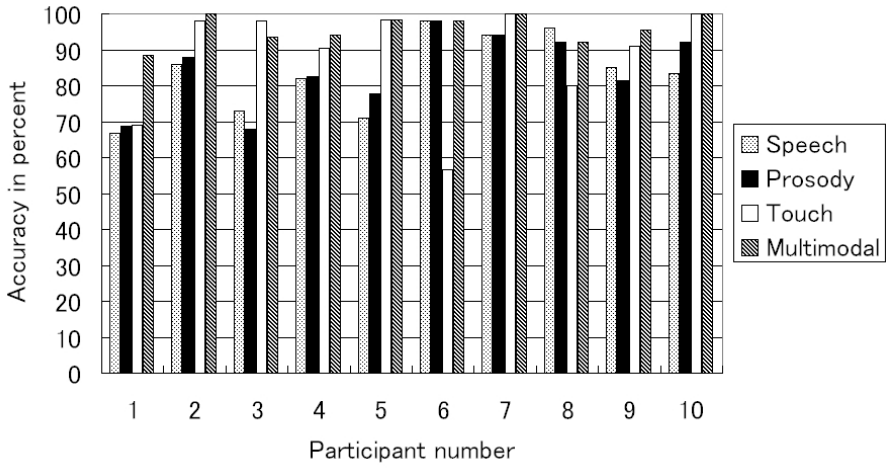

Figure 4.15: Multimodal and single modality recognition rates for all participants.

"Jede Loesung eines Problems ist ein neues Problem."
("The solution of every problem is another problem.")
Johann Wolfgang von Goethe (1749 - 1832)

5

Extension of the algorithm for learning to understand parameterized commands

5.1 Overview

This chapter shows how the method, described in chapter 4, was extended in order to enable the robot to learn not only feedback but also object names as well as parameterized commands. Understanding commands and understanding feedback are necessary functions for a robot that assists a human, for example, in everyday household tasks.

Using the training task, described in section 3.5, in which the user and the robot interact which each other using a virtual living room scenario, the system learns to understand eight commands and 16 different object names through situated interaction with a user. Like the training for feedback leaning, the training for command learning is also performed using a "virtual" training task. The training task depicts a simplified living room scene and is conducted in two successive phases, as described in chapter 3.5. In the first phase of the training, the robot learns to understand the names of objects in the living room by asking the user to name the objects that it points at. When enough object names are known, the robot continues with the second learning phase.

In the second phase of the learning process, the robot learns command patterns like "switch the <object> on!", "Please move <object> to <place>" etc. It uses the task server, which visualizes changes in the virtual living room scene, to make the user utter commands with a predefined meaning and learns the commands as well as the positions of parameters, that can be used to specify the command.

The HMMs for the object names, which have been trained in the first phase, are used in the second training phase for determining and learning the positions of parameters in commands. By searching the most likely part of an utterance, that corresponds to a trained object name HMM, e.g. finding the most likely position of an HMM associated with the meaning *TELEVISION*, when an utterance with the meaning *SWITCH_ON(TELEVISION)* is expected, the system can determine the parameter positions in the commands and model the order of command parts and parameters in the grammar for the recognizer.

In order to enable the system to learn commands, both, the stimulus encoding and the associative learning stage had to be adapted to the new task. Details of the necessary adaptations are presented in the following paragraphs.

5.2 Requirements

While positive and negative feedback utterances do not need to be segmented but can be processed as a whole, commands may contain different parameters, which need to be handled by the system. For example, the command "Put the book on the table!" contains the object name "the book" and the place name "the table" and the command "put" itself. In order to understand the meaning of the whole utterance, the command and its parameters need to be segmented and the system needs to be able to determine, which parts of an utterance belong to the command itself and which parts of the utterance belong to parameters, so that the trained command pattern can later be combined with different parameters for recognition. For example a command model for the concept *SWITCH_ON*, that was trained with the utterance "Switch the <TV> on" can be reused for the utterance "Switch the <light> on" or "Switch the <radio> on".

Users do not always give commands in an explicit way but can also give implicit commands. Instead of saying "Switch the light on!", the user might say "It is too dark here." or instead of "Switch the TV on!" he or she might say "I would like to watch TV." The learning method must be able to handle implicit commands as well as explicit commands.

When a user utters a command, some expected parameters may be omitted. For example, the user might say "Put <the ball> away" instead of "Put <the ball> in <the box>". The system must be able to detect and handle such cases.

In order to learn the correct meaning of its user's utterances, the robot needs to know in advance, which commands the user is going to utter and even make the user give commands in his or her preferred way but with a predefined meaning. However, verbal instructions must not be given to the user in order to avoid influencing the user's wording. This is done by showing situations in the virtual living room, where it is obvious which task needs to be performed by the robot. The user is informed about which actions he or she should ask the robot to perform, by typical and easily recoverable changes in the living room, such as a carpet getting dirty or a book falling from the shelf. Moreover, thought balloons with appropriate icons are used to visualize requests of the user, which cannot be understood easily from the state of the virtual living room alone, such as wanting a coffee or wanting the robot to shutdown.

Details are given in section 3.5. Some examples of command visualizations and possible commands from the user are:

- It is getting dark and the light is still switched off:
  "Switch the light on!"
- A dirty spot on the carpet:
  "Clean the carpet, please!"
- A book has fallen off the shelf:
  "Can you put the book on the shelf?"
- An icon showing a battery and a question mark?:
  "What is your battery status?"
- A thought balloon showing a battery and a connector:
  "Go to your charging station!"

5.3 Outline of the implemented extensions

In commands or requests, uttered naturally by a human to another human, a pet or a robot, only a part of the utterance actually conveys the meaning, while a relatively large number of words could be omitted without affecting the understanding of the command and often make a grammatical and semantic analysis more difficult. Such words are, for example "please", "can you" or "for me" as well as articles and several other grammatical constructs. What the robot actually needs to know to execute a naturally spoken command like "AIBO, can you please switch off the light for me?" can be reduced to the simple meaning *SWTCH_OFF(LIGHT)*. Therefore, the proposed system tries to learn models of the actual utterances of a user, without analyzing their structure more than necessary, and map the learned utterance models to their meanings.

The system does not try to analyze the grammatical structure of a command, but rather attempts to learn command-patterns as a whole and just determine the positions, where parameters can be inserted, instead of parsing the utterance and analyzing it word-by-word. A **command-pattern** is a sequence of utterance-part HMMs and placeholders for parameters. Each of the utterance-part HMMs can cover a sequence of multiple words, which are not further analyzed.

For example, if the user utters the command "Bring me a coffee please!", the system would first determine the position of the word "coffee" as an already learned object name, and then use the parts "Bring me a" and "please" to train the utterance-part HMMs *BRING1* and *BRING2* respectively and create the command-pattern *BRING1* {*PAR1*} *BRING2* for the command *BRING*. {*PAR1*} is a placeholder, where the system can insert parameters at recognition time. So the same command pattern, which has been trained on the utterance "Bring me a coffee please!" can be used for recognizing the utterance "Bring me a tea please!", if the word "tea" has been learned before.

While this approach is less flexible than a full grammatical analysis it can be used relatively easily to model a user's typical ways of uttering commands without requiring any prior knowledge on the user's language or typical way of speaking.

In a real-world scenario, the training tasks, which allow the robot to adapt to its user, would have to be performed before actually starting to use the robot. In order to allow for a quick and easy training, for example in front of the TV/PC screen, "virtual" training tasks are used. For the experiments on command learning an animated virtual living room was created as a simplified 3D-model of a living room, because this is a typical scenario in which a household robot would be applied. It is described in more detail in section 3.5 of this thesis. The virtual living room is projected on a white screen and the robot uses motions, sounds and its LEDs to show which moves it is making. Appropriate animations are shown in the virtual living room for each move.

5.4 Learning command patterns vs. symbol grounding

A lot of research has been done on automatic symbol grounding for robots [40] [72]. Symbol grounding is a complex task in which symbols, such as the words of a natural language, are connected with meanings, that is objects, places, actions etc. in the real world. It often involves visual recognition and naming of objects or actions.

This work has a slightly different focus. The proposed approach concentrates on learning how a certain user utters commands and feedback, but it assumes that the robot already knows basic symbolic representations of the actions, that it is able to perform and the objects/places, it can recognize. For example, it knows, how to perform the task, represented by *MOVE(objectA, placeB)*. The approach further assumes, that these actions can be performed in the virtual training task and also be transferred to a real-world task.

In order to react to natural, multimodal commands and feedback, based on its known capabilities, the robot needs to learn a mapping between its existing symbolic representations of commands, object/place names or feedback and the way, they are expressed naturally by a certain user, using speech, prosody and touch. This enables the robot to deal with instructions given by the user in his or her preferred way. Assuming, that the robot already knows basic grounded symbols by the time of the training with the user is a relatively hard requirement. However, as this knowledge can be acquired in a user-independent way, this is likely to be the case for typical service- or entertainment robots, which normally have a set of built-in functions and can visually recognize and manipulate certain objects in their environment. While commands and feedback depend on the user and are particularly suitable for acquisition in a training phase, the required mapping of objects to symbols could be performed without training by a human teacher, for example based on markers, pre-trained recognizers or RFID tags attached to objects. Therefore, the proposed learning

method focuses on the mapping between simple symbolic representations and users' actual utterances.

5.5 Extensions to the training tasks

The design of the training phase is a key point for the learning method, because it needs to enable the robot to provoke commands as well as feedback from the user, which are leaned in the virtual task but can be used to control the robot in the real world. For training the robot, computer-based "virtual" training tasks are used, like in the previous study on feedback learning. However, in order to provoke useful commands, the system cannot use the previous game-like tasks but instead uses a "virtual living room", a simplified 3D model of a living room. In this "virtual living room", the robot is able to perform all the commands, that it needs to learn. The basic idea behind this approach is, that the user would perform the training together with the robot in front of the PC or TV and then use the learned commands and feedback to control the robot in the real world.

The virtual living room, that has been used for the experiments, is shown on a large screen and the robot uses motions, sounds and its LEDs to show which move it is performing. During the training, the robot cannot actually understand its user but needs to react appropriately to allow natural interaction. This is ensured by designing the training task in a way, that the robot can anticipate the user's commands.

During the training phase, the robot sends the requests, which object, place or command and reward it wants to learn, to the task server. The task server then visualizes the expected command or highlights the requested object/place on the screen in a way that the user can understand it easily. It also sends relevant information, such as the coordinates of objects back to the robot, so that it can, for example, perform a pointing gesture to ask for an object or place name. When the user utters a command, the robot can either perform a correct or incorrect action to provoke positive or negative feedback from the user. This way, the robot is able to explore the user's way of giving different commands as well as feedback.

The system can only learn verbal representations of simple commands consisting of one action and the related objects. Table 5.1 shows the set of commands that the robot learns in the experiments along with their parameter signature and an example of a sentence that the user might utter.

5.6 Extensions to the learning algorithm

Like the algorithm for feedback learning, the command learning algorithm is divided into a stimulus encoding phase and an associative learning phase. In the stimulus encoding phase, the system trains Hidden Markov Models (HMMs) to model command patterns,

Table 5.1: Commands that were used in the training task

Command	Parameters	Example sentence
move	object, place	Put the ball into the box.
bring	object	Bring me a coffee, please.
clean	object	Please clean up the carpet.
switch on	object	Robot, switch on the light.
switch off	object	Switch off the radio.
charge battery	-	Recharge your battery.
call	person	Please make a phone call to Rita!
show status	-	What is your status?

object/names which are used as parameters, as well as positive and negative rewards based on speech, prosody and touch stimuli from the user.

In the associative learning phase, the system associates the trained models with a known symbolic representation, integrating the data from different modalities. For example, it associates an HMM sequence, representing the utterance "Could you please move <A> to <B>" with the known symbolic representation *MOVE(object, place)* or the utterance "Good robot" and a touch of the head sensor with positive reward.

An example of a data structure resulting from this learning process is shown in Fig. 5.1. The rewards are modeled in the same way as described earlier in chapter 4. Command patterns are modeled as sequences of HMMs for speech and touch. Prosody models are associated with either positive reward, negative reward or command, to distinguish these three classes of utterances, in case this information is not given by the context. The representations of place and object names as well as details on the structure of the command patterns are not shown in the figure. It can be found in Fig. 5.2.

5.6.1 Stimulus Encoding

The stimulus encoding phase was modified to enable the robot to learn commands and object/place references. As described in the previous chapter, the learning algorithm for feedback is based on Hidden Markov Models for speech as well as for prosody and a simple duration-based model for touch. However, for learning commands and object names the contents of the speech utterances plays the most important role, while prosody and touch have relatively little impact.

As in the feedback learning algorithm, the system determines the best matching models for each command or object name, spoken by the user, using two recognizers in parallel. If there is no good existing model, a new one is created. Otherwise, the best matching model

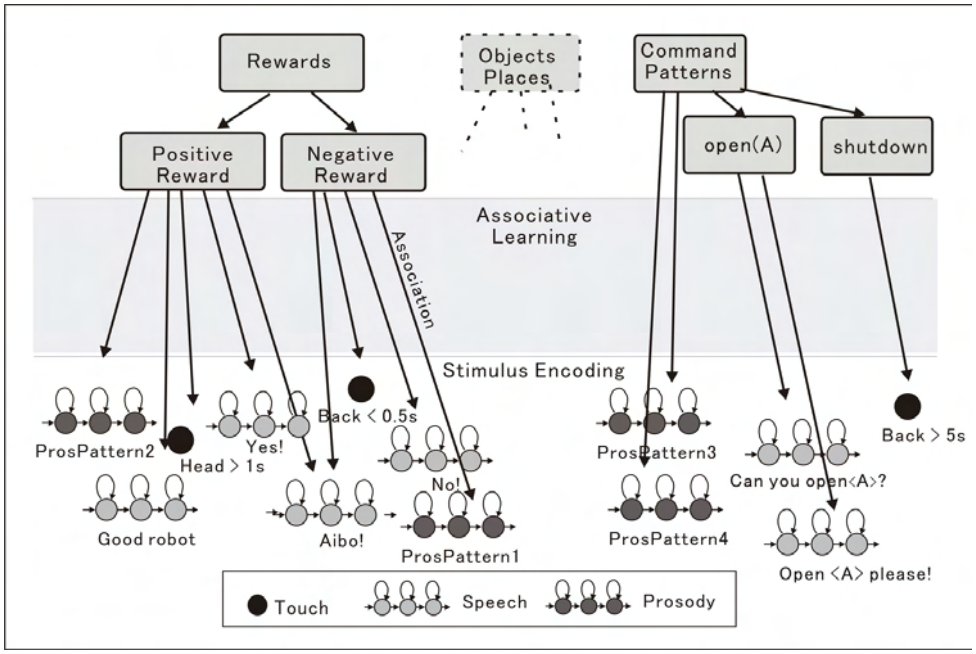

Figure 5.1: Data structure created by the learning algorithm

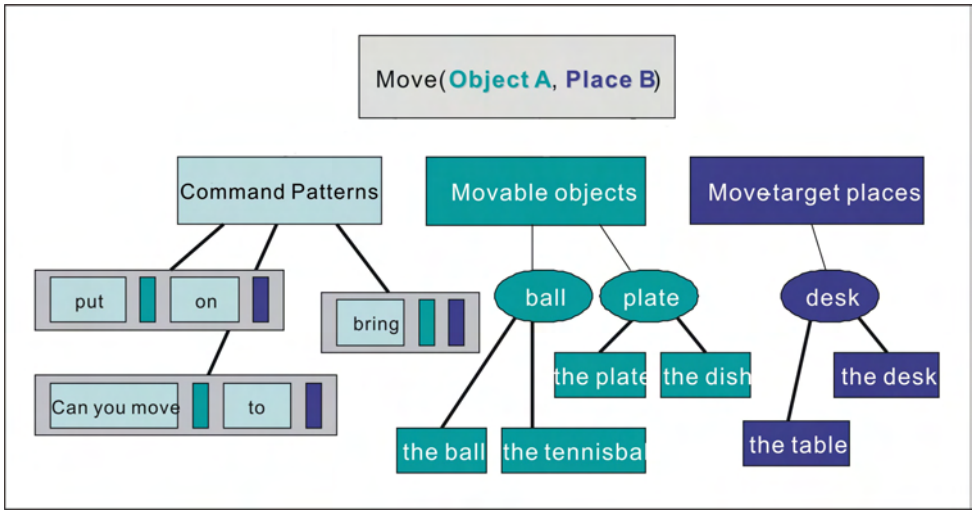

Figure 5.2: Data structure of a command

is retrained with the data corresponding to the observed command or object name. When retraining has finished, the models are passed on to the association learning stage.

Speech

For learning commands, it can be assumed, that speech is by far the most important modality. Therefore the development of extensions for learning commands and object names was focused on speech.

There are three different kinds of utterances, that the speech stimuli encoding needs to deal with: positive/negative feedback, names of objects/places and command-patterns. The learning of feedback has already been described in the previous chapter. Object names can be learned in exactly the same way as user feedback, because the whole utterance can be modeled as one HMM.

As opposed to user feedback and object names, command-patterns can have a variable number of slots for inserting parameters. Typical parameters are object- or place-names. The commands in the experiments had zero to two parameters, like "Recharge!", "Clean <object> please" or "Can you move <object> to <place>?". An example of a command structure is shown in 5.2. The leaves of the tree are trained HMMs. The inner nodes are symbolic representations of objects and command patterns. The thick lines represent associations, learned later in the associative learning phase. Feedback-utterances, names of objects/places and commands without any parameters can be trained as single HMMs. In case of commands with one or more parameters, the system needs to model the corresponding command pattern using multiple HMMs with placeholders to allow the insertion of HMMs representing objects/places used as parameters.

In the same way as for learning feedback, the system also uses two recognizers in parallel for learning commands in order to determine whether an utterance is already known or whether a new utterance model should be created to encode an observed utterance. However, the recognizers differ from the ones, used for feedback learning.

The first, phoneme-based recognizer uses monophone models when learning object names and a combination of monophone models and trained utterance models for object names when learning commands. The second, utterance-based recognizer uses only the previously generated command and parameter models. It is initially empty. Details on the recognizers and the grammars used by them to determine object positions and parts of the command patterns are given in the following sections.

A new model is created when none of the existing models is a good match of the current utterance. This situation is detected by obtaining the recognition results of both the phoneme-based and the utterance-based recognizers and comparing the returned confidence values (log likelihood per frame). As in the feedback learning, the same difference in confidence of 10^{-5}, has been used as a cutoff value to determine whether an utterance is already known or not. Thus, If the result from the phoneme sequence recognizer is not at least 10^{-5} better than the result from the utterance-based recognizer, then it is assumed that the utterance is already known and it is retrained with the current utterance. Oth-

erwise it is assumed to be unknown and a new model is generated by concatenating the phoneme models corresponding to the recognized phoneme sequences.

If a model is created for an object name or for a command which does not have any parameters, one model is created for a whole utterance. However, when learning commands with parameters, multiple models are created. One model is created for each non-empty phoneme sequence either before the first parameter, after the last parameter or between parameters. Information about the order of the generated utterance models as well as the parameter positions and the order of the parameters is stored separately, because this information is needed for creating the grammar for the utterance-based recognizer. For example, if the expected command is *MOVE(BALL, BOX)* and the user says "Put the ball into the box!", then the system will create two new models, named *"MOVE1"* and *"MOVE2"* based on the phoneme sequences for "put the" and "into the" and stores the command pattern *"MOVE1{PAR1}MOVE2{PAR2}"*.

It is possible, that a user utters a command with less parameters than expected from the symbolic representation. For example the user might say *"It is too dark here"* to utter the command *SWITCHON(LIGHT)*. In this situation the number of parameters in the command pattern and in the symbolic command representation do not match and the missing parameter is usually implicitly contained in the utterance. This is handled in the associative learning stage of the algorithm, which is explained in section 5.6.2. An overview of the training for learning a command pattern is shown in Fig. 5.3.

Creation of the search grammars for training

In order to learn a command pattern consisting of multiple HMMs, the system must first determine which parts of the utterance belong to the verb pattern itself and which parts belong to its parameters. From the training task, the system knows which parameters to expect. The algorithm uses this information to locate object/place names in the utterance. The search grammars for the recognizers, used for learning object names and command patterns, are created and updated on the fly during the learning process.

For learning *object names*, the phoneme-based recognizer accepts an arbitrary sequence of phonemes with an optional beginning and ending silence. The utterance-based recognizer accepts exactly one known utterance model of an object name with an optional beginning or ending silence.

When learning *commands*, the system has to handle parameters and their positions within the utterance. Therefore search grammars for both recognizers need to be created for each observed utterance. The search grammars are used to find the the parameter positions in the spoken utterances from the user and model the rest of the utterance either by phoneme sequences or by previously learned command patterns in order to decide whether to train an existing command pattern model or create a new one.

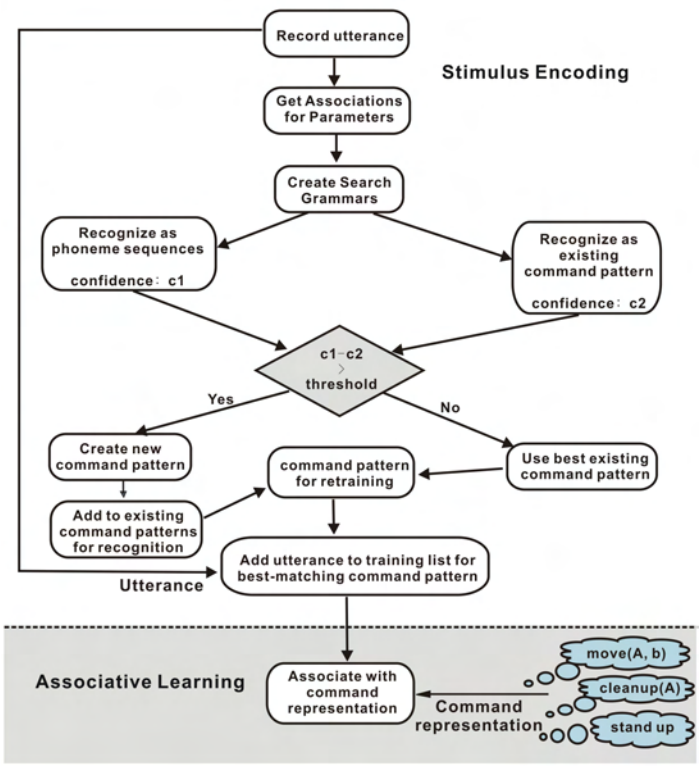

Figure 5.3: Extended learning algorithm for speech for command learning.

If, for example, the system perceives an utterance that is assumed to correspond to the symbolic representation *MOVE(BALL, BOX)*, then the system first uses the trained association matrix to find all models that have an association to the symbol *BALL* and all models that have an association to the symbol *BOX*. Based on this information, search grammars for the phoneme-based recognizer and the utterance-based recognizer are created:

The search grammar for the phoneme based recognizer allows both parameters in either order or one of the parameters or no parameter to handle cases, when the user omits one or all of the expected parameters when uttering a command. Arbitrary phoneme sequences are inserted before, between and after the parameters. Listing 5.1 shows the search grammar for a command with two parameters, object1 and object2.

The utterances 1 to 3 in this grammar are all utterances, found in the association matrix, which have an association to object 1. The utterances 4 and 5 are utterances, that have an association to object 2. The phoneme sequences are arbitrary sequences of phonemes, generated from the Julius monophone models that are also used for the phoneme-based recognizer in the learning of feedback and object names. The silence model is trained with only background noise.

Listing 5.1: Search-grammar of the phoneme-based command recognizer.

```
$Object1 = Utterance1 | Utterance2 | Utterance3

$Object2 = Utterance4 | Utterance5

$Phonemesequence = {N | a | b | d | ... | ts}

$Searchstring = [SilB] $Phonemesequence $Object1
  $Phonemesequence $Object2 $Phonemesequence [SilE]|
  ([SilB] $Phonemesequence $Object2
  $Phonemesequence $Object1 $Phonemesequence [SilE]|
  [SilB] $Phonemesequence $Object1 $Phonemesequence [SilE]|
  [SilB] Phonemesequence $Object2 $Phonemesequence [SilE]|
  [SilB] $Phonemesequence [SilE]
```

The grammar for the utterance-based recognizer, shown in listing 5.2, contains all known command utterances and inserts the models, which are associated with the expected parameters into the positions where parameters are expected. This information is contained in the command patterns, that were generated along with the utterance models. At the position in the command pattern, where {*PAR1*} is found, the system inserts a nonterminal which expands to all models associated with *BALL*. At the position, where {*PAR2*} is found, the system inserts a nonterminal which expands to all models associated with *BOX*.

Listing 5.2: Search-grammar of the utterance-based command recognizer.

```
$Object1 = Utterance1 | Utterance2 | Utterance3

$Object2 = Utterance4 | Utterance5

$Searchstring = [SilB] Commandpattern1-1 $Object1
  Commandpattern1-2 $Object2 Commandpattern1-3 [SilE]|
  ([SilB] Commandpattern1-1 $Object2
  Commandpattern1-2 $Object1 Commandpattern1-3 [SilE]|
  [SilB] Commandpattern2-1 $Object1
  Commandpattern2-2 $Object2 Commandpattern2-3 [SilE]|
  ([SilB] Commandpattern2-1 $Object2
  Commandpattern2-2 $Object1 Commandpattern2-3 [SilE]|
  [SilB] Commandpattern3-1 $Object1 Commandpattern3-2
  [SilE]| [SilB] Commandpattern3-1 $Object2
  Commandpattern3-2 [SilE]|  [SilB] Commandpattern4 [SilE]
  ...
```

Figure 5.4 shows how the search grammars for the phoneme-based and the utterance-based recognizers are generated when learning an utterance for the command *MOVE(BALL, BOX)* using the HMMs associated with the expected parameters.

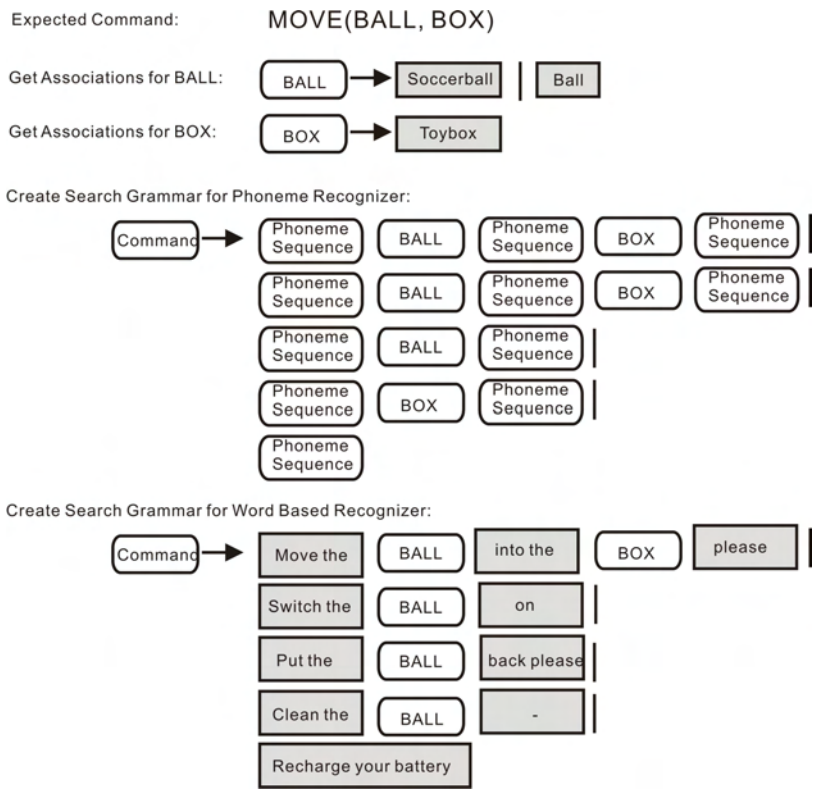

Figure 5.4: Grammar for the recognition. (grey: terminals, white: nonterminals)

Prosody

The recognition of prosody was mainly implemented to enhance the learning and recognition of positive and negative feedback. As it is unlikely, that prosody can be effectively used to discriminate between different commands or object names, only three classes of prosody are used for the prosody based recognizer: positive reward, negative reward and commands. While speech and touch stimuli are associated with individual commands, prosody is only used to discriminate between these three categories. While in the current training tasks, the distinction between positive and negative feedback as well as commands can be made easily, based on information about the current state of the training task, it may be necessary to use the recognizers to distinguish commands and feedback in other kinds of tasks.

For the prosody recognition, utterances are always processed as a whole without locating and cutting out parameters. The HMMs that are used for interpreting prosody are based on feature vectors extracted from the speech signal. The same feature vectors, as for learning feedback, containing the pitch, the pitch difference to the previous frame, the energy, the energy difference to the previous frame and the energy in frequency bands 1-n are also applied for the extended recognizer. Apart from introducing the class "commands" in addition to "positive feedback" and "negative feedback", there have been no changes for the prosody-based recognizer.

Touch

The user can also interact with the robot using its touch sensors on the head and on the back. Similar to prosody, touch is more important for learning rewards than for learning commands. However, the implementation gives users the possibility to use touch to express commands because there may be some commands, that are intuitive for the user to express by touch: e.g. use a long press of the back touch sensor to put AIBO into sleep mode. As it is unlikely, that users use touch to encode names of object or places, no associations are learned between touch patterns and objects/places.

To encode touch, the system uses its duration as well as the information whether the head or the back sensor was touched. There are three categories for short (< 0.5 s), medium ($0.5\text{s} < x < 1$ s) and long (> 1 s) touches.

In the first implementation on learning to understand positive or negative feedback, the algorithm did not take into account the exact sequence of short, medium and long touches. However, if the user employs touch to encode commands, the exact sequence may be important. The observed sequences of short, medium and long touches representing a command or feedback are encoded as strings, such as "LB,SH,LH" for a long touch of the back sensor, a short touch of the head sensor and a long touch of the head sensor.

A table is used to store all learned touch patterns in the stimulus encoding phase. For each observed command or feedback the stimulus encoding tries to find an existing pattern in the table and creates a new entry if necessary. The entry number is then passed on to the associative learning stage.

However, during the experiments, no touch stimuli were observed for encoding commands. Therefore the implementation could not be evaluated.

5.6.2 Associative learning

In a similar way as for feedback learning, the extended learning algorithm for commands uses classical conditioning to establish associations between the known symbolic representations of actions, rewards and objects/places and the trained HMMs for command

patterns and parameters. As in the algorithm for learning positive and negative rewards, the Rescorla-Wagner model [67] is employed to learn and update the associations.

When an utterance is recognized with a HMM sequence while a certain command is expected, the association between the corresponding command pattern and the symbolic representation of the command or object name is reinforced using the Rescorla-Wagner formula.

The symbolic representations of feedback, commands and their parameters are used as unconditioned stimuli. The HMMs, encoding stimuli coming from the user, are used as conditioned stimuli. The three different kinds of stimuli – feedback, command patterns and parameters – are handled separately from each other during the associative learning, as the training tasks enable the system to distinguish commands and feedback easily, based on whether the stimuli from the user were perceived before or after an action of the robot.

For speech, associations to HMMs are learned for the symbolic representations of feedback, of objects/places and for the different commands. For prosody, associations are learned toward either positive or negative feedback or the symbol "command", which stands for any command. This way, prosody can help to distinguish between feedback and commands from the user, when the robot is used in a real world task and faces situations, when it cannot reliably decide in advance, whether a command or a feedback is expected. Touch models can be associated with positive or negative feedback and with different command patterns, but not with objects/places, as it is unlikely, that users encode object or place descriptions into touch patterns.

When an utterance is recognized with a HMM sequence, while an object name or command is expected, then the association between the corresponding command pattern and the symbolic representation of the command or object name is strengthened. Using classical conditioning, multiple utterance models can be associated with the same symbol and vice versa.

When a command is learned and the command pattern, received from the stimulus encoding, contains less parameters than the symbolic representation expects, the command pattern is not only associated with the symbolic representation of the command but with a combined symbol consisting of the command and the parameters that were expected but not found in the utterance. For example, the system would create a new combined symbol *MOVE_BOX(PAR1)*, when the system expected a command with the meaning *MOVE(BALL, BOX)*, but was given the command "Put away <the ball>.", which lacks a parameter with the meaning BOX. The association of the command pattern "Put away {PAR1}" to the combined symbol *MOVE_BOX(PAR1)* is used to store the information, that the second parameter *BOX* of the original symbol *MOVE(PAR1, PAR2)* is included implicitly in the command pattern "Put away {PAR1}". When the system is given the new command "Put away <the toy car>" later, it can use the trained association to determine most probable second parameter, *BOX*, automatically. However, the command "Put away {PAR1}" can imply different second parameters depending on the given parameter PAR1. For example "Put away <the

ball>" means *MOVE(BALL, BOX)*, while "Put away <the book>" means *MOVE(BOOK, BOOKSHELF)*. So the command pattern "Put away {PAR1}" would get associated with both combined symbols *MOVE_BOX(PAR1)* and *MOVE_BOOKSHELF(PAR1)*. Based on the associations between the given parameters, learned in the training, the system is able to estimate the most likely missing parameter when recognizing an utterance, such as "Put away the book" or "Put away the ball".

Classical conditioning has different desirable properties, such as blocking, secondary conditioning has different desirable properties, such as blocking, secondary conditioning and sensory preconditioning which allow the system to integrate and weight stimuli from different modalities, emphasize salient stimuli and establish connections between multimodal conditioned stimuli, e.g. between certain utterances and touches or prosody patterns, which have been described earlier in chapters 2 and 4.

5.6.3 Recognition

After the training, the trained models and the trained association matrix can be used for recognizing commands. Like the training, the recognition works in two steps: First, the utterance is preprocessed and recognized by the trained speech recognizer, the prosody recognizer and the touch recognizer. The speech recognizer returns the then most likely sequences of recognized utterance parts along with their confidence values. Then the system determines, based on the learned command patterns, which of these utterance parts are parameters and which of them belong to the command pattern.

Finally, it uses the learned association matrix to determine the meaning of the command pattern or patterns using the recognition results from speech and touch. If multiple utterances or touches occur, while one command is expected, all utterances are combined to get the symbol with the highest associative strength. After the most likely meaning of the command has been determined, the system looks up the meaning of the parameters in the association matrix. The recognition result is constructed from the symbolic representations of commands and objects with the highest combined associative strength to the recognized sequence of utterance parts.

Recognizing feedback is done in exactly the same way, as described in the previous chapter, if the system can determine, based on the situation, whether commands or feedback are expected. If the system needs to distinguish between feedback and commands without external information, only based on the stimuli from the user, prosody can help to improve the recognition result. The system first tries to recognize the stimuli from the user separately with the trained speech and touch models and the association matrix for feedback and with the trained speech and touch models and the association matrix for commands.

The prosody recognizer is then applied to the speech utterances and returns either the symbol "command", "positive feedback" or "negative feedback". The associative strength to the symbol "command" is added to the associative strength, that is returned from the

recognition as a command, the associative strengths to the symbols "positive feedback" and "negative feedback" are added to the associative strengths of positive and negative feedback of the feedback recognizer. Now the model with the highest modified associative strength is selected as the recognition result.

The system can use available world knowledge when creating the recognition grammar. Usually not all possible parameters make sense for all possible commands. If information is available on which parameters can be used with which commands, the system can use the association matrix to create a grammar that allows only HMMs, that are associated with possible parameters to be inserted into a model sequence for a command.

5.7 Experiments

The experimental setting is shown in Fig. 5.5. The system recorded speech using a close-talk microphone. Video was recorded for later integration of gesture recognition.

Figure 5.5: AIBO performing "virtual living room" task.

The participants were instructed to teach the robot in two phases. In the first phase, they were asked to teach object and place names to the robot. After the object learning has finished, the experiment continues with the teaching of commands. The users are instructed to utter commands, which match the situation shown in the "virtual living room" scene, and

give positive or negative feedback depending on whether the robot has reacted correctly or not.

The learning algorithm was evaluated with data from ten participants (7 male, 3 female). Five of them interacted with the pet-robot AIBO and five of them interacted with the humanoid robot. They used the task, explained in section 3.5, to teach object names and parameters to the robot. Every participant interacted with the robot for roughly 45 minutes until every object name and command was trained ten times. The language used in the experiments was Japanese.

The system was trained and evaluated with the recorded data using 10-fold cross evaluation. It was evaluated under three different conditions:

- In the first condition, training and evaluation data were checked for false examples, that were caused by users misinterpreting the situation, shown in the virtual living room, training samples, that were actually breathes, but misdetected as speech by the voice activity detection, utterances like "hmm", "ah" etc. Only training examples, that were clearly false were removed. Implicit commands or feedback were still used for training and recognition. With this manual preprocessing of the data, the recognition rate for commands was 84.45% with a standard deviation of 8.23%.
- In the second condition, the uncorrected data was used for training while the corrected data was used for evaluation. In a real world teaching scenario the raw audio data and the expected meanings are the only information available for training. However, the system should recognize the correct commands, even if there have been some incorrect ones during the training. Therefore it can be assumed that this way of determining the recognition rate is the best approximation of the real world performance of a robot trained with the proposed method. In this case, the recognition rate dropped to 80.89% with a standard deviation of 7.23%.
- In the third condition, both, training and evaluation data, were not checked or modified manually but used as recorded during the experiments. That means the error rate contains not only speech recognition problems but also other kinds of errors, such as wrong or no commands uttered by the participants, incorrect speech onset detection, noise from the robot's motion etc. The recognition rate in this case was 75.61% with a standard deviation of 8.52%.

The accuracies for the individual speakers and conditions are shown in Fig. 5.6.

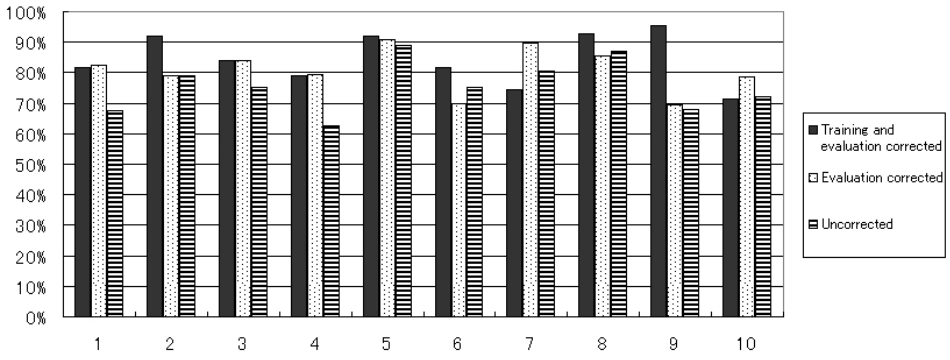

Figure 5.6: Recognition rates for the participants.

"The best theory is inspired by practice and the best practice is inspired by theory."

Donald Knuth

6 Implementation of the framework

6.1 Overview

This chapter gives an overview over the architecture and implementation of the system. The system consists of four different components, which communicate via TCP/IP. This architecture was chosen, because it is easily extensible and can be distributed between different machines.

The focus of the actual implementation of the system was to develop a framework for conducting experiments that is easy to adapt to new tasks and to different robots or virtual agents.

6.2 Architecture overview

The basic architecture of the system is shown in Fig. 6.1. It is implemented using a client-server based architecture consisting of four components which are linked via TCP/IP. It consists of the **task server**, the **perception server**, the **robot controller** and **the robot** itself. The task server and the robot controller are connected through a TCP/IP connection. The robot controller is also connected via TCP/IP to the perception server and to the robot.

6.2.1 The task server

The task server provides the display and handling of the playfield, an evaluation function for the robot's moves as well as the opponents' artificial intelligence in case of a game for multiple players. It can use a projector or a computer screen to display the playfield.

The task server provides all necessary background information on the state of the training task, about which moves are good or bad etc. to the robot controller. The robot controller can also request tasks from the task server, e.g. to visualize a certain command, that the user is expected to give to the robot.

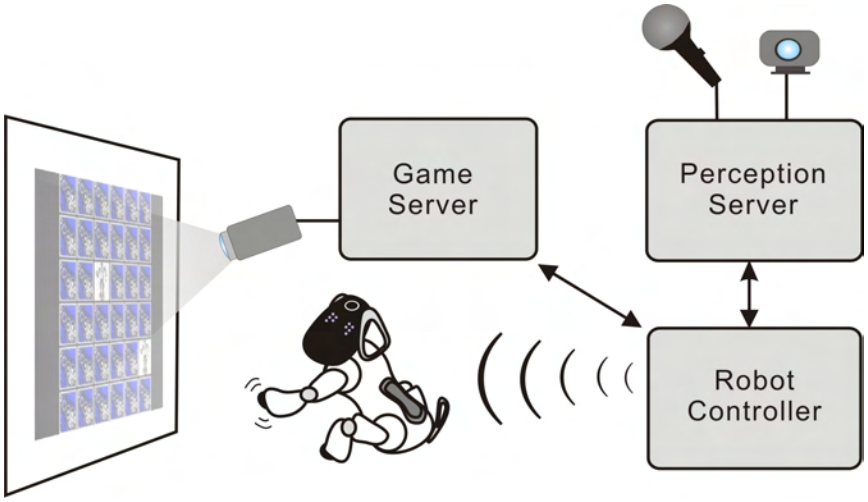

Figure 6.1: Architecture of the experimental framework.

6.2.2 The perception server

The perception server records and processes audio and video data of the user's interaction. It receives data from the robot's touch sensors, video data from two Logitech Fusion web cameras as well as audio data from a wireless lavalier microphone that is attached to the user's clothes or from a headset microphone. The data from different modalities is synchronized and stored, while the information, which is extracted from the audio and video data streams is sent to the robot control software. Learning to interpret the user's behavior using the method described in chapters 4 and 5 takes place in the perception server

6.2.3 The robot controller

The robot controller is connected to the task server as well as the perception server and the robot. It uses information about the task state, which are received from the task server, to calculate the next moves of the robot and sends control commands to the robot. Moreover, it uses information from the perception server in order to assess whether interaction has been perceived, so that it can react appropriately.

The AIBO Remote Framework [2] is used by the robot control software for wireless control of AIBO and for reading its sensor data. The robot controller is the only component that needs to be adapted to use other robots or agents with the learning framework.

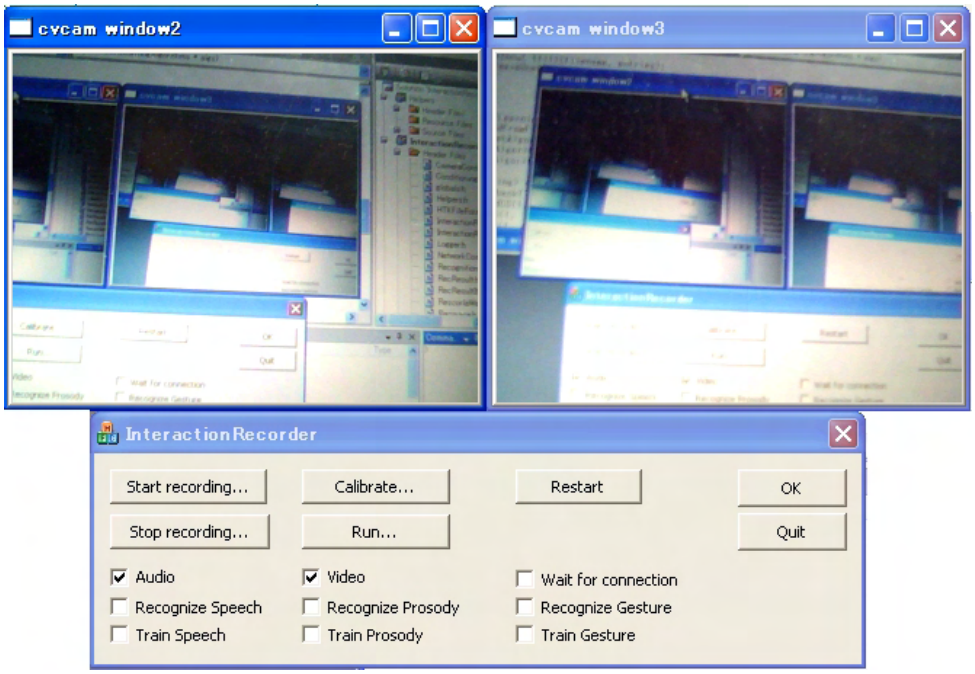

Figure 6.2: User interface of the perception server.

The robot controller was modified to communicate with the humanoid robot. However, details of the connection and control of the humanoid robot cannot be included in this thesis for confidentiality reasons.

6.2.4 The robot

An AIBO ERS-7 [26] as well as a child-sized the humanoid robot was used for the experiments.

Aibo is a dog-shaped robot which has roughly the size of a cat. It possesses twenty degrees of freedom and has touch sensors on its head and back as well as under each of its feet. Moreover, it is equipped with a camera as well as stereo microphones. It is shown in Fig. 6.4.

The humanoid robot is 120 cm tall, which is roughly the size of an 8-year-old child. It possesses 26 degrees of freedom.

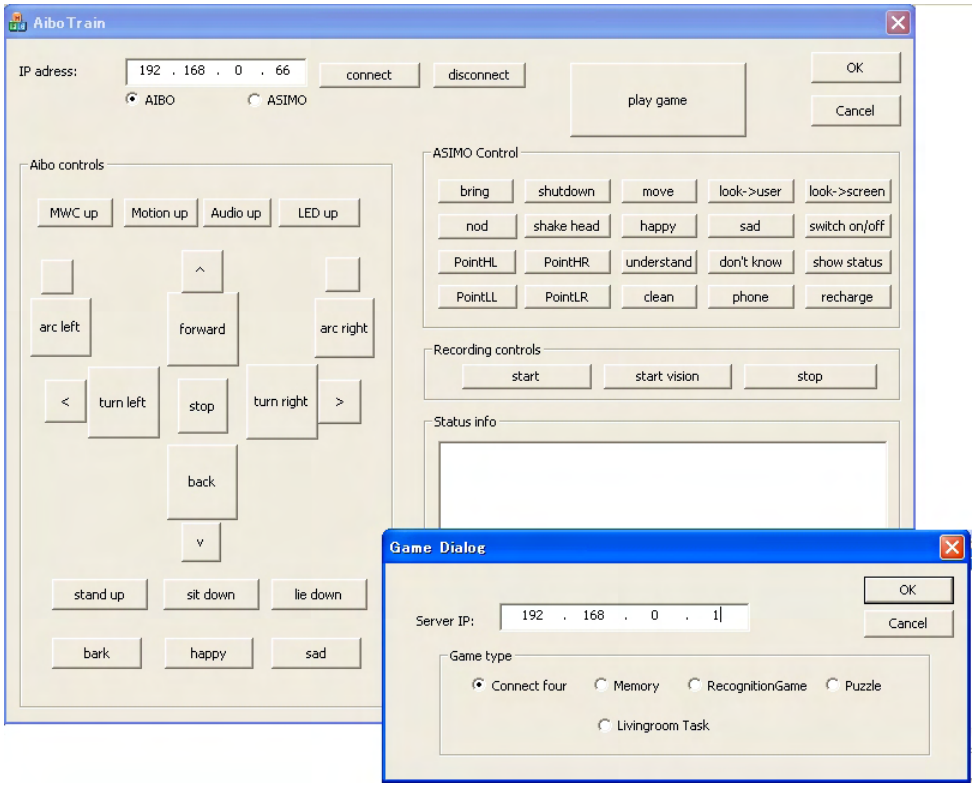

Figure 6.3: User interface of the robot controller.

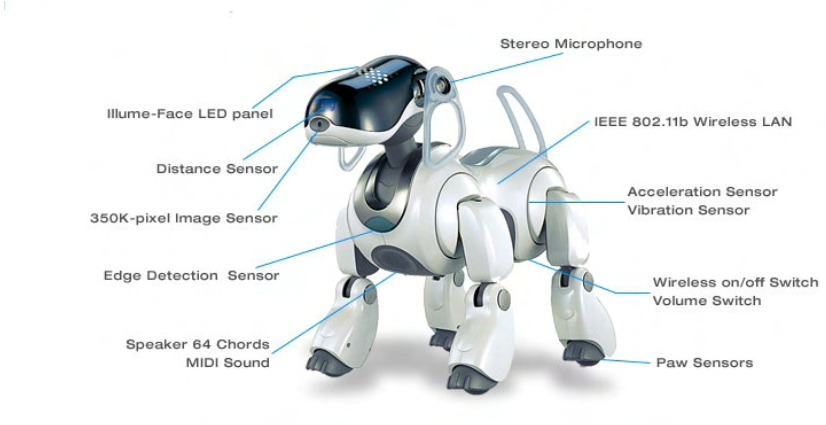

Figure 6.4: The pet-robot AIBO

6.3 Implementation details of the learning method

The framework was implemented in Java and C++. The robot controller and the perception server both run on a Windows PC and were implemented in Visual C++. A platform dependent solution using C++ has been selected, as these components need to process audio and video data quickly and also need to use OpenCV [12] and the Intel Performance Primitives [37] as well as the Aibo Remote Framework [1] which are only available as C++ libraries.

The task server as well as the component for offline training and evaluation of HMMs has been implemented in Java. This way they can be used platform-independently. As the task server does not have to perform any time-consuming calculations and the training and evaluation are performed offline, performance is not critical.

6.3.1 The HTK toolkit

The Hidden Markov Model Toolkit (HTK) [89], the HMM implementation, used in this thesis, was developed at the engineering department of Cambridge University. It consists of various tools for training and recognition using Hidden Markov Models and is widely used in research as well as for commercial purposes.

In this work, the tools HInit and HErest for HMM initialization and Baum-Welch Re-estimation and HVite for Viterbi recognition have been used. The tool HParse is used to convert the grammar files into a format that the other HTK tools can handle. The basic algorithms for training and recognition with HMMs are described in section 4.4.1.

HInit

HInit initializes a Hidden Markov Model using speech data by uniformly segmenting the data for training and using it to initialize the output distributions of the HMMs. The transition probabilities for all states are by default initialized to the same values for all outgoing transitions of a state.

The system uses HInit for initializing the HMMs of the prosody recognizer. As the speech recognizer creates new HMMs by concatenating existing monophone models, it does not have to rely on HInit for initializing the speech HMMs.

HErest

HErest re-estimates the parameters of an initial HMM using a set of training data based on the Baum-Welch re-estimation method. To do so, it needs the following information as input:

- A list of training data files
- The initialized or pre-trained HMM(s) to be re-estimated
- A list of labels corresponding to the training data files

It returns a set of trained HMMs, which can then be used for recognition.

HVite

HVite is the HTK implementation of the Viterbi algorithm for continuous speech. It is used in the proposed learning algorithm to recognize utterances either based on phoneme sequences or based on the already learned models.

HVite takes the following information as input:

- The audio file to be recognized (or another data file in HTK format)
- The grammar for the recognizer
- The dictionary for the recognizer
- The trained HMMs

It returns a transcription of the recognized file, which is then further processed by the system to determine the final recognition result based on the stored associations of the recognized models to feedback, object names and commands.

HParse

The proposed learning algorithm generates all grammars for the different recognizers in Extended Backus-Naur Form (EBNF), which is a standard notation for expressing context-free grammars. HParse is used in the system to convert a grammar written in EBNF format to a grammar in lattice format that can be used by the HTK.

6.3.2 The HTK format

For speech data, HTK can process files in various formats, including standard wave files, from which MFCCs can be calculated automatically.

When training HMMs for custom data, such as the ones used for prosody recognition in this thesis, the data has to be stored in HTK format. Files in HTK format consist of a header and data part. The header is twelve bytes long. It contains the following information:

- The number of samples as a four-byte integer
- The sampling period as a four-byte integer in 100 ns units

- The sample size as a two-byte integer
- The format as a two-byte integer

There are different formats, that can be specified for speech signals, such as waveforms, MFCCs etc. The format, that is used for custom data is "USER", which is specified by the number nine (001001), given in the first six bits and 0 in the remaining ten bits.

After writing the header, the entries of the feature vectors of the prosody data are written to the file in float format. The number of samples must not exceed the value, specified in the header.

"The important thing is not to stop questioning."
Albert Einstein (1879 - 1955)

7 Conclusion and outlook

7.1 Overview

The previous chapters presented an approach for learning multimodal user feedback and commands through natural, situated human-robot interaction in virtual training tasks. Based on the findings from three user studies, requirements for the learning algorithm were identified and an extensible framework for teaching a robot through virtual training tasks implemented. In this chapter, the findings from the user study as well as the results from the training are discussed, limitations are identified and possible future directions of this work are outlined.

7.2 User's ways of giving feedback and commands

In a series of three experiments, different properties of user feedback and users' commands in human-robot interaction were analyzed. The findings from the experimental studies indicate that users react rather sensitive to restrictions in acceptable reward behavior, and that there is a need for techniques, like the one, proposed in this thesis, that allow a robot to process reward given freely by the user.

The studies also found that users' ways of giving feedback differed between users and also differed, depending on the robot, while it was relatively similar for one user between different training tasks. Moreover, detailed information on how users give positive and negative feedback to a robot were determined by analyzing the audio and video data from the training tasks.

In the analysis of commands given to the humanoid and the pet-robot it was found that commands, like feedback, varied strongly between users, while no significant dependence on the appearance of the robot could be observed. Differences between users were found for different characteristics of command utterances, such as the use of polite expressions or the use of explanations when giving feedback.

These findings lead to the assumptions, that learning to understand users' feedback and commands is useful and feasible, because the observed feedback and commands would be very difficult to hard-code for the system designer but still do not vary too strongly for one person, so that they can be learned by the robot from its user in a reasonable amount of time.

One important question that has not been fully answered by the user studies, presented in this thesis, is the similarity of user behavior between virtual tasks and real world tasks. The proposed approach assumes that user's commands and feedback are actually similar between real and virtual tasks, so that interaction patterns, learned in a virtual tasks can be transferred to a real tasks. However, confirming this for the proposed command-learning task would require user studies with robots, that can actually perform useful, real world household tasks.

In order to obtain a first, rough idea about whether the assumption holds or not, the study on feedback learning contained one control task, which was not a virtual task but required real-world performance by the robot. In this task, the user had to teach "dog-like" commands to the AIBO pet-robot, such as "sit down", "stand up" etc. No visible differences in users' interaction between the virtual game tasks and the dog training task in the experiment were observed. However, more experiments would be necessary to conclusively answer the question, if and under which conditions it is possible to train a robot for a real world task using a virtual task.

7.3 Results of the learning algorithms

Chapters 4 and 5 described and evaluated a method for learning a user's feedback and commands for human-robot interaction, based on the findings from the user studies. The performance for interpreting speech, prosody and touch feedback as well as commands from a human can be considered sufficiently reliable for being used to teach a robot, for example, by reinforcement learning and instruct it using simple commands.

The first version of the algorithm, which was implemented for learning feedback, has been extended for learning parameterized commands for human-robot interaction. The main restriction of the proposed approach is that it is only applicable as long as the number of commands, that the robot needs to understand, does not grow too large. Otherwise, the learning process for commands could become too time-consuming for real-world use. The learning of object names with the proposed approach can continue after the training phase in a real environment, provided the robot can visually identify objects. However, the learning of commands heavily relies on the virtual training tasks to make the user utter the commands that the robot wants to learn. Determining experimentally how much time for training is acceptable for users, would require a long-time study with real, useful household

or entertainment robot. Therefore this question cannot be answered conclusively within the scope of this thesis.

Based on analyzing the typical misrecognitions and the amount of training data, used for each HMM, the main reason for misrecognitions appears to be the relatively low amount of training data per HMM especially for commands but also for participants who gave very variable feedback. For persons with a very variable way of uttering commands, there were often only one or two examples of an utterance available to train a HMM. In such cases, the concatenated monophone models had to be used as they were and could not be retrained with the user's speech data, as the current system cannot train HMMs incrementally.

It can be expected that taking more time for training, using incremental training or applying approaches for speaker adaptation to the generated phoneme sequences instead of simply retraining the generated HMMs with the utterances would result in an improved recognition accuracy.

One potential drawback of the proposed approach is that the robot has to complete a training phase with every user who wants to interact with it in order to adapt to the user's way of giving rewards. However, a typical pet robot or entertainment robot only interacts with a very limited number of persons in a household and usually interacts with the same users frequently and for a long time. Therefore, user-specific adaptation can be considered an efficient means to facilitate interaction, even though it needs some initial training effort.

The learning method, proposed in this thesis, focuses only on a small part of the whole process, that is covered, for example, by the system, developed by Iwahashi et al. [42] [43]. Capabilities like disambiguation of ambiguous utterances, performance by manipulating objects in a real-world task, visually recognizing objects or actions or answering questions have not yet been implemented and are not in the scope of this thesis.

While the system does not enforce any restrictions on the grammar of the user's utterances, it can only map utterances to simple meanings, consisting of a command with up to two parameters or positive/negative feedback. While sentences and object names, learned by the robot, may contain adjectives or prepositions, the system does not learn to understand the meaning of adjectives or prepositions independent from the learned sample utterances. It is also not possible to give more than one command in one utterance or connect parameters with "and" like in "Clean the floor and the table".

7.4 Outlook

While the system learned the participants' commands from scratch during interaction with the user, the approach can be used very easily to adapt an existing predefined set of commands to a user by only learning commands that are not matched well by the existing model. In this case the utterance-based recognizer and association matrix would not be

empty upon initialization but filled with the predefined utterances. The system would then start from the known utterances and generate new models only if unknown utterances are observed.

Currently the system is working offline. It first records all training data and then trains the models based on the recorded data. Using incremental training of the HMMs, the system can be made capable of online learning through actual communication with a user during the training phase. Main issues that need to be targeted for implementing an online version of the algorithm are the clustering of the training samples for prosody as well as the incremental re-training of the HMMs for speech and prosody.

At the moment, the system can only deal with names of objects or places, not with more detailed descriptions, containing attributes or prepositions. "The blue cup" or "the cup on the table" would be learned as one object name. In order to allow for more flexible instructions from the user, it is necessary to extend the learning method to enable the system to learn prepositions and certain attributes, such as colors, which are commonly used to distinguish different objects of the same class. The algorithm is able to handle this almost without modifications, by using the extensions, which have been implemented for command learning. In addition to segmenting utterances into known object names and unknown command parts, and then learning the unknown command-parts in the command learning phase, the system would perform another learning step with the same algorithm, in which it learns to segment utterances into previously learned object names and unknown parts, which correspond to modifiers, such as “blue” or “big” to train new HMMs. Combinations of object names and modifiers could then be used in the command learning and recognition stage.

For the user, this would result in an additional training phase between the object learning phase and the command learning phase, in which the system learns attributes and prepositions used for distinguishing objects. In this phase, the user and the robot have to perform a task, in which the modifiers to be learned are used to disambiguate object names. For example, if the robot has learned to understand the word “box” in the first stage, the system could show a virtual scene with multiple boxes with different colors and sizes and ask the user teach the robot to pick the right one using modifiers such as “the big box” or “the red box”. As in the command learning task, the system would then search for the known parts of the user’s utterance, that correspond to the already known object names. Then it would use the unknown parts of the utterance to train an HMM and associate it with the expected modifier, such as “big” or “red”.

Pointing gestures could be integrated and used to disambiguate or even replace spoken object references. Using the learned command patterns, which contain the expected parameters as well as their order, the system could be extended to allow replacing parameters of commands by pointing gestures or use pointing gestures as additional information, when the spoken parameter names are ambiguous.

Another possible extension would be learning to understand users’ states and implicit user feedback for a robot in addition to the explicit commands and feedback, which are the

focus of this thesis. In human communication, information is not only conveyed by explicit verbal utterances, but implicit, non-verbal information also plays an important role to ensure smooth and pleasant interaction. In addition to learning, how a user expresses feedback, object names and commands explicitly, it is therefore desirable to also learn to recognize users' internal state by processing implicit feedback from the user to the robot. This would allow the system to detect communication problems and adapt its behavior to the state of the user in order to avoid disturbances for the user and recover from communication problems smoothly.

While this thesis deals with the application of the learning algorithm for learning commands and feedback for natural human-robot interaction, other applications are possible, as long as training utterances or other stimuli, that can be modeled by HMMs, along with their meanings are provided to the learning algorithm. The proposed algorithm for learning feedback, given to a pet-robot, has already successfully been applied to classify positive and negative answers from users in a hotel reservation task.

Bibliography

[1] AIBO Remote Framework http://openr.AIBO.com

[2] Aibo Website, http://eu.aibo.com

[3] Hiyan Alshawi, "Effective utterance classification with unsupervised phonotactic models", Proceedings of NAACL '03, 2003, pp. 1-7

[4] A. Austermann, S. Yamada: ""Good Robot, Bad Robot" - Analyzing User's Feedback in a Human-Robot Teaching Task", In Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication 2007 (RO-MAN 08), pp. 41-46, 2008

[5] A. Austermann, S. Yamada: "Learning to Understand Multimodal Rewards for Human-Robot-Interaction using Hidden Markov Models and Classical Conditioning", In Proceedings of the IEEE World Congress of Computational Intelligence (WCCI 2008), pp. 4096-4103, 2008

[6] A. Austermann, S. Yamada, Learning to Understand Parameterized Commands through a Human-Robot Training Task, IEEE International Symposium on Robot and Human Interactive Communication (ROMAN'09), (2009), 757-762

[7] C. Balkenius and J. Morn: "Computational models of classical conditioning: a comparative study.", In Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior, pp. 348 - 353, 1998

[8] D. H. Ballard, C. Yu: "A multimodal learning interface for word acquisition" , In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 1520-6149, 2003

[9] H. Benedict, "Early lexical development: Comprehension and Production", Journal of Child Language, 21, pp. 85-124, 1994

[10] B. Blumberg, M. Downie, Y. Ivanov, M. Berlin, M.P. Johnson, B. Tomlinson: "Integrated Learning for Interactive Synthetic Characters", In Proceedings of the 29th annual conference on Computer graphics and interactive techniques (SIGGRAPH 2002), 417 - 426, 2002

[11] R. Bolt, ""Put-that-there": Voice and gesture at the graphics interface", SIGGRAPH 80 Proceedings of the 7th annual conference on Computer graphics and interactive techniques (1980), Volume: 14, Issue: 3, ACM, pp 262-270

[12] G. Bradsky, A. Kaehler: Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly, first edition, 2008

[13] C. Breazeal, L. Aryananda: "Recognition of Affective Communicative Intent in Robot-Directed Speech", Autonomous Robots, Volume 12 , Issue 1, 2002, pp. 83 - 104

[14] W. F. Brewer: "Bartlett's Concept of the Schema and Its Impact on Theories of Knowledge Representation in Contemporary Cognitive Psychology." Bartlett, Culture and Cognition, ed. Akiko Saito. Hove, Eng.: Psychology Press, (2000), 69-89

[15] B. Burns, C. Sutton, C. Morrison, P. Cohen, "Information Theory and Representation in Associative Word Learning", Epigenic Robotics (EpiRob 2003), 2003

[16] D. Byrd, T.H. Mintz: "Discovering speech, words and mind", Wiley-Blackwell Publishing, 2010

[17] A. de Cheveigne, "'YIN, a fundamental frequency estimator for speech and music'", 2001

[18] C. M. Connine, C. Clifton: "Interactive use of lexical information in speech perception", Journal of Experimental Psychology: Human Perception and Performance, Vol 13(3), pp. 291-299, 1987

[19] Mac Dorman, K. F. Androids as an experimental apparatus: Why is there an uncanny valley and can we exploit it? CogSci-2005 Workshop: Toward Social Mechanisms of Android Science, (2005), 106-118

[20] W. K. Estes, "Toward a statistical theory of learning", Psychological Review. Vol 101(2), Apr 1994, 282-289.

[21] M. W. Eysenck, M. T. Keane: "Cognitive Psychology - A Student's Handbook", Psychology Press, 2005

[22] G. A. Fink, J. Fritsch, S. Hohenner, M. Kleinehagenbrock, S. Lang, and G. Sagerer: "Towards multi-modal interaction with a mobile robot", Pattern Recognition and Image Analysis, 14(2):173-184, 2004

[23] J. A. R. Fonollosa, W. A. Ainsworth, E. Mousset, "'A Comparision of several recent methods of fundamental frequency and voicing decision estimation'", Proc. ICSLP '96, 1273-1276, 1996

[24] D. H. Friedmann: "Pseudo-maximum-likelihood speech pitch extraction", IEEE Trans. ASSP-25 (3), pp. 213-221, 1978

[25] Matteo Frigo and Steven G. Johnson, "The Design and Implementation of FFTW3," Proceedings of the IEEE 93 (2), 216-231, Invited paper, Special Issue on Program Generation, Optimization, and Platform Adaptation, 2005

[26] M. Fujita and H. Kitano: "Development of an Autonomous Quadruped Robot for Robot Entertainment", Autonomous Robots, No. 5, Vol. 1, pp.7-18, 1998

[27] W. F. Ganong: "Phonetic categorization in auditory word perception." Journal of Experimental Psychology: Human Perception and Performance, 6, 110-125, 1980

[28] D. Gentner: "Why nouns are learned before verbs: Linguistic relativity versus natural partitioning", Language development, Vol. 2. Language, cognition and culture, Erlbaum, Hillsdale, 1983

[29] J. Goetz, S. Kiesler, and A. Powers, Matching robot appearance and behavior to tasks to improve human-robot cooperation, IEEE Workshop on Robot and Human Interactive Communication (ROMAN'03), (2003), 55 - 60

[30] S. Gokhun Tanyer and Hamza Ozer, "Voice Activity Detection in Nonstationary Noise" in IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING, VOL. 8, NO. 4, JULY 2000, pp. 478-482

[31] A. L. Gorin and D. Petrovksa-Delacretaz and G. Riccardi and J.H. Wright, "Learning Spoken Language without Transcriptions", Proceedings of ASRU '99, 1999

[32] Manuel Giuliani, Michael Kasecker, Stefan Schwarzler, Alexander Bannat, Jurgen Gast, Frank Wallhoff, Christoph Mayer, Matthias Wimmer, Cornelia Wendt, and Sabrina Schmidt. MuDiS - a multimodal dialogue system for human-robot interaction. In Proceedings of the International Workshop on Cognition for Technical Systems, Munich, Germany, October 2008.

[33] A. Haasch , S. Hohenner, S. Huewel, M. Kleinehagenbrock, S. Lang, I. Toptsis, G. A. Fink, J. Fritsch, B. Wrede, G. Sagerer, "BIRON - The Bielefeld Robot Companion" (In E. Prassler, G. Lawitzky, P. Fiorini, and M. Haegele, editors), Proc. Int. Workshop on Advances in Service Robotics, pages 27-32, Stuttgart, Germany, May 2004. Fraunhofer IRB Verlag.

[34] Hartigan, J. A.; Wong, M. A.: "Algorithm AS 136: A K-Means Clustering Algorithm". Journal of the Royal Statistical Society, Series C (Applied Statistics) 28 (1), pp. 100-108, 1979

[35] F. Hegel, M. Lohse, B. Wrede: "Effects of Visual Appearance on the Attribution of Applications in Social Robotics", IEEE International Symposium on Robot and Human Interactive Communication (ROMAN'09), (2009), 64-71

[36] J. Huttenlocher, P. Smiley: "Early word meanings: The case of object names", Language Acquisition: core readings, pp. 222 - 247, MIT Press, Cambridge, 1994

[37] Intel Integrated Performance Primitives for Intel Architecture, Reference Manual, Volume 1: Signal Processing, March 2009

[38] IRobot Roomba Website, http://www.irobot.com/

[39] IRobot Scooba Website, http://www.irobot.com/uk/home_robots_scooba.cfm

[40] N. Iwahashi: "Active and Unsupervised Learning for Spoken Word Acquisition Through a Multimodal Interface", In Proceedings of the 13th IEEE international workshop on robot and human interactive communication (RO-MAN 2004), pp. 437

- 442, 2004

[41] N. Iwahashi, "Robots that Learn Language: A Developmental Approach to Situated Human-Robot Conversation", in Sanker, N. ed. Human-Robot Interaction, pp.95-118. I-Tech Education and Publishing, 2007.

[42] Iwahashi, N., Taguchi, R., Sugiura, K., Funakoshi, K., Nakano, M. "Robots that Learn to Converse: Developmental Approach to Situated Language Processing" Proc. Int. Sym. Speech and Language Processing, pp.532-537, 2009

[43] Taguchi, R., Iwahashi, N., and Nitta, T. "Learning Communicative Meanings of Utterances by Robots," In Hattori, H., Kawamura, T., Ide, T., Yokoo, M., Murakami, Y. (eds.), New Frontiers in Artificial Intelligence, Springer, LNCS / LNAI 5447, pp. 62-72, 2009

[44] The Julius Speech Recognition Project: http://julius.sourceforge.jp/

[45] P.W. Jusczyk, "From general to language-specific capacities: the WRAPSA model of how speech perception develops", Journal of Phonetics, 21, pp. 3-28, 1993

[46] R. el Kaliouby, P. Robinson (2004). Mind Reading Machines: Automated Inference of Cognitive Mental States from Video. Proceedings of The IEEE International Conference on Systems, Man and Cybernetics.

[47] F. Kaplan, P.-Y.-Oudeyer, E. Kubinyi, A. Miklosi "Robotic clicker training." Robotics and Autonomous Systems 38(3-4), 2002, 197-206

[48] Z. K. Kayikci, H. Markert, G. Palm: "Neural Associative Memories and Hidden Markov Models for Speech Recognition", In Proceedings of the International Joint Conference on Neural Networks (IJCNN 2007), pp. 1572 - 1577, 2007

[49] Analysis of Humanoid Appearances in Human-Robot Interaction, IEEE TRANSACTIONS ON ROBOTICS, VOL. 24, NO. 3, (2008), 899 - 906

[50] Kelley, J. F.: "An iterative design methodology for user-friendly natural-language office information applications". ACM Transaction son Office Information Systems, 2, 26-41, 1984

[51] T. Komatsu, S. Yamada: Effect of Agent Appearance on People's Interpretation of Agent's Attitude, CHI-2008 Work-in-Progress, (2008), 2919 -2924

[52] Sarah Kriz, Gregory Anderson, J. Gregory Trafton: "Robot-Directed Speech: Using Language to Assess First-Time Users' Conceptualizations of a Robot", 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI 2010), (2010), 267-275

[53] C. Lee, Y. Xu, "Online Interactive Learning of Gestures for Human/Robot interfaces", IEEE Int. Conf. on Robotics and Automation, pp 29822987, 1996.

[54] C. Li, G. Biswas: "A Bayesian Approach to Temporal Data Clustering using Hidden Markov Models", In Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), pp. 543-550, 2000

[55] E. S. Kim, B. Scassellati: "Learning to Refine Behavior Using Prosodic Feedback", In Proceedings of the 6th IEEE International Conference on Development and Learning (ICDL 2007), pp. 205-210, 2007

[56] B. Lowenkron: "Word meaning: A verbal behavior account", Presented at the annual convention of the Association for Behavior Analysis, Washington DC, May, 2000

[57] F. Mondada, E. Franzi and P. Ienne: "Mobile robot miniaturisation: A tool for investigation in control algorithms", Experimental Robotics III, In Proceedings of the 3rd International Symposium on Experimental Robotics, pp.28-30, 1993

[58] A. Nogueiras, A. Moreno, A. Bonafonte, J. B. Marino: "Speech Emotion Recognition Using Hidden Markov Models", In Proceedings of the 7th European Conference on Speech Communication and Technology (EUROSPEECH), pp. 2679-2682, 2001

[59] T. L. Nwe, S. Foo, S. Wei; L. De Silva, "Speech emotion recognition. using hidden Markov models", Speech communication 41,4, 2003

[60] S. Oviatt, "Ten myths of multimodal interaction" Communications of the ACM, Vol. 42 , No. 11, pp. 74 - 81, 1999

[61] I. P. Pavlov: "Conditioned Reflexes: An Investigation of the Physiological Activity of the Cerebral Cortex (translated by G. V. Anrep), Oxford University Press, 1927

[62] Pleo Website, http://www.pleoworld.com

[63] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery: "Numerical Recipes: The Art of Scientific Computing", Third Edition, Cambridge University Press, 2007

[64] J. G. Proakis, D. K. Manolakis: "Digital Signal Processing (4th Edition)", Prentice Hall, 2006

[65] L. R. Rabiner, "'A Comparative Study of several Pitch Detection Algorithms'", 1973

[66] L. R. Rabiner "'A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition'", Proceedings of the IEEE Vol. 77, 1989

[67] R. Rescorla, A. Wagner: "A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement.", Classical Conditioning II: Current Research and Theory (Eds Black AH, Prokasy WF) New York: Appleton Century Crofts, pp. 64-99, 1972

[68] http://www.robothalloffame.org/unimate.html, Unimate, the first Industry Robot, presented in the Robot Hall of Fame

[69] D. Roy, "Grounded Spoken Language Acquisition: Experiments in Word Learning". IEEE Transactions on Multimedia, 5(2) pp.197-209, 2003

[70] D. Roy, "Learning Words from Sights and Sounds: A Computational Model", Ph.D. thesis, Massachusetts Institute of Technology

[71] S. J. Russel, P. Norvig, "Artificial Intelligence: a Modern Approach", Prentice Hall, 2nd edition, 2002

[72] L. Steels and F. Kaplan, "AIBO's first words : The social learning of language and meaning", Evolution of Communication, 4(1) pp. 3-32 , 2001

[73] B. F. Skinner: "Verbal Behavior". Copley Publishing Group, Acton 1957

[74] Staats, C. K., and Staats, A. W. Meaning established by classical conditioning. Journal of Experimental Psychology, 1957, 54, 74-80

[75] R. Standke, "'Methoden der digitalen Sprachverarbeitung in der vokalen Kommunikationsforschung"', Peter Lang Verlag, 1993

[76] S. S. Stevens, J. Volkman; E. Newman, (1937). A scale for the measurement of the psychological magnitude of pitch. Journal of the Acoustical Society of America, 8 (3), pp. 185-190

[77] R. Stiefelhagen, C. Fugen, P. Gieselmann, H. Holzapfel, K. Nickel and A. Waibel "Natural Human-Robot Interaction using Speech, Head Pose and Gestures" Proceedings. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. (IROS 2004). 2422- 2427 vol.3, 2004

[78] R. Stiefelhagen, H. Ekenel, C. Fugen, P. Gieselmann, H. Holzapfel, F. Kraft, K. Nickel, M. Voit, A. Waibel: "Enabling Multimodal Human-Robot Interaction fort the Karlsruhe Humanoid Robot" , IEEE Transactions on Robotics, Special Issue on Human-Robot Interaction, Vol. 23, No.5, October 2007

[79] A. L. Thomaz, and C. Breazeal: "Reinforcement Learning with Human Teachers: Evidence of feedback and guidance with implications for learning performance." In Proceedings of the 21st National Conference on Artificial Intelligence (AAAI '06), 2006

[80] A. L. Thomaz, G. Hoffman, and C. Breazeal. "Reinforcement Learning with Human Teachers: Understanding how people want to teach robots." Proceedings of the 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2006.

[81] I. Toptsis, S. Li, B. Wrede , and G. A. Fink: "A Multi-modal Dialog System for a Mobile Robot", Proc. Int. Conf. on Spoken Language Processing, volume 1, pages 273-276, 2004

[82] R. Tucker, "Voice activity detection using a periodicity measure, " IEE Proceedings,. Communications, Speech and Vision, vol. 139, no. 4, pp. 377 - 380, 1992.

[83] M. Ullerstam, M. Mizukawa, "Teaching robots behavior patterns by using reinforcement learning: how to raise pet robots with a remote control", SICE

[84] J. Wainer, D. J. Feil-Seifer, D. A. Shell, M. J. Mataric "Embodiment and Human-Robot Interaction: A Task-Based Perspective", , Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication 2007

[85] F. Wallhoff, T. Rehrl, J. Gast, A. Bannat, G. Rigoll: "Multimodal Data Communication for Human-Robot Interactions", IEEE International Conference on Multimedia and Expo, ICME, pp. 1146-1149 2009

[86] J. F. Werker, H. Henny, Yeung: "Infant Speech Perception Bootstraps Word Learning", Trends in Cognitive Sciences, Vol. 9, No. 11, pp. 519-527, 2005

[87] M. Xin, E. Sharlin: "Sheep and wolves: test bed for human-robot interaction.", In CHI '06 Extended Abstracts on Human Factors in Computing Systems, Computer/Human Interaction (CHI '06), 2006

[88] S. Yamada and T. Yamaguchi: "Training AIBO like a Dog", In Proceedings of the 13th International Workshop on Robot and Human Interactive Communication (RO-MAN 2004), pp. 431-436, Kurashiki, Japan, 2004

[89] S. Young et al., "The HTK Book" HTK Version 3, 2006 http://htk.eng.cam.ac.uk/

Appendices

1 Pilot study on user feedback

1.1 Instructions

Introduction
Welcome to this experiment! Thank you very much for your participation.

Please read the following instructions carefully. If there are points in the explanation, that remain unclear, or if you have any further questions, please ask the instructor. The experiment, we are going to conduct, deals with human-robot interaction in the field of service robots.

The robot, you will be interacting with, throughout the experiment, is a Sony Aibo robot. In this experiment, it is used to simulate a service robot that helps you with tidying up your room and will tidy up the objects found in the room in the way that you instruct it to.

The task of the robot is putting differently coloured objects to their designated places. The robot does not know, which object belongs where, so it has to learn the desired positions of the objects through your guidance.

You will receive a deck of cards which show the desired final configuration of objects on their places. Please instruct the robot to arrange the objects according to the configuration, depicted on the card. The position of objects, not shown on the card, is of no importance for the result of the experiment

Please follow the instructions below:

2) Take the card on top of the stack, read at it carefully and memorize the desired position of the objects as depicted on the card.

3) Put away the card and any other objects, you may hold in your hands, that might affect your movements.

4) Aibo will make a sound, when he is ready to receive instructions

5) You may use your voice as well as your body freely for instructing the robot

6) Please do not turn your back to the cameras.

7) If the robot does not know, what to do as a next step, please assist it, by giving more detailed instruction or repeating your instruction

9) The robot needs your feedback to accomplish its task. Please give positive and negative feedback for the robot's behaviour. More information on giving feedback is provided on a separate sheet of paper.

10) In case you forget the contents of the card or give incorrect instructions to the robot, please inform the instructor. He will put objects, that already have been tidied up, back to their original place and you can start the instruction again from the beginning of the card

11) When you finish a card, give feedback to the robot, that the task has been solved and inform the instructor, so that he can put the objects, that have been tidied up, back to the field. Then you can resume with the next card.

Reinforcement Principle:

All of your feedback to the robot that contains information whether the robot is performing correctly or incorrectly is considered as a reward.

There are basically two types of reward you can give to the robot: *positive* and *negative* reward. *Positive* reward will cause the robot to continue its behavior. *Negative* reward will cause the robot to change its behavior.

If the robot is performing in the way, you expect it to perform, please give *positive* reward. If it is not performing in the expected way, please give *negative* reward. Reward can be given at any moment during the interaction.

The choice of positive and negative reward is up to you. Please give the kind of feedback you like freely using voice, gestures or by touching the robot's head and back sensors.

Reinforcement Principle:

All of your feedback to the robot that contains information whether the robot is performing correctly or incorrectly is considered as a reward.

There are basically two types of reward you can give to the robot: *positive* and *negative* reward. *Positive* reward will cause the robot to continue its behavior. *Negative* reward will cause the robot to change its behavior.

If the robot is performing in the way, you expect it to perform, please give *positive* reward. If it is not performing in the expected way, please give *negative* reward. Reward can be given at any moment during the interaction.

Please decide, in what way you want to give positive reward. You can choose any combination of spoken words, gestures as well as touching the robots head and back sensors.

Please decide, in what way you want to give negative reward. You can choose any combination of spoken words, gestures as well as touching the robots head and back sensors.

Before starting to instruct the robot, please have your choice of behavior for positive and negative reward recorded. Please stick to the chosen behavior whenever you want to give positive or negative reward to the robot. Please do not change your reward behavior after the beginning of the experiment.

Reinforcement Principle:

All of your feedback to the robot that contains information whether the robot is performing correctly or incorrectly is considered as a reward.

There are basically two types of reward you can give to the robot: *positive* and *negative* reward. *Positive* reward will cause the robot to continue its behavior. *Negative* reward will cause the robot to change its behavior.

If the robot is performing in the way, you expect it to perform, please give *positive* reward. If it is not performing in the expected way, please give *negative* reward. Reward can be given at any moment during the interaction.

If you want to give positive reward to the robot, please touch its HEAD sensor.
If you want to give negative reward to the robot, please touch its BACK sensor.

1.2 Questionnaire

Experiment number: _____ Order_____

Questionnaire:

1) I was able to instruct the robot in a natural way (without feeling restricted in my movement or speech utterances)
Reinforcement Principle 1: absolutely agree 1..2..3..4..5 absolutely disagree

Reinforcement Principle 2: absolutely agree 1..2..3..4..5 absolutely disagree

Reinforcement Principle 3: absolutely agree 1..2..3..4..5 absolutely disagree

If not, what restrictions did apply?

_____

_____

_____

2) I would like to give feedback to a real world service robot in the same way
Reinforcement Principle 1: absolutely agree 1..2..3..4..5 absolutely disagree

Reinforcement Principle 2: absolutely agree 1..2..3..4..5 absolutely disagree

Reinforcement Principle 3: absolutely agree 1..2..3..4..5 absolutely disagree

If not, what would you like to change?

_____

_____

_____

3) Throughout the experiment, I was always sure about my next step to instruct the robot correctly.
Reinforcement Principle 1: absolutely agree 1..2..3..4..5 absolutely disagree

Reinforcement Principle 2: absolutely agree 1..2..3..4..5 absolutely disagree

Reinforcement Principle 3: absolutely agree 1..2..3..4..5 absolutely disagree

If not, what kind of situations remained unclear?

_____

_____

_____

4) Apart from the things, mentioned above, did you notice any problems during the communication with the robot and what problems did you notice?

Reinforcement Principle 1: absolutely agree 1..2..3..4..5 absolutely disagree

Reinforcement Principle 2: absolutely agree 1..2..3..4..5 absolutely disagree

Reinforcement Principle 3: absolutely agree 1..2..3..4..5 absolutely disagree

_____

_____

_____

4) Apart from the things, mentioned above, did you miss anything in the interaction with the robot, either in the robot's behaviour or in the systems understanding of your instructions, that would make interaction easier for you?

_____

_____

_____

5) Apart from the things, mentioned above, are there any thoughts about the research topic, the experiment or its execution, that you would like to share?

_____

_____

_____

_____

_____

_____

6) My preferred way of giving reward to the robot was

O Reinforcement Principle 1
O Reinforcement Principle 2
O Reinforcement Principle 3

2 Study on learning user feedback

2.1 Instructions

Thank you very much for participating in this study.
実験に参加してくれてありがとうございます。

The experiment comprises four different subtasks. In each of the subtasks, please teach the robot to accomplish the given tasks by providing positive and negative feedback to it.
実験は 4 件の課題を含みます。各課題で正のフィードバックと負のフィードバックにより、ロボットに課題を実行する方法を教えてください。

For giving feedback to the robot, you can use voice and gesture freely. You can also use the touch sensors, located on the robot's head and back to provide feedback. However, please use simple sentences, when talking to the robot.
ロボットにフィードバックを与える為、音声や身ぶりを自由に使って下さい。ロボットのタッチセンサーでもフィードバックを与えることが可能です。タッチセンサーがロボットの頭と背中にあります。ただし、ロボットと話すと、簡単な文書で話してください。

After the experiments are finished, please answer the questionnaire.
実験の後で、アンケートを答えて下さい。

Task1: Comparing Images (画像を比べる)

Please teach the robot to pick the image that is equal to a sample image.
サンプル画像と同じ画像を選ぶことをロボットに教えてください。

Task description

To teach the robot, which image is equal to the sample one, please give instruction to the robot, while it is looking at the different images.
While the robot is browsing the images, the one, that the robot is currently looking at, is shown with a green frame on the screen.
As soon as the robot has decided to choose one of the images, it points to the image and a red frame is shown around it. The other images disappear.
After the robot has chosen an image, please give positive or negative feedback to it depending on whether it has chosen the correct or incorrect one.
The robot needs to adapt to your teaching, so it may not understand you from the beginning.

何の画像がサンプル画像と同じだとロボットに教えてください。そうする為、ロボットが画像を見ている間に命令を与えて下さい。
ロボットが画像を見るときに、ロボットが見る画像がスクリーン上で緑のフレーム付に表示されます。
ロボットが画像を選択すると、画像が赤いフレーム付に表示されて、選択されない画像が消えます。
選んだ画像は正しいかどうかによって、正のフィードバックまたは負のフィードバックを与えて下さい。
ロボットがあなたの教え方になれる必要があります。それで、命令を分からない場合があります。

Task2: Pairs solitaire (ペーアズ・ソリテール)
Please teach the robot, how to play the game “pairs solitaire”.
ペーアズ・ソリテールの遊び方をロボットに教えて下さい。

Explanation of the game:
“Pairs solitaire” is a game for one person. The goal of the game is, to find pairs of cards showing the same image. At the beginning of the game, all cards are turned upside-down so that their front sides cannot be seen. In every move, the player turns around two cards. If their images match, the cards remain open. If their images do not match, the cards are turned back upside down. Then the player continues with the next move until all pairs have been found. The lower the number of moves, necessary to find all pairs, the higher the score.

ペーアズ・ソリテールは一人で遊ぶゲームです。目的は同じ画像が付いている 2 枚のカードを見つけることです。最初は全てのカードがひっくり返ります。表が見えません。プレイヤーがカードを２枚をひっくり返します。カードの画像が同じであれば、カードをそのままにしておきます。画像が同じではなければ、カードをまた表が見えないようにひっくり返します。そして、ゲームは全ての一組のカードが見つかるまで続きます。手の数が低くなるにつれてスコアが高くなります。

Task description:
To make a move, the robot first points at one card and the card is turned around on the playfield, so that its surface can be seen. Then it points at a second card which is also turned around. After selecting two cards, the robot waits for feedback from the user. Please teach the robot without giving commands, just by providing positive or negative feedback on its draws.

手を打つ為、ロボットがカードに指差します。そしてカードの画像が見えるようになります。次、ロボットが二番目のカードに指差して、そのカードの画像も表示されます。２枚のカードを選んだ後にロボットがユーザからのフィードバックに待ちます。
命令を与えないで、正のフィードバックまたは負のフィードバックだけでロボットにゲームの遊び方を教えて下さい。

Task3: Connect four (四目並べ)

Please teach the robot, how to play the game “connect four”.
ロボットに四目並べの遊び方を教えてください。

Explanation of the game:

Connect four is a game for two persons. One player takes the red stones, the other player takes the yellow stones. The goal of the game is, to arrange four stones of one’s own color in a row, either vertically, horizontally or diagonally.
The board stands upright and the players take turns to drop a stone of their own color into one of the columns of the board. Because of the gravity, the stone will then occupy the lowest free space of that column.

四目並べは二人で遊ぶゲームです。一人のプレーヤが赤いマークを使って、他のプレーヤが黄色いマークを使います。自分の色のマークを横または縦または斜めに4個並べると勝ちです。
ゲーム盤が直立しています。プレーヤーが順々に一個のマークを一列に入れます。重力の為、マークが列の一番下の空いている所に入ります。

Task description:

The robot plays the game against a computer player. Your task is, to provide feedback to the robot for good and bad moves, so that it learns how to win against the computer. To make a move, the robot points at a row on the board. The robot’s move is recognized and displayed on the screen. After making a move, the robot waits for your feedback. Then it is the computer player’s turn to make a move.

ロボットがコンピューターを相手にします。ロボットの良い手または良くない手にフィードバックを与えて下さい。あなたのフィードバックにより、ロボットがコンピューターを負かせるようになります。手を打つ為、ロボットがゲーム盤の一列に指差します。ロボットの手が認識されて、ゲーム盤に表示されます。手を打った後で、ロボットがフィードバックに待ちます。次はコンピューターが手を打ちます。

Task4: Commands (命令)

Task description

Please teach the robot the following commands:
Mae (forward)
Ushiro (back)
Hidari (left)
Migi (right)
Suwatte (sit down)
Tatte (stand up)
In order to make the robot learn the commands, please utter the commands and give positive feedback if it reacts correctly and negative feedback if it does not react correctly.
Please take care, to make sure that the robot does not drop from the table.

ロボットに以下の命令を教えて下さい：
前
後ろ
左
右
座って
たって

ロボットに命令を教える為、命令を言って下さい。ロボットが命令に正しく応じると、正のフィードバックを与えて下さい。間違って応じると、負のフィードバックを与えて下さい。
ロボットがテーブルから落ちないようにご注意ください。

2.2 Questionnaire

Questionnaire (アンケート)

1) Teaching the robot through the given task was enjoyable
ロボットをこの課題により教えることは楽しい。

fully agree 1....2...3...4...5 fully disagree
そう思う　　そう思わない

2) The robot understood my feedback
ロボットが私のフィードバックを理解できた。
fully agree 1....2...3...4...5 fully disagree

3) The robot learned through my feedback
ロボットが私のフィードバックにより学習できた。
fully agree 1....2...3...4...5 fully disagree

4) The robot adapted to my way of teaching
ロボットが私の教え方に適応した。
fully agree 1....2...3...4...5 fully disagree

5) I was able to instruct the robot in a natural way
私はロボットに慈善的に教示を与えられた。
fully agree 1....2...3...4...5 fully disagree

6) Throughout the experiment, I always knew what instruction or reward to give to the robot.
実験のうちに、どんな教示またはフィードバックをロボットに与えたら良いか分かった。

fully agree 1....2...3...4...5 fully disagree

If not, what kind of situations remained unclear?
そうではない場合は、どんな状況が不明でしたか？

1) Which task did you prefer? why?
あなたにとって、何の課題が一番やりやすかったですか？ 理由は何ですか？

2) Did you give different kind of feedback to the robot dependent on the different tasks?
それぞれの課題のうちに与えたフィードバックをタスクによって変えましたか？
If yes, why?
「はい」と答えた場合：　理由は何でしたか？

3) Did you notice any problems during the communication with the robot？ What kind of problems did you notice?
ロボットとの相互作用の間、何か問題がありましたか？どのような問題でしたか？

4) Do you have any experience in interacting with entertainment robots？
エンターテインメント・ロボットとの相互作用の経験がありますか？

5) Have you ever kept a dog or any other pets?
If yes, which ones?
犬または他のペットを飼ったことがありますか？
「はい」と答えた場合：　どのペットですか？

6) Do you have any other remarks about the experiment?
実験について何か付け加えることはありますか？

3 Study on learning commands

3.1 Instructions

Thank you for participating in this experiment!

The goal of my research is enabling a robot to learn to understand basic human commands and feedback.
For this experiment, please imagine, you have just bought a new robot. (The one, standing on your right.) It has a training-software installed, which allows you to teach it commands and feedback, so that it will be able to understand you later. All you need to do is running through an adaptation phase together with the robot in front of your TV screen.

You will see a living room scene on the screen. The robot is able to manipulate objects in the living room scene using gestures. The robot's actions are also visualized on the screen through icons. If there is any object in the picture, that you cannot identify, please ask before the start of the experiment.

This experiment has two phases.

1) Learning object names
In the first phase, the robot will ask you for object names. When the robot asks for an object, you will see a green arrow and a white spotlight, which highlights, the object, that the robot is currently asking about.

In this phase, please say only the name of the object, which the robot is pointing at, without any additional words.
Example: “Ball”, not “This is a ball”.
It may take some time and several tries for the robot to remember the object names. Please be patient.

2) Learning commands

In the second phase, you will teach commands to the robot. Please have a look at the scene on the sheet of paper that you received together with this instruction. It shows the correct state, in which you would like the room to be. If something is wrong (there is only one thing in each scene), please instruct the robot to return the room to its correct state. E.g. you see, that the toy car is lying around on the floor, then please instruct the robot to put it into the box, or if the floor is dirty, then please ask the robot to clean it.
Thinking bubbles with icons mean that you want the robot to do something that cannot be known from the state of the living room scene only. E.g. you want to know the robot’s battery status or you want the robot to bring you something. Please give instructions accordingly.
After your instruction, it is the robot’s turn to react to your command. You can see the changes on the screen. Speaking bubbles with icons mean that the robot gives you an answer – e.g. when you asked it about the status of its battery, it will show an icon to visualize its battery state.

Teaching a command to the robot is done as follows:

1) A new scene is shown on the screen

2) The robot waits for your instruction. Please give an instruction according to what you see on the screen. (You can start giving instruction, as soon as you see the new scene)

2a) The robot may ask you to repeat your instruction because it did not understand you. In that case, please repeat your instruction.

2) The robot performs responds to your command by performing an action. The action will be visualized on the screen and also by gestures and sounds of the robot. Please understand that the robot will make mistakes, especially at the beginning, but it will get better, over time.

3) The robot looks at you to ask you for feedback. Please give positive feedback, if the robot performed correctly. Otherwise, please give negative feedback

For giving positive and negative feedback or commands to the robot, you can **use speech and gesture *freely*** and also **touch the robot's head**.

Please talk to the robot in JAPANESE.

On the sheet of paper that you received together with this instruction, you can find an image of the correct living room scene as well as three faces. Before starting the experiment, please write down and remember a name of your choice for each of the three persons. Put the sheet in front of you during the experiments, so that you can see it easily, and use it as a reference, if necessary..

Please make sure to:

- …give instructions only once, unless the robot asks you a second time
- …give positive and negative feedback in the way you prefer after the robot has finished its action and looks at you
- …behave naturally and talk spontaneously, as you would, when you actually had to train a robot to use in your home. ***Please do not try to test the limits of the system or adapt your teaching to how you think the system works on a technical level. (The teaching method has been designed for novice users without any computer science or engineering background!)***
- …avoid talking about the contents of the experiment to anyone who is going to participate later.

After the experiments are finished, please answer the questionnaire.
Thank you for your cooperation.

Anja Austermann


Toys

3.2 Questionnaire

Questionnaire:

Age:

Gender

1) Teaching the robot through the given task was enjoyable

fully agree 1....2...3...4...5 fully disagree

2) The robot understood my feedback
fully agree 1....2...3...4...5 fully disagree

3) The robot learned through my feedback
fully agree 1....2...3...4...5 fully disagree

4) The robot adapted to my way of teaching
fully agree 1....2...3...4...5 fully disagree

5) I was able to instruct the robot in a natural way
fully agree 1....2...3...4...5 fully disagree

6) The robot took too much time to learn
fully agree 1....2...3...4...5 fully disagree

7) The robot is intelligent
fully agree 1....2...3...4...5 fully disagree

8) The robot behaves autonomously
fully agree 1....2...3...4...5 fully disagree

9) The robot behaves cooperatively
fully agree 1....2...3...4...5 fully disagree

10) The robot repeated its queries too often
fully agree 1....2...3...4...5 fully disagree

11) Throughout the experiment, I always knew what instruction or reward to give to the robot.
fully agree 1....2...3...4...5 fully disagree

If not, what kind of situations remained unclear?

3) Did you notice any problems during the communication with the robot ? What kind of problems did you notice?

4) Do you have any experience in interacting with humanoids or entertainment robots ?
If yes, which ones?

5) Have you ever kept a dog or any other pets?
If yes, which ones?

6) Do you have any other remarks about the experiment?