University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics

Crossmodal Learning and Prediction of Autobiographical Episodic Experiences using a Sparse Distributed Memory

Doctoral Thesis

submitted by
Sascha Jockel
of Hamburg

November 2009

Dissertation

zur Erlangung des akademischen Grades
Dr. rer. nat.
an der Fakultät für Mathematik, Informatik und Naturwissenschaften
der Universität Hamburg
eingereicht beim Department Informatik

Genehmigt von der MIN-Fakultät, Department Informatik
der Universität Hamburg auf Antrag von

Jianwei Zhang, Prof. Dr. (Erstgutachter, Betreuer)
Bernd Neumann, Prof. PhD (Zweitgutachter)

Hamburg, 12. Mai 2010 (Tag der Disputation)

Abstract

This work develops a connectionist memory model for a service robot that satisfies a number of desiderata: associativity, vagueness, approximation, robustness, distribution and parallelism. A biologically inspired and mathematically sound theory of a highly distributed and sparse memory serves as the basis for this work. The so-called *sparse distributed memory* (SDM), developed by P. Kanerva, corresponds roughly to a random-access memory (RAM) of a conventional computer but permits the processing of considerably larger address spaces. Complex structures are represented as binary feature vectors. The model is able to produce expectations of world states and complement partial sensory patterns of an environment based on memorised experience. Caused by objects of the world, previously learnt experiences will activate pattern sequences in the memory and claim the system's attention. In this work, the sparse distributed memory concept is mainly considered a biologically inspired and content-addressable memory structure. It is used to implement an autobiographical long-term memory for a mobile service-robot to store and retrieve episodic sensor and actuator patterns.

Within the scope of this work the sparse distributed memory concept is applied to several domains of mobile service robotics, and its feasibility for the respective areas of robotics is analysed. The studied areas range from pattern matching, mobile manipulation, navigation, telemanipulation to crossmodal integration. The robot utilises properties of sparse distributed memory to detect intended actions of human teleoperators and to predict the residual motion trajectory of initiated arm or robot motions. Several examples show the model's fast and online learning capability for precoded and interactively provided motion sequences of a 6 DoF robot arm. An appropriate encoding of sensor-based information into a binary feature space is discussed and alternative coding schemes are elucidated.

A transfer of the developed system to robotic subfields such as vison-based navigation is discussed. The model's performance is compared across both of these domains, manipulation and navigation. A hierarchical extension enables the memory model to link low-level sensory percepts to higher-level semantic task descriptions. This link is used to perform a classification of demonstrated telemanipulation tasks based on the robot's experience in the past. Tests are presented where different sensory patterns are combined into an integrated percept of the world. Those crossmodal percepts are used to dissolve ambiguities that may arise from unimodal perception.

Zusammenfassung

In dieser Arbeit wird ein konnektionistisches Gedächtnismodell für einen Service-Roboter realisiert, das eine Riege von Desiderata erfüllen soll: Assoziativität, Unschärfe, Approximität, Robustheit, Verteiltheit und Parallelismus. Als Grundlage dient die von P. Kanerva entwickelte und biologisch inspirierte Theorie eines hochgradig verteilten und dünn besetzten Speichers, engl. *Sparse Distributed Memory* (SDM). Es entspricht generell einem Speicher ähnlich dem Random-Access Memory (RAM) eines Computers wobei ein weitaus größerer Adressraum abgedeckt werden kann. Komplexe Strukturen werden als sehr lange Vektoren eines binären Merkmalsraums auf das Gedächtnismodell abgebildet. Das Modell erzeugt Erwartungen und vervollständigt partielle Wahrnehmungen der Umwelt mittels gespeicherter Sensordaten. Ausgelöst durch Objekte der Umwelt werden zuvor gelernte Erfahrungen durch Folgen von Aktivierungsmustern im Fokus der Aufmerksamkeit des technischen Systems dargestellt. Primär wird in dieser Arbeit das Sparse Distributed Memory als eine dem menschlichen Vorbild ähnliche Gedächtnisstruktur zur autobiographischen Langzeitspeicherung von Erfahrungsmustern diskutiert.

Diese Arbeit präsentiert die Übertragung des Sparse Distributed Memory Konzepts auf verschiedenste Domänen der mobilen Service-Robotik und analysiert dessen Eignung für die jeweiligen Bereiche. Diese Bereiche umfassen die mobile Manipulation, Navigation, Telemanipulation und die kreuzmodale Integration verschiedenartiger Sensormuster. Der Roboter nutzt die prädiktiven Eigenschaften des Modells um beispielsweise Intentionen von Teleoperatoren zu erkennen und initiierte Roboterarm-Bewegungsmuster sowie mobile Navigationsaufgaben autonom zu Ende zu führen. Verschiedenste Anwendungsszenarien zeigen die schnelle Lernfähigkeit von kodierten sowie interaktiven Manipulationssequenzen eines Roboterarms mit sechs Freiheitsgraden mittels einer vorwärtsgerichteten, neuronalen Architektur, die das SDM darstellt. Dabei werden u.a. die Probleme der Informationsenkodierung von Sensordaten in einen binären Merkmalsraum erörtert und weitere Kodierungsmöglichkeiten untersucht.

Die Übertragung des Modells auf andere Modalitäten zur Lösung von visuellen Navigationsaufgaben wird dargestellt und das Verhalten des Modells bezüglich der Manipulationsdomäne verglichen. Durch eine hierarchische Erweiterung des Gedächtnismodells wird es ermöglicht, Sensorwahrnehmungen mit semantischen Konzepten höheren Abstraktions-

grades zu verknüpfen um beispielsweise Ziele einer interaktiven Telemanipulationsaufgabe frühzeitig zu ermitteln. Es werden Untersuchungen präsentiert, die eine kreuzmodale Integration verschiedenartiger Sensormuster zu einem multimodalen Perzept der Umgebung darstellen, um Ambiguitäten unimodaler Wahrnehmungen zu kompensieren.

Contents

1 Introduction

Memory is the means by which we draw on our past experiences in order to use this information in the present.
(Robert J. Sternberg, Professor of psychology, 1999)

Contents

Motor learning in biological beings exhibits properties such as generalisation, learning, discrimination, and forgetting. Recognising and reconstructing motions is important for analysing behaviours, activities and movements. Trajectory learning is a fundamental component to teach a robot complex manipulation patterns. If this is achieved through an interactive control by a human instructor it is known as *learning by demonstration* (LbD). However, human demonstrations are noisy and inevitably inconsistent.

The goal of this work is to build a memory system that is able to learn and predict low-level sensor and actuator patterns of a service robot. The memory system should allow the robot to learn motion sequences and to autonomously control its actuators by retrieving sequences at a later date. It should allow making predictions of the consequences for a perceived situation similar to already learnt circumstances. However, a major requirement for such a system is a flexible and fast memory-based learning component that exhibits

the ability to *recognise*, *cluster*, and *approximate* trajectories demonstrated by humans even under consideration of noisy and incomplete patterns.

To pursue the development of conscious machines and cognitive robots, the memory system needs to be capable of associating new input patterns to already learnt contexts. Similarly, the memory system needs to be flexible enough to memorise and generalise interactively taught motion trajectories from both individual and multiple user demonstrations. Because robots are generally operated with more than a single sensor, the system needs to facilitate an extension with further sensoric modalities. The implementation of the robot memory structure presented in this work is based on the work about *sparse distributed memory* (SDM) proposed by Kanerva (1988), also known as *Kanerva coding* (Sutton and Barto, 1998).

1.1. Motivation

The following sections will briefly outline the four research areas that mainly motivated the study presented in this work.

1.1.1. Learning from Experience

Biological beings reveal remarkable learning capabilities to constantly meet new challenges of life. The ability to relate and generalise new circumstances to once memorised experiences, e.g. to select appropriate actions to interact with the environment, plays a crucial role in their struggle for survival.

Let us consider how human beings model the world. While interacting with the world over and over again, we become better at dealing with it. It is said that we *learn from experience*. Records of experiences are stored in our *memory* and we constantly relate them to sensations of the environment to predict what is likely to happen. Based on the resulting *predictions* due to our experience we choose appropriate actions, e.g. to avoid danger or to avoid mistakes we have made before. We understand what is happening only to the extent that we are able to predict what is going to happen. In this, the *internal model* is our means of prediction. Apart from just observing the world through sensors and learning about it, a system also acts and learns from its interactions with the world. Learning to perform actions, thus, relies on learning to reproduce sequences of motion patterns. The more experience we have, the more faithful are the dynamics of the world reproduced by our model of the world. Consider for instance the growing habitual formation of expectations. Thereby, the model just captures statistical regularities of the world reported by its senses and the system's own actions. In doing so, not just the main effects of actions but also the side effects are recorded in our memory.

Human attention is attracted if something *unusual* happens caused by discrepancies between what is predicted by the internal model and is reported by the senses. The internal model affects our perception profoundly. As an example, consider driving along your thoroughly known way to work. You will hardly pay attention to things that stay unchanged and agree with your internal model. But if a new sign or parking bay appears overnight, you will become aware of the changes after passing them, as your experience startles. Comparing experiences with current circumstances yields a criterion for updating and revising the world model.

Let us consider an individual's subjective perception of the world at any given moment as a state in which the world resides in a particular moment. The flow of the world states over time can then be described as a sequence of subjective perceptions. Accordingly, a simple way to build a world model is to store the reports of the senses in the memory and to retrieve them from there later. Thus, a mechanism is required to store subjective perceptions in a memory in a way that the system is able to retrieve the information later for predicting what might happen. Cognitive psychologists have identified three common operations of memory: encoding, storage, and retrieval. A memory structure possessing above-mentioned functionalities will be implemented for a service robot to encode, store and retrieve low-level sensor and actuator patterns.

Some general principles are known from studies ranging from neuroscience to cognitive science and psychology. The brain remembers recurring activity patterns and detects regularities in them (Hawkins, 2005). That also explains why things become familiar and why we rehearse mental or physical skills when presented with similar patterns over and over again. Memory plays a very important role in this process. The *artificial intelligence* (AI) community focus on how humans reason and solve problems. They reveal one of the highest levels of human behaviour. Traditional AI mainly focuses on reasoning, but often neglects that thinking is also based on information storage and retrieval. It mostly considers what to do, but not what these decision processes and actions are based on. Mathematical models that study principles for detecting regularities within different patterns include *associative memories* and *self-organising maps* (Willshaw, 1981; Hopfield, 1982; Kanerva, 1988; Kohonen, 1989; Hassoun, 1993).

1.1.2. Cognitive Robotics

Humans know how to behave according to a context they are in. This is subtle and difficult for a robot to understand. Nevertheless, by refining this ability to anticipate, it should be possible to produce robots that are proactive in what they do and that are more like a companion rather than remain a worker. Service robots that are able to predict the intentions of its human partner and to anticipate imminent actions could make human-robot interactions more natural.

Levesque and Lakemeyer (2007) define cognitive robotics as the study of knowledge representation and reasoning problems faced by an autonomous agent (a robot) in a dynamic and incompletely known world. Central to this effort is an understanding of the relationship between the knowledge, the perception, and the action of such a robot. In contrast, AI is not sufficiently concerned with the dynamic world of an embodied agent regarding low-level quantitative perception. Uncertainty, vagueness and associativity are also great challenges for knowledge processing in the field of AI. AI methods perform poorly on low-level tasks such as pattern recognition which are automatic. Combining AI methods with connectionist memory models into hybrid models can be of a certain benefit. Methodologies to handle major problems in the course of cognitive robotics include not only approaches of classical symbolic AI, but also biologically inspired approaches that use distributed representations. Such approaches are artificial neural networks, sparse distributed memory, connectionist models and parallel distributed processing (PDP).

In order to develop cognitive robots that exhibit behaviour which resembles some form of human-like intelligence, one way is to model robot control structures similar to those of humans. A cognitive developing robot needs robust and flexible learning mechanisms to acquire

and memorise relevant world knowledge and to organise this knowledge to facilitate interaction with its environment. D'Mello and Franklin (2009) propose four fundamental types of learning that would be essential for cognitive, autonomous developing robots: learning of perceptual qualities, known as *perceptual learning*, *episodic learning* of events, *procedural learning* of new actions and action sequences, and *attentional learning*. Perceptual learning in humans, for instance, occurs incrementally; there is no need for large training sets. Learning and knowledge extraction is achieved by a dynamic system that can adapt to changes in the nature of the stimuli perceived in the environment. Attentional learning is concerned with the principles that drive attention.

One question addressed in this work is whether robots can benefit from a system that is based on human-like learning mechanisms in the sense that they become more flexible according to: the required training cycles, knowledge integration, transfer of existing knowledge to new but similar tasks and memory organisation. Haikonen (2009) identifies *associative processing* as the most elementary cognitive process for determining the proper motor response to a perceived situation. This could be managed either by reactive approaches as proposed by Brooks (1986, 1991) or by memory-based approaches. This work particularly focuses on a memory-based approach to learn associations of sensor and actuator patterns while being constrained to few exposures of such events. I will treat a particular aspect of cognitive robotics which involves memory performance. Thus, I will outline the state-of-the-art in cognitive science regarding memory to underline the important aspects for a transfer to cognitive robotics in this work.

1.1.3. Distributed Representations

Computing architectures such as artificial neural networks, connectionist models and parallel distributed processing (PDP Research Group, 1986a,b; Anderson, 2007) have been motivated by the fundamental architectural differences between the brain and the conventional von-Neumann architecture. Behaviours more similar to the brain are expected to be obtained by studying architectures that are more similar to the brain.

According to Sjödin (1998), the architecture of the brain suggests that it uses distributed representations, i.e., somewhat exaggerated *everything* is stored *everywhere*. A major contribution arose through studying distributed representations in cognitive and neural nets instead of previously used local representations. Neural-nets research mainly focuses on distributed representations which spread information largely over a network architecture instead of representing a concept by a single unit. This essential mechanism of the brain leads to the fundamental requirement of studying how to encode and operate with distributed representations (Hinton et al., 1987; Plate, 1994; Field, 1994). This research showed that such distributed representations are robust and support generalisation. Distributed representations are suitable for the learning and clustering of similar or closely related—according to some metric—concepts. With such mechanisms, behaviours can be produced that look like being rule-governed although no explicit rules exist, as usually used in traditional AI.

Our brains model the world as the world is presented to them by our senses. *Sparse coding* of sensory input plays an important role in this capability. Sparse coding means that neurons encode sensory information by just using a small subset of active neurons out of a larger set. According to Olshausen and Field (2004), sparse coding provides an efficient means of representing data found in the natural world and provides a means of efficiently forming associations and storing memories. It also appears that sparse representations constitute an

important processing strategy of the nervous system, which massively increases the storage capacity of associative memories while saving energy (Field, 1994; Zhao, 2004). The sparse and distributed encoding of sensorimotor information in an associative memory constitutes the major object of research in this following study on developing a biologically-inspired cognitive memory system for service-robots.

1.1.4. Sequence Learning

The internal representation of a real-world problem mostly consists of a spatio-temporal ordered set of events and actions. Sequential organisation is fundamental in human behaviour (Tanji, 2001; Bapi and Doya, 2001; Sun and Giles, 2001; Serrien, 2009). According to Sun and Giles (2001), sequence-learning problems can be divided into the categories of *sequence prediction*, *generation*, *recognition* and *sequential decision making*.

In this work, temporally ordered sequences will mainly be created through a discrete sampling of robot trajectories (and further parameters) during a complex movement of, e.g., a redundant manipulator or a mobile platform. Generating a sequence through actions, making a sequential decision, is considered as being trajectory-oriented in this work. This means that a given sequence $s_i, s_{i+1}, \ldots, s_j; a_j \rightarrow s_{j+1}$ should determine an action a_j at time step j that leads to the desired state s_{j+1}.

Prominent approaches to sequence learning are *recurrent back-propagation networks* (Hochreiter and Schmidhuber, 1997), *associative networks* such as Hopfield networks (Hopfield, 1982), *hidden Markov models* (Baum et al., 1970), *temporal difference* and *reinforcement learning* (Sutton and Barto, 1998) and *self-organising maps* (Kohonen, 1987, 1989). Another approach to sequential decision making is *explicit symbolic* planning. The latter approach tries to reach its goal through iterative problem decomposition. Unfortunately, this method is quite complex, requires substantial prior domain knowledge which is not always available and is unnatural for describing situated actions (Agre and Chapman, 1990). In this work I will use a sparse and distributed memory model (Kanerva, 1988) to recognise a given partial trajectory based on the memory contents to establish an autonomous generation of a potential sequence remainder.

1.2. About this Work

This section highlights the main objectives of this work. The decision for the SDM model is elucidated based on requirements of cognitive robotics that are based on features of natural cognitive systems.

1.2.1. Objectives

The service robot TASER from the Technical Aspects of Multimodal Systems[1] (TAMS) division at the Dept. of Informatics, MIN Faculty[2], University of Hamburg[3] does not have any memory at all to store sensory experiences. Nor is it able to relate current situations to subjectively experienced situations of the past. By using a memory model the service robot TASER should be enabled to learn new manipulation and classification tasks fast and

[1] `http://tams.informatik.uni-hamburg.de`
[2] `http://www.min.uni-hamburg.de`
[3] `http://www.uni-hamburg.de`

online while being robust against noise. Furthermore, it should be enabled to relate current situations to learnt, similar tasks of its past to predict possible consequences based on the present context. In order to reach the intended goal of a remembered task autonomously, the robot should be allowed to control its next movements based on the predicted consequences. The memory system has to provide mechanisms for determining the similarity of situations it currently faces to those stored in its memory. The main objective of this work is to build a biologically-inspired associative memory system that possesses basic cognitive capabilities for memory storage and retrieval and is able to learn and predict low-level sensor and actuator patterns. This work realises a concrete implementation of the highly abstract SDM model and presents a feasibility study of making the SDM model accessible for robotic applications and cognitive robotics research. The motivation of using a memory for robotic applications is to mimic human memory with a model that can make associations between information, recall more salient information more accurately, and have a comparably fast memory recall. Motion sequences that are remotely executed by a teleoperator should be recognised by the robot to help the user accomplish the intended action. The following research questions will be addressed in this work:

- To which extend can the connectionist SDM concept be used to endow robots with basic cognitive capabilities to memorise robot motion sequences?
- How can sensor-based information that originates from laser range scanners and cameras be encoded such as to make it usable in a connectionist network?
- Is it possible to relate new input patterns and sequences of patterns to already learnt situations by some kind of generalisation?
- Can a memory be used to predict a certain behaviour regarding a given situation based on past experience by means of mechanisms of abstraction?
- Is it possible to gain memory-based autonomous robot control by retrieving sequences from experience with respect to the current context?
- Does an SDM-based model support different perceptual modalities equally?
- Can an SDM be used to identify the intent of a human operator that uses a robot for a remote task execution?
- Is an SDM suitable for accomplishing perceptual disambiguation of classifications provided by different sensoric modalities?
- What are the problems and advantages of such a system as compared to other approaches?

1.2.2. Requirements

This work utilises Kanerva's SDM model (Kanerva, 1988) to examine the above-mentioned research questions. Based on associative memory principles the model should be able to autonomously generate and execute spatial trajectories when confronted with positions or arm configurations similar to a learnt trajectory. The system learns by either hard-coded

trajectory examples or interactively human-guided movements to fulfil an arbitrary high-level task. The system is trained with a discrete set of spatial positions resulting from a desired continuous motion sequence. The model should be able to interpolate points and joint angles based on the trained motion patterns and to extrapolate motion sequences based on its experience.

Equipping a robot with an associative memory that bears a resemblance to some human long-term memory characteristics must be based on a definition of requirements. In the following some features of human-long term memory are emphasised and their importance with respect to cognitive, autonomous developing robots are elucidated.

Pattern recognition: An essential mechanism of the brain is to recognise and discriminate high-dimensional sensorial input patterns, e.g. speech signals, human faces, movements and to classify them based on already learnt patterns. Closely related individual patterns are grouped into categories based on common properties. The recognition and classification of patterns is the most fundamental requirement for robots in identifying meaningful entities in physical signals. Caused by an attractor characteristic, successive memory reading will lead the robot to find a similar pattern or sequence in its memory. Modern and future robots that sense their environment through various sensors have to deal especially with high-dimensional feature patterns. The SDM theory appears to be highly suitable to deal with large pattern spaces. Patterns discussed in this work originate from laser range scanners, omnidirectional cameras, redundant manipulators and further sensors and actuators.

Content-addressable: Human long-term memory is addressed by contents rather than specific addresses[4]. A robot that retrieves a solution to a problem or the consequences of an action by cueing its memory with the current sensorial description of the world would be clearly beneficial. If sensor-based patterns are used as the address where they are to be stored in memory, it becomes possible to converge on a stored pattern starting from an inaccurate (or incomplete) version of that input-patterns by an iterative process. In an SDM, patterns read from an inaccurate memory-address will retrieve a more accurate version of the same memory-address because the value was originally written into many memory locations within a given access diameter.

Auto-associativity: The brain essentially memorises a set of meaningful objects of the world and the relations and associations between those objects. Auto-associative memories enable the retrieval of an entire memory even from a fragmentary sample of itself, e.g. several notes allowing us to retrieve a whole song. If a robot cues its memory with a trajectory that is similar to a trajectory of some previously encountered tasks, auto-association allows to retrieve the consequences of that previously executed trajectory with respect to the current context.

Generalisation: The capability to extend a concept to some less-specific criteria is a fundamental characteristic of reasoning. Generalisation is directly related to abstraction. A system's ability to acquire and store new information is quantified by its ability to distinguish among stimuli and to associate an appropriate stimulus with the current, unconditioned stimulus. Generalisation diminishes the sensitivity to input noise,

[4] Although memory does not use addresses to access memory contents, I will use the term to refer to memory cells. The term address space will refer to the entire amount of cells in a memory.

variations of input stimuli and the need for storage capacity at the cost of a possible increase of false-positive responses. With respect to this work, using individual but similar patterns as addresses for the SDM leads to an averaging of the stored data patterns. Features that are common to all or most patterns in the neighbourhood of the used memory locations will stand out as an encoding for a cluster of patterns and thus cause generalisation. The attraction by stored patterns allows to read information from the memory with a vague address pattern which can be considered a kind of abstraction[5].

Sparse coding: The address space resulting from low-level sensory perceptions of a robot including various sensors is usually quite large. Memorising reference vectors for each possible scenario or situation therefore becomes infeasible. Sparse coding principles, which are used in the brain, should reduce the address space to a more reasonable size. Accordingly, system inputs are not required to match stored memory addresses exactly but must fall within a specific distance of an address to activate closely related concepts.

Robustness to noise: Noise caused by the inaccuracy of sensors and complex dynamics in the perceived world is still challenging. A robot must use mechanisms to deal with noise in a robust fashion to match input stimuli with memorised patterns. Generalisation and abstraction as mentioned above play a crucial role for handling noise.

Resilience to memory damage: A concept, e.g. "*Grandmother*", is spread across a network of neurons rather than using a single unit. This is the opposite of symbolic representations mostly used in classical AI. When a robot uses distributed representation, minor damage to the underlying network will not cause the loss of entire concepts.

One-shot learning: The ability to store and recognise complex patterns after a single or few exposures is called one-shot learning. A robot needs this ability to avoid long training phases.

Forgetting and graceful degradation: A major instrument for memory management which contributes to ensuring that primarily the relevant memories are recalled. Different theories have emerged from extensive research on forgetting, e.g. trace decay, interference and repression theory[6]. When enhancing intelligent systems, e.g. robots, with human-like memory capabilities, forgetting cannot be neglected. It could facilitate the acquisition of new skills by weakening distractive, older ones. Graceful degradation means that a system responds with the best approximation to some lost information. If a robot has forgotten an exact execution of an action it should be able to respond with an appropriate approximation of it.

[5] The details of memory storage and retrieval are elucidated in Section 3.2.

[6] According to trace decay theory, time is the cause of forgetting. The interference theory describes interfering and inhibiting effects of previously learnt and retained memory items while repression theory argues that unpleasant experiences are pushed into the unconscious to avoid mental re-exposure. In the course of this work we agree with the definition that forgetting occurs because other information alters or interferes with already stored memory contents both proactively and retroactively.

1.2.3. Evaluation and Validation

The model should provide a robot with capabilities for learning and predicting autobiographical episodic sequences, mainly robot arm motion sequences, based on a biologically-inspired memory model. The model should provide simple storage and retrieval mechanisms. To the best of the author's knowledge there is no work known regarding robot arm manipulation based on an SDM. Since this work serves as a kind of feasibility study it is hard to benchmark the results arising from this thesis. Nevertheless, the memory will be judged by the accuracy it displays during the retrieval of motion patterns, its resistance to noise, if predictions of consequences finally reach the end of a trajectory and so forth. Thus, a number of experiments will be conducted to identify the operationality of an SDM in different research domains of robotics according to the requirements mentioned above.

1.2.4. Further Information

Other work has been proposed that uses associative memories to learn motion trajectories of robot manipulators. To mention some of it: Albus (1975) developed the CMAC controller based on structures related to the cerebellum. Araujo and Vieira (1998); Barreto and Araujo (1998) used a temporal multidirectional associative memory (TMAM) together with a radial basis function (RBF) to tackle the trajectory generation and inverse kinematics problem. Ito and Tani (2004) used a recurrent neural network with parametric bias (RNNPB) to generate synchronous robot arm motion patterns with respect to a human demonstrator. Reinhart and Steil (2008) use reservoir computing with recurrent neural networks to learn the forward and inverse kinematics of a redundant PA10 manipulator.

CMAC belongs to the class of linear function approximator called *tile coding*. Tile coding has been used in many reinforcement learning tasks and is well-suited for efficient online learning. Unfortunately, tile coding and radial basis function networks become impractical when used for tasks with very high dimensionality, e.g. several hundreds of dimensions. This is because their computational complexity increases exponentially with the number of dimensions (Sutton and Barto, 1998). Existing approaches to reduce this growth, e.g. such as hashing, even become impractical after a few tens of dimensions. Alternative approaches choose binary features that correspond to particular prototype states. The strength of such methods is that the complexity of the functions that can be learnt depends entirely on the number of features, which bears no direct relationship to the dimensionality of the task. Such methods are, e.g. Kanerva coding (Kanerva, 1988) used in this work and the random representation method (Sutton and Whitehead, 1993).

Connectionist approaches to learning can be grouped in two broad types: those with high capacity but requiring many learning steps, and those that have low capacity but can do one-shot learning. Models of the first type are able to extract statistical regularities or hidden variables gradually from the input. These include back-propagation networks, Boltzmann machines and competitive learning networks. Models of the second type rapidly memorise the input without recoding it into hidden features. These include linear pattern associators (Kohonen, 1972; Anderson, 1977), Hopfield networks (Hopfield, 1982) and convolutional memory models.

The main research contribution of this work is to study the application of a connectionist memory model that is able to store and retrieve motion sequences of a mobile robot and a redundant manipulator. To pursue the development of conscious machines and cognitive

robots, such systems need to memorise a model of the world and should be endowed with fundamental capabilities to associate actual circumstances to past experiences and to constantly adapt the underlying model. The neurobiologically plausible SDM model has certain similarities with the cerebellar cortex, a part of the mammalian brain specialising in sensorimotor coordination. This work might be of particular interest to researchers that seek a new computational model exhibiting robust and scalable characteristics and normally are concerned with neural networks.

1.3. Thesis Structure

This chapter gives an introduction to the thesis and the motivations behind it. The remainder of the thesis is organised as follows.

Chapter 2 illustrates the main memory functions such as acquiring, storing and accessing knowledge. An introduction is given to recent findings in human memory research from a neuroscience perspective and various cognitive architectures are presented. The chapter concludes with various associative network models which form the basis for equipping a robot with a predictive memory mechanism in this work.

Chapter 3 discusses Kanerva's sparse distributed memory model in detail, which is the basis of this work. Related models and extensions are presented. Several SDM-based robotic and non-robotic applications are introduced.

Chapter 4 presents all necessary details about the hard- and software of the utilised robot systems, namely TASER and LIZARD. Furthermore, a brief introduction to a telemanipulation system is given that has been developed for this work. The system is used for an interactive creation of diverse robot arm motion sequences through demonstrations by arbitrary human instructors.

Chapter 5 describes the implementation of an SDM for robotic applications. Some deficiencies of the source model are eliminated by including a number of functional extensions. Alternative information encoding methods are proposed and analysed in detail after facing several practical problems. A series of manipulation experiments is conducted with a 6 DoF robot arm for evaluation.

Chapter 6 approaches the transfer of the SDM model to other robotic tasks in distinct fields. The memory model, mainly used for manipulation tasks throughout this work, is applied to the domain of robot navigation based on view-sequences. The behaviour of the SDM is analysed and compared across both domains.

Chapter 7 studies an intention-detection system based on episodic experiences. To ground low-level information with high-level semantic task descriptions, a multi-SDM architecture is developed to maintain and utilise a number of SDM instances. The memory model generalises several training trajectories given by human instructors and classifies arbitrary trigger trajectories into a set of known task trajectories. Several experiments are reported and evaluated to account for varying skill levels of the instructors.

Chapter 8 assesses crossmodal effects of multimodal percepts, e.g. based on laser range finders and vision sensors. Those are utilised to gain higher prediction accuracy. The multi-SDM model is used to establish a rough localisation of the robot within an office environment.

Chapter 9 concludes the work presented in this thesis. It illuminates the major results of this study, finally discusses the established research contributions and some future work.

2

Memory—an Unrevealed Mystery

. . . A farmer went out to sow his seed. As he was scattering the seed, some fell along the path . . . some fell on rocky places . . . other seeds fell among thorns . . . [and] other fell upon good soil. . .
(Mark, Chapter 4, The Parable of the Sower: 3–20)

Contents

This impressive, metaphorical quotation can be used to interpret what happens in human long-term memory during the everydays process of memorising events, concepts, *et cetera* in our minds. It has been written a couple of thousand years ago, though in a different context. The long-term memory commonly plays tricks while memorising something that are caused

by mechanisms like generalisation, forgetting and the minimisation of discrepancies. Some impressions may never make it into the mind and *fell along the path* while others may not be consolidated and *fell upon rocky places*. The latter perhaps result from conflicts with already existing knowledge. That other seeds *fell among thorns* can be seen as interferences of impressions, arising form competitive contexts where some have more urgent concerns, or emotional factors that are leading to a kind of repression. Those memories that make it into the mind will be memorised in the long-term memory and *fell upon good soil*.

2.1. Memory

Memory is one of the most widely studied brain functions. According to the handbook of cognitive science, the term *memory* refers to different forms of acquiring, storing and accessing knowledge (Strube, 1996). The contents of memory are not only acquired by consciously controlled cognitive processes. Rather, they predominantly result from individual interaction with the environment. Furthermore, Strube (1996) defines *knowledge* as acquirement about the reality, also-called *declarative knowledge*. *Skills* are cognitively learnt sensorimotor and perceptual capabilities. *Memorising* is a conscious operation and strategy to retain information over a long period of time to *recall* and re-use contents of the memory at a future date. *Recall* is partially based on search processes that can be cued by perceptual stimuli caused by the current environment. Subsequent *recognition* is characterised as an identification of familiarity when a given concrete circumstance is related to previously encountered experiences.

Amnesia, a memory disorder, results from injuries to parts of the brain that record and recall memories. It has thus been the major subject of studies that support the theory on different memory systems (Baddeley, 2001; Squire et al., 1993). The first, simple multistore-model of memory was described by Atkinson and Shiffrin (1971). It postulates that long-term memory is composed of separate memory components.

Yet, the mystery of memory has not been solved and there is no unique theory on this major brain function. However, the analogy between differing theoretical views is that the memory is suggested to comprise two main information storage systems with respect to the time that information is retained: namely *short-term memory* and *long-term memory*[1].

2.1.1. Short-term Memory

The *short-term memory* (STM) allows for a fast recall of a limited number of memory items without any rehearsal. The period of decay lies usually in the range from a few seconds to several minutes. In more recent work, STM is characterised as a component that is actively involved in the processing of information rather than just being a temporal storage system (Atkinson and Shiffrin, 1971; Baddeley, 2001). A refinement of STM leads to the so-called *working memory* that comprises internally generated and externally perceived information as well as results of cognitive operations.

Alternative memory models exist, where the STM contains current activated contents of long-term memory at a certain point in time. Within the scope of the ACT theory (see Section 2.2.1), the working memory consists of that active contents that becomes part of a

[1] In addition to the broad categories of STM and LTM, psychologists distinguish subdivisions of primary memory, secondary memory, reference memory, episodic memory and semantic memory.

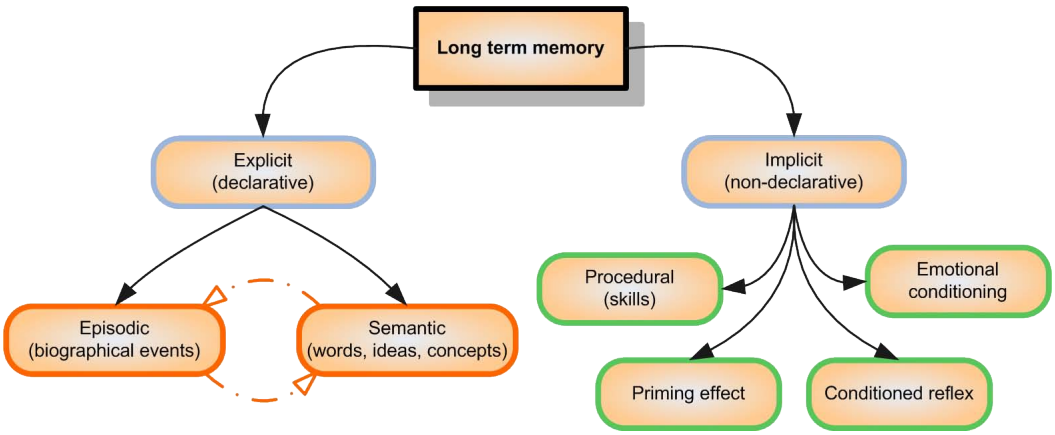


Figure 2.1.: A taxonomy of mammalian long-term memory.

cognitive process. STM models were partly inspired by neurological patients with organic amnesia caused by bilateral damage to brain structures like the medial temporal lobe and hippocampus, e.g. the famous patient Henry Gustav Molaison, better known as HM.

2.1.2. Long-term Memory

The *long-term memory* (LTM) can store much larger quantities of information for a potentially unlimited duration that can last for a lifetime and is expected to have an infinite capacity. Information can be transferred from short-term to LTM by rehearsal and repetition mechanisms.

During all those investigations on memory, different views on LTM emerged over time. Most of them assume multiple memory subsystems within the LTM. Cohen and Squire (1980) divide LTM into *non-declarative* procedural memory, the "knowing how" memory for basic skills such as walking, riding a bike *et cetera*, and the "knowing that" *declarative memory* (see Figure 2.1). The latter memory stores knowledge about factual information and personal events in our life. Tulving (1972) argues that declarative information is further divided into *episodic memory*, which contains details about our life, and *semantic memory*, which contains factual information of the world and how it works and describes it as "mental thesaurus" (Tulving, 1972). Another interpretation is given by Schacter (1987) who focuses on memories that can be thought about consciously (*explicit memory*, similar to declarative memory) and memory that is recalled unconsciously (*implicit memory*, similar to non-declarative memory). Each of these approaches is consistent with certain features of existing data but also has difficulty accommodating others. A taxonomy of hypothesised LTM subclasses is illustrated in Figure 2.1.

2.1.3. An Everlasting Debate

There is a significant amount of research that supports a clear differentiation of short-term and long-term memory. While information in an STM is stored and retrieved sequentially, the LTM stores and retrieves information by association.

The multistore models, pervasively influenced by neuropsychology and supported by studies of Baddeley (2001), view STM and LTM as separate systems that rely on distinct represen-

tations. In the theory by Baddeley (2001), the STM is decomposed into information-specific buffers. He distinguishes between processes for memory storage and executive functions.

According to unitary-store models, supported by the work of Squire (1986, 2004); Atkinson and Shiffrin (1971); Hebb (2002), STM and LTM rely largely on the same representation but differ in the level of activation of those representations and respective processes. Prominent theories characterise memory as a flow of information, e.g. Atkinson and Shiffrin (1971) with the *Multi-Store Model*, Craik and Lockhart (1972) designed the *Depth Processing Model*, and Baddeley and Hitch (1974) proposed a *Working Memory Model* with a central executive[2] to replace the STM model. In contrast, connectionist approaches to memory and information processing suggest that information is stored in several interconnected units rather than in a memory trace (or related structures).

2.1.4. Three Processing Stages of Memory

Regardless of the ongoing debate on whether STM and LTM are architecturally separable systems or not, current theories agree on three core memory processes to retain (learn) and recall past experiences. Those major procedural and functional aspects of memory are:

Encoding: The processing of incoming perceptual information (from sensory input) into a representation the memory system can cope with.

Storage & maintenance: Creation and processing of encoded information.

Retrieval: Recovering encoded information of the past from our memory and returning it into the cognitive focus as a response to an arbitrary cue.

Though these stages are distinguishable, they are clearly inter-related. Jonides et al. (2008) describe biological mechanisms that might support psychological processes on a momentum-by-momentum basis. An item is encoded, maintained over a delay with some forgetting and ultimately retrieved. Further, electrophysiology and neuroimaging indicate that working memory, like LTM, is a widely distributed function, largely neocortical (Fuster, 1998).

The memory system proposed in this work will follow the above-mentioned memory processes. Sensory-based signals such as range measurements, joint angles, edge pixels and so forth will be encoded into an appropriate representation for the memory. It will memorise these representations with respect to certain similarities within the input patterns, which can be seen as kind of generalisation. The most interesting functionality of an artificial memory for a cognitive robot is the retrieval of information stored in the memory. Biological systems constantly retrieve information from memory according to the given situation to establish a cognitive focus. The retrieved informations are used to hypothesise what is supposed to happen. In this work we will allow a service robot to constantly retrieve information from its memory of the past to make predictions or assumptions of what is supposed to happen according to a given circumstance. Abstraction will help to make predictions for situations that are similar but not equal.

[2]The versatile central executive component of working memory resembles an attentional system and controls other components.

2.2. Memory Models and Cognitive Architectures

Memory models differ regarding the format of stored information, namely associations or memory traces. The former format stores memories as links between memory units (Strube, 1996). Accordingly, learning is the process of creating new and modifying existing associations. Remembering is defined as a spreading activation between associated memory units (Anderson, 2007). The latter model, also known as *search of associative memory* (SAM) by Raaijmakers and Shiffrin (1981), hypothesise that learning is characterised by the creation of memory traces, also-called engrams, driven by relation- and context-specific information. A trace gets activated if parts of the context are perceived. Important characteristics of such trace models are:

- Recognition is cued by partial patterns and depends on the correlation between cue and memory trace
- Activation of memory traces occurs in parallel
- Memory traces are content-addressable

Distributed models address the issue of:

- Separate vs. composed memory traces
- Effects of reapplied stimuli
- Information representation within memory traces
- Encoding of contextual information

The difference to other models depends on the search heuristics. Regarding the cause of forgetting the models mainly assume interferences between memory contents.

Many researchers of cognitive architectures ignore constraints posed by human cognition, apparently because they focus on studying effective interactions of an agent with its environment. Thus, the term of cognitive architecture is often misleading and should rather be termed agent architecture. In the following, architectures are described that try to bridge the gap between psychophysics and memory.

2.2.1. Adaptive Control of Thoughts Theory (ACT)

The *adaptive control of thoughts* (ACT) production system describes a model of the mind and is proposed by Anderson (1990). It is based on two LTM components, the declarative and the procedural memory. The fundamental assumption of Anderson is that the memory contents of the procedural memory are built from declarative knowledge, the knowledge about the facts of the world. The procedural memory consists of numerous production rules. According to Anderson (2007), the activation speed and activation probability of a concept in memory (a chunk) depends on its activation level. The activation level depends on the sum of a baselevel activation, reflecting its past frequency of use, and an associative activation, reflecting its relevance to the current goal. An activation spreads from a presented item through a network towards memory items that bear a relation to the presented one.

A neural plausible implementation of such production system was shown in the later theory called *adaptive control of thought-rational* (ACT-R). To obtain an optimal information processing the system is parameterised and structured through rational analysis. ACT-R is the most adequate type of architecture and the best known of the current psychology-based systems.

2.2.2. Memory Prediction Framework

Building upon work of numerous neurobiologists Hawkins (2005) designed the *memory prediction framework* (MPF). His theory describes intelligence as "the capacity to predict the future by analogy to the past" (cf. Section 1.1.1). The brain uses a memory-based model to make continuous predictions of future events. If those predictions are disproved, the brain learns, e.g. by novelty detection (Barakova and Lourens, 2005), and adjusts its memorised information according to the new data.

According to the memory prediction theory, the neocortex learns invariant representations of pattern sequences in a hierarchical neural network. The structure of an invariant representation captures the important relations in the world, independent of any detail. Given some partial or distorted sensorial inputs of a known environment, the memory recalls the stored patterns of the past in an auto-associative manner. When unknown patterns occur, they violate the predictions and capture our attention by reaching the highest cortical layer, the hippocampus, which depicts the short-term repository for new memories. Thus, the primary function of the neocortex is to make predictions by comparing the knowledge of the invariant structure with the most recent observed details (Garalevicius, 2007). These predictions arise from the comparison between feedforward information (what is happening) and feedback information (what is expected to happen).

The main limitation of this theory, and the reason it is not discussed in more detail here, is its lack of detail. Open questions remain on how to create invariant representations—curiously the most important part of his theory, how the brain handles associations and how it binds together knowledge by the cerebellum and the neocortex is unresolved. The underlying brain algorithm remains undiscovered. Nevertheless, Hawkins' theory is an innovative hypothesis on how the brain works.

2.2.3. Connectionist Models

Connetionism disagrees with the idea that the mind uses rule-based and semantic information processing as proposed by traditional AI. The major principle in connectionism is to describe mental phenomena by interconnected, weighted networks of simple and often uniform units (analog to neurons). Each unit represents a microfeature of the world. Further principles are:

- Information is represented in a distributed manner (this contradicts the local and one-unit-one-concept representations)
- Encoding of knowledge takes place in the weights between units
- Learning results from modifying the weights
- Memory is content-addressable

Weights are generally represented as an $N \times N$ matrix. Recognition is described as a reconstructive process where the system activates patterns similar to certain cue stimuli. The most common connectionist models are neural networks. Future connectionistic models are expected to explain neurophysiological and neurobiological phenomena in conjunction with experimental cognitive science (Kandel et al., 1996).

An extension of the connectionist approach is called the *embodied cognitive science*. Similar as in connectionism, cognition is seen as a unpredictable selective, constructive, context-dependent and non-verbal process. Contrary to connectionism, the embodied and embedded cognitive science community defines intelligent behaviour as an interplay between brain, body and a dynamic physical world.

2.3. Associative Memories as an Instrument of Prediction

Neural networks whose main functionality is to associate patterns are called associative memories. All the information is stored in the weights of such networks as proposed by the connectionist approach. If triggered by an arbitrary input pattern, they retrieve stored patterns that provide the highest degree of similarity with respect to the input.

Usually they provide one-shot learning[3] capabilities and their weight matrix is constructed by summing up the outer product of all input–output constellation. The output of an associative memory is computed by the dot product of the weight matrix and the input vector followed by a thresholding. There are two main derivates of associative memories: an *hetero-associative memory* associates two different types of patterns while a *auto-associative memory* associates patterns with themselves. The latter type is capable of retrieving associated patterns even if the input patterns are noisy or incomplete. Most associative memory models are linear and feedforward in nature and use Hebbian learning. Common neural networks that belong to the class of associative memories are Hopfield networks, correlation memories and backpropagation.

2.3.1. Hebbian Learning

Hebb's rule explains long-term strengthening (so-called *long-term potentiation*, LTP) and long-term weakening (*long-term depression*, LTD) of synaptic connections under the condition that activation of two connected neurons are correlated. Hebb's rule is defined as follows:

> When an axon of cell A is near enough to excite cell B or repeatedly or consistently takes part in firing it, some growth or metabolic change takes place in one or both cells such that A's efficiency as one of the cells firing B, is increased. (Hebb, 2002)

Since the modification of a synapse relies only on pre- and postsynaptic neurons, Hebb's rule implies locality of the neural plasticity. Furthermore, it introduces the concept of activity-induced reinforcement or weakening of the synapse (Floreano and Mattiussi, 2008). The role of synaptic plasticity has been suggested as a possible mechanism for associative learning.

[3] They are able to learn in a single pass.

2.3.2. The Hopfield Model

Contrary to a *McCulloch-Pitts Network* (McCulloch and Pitts, 1943), where an output is determined based on the neurons' input at each time step, a *Hopfield Network* (HN) consists of neurons with symmetric weights ($w_{ij} = w_{ji}$) and asynchronous updating mechanisms exclusive of self connections ($w_{ii} = 0$).

Hopfield (1982) explains the physical meaning of a content-addressable memory by an appropriate phase space flow of the state of a system. Such a system responds to an ambiguous starting state by a statistical choice between the memory states it most resembles. A brief summary of the Hopfield model will be given in the following. For more details the reader is referred to Hopfield (1982); Amit et al. (1989); Rojas (1996); Hendrich (1996); Arbib (2002).

- The network consists of N neural units with possible states $S_i \in \{0, 1\}$, which cause a neuron to either not fire or fire. At timepoint t the system is describable by a vector $\{S\} = \{S_1, \ldots, S_N\}$.
- Each neural unit is connected to all other units and thus, together they form a fully-connected network. Two neurons S_j and S_i are interconnected via symmetric weights ($w_{ij} = w_{ji}$).
- At each time step a unit is picked at random (asynchronous network change[4]). If unit i with threshold θ_i is chosen, $S_i = 1$ if and only if $\sum w_{ij} S_j \geq \theta_i$, otherwise $S_i = 0$.

The symmetric connection of neurons as well as the absence of self connections is crucial to guarantee a stable system behaviour. Otherwise such nonlinear, dynamic systems will show chaotic behaviour (cf. Hendin et al. (1991)). Hopfield defined mathematical quantity based on the *Lyapunov function* that describes the activity dynamics as the *energy* E of the network, such that:

$$E = -\frac{1}{2} \sum_{ij} w_{ij} S_i S_j + \sum_i S_i \theta_i \tag{2.1}$$

If a neural unit S_k changes its output value (in the asynchronous case), the energy will change according to the following Equation:

$$\delta E = S_k \sum_{j \neq k} w_{kj} S_j + S_k \sum_{j \neq k} w_{jk} S_j \tag{2.2}$$

In such a symmetric network with asynchronous update, when starting from an initial state S^0, the dynamics of the Hopfield network will move to a lower potential energy when a neuron changes its state. Thus, the system will converge to a global minimum after finite number of steps where, unaffected by any state changes of a neuron $S_k = 0 \leftrightarrow S_k = 1$.

A variation of the Hebbian learning rule (cf. Section 2.3.1) is used for the storage of P binary patterns ξ^μ of n-bit length:

$$w_{ij} = \frac{1}{n} \sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu (1 - \delta_{ij}), \tag{2.3}$$

[4]The asynchronous network change is commonly used. In a synchronous network, each unit will be updated in a single time step.

where $\delta_{ij} = 1$ for $i = j$ and 0 for $i \neq j$. The Hopfield model achieves a storage capacity of $K = 0.14$ bits per synapse (Golomb et al., 1990).

2.3.3. The Willshaw-Palm-Model

Willshaw et al. (1969) and Palm (1980) proposed a simple and linear model of a fully-connected associative memory with stochastic asynchronous dynamics. It is one of the earliest neural network models of associative memory. Contrary to the Hopfield model it does not use any feedback loops.

In most simple neural network models, excitation and inhibition play identical roles. The Willshaw model stores memories in excitatory synapses using an extremely simple version of Hebb's rule (Golomb et al., 1990). The network is composed of a matrix of binary synapses $w_{ij} \in \{0, 1\}$ that feed into binary output units with possible states $S_i \in \{0, 1\}$ interconnected via a binary link . The network's function is to map input patterns ξ^μ of length M onto output patterns Ξ of length N by thresholding the sum of the input signals.

The Willshaw model achieves an asymptotic storage capacity of $K = 0.7$ bits per synapse, which exceeds the capacity of most alternative models. While the model offers an extremely simple way of storing sparsely coded memories, it exhibits a poor performance with regard to random, uncorrelated memories (Golomb et al., 1990).

2.3.4. Biologically-inspired Associative Memory: Cerebellar Models

Some associative memory models have a strong correlation to biological structures, for instance the cerebellum. The cerebellar cortex is involved in the integration of sensory perceptions and the coordination of movements. According to Albus (1975), the cerebellum provides precise coordination of motor control for such body parts as the eyes, arms, fingers, legs, and wings. It stores and retrieves information required to control thousands of muscles in producing coordinated behaviour as a function of time. It receives proprioceptive and kinesthetic information from the muscle spindles, joints and tendons and gets a copy of motor commands sent by the cortex. The cerebellum compares how well motor commands coming from the cortex are executed (see Figure 2.2).

The cerebellum is located in the inferior posterior portion of the head. It is rather involved in modulating, then initiating movements and therefore also termed the silent brain[5]. It guides movements based on the sensory feedback. Although recent neurophysiological evidence supports the hypothesis that the cerebellum learns from experience, the cerebellum is not generally considered as a memory area of the brain.

According to Arbib (2002, Chapter Part II) the Cerebellum can be decomposed into cerebellar nuclei and cerebellar cortex. The only output cells of the cerebellar cortex are the Purkinje cells (see Figure 2.3), and their main function is to provide varying levels of inhibition on the cerebellar nuclei. Each Purkinje cell receives two types of input—a single climbing fibre, and many tens of thousands of parallel fibres. The most influential model of cerebellar cortex has been the Marr-Albus model of the formation of associative memories between particular patterns on parallel fibre inputs and Purkinje cell outputs, with the climbing fibres acting as training signals. Therefore, the models by Marr and Albus that were initially developed in parallel will be introduced briefly. Figure 2.3 outlines the three layers of the cerebellum. The functions of the main cells and fibres of the cerebellar cortex

[5]Electrical stimulation of the cerebellum does not cause movements or sense impressions.

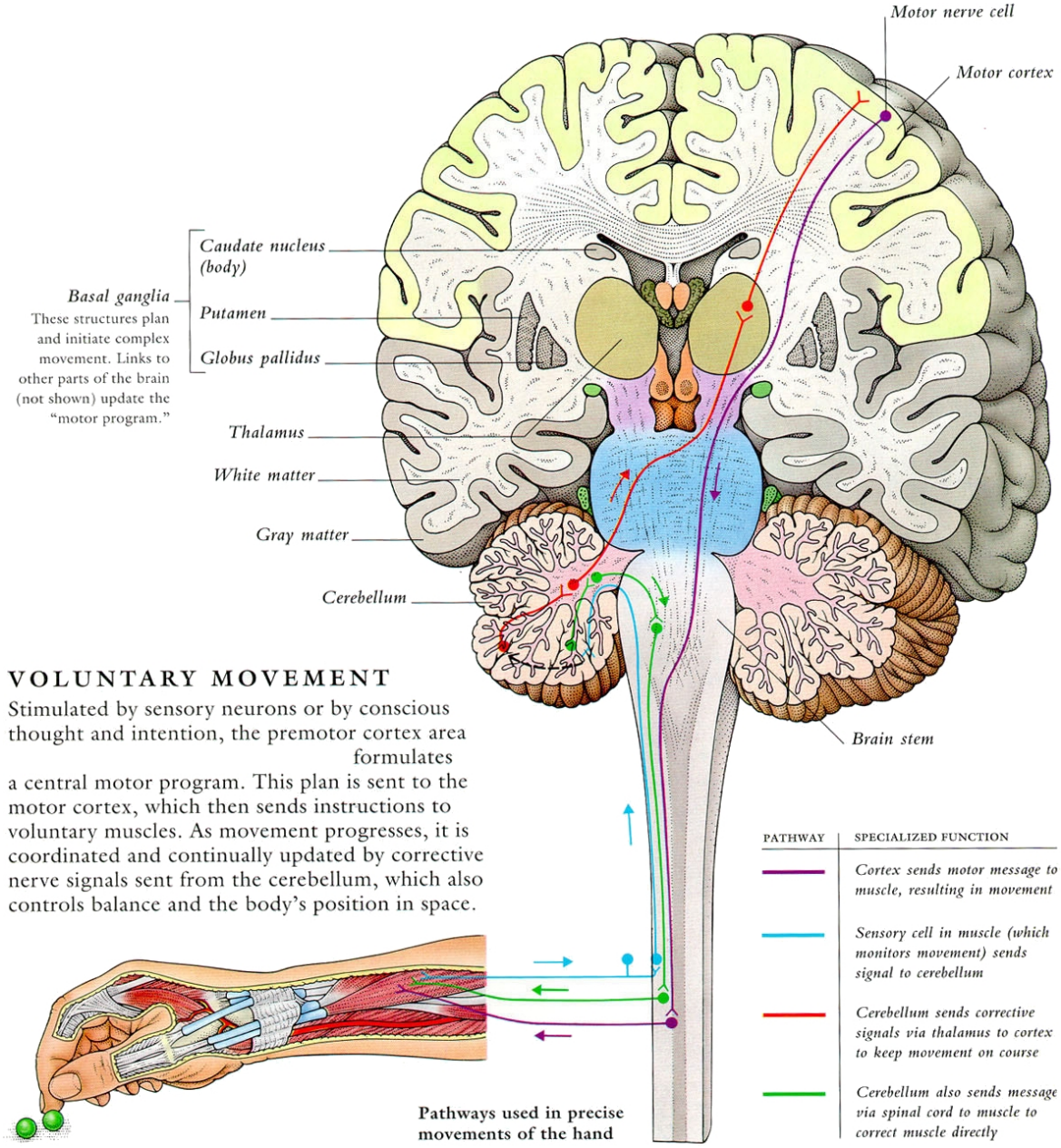


Figure 2.2.: Schematic diagram of the neural pathways involved in precise hand movements. Source: `http://163.178.103.176/Fisiologia/neurofisiologia/Objetivo_8/Clayman78.jpg`

are summarised in Table 3.1 in one of the following sections when compared to the SDM model.

2.3.4.1. Marr's Theory of the Cerebellar Cortex

In his dissertation, David Marr, one of the founders of the discipline Computational Neuroscience, detailed a model of the functions of the cerebellar cortex. He draws the conclusion

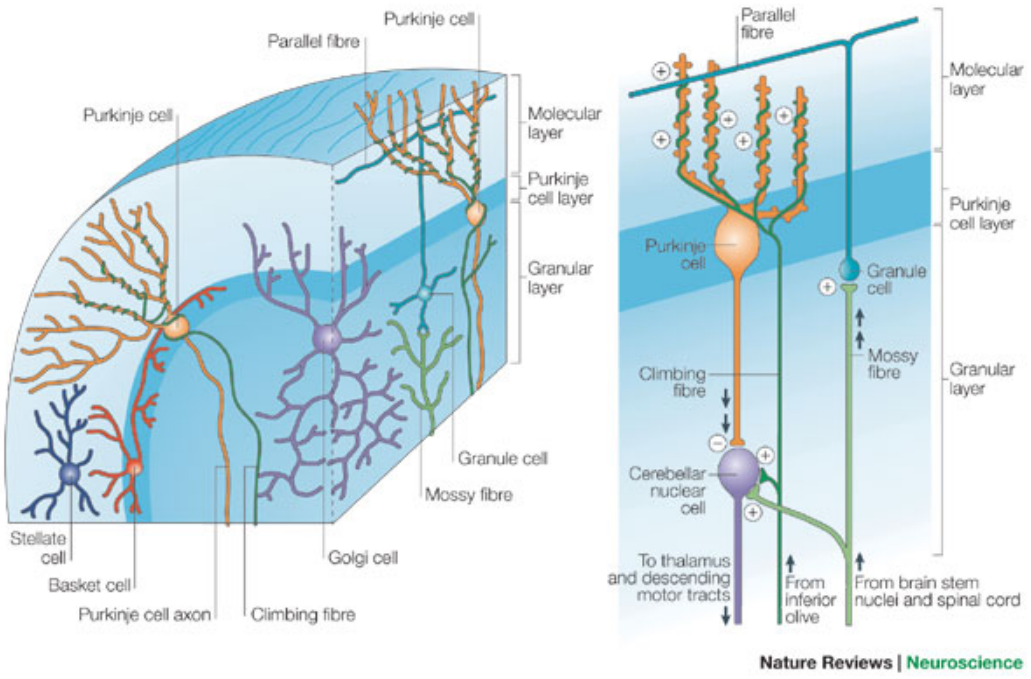


Figure 2.3.: Neuronal connections between the cerebellar cortex and the deep cerebellar nuclei. The cerebellar cortex is built of three layers. The superficial molecular layer with axons, dendrites and many synapses, the Purkinje-cell layer and the granular layer. Image from Apps and Garwicz (2005).

that the cerebellum learns to perform motor skills. In particular, it is involved in learning movements and maintaining reflexes by just two forms of input-output relations that are consistent with the cortical theory (Marr, 1969). A particular action and the context in which it is performed is associated via synaptic modifications of Purkinje cells. Hence, the context alone causes a Purkinje cell to fire, which in turn precipitates the next elemental movement (Edelman and Vaina, 2001).

According to Marr (1969) the cerebrum organises a movement during learning, and in doing so, causes the olivary cells to fire in a particular sequence. This causes the Purkinje cells to learn the context in which corresponding elemental movements are required. The next time such a context occurs, the mossy fibres activity stimulate the Purkinje cell, which evokes the relevant elemental movements (see Figure 2.2).

Central to Marr's theory is that afferent input events that are communicated from the olivary cells by the mossy fibres to the cerebellar cortex are turned into small subsets of active mossy fibres. This is called the codon representation of an input pattern. Marr (1969) identifies a granular cell as a codon cell that will only fire if activated by a certain codon. The number of codons associated with a collection of active mossy fibres is hardly related to the way Kanerva chooses d coordinates from a total of n coordinates (see Equation 3.9). Mossy fibres are assumed to be connected randomly among granular cells.

2.3.4.2. Albus' Cerebellar Model Articulation Computer (CMAC)

Independently of Marr's theory of the cerebellar cortex, Albus (1971, 1981) proposed a comprehensive theory on cerebellar functions that ties together anatomy and physiology into a pattern-recognition system. He infers an equivalency between the functionality and structure of the cerebellum to a modified, classical Perceptron-like classification system. CMAC is a sparse coarse-coded associative memory algorithm designed to provide motor control for robotic manipulators. It belongs to the class of tile coding. It is a kind of memory that is capable of learning motor behaviour and exhibits properties such as generalisation, learning inferences, discrimination, and forgetting, which are major characteristics for motor learning in biological creatures (Albus, 1975). He treats the problem of learning as function approximation for a given input, such that the system should learn to predict the commonly observed output. CMAC has been extensively used in reinforcement learning (Tham, 1995; Xu et al., 2002), as a classifier, for the adaption of PID control parameters, for industrial robot arms, and for hand-eye systems and biped walking.

According to Siciliano and Khatib (2008), Albus' CMAC can be described in terms of a large set of overlapping, multidimensional receptive fields with finite boundaries. Every input vector falls within the range of some local receptive fields. The response to a given input is determined by the average of the responses of the receptive fields excited by that input. Similarly, the training for a given input vector affects only the parameters of the excited receptive fields.

Albus (1971) complies with the theory proposed by Marr (1969) and extends it by several modifications based on the principles of information theory. He argued that the cerebellum accomplishes stable learning processes by weakening synaptic strengths rather than by strengthening them during the pattern storage phase.

The quintessence of the studies by Marr (1969) and Albus (1971) is the Purkinje cell plasticity as a basis of cerebellar learning due to the long-term depression of parallel fibre synapses. However, the rule on the weakening of synapses is still known as the Marr-Albus model, and remains the reference model of studies of synaptic plasticity of cerebellar cortex (Siciliano and Khatib, 2008).

2.4. Concluding Remarks: Towards a Predictive Autobiographical Robot Memory

Another model equivalently discussed with the Marr-Albus Model is the sparse distributed memory proposed by Kanerva (1988). The SDM model shows many of the characteristics that a human memory possesses (cf. Section 1.2 for those of special interest for robotics). Kanerva developed a highly abstract mathematical model for his theory whose application to robotics is the major concern of this work. CMAC, Albus' above-mentioned model of the cerebellum, becomes impractical when used for tasks with very high dimensionality, e.g. several hundreds. CMAC's computational complexity increases exponentially with the number of dimensions. In contrast, Kanerva's model depends entirely on the number of features which bears no necessary relationship to the dimensionality of a task. This constitutes one major reason for selecting the SDM model.

Even if somehow over-anticipated at this stage, where the model has not been introduced, lets consider the SDM model in the frame of natural memory systems to further clarify the

authors choice of it. It is declarative in the sense that it uses data as address, making it content-addressable, and uses only a sparse subset of memory addresses, making it scalable. Along with all afore-mentioned arguments, the SDM model is procedural in that it forms sequences that could be considered as procedures. Learning is realised by incrementing weights across a probability distribution. With respect to the embodied connectionist approach, it can learn a set of situation assessments and is able to generalise these. The SDM model is related to the PDP neural memory models mentioned in Section 1.1.3 and meets most of the requirements mentioned in Section 1.2. Sensing and perception can be used to recall the nearest stored memory to any encoded perceptual unit. Thus, an SDM seems to offer powerful human-like ways of recalling nearest matches to present experience in a best-first manner (Ritter et al., 2003)

The model can be used to store and recall large amounts of low-level sensory data efficiently, without requiring the data to be completely accurate or that we know exactly what we need to recall. Though it may not be an exact model of human memory, it shares enough characteristics to suggest that human memory works in a similar way.

Kanerva's memory model overcomes limitations in the Hopfield memory model such as dependence on storage capacity or the number of neurons, the inability to store temporal sequences, symmetric interconnections, and a new limited ability to store correlated inputs. Kanerva discussed the application of his ideas to the "frame problem" of Artificial Intelligence and showed that parts of the problem concerned with manipulating vast quantities of data about the real world can be handled with his model.

Keeler (1988) developed a mathematical framework for comparing the sparse distributed memory to the Hopfield network. He extended Kanerva's SDM model and showed that Hopfield's model was a special case of this extension. Keeler showed that Kanerva's model corresponds to a three-layer network with the middle layer consisting of many more neurons and that Kanerva's formulation permits context to aid in the retrieval of stored information. The following chapter will introduce Kanerva's model of SDM in much more detail.

3

A Sparse and Distributed Memory Model

The internal plasticity of memory which 'distributed' models suggest is one of the most curious and characteristic features of human memory, and one which clearly differentiates our cognitive systems from the 'memories' of current digital computers.
(Sutton, 2008)

Contents

Cognitive scientists mainly investigate memory in action to model its functionality. Regretfully this happens almost exclusively in laboratory contexts where natural interferences are minimal. The brain constantly uses records of past experiences that bias the course of upcoming actions and interactions in an environment and are accumulated without paying particular attention to this process.

Memory has the remarkable property of retrieving a stored sequence based on some part of an observation and then to follow the sequence from that point on, e.g. a melody, a path *etc.* Memory has a permanent addressing framework that is independent of what we have learnt so far and accordingly uses patterns that serve as addresses to the memory. Kanerva incorporates this theory of a *content-addressable* memory in his idea of *sparse distributed memory* (SDM). He takes an engineer's view on the theory of a memory to answer mainly two questions:

1. How do people recall past experiences and distinguish between familiar and unfamiliar events. What is the organisational structure of such a record?

2. How to implement a physical model that permits appropriate storage and retrieval of a record?

Kanerva uses the word *record* as the experience of a person represented by a sequence of memory *items* whereas the term *memory* depicts the medium for storage. SDM is an idealised model of an highly abstract theory that deals with several aspects of human LTM. Kanerva mainly focuses on the possibility of using the SDM framework for discussing whether human memory is of an associative nature and content-addressable.

The correspondence of the distance between concepts in our minds and the distance between points of a high-dimensional[1] space led to his idea of an SDM. SDM uses binary vectors of length n as input and simply matches them against target vectors. This is realised by finding the closest target with respect to a distance measure—the Hamming distance. The vector space exhibits interesting properties if n is sufficiently large.

An SDM is a form of *associative memory* that is popular in both computer science and psychology. In the latter discipline associative networks are used primarily to model human processes underlying the retrieval of information. An individual's long-term memory of a concept is given by the strength associated with the node representing the concept (Suppes, 1994). The memory model can be seen as both a generalised form of a random-access-memory and as a two-layer feed-forward neural network with generalised Hebbian learning with weightless nodes within a Hopfield network.

As mentioned in earlier chapters, artificial neural networks (ANN) are common models to approach human memory capabilities, to learn and infer functions from certain observations and to recognise learned patterns. Interconnected layers of artificial neurons represent a

[1] High-dimensional means that the number of dimensions is at least in the hundreds.

connectionist approach to model complex relations between input and output data. An artificial neuron, also called node, is a crude model, an abstraction of a biological neuron expressed by a mathematical function. Such models work fine with a small number of nodes but quickly lead to the problem of computational expenses when scaling up to millions of nodes. The problem in classical ANN models occurs while trying to provide communication paths for a fully connected network.

Kanerva's description of a hardware implementation of the SDM model uses a considerable number of counters that do not incorporate information from the other nodes, thus a communication bottleneck as in alternative NNs does not appear. The strength of SDM as a physiological model of memory stems from its elegant and simple theory, from the ability to scale well and the absence of complex requirements for an implementation.

Addressing is a central aspect in the work proposed by Kanerva (1988). The author claims that in the unifying idea of his theory no fundamental distinction between memory addresses and data is made. The data to be stored in the memory are addresses to the memory (Kanerva, 1988). The main aspect of SDM is the mapping of a vast binary memory onto a smaller set of physical locations, which are called *hard locations*. A further aspect is a uniform distribution of data stored to the set of hard locations to simulate a considerably larger virtual space.

This chapter serves as a basic introduction to the concepts and foundations behind the theory of SDM. Later in this work, this theory will be applied to many fields of modern robotic research. Its suitability for robotic applications will be discussed in detail within this work.

3.1. Boolean Geometry and Characteristics of Boolean Space

The most significant construct used in SDM theory is Boolean geometry. Boolean geometry is the geometry related to the Boolean space $Z_2 = \{0,1\}^n$, henceforth denoted by 2^n or simply N, with the dimension n. A point[2] of N is represented by a binary n-tuple or a binary number, e.g. $b_0, \ldots, b_n$. Thus, a one-dimensional Boolean space consists of two elements (0) and (1), a two-dimensional Boolean space consist of a four-element set $\{(0,0),(0,1),(1,0),(1,1)\}$ *et cetera*. There is no particular ordering of binary numbers necessary for analysis issues, they are just points in a binary space. Figure 3.1 visualises Boolean spaces from one to five dimensions, if the dimension exceeds five it gets quite hard to visualise the resulting space. It becomes clear, that the number of points increases exponentially with respect to its dimension.

The main functional principles of the SDM theory rely on *distances* d between points of N. Kanerva denotes this as the *distribution* of N. In Boolean geometry the *Hamming distance* (see Equation 3.5) between two points is the number of coordinates, or number of bits, at which the binary vectors differ, e.g. $d((011010),(000111)) = 4$. Thus, the distance represents a measure of the similarity between two memory items.

If a vector as a collection of n features describes a certain object[3], each feature of the object can either be present (1) or absent (0). Accordingly, the more features of two feature vectors are similar, the closer these vectors are together and consequently the smaller is the

[2] A "point" in memory is an address that can represent a word, a data item, a pattern, an event or a memory item.

[3] It can also represent different things from an object, e.g. concept *et cetera*.

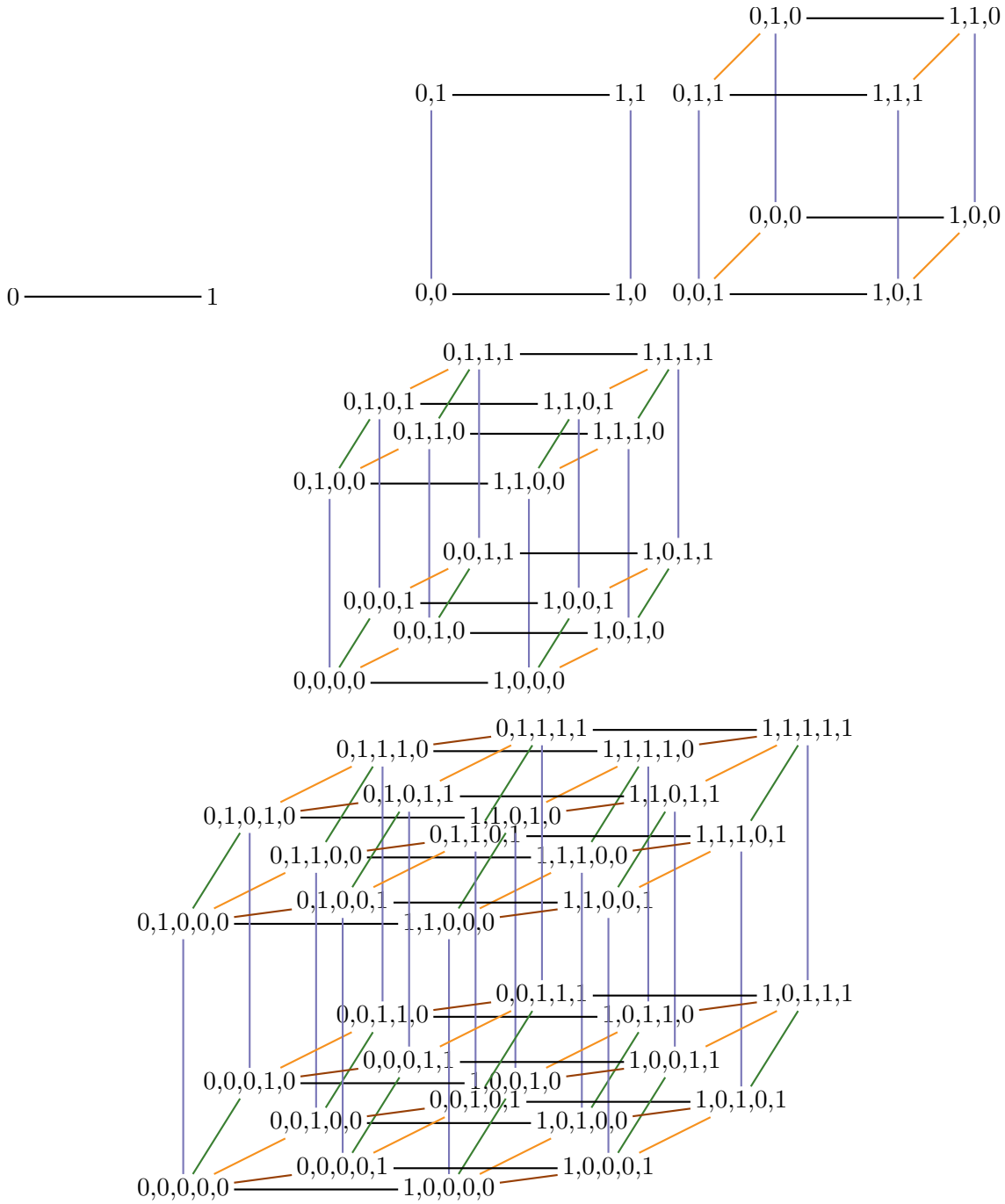


Figure 3.1.: Visualisation of a one-, two-, three-, four-, and five-dimensional space. According to Kanerva (1988), the space of an SDM with $N = 1000$ would map to the corners of a unit hypercube in 1000-dimensional Euclidean space, which is quite hard to visualise in the same fashion.

Hamming distance between them. It is obvious that the choice of the features representing any concept is a crucial issue. Let us consider a set of animals: a lion, a cheetah, a house cat, a blackbird, a finch and an ostrich that are represented by the features "is an animal" and "can fly". Then, the feature vector $x = (1, 0)$ will reveal that due to the Hamming distance, an ostrich as an animal that cannot fly is conceptually closer to the lion and the cheetah rather than to the blackbird and the finch, represented by $x = (1, 1)$. In turn it will be conceptually closer to the birds if the features are: "is animal", "has wings", "can fly", "is predator", "has two legs", "has four legs".

The main concepts related to Boolean space are examined in Kanerva (1988, Ch. 1, pp. 15) and more extensively in Blumenthal and Menger (1970). Those concepts are summarised in the following. Fundamental to all properties is that a point x in N is represented by a binary n-tuple or a binary number, e.g. $x = b_0, \ldots, b_n$, with $b \in \{0, 1\}$.

Origin

$$0 = 000 \ldots 000 \tag{3.1}$$

The point with zero coordinates.

Complement

$$'x \tag{3.2}$$

The inverted n-tuple of a point x is called its *complement* or *opposite* where $d(x,'x) = n$.

Norm

$$|x| \tag{3.3}$$

The number of ones in a binary representation of a point x.

Difference

$$|x - y| = |y - x| = x \oplus y \tag{3.4}$$

The number of coordinates at which two binary points x and y differ (*exclusive or*). The difference commutes.

Distance

$$\begin{aligned} d(x, y) &= |x - y| \\ &= |x_0 - y_0| + |x_1 - y_1| + \ldots + |x_n - y_n| \end{aligned} \tag{3.5}$$

The norm of the difference of two binary points is the distance. Distances expressed in bits are called *Hamming distance*, while their square root is the *Euclidean distance*.

Circle

$$O(r, x) = \{y | d(x, y) \leq r\} \tag{3.6}$$

A set of points that are less than r bits distant from a centre x. A circle with radius n comprises the whole space N.

Betweenness

$$x : y : z \quad \texttt{iff} \quad d(x,z) = d(x,y) + d(y,z) \tag{3.7}$$

Point y is between two points x and z if and only if the distance from x to z is the sum of distances from x to y and y to z. Betweenness is:

- recursive $u : x : y : z \rightarrow u : x : y$ and $x : y : z$,
- symmetric $u : x : y \rightarrow y : x : u$,
- not transitive $u : x : y : z \rightarrow u : x : z$

Orthogonality

$$x \perp y \quad \texttt{iff} \quad d(x,y) = \frac{n}{2} \tag{3.8}$$

Two points x and y are orthogonal (*perpendicular* or *indifferent*) if and only if they differ at precisely half their coordinates. The fraction $\frac{n}{2}$ is called the *indifference distance*. If $x \perp y$, it follows that x is also indifferent to its complement, thus $x \perp' y$. x is halfway between y and $'y$.

To get a more concrete idea of the concepts related to Boolean space as shown above, a few examples with a sufficient number of dimensions will be used for demonstration. Let the dimension $n = 6$, a radius $r = 1$, and the points $x = 011001$ and $y = 111011$. Then, regarding the characteristics of Boolean space:

Complement:	$'x = 100110$, and $'y = 000100$.
Norm:	$\|x\| = 3$, and $\|y\| = 5$.
Difference:	$x - y = y - x =' x -' y = x \oplus y = 100010$.
Distance:	$d(x,y) = \|x - y\| = \|100010\| = 2$.
Circle:	$O(1,x) = \{011001, 011000, 010001, 001001, 111001, 011101, 011011\}$.
Betweenness:	Any $z =$ \$001\$0, where each \$ is either 0 or 1, is between the points x and y, e.g. $x : 100110 : y$.
Orthogonality:	Two points can only be perpendicular if the dimension n is even. Since in this example $n = 6$, $010101 \perp 011011$.

The space N can be represented by the vertices of an n-dimensional unit cube in Euclidean space. Furthermore, an n-dimensional cube can have its vertices placed onto the surface of a sphere with radius $r = \frac{n}{2}$. This gives rise to the analogy to a spherical representation of the space N. Vectors are points on the surface of the n-dimensional sphere, the point x and its complement $'x$ are as two poles with the distance n that embrace the entire space. The points on the *equator* correspond to points at a distance $\frac{\sqrt{n}}{2}$ from x and $'x$, respectively (cf. Figure 3.2).

The number of points that are exactly d-bits away from an arbitrary point x is the number of possibilities to choose d coordinates from a total of n coordinates (see Equation 3.9). Let the arbitrary point be the origin 0, then the number of points that are d-bits away are the vectors containing exactly d 1's. Therefore, the distribution of space follows a binomial distribution with a mean of $\mu = \frac{n}{2}$ and a variance of $\sigma = \frac{\sqrt{n}}{2}$.

$$(n : d) = \binom{n}{d} = \frac{n!}{d!(n-d)!} \tag{3.9}$$

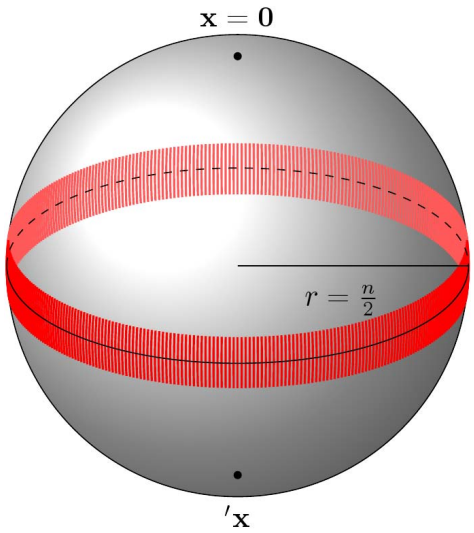


Figure 3.2.: SDM analogy to a spherical representation. The points of the N-dimensional hypercube lie on the surface of a sphere with radius $r = \frac{n}{2}$ in which more than 99% of the space lies around the equator (highlighted belt) between the poles that depict point x and its complement $'x$.

Consequently, in a space with $N = 1000$, a circular region of radius 425 bits centred on x encloses only about a millionth of the space.

The distance $\frac{n}{2}$ is called the *indifference distance*. Kanerva (1988, Ch. 1, p. 19) proves that almost all of any Boolean space is *almost* indifferent to any given point and lies near the equator (see Figure 3.2). For $N = 1000$, thus 99.9999% of the space lies between distance 422 and distance 578 from a given vector. Almost all of the space is far away from any given vector. A Boolean space of high-dimension is thinly populated, which is an important property for the construction of the model (Franklin, 1995). Kanerva calls this outstanding property of N the *tendency to orthogonality*. The larger the n, the more pronounced this effect gets. The distance from a point to the bulk of the space[4] is obtained by dividing the mean distance $\frac{n}{2}$ by the standard deviation with a distance of $\sqrt{\frac{n}{2}}$. As in the example used in Kanerva (1988) let $n = 1000$, the mean distance is 500. According to this, $99,9999\%$ of the space lies within $\sqrt{n} \pm 5$ standard deviations from a point x of N or 5 standard deviations from the mean value (cf. Figure 3.3), which amounts to 78 bits, thus within the interval $[422, 578]$.

Kanerva (1988, Ch. 1, p. 26) deduces how many features must be extracted correctly to classify a memory item out of a million memory items. The distribution of space follows a binomial distribution and a circle with radius 400 will encircle 10^{-10} of the space. To recognise a test item, just 20% of features have to be determined with certainty while the remaining 80% of features can be assigned with zeros and ones at random, with a probability of 0.5. This is due to the probability of a correct bit is $0.2 \cdot 1 + 0.8 \cdot 0.5 = 0.6$. This interpretation could explain the recognition of an object in different contexts.

[4]Which is from the pole to the equator of a sphere

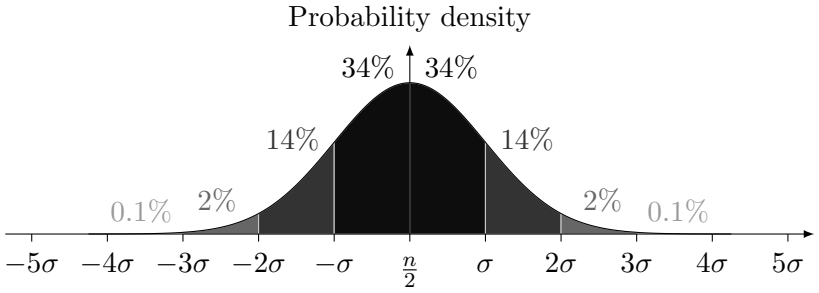


Figure 3.3.: According the normal distribution, over 99% of the space lie within 5 standard deviations of the mean or within $\sqrt{n} \pm 5$ standard deviation from x.

3.2. Memory Storage and Retrieval

The main question addressed by the work of Kanerva (1988) is how information should be stored in a memory so that it can be retrieved later when the situation warrants it. These questions claim to implement a memory with two main functionalities:

1. A present situation has to be recognised as being similar to some situation in the past.
2. If a past situation has been recognised, the consequences of that situation have to be retrieved.

Conventional computer systems use a memory structure where each memory location is accessed by its address. The feature of SDM is that the input data patterns can serve both as address to the memory location and as the data to be stored (Hely, 2006). This is called the *unifying principle*. Information can be accessed via its content. Thereby the model exhibits a form of content-addressability which is considered as one of the most important characteristics of human memory (Hawkins, 2005) and of major concern for fulfilling the two above-mentioned requirements.

Kanerva constructs a model of a random access memory capable of being implemented on a digital computer with three special registers—address, word in and word out. It is also implementable as an artificial neural network.

A Boolean space with 1000 dimensions leads to an enormous address space of 2^{1000} locations[5]. Most literature on SDM compares this number to the number of atoms in the universe (Hely, 2006; Franklin, 1995) or the number of neurons in the nervous system (Kanerva, 1988) which is far smaller.

Since it remains infeasible with current hardware to assign a memory address to each possible input of a 1000-dimensional space, Kanerva proposes to choose a *sparse* subset of physical addresses. For this subset a reasonably large, uniform random sample of binary patterns is selected as representative and physical memory locations of the whole space. The resulting subset of memory addresses is called *hard locations* (HL). To make the sparseness clear, consider the density ratio resulting from the mapping of a huge binary space with 2^{1000} locations onto a subset of, e.g. 2^{20} hard locations, which is 2^{-980} and indeed very sparse.

The set of sample locations from the entire space can be chosen in many ways. According to many early proposals, the hard locations are chosen *a priori* during the initialisation

[5]That is $1.07150861 \cdot 10^{301}$

of the memory. The memory addresses are distributed randomly in the address space 2^n and cannot be modified during normal operation, except for their content. With regard to Kanerva, the subset of the address space will be denoted as N' of N, and a vector to a hard location as x', respectively. Thus, every hard location x' accounts for a small portion of N.

A *randomised reallocation algorithm* for dynamic online allocation and adjustment of memory resources which eliminates the need for choosing the SDM size and structure *a priori* is used in this work. The dynamic allocation algorithm by Ratitch and Precup (2004) starts with an empty memory, and locations are added based on the observed data. New locations are allocated randomly in the neighbourhood of a location when there is a new datum which cannot be stored in the existing ones anymore. This approach is used in this work an will be explained in more detail in Section 3.5.3 and 5.1.1.

In conventional memories, a datum is stored into a single memory location. In Kanerva's type of memory, many hard locations participate in storing and retrieving each datum. Accordingly, a single hard location is involved in the storage and retrieval of many data items. This is what makes a normal memory a *distributed* memory.

The SDM associates two binary vectors x and y by projecting x into a very high-dimensional intermediate word-line vector u, and then associating $x \rightarrow u$ and $u \rightarrow y$. Through its analogy with a conventional computer *random access memory*, the association of $x \rightarrow u$ is a kind of *address decoder memory*, and the association of $u \rightarrow y$ depicts the *data memory*.

The SDM is a feed-forward associative memory architecture that has two layers of input coefficients or weights that are represented by the matrices A and C. The matrix A is constant, and the matrix C is variable. The rows of matrix A are interpreted as a random sample of the hard-address space, and the rows of C are interpreted as the contents of those memory locations.

3.2.1. Storage

According to the above-mentioned analogy, a write process consists of associating a vector x to a vector u, and then associating the vector u to a vector y. A datum is written to the memory at all those hard locations that lie sufficiently close to the input address vector x.

In the initial SDM proposal, all hard locations consist of a set of integer counters that are initially set to zero, so no information is stored. During the write process an access radius ($r = 3$ in Figure 3.4) is used to select the hard locations closest to the input cue. The Hamming distance $d(x, x')$ between the input address x and a hard location x' must be less or equal than r. Each counter of the set of active hard locations (shaded rows in Figure 3.4) will be updated by adding a 1 if the corresponding position of the input data vector contains a 1 and subtracting 1 from each counter if the corresponding position in the input data vector contains a 0. However, another storage mechanism is to replace the contents of a vector instead of using de- and increasing counters.

An input pattern is stored into regions of memory where similar information has previously been stored. If a write operation is repeated several times, the contents of a particularly hard location will consist of the sum of all previous inputs. New data will simply be added to the existing one. This makes clear that every data pattern presented to an SDM will be stored, old patterns are not removed. Nevertheless, only little information of a once-stored particular datum may reside in a frequently modified hard location.

The probability of increasing or decreasing a counter value is equal if random binary data is stored into the memory. Regardless of how many patterns are stored, the statistical

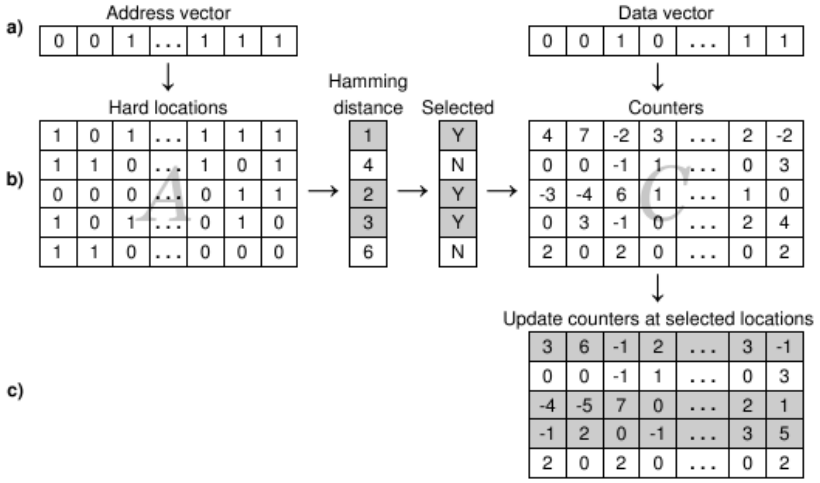


Figure 3.4.: Write a data-item into SDM (autoassociative case): (a) The input address pattern x is presented to the memory. (b) Locations x'_i with a Hamming distance $d(x, x'_i) \leq r = 3$ are selected (shaded areas). The activation radius r is set *a priori*. (c) Data patterns of the selected locations are updated by adding 1 to each counter in the array corresponding to 1, and subtracting 1 from each counter corresponding to 0 in the input data pattern.

expectation of each counter will be $\frac{1}{2}$. If the input address pattern is different from the input data pattern, in the sense that their lengths or their contents differ, the SDM is called to operate *heteroassociatively*. In case of identical input address and input data patterns, the SDM works as an *autoassociative* memory.

Whereas a heteroassociative mode is used to store associations only, an autoassociative mode is used to create a content-addressable memory, storing individual items that can be retrieved when given a partial input cue. But there is no reason why an autoassociative mode cannot be used to store associations between items (Kahana, 2002). It can be seen as a special case of the heteroassociative mode where the number of inputs is equal to the number of outputs.

3.2.2. Retrieval

According to the above-mentioned analogy, a read process is simply the presentation of a vector x, resulting in a word-line vector u, which is then used to read a value y out of the data memory.

Presenting an input address to the memory will collect the contents of each counter of the hard locations within a critical distance, the activation sphere, and reconstruct the output vector y based on a majority rule. Indeed, this majority rule can already be found in the *redundant hashing addressing* method of Kohonen (1989), even if its realisation is very different from Kanerva's. Since retrieved words are statistical approximations, the memory model has a sensitivity to similarity. In other words, similar (sensor) patterns trigger similar areas of the memory.

Similar to memory storage, a distance measure is used to determine whether an address will contribute to the output vector or not. Figure 3.5 depicts how information is retrieved

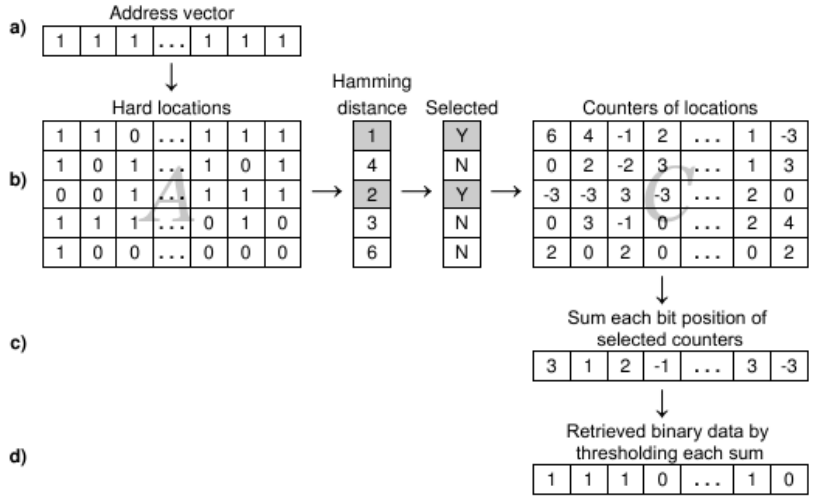


Figure 3.5.: Retrieve a data item: (a) Given an n-element binary address vector x. (b) Select all storage locations x'_i whose addresses lie within a Hamming distance of $d(x, x'_i) = r = 2$. (c) Add the values of these selected locations in parallel (i.e. vector addition) to yield a sum vector $sum(x)$ that contains k sums. (d) Threshold these k sums at 0 to obtain the data vector y with $y_i = 1$ if $sum_i \geq 0$ and 1 otherwise.

by polling the contents of all storage locations with a Hamming distance smaller or equal than $r = 2$ and finding for each bit whether the zeros and ones are in majority (see Figure 3.5). The selection of active hard locations is the same as in the storage procedure, however, the counters are not altered during this procedure. Figure 3.6 illustrates an alternative representation on how data is stored into and read from a memory with respect to the access radius r.

Various other methods to poll data from the memory are also described in Kanerva (1988, Ch. 7, pp. 66), but most are statistically impractical or computationally expensive. The methods vary from taking the datum that is stored in the closest hard location x' to the input address x or taking the most frequent word in the set of activated hard locations within the access radius. Another method is to randomly select a word from the set of active locations.

3.2.3. Convergence

If the input patterns that are stored to the memory are random, the point of each pattern in input space may be considered as an attractor. If a noisy pattern is presented to an SDM that is filled with multiple patterns and used *autoassociatively*, the resulting outputs can be iteratively fed back until the network converges on a *stable solution*. Kanerva (1993) shows that a clean version of a target pattern will be obtained if the Hamming distance between the input and target pattern initially decreases if the output is fed back to the SDM. The memory will converge to a different attractor if the distance between input and target patterns initially increases. This means, retrieving data from an address x' that is sufficiently similar to a physical hard location x will retrieve the target word y' that will be even more similar to y than x' is to x. If the input address pattern x' is beyond the critical

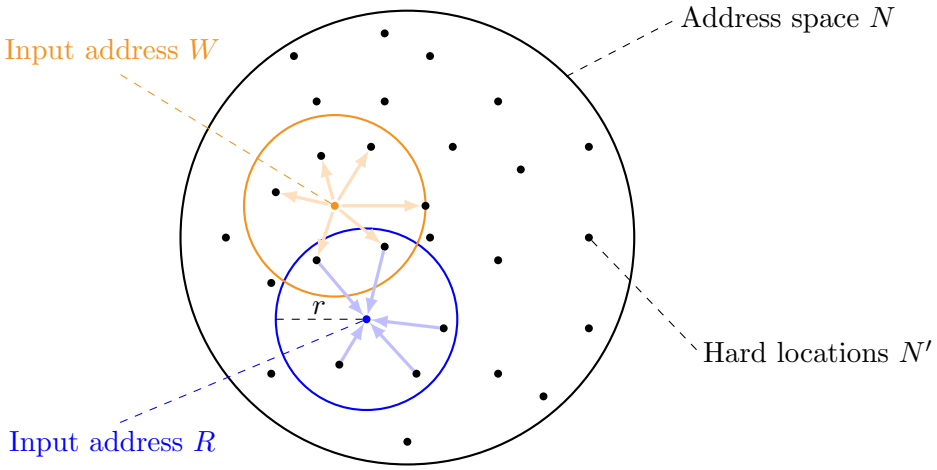


Figure 3.6.: The outer circle represents the boundary of the entire memory space N, each black dot is a physically existing hard location of the sparse representation N' of the memory space N. Though the trigger cues W and R may not correspond to any physical memory location, they will write memory contents into and read contents from sourrounding hard locations with respect to an arbitrary access radius r.

distance of x, it will fail to converge.

Nevertheless, iterative reading within the critical distance will converge quite fast to a clean version of the best-match word. This retrieval of the target pattern will not work if the memory is saturated or if many almost identical copies of a pattern have previously been stored (Hely, 2006). A fast convergence could indicate the phenomenon of "knowing that one knows" or "tip of the tongue" and has been described by Kanerva as *self-propagating search*.

3.2.4. Capacity

Various studies such as Kanerva (1988); Keeler (1988); Chou (1989) analysed the capacity of an SDM. Some also compared the capacity of an SDM to neural networks (Keeler, 1988; Fowler, 1991; Bose et al., 2005; Furber et al., 2007). If too many data items are stored in an SDM they can overlap and interfere. The retrieval of memories from a set of locations may include vectors that are stored to other addresses. Even if not intuitively obvious, the model permits remarkably dense storage due to the characteristics of the vector space before the capacity is exceeded (Wasserman, 1993). Nevertheless, if too many feature vectors are stored, the resulting input can be incorrect. Storing the same feature vector several times into the memory will increase its contribution when a nearby vector is retrieved. This is particularly related to the exposure frequency and recency property in long-term memory that affects the speed and probability of memory recall.

3.3. Different Represention of an SDM

An SDM, as mentioned before, can either be considered as a generalised random-access memory or as a neural network.

3.3.1. SDM as Generalised Random-Access Memory

By a proper choice of the SDM parameters, the model will exhibit the same behaviour as an ordinary *random-access memory* (RAM). The physical address space N' must contain all addresses of space N, this guarantees that all possible addresses x point to at least one hard location x'. The activation radius has to be zero, accordingly just a single address will be activated at any one time. A capacity of one bit per counter guarantees that old memory contents will be replaced by the new datum. According to this parametrisation, SDM is a generalised form of a random-access memory.

3.3.2. SDM as Artificial Neural Network

An SDM can be seen as a synchronous[6], fully connected[7], three-layer feed-forward *artificial neural network* (ANN) with generalised Hebbian-learning as depicted in Figure 3.7. The neural representation is composed of:

Input layer: The input pattern x will be presented to the input layer and its dimension n corresponds to the dimension of an input pattern x. It can be modeled as a set of perceptrons with fixed weights ± 1. A neuron will *fire* if the Hamming distance $d(x', x)$ of an encoded address equals or is lower than the activation radius r.

Hidden layer: The hidden layer contains m units each with values 1 or 0. The input coefficients of the hidden layer are fixed weights that correspond to the address matrix A. The k^{th} unit u_k of the hidden layer represents the k^{th} row of address matrix A (see Figures 3.5 & 3.4).

Output layer: The output layer has the same size n as the input layer. The input coefficients $C_{m,n}$ are related to the contents matrix C (cp. Figures 3.5 & 3.4). The values of the neurons are ± 1 and correspond to the computation of the counter arrays as shown in Figure 3.5. Hidden and output layer are linked via weighted synaptic connections w_i. The weights represent the bit counters mentioned in Section 3.2.1. The output layer represents the summing and thresholding unit according to the output computation of Equation 3.10:

$$y_n = \begin{cases} 1 & \text{if} \quad w_i > 0 \\ 0 & \text{if} \quad w_i < 0 \end{cases} \tag{3.10}$$

Some problems occur in neural models when used with high-dimensional algorithms. The storage capacity depends on the dimension of the input vector and only rather slow training algorithms are available. In an SDM the dimension of the input vector is independent of the capacity of the SDM. Since it is composed of only few layers, fast training algorithms are available.

3.3.3. SDM Analogy to the Cerebellum

Indeed, during development the SDM was never intended as a biologically plausible model of short-term (working) or long-term memory (Hely, 2006). Nevertheless, it has been considered

[6] All computations are completed in a single machine cycle, after which the network is ready to perform the next one.

[7] All elements of the matrices can assume nonzero values.

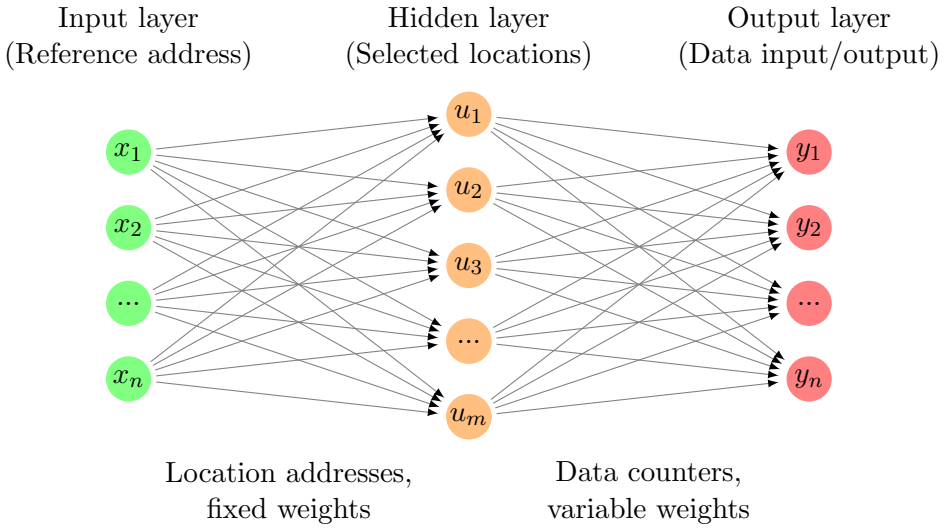


Figure 3.7.: Schematic diagram of an SDM as a three-layer feed-forward artificial neural network (ANN). For a reasonably large memory, there might be 1000 nodes in the in and output layer, 1000000 in the hidden layer.

alongside the theoretical models of the cerebellum by Marr (1969) and Albus (1971) as mentioned in Section 2.3.4.1 and 2.3.4.2, and represents the biological inspiration for the SDM model.

Kanerva (1988, p.90) indicates, as one of the major results of his study, where in the brain such a memory structure as the SDM might be found. Even if he does not assign the functionalities to all its cells or all its connections, he compares the SDM with the cerebellar cortex. Note that Kanerva himself mentions that the fitting of the SDM model to the cerebellar cortex is by no means perfect. He also leaves open the issue of whether addresses to the memory are used as data in the cerebellum (Kanerva, 1988). However, Table 3.1 summarises the relation between the SDM architecture and the cerebellum. This section, thus, depicts an extension of Section 2.3.4 and illustrates the functions of the neural units found in the cerebellum in the second column of Table 3.1. Accordingly, the third column describes the relations between the SDM model and the cerebellum.

The SDM is like the models of Marr (see Section 2.3.4.1) and Albus (see Section 2.3.4.2) in that they have a hidden layer with fixed coefficients that maps input patterns to subsets of storage locations. The storage and retrieval of a pattern takes place in the currently active subset, with most of the locations inactive at any one time. Common to all these models is that stored patterns are distributed over many memory locations. A single location is involved in the storage of different patterns and mathematical mechanisms are used to reconstruct the information from many locations (Kanerva, 1988).

Such distributed representations are typical for most associative memory models as mentioned earlier in this work. Some of lesser resemblance to SDM are mentioned by Anderson (1977), Hopfield (1982), Kohonen (1989), Willshaw et al. (1969); Willshaw (1981). A comparison between Kanerva's SDM and some Hopfield-type neural networks has been made by Keeler (1988).

Table 3.1.: Functional interpretation of the SDM framework related to structures of the cerebellar cortex. Summary of the comparisons made by Kanerva (1988) and Hely (2006). The structure of the cerebellum is shown in Figure 2.3

.

Structure	Function in cerebellum	Function in SDM
Mossy fibres	Originate from outside of cerebellum; Synaptic connections with granular cells	Address lines; transmission of input pattern to the memory
Granula cells	Located in the inner layer; Receive input from the mossy fibres or granular fibres and mossy cells	Address decoders; correspond to hard locations N'
Parallel fibres	Axons of granular cells; Each has from 200 up to 450 synapses to the Purkinje cells' dendrite planes	Activation of counters for memory locations; counters for given bits
Purkinje cells	Their axons form the only output of the cerebellum; Intersect and have synapses with up to 400000 parallel fibres each	Data polling for a single output bit; fires when sum is above threshold
Stellate and basket cells	Interneurons; receive input from parallel fibres; inhibition of Purkinje cells	Threshold adjustment; Decision of the majority of zeros or ones in polled data
Climbing fibres	Branch throughout Purkinje dendritic trees; Firing of climbing fibre guarantees firing of connected Purkinje cell	Data input line; indication of learning
Golgi cells	Inhibition of granular cells by feedback inhibition;	No interpretation made by Kanerva

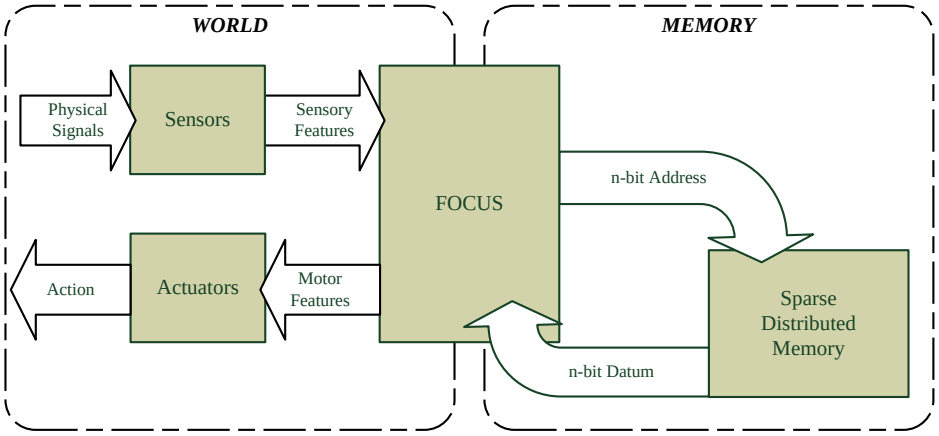


Figure 3.8.: The focus accounts for the system's subjective experience. Adapted from Kanerva (1988).

3.4. An Adaptive Autonomous Agent

According to Franklin (1995), and embodied intelligence an autonomous agent (e.g. robot) must not only sense, remember, recall and predict but also act upon its environment so as to affect subsequent sensory input. In animals, actions are mostly produced by subsequences of neural patterns driving muscles. Actions are included in the world model by storing these motor sequences in memory.

Kanerva's model already provides such a mechanism where deliberate actions become part of an agent's experience. Moreover, it is of major importance that the patterns from which the world model is constructed contain action components. The flow of information in and out of an SDM is through the *focus*, as shown in Figure 3.8. When the memory reproduces patterns in the focus it also reproduces the corresponding motor action to that particular pattern, ready to drive the motors. Kanerva (1988) considers this kind of system's response to situations and messages via actions as very basic mechanism of interpretation and meaning[8].

An entity in the focus is a high-dimensional feature pattern that encodes everything about a particular moment in time. Everything means the action, overall context, any specific things the system is attending to, *et cetera*. Oversimplified, Kanerva suggests to assign 80% of the focus' components to current sensory input and the remaining 20% to the corresponding motor output.

The memory is addressed by the focus, the contents are written into the memory, and the data from the memory fed into the focus. When a sensory sequence is used to address the memory, the memory responds with similar consequences as in the past. Information supplied by the senses and information supplied by the memory can produce the same subjective experience. Hence, the memory includes an expectation of an action's result. Accordingly, to plan an action, the system must initiate the "thought" in the focus while blocking the physical execution. The memory then retrieves the likely consequences of the contemplated actions into the focus and can use them as subsequent cue to trigger the next actions iteratively.

[8]The complexity arises from the infinite ways of deriving new meanings from old ones.

Kanerva's internal model is built from exposure to the world. Side effects of an action and its main effects are stored in the SDM. When the system moves the model is checked against the world. thus, the *frame problem* should never arise in an SDM[9].

3.5. Improvements of the SDM Design

Improvements in SDM design mainly differ in the selection of locations based on a certain input address. Some of those approaches will be shown in this section. The SDM has been subject to various studies on the capacity as analysed by Kanerva (1988); Keeler (1988); Chou (1989) and was compared to neural networks by Keeler (1988); Fowler (1991); Bose et al. (2005); Furber et al. (2007). Bouchard-Côté (2004a,b) investigated convergence properties of various reinforcement learning algorithms coupled with an SDM as linear, local function approximation architecture. Kristoferson (1995, 1998) showed that SDM performance does not scale up[10] in the original SDM, but does if sparse vectors are used.

3.5.1. Jaeckel/Karlsson's Selected-Coordinate Design

In the original SDM model every location is given a binary address. Hard locations that have a sufficiently small Hamming distance to an input address x are activated. Jaeckel's design is based on the assumption that two addresses with small Hamming distance should roughly activate the same set of locations (Sjödin, 1995). Jaeckel (1989) hypothesises that an activation pattern for each hard location is defined by specifying just a few *selected coordinates* and a set of corresponding *assigned values*, consisting of one bit for each selected coordinate.

Each hard location x' is assigned by a $mask(x')$ of K coordinates, i.e. a set of indices ranging from 1 to n. In the interpretation of SDM as a model of cerebellum, K corresponds to the number of mossy fibre synapses of a granula cell. Each of those indices is associated with either 0 or 1. A given location is activated, if and only if each of the k associated values of the $mask(x')$ coincides with the respective value of the input vector x. In this design with e.g. 1000-bit binary addresses, only 10 of the 1000-bits might be set to 0 or 1. The remaining 990 bits are not involved in the activation process. Figure 3.9 exemplifies the above-mentioned activation mechanism. The activation probability of a location regarding Kanerva's model is decreased to $p = 2^{-k}$.

The advantages of this design are: it is faster due to the abandoned computation of the Hamming distances, it uses a simpler XOR address-decoding and the selection of a hard location is independent of n. That the access circles expand, compared to Kanerva's design[11], can be seen as both a favourable and unfavourable side effect. The disadvantages are that the original error-tolerant SDM design becomes very sensitive with respect to the selected coordinates while insensitive to non-selected coordinates. A similar approach by Jaeckel is the *hyperplane design*, where fewer selected bits are used for addressing, e.g. 3. This works well for non-random data.

[9] The brain produces high-level representations that are stored into a memory. This would require less memory space than it would need when used for raw sensory data—as in this work.

[10] The ability to tolerate noise in the retrieval cue decreases with memory size.

[11] By using just a limited number of selected coordinates, e.g. 10, $2^{10} = 1024$ access circles can be represented regardless of the total number of bits.

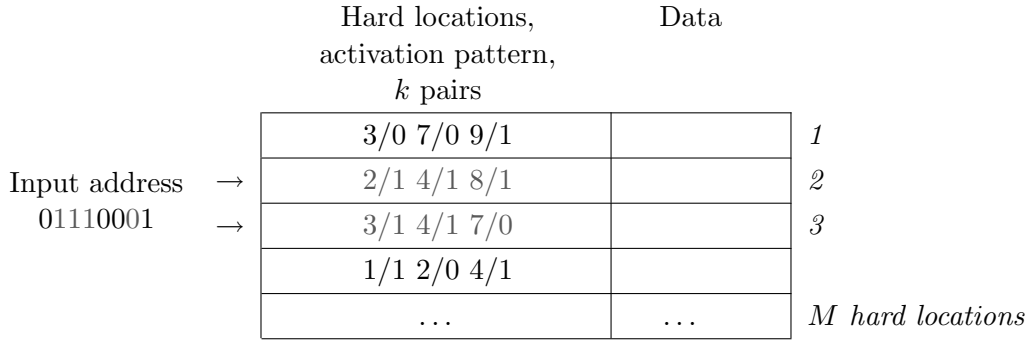


Figure 3.9.: Address activation (red) with $k = 3$ according to the selected-coordinate design by Jaeckel (1989).

A restriction of Jaeckel's hyperplane design by Karlsson (1995b) led to an advantage for realising SDM with standard, sequential computer architecture. The author proposes a slightly modified access mechanism that reduces the time-consuming search for the set of active hard locations within the entire physical memory space. Karlsson (1995a,b) divides Jaeckel's hard location memory into A blocks each with 2^k locations which yields to a fixed number of activations per access and thus considerably accelerates Jaeckel's model. For the selection of specific and representative coordinates, the author claims that fewer used bits have to be chosen preferentially. This leads to more randomised sets of activation patterns than in Jaeckel's design.

3.5.2. Spatter and Sparchunk Code

The *spatter code* (Kanerva, 1994) is a method of forming analysable holistic representations of higher-level, nested concepts[12] using real-valued or dense binary vectors, i.e. vectors with roughly the same number of 1s and 0s and fixed dimensionality.

A concept or item is represented by two N-bit *codewords*. The first codeword is a *high-level* (or *dense*) word with many randomly placed 1s and the second codeword is a *low-level* (or *sparse*) word with a few 1s. Dense codewords can be used as inputs to an associative memory while sparse codewords are used in encoding new concepts. A new item is formed by combining several items, e.g. attributes, concepts, chunks. If parameters are chosen properly, the two respective codewords are generated from the sparse codewords of its constituents as follows:

- the new dense word is the logical OR of the constituents (i.e. their sum thresholded at 0.5),
- and the new sparse word has 1s where the constituent words *overlap* (i.e. their sum thresholded at 1.5).

An interesting feature of spatter coding is a more efficient way of reading a memory by using implicit information to remove noise. Any bit or set of bits from a bit-vector can be lost or erroneous without losing any associations between concepts. The probability of finding

[12] In the following, the term "concept" is used to refer to the meaning of "representation of a concept". Concepts are represented by binary, real-valued, or complex values.

all of the associations is just decreased. In order to continue computation, noise has to be removed from the retrieved vectors. Denoising is established via a "clean-up mechanism" that searches for familiar concepts using noisy versions.

The *sparchunk code* (Sjödin, 1998) solves the same problem as the spatter code though using sparse codes. This representational model can be used as clean-up memory for finding constituent parts of a composite pattern while keeping same sparsity. Both of these methods can be used hierarchically (Kanerva et al., 2001).

3.5.3. Value-Based Reinforcement Learning

Ratitch and Precup (2004) contribute a remedy for the lack of *a priori* memory size assignment for SDM-based mapping of a large high-dimensional address space onto a smaller physical memory. The authors propose a *randomised reallocation algorithm* for dynamic online allocation and adjustment of memory resources, which eliminates the need for choosing the memory size and structure *a priori*. Their algorithm, that has been evaluated in Ratitch et al. (2004), starts with an empty memory. Hard locations are added based on the distribution of observed data. New locations are allocated randomly in the neighbourhood of an input address when there is a new datum which cannot be stored into a certain number of existing hard locations. For this, the authors presume that the activation radii of the memory locations are uniform and fixed by the user.

The minimal number of desired active locations for any data sample is denoted by F. A new hard location x' to an input address x is added to the memory centred at x, if the similarity condition (Equation 3.11) is not violated.

$$\mu(x'_i, x'_j) = \begin{cases} 1 - \frac{1}{F-1} & \text{if } F \geq 3 \\ 0.5 & \text{if } F = 2 \end{cases} \tag{3.11}$$

This similarity condition for any pair of hard locations x'_i, x'_j ensures that the fewer locations are required, a small F, the farther apart these hard locations should lie within the activation sphere. Should the number of active locations $F' < F$, then $(F - F')$ locations are randomly placed in the environment according to Equation 3.11. This method obtains an appropriate coverage of address space while controlling memory size. However, should a maximal memory size be reached and for any new input address x still hold condition $F' < F$, then, at random an inactive hard location is picked and removed and the corresponding number of new locations is added in the neighbourhood of x.

If condition 3.11 fails and a location x' has to be removed from an active set of locations, x' and its nearest active neighbour x'' are replaced by a newly added, intermediate hard location whose datum results from averaging x' and x''.

The proposed linear and local function approximation scheme is especially suited for online value-based reinforcement learning. It is cheap in terms of computational costs and the space required. Its randomised nature of removals and the care for a sufficient number of hard locations within visited regions ensures that it does not affect particular areas of the input space dramatically.

Further online, but unsupervised learning methods adjusting the SDM memory layout are proposed by Rao and Fuentes (1998); Sutton and Whitehead (1993). The latter approach by Sutton and Whitehead slowly moves existing inactive memory locations[13] toward observed

[13] Any inactive location is selected at random.

data if the number of active locations for a given training sample is too small.

3.5.4. Signal Propagation Model

An alternative approach by Hely et al. (1997) *propagates* an input signal uniformly throughout the memory space. The signal strength is diminished each time it encounters a hard location further from the input address. This is also known as a *radial basis function* (RBF), but here diminishing follows an arbitrarily defined, decreasing step function. Rather than binary, the hard locations receive a copy of input data proportionally weighted with a real value from $1.0 \mapsto 0.05$. Nearest locations are rewarded with greater signal strengths. Information retrieval is realised *vice versa*. The model periodically deletes rarely activated hard locations to free resources for memory allocation elsewhere. This technique referred to as *pruning* yields memory with only "fit" locations that survived the erasing period. This dynamic memory is better suited to deal with nonrandom data than Kanerva's original model.

3.5.5. Genetic Sparse Distributed Memory

Genetic algorithms (Holland, 1992) are adaptive procedures for optimisation, search and machine learning problems. They simulate an evolutionary process based on natural selection, mutation, crossover and inheritance. Whenever there is a large search space without proper algorithms for pruning and trimming, genetic algorithms can be used.

Several authors like Fan and Wang (1997); Anwar et al. (1999); Anwar (2005) propose to use SDM in combination with genetic algorithms. The former approach uses a genetic initialisation of hard locations while the data reading procedure is preserved. Thus, memory requirements are reduced at the expense of generalisation and learning capabilities as the memory becomes saturated. The latter approaches keep SDM as it is and invoke genetic algorithms to obtain a uniform distribution of hard locations. Anwar (2005) mentions a difference between the similarity of hard locations and the similarity of written memory words. Thus, their goal is to reach a certain group fitness (see Equation 3.12), meaning that hard locations of a certain population[14] are as far apart as possible from each other on average, with a limit on the standard deviation. The applied fitness measure for a population of hard locations is to maximise their distances to each other:

$$\max \sum d(x'_i, x'_j) \qquad \text{with } i \neq j \text{ and } i, j \in \{1, \ldots, n\} \tag{3.12}$$

If seen as SDM, a genetic algorithm changes the weights in the connections between the input layer and the hidden layer, while connections between hidden layer and output layer are changed according to the standard SDM method.

3.6. Applications

Various applications have been implemented since Kanerva firstly proposed his SDM. Simple memory applications vary from two-dimensional character recognition (Hong and Chen, 1991; Fan and Wang, 1997), to speech recognition and pronunciation (Clarke et al., 1991; Joglekar, 1989), to parallel tree-structure realisation (Hämäläinen, 1996) and connection

[14] Hard locations are the population of binary strings that serve as the domain of the genetic algorithm.

machine implementation (Rogers, 1988; Turk and Görz, 1995). Some of the approaches of particular interest for this study, by no means all, will be introduced while focusing on the diversity of application and major relevance to robotics.

3.6.1. Cultural Evolution

Memetics is a scientific effort to apply evolutionary models to the development and transfer of ideas within a culture (Gabora, 1996, 2002). The term *meme* is popularly attributed to Dawkins (1976), who proposed the idea in his book "The Selfish Gene". The Merriam-Webster Online Dictionary defines a meme as an idea, behaviour, style or usage that spreads from person to person within a culture.

Gabora (1996) outlines SDM as a way to generate variations of patterns by exploring or transforming the space. Working memory (WM) can be viewed as memes that lie within a given Hamming distance of a meme in the focus such that they are retrievable with a certain number of iterations. She stated that associations between memes are not explicitly represented as connection strengths but as proximity in multidimensional space. And this reflects the neurons' connectivity.

Gabora (1996) outlines that memes in an SDM have a self-replicant capacity via implicit pointers and details how an SDM-like *stream of thought* might get established. Gabora (1996, Ch. 7.1) states the interesting hypothesis that *unconsciousness* may be a result of fleeting experience of memes that are dynamically reconstructed as in an SDM but which do not readily assimilate with other memes and thus get discarded from the focus.

3.6.2. Part of a Cognitive Architecture: LIDA

The Learning Intelligent Distribution Agent (LIDA) (D'Mello et al., 2005, 2006; Franklin, 2005) is a software agent based on a global workspace theory that offers perceptual, episodic and procedural learning capabilities. It provides a conceptual and computational model of cognition and is partly symbolic and partly connectionist with symbols being grounded in the physical world (Harnad, 1990). It uses variants of SDM to computationally model declarative memory for the long-term storage of autobiographical and semantic information as well as short-term *transient episodic memory* for detailed sensory-perceptual information with a retention rate within hours.

A transient episodic memory holds activation patterns gathered from perceptual entities by primitive feature detectors. Not decayed contents of transient episodic memory are periodically consolidated into declarative memory. Recall of an event is established by recovering perceptual symbols from the activation trace within a perceptual associative memory.

3.6.3. Mobile Robot Navigation

The most interesting field regarding this work are autonomous mobile robots that use an SDM. A framework for learning perception-based navigational behaviours is proposed by Rao and Fuentes (1998). The SDM is used to attain high-level goal-directed navigation. It is combined with a stochastic hill-climbing method to achieve low-level reactive behaviours essentially based on Brooks' subsumption architecture (Brooks, 1986).

Contrary to the nearest-neighbour look-up table technique, the curse of dimensionality[15] is avoided by employing only a sparse number of active memory locations and a layered architecture for goal-directed navigation.

Within a training phase, they feed the SDM with current motor inputs indexed with probability vectors of the current and two preceding perceptions. Such that the robot is able to predict the next two motor commands according to an estimation during navigation. The robot is remotely controlled by a human user during a training stage. An estimation of the current motor output $\hat{a}_t$ during autonomous memory-based navigation is obtained by a weighted averaging method:

$$\hat{a}_t = \sum_{i=1}^{s} \gamma_i(t) a_t^i \tag{3.13}$$

where $\gamma_i(t)$ is a weight according to a predicted estimate i of a current situation based on the number s of current and past sensory inputs involved. A drawback of this approach is that the weights are predefined and not determined online in consideration of the current and past perceptions.

Another work on SDM-based robot navigation is proposed by Mendes et al. (2007, 2008, 2009). A tank-style robot is remotely controlled during a training phase and images from a digital camera together with the corresponding motion command are stored into the SDM. During an autonomous run the robot predicts the associated motion based on images presented to the memory. Instead of *a priori* memory allocation, the authors use the randomised reallocation algorithm (cf. Section 3.5.3) to dynamically allocate new hard locations if needed. While following a learnt path autonomously, small drifts of the robot are adjusted according to a horizontal displacement of the current to the predicted view. Images are enhanced by a histogram equalisation before storing into memory to make the system more robust to changing lighting conditions.

The work at hand, in parts, is closely related to the approach mentioned above. A comparison of the performance of the latter approach and the studies presented in this book is extensively discussed in Chapter 6. Hence, the LIZARD robot system is described in more detail in Section 4.1.2 & 4.2.2. Moreover, this study presents the first SDM-based approach to mobile robot manipulation and crossmodal interactions.

3.6.4. Weather Forecasting

Rogers (1990) uses an SDM and Holland's genetic algorithm to obtain weather forecasts for the Australian coast based on 58000 weather samples taken every 4 hours over 25 years. The goal is to predict rain given 15 local weather features such as air pressure, cloud cover, month, temperature, *et cetera*. The main mechanism used is genetic recombination, also called crossover (see Figure 3.10). Based on a fitness function as a measure of statistical predictiveness of a memory locations, two highly-ranked hard locations are used to replace a low-ranked one. The result of a recombination of two highly-ranked hard locations as shown in Figure 3.10 is added to the memory while a low-ranked hard location is erased. Thus, a better distribution of address space is reached.

[15] The number of samples per variable increases exponentially with the number of variables to maintain a given level of accuracy. The curse of dimensionality can only be avoided if several or all input features are somehow dependent among each other or if the input-output correlation is "simple" such that an output just slightly changes if an input is modified.

```
1010101011111111...   First parent
1101101101101101 0...  Second parent
            ↓
101010101110110111...  New member
```

Figure 3.10.: Crossover of two binary strings.

The weather samples are coded into 256-bit words such that each feature assigns a thermometer-style 16-bit field in the address. The mixture of SDM and genetic algorithms showed a better weather predictability than the mere SDM model.

3.6.5. Speech Recognition and Pronunciation

Based on a modified SDM model, Clarke et al. (1991) implemented a short word recognition system based on continuous speech inputs. First, they use a nonlinear mapping (location matching) of preprocessed speech into a high-dimensional representation followed by a single layer of conventional adaptive links that perform a linear mapping for complex pattern discrimination.

Trained by the multi-speaker *Alvey Hotel Speech Corpus*, they reached a recognition accuracy of 95% without syntactic constraints for the recognition of 133 different, but not considerably varying small words in continuous speech. The modified SDM is fed with spectral information from one second of speech to determine which of the small words[16] were presented during the middle $\frac{1}{10}th$ of the second. A Holmes 19-channels filter bank with an energy function is used to provide 20 parameters for each $20ms$ of speech. 50 of these $20ms$ sections are combined to form an input vector composed of 1000 integers. The hidden layer consists of 9600 units. The hypercube metric is used for the location matching an the radius is chosen such that about 10% of the locations are active for any given input pattern.

The model is trained with 347 labelled utterances of 25 sentences with different features deriving phonetic labels. 350 unlabelled utterances of the sentences are used for testing. Along with the above-mentioned accuracy, the results indicate that recognition of larger speech units is more reliable and repays the computational effort needed to process large sections of the speech signal as a single pattern. The recognition accuracy decreased while adding background noise. It shows a higher sensitivity to the speech volume than to the speech speed.

The interested reader is referred to further SDM-based approaches for the detection of phonemes (Joglekar, 1989), phonemes and vowels (Prager and Fallside, 1989) and spoken numbers (Danforth, 1990) in continuous speech.

3.7. Concluding Remarks

The SDM model depends fundamentally on subtle, nonintuitive properties of high-dimensional metric spaces and random distributed representations. A unique feature is that SDM can be mapped onto physiological structures as shown in Section 3.3.3. Many neural models only duplicate a style of computation and are not intended as models of brain functions. For instance, back-propagation is a paradigm commonly used in neural network research.

[16] Inspired by the morph unit in MITalk.

Unfortunately, after evaluating an error function the result is propagated backwards through the whole network to modify the corresponding weights. There are hardly any mechanisms found in the brain that justify backpropagation.

SDM's concept of distributed storage can account for the non-specific affects resulting from brain damage. A hardware implementation of an SDM is simpler as opposed to other neural models, which normally work fine with a small number of nodes in their network. Problems of classical NN models, especially in the class of backpropagation networks, occur while trying to provide communication paths for a fully connected network. Kanerva's description of a hardware implementation uses a considerable number of counters that does not incorporate information from other nodes, thus a communication bottleneck does not appear as in classical NN models.

Kanerva raises an analogy between the focus of attention of a human and an address-datum register of a standard computer. Both hold data and serve as a reference address to a memory register where information can be stored into or retrieved from. His mathematical theory explains how humans might be able to associate similar situations or fuse different impressions to a single one without even actively thinking about it. Each element of such addresses, such long binary vectors, encodes a feature of a prevailing situation of an agent. According to Hely (2006) it is not necessarily important that the vectors are binary. It is assumed that similar situations cause similar feature-vectors, which makes the information and feature encoding a central issue within SDM research.

If any input address is used to access the sparse memory, no distinction is made between physically existing or not existing addresses. An address-decoder takes care that all those memory locations get activated which are sufficiently similar to the target input address. Such similarity is measured as the number of differing bits which is the Hamming distance. This generalisation of the read and write operation permits features of adjacent regions to get intermixed. The memory locations in such overlapping regions contain an average of the stored information. If a memory address is considered as the property of a situation and its content as the address to a succeeding situation, sequences of actions are represented.

3.7.1. Advantages of the SDM Model Regarding Cognitive Robotics

The main advantages of using a memory model like SDM for robotic applications will be summarised with respect to the research questions and requirements mentioned in Section 1.2.

- The SDM model is suitable for working with high-dimensional feature patterns which will easily be reached in robotic domains when dealing with several low-level sensors and actuators.

- The SDM model is sensitive to conceptual similarity due to the statistical reconstruction mechanism of stored information. Thus, similar percepts such as robot motion patterns can be reconstructed based on a kind of abstraction. Motion patterns that share similar features will be favourable reconstructed by the memory. The robot will be able to reconstruct predict data based on the current context.

- The inherent tendency towards orthogonality in high-dimensional spaces causes the favourable matching properties in SDM.

- The model provides a huge address space by just allocating a non-exorbitant number of physical locations. For identifying and activating data locations, input addresses are not required to match stored addresses exactly but rather lie within a specified radius. This will enable a robot to process data patterns that are not identical but similar to previously learnt patterns and thus provides a mechanism for generalisation and abstraction.

- The SDM model is applicable to parallel processing, when implemented in hardware. Activation then will be immediate and the number of hard locations does not influence processing speed. This beneficial feature would allow a robot to quickly access contents of its memory, e.g. for comparing the current situation with a large amount of past experience to choose an appropriate action.

- Due to the distribution of multiple copies of the input vector, the memory is robust to noise, failure of individual locations and incomplete input patterns. Robots are frequently confronted with partial input patterns that have to be compared with learnt information.

- Due to the location pruning algorithm, location matching networks can be trained in roughly $\frac{1}{2}$ to $\frac{1}{10}$ of the training iterations required by commonly used single or double layer adaptive networks for an arbitrary task (Prager, 1993). Associative memories have a far quicker learning cycle than backpropagation networks, and have been shown to have preferential characteristics after training in some domains. Reducing the number of training cycles is an important requirement for cognitive developing robots, as well as for the robots proposed in this work.

3.7.2. Disadvantages of the SDM Model Regarding Cognitive Robotics

The main disadvantages of Kanerva's model with respect to robotic applications that operate in natural environments in the opinion of the author are as follows:

- The standard SDM model requires *binary address and data vectors*. But most natural environments yield multivalued input patterns. Therefore, either the indexing mechanism must be modified or all input patterns have to be transferred into binary space. As shown in this chapter, binary spaces have some disadvantages.

- The standard SDM model assumes a *random and uniform distribution* for the input vectors. Natural environments are highly structured and therefor hardly ever random. Normally, they tend to be clustered in many correlated groups distributed over a large portion of a multidimensional address space. Approaches that self-organise the input address space should help here.

- Using counters degrades the storage capacity K per bit. Avoiding counters would decrease the required memory space while memorising the same amount of data. Robots that are controlled by a single PC may have to share their memory space with other applications that use image and video processing which usually demands a large memory capacity.

- The SDM needs to cycle through all the addresses during training or recall. The amount of cycles needed directly scales with the number of neurons, the number of hard

locations respectively. This disadvantage is negligible when implemented in parallel hardware.

- If the SDM model is realised in software and not in hardware, the potential parallel nature cannot be fully deployed. Processing time increases.
- The memory is good only for the sparse case. The performance degrades rapidly if the number of items to be filled up grows (Bose, 2003).
- Contents written to the SDM cannot be removed but only forgotten. Dispensable contents may interfere with more recent contents of the memory. This can also be a desired feature of a technical system.

Most of the presented disadvantages will be compensated throughout this work by using and developing appropriate extensions.

4

Experimental Platforms

Those parts of the system that you can hit with a hammer (not advised) are called hardware; those program instructions that you can only curse at are called software.
(Levitating Trains & Kamikaze Genes: Technological Literacy for the 1990's)

Contents

This Chapter presents all necessary details about the hard- and software of the utilised robot systems. Further, a brief introduction is given about a telemanipulation system that has been developed for and used in this work. Within the scope of this work, the service robot TASER is mainly used for SDM-based manipulation. LIZARD is used for SDM-based navigation. The telemanipulation system is used for an interactive creation of diverse robot arm motion sequences by human instructors.

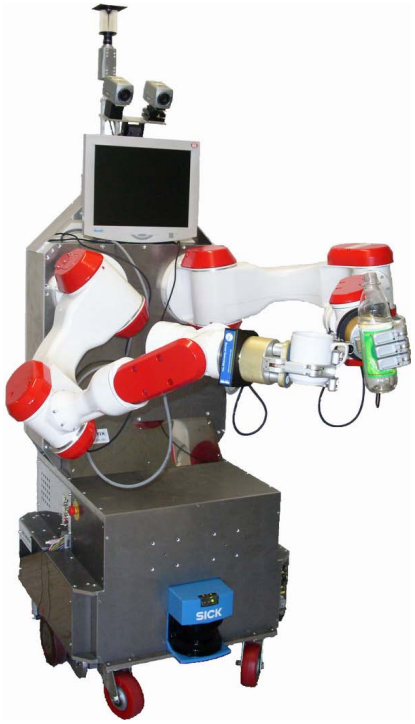
4.1. Hardware

This section illustrates and explains the setup of the hardware components used in this work. The main platform for scientific investigations on SDM for robots is the *TAMS Service Robot* (abbr. TASER). Further hardware to compare the performance of an SDM between different

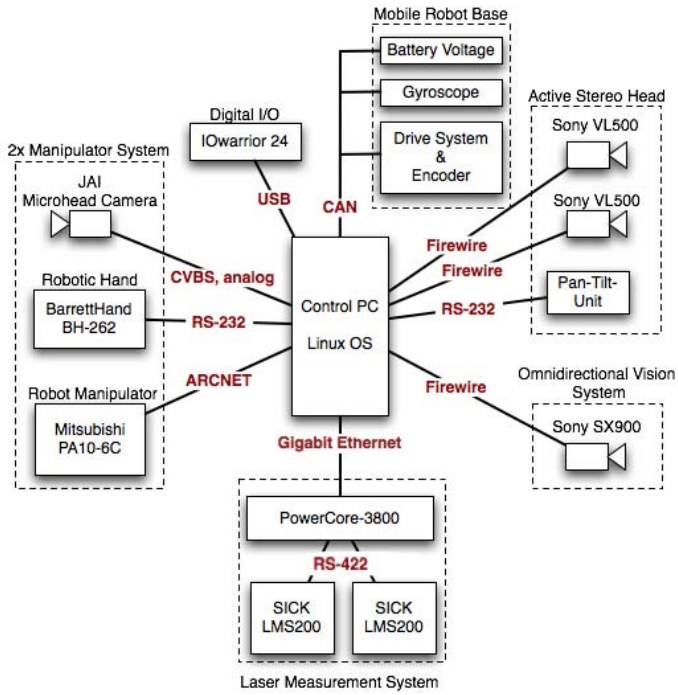
robots and domains is the LIZARD robot. Robot arm trajectories, the main subject of this research, have been produced by both direct programming and interactive control using a haptic interface.

4.1.1. TAMS Service Robot TASER

TASER consists of a mobile platform with a differential drive and wheel encoders, two laser range finders, a Pentium IV 2.4 GHz industrial computer as central control unit, a PA10-6C manipulator, a three-finger robotic hand and several IEEE 1394-cameras as shown in Fig. 4.1(a). The robot has a body height of appr. $2m$. In this work, SDM learning focuses on 6 degree-of-freedom (DoF) robot arm trajectories for manipulation tasks.



(a) TASER in its aspired final configuration.



(b) Current TASER I/O structure. Adapted from Bistry et al. (2007a).

Figure 4.1.: The TAMS service robot.

4.1.1.1. Mobile Robot Platform

The mobile platform is a modified version of NEOBOTIX[1] MP-L655. It consists of an aluminium chassis with shelves as housing for the internal power supply, the control PC and all controllers of the remaining hardware units. Eight lead-gel-batteries provide the robot with a supply voltage of $48V$. The platform is equipped with a motor system consisting of a differential drive and wheel encoders. A single Pentium IV 2.4GHz industrial computer with 1GB RAM and a Linux operating system serves as the central control unit for the whole

[1] `http://www.neobotix.de`, last accessed Mai 09, 2009.

Table 4.1.: Joint limits for PA10-6C.

	Limit			
Name of axes	Mechanical	Servo	Software	max. Velocity
S1 (Rotation)	$\pm 180°$	$\pm 178°$	$\pm 177°$	± 1 rad/sec
S2 (Swing)	$+127°, -67°$	$+125°, -65°$	$+124°, -64°$	± 1 rad/sec
E1 (Swing)	$+164°, -113°$	$+159°, -108°$	$+158°, -107°$	± 2 rad/sec
E2 (Rotation)	$\pm 270°$	$\pm 256°$	$\pm 255°$	$\pm 2\pi$ rad/sec
W1 (Swing)	$\pm 180°$	$\pm 166°$	$\pm 165°$	$\pm 2\pi$ rad/sec
W2 (Rotation)	$\pm 270°$	$\pm 256°$	$\pm 255°$	$\pm 2\pi$ rad/sec

robot system. The sensor/actuator peripherals are connected to the control PC as shown in Figure 4.1(b). For further details on TASER, cf. the work by Baier-Löwenstein (2008) and Weser (2009).

To perceive the environment, the robot possesses two laser range finders, two IEEE 1394-cameras as a stereo head with a pan-tilt-unit, an additional omnidirectional vision system and force sensors inside the fingers of the robotic hands. A gyroscope can be used to measure the robot changing its orientation based on angular momentum.

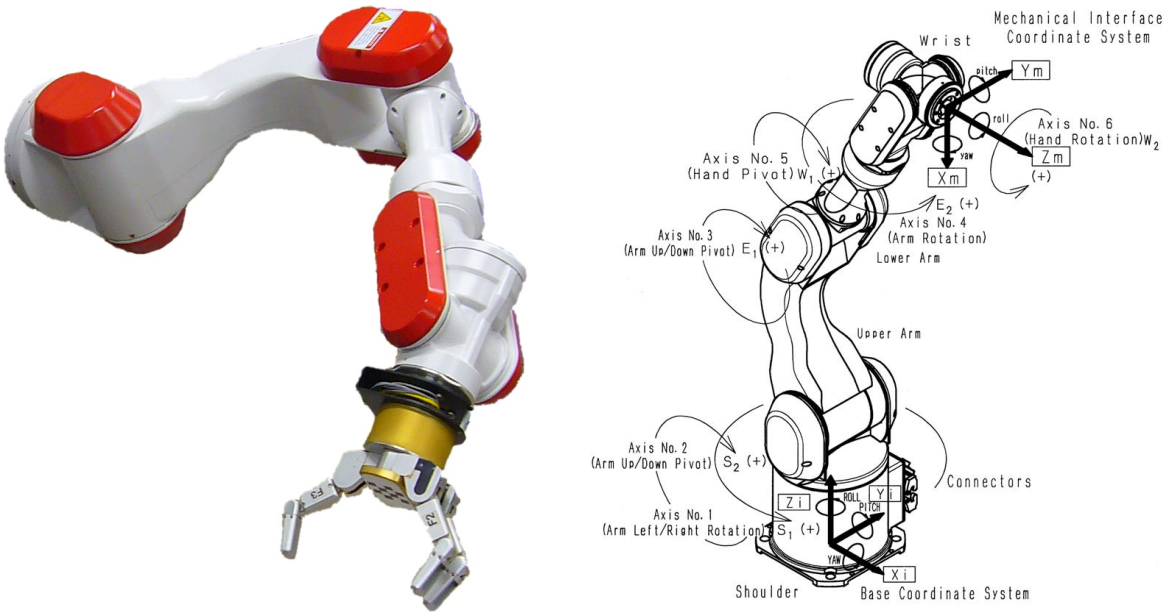
Range measurements are used for different tasks such as collision prevention, self-localisation or people tracking. TASER uses two SICK[2] Laser Measurement Sensors (LMS) 200 mounted at the front and at the back of the robot. Each sensor has a 180° field of view and is operated with an angular resolution of 0.5°. A special purpose RS422-to-Ethernet adapter based on a Rabbit PowerCore 3800 microcontroller was designed by Bistry et al. (2007a,b) to preprocesses the range data and send UDP-packages over Ethernet to the robot's host PC.

4.1.1.2. Manipulator

The manipulation unit of TASER is composed of an MHI PA10-6C robot arm with six degrees of freedom (DoF) manufactured by Mitsubishi Heavy Industries[3] and a three-finger gripper by Barrett Technologies Inc[4]. It is shown in Figure 4.2(a). The robot arm is mainly used for fine-positioning of the gripper to grasp objects precisely. Trajectories of the robot arm for the positioning of the robot hand are recorded and used in this work for SDM-based learning and recognition.

MHI PA10-6C Robot Arm TASER's robot arm is a commercially available industrial robot. It has a total length of $1317mm$ (see Figure 4.3), a weight of $39kg$ and a maximum payload of $10kg$. The robot arm has six DoF. Its working range is comparable to a human arm with seven DoF[5]. In general, the higher the degrees of freedom, the more positions a robot arm can reach with its tool centre point (TCP).

[2] `http://www.sick.com`, last accessed Mai 11, 2009.
[3] `http://www.mitsubishi-heavy.de/`, last accessed August 13, 2009.
[4] `http://www.barrett.com/`, last accessed August 13, 2009.
[5] The kinematic chain of a human arm consists of different joints, namely the shoulder with three DoF, upper and lower arm one DoF, lower arm torsion one DoF, wrist two DoF.



(a) MHI PA10-6C and BarrettHand.

(b) Alignment of PA10-6C axes.

Figure 4.2.: PA10-6C robot arm by Mitsubishi Heavy Industries Ltd. combined with a gripper by Barrett Technologies Inc.

The arm is driven by low-maintenance, brushless DC motors and each joint has a brushless DC resolver for a positioning repeatability of $\pm 0.1mm$. The alignment of the PA10-6C axes is shown in Figure 4.2(b) while the limits of the rotation angles are listed in Table 4.1.

The PA10-6C is controlled by a standard PC via ArcNet interface. The Robot-Control-C-Library (RCCL) developed by Lloyd and Hayward (Lloyd and Hayward, 1992; Hayward and Richard, 1986) was supplemented for PA10-6C and PA10-7C control by Scherer (2004). A C++ extension of RCCL to RCCL++ developed by Markus C. Ferch and Yorck O. von Collani at the University of Bielefeld supplements the control software with methods for complex movements of multiple types of robots. The latter extension permits torque-based arm control[6].

BarrettHand The commercially available BH8-262 BarrettHand produced by Barrett Technologies Inc. is a multi-fingered programmable grasper. It is mounted as end effector onto the MHI PA10-6C robot arm and is connected to the control PC via RS-232. The hand has 8 axes and is driven by four brushless DC motors. The dimensions are $298mm \times 149mm \times 42mm$ (see Figure 4.4) and its weight is $1.2kg$. The BarrettHand has a payload of $6kg$. According to the above-mentioned specification and if operated at a supply voltage of $48V$, the payload of the whole manipulation unit (arm and hand) recedes to approximately $4kg$.

Along with the servo-controllers and the four brushless motors, the BarrettHand houses a CPU with an industry-standard RS-232C serial communication link. A single motor controls

[6]Since the PA10-6C does not have any force-torque sensors. The mentioned torque-based arm control is based on forces measured with other devices connected to the robot arm, e.g. the BarrettHand (Baier-Löwenstein, 2008).

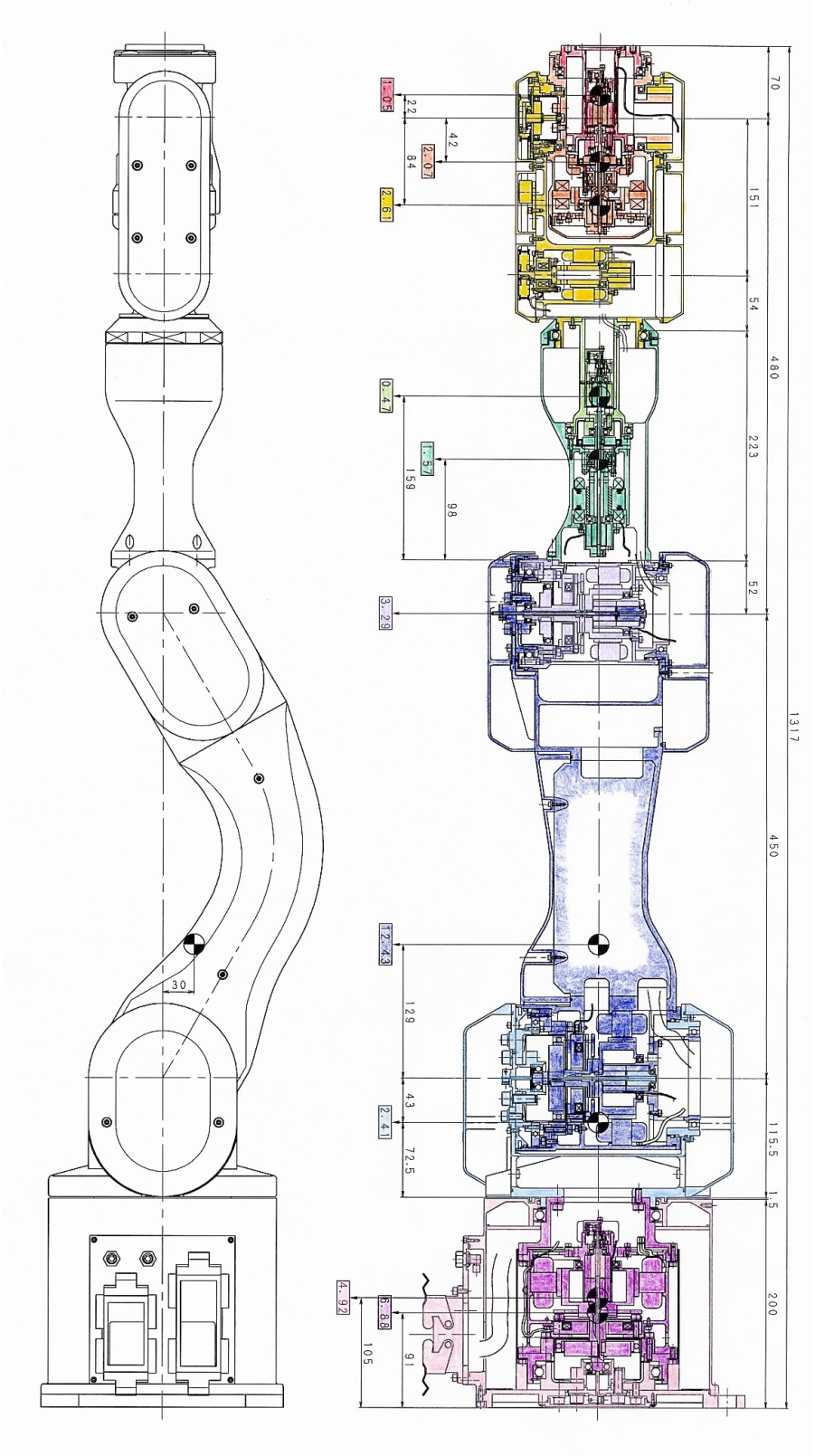
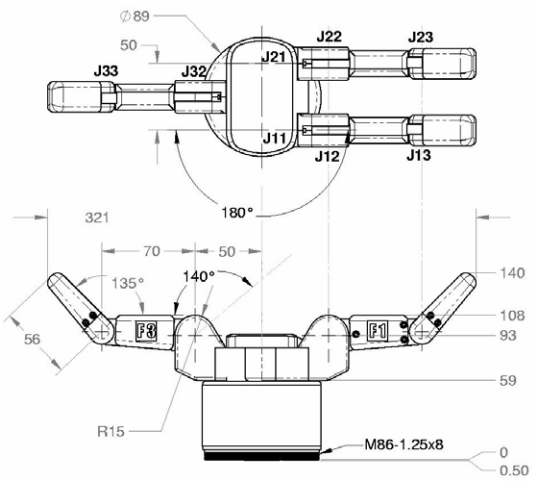
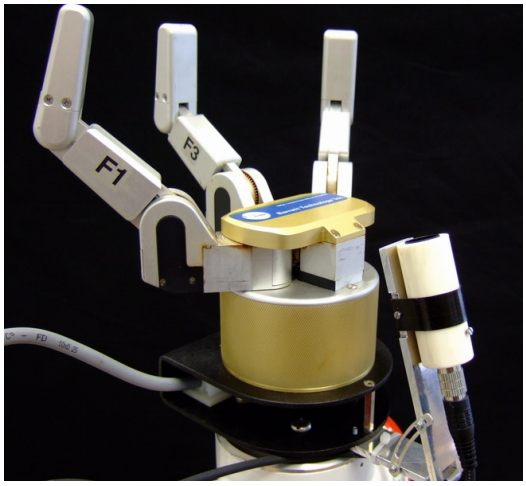


Figure 4.3.: Dimensions of the PA10-6C robot arm by Mitsubishi Heavy Industries, Ltd.



(a) Metric dimensions of the BarrettHand in *mm* and the joint radii. Source: `http://www.barrett.com`



(b) BarrettHand mounted at the robot arm with micro-head camera.

Figure 4.4.: BarrettHand BH8-262.

both joints of a finger (Jx1 and Jx2 in Figure 4.4(a)) via a special gear box mechanism called *TorqueSwitch*™ (Townsend, 2000). According to Brown (2008), the switch drives the extensions above both joints to curl forward at the same time. When one meets resistance, it stops, but the other keeps going until it meets resistance, too. A fourth motor rotates two fingers (F1 and F2 in Figure 4.4(a)) together up to 180° around the palm (J11, J21 in Figure 4.4(a)) which finally leads to eight DoF of the robot hand.

The BarrettHand uses optical incremental encoders that lead to a sensing resolution of 0.008° at the finger base joints. Each finger of the BarrettHand houses a resistance strain gauge (RSG) in the proximal finger limb to measure forces that act on the gear cable. Only perpendicular forces exerted onto the distal finger limb are noticeable in consequence of the mechanical architecture of the BarrettHand. C++ libraries developed at the Dept. of Informatics, University of Hamburg by Bernd Rössler and Baier-Löwenstein (2008) are used for the programming.

4.1.1.3. Vision Sensors

Besides the sensors already mentioned in the sections above, TASER is equipped with several IEEE 1394-cameras (see Figure 4.5). Two are used in combination with a pan-tilt-unit as a stereo-vision head. Furthermore, an omnidirectional vision system is mounted at the top of the robot.

Stereo-Vision System Two Sony DFW-VL 500 cameras are mounted with a baseline of $110mm$ onto a pan-tilt-unit PTU-46-17.5 produced by Directed Perception[7] on top of the robot. The maximal resolution of the cameras is $640px \times 480px$ in a YUV422 colour space

[7] `http://www.dperception.com`, last accessed May 14,2009.

Figure 4.5.: The camera systems mounted on top of TASER.

at 30 frames per second. They are connected to the control PC via IEEE 1394 while the PTU is connected via RS-232.

The resulting stereo system can be used for natural learning of manipulation skills (Hüser et al., 2005, 2006, 2007), 3D scene and depth reconstruction (Jockel et al., 2007a; Klimentjew et al., 2008) and people tracking without moving the mobile platform (Weser et al., 2006).

Omnidirectional-Vision System A high-resolution Sony DFW-SX 900 camera with a maximal resolution of $1280px \times 960px$ is used for an omnidirectional-vision system. The camera captures light of nearly all directions falling onto the focal point of a hyperboloidal semisphere and thus has a 360° field of view. The camera uses YUV422 colour space and captures 7.5 frames per second if operated at full resolution. Omnidirectional cameras are mostly used for visual odometry and simultaneous localisation and mapping (SLAM) problems.

4.1.2. LIZARD Robot

The LIZARD robot (Mendes et al., 2008; Jockel et al., 2009) shown in Figure 4.6 is based on a Surveyor SRV-1 Robot[8]. The name is derived from the Portuguese word "*lagarto*" and has similar meaning. It is an open-source wireless robot with video for telepresence that can be operated as a remotely-controlled webcam or self-navigating autonomous robot.

[8] `http://www.surveyor.com`, last access April 29, 2009.

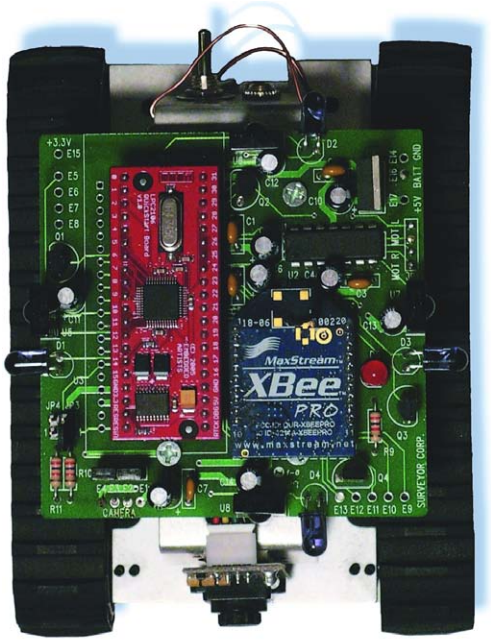


Figure 4.6.: The Surveyor SRV-1 robot LIZARD.

4.1.2.1. Surveyor SRV-1 Robot

The Surveyor SRV-1 is made from machined aluminium and has a weight of approximately $300g$ with the dimensions of $120mm \times 100mm \times 80mm$. It is powered by a $7.2V$ Lithium-ion polymer battery with $2Ah$ which permits a maximum operation duration of four hours per charge. LIZARD has two precision DC gearmotors controlled by a FAN 8200 with differential drive that actuate two tank-style treads. The operational speed lies between $20\frac{cm}{s}$ and $40\frac{cm}{s}$ with an operating distance of up to $100m$ indoors and $1000m$ outdoors (line of sight) limited by the communication module. The processor is a 60mips 32bit ARM7TDMI mounted on a Philips LPC2106 QuickStart Board, version 1.0, that operates with $3.3V$ and has 64Kbyte SRAM, 128Kbyte Flash and JTAG interface[9]. The robot is able to execute interpreted C control programs stored in its on-board Flash memory.

4.1.2.2. Sensors

The Surveyor SRV-1 is equipped with an on-board digital video camera and four infrared sensors for the perception of the environment. The camera, mounted at the robot's front, has a maximum resolution of $640px \times 480px$ providing JPEG images. The field of view was empirically determined to be approximately 40°. Infrared sensor units consisting of an emitter and a receiver are mounted at each side of the robot for proximity detection.

[9] Joint Test Action Group (JTAG) is the commonly used name for Standard Test Access Port and Boundary-Scan Architecture.

4.1.2.3. Communication Module

An XBee Pro wireless radio unit is used for the communication between the robot and the remote PC. The wireless interface is based on the ZigBee specification maintained by the same-named Alliance[10] and operates on an 868–868.8MHz unlicensed frequency band. ZigBee focuses on low energy consumption and low-speed ubiquitous communication between devices. It is based on the IEEE 802.15.4-2003 standard for wireless personal area networks (WPANs).

4.1.2.4. Control PC

Since the robot has limited on-board processing capacities while used for image-based tasks, it is controlled via a ZigBee wireless connection with an off-the-shelf laptop. The laptop is equipped with an Intel Pentium IV 1.8GHz processor and 1GB working memory.

4.1.3. Haptic Force-Feedback Device

By using a haptic interface, users are enabled to interact with and feel a variety of virtual objects with physical properties such as weight, shape, texture. The generation of such virtual objects that are just represented by forces is called *haptic rendering*.

Allmost all haptic interfaces have a tool—e.g. stylus—that can be freely moved by the user to touch virtual surfaces. When the tool is moved into a specific region of a virtual object, the haptic interface generates forces that simulate contact, friction, slip and so forth.

4.1.3.1. PHANTOM® Desktop™

The PHANTOM® Desktop™ device produced by SensAble technologies[11] is a precision positioning input and high-fidelity force feedback output interface for hand movements pivoting at the wrist (see Figure 4.7). It consists of a footprint and a freely movable molded-rubber stylus-gimbal. The device is constructed of injection-molded, carbon fibre reinforced plastics and metal components with a total weight of $3.6kg$. It has six DoF and provides 3D positioning in x, y, z space with an orientation α, β, γ. The force-feedback of this model is limited to the x, y, z position of the stylus.

Figure 4.7.: PHANTOM® Desktop™ haptic force-feedback device.

The range of motion lies within the radius of $60mm$ with a nominal positioning resolution of $1100dpi$. The exertable force ranges from $1.75N$ to $7.9N$ when arms are aligned orthogonally. According to the manufacturer's data, the inertia at the tip of the stylus is denoted with $< 75g$. The device is connected to a control unit, which is an arbitrary PC that provides the OpenHaptics® Toolkit and drivers via standard parallel port.

[10] `http://www.zigbee.org`, last accessed April 29, 2009.
[11] `http://www.sensable.com`, last accessed Mai 11, 2009.

Since the PHANTOM® Desktop™ device is just an interface, data analysis is driven by a WinXP workstation with dual Xeon 2.4GHz processor, 1GB RAM, a high-performance 3D graphics accelerator and high-resolution colour display.

4.2. Software

This section details the software architectures and outlines the control and interaction of the used hardware components.

4.2.1. TASER Control Architecture

The software architecture of the service robot TASER is based on the Roblet®-Technology[12] (Westhoff et al., 2006). With this technology it becomes possible to develop, compile and execute an application for the service robot on a workstation. The Roblet®-Technology permits the easy implementation of distributed systems which generally ease the development process for service robots. The goal of this technology is to encapsulate both high-level functionality and low-level access to robot hardware into a distributed software architecture. An advantage of using client programs running on remote PCs is that this provides more processing power for complex algorithms. This yields a load reduction of the robot's control PC.

The Roblet®-Technology is a client-server architecture where clients can send parts of themselves, referred to as Roblets, to an available server and spread them in the local area network. The server, referred to as Roblet-server, then executes the Roblets with well-defined behaviour in case of malfunctions. Notice that not only data is transmitted between the client and server but complete executable programs. This can be compared to Java Applets but with the difference that Roblets are not downloaded but sent. The network is transparent and developing distributed applications based on Roblet-Technology is like developing an application for a single workstation (Baier et al., 2006). Roblet-servers provide interface units and encapsulate low-level C/C++ hardware drivers of the robot via the Java native interface (JNI) while client applications provide interfaces to users at an arbitrary workstation.

Control relevant Roblet-servers provide interfaces to the PA10-6C robot arm, the BarrettHand and its force sensors, laser range scanners, pan-tilt-unit, IEEE 1394 cameras, the TCP-based control of the drive wheels and to other internal monitoring services like temperature, battery voltage, *et cetera*. An overview of TASER's software architecture is shown in Figure 4.8.

4.2.2. LIZARD Control Architecture

The software architecture of the LIZARD robot consists of different layers (see Figure 4.9). The first layer handles the IO's of the robot hardware and software. Inspired by *Brooks' Subsumption Architecture* (Brooks, 1986, 1991) it provides basic functionalities as collision avoidance based on image focus detection and motor control. The first layer is also responsible for the transfer of sensorial inputs to and converting commands from higher levels.

The systems input is 256-bit grayscale images with a resolution of either $80px \times 64px$ or $160px \times 128px$. Images are converted from JPEG to PGM in real-time. To preserve and

[12] `http://www.genrob.com/de/roblet_org.html`, last accessed Mai 9, 2009.

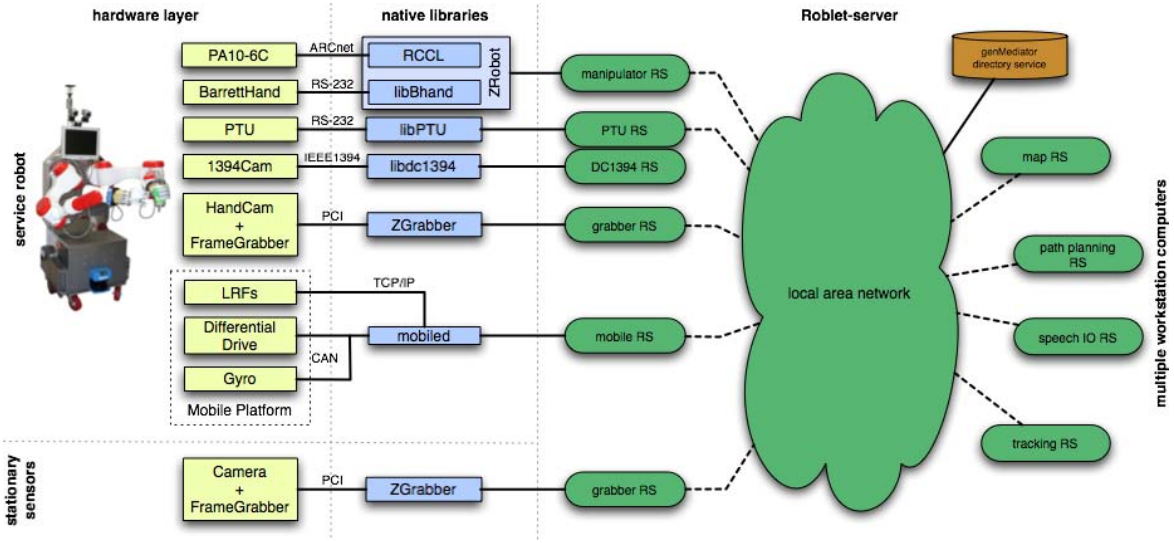


Figure 4.8.: The software architecture of the service robot TASER. Hardware components are shown in yellow boxes, corresponding C/C++ libraries have blue boxes and the Roblet servers are green (Baier et al., 2006).

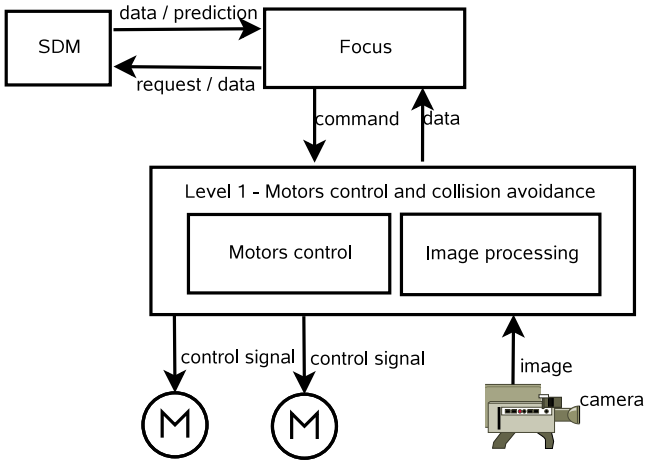


Figure 4.9.: The software architecture of the LIZARD robot according to Mendes et al. (2008).

improve image quality for further processing steps, the input image is enhanced by applying a histogram equalisation[13]. Three consecutive captured images are used to determine the noise levels for further processing steps.

According to Kanerva's terminology, the second layer is the system's focus. Established in this layer, a navigation algorithm sends control commands to the lower, first layer, where they are finally translated into concrete motor commands. Furthermore, this layer adjusts lateral drifts and determines if the robot was kidnapped from a learnt trajectory.

The third layer is the SDM where the robot stores and retrieves experiences through interaction with the focus. Mendes et al. (2007) studied different versions of the SDM for robot navigation. A comparison of the performance of SDM used for manipulation and navigation is detailed in Chapter 6.

4.2.3. Teleoperation Control Architecture

A teleoperation system was developed to obtain a higher flexibility in producing arm motion sequences for TASER (cf. Bruder (2009) for details). A client-server based architecture provides interactive control of the MHI PA10-6C robot arm via a PHANTOM® Desktop™ device.

The server runs on the robot's control PC. It is designed as a resource-friendly application that provides an interface to the low-level C/C++ hardware drivers of the robot arm. It manages contingency plans in the case of malfunctions, e.g. catastrophic network break down and operating errors. The tasks of the server are:

- waiting for and managing incoming client connections,
- monitoring the state $\vec{s}_t = (j_t, f_t)^T$ of the robot arm[14], with joint angles $j_t = (j_1, j_2, \ldots, j_m)_t^T$ and forces $f_t = (f_1, f_2, \ldots, f_n)_t^T$ at all time points t,
- notification of clients when a state change appears,
- monitoring and treatment of control commands $\vec{c}_t$ received by a client.

A client can run on an arbitrary workstation that provides the OpenHaptics library and is linked to a PHANTOM® Desktop™ device. It communicates with the server via LAN. A connection between client and server is established autonomously via TCP/IP request. To accomplish a quick intercommunication between client and server, an UDP/IP-based unicast `CommandReceiver` thread waits for incoming control commands $\vec{c}_t$ from a client. Moreover, a `StateStreamer` thread continuously sends the robot's state $\vec{s}_t$ via a UDP/IP multicast connection. For a more detailed structure of the transmission protocol the interested reader is referred to Chapter B. The tasks of a client are various:

- monitoring the desired robot arm position and orientation interactively supplied by the PHANTOM® Desktop™ operated by a user,

[13] A normalisation of the image contrasts has also been studied, but equalisation leads to better images for further processing steps with respect to the proposed system.

[14] In case of our robot TASER, $m = 10$ due to the robot arm with six joints and the BarrettHand with four joints, and $n = 3$. Due to the TorqueSwitch mechanism and single RSG each finger of the BarrettHand is considered as having one DoF in $\vec{s}_t$ instead of two as mentioned in Section 4.1.1.2.

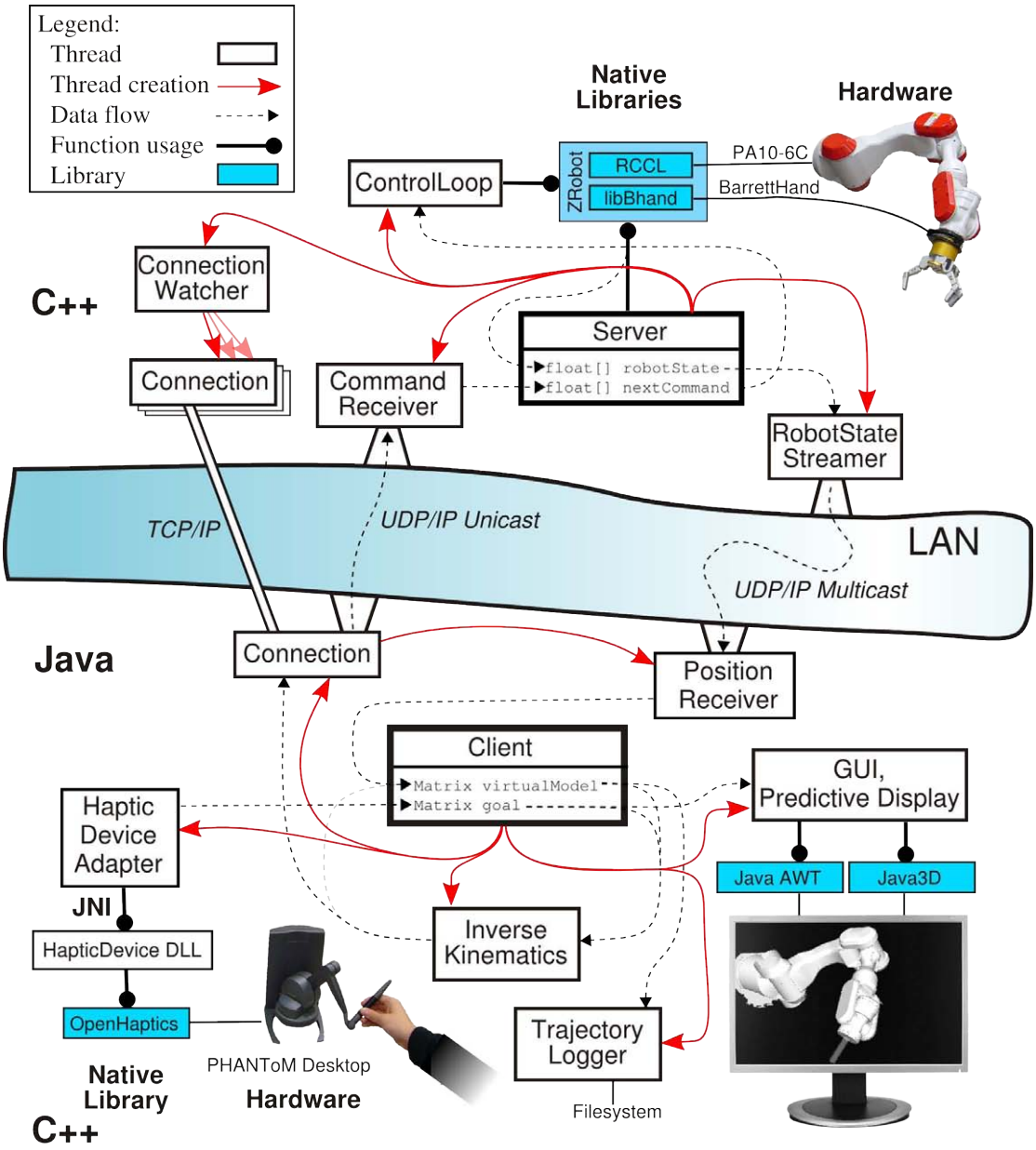


Figure 4.10.: Overall structure of the telemanipulation systems with the server application (top) and client application (bottom). Adopted by courtesy of Jan Bruder.

- transferring current joint configuration $\vec{f}_t^P$ of the PHANTOM® Desktop™ into a matrix representation[15] V_t at which the inverse kinematics is solved through a *gradient descent optimisation* algorithm (cf. Chapter B),
- sending command $\vec{c}_t$ to the server,
- monitoring network for incoming $\vec{s}_t$,
- transfer of incoming $\vec{s}_t$ into V_t,
- displaying a virtual robot model of current V_t to the user.

An inverse kinematic computation is necessary to map the different kinematic chains of the PHANTOM® Desktop™ and the PA10-6C. The considerable advantages of client-driven inverse kinematic computation are that the processing load of the robot's host PC is minimised and the simulation and training of manipulation tasks without any server connection becomes feasible. The benefits of using a gradient descent method for the inverse kinematics are that it computes meaningful solutions even for unreachable arm positions and singularities. A disadvantage is the slow convergence behaviour.

The overall structure of the telemanipulation system is shown in Figure 4.10. From a theoretical point of view, the system uses a two channel position–to–position architecture with joint space data transmission. The force feedback is based on a positioning error from Cartesian space coordinates. Differences between vectors $\vec{s}_t$ from the server and $\vec{s}_t$ from the PHANTOM® Desktop™ input are presented to the user as forces applied to the spatial position of the tip of the stylus. For more details, the reader is referred to Bruder (2009).

4.3. Concluding Remarks

In this Chapter, important hardware and software components have been introduced that are used during the studies on SDM-based robot learning. Due to the fact that each robot and its respective control mechanism is a complex system *per se*, the reader has been referred to more extensive literature on those particular systems.

The robot arm is used throughout the whole work and thus occurs in each Chapter. A comparison of SDM-based learning and prediction between other modalities and robotic domains is established with the unfailing aid of Mateus Mendes from ESTGOH, Portugal and his LIZARD robot system in Chapter 6. For a more flexible construction and comparison of diverse robot arm trajectories the telemanipulation system based on interactive user-control is used in Chapter 7.

[15] The matrix representation V_t describes the kinematic chain and stores the current constellation of the robot arm joints.

5

Implementation of and Proper Encoding for Sparse Distributed Memory

The everyday manipulation tasks we take for granted would stump the greatest robot bodies and brains in existence today.
(Kemp et al., 2007)

Contents

Albus (1993) argues that intelligence requires the abilities to sense the environment, make decisions, and control action. Higher levels of intelligence require the abilities to recognise objects and events, store and use knowledge about the world, and to reason about and plan

for the future. Albus (1981) also argues that intelligent behaviour of animals and robots in complex environments requires not just one associative memory but a large hierarchy of them, with the sensors and the actuators at the bottom of the hierarchy.

The SDM model by Kanerva (cf. Chapter 3) provides such an associative memory mechanism suitable for the bottom of the hierarchy that can be used for robots to learn actions related to certain circumstances in a flexible way. The SDM is used to store a world model that associates sensory input of a mobile robot system to certain actions. The flow of events in the world is presented to the memory as a sequence of large patterns that encodes sensor data, internal-state variables, and commands for the actuators. The memory's ability to store and to recall such sequences under conditions that resemble the past makes its use for prediction and planning possible.

The ability to learn new associations, concepts, actions, *et cetera* without protracted trials of learning is an important characteristic for grounded (multimodal) memory in robots. In several cases it is desired getting the best prediction after a single presentation of a sequence instead of a perfect learning after many trials. One-shot learning is provided by the SDM. Being roboust against noisy input data constitutes another crucial characteristic for a robot memory. A summary of design requirements for memory-based cognitive robots is shown in Section 1.2.

The goal of equipping a robot with a memory is to preserve subjective experiences on how to solve tasks based on the current sensed circumstance. Compared to embodied connectionist approaches, an SDM is used to process raw sensory data. The memory provides a robot with basic mechanisms for the detection of perceptual familiarity while acting in a dynamic and real office environment. A clear separation of a learning and testing phase as done by conventional classification and prediction models makes no sense and thus disqualifies most of them for an autonomous mental developing system.

A memory optimisation mechanism in human beings to ensure that just relevant experiences are recalled is to forget irrelevant and unpleasant ones. Forgetting is a major subject in cognitive science. A technical system equipped with memory will also gain from forgetting to achieve a kind of memory management. A robot that e.g. learns a multitude of paths to navigate through an environment may not benefit from keeping a memory trace of all less interesting tracks that only have been used once a long time ago.

This chapter includes the implementation of an SDM for robotic applications, especially for storing and retrieving mobile manipulation actions of a MHI PA10-6C robot arm (see Section 4.1.1.2). Sequences of events are stored as subjective experience and are later used to guide robot arm behaviour based on its memory content. Several simple manipulation tasks, such as lifting and placing a wastebin from and on the floor, pushing an object aside on a table-top, and drawing shapes in the air are analysed under different operational modes. Previous work mainly focuses on SDM-based robot navigation, as studied by Rao and Fuentes (1998); Mendes et al. (2007, 2008).

The main purpose of this chapter is to illustrate the implementation, to study different modes for an appropriate encoding of sensory information and to show reconstructive abilities of task-dependent arm trajectories based on an SDM.

5.1. Implementation

Since the implementation should act as an autobiographical memory of a robot, an *a priori* fixed memory size assignment seems inappropriate for a life-long learning system. Thus the dynamic reallocation algorithm is used to provide a flexible growing memory while keeping the possibility of changing the memory structure without retraining (Ratitch and Precup, 2004). Life-long learning means that a process of learning that once started never ends.

Kanerva's implementation only relies on Hamming distances. Due to some problems of the non-proportionality of such distance metric when used with natural binary code, some investigations on further encoding modes are made and outlined in this section. Parts of this work are also published in Jockel et al. (2008a).

5.1.1. Dynamic Memory Allocation

Although theoretically sound and attractive, the SDM model has some weaknesses that yield practical problems. One of them is the question of where to place the hard locations in the address space. Kanerva proposed a fixed and random distribution during memory creation. Several authors propose better solutions to keep the memory flexible. By using genetic algorithms, Rogers (1990) determines the most suitable hard locations. Hely et al. (1997) propose that the hard locations have to be created where a lot of data should be stored. Hard locations that make up the memory are not known at memory startup. An idea that is essentially based on the same idea is proposed by Ratitch and Precup (2004), which is used in this work. The difference between both latter approaches is as follows. While Hely et al. (1997) starts with a randomised memory and dynamically reorganises hard locations according to the data to be stored, Ratitch and Precup (2004) start with an empty memory.

The randomised reallocation algorithm (cf. Section 3.5.3) is used for a dynamic online allocation and adjustment of memory resources for the SDM. This yields a higher flexibility to the robot's sensory data and an advance determination of memory size becomes obsolete. Memory locations are added based on the observed data. New hard locations are allocated randomly in the neighbourhood of an input address if the data cannot be stored into enough existing hard locations anymore.

5.1.2. Data Acquisition

Mobile robot arm manipulation is based on a temporal sequence of spatial coordinates and respective orientation of the manipulation tool given by a programmer. The MHI PA10-6C portable robot arm by Mitsubishi Heavy Industries mounted on the service robot TASER is used for this purpose. The arm has six DoF, an operational range of $1317mm$ and is controlled by the Robot-Control-C-Library (RCCL) via ArcNet interface. Further details about the robot arm and the used hardware are mentioned in Section 4.1.1. The spatial coordinates are the 3D position x, y, z of the tool centre point (TCP) and its orientation expressed by the roll χ, pitch ψ, and yaw ω angles.

The robot arm motion sequences used in this chapter are generated by specifying a limited number of TCP positions[1] within a basic control program. The inverse kinematics to route

[1] A TCP position in the context of manipulation actions is hereinafter referred to as a 3D spatial position x, y, z and orientation χ, ψ, ω.

the MHI PA10-6C robot arm from one TCP position to another is computed by the underlying Robot-Control-C-Library (cf. Section 4.1.1.2, RCCL), and not by the programmer himself. During a learning execution, the 6 joint angles[2] of the robot arm, the TCP, the tool orientation and some additional information (see Equation 5.1 & 5.2) are stored to the SDM with a sampling rate of 6-10Hz. Different parameters like joint angles and TCP position express the same, but are stored for redundancy[3]. The additional information consists of a unique sequence id *seq_id* for each motion trajectory and a unique index i for each vector within a sequence. The storage and recall is described in some more detail in the following section. The input pattern x of the SDM system is as follows:

$$x =< J, P, O, E > \tag{5.1}$$

J is the joint constellation of the 6 joints of the robot arm with $j_n,\ n \in \{1, 2, \ldots, 6\}$.

P is the 3D position of the tool mounted at the end of the effector (*tool centre point, TCP*) depicted with x, y, z.

O is the the tool orientation depicted by χ, ψ, ω describing the roll, pitch, yaw angle.

E is the additional parameter set consisting of a *seq_id*, a unique 4-byte *id* for each sequence, and i is unique for each vector.

The length of the input vector depends on the representational form described later in this chapter. Two of three investigated operational modes represent the above-mentioned parameters as 8-byte double value, except the *seq_id* and index i which are represented by 4-byte each. Concatenating all those parameters into one long binary input vector leads to a total vector length of 832 bits. The resulting vector is as follows:

$$x_i =< j_1, j_2, j_3, j_4, j_5, j_6, x, y, z, \chi, \psi, \omega, seq\_id, i > \tag{5.2}$$

5.1.3. Memory Storage and Retrieval

The locations around a certain input address x have to be determined to accomplish information distribution for any reading or writing process. Let N' be the set of actual memory hard locations, x the reference address, n the number of bits in the address and r be the selected radius. Then the set X'_A of selected locations is given by $X'_A = \{x' | x' \in N' \wedge d(x', x) \leq r\}$, where $d(x', x)$ is the distance between reference address and the hard locations $x' \in N'$.

When storing a pattern to the memory, each counter c_i for all selected locations is incremented or decremented according to the bit-value x_i being 1 or 0 respectively. Consequently, each write operation modifies the abstraction of the stored pattern. With an SDM the problem of learning tasks is transformed to storing and retrieving encoded information.

A prediction of an SDM as described by Algorithm 1 is the average of the information stored within the set of active hard locations. Each bit position of the content vector of all

[2] Each joint has a software-limited operation range. Please confer Table 4.1 of Section 4.1.1.2.

[3] The joint constellation and the TCP position are not symmetrically related in a redundant manipulator. A certain joint constellation always implies a certain TCP position. Whereas a certain TCP position might be described by several, varying joint constellations.

Algorithm 1 The prediction of an SDM

Require: *Active hard locations* $X'_A \leftarrow$ hard locations x' within the activation radius r

```
result ← ()
for i = 0 to length of content vector do
    for j = 0 to |X'_A| do
        result[i]+=X'_A[j][i]/|X'_A|
    end for
end for
return result
```

active locations is summed up and divided by the total number of active locations. Thus, a prediction is an average of sub-predictions.

5.1.4. Architecture

The main relations and functionalities of the implemented SDM are illustrated in Figure 5.1. The autonomous service robot TASER provides a distributed software architecture (cf. Section 4.2.1) that facilitates a simple integration of new software modules. For performance reasons, the SDM is running on a workstation and is integrated into the robot's framework via Roblet®-Technology. Thus, the workload of the robot's control PC is reduced substantially.

The system offers two ways to accomplish initial training of the memory. First, the robot's manipulator is controlled by a program and logs all the parameters of the manipulation sequence. The SDM is trained on a workstation by interpreting the log files gathered from the robot. Secondly, the SDM can also be fed with the relevant data directly while being remotely connected.

After a training phase, when testing the memory, the robot is triggered with an initial arm configuration. The sensor data of the manipulator is captured by the robot's control PC and is transmitted to the workstation that hosts the SDM. The SDM makes its prediction of the next state and sends the predicted arm configuration to the robot controller. Hereinafter, the predicted state is used to predict the next subsequent state.

5.1.5. Complexity

The complexity of our system is as follows: Let N' be the set of all hard locations that represent the address space N. For any write or read operation it has to be determined which memory locations lie within an activation radius r around an arbitrary input address pattern x. Thus, each hard location has to be considered for the Hamming distance $d(x'_i, x) \leq r, x'_i \in \{1, \ldots, |N'|\}$. This yields a linear complexity $O(|N'|)$. The computational time increases linearly with the number of memory locations. Due to a constant vector dimension n, the computation of the Hamming distance itself remains constant.

5.2. Experiments

Initial experiments are carried out to prove an SDM's (cf. Chapter 3) ability to predict manipulation sequences of a robot arm, the MHI PA10-6C in particular.

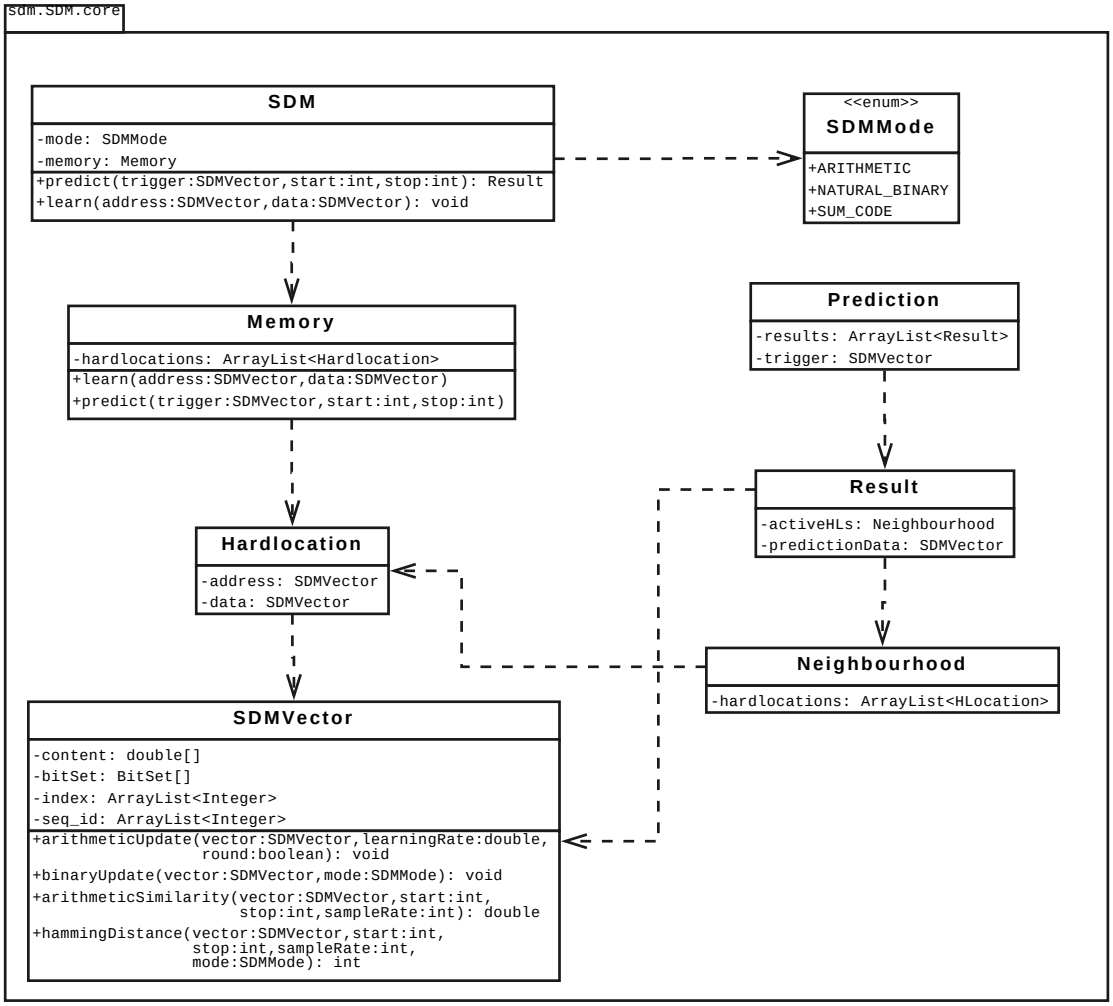


Figure 5.1.: A UML diagram of the SDM architecture that illustrates the main relations and functionalities.

Within the initial tests the memory is used to learn an action sequence carried out with the robot arm. Three kinds of action sequences are investigated primarily for this series of tests. The studied sequences differ in their length and range from 125 up to 277 discrete arm configurations. The high-level transcriptions of the manipulation action sequences are:

- Place the robot hand closely above a table in front of the robot, and push a certain object aside by a lateral cartesian motion on a transversal plane.
- Lift and place a wastebin from and on the floor.
- Draw a U-like shape in the air with the robot's hand.

After learning, during an autonomous run the arm is placed at an arbitrary initial position within the sequence. The sensor readings at that position are transferred into the vector x (cf. Equation 5.2) and are presented to the memory as the reference address. Note, that the memory can be triggered with partial cues and thus can compare parts of an input vector separately, e.g. the joint constellation. The remaining values of the vector are neglected. The memory predicts the best match for the next associated position by its experience according to Algorithm 1. The control program, then, guides the robot arm to the respective predicted position. If the new position is reached, again the sensor readings of that position are presented to the memory such that it predicts the next position over and over again. Since the data was captured at a rate of 6-10Hz during learning, each predicted movement is quite small. Figures 5.3, 5.2 and 5.4 show a) freeze images of the robot during the execution of several tasks and the corresponding b) x, y, z tool trajectory in a 3D plot with separate 2D plots for c) the joint constellation and d) the TCP spatial position and orientation over time.

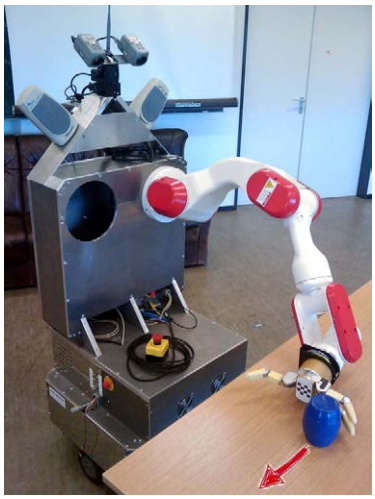
5.2.1. Comparing Different Encoding

Kanerva used plain binary code. In such code the represented value depends on the positions of each bit, e.g. the binary pattern $p_1 = 0100$ represents the value 4 while the pattern $p_2 = 1000$ represents the value 8. The Hamming distance of $d(p_1, p_2) = 2$, while the represented value doubles. If compared to a third pattern $p_3 = 0001$ the Hamming distance $d(p1, p_3) = 2$, while p_3 represents just a fourth of the value of p_1. Thus, if the Hamming distance of two patterns is of equal length, the number of bits in which both patterns differ is not proportional to the represented values. The Hamming distance sometimes even decreases when the arithmetic distance increases, e.g. the binary patterns $p_4 = 0100$ and $p_5 = 0011$.

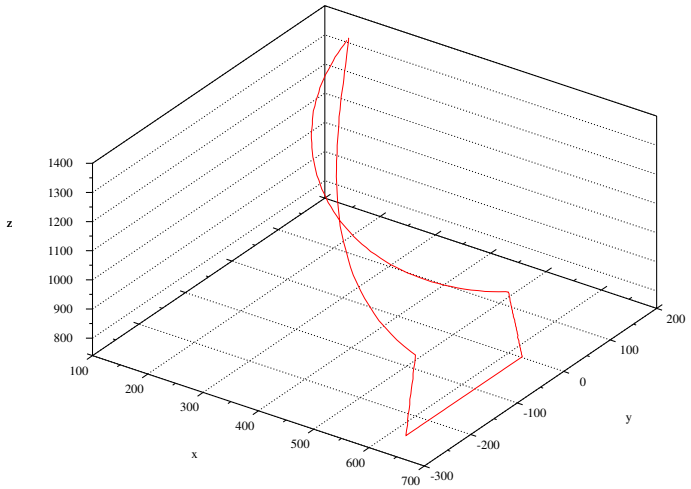
Mendes et al. (2009) propose some initial ideas to improve the encoding problem of binary memory features. They studied the resorting of bits representing a gray value within an interval $[0, 255]$. They mainly tested tens of thousand randomly chosen permutations of the higher-order bits $[244, 255]$ to reduce undesirable transitions. In the following paragraphs alternative encoding modes for the SDM are proposed to deal with robotic domains.

5.2.1.1. Natural Binary Mode

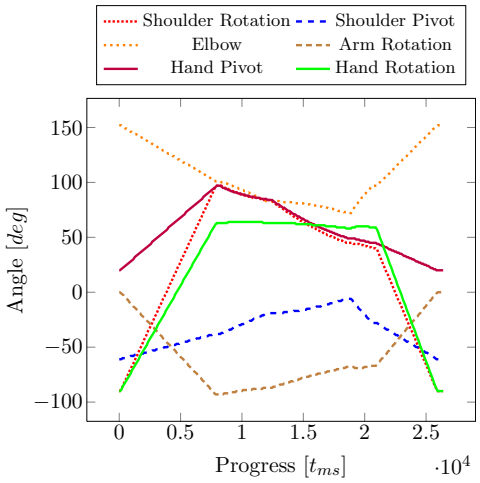
The bitwise implementation by Kanerva works with counters, which is one of the model's weaknesses. That is also the reason why his implementation requires a lot of unnecessary processing time. A counter decreases the storage capacity per bit of a traditional computer



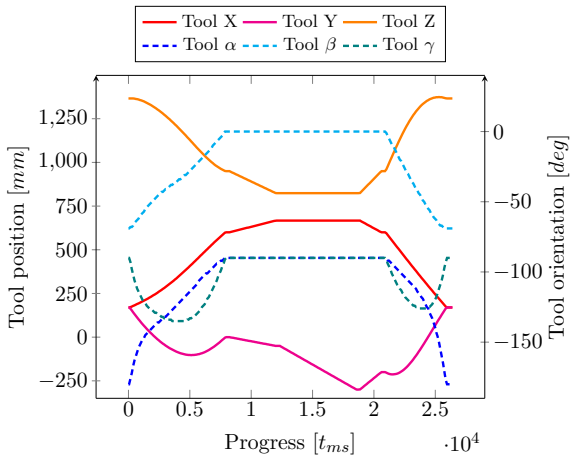
(a) Pushing an object aside.



(b) Trajectory of the x, y, z spatial tool position.



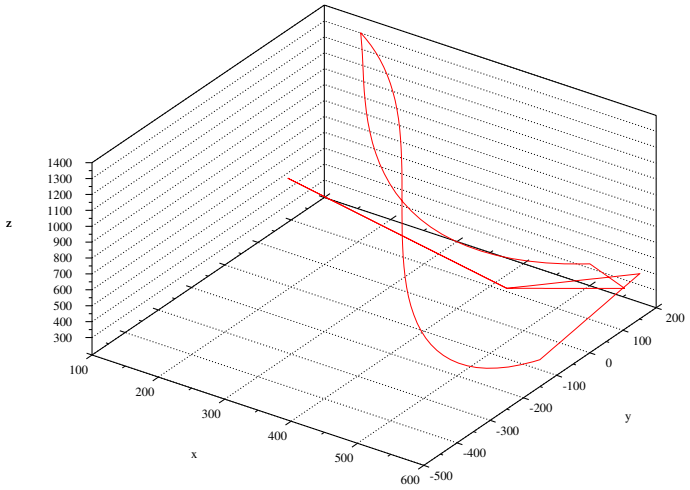
(c) Joint values.



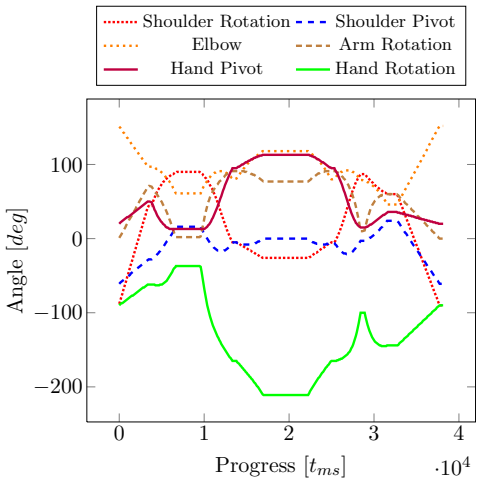
(d) Cartesian TCP coordinates and tool orientation.

Figure 5.2.: Images of the test sequence of TASER pushing an object aside by a lateral cartesian motion on a transversal plane.

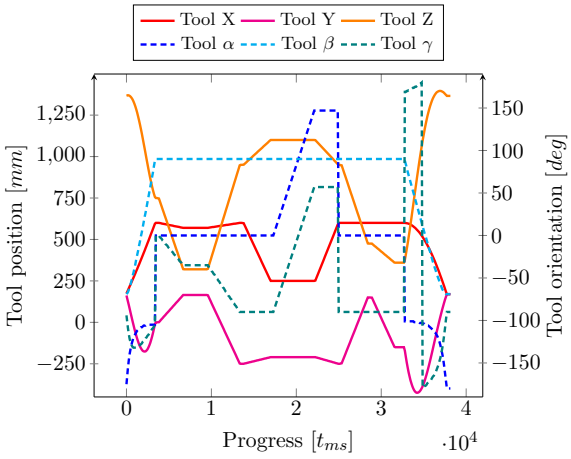
(a) Lifting a bucket from the floor.



(b) Trajectory of the x, y, z spatial tool position.



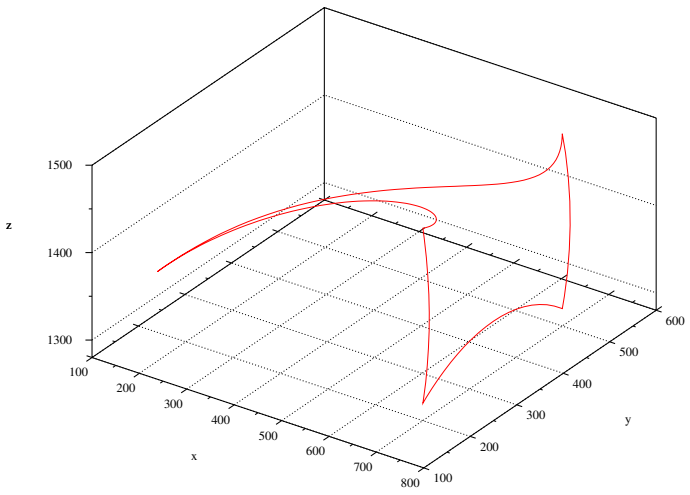
(c) Joint values.



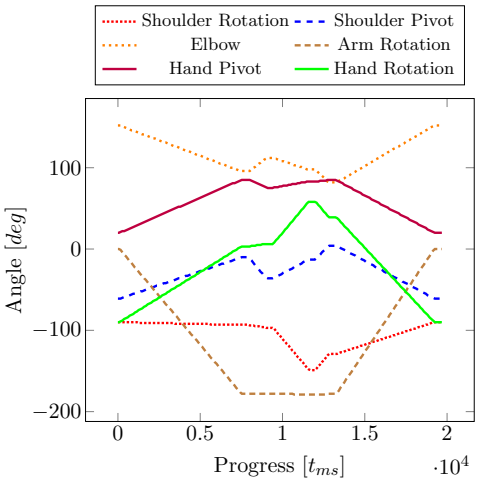
(d) Cartesian TCP coordinates and tool orientation.

Figure 5.3.: Images of the test sequence of TASER lifting and placing of a wastebin from and on the floor.

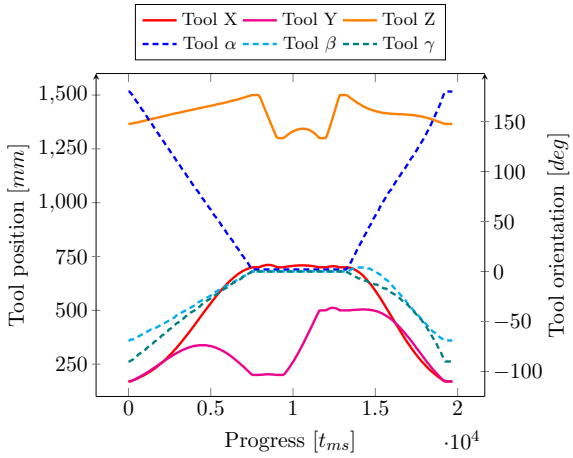
(a) Drawing an U-like shape in the air.



(b) Trajectory of the x, y, z spatial tool position.



(c) Joint values.



(d) Cartesian TCP coordinates and tool orientation.

Figure 5.4.: Images of the test sequence of TASER drawing an U-like shape in the air.

to 0.1 bits (Kanerva, 1988). To reduce the memory size[4] and to increase the processing power the bit counters of Kanerva's model are replaced by a single bit to store one bit of information according to the proposal by Furber et al. (2007). The contents of the memory is simply overwritten based on an OR function if a new data word is stored into the same memory location. The structure remains the same as in Kanerva's model. The first layer depicts the address decoder which decodes the input addresses into high-dimensional space. The second layer is the memory where associations are learnt and stored. Furber et al. (2004) proved that the memory performance is not significantly affected in doing so. Accordingly, the SDM's content matrix C (cf. Figure 3.4 and 3.5) comprises 1s and 0s instead of the counters. Writing any information to the memory is done by replacing an old datum by the new one. Now that the values of C are within $[0, 1]$, 0.5 will suit as threshold value. This kind of implementation depicts the *bitwise operation mode* (Mendes et al., 2008; Jockel et al., 2009).

5.2.1.2. Arithmetic Mode

A modified form of SDM proposed by Ratitch and Precup (2004) (cf. Section 3.5.3) is used to implement an *arithmetic operation mode*. Values are represented in $\mathbb{R}^n$ for a higher flexibility concerning the precision of stored information, e.g. joint values. The pattern is grouped into several chunks of 64 bit double precision patterns. Learning is achieved by using reinforcement learning and addressing is achieved by using an Euclidean distance instead of the non-proportional Hamming distance. The vector values are updated by applying the following equation:

$$\Delta x'^k = \alpha(x^k - x'^k), \quad \text{with } \alpha \in \mathbb{R} \wedge 0 \leq \alpha \leq 1, \tag{5.3}$$

where x'^k denotes the k^{th} 64 bits of the hard location, x^k is the corresponding value in the respective block of the input vector x and the learning rate is denoted with α. The learning rate specifies how much current information influences the already stored information within a hard location.

5.2.1.3. Sum Code Mode

Due to the above-mentioned problems with the non-proportional Hamming distance, the sum code is tested. A number u within an interval $[a, b] = \{u \in \mathbb{Z} | a \leq u \leq b\}$ is defined by the number of 1-bits according to Equation 5.4. This representation is known as *sum code* (Jockel et al., 2008a, 2009), a derivate of the *1-of-N code*[5], or thermometer code. The boundaries of the intervals are defined by the maximum and minimum values for the particular operating range of each parameter (joints, x, y, z and χ, ψ, ω) of the pattern vector.

$$sc(u) = u - a \tag{5.4}$$

Table 5.1 shows the difference between the non-proportional binary and proportional sum code representation. Strictly speaking, the sum code representation is another form of the

[4] Although the memory capacity of current available hard disks allow to store huge amounts of data, a reduction of memory size was required in this project. The TASER robot is equipped with a single standard PC and it has to share its lean memory with several other applications that require a lot of memory space, e.g. to store image data from various cameras.

[5] 1-of-N codes are a special case of constant weight codes that encode $\log 2N$ bits in a code-word of N bits.

Table 5.1.: An example of different representational types.

Decimal code	Binary code	Sum code
0	0000	00000000
1	0001	00000001
2	0010	00000011
3	0011	00000111
4	0100	00001111
5	0101	00011111
6	0110	00111111
7	0111	01111111
8	1000	11111111

arithmetic representation, the Hamming distance in such code corresponds to the Euclidean distance. Thus, the sum code mode is a supersized version of the arithmetic mode such that the representation is blown up. However, the mode is chosen to test the advantageous SDM characteristic for high-dimensional binary vectors. With the sum code representation we stay in the binary address space, flipping of bits is still possible while the problematic distance measure is amended.

Coding theory provides many other interesting codes that have not been considered in the scope of this work. Gray code, for instance, also offers better distance properties than natural binary code. The above mentioned sum code provides features similar to Gray code: Successive values differ in only one bit.

5.2.2. Dealing with Memory Damage

Humans have a the remarkable capability that stored information within their memory is retained even when individual neurons die, e.g. caused by heavy intoxication. Beside other reasons, this may be caused by an information distribution within the underlying neural network. Tests have been made to show that an SDM is suitable to retain information even when individual locations are erased.

The distributive nature of SDM has the side effect that many locations participate in information storage and retrieval. The output of the system represents an average of the information of the participating memory locations. The remaining prediction accuracy when memory locations are erased is studied to highlight the differences between an SDM and a generalised RAM according to Section 3.3.1. To simulate memory loss a certain percentage of hard locations are randomly erased after a training stage. Note that the same trained memory is used for each test. The learnt trajectory corresponds to the trajectory shown in Figure 5.4. After training, the memory is applied to predict residual arm trajectories by triggering it with the first sequence element. The elapsed time until the first prediction error occurs is given as a percentage of the original trajectory. A prediction error is defined as a non-intended prognostication, e.g. a prediction of a previous or identical state with respect to the state the robot currently resides in. Table 5.2 and Figure 5.5 show the mean of ten runs for predicting a trajectory with gradual memory injury. Figure 5.5 shows very plainly that the SDM is able to cope superiorly with memory damage than a generalised RAM. While it shows the elapsed time until the first occurrence of a prediction error, it does not

Table 5.2.: Gradual decrease of associative capability after suffering a loss of memory locations. This table indicates when the first errors occurred during sequence reproduction.

Randomly erased	Elapsed time till first prediction error		
hard locations	SDM $r = 0$	SDM $r = 2$	RAM
0%	100.00%	100.00%	100.00%
1%	79.11%	94.07%	53.89%
2%	39.69%	68.70%	25.68%
3%	30.12%	52.84%	22.84%
4%	21.91%	57.59%	17.47%
5%	27.72%	57.47%	7.41%
6%	20.25%	47.47%	8.52%
7%	16.05%	38.52%	4.63%
8%	40.62%	28.64%	5.25%
9%	15.43%	40.37%	7.72%
10%	36.85%	37.04%	5.25%
15%	46.48%	20.25%	3.09%
20%	12.72%	18.95%	0.80%
25%	70.49%	8.77%	1.23%
30%	42.10%	33.70%	1.11%
40%	30.86%	12.72%	0.37%
50%	50.99%	30.74%	0.62%
60%	11.98%	21.36%	0.37%
70%	31.42%	21.17%	0.25%
80%	21.48%	30.68%	0.12%
90%	3.77%	21.79%	0.12%
99%	38.52%	53.21%	0.00%

give any information about final success to reach the goal of the trajectory. It merely shows at which stage the first error occurred, while the sequence may be continued till the end from a sequence element $x_{t-k}, k \leq 0$ predicted at time point t. Also, one and the same sequence element may be predicted twice (or more frequently) but the memory may escape the stagnation. The system does not necessarily get to be stuck when a prediction error occurs as later sections will show. Accordingly, a correct prediction will predict a sequence element x_{t-k}, with $k > 0$.

Contrary to a generalised random-access memory (RAM), an SDM with activation radius $r = 0$ uses the closest location when confronted with an input address that is physically not existent within the memory. This constitutes one main advantage of an SDM, it behaves like an attractor network. Partly, the prediction accuracy can be improved when data is distributed into more than a single memory hard location. In case of $r = 0$, the SDM used as much hard locations as world states to be represented, 162. In case of $r = 2$, around 250 hard locations have been used by the memory. It can be undoubtedly seen that an SDM outperforms a generalised RAM. Nevertheless, increasing the activation radius and thus the number of locations that participate in information storage, prediction errors may

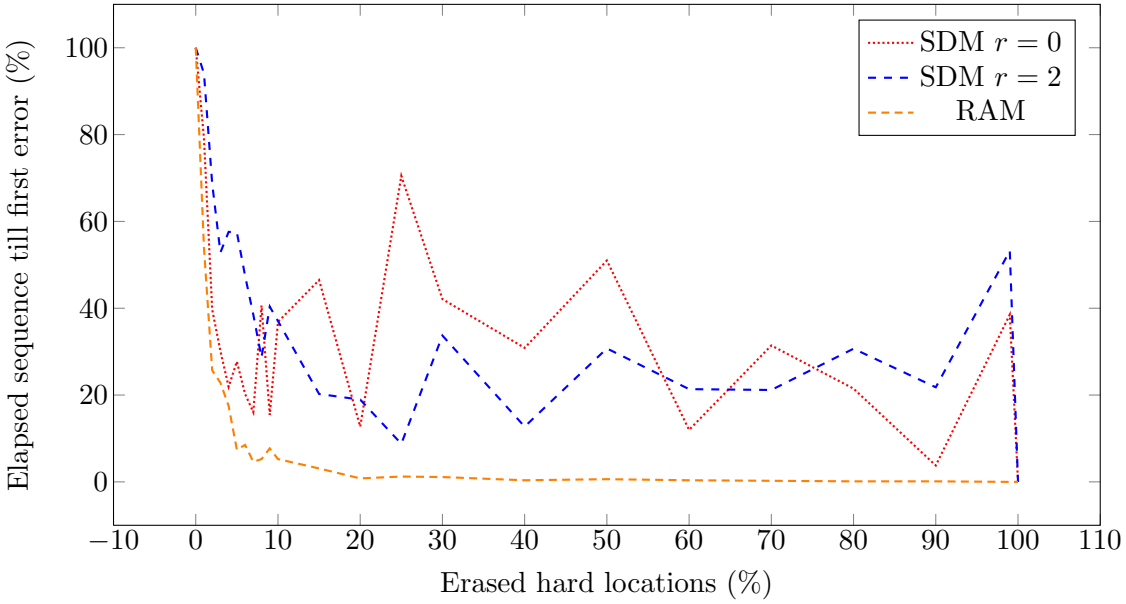


Figure 5.5.: Decrease of associativity after suffering a loss of memory locations according to Table 5.2. Line ····· shows an SDM with activation radius $r = 0$ that choose the closest hard location if an input address does not physically exist. Line - - - results from an SDM that uses distribution ($r = 2$). Line - - - - corresponds to the behaviour produced by a conventional random-access memory (RAM).

occur without any memory injury. This is due to the averaging nature of the SDM when a memory location participates in storing more than one content item.

Equipping a robot with a memory that retains its predictive capability when it is injured would be a great advancement to bear intelligent agents with a long-term memory. Further, the memory is able to produce an output concerning an arbitrary input cue that is not precisely known with respect to the particular feature configuration. Further studies are necessary to analyse the SDM behaviour regarding failures in a more technical context, e.g. a malfunction of entire memory blocks or banks. However, the results show that an SDM can be used as a memory that is able to compensate memory damage up to a reasonable degree. Additional results on the robustness of the memory against noise is presented in the context of learning multiple heterogeneous tasks. Thus, the reader is also referred to the latter Section 7.5.7.

5.2.3. Overcoming Problems with Cross-Sequences

Perceptual aliasing refers to the situation where two or more identical perceptual inputs require different responses from an autonomous system. The effects of aliasing can be reduced to some extent by incorporating additional sensory information that suffices to distinguish between any two given situations (Whitehead and Ballard, 1991).

Initial experiments revealed predictability problems of consecutive states within learnt cross-sequences. Accomplished tests comprise sequences where circular or recurring movements are involved, e.g. to beckon to a person, sequences with a start position similar to a sequence end position, drawing an “8” *et cetera*. Other experiments just contained a short

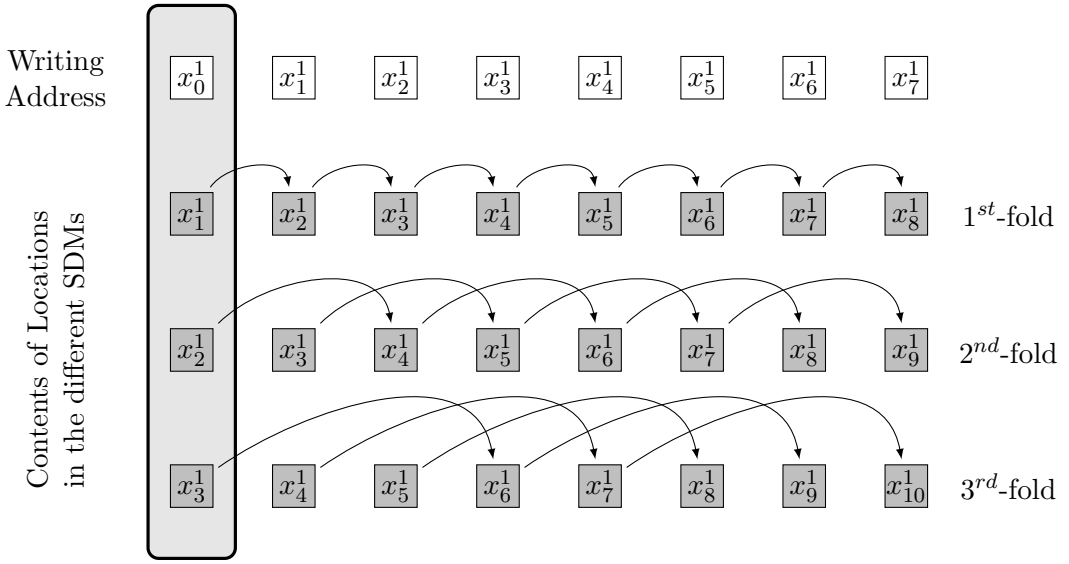


Figure 5.6.: Schematic diagram of a k-folded memory with three folds. Adapted from Kanerva (1988).

sequence of nearly identical arm configurations for a small part of two trajectories.

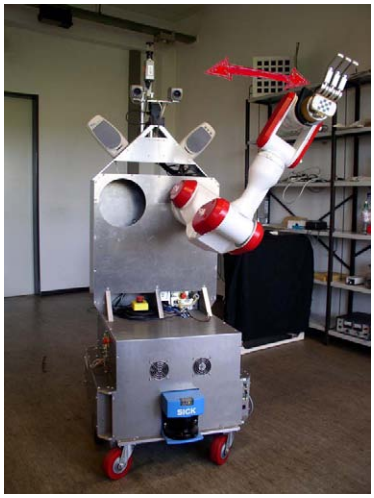
Kanerva (1988) proposes to use a k-folded memory such that an input consists of the last k sensory inputs to provide the necessary context for disambiguation of potentially similar perceptions. The implemented SDM model for the robot domain was extended to use such k-folded memories. Figure 5.6 explains the folding concept.

Instead of using a single SDM, a composite of several SDMs will be utilised in accordance with the desired number of folds. Each SDM gets the same address matrix, which means that all SDMs describe the same set $N' \in N$ of hard locations. The SDMs differ in their content matrix. While a single-folded SDM stores a pointer to the next sequence element, a twice-folded SDM stores a pointer to the second element of a sequence and a triple-folded memory to the third element of a sequence and so forth. The prediction of an address, e.g. x_t results from k simple predictions $x_{t-1}, x_{t-2}, x_{t-3}$ (in case of a three-folded memory).

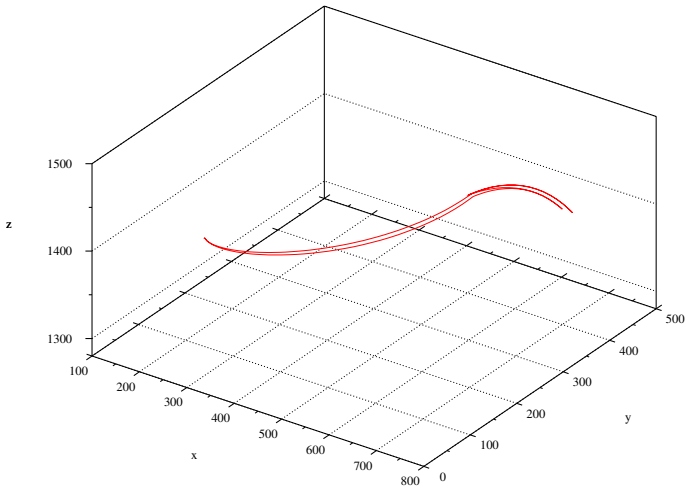
An example shall clarify this issue. Assume that a sequence $A \rightarrow B \rightarrow C$ is usually followed by D and a sequence $E \rightarrow B \rightarrow C$ by F. Both sequences are equally probable. To predict the consequences of C it is necessary to know what happens two steps before C. A third-order prediction is required to reliably choose between D and F.

An experiment in the scope of this work where standard first-order memory fails is a beckoning sequence where the robot arm is shaken to the left and the right several times. Each sampled arm configuration occurring while moving the arm to the left is strikingly similar to those occurring while moving it back to the right. While testing this sequence with a first-order memory, the robot arm occasionally changes its moving direction in between. Three and five folded memories proved to be sufficient to overcome such simple prediction errors and to let the arm autonomously move to the leftmost and rightmost position consecutively.

A crucial issue: the folding concept only leads to reliable predictions as long as the number of folds exceeds the number of identical states of various sequences. Whereas the three- and five-folded memory is applicable to solve short sequence overlays it is not suitable for a reliable prediction on when to exit the shaking motion, as in our experiment after three



(a) Beckoning.



(b) Trajectory of the x, y, z spatial tool position.

Figure 5.7.: TASER beckons by waving the hand and the bent forearm several times from the right to the left and back again. The trajectory contains lots of similar arm configurations over a longer period of time what makes reliably prediction difficult.

iterations. In the beckoning experiment approximately 145 states overlap or are strikingly similar from the entrance point until the exit point of the recurring shaking motion. At least a 146-folded SDM must be used to predict the whole sequence trustworthily. Higher-level concepts are required to solve such problems due to the fact that an increase in the number of folds yields a growth in the required memory space. Some ideas to include higher-level concepts will be mentioned in Section 7.4 and 9.2.1. Nevertheless, SDM provides useful characteristics making it suitable for the prediction and familiarity detection of manipulation action sequences based on experience as subsequent chapters will show.

Although the TCP spatial position in the graphs of Figure 5.7(b) show several intersections the system achieves mostly correct predictions. The remaining parameters of the vector, so to speak the arm configuration, differ enough even for similar spatial positions of the robot's tool. Nevertheless, three- to five-folded memories constitute an improvement of the predictive value of basic action sequences, whereas the above-mentioned problems still occur.

5.3. Results

Various test are made to gauge the performance of an SDM while focusing on different encoding modes. The information stored to the memory is distributed among active hard locations with regard to the activation radius. Experiments with three different representational modes have been implemented. The averaged results of 30 runs of sequence predictions with respect to three different tasks are summarised in Table 5.3. The columns of the table describe:

Task: Three different tasks are learnt by the memory: A) placing the robot hand closely above a table in front of the robot, and pushing a certain object aside by a lateral

Cartesian motion on a transversal plane, B) lifting and placing a wastebin from and on the floor, and C) drawing a U-like shape in the air.

Operation mode: Three different modes are studied:

Arithmetic mode: Values are represented in $\mathbb{R}^n$. The Euclidian distance is used to define a metric in order to calculate distances between hard locations. A more detailed description of this mode can be found in Section 5.2.1.2. The dimension n of a memory vector is 832 bits.

Bitwise mode: The data is represented in a natural binary manner as described in Section 5.2.1.1. The Hamming distance is utilised as a distance metric for this mode. The dimension n of a memory vector again is 832 bits.

Sum code mode: A binary representation of the arithmetic mode as described in section 5.2.1.3. Compared to the arithmetic mode, the vector length is massively enlarged to 22320 bits while retaining the nice characteristic of boolean space as the possibility of shifting bits which disappears when using normal arithmetic mode.

Number of hard locations: The number $|N'|$ of hard locations that have been created and used by the memory with respect to the particular operation mode and access radius.

Distance to closest hard location: Describes the distance to the closest hard location $x'_i \in N'$ to a given n-bit input address x.

Distance to second closest hard location: Describes the distance to the second closest hard location $x'_i \in N'$ to a given n-bit input address x.

Average distance to all hard locations: The average distance is defined as follows:

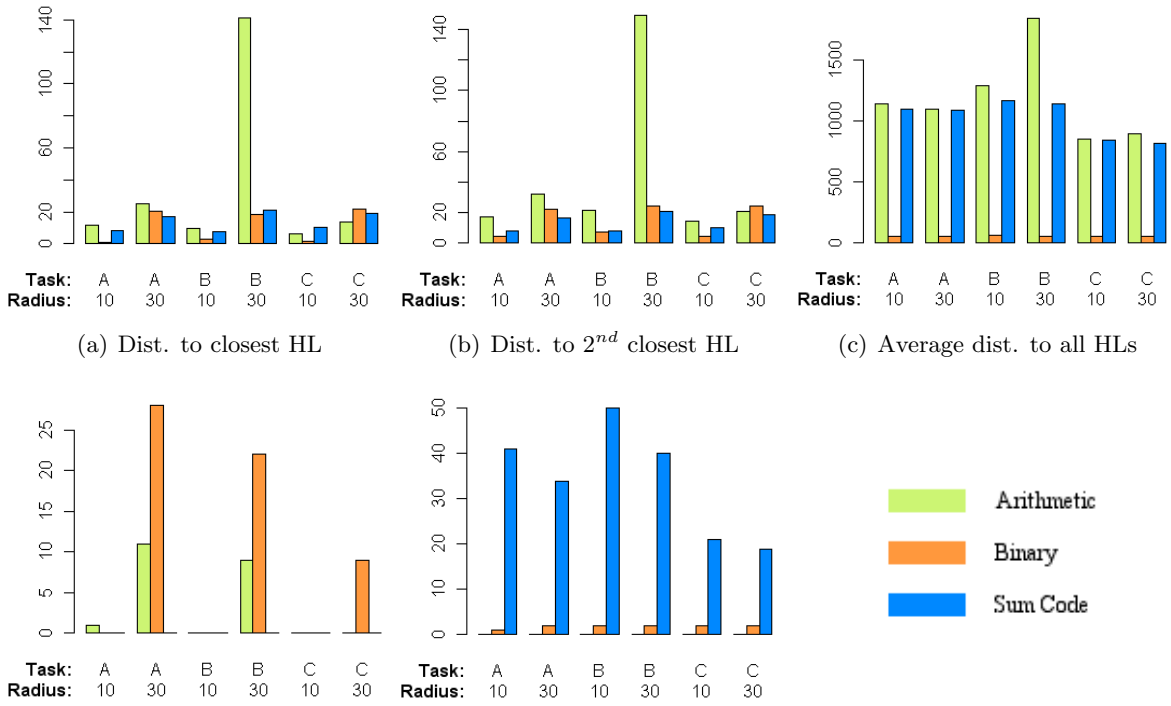
$$d_{\varnothing}(x) = \frac{1}{|N'|} \sum_{j=0}^{|N'|} d(x, x'_j) \tag{5.5}$$

The larger the difference between the distances of the first and second closest location to the average distance of all locations, the lower the confusion caused by other trajectories stored in memory.

Increment: These percentages result from the above-mentioned distances between the closest and the second closest hard location and from the closest hard location to the average distance of all hard locations respectively.

Errors: A prediction error is defined as an unintended prognostication, e.g. a prediction of a previous or identical state with respect to the state the robot currently resides in.

Processing Time: Average number of milliseconds needed to predict the next state within a sequence over all runs. This time measure just takes into account the time needed for the prediction. Learning phase, previous polling of sensor data as base for triggering the memory, the transfer of sensor data from and control command to the robot as well as the command execution are not considered within this measure. The transfer rate of the wireless network varies whenever several users access the robot and when



(a) Dist. to closest HL (b) Dist. to 2^{nd} closest HL (c) Average dist. to all HLs

(d) Prediction errors (e) Processing time (ms)

Figure 5.8.: The distances to the closest and 2^{nd}-closest hard locations of the three tasks according to the operation mode after 30 predictions of a learnt sequence, see Fig. 5.8(a), 5.8(b). The average distance to all remaining hard locations and the sequence errors, Fig. 5.8(c), 5.8(d). Fig. 5.8(e) shows the average precessing time to predict the next state of a sequence. The performed action tasks have been A) placing the robot hand closely above a table in front of the robot, and pushing a certain object aside by a lateral Cartesian motion on a transversal plane, B) lifting and placing a wastebin from and on the floor. All tasks have been tested with an activation radius of 10 and 30.

the robot communicates or swaps programs with workstations due to its distributed software architecture. To eliminate such varying delays we focus on the time needed for prediction.

Due to the random characteristics of the SDM model during memory storage, results cannot be absolutely exactly reproduced. Thus, analysis of SDM behaviour is a non-trivial task. That is why most of the experiments show averaged values of several runs. Nevertheless, valuable information and tendencies can be extracted from the series of tests.

5.3.1. Activation Radius

One of the goals is to find an appropriate activation radius to obtain the highest reconstruction accuracy of a motion trajectory, while retaining the advantages of distributed information storage. Empirical tests showed that the activation radius significantly influences the degree of generalisation. It turned out to be a crucial parameter concerning the reliability

Table 5.3.: The distances to the closest and 2^{nd}-closest hard locations (HL) according to the operation mode after 30 predictions of a learnt sequence. Also, the average distance to all remaining hard locations, the number of hard locations, the used activation radius and the sequence errors are shown. The performed actions were a) placing the robot hand closely above a table in front of the robot, and pushing a certain object aside by a lateral Cartesian motion on a transversal plane, b) lifting and placing a wastebin from and on the floor, and c) drawing a U-like shape in the air.

Task	Operation mode	$\|N'\|$	Dist. to closest HL	Dist. to 2^{nd} closest HL	Inc. %	Aver. dist. to all HLs	Inc. %	Errors	Processing time (ms)
			Activation radius: $r = 10$						
A	Arithmetic	495	11.50	17.60	53.04	1142.67	9836.26	1	< 1
	Bitwise	328	1.23	4.67	278.38	58.89	4674.68	0	1
	Sum Code	540	8.37	8.37	0.00	1100.44	13052.69	0	41
B	Arithmetic	495	9.93	21.87	120.13	1293.37	12920.49	0	< 1
	Bitwise	328	2.90	7.77	167.82	62.65	2060.33	0	2
	Sum Code	540	7.87	7.87	0.00	1170.62	14780.74	0	50
C	Arithmetic	495	6.50	14.23	118.97	851.38	12998.15	0	< 1
	Bitwise	328	1.43	4.90	241.86	56.93	3871.58	0	2
	Sum Code	540	10.10	10.10	0.00	845.30	8269.32	0	21
			Activation radius: $r = 30$						
A	Arithmetic	387	24.97	32.30	29.37	1094.74	4282.80	11	< 1
	Bitwise	136	20.33	22.33	9.83	58.25	186.50	28	2
	Sum Code	540	16.77	16.77	0.00	1090.53	6404.14	0	34
B	Arithmetic	387	141.00	149.00	5.67	1841.13	1205.77	9	< 1
	Bitwise	136	18.53	24.10	30.04	60.47	226.26	22	2
	Sum Code	540	20.90	20.90	0.00	1144.38	5375.48	0	40
C	Arithmetic	387	13.97	21.13	51.31	898.46	6332.89	0	< 1
	Bitwise	136	21.87	24.30	11.13	53.65	145.36	9	2
	Sum Code	540	19.07	19.07	0.00	819.42	4197.64	0	19

of any SDM prediction. Different activation radii have been studied, two of them are shown in Tab. 5.3. The smaller the activation radius, the less error-prone are the predictions of successive arm positions by the SDM. On the other hand, if choosing smaller activation radii, the overlapping regions in an SDM become smaller and the generalisation capability decreases.

The predictions in the arithmetic and the natural binary mode tend to get flawed when the activation radius exceeds 25, while the sum code mode turns out to be very robust up to a value of 11000. The latter is explained by the huge and redundant representation of the features in sum code mode.

5.3.2. Confusion of Patterns

How good the concepts, here the arm configurations, are in contrast to the rest of the memory can be expressed by the distance from the closest hard location to the mean distance to all remaining locations of the memory. The average distance of the current arm configuration to all remaining configurations within the SDM is conspicuously larger in the sum code and arithmetic mode than in the binary mode. This means that the binary mode is more susceptible to confusing prediction results of associated actions.

5.3.3. Prediction Errors

If prediction errors occur, the trajectory gets flawed—the action sequence may converge to a standstill or the robot may execute retrograde actions. Nevertheless, prediction errors not necessarily cause action sequences to essentially fail. Further predictions may unstuck the robot and the robot may proceed its execution of a learnt path successfully until its end.

The binary mode causes the most prediction errors as seen in Figure 5.8(d) and Table 5.3. The bit position has a vast impact on the represented value and flipping particular bits[6] causes major changes and inconsistencies in memory content. While accumulating information from several hard locations upon predicting this may lead to wrong results for the next associated address. This kind of problem is not observed in the sum code mode. The arithmetic mode shows worse predictive behaviour according to an increasing activation radius too, but still performs several degrees better than the binary mode.

An interesting fact is that both binary and arithmetic mode show the largest amount of prediction errors for task A; pushing an object on a table top. This result may be caused by the reduced operational speed during the pushing phase. Learning of arm configurations in that particular period results in many similar but unequal locations in a certain region of the memory. Information retrieval in such piled memory regions leads to prediction errors more quickly, as already explained in Section 5.2.3.

5.3.4. Prediction Time

Prediction time is mainly determined by search effort and metrical calculation. In the current implementation there is no search-space optimisation. For every prediction the algorithm has to check every hard location for whether it is in the neighbourhood of the current sensory input in order to consider it for output calculation.

[6] As happens in a random manner during learning and data distribution into memory.

Due to their compact representational form, both arithmetic mode and bitwise mode have fast prediction times. Moreover, the current implementation uses optimised built-in operators for distance calculation (e.g. addition, potentiation and XOR, respectively) of the used programming language. Whereas in sum code mode, the address vector of a hard location is represented by 22320 bits, which is 26 times more than in the other modes. But, concerning runtime, all the scenarios produce results in an admissible time.

5.3.5. Memory Size

An important parameter not listed in Tab. 5.3 is the memory size, which is the maximum number of hard locations N' the memory possesses. A fixed value of 10000 was set prior to the test, so that the learnt trajectories could entirely be represented by hard locations in the memory. Reducing the maximum number of hard locations results in a poor associative ability and yields a higher forgetting rate because the memory gets saturated faster. If it reaches its maximum capacity, memory items will get lost by forgetting but with a graceful degration. The maximum capacity has been extensively examined by Chou (1989).

5.3.6. Time

The date and time of events are neglected in an SDM. The reason is simple: The memory allows that closely related situations could be stored into memory locations that already host some information. The underlying mechanisms of memory storage and retrieval perform an averaging of new and already learnt information. As a result, different dates of closely related trajectories will influence each other and deny a reliable retrieval of any temporal information.

5.4. Discussion

Kanerva's model proposes to determine the similarity of memory items by using the Hamming distance. Due to the correlation of the bit position to the represented value the natural binary code showed problematic system behaviour when used with non-random data. Three varying approaches that differ in their information coding scheme and also in their distance metrics were discussed to assess the SDM performance. The bitwise implementation with Hamming metric does not performs very well. The separability of memory items was insufficient and the memory's sensitivity to random bit flips in the distribution phase during the memory storage process results in the highest prediction error rate of all tested modes. Note that the SDM performance was mentioned to work best when used with random data. As mentioned before, a robot's environment is hardly random. Grouping of the bits as double sets and making use of an Euclidean distance metric significantly improved the performance of the memory but with the side effect of losing some characteristics of the original model. When swelling the memory with large groups of thermometer style sum code mode the characteristics of binary space still hold, while the performance is similar as or slightly better than in arithmetic mode.

Nevertheless, the analogy to the brain's behaviour to retrieve similar solutions to previous problems from memory rather than to compute new ones inspired the current investigations on a biologically plausible memory model for mobile robot arm manipulation. Though never be intended as a biologically fully plausible model of short-term (working) or long-term

memory (Hely, 2006), the SDM was considered alongside the eminent cerebellar models by Marr (1969) and Albus (1971) (c.f. Sections 2.3.4.1 and 2.3.4.2).

A content-addressable, associative memory in the form of an SDM is capable of storing and retrieving robot arm trajectories for manipulation tasks. The prediction of a residual arm trajectory based on simple excitation of memorised information has been proven. If the arm is placed somewhere close to a learnt trajectory, the robot is able to perform the motion autonomously.

The distribution of data into several memory locations makes such a system robust against memory loss. If physical memory locations are erased or lost by accident, the memory retains its ability to predict motion sequences successfully up to a high amount of damage.

A central issue of memories that should remember action sequences are higher-order state transitions. So called cross-sequences and higher-ordered state transitions, to a limited degree, can be tackled by folding the memory and thus incorporating second-order, third-order transitions *et cetera* in the memory. Unfortunately, this linearly raises the required storage capacity and is just eligible as long as the number of folds exceeds the number of concurrent states.

One of the major difficulties while working with systems based on binary space is the appropriate encoding of the information representing the world. Which features are the most important features to represent the world? Many approaches provide an exploratory data analysis to reduce myriads of stochastic input variables to comparatively few salient features for meaningful system outputs, e.g. exploratory data analysis approaches such as *principle component analysis* (PCA). When, however, their classification parameters are fixed after an extensive exploratory training phase, such systems lose their flexibility to progressive learning. They become sensitive to the salient features only. If, however, additional input-output relations should be learned continuously the system has to be retrained to derive a conclusion again. The SDM, however, bears the possibility for online learning and adaption of the memorised model of the world.

The SDM provides characteristics that makes it attractive for an application in cognitive robotics research. The developed system provides a computational model to test the hypothesis whether memory retrieval in humans is a convergent or non-convergent process. The memory model can be used as a pattern recognition system that is immune to noise up to a high degree and also robust to failure of individual memory locations. Memory structure can be changed without explicit retraining and it supports one-shot learning, which is a promising and worthwhile characteristic in modern robotics. It will come into sharper relief in later chapters. If the memory reaches its maximum capacity or if memory locations are damaged it degrades gracefully. Through the natural type of forgetting, motion sequences that have not been recently used have a higher probability of being forgotten first. The SDM provides a nice model to detect conceptual familiarity based on raw sensory data. Nevertheless, problems of the SDM model are that once learnt data cannot be selectively erased, e.g. sequences that do not achieve a certain goal. Thus, undesired or unnecessary memory contents may interfere more recent and important data. A disadvantage of the SDM model, when used with counters, is its information capacity of just 0.1 bits per bit (Kanerva, 1988). This disadvantage has been overcome by replacing the counters according to the approach proposed by Furber et al. (2007). The non-modifiable *a priori* given random address space has been eliminated by using a randomised reallocation algorithm for dynamic memory allocation. The real-time allocation saves a lot of computational time for memory setup. For tests that do not need all memory hard locations it is also much faster.

6

Sparse Distributed Memory Compared across Different Modalities

We need creativity in order to break free from the temporary structures that have been set up by a particular sequence of experience.
(Edward de Bono, physician, author and inventor, born 1933)

Contents

While the work of this book focuses on studying SDM-based manipulation, a more comprehensive study on SDM-based robot navigation is made in a yet unpublished PhD thesis by Mateus Mendes (personal communication, Jan 3, 2009). The comparison outlined in this chapter describes a short-term joint cooperation, resulting from two separate projects, that finally engender a conference article (Jockel et al., 2009).

The power of using SDM for robotic manipulation applications has already been shown in Section 5. The question arose if this power persists if transferred to other robotic domains, particularly to other modalities. Hence, tests were performed on two different platforms, designed for different purposes: one for navigation (LIZARD), the other for manipulation (TASER). Both platforms together are shown in Figure 6.1. A comprehensive introduction to both technical robot systems can be found in the particular sections of Chapter 4. In both cases, the SDM is used to store sequences of events which are later to be repeated. The main

purpose of this study is to show that the same SDM model can be applied to totally different subfields of robotics that moreover use different modalities to sense the environment[1].

6.1. Some Remarks on Robot Navigation

Localisation and recognition of the environment is a fundamental topic in mobile robotics. According to Matsumoto et al. (2000) different techniques for image-based navigation can be categorised as:

Model-based: These approaches utilise distinct objects such as edges, corners or (artificial) landmarks in visual images to match them against learnt 3D models of objects.

View-based: These approaches consider the appearances of the environment, mostly in a sequential order, rather than particular landmarks.

A major problem in model-based robot navigation is to structure the information to allow for an efficient indexing of the objects especially when the model is rich in details. The latter approaches can be used in unstructured environments, making any installation of artificial landmarks becomes superfluous. Rather than working on 3D object shapes, they employ the 2D image information of a view sequence that is memorised somehow. The information of an environment in the memory is defined by the corresponding sensor patterns directly. Nevertheless, requiring a huge memory and high computational costs for matching are some of the disadvantages of view-based approaches. A general question concerning the latter approach is: how to memorise views of the environment and how to realise the matching between a currently sensed and memorised images?

Matsumoto et al. (2000); Ido et al. (2009) propose an approach that is based on a view sequence of static images of the environment and a corresponding control commands look-up table. During a supervised learning stage, a view sequence and corresponding motor commands are learnt while guided by a human instructor. Images are memorised as high-dimensional vectors $(x_1, x_2, \ldots, x_n)$ with n being the number of pixels. During an autonomous run, the matching properties between current views and memorised ones constitute the localisation of the robot. A simple block matching algorithm is used to compute the distance between the current and all memorised views. Therefore, a central rectangle area of the memorised view is considered as template and the current view is considered as the search area. The memorised image with the smallest horizontal displacement of the template rectangle to the current view defines the hypothesised position of the robot according to the image's sequence position. Thereafter, the related motor commands are retrieved to guide the robot through the learnt path autonomously. The displacement of a current view and its correlating view in the memory can be used to realign the robot when the displacement exceeds a certain threshold.

The SDM theory provides a memory structure that seems to be made for such appearence-based navigation techniques. It can memorise static views of the environment together with the corresponding motor commands as bit patterns. Matching current sensations to

[1] This chapter outlines the result of a collaborative study between the CINACS Int'l. Research Training Group, Dept. of Informatics, University of Hamburg, Germany and the ISR - Institute of Systems and Robotics, Dept. of Electrical and Computer Engineering, University of Coimbra, Portugal and is mainly based on the work presented in Jockel et al. (2009).

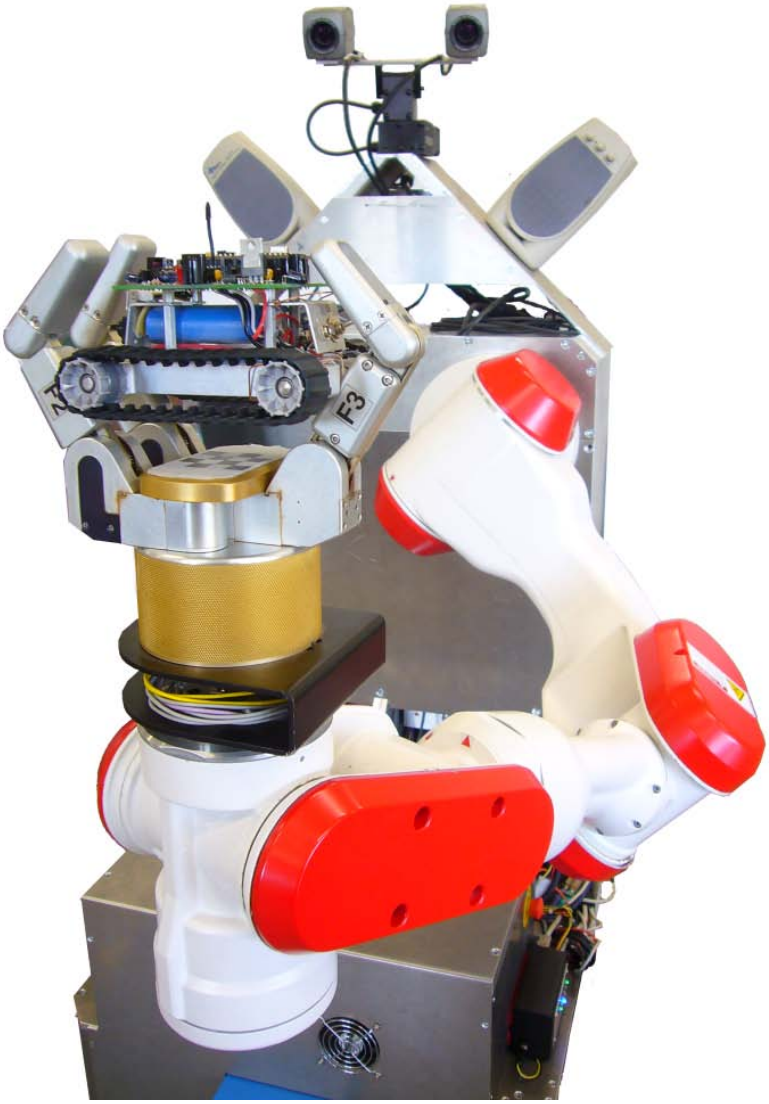


Figure 6.1.: Platforms used in the experiments: TASER's robotic arm is grasping the small tank-style robot LIZARD.

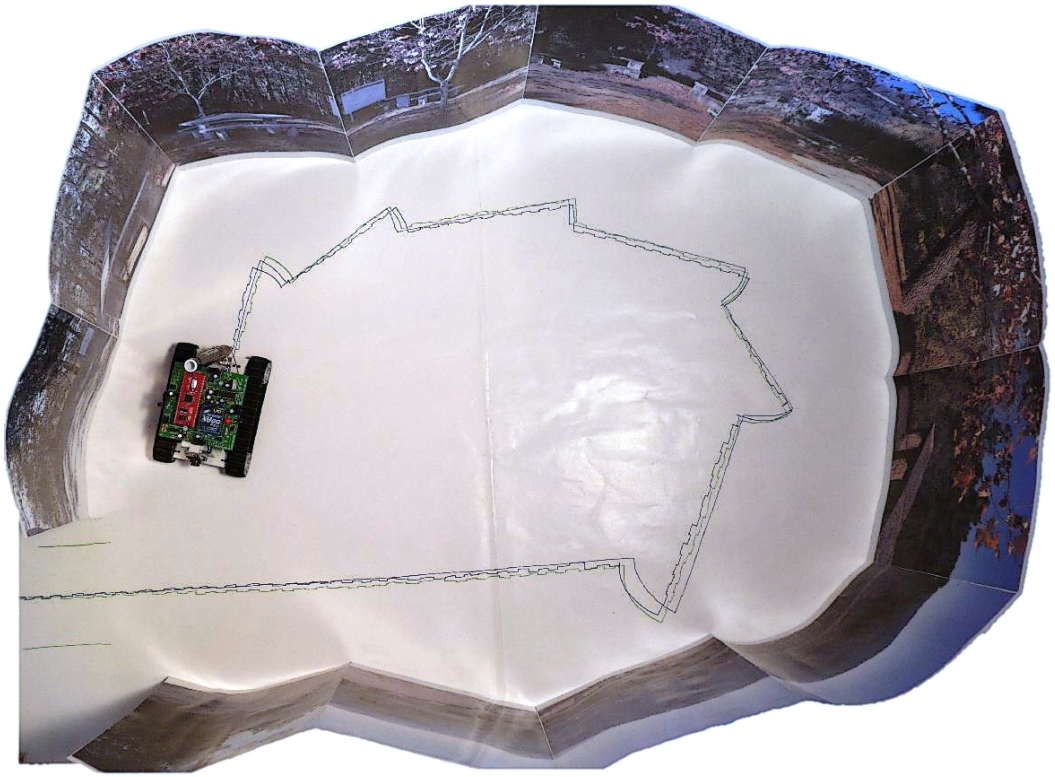


Figure 6.2.: LIZARD in its experimental environment. Figure adapted from Mendes et al. (2008). The blue line depicts the path taught during a supervised learning stage. The other lines result from autonomous runs of the robot guided by its memory.

experienced ones is realised while performing a normal storage and retrieval procedure via its built in distance metric. An implementation of such an SDM-based navigation algorithm is proposed by Mendes et al. (2008) and will be briefly described in the following section.

6.2. SDM-based Navigation Using a View Sequence

The LIZARD robot, a Surveyor SRV-1, is a tank-style mobile robot system. The robot is equipped with an SDM to study its navigational capabilities based on viewing sequences (Mendes et al., 2008). For comprehensive technical details of the hardware and software the reader is referred to Section 4.1.2 and Section 4.2.2. Even if not bound to it, LIZARD is operated in a testbed surrounded by a panoramic image shot on a famous mountain in Portugal (see Figure 6.2). It is equipped with a small CCD camera and the navigation is based on a sequence of images inspired by the above-mentioned approach of Matsumoto et al. (2000).

During a supervised learning stage an instructor remotely guides the robot through its environment. Images and additional information such as the motion that leads the robot to the next image position are stored in the SDM. It is obvious that two images of the same real world scene are hardly ever identical when dealing with digital image sensors. Ubiquitous changes in the lighting conditions as well as the sensor noise are hard to avoid. Thus, some

preprocessing has to be applied to the images before storing them into the memory. The images are preprocessed by applying a conversion from JPEG to 8-bit PGM followed by a brightness equalisation (Mendes et al., 2007). An illustration of alternative preprocessing steps and their respective effects when used for SDM is detailed in Mendes (2009). The gray-value images that will be stored in the memory have a resolution of $80px \times 64px$ with each pixel taking a value between $[0, 255]$. Thus the input and output patterns of the SDM consist of arrays of bytes where each image pixel is represented as an 8-bit integer. The composition of an SDM input vector is as summarised in Equation 6.1:

$$x_i =< im_i, seq\_id, i, timestamp, motion >, \tag{6.1}$$

where im_i is the last image[2], *seq_id* is an auto-incremented unique 4-byte integer for each sequence. The *seq_id* is used to identify the sequence to which the vector belongs. i is also an auto-incremented unique 4-byte integer index for every vector in the sequence. It is used to quickly identify the order of the images within a sequence. The *timestamp* is a 4-byte integer to represent Unix timestamps. The timestamp is read from the operating system, but is not used for navigation purposes yet. *motion* is a single character that represents the movement the robot was performing when the associated image was grabbed. According to the above-mentioned parameters they yield a vector size of 41064 bits consisting of 5120 bytes for the image of resolution $80px \times 64px$ and 13 bytes for the overhead information such as motion, timestamp, sequence and vector *id*.

Before storing any information, the noise level is computed to dynamically adjust the SDM's access radius by comparing three images recorded at one and the same location[3] (Mendes et al., 2007). In doing so, the noise level is defined by the mean distance between the three resulting SDM input vectors that represent the images as bit patterns.

The SDM is used to store the input vectors as shown in Equation 6.1. However, addressing the memory in an autonomous run is done by using just one image. During the autonomous run, the robot grabs an image and identifies its position within the view sequence by recognising the closest image of the learnt sequence (im_i). Afterwards, it performs the corresponding motion to proceed to the next decision point. In an SDM, the closest image is defined by the vector that provides the smallest distance with respect to the used distance metric. By reaching the next decision point, the robot grabs a new image and so on. At each decision point, if the robot detects a horizontal shift between the grabbed image and the stored one, it tries to align itself iteratively by slightly turning to the left or to the right.

As mentioned above, addressing is done by using only im_{i-1} and not the whole vector. The remaining bits could be set at random, as Kanerva suggests, but it was considered preferable to tune the software to be able to calculate similarity between just parts of two vectors, ignoring the remaining bits. This saves computational power and reduces the probability of false positives being detected.

6.3. SDM-based Manipulation

TASER is a service-robot equipped with a 6 DoF robot arm. Manipulation is mainly based on the same implementation as presented in Chapter 5. To avoid a redundancy here, the reader is referred to Section 4.1.1 and Section 5.1 for a comprehensive discussion of the hardware

[2]This image is not available when only the first image has been captured.
[3]Without any environmental changes being applied.

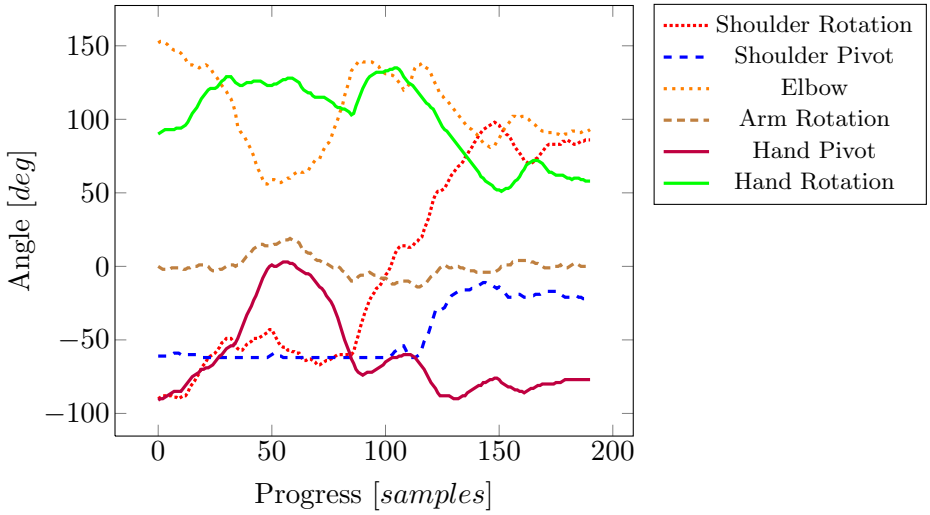


Figure 6.3.: The trajectory of the 6-DoF robot arm stored to the SDM. The graph only represents the joint angles that correspond to $j_1, \ldots, j_6$ of x_i (see Equation 6.2). Further information on x_i, also stored to the SDM, is omitted for the sake of clarity.

and software aspects of the implementation. To refresh at least some details necessary for the subsequent comparison, the implementation will be briefly summarised in the following paragraphs.

In this experiment, manipulation is based on a sequence of 3D coordinates and roll, pitch, and the corresponding yaw angles of the manipulation tool. Henceforth, it is also used as arm configuration. During a learning phase, a sequence of robot arm joint angles and the corresponding tool centre point (TCP) as well as the tool orientation are captured at discrete, consecutive time steps. According to vector x_i presented in Equation 6.2 (also cf. Equation 5.2, Section 5.1.2) the data is stored into the SDM.

$$x_i =< j_1, j_2, j_3, j_4, j_5, j_6, x, y, z, \chi, \psi, \omega, seq\_id, i >, \tag{6.2}$$

where at a particular point i in time each j_n represents the angle of the corresponding arm joint. The 3D coordinates of the tool centre mounted at the end of the robot arm in relation to the robot coordinate system are depicted with x, y, z, and χ, ψ, ω describe the roll, pitch, and yaw tool orientation. Each of the above-mentioned variables are represented as 8-byte double precision value. Finally, the seq_id is a unique 4-byte id for each sequence, and i is a unique auto-incrementing index for each vector of a sequence to determine the order, also 4-byte long.

During an autonomous execution the robot is confronted with a particular arm configuration. The robot retrieves the circumstances of the particular joint constellation from the sequence of arm configurations stored in its memory. The associated arm constellation is used to iteratively guide the robot to the next position of the memorised sequence. Accordingly, the robot should follow the learnt trajectory (see Figure 6.3) autonomously. Addressing the memory is done by only presenting a single arm configuration represented by the particular joint angles (j_n with $n \in \{1, 2, \ldots, 6\}$) of time point $t-1$. During an autonomous execution the SDM will predict j_n^t through its association to j_n^{t-1}.

Table 6.1.: Comparison of the performance of the SDM in two different domains: navigation and manipulation. The memory only contains one copy of each datum.

Operation mode	Dist. to closest HL	Dist. to 2^{nd} closest HL	Inc. %	Aver. dist. to all HLs	Inc. %	Errors	Vector Bits
Manipulation domain (*access radius: 10*)							
Arithmetic	86.40	103.97	20.33	1797.54	1980.49	0	832
Bitwise	14.17	33.07	133.41	63.52	348.37	7	832
Sum Code	13.10	82.40	529.01	1274.55	9629.36	0	22320
Navigation domain (*access radius: dyn.*)							
Arithmetic	35240	56511	60.36	172236.53	388.75	29.0	41064
Bitwise	7575	8195	8.18	9738.01	28.56	31.4	41064
Sum Code	1980	3155	59.34	9460.60	377.75	29.0	82024

6.4. Experiments and Comparison

Different tests were performed in order to assess the behaviour of both systems. In order to find a better solution to the problem of non-random data encoding and its distance calculation, different solutions are studied in both of the proposed domains. These are: 1) Natural binary code and arithmetic distance as proposed in Section 5.2.1.2 (arithmetic mode), 2) the natural binary code with Hamming distance according to Section 5.2.1.1 (bitwise mode), 3) and the sum code with Hamming distance presented in Section 5.2.1.3 (sum code mode). The results are shown in Table 6.1 and Table 6.2 and will be discussed in the following paragraphs.

Table 6.1 outlines the results that are obtained while the memory contains just one copy of each datum. This means that there is no distribution of the data during the storage procedure. In this special case, the SDM acts as a conventional random-access memory that has the feature of being content-addressable while retrieving the closest physical address. In turn, Table 6.2 represents the same values as Table 6.1, but this time obtained with five copies of each datum stored to the memory.

The values shown in the tables are: the distance from the input address to the closest hard location (the desired prediction), the distance from the input address to the second closest hard location, and the average distance from the input address to all remaining hard locations in the memory. There is also a measure of the increases, in percentage, which expresses how successful the system is in separating the desired datum from the pool of information in the SDM. The vector bits characterise the size of the input and output vector applied to the SDM.

The tables also show the number of prediction errors. A prediction error occurs every time there is a step back in the sequence. For instance, if at time t and $t+1$ the predictions are, respectively, x_t and x_{t-1}, that is a prediction error, the robots are not expected to return in a sequence.

6.4.1. Navigation

The results for the SDM-based navigation are obtained by using a sequence of 55 images, which are equalised before processing. The tables show the average of 30 operations, except

Table 6.2.: Comparison of the performance of the SDM in two different domains: navigation and manipulation. The memory contains five copies of each datum.

Operation mode	Dist. to closest HL	Dist. to 2^{nd} closest HL	Inc. %	Aver. dist. to all HLs	Inc. %	Errors	Vector Bits
Manipulation domain (*access radius: 15*)							
Arithmetic	12.35	14.4	17.32	1275.30	10227.33	0	832
Bitwise	2.87	6.03	110.47	62.83	2091.90	0	832
Sum Code	12.67	12.67	0.0	1271.31	9936.67	0	22320
Navigation domain (*access radius: dyn.*)							
Arithmetic	35528	36467	2.64	172249.43	384.85	23.2	41064
Bitwise	7547	7831	3.77	9740.06	29.06	26.0	41064
Sum Code	1913	1968	2.90	9446.52	393.83	24.4	82024

for the navigation errors, which are the average of five autonomous runs. The tests are performed using the arithmetic mode, the bitwise mode and the sum code mode. The bitwise mode uses the natural binary code and the Hamming distance and uses 8 bits per pixel to encode 256 graylevels. Furthermore, the sum code mode uses 16 bits per pixel such that the dimensionality of the SDM input vector almost doubles. In this mode the encoding is limited to 17 graylevels. In each case the access radius was computed dynamically in the beginning of the sequence as described above and in Mendes et al. (2007) and hence has been set to 20 % above the noise level. The offset was experimentally determined.

6.4.2. Manipulation

The results for the SDM-based manipulation are obtained by using a sequence that consists of 190 arm configurations. The joint angles of this sequence are shown in Figure 6.3. The trained memory is addressed by an initial arm position. Consequently, the memory iteratively predicts the next associated arm configuration and guides the robot arm to the particular positions autonomously. The average of 30 such predictions is shown in Tables 6.1 and 6.2.

As in the SDM-based navigation experiment, the tests are performed by using three different encoding modes: The arithmetic mode (cf. Section 5.2.1.2), the bitwise mode (cf. Section 5.2.1.1) and the sum code mode (cf. Section 5.2.1.3). The latter mode enlarges the vector size from 832 bits to a total length of 22320 bits according to the maximal operational range of the robot arm with respect to Equation 5.4. In contrast to the image-based navigation, which has to deal with a lot of image noise, the access radius was chosen experimentally as the noise level was reasonable.

6.4.3. Comparison

The random characteristics of the SDM, the structured environments in which the tests were performed, suggest that the results obtained may be different even in very close circumstances. However, the robustness of the implemented models can be emphasised in consideration of the consistent results that are obtained in two completely different domains: the domain of vision-based navigation, in which the data is of very high dimensionality and contains a high amount of noise, and in the domain of robot manipulation, in which the data

contains small amounts of noise and the dimensionality is relatively small. The approach is the same in both domains: Store sequences of events during a supervised learning stage, and use them later to autonomously perform the same tasks again. In both cases, the models proved adequate and behaved as the SDM theory (cf. Chapter 3) predicts.

In the case of vision-based navigation, the results show a higher number of sequence prediction errors, which is probably due to the presence of noise in the images, but in no way seemed to compromise the ability of the robot to successfully follow the sequences in which the images were rich enough.

In the case of manipulation, the high precision of the robot arm shows—if at all—only a very small level of noise. Robot arm trajectories are successfully learnt and followed.

The distribution of the data proves to be beneficial in regard to the robustness of the system, improving its ability to retrieve the right datum from the pool of data and reducing the number of prediction errors. Additionally, the bitwise mode performs slightly below the arithmetic and the sum code operational modes in both domains.

An interesting characteristic of this approach is that regardless of the initial position, the robot is able to converge to the closest point in the sequence, thus always executing the assigned task with success. This happens in manipulation, because we are using absolute positioning of the joints, but not in navigation, where the robot has no information on absolute positioning.

6.5. Discussion

This comparing study, too, unveils the most difficult problem of the SDM theory for robots. The SDM theory is best suited to deal with random data, but real world sensorial inputs are hardly random. The performance of the memory is significantly affected. The results of both domains show, as already mentioned in Section 5.4, that the performance of the system depends on the method of encoding the information. The arithmetic and sum code modes show better results in both domains. However, those methods require additional processing and the former may weaken some characteristics of the original SDM. It was proven that the distribution of data stored in the memory makes the system more robust by reducing the number of prediction errors.

Besides the already shown SDM capability of storing and retrieving robot arm motion trajectories, the same SDM model can successfully accomplish tasks of various domains. Input data that originate from a totally different type of sensor, a CCD camera, has been successfully applied to the memory. It is possible to navigate a robot based on its memory of experience which contains motion commands and view sequences. With the proposed mechanism the kidnapped robot problem could be solved. A robot that is kidnapped from one place and repositioned anywhere else in or close to the sequence can use its SDM to realise the abduction. What matters is just its current view that is matched against memorised views. If replaced within any known sequence the robot will be able to proceed along its path according to its memory content.

Compared to the proposal of Kanerva, who suggested an SDM with $n = 1000$ dimensions, we realised a software implementation that is suitable to work with up to $n = 82000$ dimensional vectors. This, at least to some extent, shows how good the model scales to problems of higher dimensionality.

7

Sparse Distributed Memory for User Intention Detection and Learning

A retentive memory may be a good thing, but the ability to forget is the true token of greatness.
(Elbert Hubbard, American editor, publisher and writer, 1856-1915)

Contents

A major goal of robotics is that robots become a part of everday human life. A key challenge is to provide robots with human-like capabilities, e.g. sophisticated motor skills

for flexible and precise motions to interact and cooperate with the environment and human users. Most robot interfaces do not support easy and natural interaction nor an easy teaching of robot skills. An alternative is to use learning and adaption techniques for online and offline learning of low-level motor behaviours. A favoured approach is to teach robots by demonstration without explicitly programming and reprogramming new motor skills. *Learning by demonstration* (Billard et al., 2008), also known as imitation learning, is the ability to recognise and reproduce the actions of others.

Within this study the SDM will be used to learn motion sequences of a remotely controlled robot arm. Remote operation is realised by employing a telemanipulation system. Human operators use a haptic force-feedback device with similar DoF as the robot arm and a stylus-like manipulandum for an interactive control. It will be shown that SDMs are suitable to model teleoperative tasks with the advantage of fast learning and generalisation of motion trajectories.

The strongest motivation for this work is to recognise the intention of a human operator and to improve the task execution by allowing the system to adapt to the operator's need, e.g. take over a task, offer help wittingly. To offer the operator a certain aid, it is fundamental to successfully interpret the desired goal behind his actions. The system for learning, representing, modelling and transferring skills have to deal with highly nonlinear relationships between the stimuli and responses (sensor/actuator systems). In the area of nonlinear control, neural network techniques show great potential by eliminating the need for solving difficult nonlinear mathematical models. Neural networks can learn from examples online and can continuously optimise the system's performance.

7.1. Intention Recognition

Intention recognition, also-called intention detection, is an important research issue in human-computer interaction (HCI) and human-robot interaction (HRI). Our understanding of other people's behaviour relies significantly on assumptions about their intentions. Thus, identifying underlying intentions is an essential functionality of a technical system, e.g. a robot, to provide adequate support to a human user when engaged in a collective activity.

Humans understand each other's intentions by interpreting information received by the senses and communication. Several authors propose probabilistic models based on Basian networks for intention detection of a human user (Horvitz et al., 1998; Schrempf et al., 2007; Rößler, 2009). Further models are based on Hidden Markov Models (HMM) (Yu et al., 2004; Kelley et al., 2008) or Layered Hidden Markov Models (LHMM) (Aarno and Kragic, 2008) to classify decomposed sets of complex human actions. HMMs are widely used for pattern classification. Among several features they have some disadvantages regarding the classification of noisy input trajectories. Furthermore, they need sufficiently large training sets (Yu et al., 2004) and their performance diminish when assessing high-dimensional problems. Bayesian networks have the disadvantage that they are strongly dependent on the domain knowledge and their computational time is rather slow. A problem of both statistical approaches, HMM and Bayesian networks, is to get the initial assignment of transition probabilities between system states. The domain knowledge, e.g. *all* possible motions, have to be completely specified *a priori*.

Humans, additionally, have a vast capability to predict an arbitrary intention under a high degree of uncertainty and ambiguity. Since the SDM is particularly suitable to deal with

high degrees of noise and just requires a small amount of a priori knowledge, it will be used to identify the intention of a human teleoperator that remotely controls a robot arm.

7.2. Learning by Demonstration

Learning by demonstration (LbD), also known as *programming by demonstration*, *teaching by showing* and *imitation learning* has become a key research topic in robotics for motor learning and control, gesture recognition and visuo-motor integration. LbD can be seen as a technique that develops policies from example states to action mappings (Argall et al., 2009). The main purpose is to replace time-consuming manual programming of a robot by an automatic programming processes where an expert shows the robot how to solve a task.

The quality of a learned LbD policy depends heavily on the quality of the provided demonstrations. In reality, however, teacher demonstrations may be ambiguous, unsuccessful or suboptimal in certain areas of the state space. Suboptimality is most common in demonstrations of low-level tasks, such as movement trajectories for robot arms. Here, demonstrations rather serve as guidance to solve particular tasks than offering a complete or best solution. Some approaches remove actions that do not contribute to solutions, other approaches smooth or generalise suboptimal demonstrations to improve the teacher's performance. This can be done with training data from multiple repeated demonstrations by a single user or from multiple teachers.

Research in the area of robot LbD was initially driven by symbolic AI approaches to segment demonstrations into primitive actions by an expert to enable a technical system to reason about them in later steps. More recent research was driven by including multiple example inductive learning, case-based learning and its special form of memory-based learning and analytical methods such as explanation-based learning (Nicolescu, 2003). *Inductive learning* uses multiple example demonstrations to make predictions of future ones. *Case-based learning* stores and generalises a set of given training instances, which is also-called *lazy learning*. *Memory-based learning* techniques operate on a more detailed level of experiences as case-based methods do. *Explanation-based learning* techniques usually act on symbolic domain models that are analytically refined with each new observation of a given example.

A key challenge in learning manipulation tasks is the curse of dimensionality that arises from having several DoF causing a large state and action space. The perception of the demonstration itself, which typically involves making a good data capture of what is shown to the robot, is also a challenging problem. The demonstrations can be performed by e.g. manually guiding a robot, by teleoperating it with a remote control device, by visually capturing the motion of an instructor (Hüser et al., 2006) or by directing the robot on the basis of a common language (Zhang and Knoll, 2003). More recent approaches study biological approaches to provide imitation learning based on neural networks (Billard and Matarić, 2001; Billard, 2002; Kuniyoshi et al., 2003).

In this chapter we investigate ambiguous demonstrations by multiple teachers that will be learnt by a multi-SDM architecture. The performance of predicting intended high-level actions based on multiple and single teacher training data will be evaluated in detail with the multi-SDM operating as a task approximator. The training data consists of low-level motion trajectories that originate from interactive high-level task demonstrations via a teleoperation system. The LbD policy which will be learnt is a mapping function from a robot's state, particularly its arm configuration, to an action sequence.

7.3. Telemanipulation

Telemanipulation is part of telerobotics. It generally refers to robotics at a distance (Greek: *tele*) mostly with a human operator involved in a control-loop. Usually the human operator accomplishes high-level tasks such as cognitive decisions, planning, *et cetera* while the robot is responsible for the mechanical implementation. Teleoperation stresses the task-level, while telemanipulation highlights object-level manipulation (Siciliano and Khatib, 2008).

Telerobotic applications are split into a master and a slave side that are connected over a teleconnection, e.g. computer networks such as WAN, LAN, WLAN. At the system's local side (master) a human operator usually has input devices, e.g. mouse, keyboard, joystick, haptic device, glove, exoskeleton *et cetera*, and output devices, e.g. feedback device, monitor, and so forth, to control the remote robot side (slave) including all its sensors and actuators.

Current architectures vary from *direct control* to *supervisory control*. Though a bit older, a comprehensive summary of latter ones can be found in Sheridan (1992). The former control method operates at the motion level without automated help while the latter requires some basic system intelligence and autonomy to interpret and execute high-level commands.

Applications have been motivated by the issue of human safety in hazardous environments, reaching remote environments, and for power and position scaling. To mention just a few of the most advance telerobotic systems: the DaVinci[1] robot for minimal invasive surgery, the ROKVISS system (Preusche et al., 2006) for space telerobotics, and remotely operated underwater vehicles (Amat et al., 2001).

The study discussed in this chapter uses a bilateral[2] telemanipulation architecture (Bruder (2009) and Section 4.2.3) that provides direct control of the MHI PA10-6C robot arm (Section 4.1.1.2) by using a PHANTOM® Desktop™, mouse, keyboard and monitor as input and output devices[3]. Through an extension of the telemanipulation system with the SDM a shared control architecture will be obtained that is able to provide automated help to a human operator and provides some degree of autonomy. A rough structure of the telemanipulation system is depicted in Figure 7.1 and 7.2. For a more detailed description of the telemanipulation system and its structure confer Section 4.2.3 and Figure 4.10. The SDM is used as a classifier that relates remotely executed actions to already experienced action sequences.

7.4. The Multi-SDM Architecture

An extension of the SDM architecture becomes necessary when using it as classifier to compute the task membership of varying action sequences. To distinguish between different actions, the sequences have to be labelled or tagged. If the action labels would be encoded in the memory's content vector, e.g. each character is represented as a set of UTF-8 octets, some problems will arise. Considering the SDM characteristic that several locations participate in storing information from similar but unequal sequences: Locations that participate in storing information of more than one sequence bear the consequence that the encoded character set (the action description) will be interfered by another character set. Later, when

[1] `http://www.intuitivesurgical.com/index.aspx`

[2] In a bilateral telerobotic system the slave robot does not only measure forces, but also displays forces to provide the operator with virtual instead of simple remote environments.

[3] A video of the TASER robot being teleoperated can be found at `http://tams-www.informatik.uni-hamburg.de/research/robotics/service_robot/videos/`

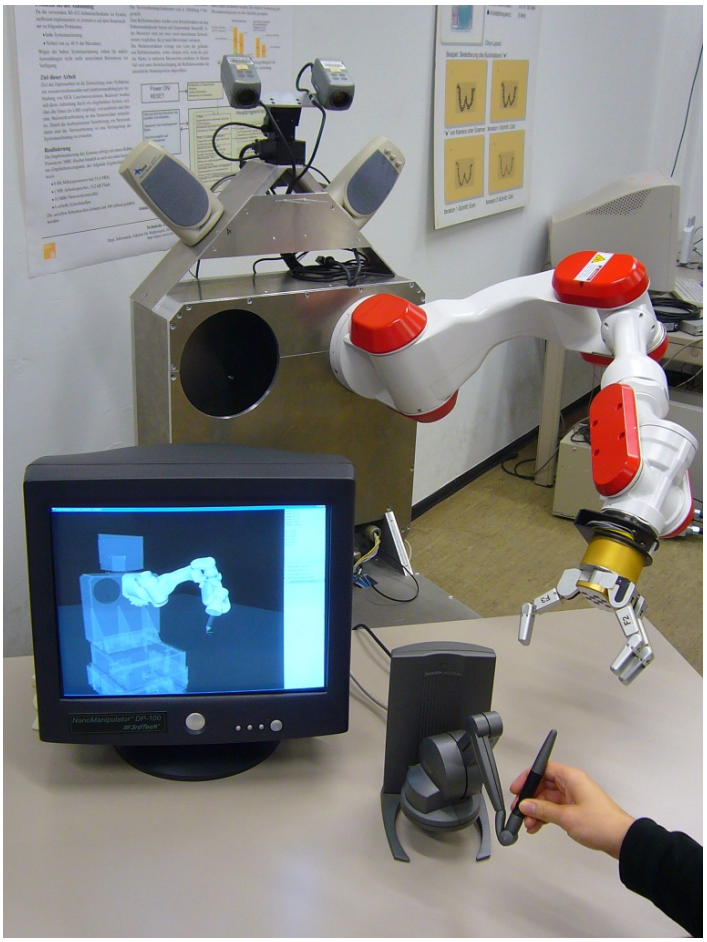


Figure 7.1.: TASER being teleoperated with a PHANTOM® Desktop™ haptic interface. A detailed schema of the telemanipulation architecture can be found in Section 4.2.3.

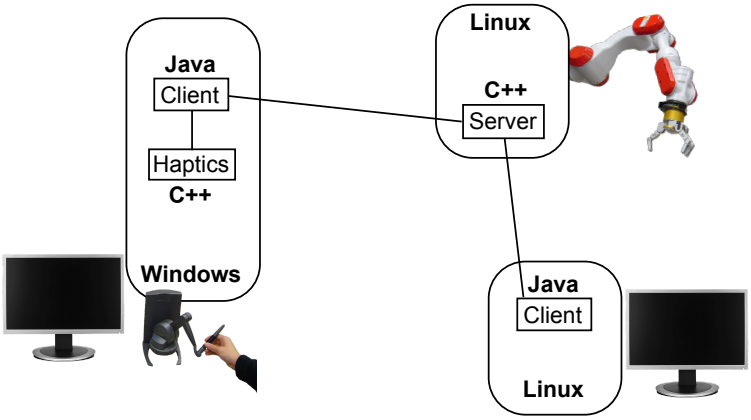


Figure 7.2.: Rough structure of the distributed telemanipulation architecture. Lines between oval boxes depict network connections. The left side shows the remote workstation to control the robot arm (upper right). Moreover, an arbitrary number of clients can also display the actions performed by the physical robot. For more details see Figure 4.10

reading information from overlapping memory regions the string that describes the membership of a certain arm configuration to a specific action sequence may not be retrieved accurately. Thus, the SDM architecture is extended with a higher hierarchical layer that maintains multiple SDM instances.

7.4.1. Multiple SDM Instances

Implementing a top layer that maintains several SDMs leads to an introduction of semantical context information into the memory framework. A number of SDM instances l will be used according to the number of tasks that should be distinguished. Let the memory be trained with several action sequences that belong to an arbitrary number of different tasks. During an initialisation phase the user has to define the number of different high-level tasks and the appropriate task titles that should be learnt by the memory. Then the respective number of SDMs will be generated and entitled with a given task description. Furthermore, the user has to assign a small number of training sequences to each SDM with respect to the task the trajectory belongs to. This can be seen as a supervised learning step. The Multi-SDM architecture can be extended with further SDM instances easily at any time. The *multi-SDM architecture* is shown in the left box of Figure 7.3. A multi-SDM class diagram is given in Figure 7.4 to illustrate the static structure of the system by its main objects, relationships, operations and attributes.

7.4.2. Generalisation

Let us consider a number of trajectories belonging to a certain task and the respective SDM instance. As described above, the robot arm motion sequence is no longer preprogrammed but rather originates from a human operator. Instead of going to much into detail—a precise description of the tasks will be given later—hardly none of the trajectories will be identical to any other. The human operators follow different strategies to solve tasks. They use the haptic

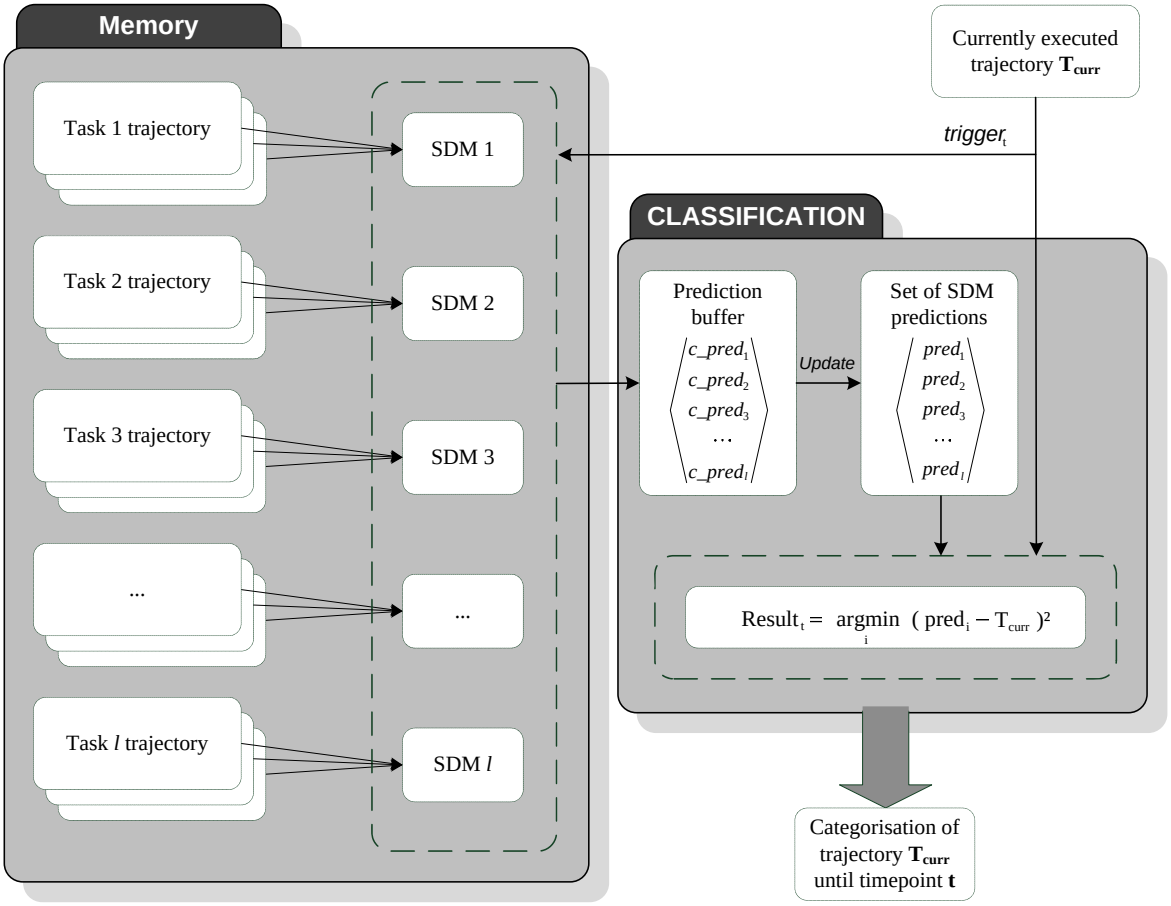


Figure 7.3.: The multi-SDM architecture consists of several SDMs, whereas each SDM is responsible for and entitled with a certain high-level task (left box). During training, task trajectories are assigned to the corresponding SDM instance. During prediction, consecutive snapshots of arm configurations of a yet unknown motion trajectory $\mathbf{T}_{curr}$ are presented as *trigger* to all SDMs continuously. Within given intervals, the quadratic distance between individual SDM predictions and the input trajectory are compared. The SDM that provides the least relative error over all intervals is the winner and the respective SDM title describes the intended action of the human operator.

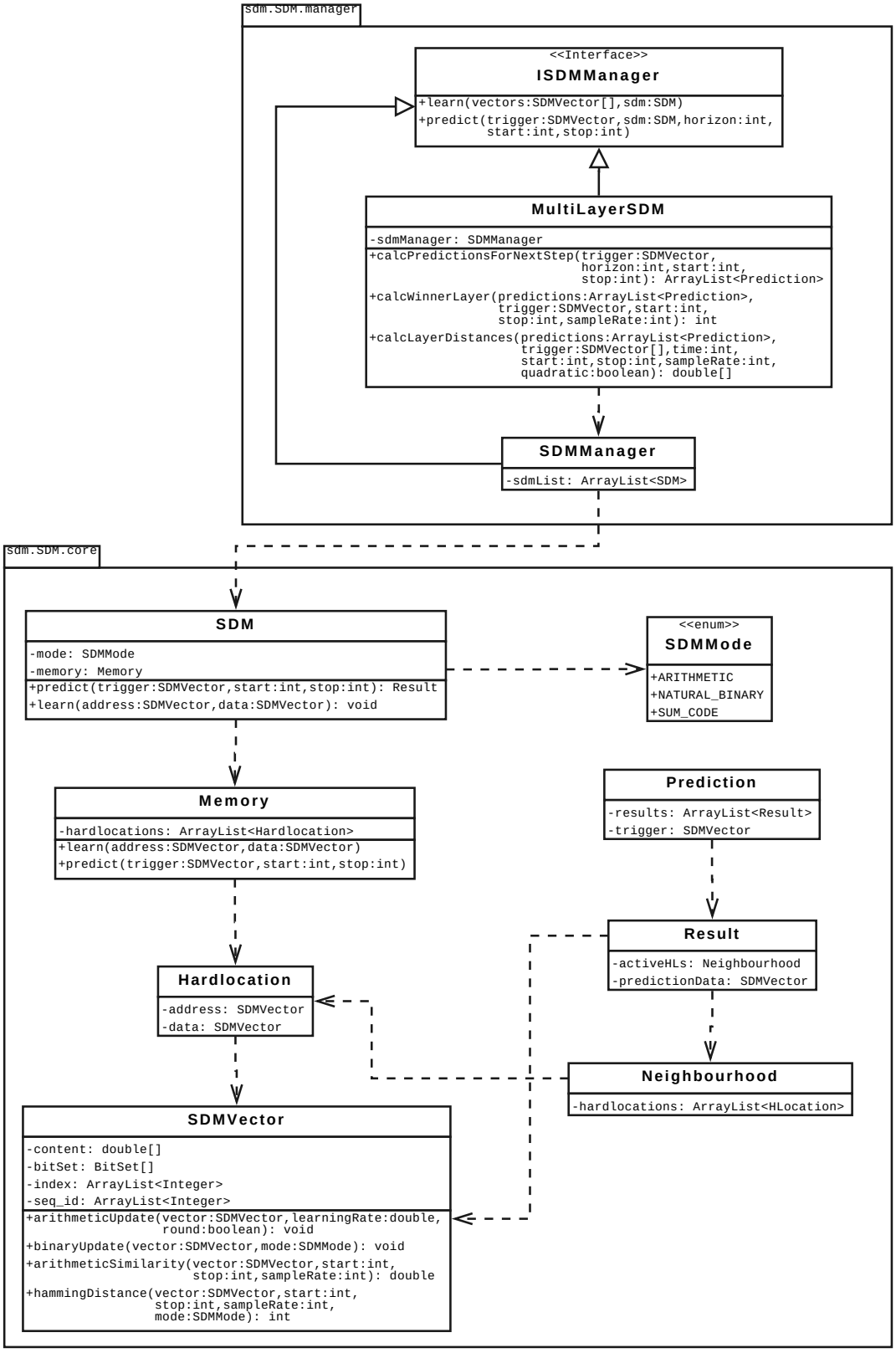


Figure 7.4.: A UML diagram of the multi-SDM architecture that illustrates the main relations and functionalities.

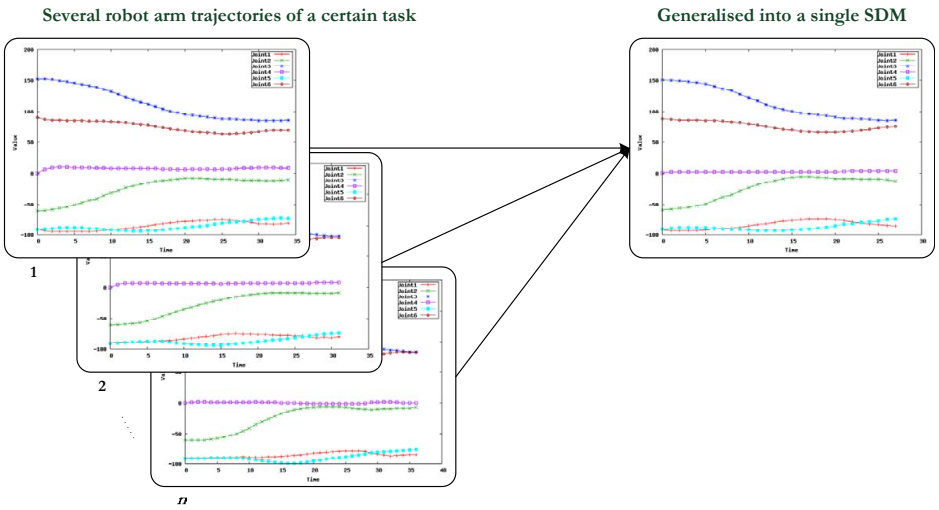


Figure 7.5.: Miscellaneous trajectories from several telemanipulation executions to solve a certain task are learnt into a single sparse distributed memory (SDM).

interface in diverse fashions to move the robot arm from one to another position. Further, the inverse kinematics is computed according to the given situations and thus contributes to the diversity of the resulting trajectories.

Figure 7.5 shows how different action sequences are generalised into a single SDM. If several executions are stored to a single SDM the information stored in the hard locations express a kind of mean execution of a certain task.

7.4.3. Abstraction

Humans often recall a datum that represents a group of experiences, but not exactly any specific instance. They produce a mental representation that is a composite of all those instances belonging to a group. Using a single datum, a representative prototype, to represent a large group vastly boosts the human intellectual ability. Consider how inefficient it would be to recall every trajectory you have ever executed to solve a certain task. The multi-SDM architecture provides the ability to produce a representative prototype for various action sequences executed by a single or many operators.

7.4.4. Predicting User Intention

This section will give some more details about the algorithms used to identify the intention of a human operator. During a training phase several (or at least few) trajectories are stored in the particular SDM that corresponds to the processed task. Each SDM is titled with the corresponding task name. After training, the multi-SDM architecture can be triggered with a yet unknown and interactive trajectory and will predict the most probable task from its experience. The whole process is illustrated by Algorithm 2.

The multitude of SDM instances are maintained and governed by an *SDM Manager* that handles the distribution of a trigger trajectory to all SDM instances. A *multi-SDM prediction* is done by continuously presenting the arm configuration during the execution of an action

Algorithm 2 MultiSDM Prediction

$h \leftarrow$ horizon
$multiResultList \leftarrow$ list of lists of all prediction results (contains entire prediction-history)
{**SDMManager Prediction**}
$startAddress \leftarrow$ current measurement
for $i = 0$ to number of SDMs **do**
  $trigger \leftarrow startAddress$
  $sdmResultList \leftarrow ()$ {list of prediction results}
  {**SDM Predictions**}
  for $j = 0$ to $horizon$ **do**
    Active hard locations $X'_A \leftarrow$ hard locations x' within the activation radius r centred by $trigger$ within the i-th SDM
    $SDMPrediction \leftarrow$ zero initialised vector
    for $k = 0$ to size of content vectors in the SDM **do**
      for $l = 0$ to cardinality of X'_A **do**
        $SDMPrediction[k] += X'_A[l][k]/|X'_A|$
      end for
    end for
    $trigger \leftarrow SDMPrediction$
    Append $SDMPrediction$ to $sdmResultList$
  end for
  Append $sdmResultList$ to $multiResultList[i]$
end for
{**Computing Square Error**}
$distancesList \leftarrow$ list of list of distances between predicted and observed trajectory
$observedTrajectory \leftarrow$ list of all observations (contains entire observation-history)
for $i = 0$ to the length of multiResultList **do**
  for $j = 0$ to the length of multiResultList[i] **do**
    $distancesList[i][j] = (observedTrajectory[j] - multiResultList[i][j])^2$
  end for
end for
{**Classification**}
$sumList \leftarrow$ list of aggregated distances
$bestSum \leftarrow$ Max-Value
$classification \leftarrow$ null
for $i = 0$ to the length of $distancesList$ **do**
  $currentSum \leftarrow 0$
  $weight \leftarrow 0$
  for $j = 0$ to the length of $distancesList[i]$ **do**
    if $j \pmod h == 0$ **then**
      $weight++$
    end if
    $currentSum += weight \cdot distances[i][j]$
  end for
  if $currentSum < bestSum$ **then**
    $bestSum \leftarrow currentSum$
    $classification \leftarrow i$
  end if
end for

to the memory module—to the SDM Manager to be correct. The SDM Manager provides each of the l SDMs consecutive snapshots of arm configurations of the yet unknown motion trajectory $\mathbf{T}_{curr}$ as input vector (cf. line 4 of Algorithm 2). This input pattern is used to query a prediction for the next few steps of the trajectory, the so-called horizon h (lines 7-12). Each SDM will commit the resulting, partial prediction of the next h configurations back to the SDM Manager, such that it assambles them to a single result list consisting of l predictions of length h (line 16).

Line 24 computes the square distances between the trigger trajectory and the l SDM predictions for a given interval $[t, t + h]$. The current implementation of the multi-SDM architecture uses an offline classification mechanism. Thus, the horizon h is defined in terms of the percentage of a trigger trajectory, whose length is known. This is due to handle sequences of different length[4].

The learnt task that best resembles the current trajectory $\mathbf{T}_{curr}$ is given by that SDM for which the sum of the quadratic distances is minimise over time. To determine the winning SDM, we include the temporal context of a trajectory in a weighting function. In that weighting function, the influence of previous predictions decreases linearly with their distance in time (cf. line 37).

Contrary to the offline classification as shown above, the online classification (Algorithm 3) will directly process the data provided by the PHANTOM® Desktop™ after translating it into the kinematic chain of the robot arm. At discrete time steps, the multi-SDM is triggered with an input vector containing the configuration of the PHANTOM® Desktop™. The data is used to predict the next assumed movement as in the offline case. The results of all SDMs are compared, and the particular SDM that exhibits the smallest square distance to the haptic data list, is assumed to be the intended goal of the current action.

Algorithm 3 Online Classification

```
1:  h ← horizon
2:  c ← 0
3:  hapticDataList ← ()
4:  predictionList ← temporary prediction
5:  currentClassification ← null
6:  repeat
7:     hapticData ← fetchDataFromPhantomDevice()
8:     Add hapticData to hapticDataList
9:     if c mod h is 0 then
10:       Compute a MultiSDM prediction with hapticData as trigger
11:       Add this result to predictionList
12:    end if
13:    Compute the winning SDM {namely the SDM with the overall least quadratic distance to
       hapticDataList}
14:    Store this result to currentClassification
15:    c ← c + 1
16: until termination
```

Let us consider the underlying telemanipulation system as described in Section 4.2.3 and depicted by Figure 4.10. This last issue of triggering a classification with either input patterns provided by the robot arm or the PHANTOM® Desktop™ makes clear that it is irrelevant

[4]Since the sampling rate of arm trajectories during task execution follows time constraints, memorised sequences of particular tasks can differ in their length.

whether the multi-SDM is implemented on the master or slave side of the telemanipulation system. This statement holds as long as the type of the input patterns are the same as during learning. Since the telemanipulation client (cf. Section 4.2.3) transfers the joint constellation of the PHANTOM® Desktop^{TM} into the kinematic chain of the robot arm before sending it to the robot's server side, an ambilateral linkage to the multi-SDM architecture becomes possible. Figure 7.6 exemplifies the possible connection points for an integration into the telemanipulation system presented in Section 4.2.3.

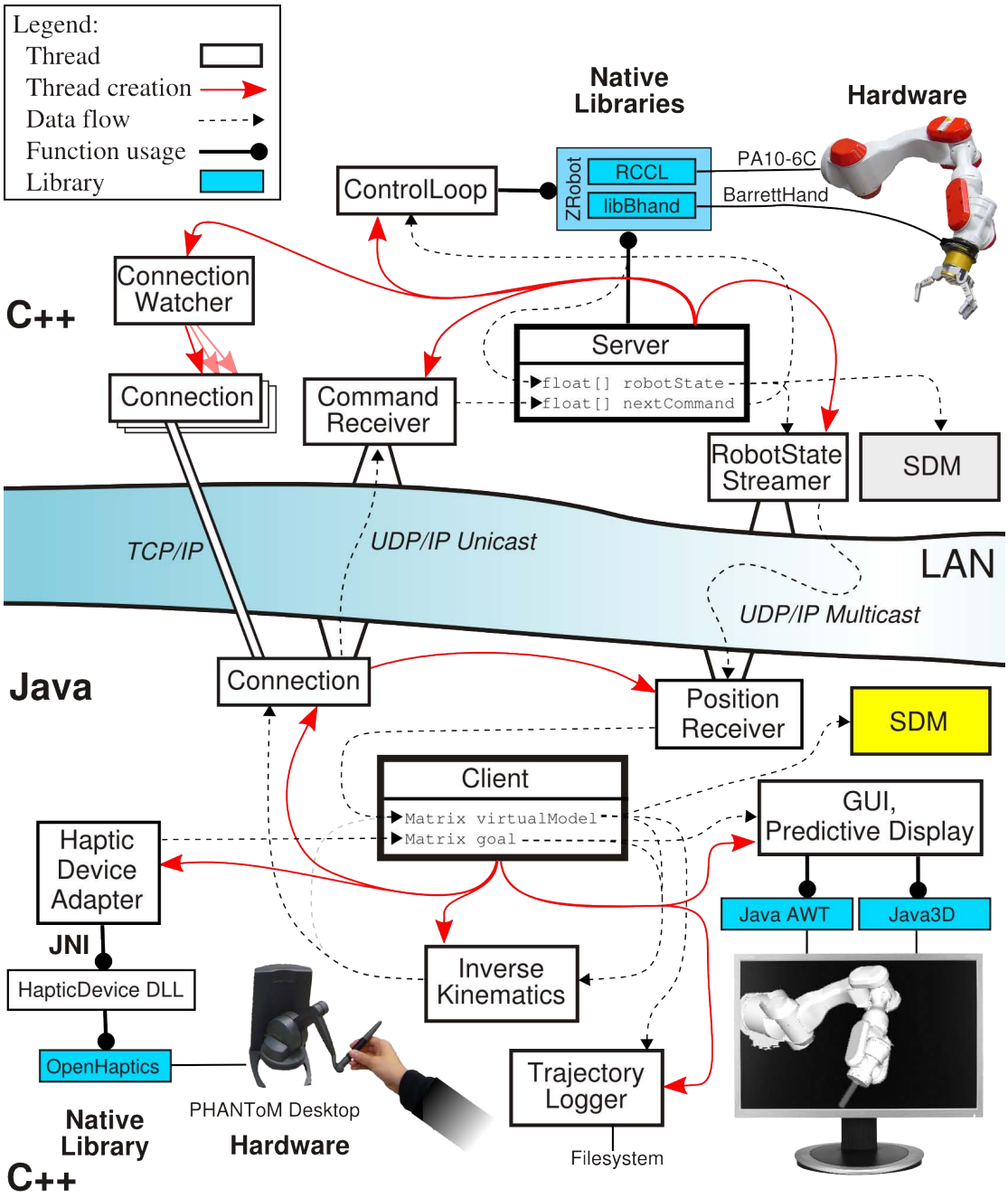
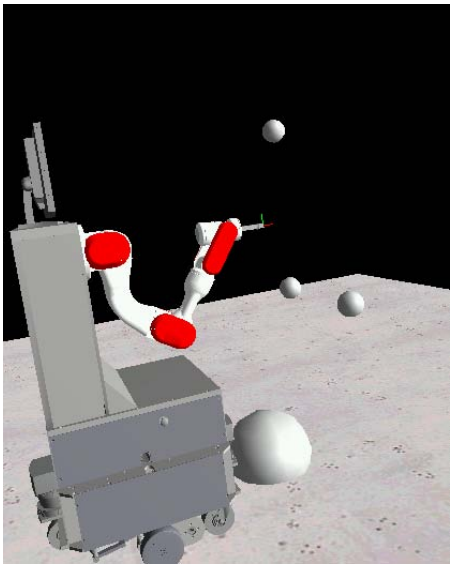
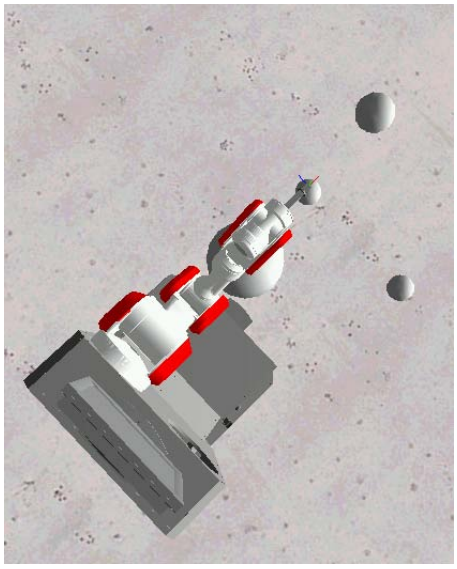


Figure 7.6.: Structure of the telemanipulation systems with the server application (top) and client application (bottom). The SDM (right) can either be connected to the client or the server side.



(a) Side view.



(b) Top view.

Figure 7.7.: Virtual dynamic views of the robot provided to the subjects while performing the remotely operational tasks. The tasks are to touch the spheres from a certain, given side.

7.5. Experiments and Results

Five male subjects had to solve 5 different tasks and thus act as a set of individual trainers for the SDM system. The subjects had to use the PHANTOM® Desktop™ to move a virtual MHI PA10-6C[5] to touch different virtual objects with the tip of a tool mounted at the end of the robot arm. Only the orientation how to touch the object, e.g. from the front, from the side or from the top of the object was given to the subjects. The decision on how to move the robot arm to reach the object was ceded to the subjects.

Figure 7.7 shows four virtual objects that constitute the different tasks. The object at the height of the robot's head and both objects at the height of a table must be touched from the front. The object on the floor must be touched at its top. The fifth task is to touch the left object at the height of the table also from the left side. Involving a single object in two tasks is used to test the classification accuracy of the multi-SDM when both final end-effector positions are closely related.

Due to the gradient descent optimisation used for the inverse kinematics[6] and the many degrees of freedom, hardly any user ever uses exactly the same trajectory twice to solve a certain task. Additionally, the operational range of the PHANTOM® Desktop™ is scaled

[5] For security reasons and to avoid damage to the physical robot arm the experiments were conducted on a virtual robot model of TASER. As shown earlier in this chapter, it makes no difference if the SDM module is linked to the server or the client side of the telemanipulation systems. Merely it must operate in the same joint space, the teleoperation system does right before sending the control command to the slave robot.

[6] The inverse kinematics for the robot arm is computed by the master client that provides the virtual robot model (cf. Section 4.2.3). That is the reason why it does not matter whether the SDM is linked to the server or client side.

Table 7.1.: The number of recorded sequences per task and subject. The total number of 200 sequences represents the test set for the forthcoming multi-SDM experiments.

		Runs per subject					
		skilled	unskilled				Sequences
Task descrition	Abbreviation	A	B	C	D	E	per task
Touch object at the height of the robot's head at its front	Top object	20	5	5	5	5	40
Touch left object at the height of a table from its front	Middle object 1 a	20	5	5	5	5	40
Touch left object at the height of a table from its left side	Middle object 1 b	20	5	5	5	5	40
Touch right object at the height of a table from its front	Middle object 2	20	5	5	5	5	40
Touch object lying on the floor at its top	Bottom object	20	5	5	5	5	40
		Sequences in total					200

with regard to the robot's manipulation space and thus responds very sensitively to any movement of the stylus which makes it impossible to exactly repeat any trajectory.

The subjects can be grouped into two skill levels: experienced and inexperienced, hereafter also addressed as skilled and unskilled. The experienced user is the designer of the telemanipulation system and gathered a lot of of experience on how to use the PHANTOM® Desktop™ and how to avoid critical arm configurations[7]. The four remaining test subjects have not used the telemanipulation system nor have they ever used a PHANTOM® Desktop™ interface before. On account of this, the inexperienced users got some trials to practice the different tasks. The subjects just needed one to three trails to feel confident with the handling of the interface. That indicates the ease of use and the natural way for giving demonstrations to a robot system as mentioned in the beginning of this chapter. Five executions per task were recorded from each of the unskilled users while the skilled user recorded twenty executions per task. The different tasks and the corresponding number of recorded sequences are summarised in Table 7.1. The resulting 200 action sequences represent the data set used for the forthcoming multi-SDM experiments.

The multi-SDM experiments—with regard to learning by demonstration—are subdivided into five different experiments according to the teacher–user relation. The term teacher is used for all those individuals, that contribute trajectories to the multi-SDM learning phase. The term user concerns those people whose intention should be classified by the multi-SDM system based on the data that has been taught by the teacher(s). This division is made to appropriately judge different aspects of the multi-SDM system for the purpose of intention

[7]Some arm configurations are called critical when the gradient descent method showed problematic state change control.

detection.

During a learning phase, the SDM system is trained with a proper subset of motion sequences out of the set of recorded task sequences. During an evaluation phase, all remaining sequences are presented to the memory successively. The multi-SDM has to classify the task the particular sequence belongs to, and thus identify the user's intention behind the execution of such a trajectory. The training and evaluation subsets are mutually exclusive.

7.5.1. Experiment A: Skilled Teacher–Unskilled Users

All task executions of the experienced user are considered as training set for the multi-SDM system. The robot arm motion trajectories of the four remaining, inexperienced users are used as a test set. This yield a training-to-test ration of 100:100 trajectories with a well-balanced training of the various tasks. The teacher–user relation can be described as logical `XOR`. This experiment studies how well the system classifies unknown trajectories when taught by a single expert.

7.5.2. Experiment B: Skilled Teacher–Skilled User

A randomly chosen subset, 75%, of task executions of the experienced user are considered as training set for the multi-SDM system. The remaining 25% of the set of robot arm motion trajectories from the same, experienced user are applied as a test set. The motion trajectories provided by the inexperienced users are neglected within this experiment. The training-to-test ration is about 75:25 trajectories. The teacher–user relation can be described as logical `AND`. This experiments studies the ability of the multi-SDM system to predict the intention of a user when he initially provides some training.

7.5.3. Experiment C: Unskilled Teachers–Unskilled Users

A randomly chosen subset, 60%, of the task executions provided by the four inexperienced users are considered as training set for the multi-SDM system. For this purpose, three of the five executions per task of each inexperienced user are randomly chosen to gain a balanced training of each SDM instance. The remaining 40% of the set of trajectories from the inexperienced user are applied as a test set. The motion trajectories of the experienced user are not considered in this experiment. The training-to-test ration is about 60:40 trajectories. The teacher–user relation can be described as logical `AND`. The purpose of this experiment is to study the systems predictive behaviour when undergoing a community-based training without any experts.

7.5.4. Experiment D: Unskilled Teachers–Skilled User

All task executions of the four inexperienced users are considered as training set for the SDM system. The robot arm motion trajectories of the remaining, experienced user are used as a test set. This yields a training-to-test ration of 100:100 trajectories. The teacher–user relation can be described as logical `XOR`. This experiment studies how well the intention of an expert, who may have specialised task solving strategies, can be predicted when the system is trained by a community of laymen.

7.5.5. Experiment E: Unskilled Teacher–Unskilled Users (Mutually Exclusive)

In this experiment, four evaluations are made where each inexperienced user is considered as trainer for the SDM system separately. The respective unskilled teacher does not contribute any trajectories to the test set. Several robot arm motion trajectories of the remaining inexperienced users are used as a test set. Each user, except for the trainer, contributes 2 trajectories per task to the training set. This yields a training-to-test ration of 25:30 manipulation trajectories per run. The difference to the above-mentioned experiment is that the trainer–user relation is mutually exclusive and thus can be described as logical `XOR` rather than a logical `AND`. Accordingly, each SDM instance in the multi-SDM just learns 5 trajectories per task.

The disadvantage of experiment C is that at least a few trajectories of each subject are considered both during learning and testing. Consequently, peculiar sequences that may be caused by a single user influence the SDM learning and may facilitate the recognition of similar peculiar action sequences later on.

This experiment considers the goal to test the SDM ability for classifying unknown motion trajectories just based on a small amount of learnt experience. Furthermore, the learnt data stems from a single person chosen randomly from a community of non-experts, but still might act as teacher to gain appropriate classification behaviour. This is like in the real world, where people also learn from each individual and not only from experts. This is particularly related to the one-shot learning mechanism which denotes a fundamental characteristic of human memory. Even if not a single shot is used here, five trajectory excitations per task consitute a rather small training set compared to the amount of training data needed in common classification algorithms.

7.5.6. Evaluation

All diagrams in Figures 7.8, 7.9, 7.10 show the relative error of each SDM that contributes to the prediction process. The trigger trajectory is portioned into several intervals q of length h. An SDM is the winner of a multi-SDM prediction, and thus defines the task membership after q intervals, if the error Θ is lower than the error of the predictions provided by all remaining SDMs at that particular time step (see Equation 7.1):

$$\Theta_{q,SDM} = \sum_{i=1}^{q} \sum_{t=(i-1)*h}^{i*h} i * (p_t - a_t)^2 \tag{7.1}$$

for each SDM $\in \{1, \ldots, l\}$ where p_t is the prediction at a particular timepoint t provided by the SDM and a_t the value of the trigger trajectory at timepoint t. The factor i weights the influence of the respective intervals such that younger ones have a more pronounced effect on the error computation.

A trajectory is defined as correctly classified after q intervals, if and only if the winning SDM represents the task the user currently solves. This is indicated by the particular title of the recorded sequence and the title of the arbitrary SDM instance. The quality of a particular task classification is judged by the moment when the respective SDM classifies the intended task correctly for the rest of the trigger sequence. Thus, the duration until correct classification is defined by the above-mentioned time point. Figure 7.8 illustrates this scheme and classifies the trigger trajectory to task bottom object, denoted by —●—.

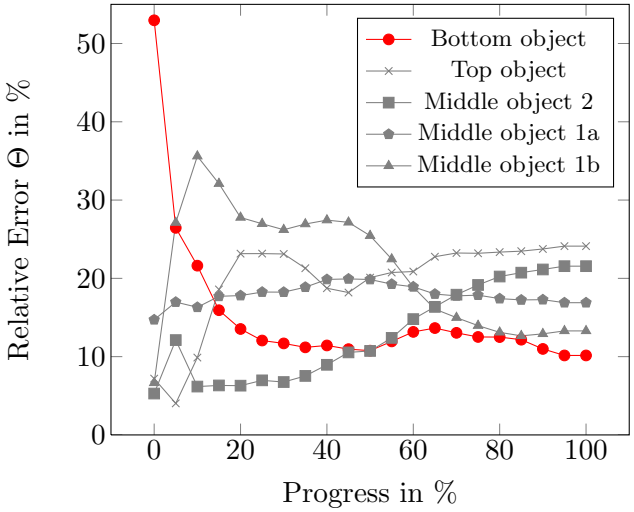
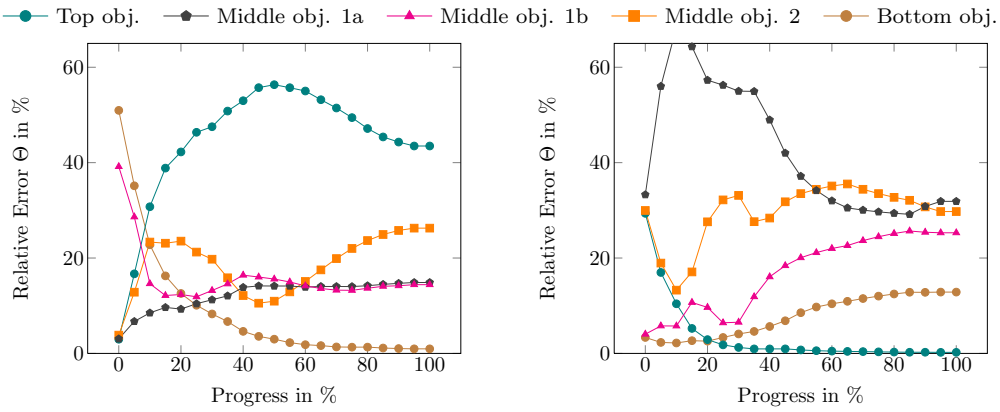


Figure 7.8.: Illustrating a multi-SDM discrimination between five tasks with the winning prediction denoted by —●—. The trajectory is classified correctly after presenting 55% of the trigger trajectory to the multi-SDM (the lower the better).

Figure 7.9 shows several correct predictions that result from triggering the trained multi-SDM architecture with a yet unknown trajectory. The subfigures show a digest of predictions from Experiments A-E (Sections 7.5.1-7.5.5). Please note that the subfigure captions give more detailed information about the corresponding teacher–user skill levels, the trigger trajectory and the duration till classification. The interested reader is referred to Appendix C for a more comprehensive version of the multi-SDM prediction diagrams. Some diagrams show quite fast and/or conspicuous classifications (see Figures 7.9(b), 7.9(c)). The winning SDM just contributes to the relative error Θ marginally. The confusions in nearly every diagram in the beginning 10%-20% of the trigger trajectory result from the fact that each tasks starts with the same robot arm default configuration. Thus, the first few steps of an arbitrary task execution are somehow similar and hard to distinguish.

Figure 7.10 illustrates some of the misclassified trajectories and such trajectories that just scrape a correct classification. Figure 7.10(a), for instance, exemplifies a multi-SDM prediction where the trigger trajectory is categorised equally to two task classes for a longer period. But finally, the trajectory is correctly classified after 80%, although with a narrow margin. Figures 7.10(b)-7.10(d) show misclassification of the trigger trajectory. In the case of misclassification, it can mostly be seen that at least the correct task is outclassed marginally. All multi-SDM classification diagrams that constitute the data set for evaluation are presented in Chapter C.

Table 7.2 summarises the success rates for correct classifications of interactive manipulation trajectories. In the case of Experiment B (the S–S cell of Table 7.2), where the SDM was trained with 50% of the trajectories produced by the skilled user, all of the remaining 100 test trajectories are correctly classified. The S–S group of the box-and-whisker diagram shown in Figure 7.11 depicts the distribution of the duration needed until correct task classification. The mean duration for a task assignment is about 15% of a trigger trajectory. 75% of the trajectories are assigned correctly when 25% percent of the trigger trajectory has been presented.



(a) Teacher–User setup: S–S. Trigger: Bottom object (7). Classification: 25%.

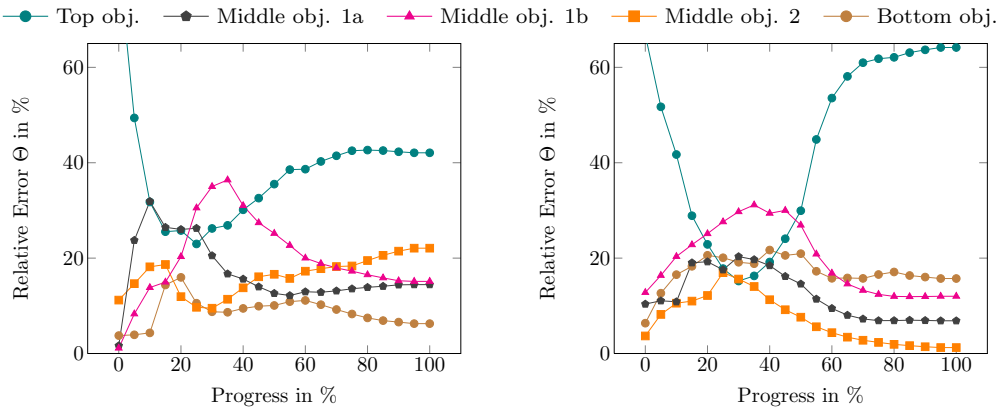
(b) Teacher–User setup: S–S. Trigger: Top object (7). Classification: 20%.

(c) Teacher–User setup: S–S. Trigger: Bottom object (16). Classification: immediately.

(d) Teacher–User setup: S–U. Trigger: Middle object 2 (7). Classification: 30%

(e) Teacher–User setup: S–U. Trigger: Middle object 1b (5). Classification: 25%.

(f) Teacher–User setup: S–U. Trigger: Bottom object (4). Classification: 65%



(g) Teacher–User setup: U–S. Trigger: Bottom object (6). Classification: 30%

(h) Teacher–User setup: U–S. Trigger: Middle object 2 (11). Classification: 30%

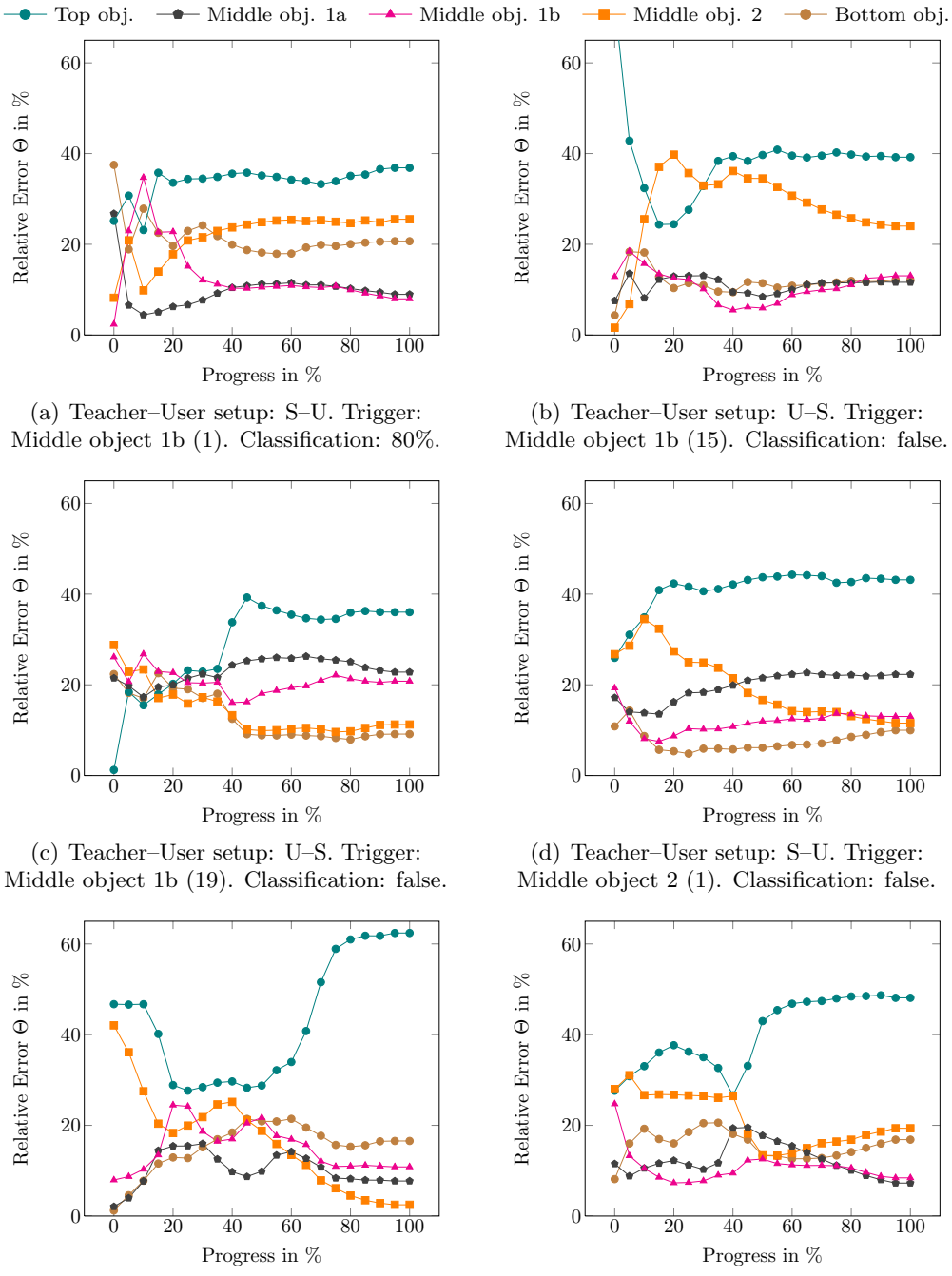
(i) Teacher–User setup: U–S. Trigger: Middle object 1a (15). Classification: 25%

(j) Teacher–User setup: U–U. Trigger: Top object (2). Classification: 50%

(k) Teacher–User setup: U–U. Trigger: Middle object 2 (5). Classification: 45%

(l) Teacher–User setup: U–U. Trigger: Middle object 1a (5). Classification: 20%

Figure 7.9.: Excerpt of correct multi-SDM predictions (the lower the better). Each subfigure gives some more information on the teacher–user category, the task membership of the trigger and the duration until successful classification.



(a) Teacher–User setup: S–U. Trigger: Middle object 1b (1). Classification: 80%.

(b) Teacher–User setup: U–S. Trigger: Middle object 1b (15). Classification: false.

(c) Teacher–User setup: U–S. Trigger: Middle object 1b (19). Classification: false.

(d) Teacher–User setup: S–U. Trigger: Middle object 2 (1). Classification: false.

(e) Teacher–User setup: U–U. Trigger: Middle object 2 (4). Classification: 60%.

(f) Teacher–User setup: S–U. Trigger: Middle object 1a (4). Classification: 80%.

Figure 7.10.: Excerpt of late and false multi-SDM predictions. Each subfigure gives some more information on the teacher–user category, the task membership of the trigger and the duration until successful classification.

Table 7.2.: Success rates of correct task classification when presenting unknown motion trajectories to the multi-SDM (according to Experiments A–E, Sections 7.5.1–7.5.5). The skill levels of teachers and users are denoted with S (skilled) and U (unskilled). The percentage in front of the slash represents the classification success rate of Experiment D and Experiment E respectively.

Teacher \ User	S	U
S	100%	93%
U	90%	97,50%/80%

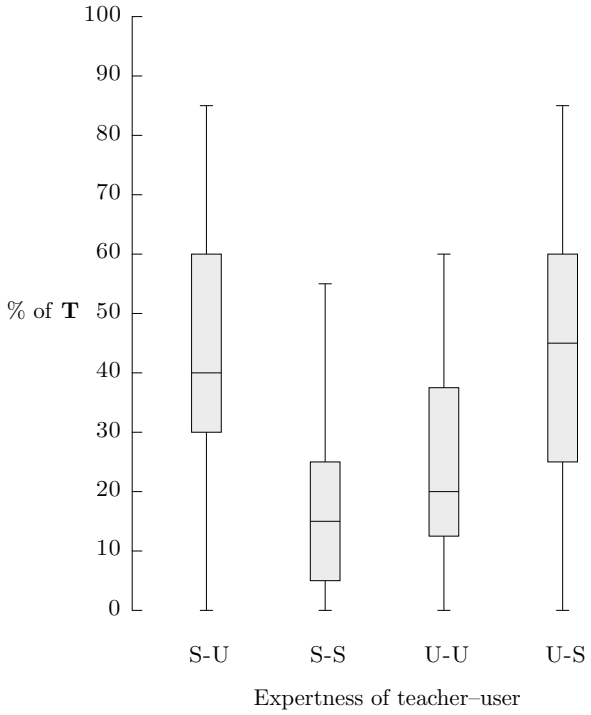


Figure 7.11.: Relative duration until correct classification of arm trajectories by the multi-SDM architecture. The letters describe the teacher(s)–user(s) skill levels (skilled–unskilled). The middle line within the boxes indicates the median. The upper and lower quartile (boxes around median) denote the highest and lowest 25% of the recognition duration. The sample maximum and minimum whiskers show the largest and smallest observed duration for correct classification. Misclassifications are not considered in this figure!

According to Experiment A (Section 7.5.1), where the skilled user acts as teacher and the memory is tested against all motion trajectories of the unskilled users, 93% of the motion sequences are classified correctly. In turn, when all unskilled users act as teachers and the memory is tested with the data of the skilled user (Experiment D, Section 7.5.4), 90% of the motion sequences are classified correctly. Of particular interest in these two experiments (S–U, U–S) is that the trajectory test set originates from a yet unknown user. The multi-SDM system has no prior information on how the unknown user might solve the tasks and just assumes his intentions by the generalised information taught by the other users. The box-and-whisker diagram (Figure 7.11) shows the mean duration for the task assignment, too.

Interestingly, 97,5% of the test trajectories are correctly classified, according to Experiment C, when a bunch of unskilled people act as trainer. Additionally, the U–U box of Figure 7.11 proves that the duration until correct task classification is comparable to the duration when the multi-SDM has to classify the motion trajectory of one and the same person (S–S). Contrary to learning trajectories of a single teacher, the U–U experiment learns and generalises trajectories provide by many users.

According to Experiment E (Section 7.5.5), an arbitrary unskilled user was randomly chosen to act as exclusive teacher. All of its five trajectories per task have been taught to the multi-SDM. In the test phase the memory was triggered with all trajectories of the remaining users. This experiment consists of four runs such that each of the unskilled users once acts as exclusive teacher. The reason for making this series of experiments that depict a modified version of Experiment C and D are: to test the SDM's predictive capability when just trained with a few example trajectories (related to one-shot learning), to test the system's robustness when triggered with motion trajectories of skilled and unskilled persons and avoiding the logical AND teacher–user relation of Experiment C. Here, an unskilled person acts either as teacher or as user, but does not provide motion trajectories to either the training or test set. This series of experiments yields admirable results. Considering the small amount of training data, the rate of 80% of correct classifications is surprisingly high as shown in Table 7.2. Table 7.3 illustrates that two of the tasks are always correctly classified even with such a small training set. Merely the middle object 2 task (Table 7.3(e)) shows a strongly low classification rate.

The box-and-whisker diagram in Figure 7.12 shows the mean recognition duration until correct classifications according to the arbitrary teacher. This figure, however, demonstrates which of the unskilled users is the best teacher for the multi-SDM system. We can infer that users with an early mean recognition rate produce the most universal or most generalised trajectory set that best matches the task solving strategy of the remaining users.

Table 7.3 is a more detailed version of Table 7.2 that outlines the classification accuracy of particular tasks. As seen in Table 7.3(a), classifying the motion trajectory at which a user virtually touches the object at the height of the robot's head always succeeds. This indicates that either the stored (memorised) generalised trajectory for this particular task is fundamentally different from those of the other tasks stored in the competing SDMs. It then would be easy to separate them from the other tasks. Alternatively, all users may apply very similar motion trajectories with regard to the stored one to solve that task. In our opinion the high success rate is caused by the fact that the task is fairly different to all other tasks.

Touching objects that lie on or close to the $x - z - plane$ of the robot's shoulder joint caused users to use quite different motion strategies. Some moved the robot's elbow up first (comparable to an excavator boom) and solved the task in such a way, others preferred to

Table 7.3.: Success rates for classifying particular tasks with regard to the skill levels of teachers and users. In case of unskilled teacher and unskilled user (UU) the first number depicts the success rate with a mixture of unskilled teachers, while the second number shows the success rate when the multi-SDM is trained by an individual, single unskilled teacher.

(a) Task: Top object.

Teacher \ User	S	U
S	100%	100%
U	100%	100%/100%

(b) Task: Bottom object.

Teacher \ User	S	U
S	100%	100%
U	100%	100%/79,17%

(c) Task: Middle object 1a.

Teacher \ User	S	U
S	100%	95%
U	100%	100%/100%

(d) Task: Middle object 1b.

Teacher \ User	S	U
S	100%	95%
U	60%	87,50%/75%

(e) Task: Middle object 2.

Teacher \ User	S	U
S	100%	75%
U	90%	100%/45,83%

Table 7.4.: Success rates ordered by tasks. The second column summarises the success rates for Experiments A,B,C,D while the third column summarises for Experiments A,B,D,E.

Task	Individual unskilled teachers	Mixture of unskilled teachers
Top object	100,00%	100,00%
Middle object 1 a	99,29%	98,75%
Middle object 1 b	79,29%	85,63%
Middle object 2	64,05%	91,25%
Bottom object	88,10%	100,00%

keep the elbow down (like a swanlike neck) or oriented it to the left or right to accomplish the task.

Contrary, the left object on the height of a table caused users to either use an elbow up or elbow to the left strategy. This holds for the task of touching it from the front (Table 7.3(c)), while users preferred arm configurations where the elbow is oriented to the left if they should touch the same object from the left side (Table 7.3(d)).

From all of the misclassified middle object 1 b tasks (Table 7.3(d)), 87.5% were erroneously assigned to task middle object 1 a. Moreover, 12.5% have been erroneously assigned to the tasks bottom object or middle object equally pronounced, with 6.25%. The success rate is affected by the close relation to the middle object 1 a task. Most classification errors occur when unskilled users act as teachers and the system is tested against the trajectories of an expert. This proves that the skilled user developed his own specific strategy to solve particular tasks, such that the strategy does not really match the common approach. However, even if the task middle object (Table 7.3(d)) shows the worst predictability along all studied tasks, in the worst case the multi-SDM was still able to classify 60% of the test set correctly (U–S combination).

Interestingly, an inversion of the close relation argument above cannot be drawn for task middle object 1 a. Although many of the middle object 1 b tasks are misclassified as middle object 1 a, the classification of the latter trajectories still retains a high success rate across all teacher–user combinations (cf. Table 7.3(c)).

The differences between the presented success rates in Tables 7.3 and 7.4 are as follows. Table 7.4 summarises the classification accuracy comprising all teacher–user combination while Table 7.3 also considers the skill levels of different teacher–user combinations. Thus, the former table presents the overall recognition rate of the particular tasks in the scope of the presented experiments. What can be seen in Table 7.4 is that training the multi-SDM by a number of individual teachers results in a reduced predictability compared to using many different teachers. As a result, it can be concluded that a community-based teaching performs better even with many moderately skilled teachers than when being taught by a single expert.

7.5.7. Robustness to Noise

Due to the theory that predicts a low sensibility of the SDM concept to noisy inputs, some tests have been conducted to prove this feature. Table 7.5 shows the mean of ten trials

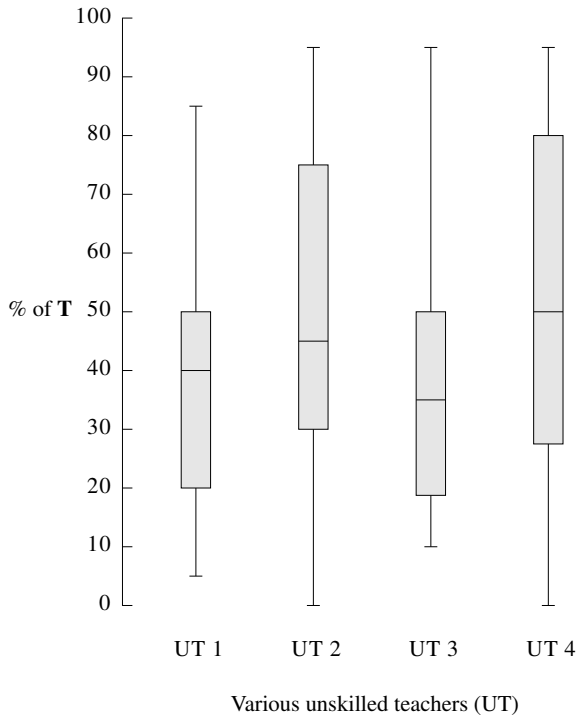


Figure 7.12.: Duration until correct task classification by the multi-SDM architecture with regard to the individual unskilled teacher. Misclassifications are not considered in this figure! The middle line within the boxes indicates the median. The upper and lower quartile (boxes around median) denote the highest and lowest 25% of the recognition duration. The sample maximum and minimum whiskers show the largest and smallest observed duration for correct classification.

Table 7.5.: Success rates if operated with a noisy trigger. Mean of ten trials according to the multi-SDM prediction of a particular sequence in which an unskilled user had to touch the object on the ground. Exactly $n\%$ of white noise is added to each component of the trigger vector.

Noise in %	Prediction success %	Error rate %
0	30,0	0
10	45,0	0
20	57,0	0
30	62,0	0
40	77,0	0
50	83,0	0
60	87,5	0
70	88,0	0
80	82,7	10
90	83,1	20
100	87,7	30

of classifying a certain trajectory of a certain user. Column prediction success shows at which progress level of the trajectory it is classified correctly as a member of a task. Each component of the input vector is modified with a variable percentage of white noise. The noisy version of the input vector is then used as trigger for the multi-SDM to detect the intention behind the particular action and to predict the next step of the trajectory. Table 7.5 shows that with 0% noise, the trajectory is classified correctly after presentation of 30% of the trigger. When the trigger vector contains 10% noise, the trajectory is correctly classified after 45%. When the trigger contains 80% noise, prediction errors occur for the first time. In this case, 10% of the runs used for evaluation (1 of 10 trajectories) could not be classified correctly. In turn, the remaining 9 trajectories have been classified correctly after 82,7%.

The important point of this test is that the predictability of a task remains stable until the input vector contains 80% of noise. First prediction errors occur with equal and more than 80% of noise. The more noise is added, the later the trajectory is classified correctly. Although the results will certainly differ for all remaining trajectories of the test set, this simple evaluation showed that the multi-SDM architecture is able to handle noise tremendously well. Experiments that consider effects caused by memory loss have already been presented in Section 5.2.2.

7.6. Discussion

The multi-SDM architecture exhibits very strong predictive capabilities in the recognition of an operator's intention during interactive guidance of a robot arm. The SDM concept provides a congenial mechanism to reduce the input address space onto a smaller subset. Especially in the case of applications with high-dimensional problems and address spaces with a huge number of possible states—such as in the learning by demonstration through telemanipulation domain. Current neural network architectures lack the amount of data

needed to train such a system appropriately. Even if the multi-SDM is trained with few training examples, it provides proper classification capabilities. Particularly its fundamental characteristic to deal with incomplete and noisy input patterns makes its usage favourable for robotic applications when dealing with various operators that act as teachers. Achieving this results with a single SDM would be hard or nearly impossible. The multi-SDM approach can be seen as a kind of specialisation that also happens in the brain.

The telemanipulation experiments confirm the hypothesis that SDMs can be used to dynamically classify high-dimensional robot sensor patterns to high-level actions and to act as a suitable memory mechanism for interactive demonstrations. Contrary to statistical approaches that need huge training sets to identify salient features of the input domain, the multi-SDM architecture provides a basic memory mechanism that allows for learning of much smaller training sets. The multi-SDM architecture establishes a linkage of low-level sensor perceptions to high-level symbolic concepts, which is commonly known as the *symbol grounding* problem.

Applications that may result from using the current or a slightly modified version of the multi-SDM architecture may primarily give rise to context-driven assistance systems. If the system recognises a certain intention of a user, it can provide purposeful assistance such as taking over the task execution, providing additional information with respect to the particular task and so forth.

The robot's memory of experience is extendable to new tasks easily. Such an extension, for instance, is to append another SDM to the multi-SDM when a new task is to be learnt. In the beginning, an additional SDM instance might be filled with just few demonstrations of a new task. If the system classifies a yet unknown trajectory to the newly learnt task, the current trigger trajectory can be used to train the particular SDM instance to form a more general representation of the task. The system does not have to be retrained completely to find another proper set of salient features that influence the system's implications. The current problem in doing this automatically is to appropriately determine a trajectory as a "new", yet not learnt task. Due to the associative nature of the SDM, it will always provide a prediction. This might be accomplished through a decision function to gain the technical system with a proper decision capability of judging tasks as yet unknown and unlearnt. Furthermore, especially the end of a manipulation sequence should receive special attention since the fine positioning and fine motor skills can be considered more important than the rough approaching procedure[8]. An initial idea to determine about a supplementary SDM is as follows:

$$addSDM = \begin{cases} true & \text{if} \quad (\theta_i * \delta_i) \leq \xi, \\ false & \text{otherwise} \end{cases} \tag{7.2}$$

where θ is the mean error contribution of SDM $i \in \{1, .., l\}$ over the whole sequence and δ depicts the tendency of the error contribution over time. Finally these parameters should not cross a certain threshold ξ. When parameters such as relative error are involved in the decision of additional SDMs, a threshold ξ has to be dynamically determined because of its dependency on the number of participating SDMs. However, implementing an appropriate decision function remains a challenging problem.

[8] Gallese and Lakoff (2005) propose to structure a *grasp* schema into role, phase, manner and value parameters. The phases consist of an initial condition defining the object's location, an approaching phase, and a central phase at which the particular object is grasped with a gripper.

Another useful study is to not train the multi-SDM architecture with complete manipulation tasks but rather with sets of motion primitives. Various approaches for deriving motor primitives and atomic actions are proposed by Jenkins and Mataric (2004); Williams et al. (2006); Wu (2008); Weser (2009). These should be represented by separate SDM instances, e.g. lifting the elbow up, stretch the forearm, lifting an object, fine positioning *et cetera*. The advantage of such a trajectory decomposition is that a multi-SDM can be used to identify the chronological order of primitive action elements that constitute a certain high-level action. Additional layers can subsume the identified motion primitives into high-level action snippets and so forth, which would lead to a hierarchical architecture. Also, reasoning on the level of motion primitives will become possible.

The multi-SDM model can be integrated within a comprehensive perception system of a robot, consisting of subsymbolic and symbolic layers. Features of an arbitrary input, so-called microfeatures, can be extracted within a low-level sensoric layer. The state of the world at a particular point in time, described by such microfeatures, constitutes the SDM address vector. Triggering the multi-SDM with such an address vector results in a classification of input data into symbols represented in a high-level semantic layer. If the sensors are read periodically, the multi-SDM is able to classify sequential data into symbols represented in a semantic layer. The system might be able to distinguish between approaching an object and manipulating an object. If so, the system could, for instance, adjust the mapping of the PHANTOM® Desktop^{TM} operational range to a more fine-scaled positioning resolution of the robot's manipulation tool. Furthermore, currently perceived and classified sequences can be learnt online as a new experience to influence future behaviour of the multi-SDM.

8

Crossmodal Interactions in Sparse Distributed Memory

All thought is a feat of association; having what's in front of you bring up something in your mind that you almost didn't know you knew.
(Robert Frost, American poet, 1874–1963)

Contents

Natural cognitive systems benefit from combining different sensory inputs. Different modalities provide information and judgements about different aspects of the world and jointly encode particular aspects of events, e.g. the location or meaning of an event.

The CINACS project[1] investigates principles of crossmodal interactions in natural and artificial cognitive systems. As part of this project, this chapter describes how to use SDM to combine perceptions of different sensory systems into a multisensory representation. Furthermore, it studies advantages of multimodal perception compared to unimodal perceptions on the basis of a localisation task of a robot.

[1]Research training group on Crossmodal Interactions in Natural and Artificial Cognitive Systems: `http://www.cinacs.org`

8.1. Crossmodal Interactions

One of the most ubiquitous features of biological systems, from the simplest to the most complex, is an ability to combine and synthesise information from the different senses. This synthesis is highly adaptive in that it allows the creation of a "multisensory" view of the world, one in which information from the different senses can be compared, integrated, and evaluated to form an accurate and meaningful representation of the external world (Wallace, 2004). In everyday life, object perception benefits from the coordinated interplay of vision, audition, and touch. The different sensory modalities provide both complementary and redundant information about objects, which facilitates the improvement of recognition speed and accuracy in many circumstances (Amedi et al., 2005).

Most research on perception considers each sense in isolation, e.g. hearing, vision, touch, olfaction and so on. It approaches the senses as being entirely separate modules. However, in many situations the different senses receive information about the same external events and objects simultaneously. The brain combines these stimuli to create multimodally determined percepts of the external world (Driver and Spence, 2000).

In natural systems, the term *crossmodal integration*[2] describes that particular external properties often stimulate several senses simultaneously and that the information from multiple modalities is combined into a convergent percept (Driver and Noesselt, 2008). Examples for crossmodal integration are the McGurk effect (McGurk and MacDonald, 1976) and the ventriloquism effect (Bertelson, 1999). In the McGurk effect, for instance, a seen lip-movement can alter which phoneme is heard for a particular sound. McGurk and MacDonald (1976) reported that on being showing a film of a young woman's talking head, in which repeated utterances of the syllable "*ba*" had been dubbed on to lip movements for "*ga*", normal adults reported hearing "*da*". With the reverse dubbing process, a majority reported hearing "*baba*" or "*gaba*". In the ventriloquism effect, a seen lip-movement can influence the apparent location of speech sounds. These effects are called *crossmodal effects*.

In case of convergent and concurrent information provided by two ore more modalities, the biological system has to discriminate such cases from those where stimuli are entirely unrelated. Several authors proposed simple heuristics for crossmodal integration of information, most of which are spatio-temporal correlations. Stimuli that occur at the same or similar time and/or place tend to be treated as referring to the same external event (Driver and Spence, 2000). Another crossmodal interaction is that judgements of one modality can be influenced by a second modality even when the latter modality cannot provide any information about the judged property itself. The multisensory nature of our perceptions has several behavioural advantages—for example, quicker response and improved recognition in noisy environments (Newell, 2004).

Since the end of the 1990's, the use of sensor fusion—also known as multi-sensor fusion—has been growing in popularity in engineering to increase the reliability, accuracy, completeness and range of individual sensors by combining the resulting information of various sensors. Recent approaches to sensor fusion differ in their provided functionality, e.g. by providing complementary, concurrent, cooperative or independent information. The most popular techniques for fusing information of multiple sensory sources are: stochastics, classifiers, Kalman filter, fuzzy-logic, logic and rule-based algorithms.

[2] The terms *crossmodal* and *multimodal* are used interchangeably by researchers of different disciplines. Thus, both terms are used in this work interchangeably, too.

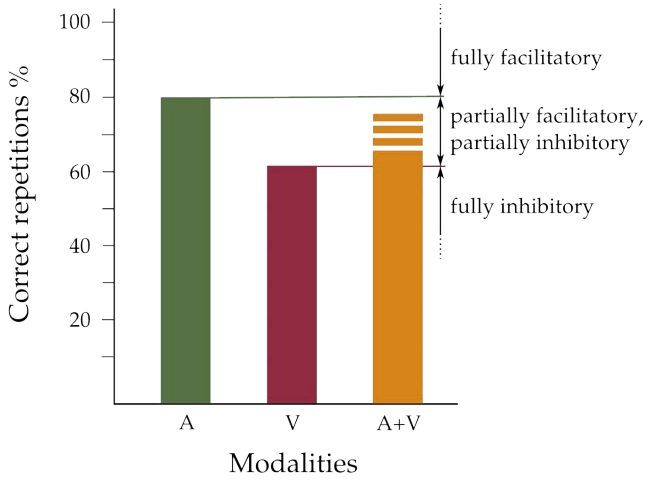


Figure 8.1.: A distinction of crossmodal (A+V) effects.

Human multimodal abilities give a framework for the development of artificial intelligent systems. Nevertheless, this does not necessarily mean that the underlying computational nature of a natural multimodal system must be the same as in an artificial system. It rather serves as an inspiration. Crossmodal interactions in robots is a very young field. One task of this thesis is to establish a robot memory that is useful for understanding crossmodal integration. Thus this chapter merely presents an approximation of natural intelligence according to crossmodal integration regarding the developed memory system but currently does not replicate the natural system. Multimodal perception in humans is still not understood enough to permit making a clear definition of the underlying mechanisms.

The question to be answered in this chapter is: Can a unified percept originating from various sensors improve the robustness and accuracy of memory-based room classification. To analyse this research question, the crossmodal effects are considered in more detail.

The term *crossmodal effects* comprises a set of alternative effects that may occur. Let us consider an exemplified repetition task to illustrate the different forms of such crossmodal effects. The task is divided into three subexperiments. In the first experiment a subject is confronted with a number of words that each are presented visually to the subject on a screen for several seconds consecutively. Afterwards, the subject is asked to recite the presented words. The second experiment consideres an auditory presentation of the words by reading them out loud, thus stimulating the subject's aural sense. Again, the subject is asked to repeat the words afterwards. In a third experiment the words are presented on a screen and are read out loud at the same time, thus stimulating the subject's visual and aural senses likewise. The number of correct word repetitions is shown in Figure 8.1 to illustrate the three types of possible crossmodal effects.

When the subject is just able to recite less words in the multimodal test than in the poorest of both unimodal cases, the crossmodal effect is defined as being *totally inhibitory*. If the subject is able to recite more words in the multimodal domain than the poorest unimodal one, but still less than in the best unimodal domain, this is referred to as *partially inhibitory* and similarly as *partially facilitatory effect*. When the results of the crossmodal recitation exceeds all accomplished results of the participating modalities, it is referred to as *fully facilitatory effect*, in psychology also termed *supramodal effect*.

8.2. Experiment: Crossmodal Robot Localisation

In order to study crossmodal interactions with an SDM, the robot was tested for a localisation task. The robot acquired data of several rooms in a real world office domain via laser range finders and an omnidirectional vision system. During a learning phase the robot stored the data of each room into a separate SDM of a multi-SDM architecture. Each SDM of the multi-SDM was labelled with a unique room identifier during a supervisory learning phase. Subsequently, a multi-SDM prediction was triggered with yet unlearnt data of an arbitrary room to establish a localisation based on subjective experience. Thus, the robot establishes a localisation based on the currently sensed situation and context. The experiment comprised three different sub-tests for analysing crossmodal influences. The tests are as follows:

Unimodal laser-based localisation: The robot learns data that is acquired through its laser range finders. Each SDM learns a vector that consists of a set of distance values. A multi-SDM is triggered with a yet unknown laser scan and predicts the most likely room it believes to be in.

Unimodal vision-based localisation: The robot learns data that is acquired through its omnidirectional vision system. Each SDM learns a vector that consists of a set of microfeatures, which are a number of pixels that result from edges in the input image. A multi-SDM is triggered with a yet unknown omnidirectional image and predicts the most likely room it believes to be in.

Crossmodal localisation: The robot learns data from both sensors, laser range finders and omnidirectional vision system. Each SDM learns a vector that consists of a set of distance values and a number of edge pixels. A multi-SDM is triggered with a yet unknown laser scan and image and predicts the most likely room it believes to be in.

While capturing the laser range and vision data at a given time, the robot also captures its orientation η from a virtual compass[3]. Its orientation η is used to determine the data sampling starting point to finally construct an appropriate input vector for the memory. These preprocessing steps that are used before creating an SDM input vector are detailed in the subsequent sections.

The office environment includes a long hallway, several offices, a kitchen, two laboratories and a workshop. Sensor data of the robot is recorded at 18 different locations inside the office domain. Table 8.1 summarises the number of acquired training samples per room. The positions of the bars in Figure 8.6 approximately illustrate the different locations for data capture. Several data acquisition runs have been made. The capture position and respective orientation of the robot varies among different runs and is hardly identical to any previous run. Since data acquisition was accomplished across different dates and times, the captured scenes may possess variations, e.g. changed lighting conditions, rearranged scenes and scene objects such as closed or opened doors, people present and so forth. It becomes obvious that the training set for SDM-based robot localisation includes differences even for one and the same room. Nevertheless, it is expected that the multi-SDM architecture is suited to deal with these variations and to successfully accomplish a robot localisation.

[3]Though the robot is not equipped with a magnetic compass its orientation is computed from the robot's navigation system.

Table 8.1.: The amount of training data per room.

Room	Training samples
Archive 1	6
Archive 2	6
Elevator	6
Hallway 1	6
Hallway 2	6
Hallway 3	3
Hallway 4	6
Hallway 5	3
Hallway 6	5
Kitchen	6
Lab 1	4
Lab 2	6
Office 1	3
Office 2	3
Office 3	6
Office 4	7
Office 5	4
Workshop	5

Note that these experiments are not used to process sequentially ordered sets of world states to predict any residual motion trajectory as described in previous chapters. These experiments are rather used to determine the robot's belief of where it currently is located within an office environment based on its past experience.

8.2.1. Laser Range Scans for SDM

The robot captures a 360° 2D laser scan of its environment via two SICK Laser Measurement Systems (LMS). Each scanner covers 180° of either the space ahead or behind the robot with an angular resolution of 0.5°. This yields a number of 722 distance values in total. Figure 8.2 shows a laser range scan in the laboratory.

In order to compensate for the robot's orientation during data acquisition, the conversion of the distance arrays provided by the laser scanners to an SDM input vector follows a certain algorithm. The first value of the SDM input vector is defined by the distance value of the range scan that corresponds to the robot's current orientation. Hence, the robot's orientation η is treated as an offset for the initial reading position of the laser range scan. The subsequent values are directly transferred to the SDM input vector. If the end of the distance array is reached, it continues from its beginning. The conversion finally ends with the distance element for $\eta - 0.5°$ and the corresponding last element of the input vector. In an SDM input vector such that:

$$x =< d_1, d_2, \ldots, d_i >, \tag{8.1}$$

each distance value $d_i,\ i \in \{1, \ldots, 722\}$ is expressed by a 64-bit double precision value. The

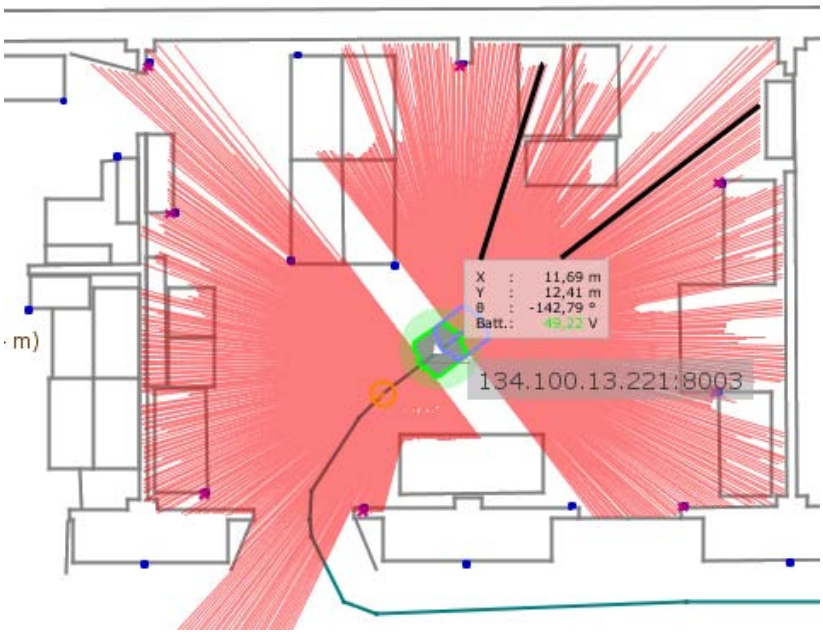


Figure 8.2.: A laser range scan of the robot standing in the laboratory. Each red line shows a laser beam and its distance of reflection.

resulting SDM vector thus has a size of $n = 46208$ bits. This vector is used for the first unimodal SDM-based localisation experiment. Such vectors are learnt by all SDMs of a multi-SDM architecture with respect to the particular room where they have been acquired. The trigger vectors for an arbitrary multi-SDM prediction are similar. Figure 8.3 shows another laser scan of the laboratory that illustrates the perceived distances.

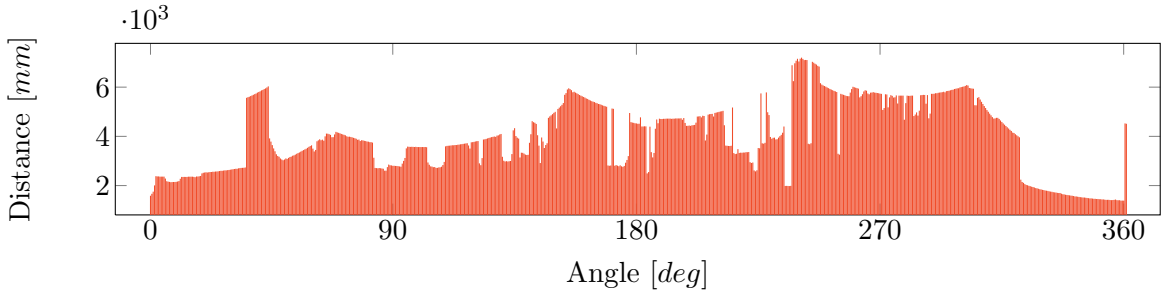


Figure 8.3.: The robot's laser range scan of the laboratory.

8.2.2. Omnidirectional Feature Images for SDM

Capturing image data with conventional cameras or a stereo system is impractical for comprehensive environment perception due to the dependency on the robot's viewing direction. To avoid this dependency, the robot captures images through an omnidirectional vision system (see Section 4.1.1.3). It thus achieves an omnidirectional view of the entire environment

around the robot. Nevertheless, the particular orientation of the robot yields a circular shift in the captured omni-image.

A circular band that contains the projections of the hyperboloid mirror is extracted from the source image. A resulting image of the robot's environment is shown in Figure 8.5(a). A filter that uses a convolution with a Gaussian function is applied for image smoothing. This is done to establish a low-pass filtering of the input image and to remove noise and high frequency parts of the image. The smoothed image is converted into an 8-bit grey image. Then, a Sobel edge detector is used to highlight sharp changes in image intensity. Figure 8.4 illustrates two 3×3 convolution kernels that are used to generate vertical and horizontal derivatives. The final edge image is produced by combining the two derivatives using the square root of the sum of the squares. Due to the low-pass filtering, only prominent edges will remain in the image. Figure 8.5(b) shows the resulting edge image which is the basis for further processing regarding the SDM-based learning.

1	2	1
0	0	0
-1	-2	-1

1	0	-1
2	0	-2
1	0	-1

Figure 8.4.: Two 3×3 convolution kernels of the Sobel operator.

To create the microfeatures for the SDM input vector, the edge image is divided into equally sized slices. Each slice is sampled from the centre of the image to the outer ring limited by r. A slice, denoted by s, is further limited by the angles ϕ and $\phi + \tau$, where τ defines the size of the slice. The number P of edge pixels within a slice s_i that exceed a certain *threshold* is given by:

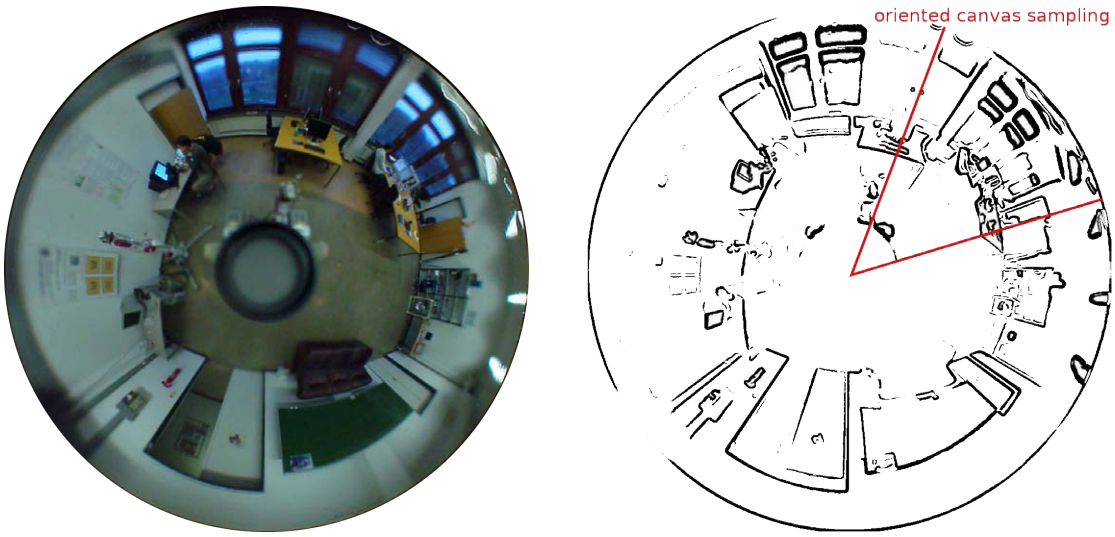
$$P_{s_i} = |\{(r, \omega) \ : \ (\phi \leq \omega < \phi + \tau) \wedge (grayValue(r, \omega) \leq threshold)\}| \tag{8.2}$$

The above-mentioned unimodal experiment uses a slice size of $\tau = 10°$. The number of edge pixels P in a particular slice s_i constitutes the corresponding SDM vector element and is presented as double precision value[4]. To compensate for the robot's orientation, which yields a circular shift of the captured omnidirectional image, the limit of the first slice is derived from the robot's orientation η during data acquisition. The first slice s_1 ranges from $\phi = \eta$ to $\phi = \eta + \tau$. The red lines in Figure 8.5(b) exemplify such a slice. The SDM input vector shown in Equation 8.3 follows from the above-mentioned construction process.

$$x =< P_1, P_2, \ldots, P_{\frac{360}{\tau}} >, \tag{8.3}$$

Such vectors of size $n = 2304$ bits are learnt by all SDMs of a multi-SDM architecture with respect to the particular room where they have been acquired. The trigger vectors for an arbitrary multi-SDM prediction are similar.

[4]Though just representing integers.



(a) Omnidirectional source image.

(b) Edge image, an one sector.

Figure 8.5.: Illustration of image-preprocessing applied to the omnidirectional views. A spherical image is cut-out from the source image (Figure 8.5(a)). Further steps are Gaussian blur, image conversion and edge detection.

8.2.3. Crossmodal Integration of Sensoric Percepts

The two above-mentioned sections detail how particular sensory-based percepts are preprocessed before storing them into an SDM. In order to study a crossmodal integration of both modalities, both SDM vectors presented in Equation 8.1 and 8.3 are combined to a single SDM vector:

$$x =< d_1, d_2, \ldots, d_i, P_1, P_2, \ldots, P_{s_j} >, \tag{8.4}$$

where each $d_i,\ i \in \{1, \ldots, 722\}$ represents the distance value sensed through the laser scanners, and the number of edge pixels P in the particular slices $s_j,\ j \in \{1, \ldots, \frac{360}{\tau}\}$ represent the microfeatures of an omnidirectional image.

Storing this kind of vector in the memory yields a unified multimodal representation of a particular state of the world, in this case rooms. This experiment comprises just two modalities but the vector can be extended with further modalities easily if necessary.

As mentioned in Chapter 5, the SDM implementation of this work is able to handle partial cues. This makes it possible to train a multi-SDM with the multimodal vectors presented in Equation 8.4 for all experiments. In the case of the unimodal experiments, the multi-SDM is triggered with a crossmodal vector where just the modality-specific vector portion contains any values. The memory will reply with the most similar experience.

In case of the multimodal perception, it can be inferred that the vector portions of Equation 8.4 have different lengths. The portion provided by the laser data shows a length of 46208 bits while the portion provided by the vision sensor features only consists of 2304 bits. This implies that the distance values of the laser scanners influence the relative error (the used distance metric) of a crossmodal multi-SDM prediction approximately 20 times more strongly than the image-based portion. Thus, a variable weighting factor has been intro-

duced to optionally strengthen the influence of each sensoric component during a multi-SDM prediction separately. In case of the experiments presented in Section 8.3, a weighting factor of 60 is applied to the portion that contains the edge pixels computed from the images. Accordingly, the image-based portion thus influences the relative error 3 times more strongly than the laser range portion.

8.2.4. Belief Value

For the purpose of memory-based robot localisation, the multi-SDM model is used to classify the position which the robot believes to currently be in. As shown in Section 7.4, the SDM that provides the smallest amount to the entire error wins. Since a low value should not be taken as a representative for any winning score, a belief value is computed according to Equation 8.5. For the sake of clarity, the belief value is used to avoid possible misinterpretations by the reader. For the belief score Ψ of SDM_i the sum of the distances to the closest hard locations of each SDM depicted by $\sum_{j=1}^{l} \theta(SDM_j)$ is subtracted from the distance of the particular SDM denoted by $\theta(SDM_i)$. Afterwards, the result is divided by the sum of all such individual distance computations.

$$\Psi(SDM_i) = \frac{\sum_{j=1}^{l} \theta(SDM_j) - \theta(SDM_i)}{\sum_{k=1}^{l} \sum_{j=1}^{l} \theta(SDM_j) - \theta(SDM_k)} \tag{8.5}$$

8.3. Results

Figure 8.6 shows the robot's belief values based on several multi-SDM predictions across the entire office environment. Subsequent bar charts illustrate the robot's belief values regarding a particular room of the office environment when it is placed in that room. Further charts are presented in Chapter D. Figure 8.6 shows that the number of correct classifications can be increased by using a crossmodal representation. In the case of the crossmodal classification, 14 patterns have been identified correctly while in unimodal classifications each modality identified 12 rooms correctly.

Note that the underlying map of the hallway in Figure 8.6 is unknown to the robot and is only presented for the sake of clarity. The robot's localisation algorithm presented in this chapter relies only on sensorial data it has perceived and stored in its memory before.

8.3.1. Advantages of Crossmodal Classification

As mentioned above, using more than a single modality can increase the classification accuracy. Therefore, four rooms will be considered in more detail. The rooms *Hallway 6*, *Hallway 4*, *Office 3* and the *Workshop* presented in Figures 8.6, 8.7, 8.8, 8.9 and 8.10 illustrate that one of both unimodal classifications fails.

In this particular cases, a crossmodal classification—where the world is represented by several modalities—can compensate for the unimodal misclassifications. Thus, a more reliable localisation of the robot can be accomplished. Unfortunately, this is not generally valid for all cases. Examining Figure 8.10 in more detail reveals that the robot's crossmodal belief of being in *Hallway 6* strongly competes with *Hallway 5* (Figure 8.13). However, both rooms finally win the predictions due to the same belief values but still yield an inconclusive prediction result. It can be seen that for most of the data samples the multi-SDM provides similar

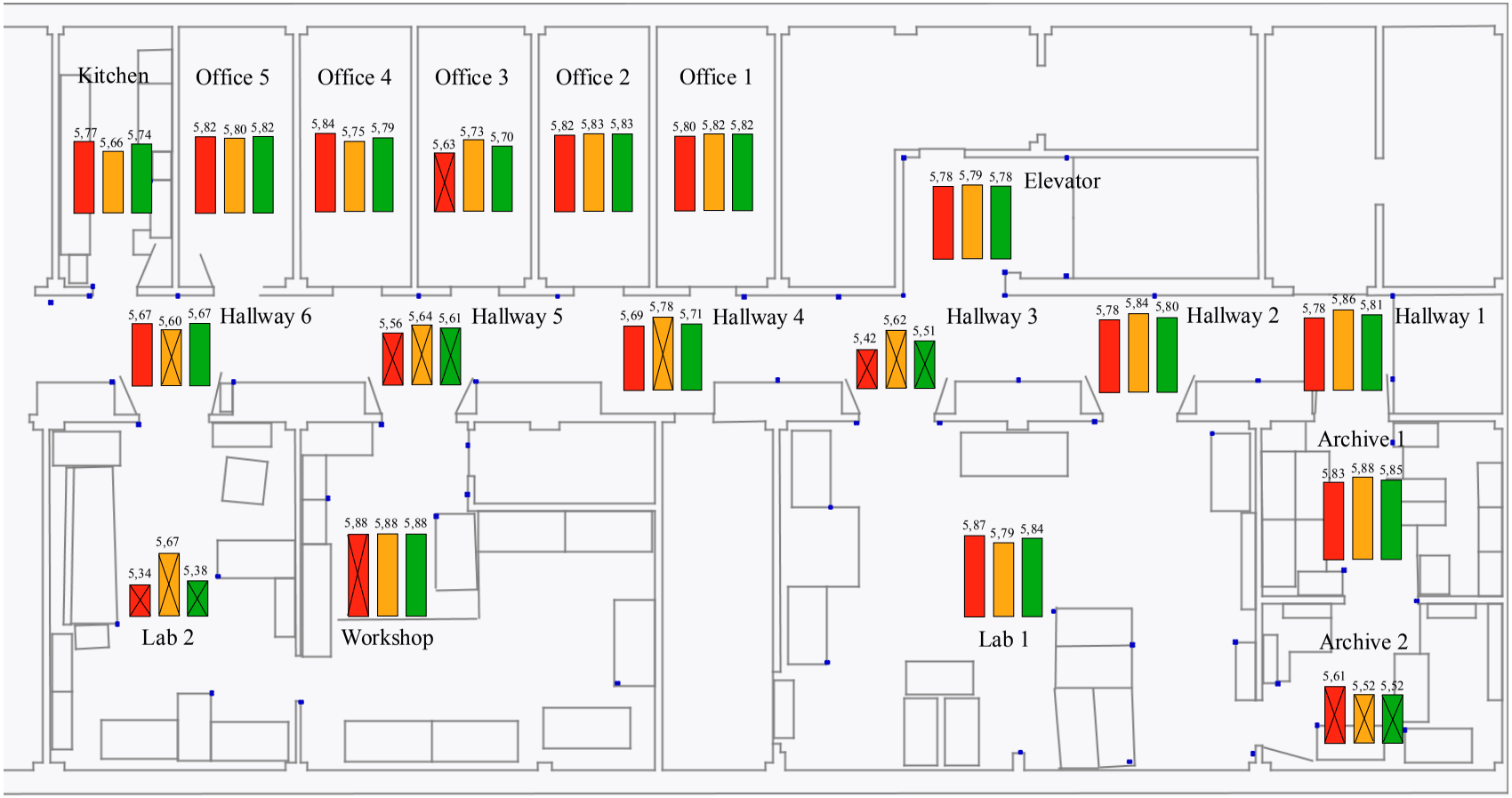


Figure 8.6.: Crossmodal integration of laser range and visual data for contextual robot localisation. The unimodal memory-based robot localisation for the laser (red) and visual modality (yellow) are illustrated in contrast to the multimodal localisation (green). The numbers depict a belief-value (the higher the better). A crossed bar shows a misclassification regarding the particular modality.

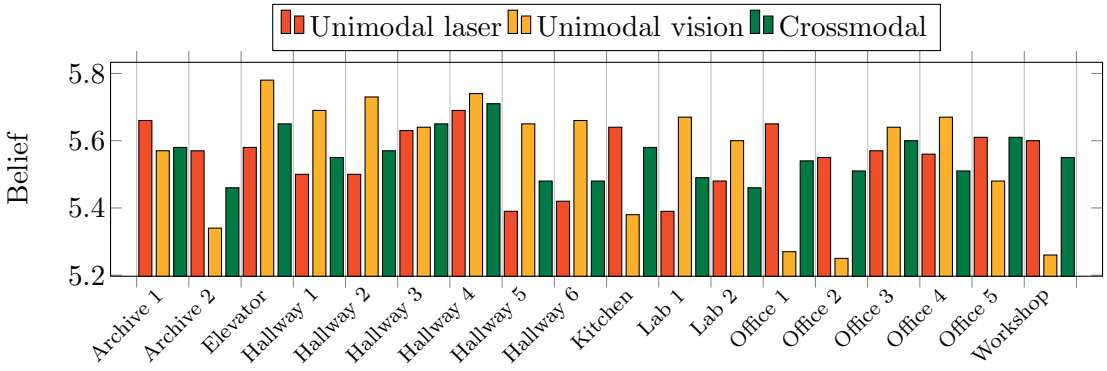


Figure 8.7.: The robot's belief if triggered with pattern *Hallway 4*.

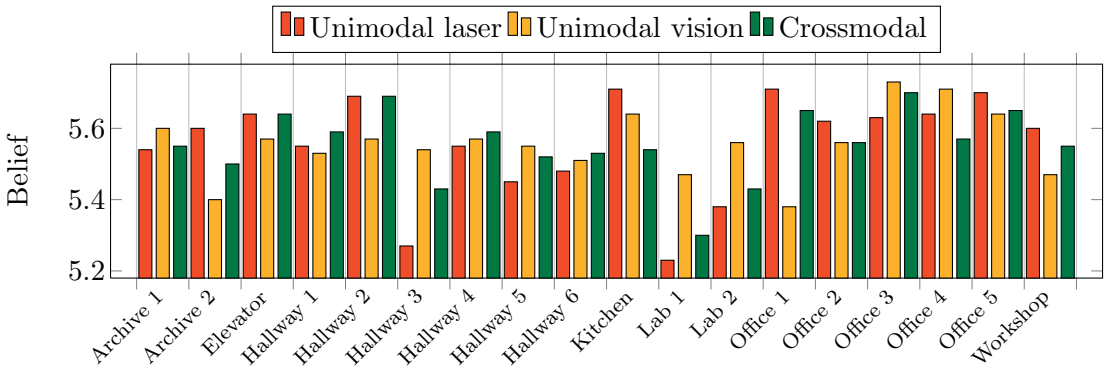


Figure 8.8.: The robot's belief if triggered with pattern *Office 3*.

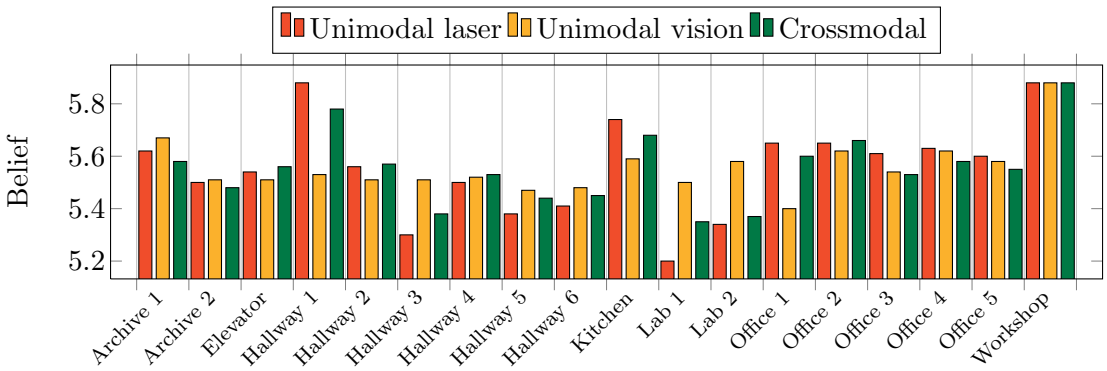


Figure 8.9.: The robot's belief if triggered with pattern *Workshop*.

beliefs for rooms that look similar, e.g. for the hallways. Looking at the misclassification of *Hallway 3* and *Hallway 5* shows that the robot preferably beliefs to be located in any of the other hallway rooms.

Contrary to what has been expected, the crossmodal classification is not reliable for identifying the robot's position correctly when both unimodal classifications fail. Examples for this are rooms such as *Archive 2*, *Hallway 3* and *Hallway 5* presented in Figures 8.11, 8.12 & 8.13 respectively.

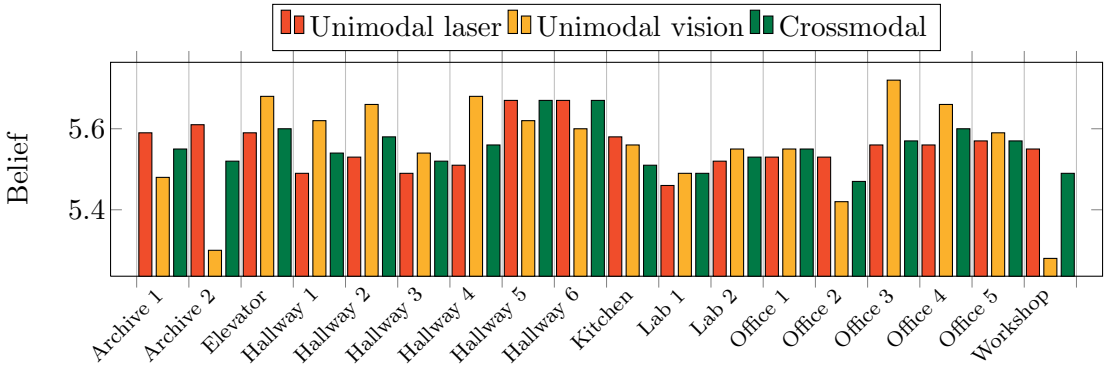


Figure 8.10.: The robot's belief if triggered with pattern *Hallway 6*.

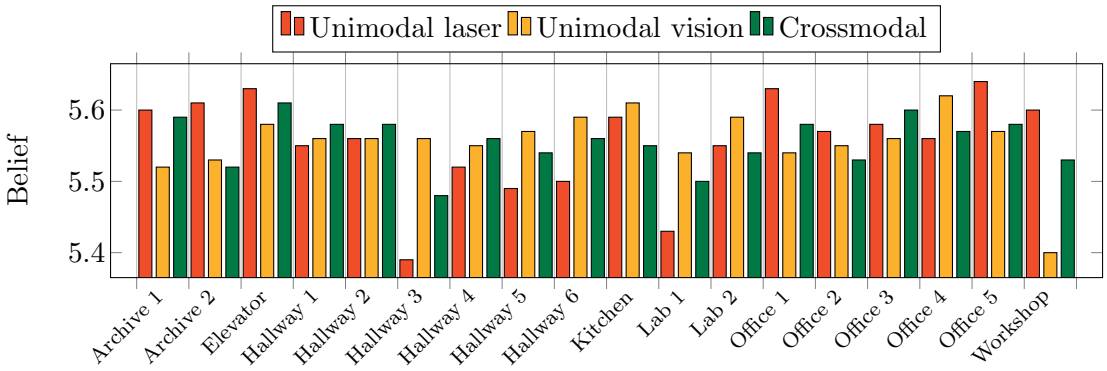


Figure 8.11.: The robot's belief if triggered with pattern *Archive 2*.

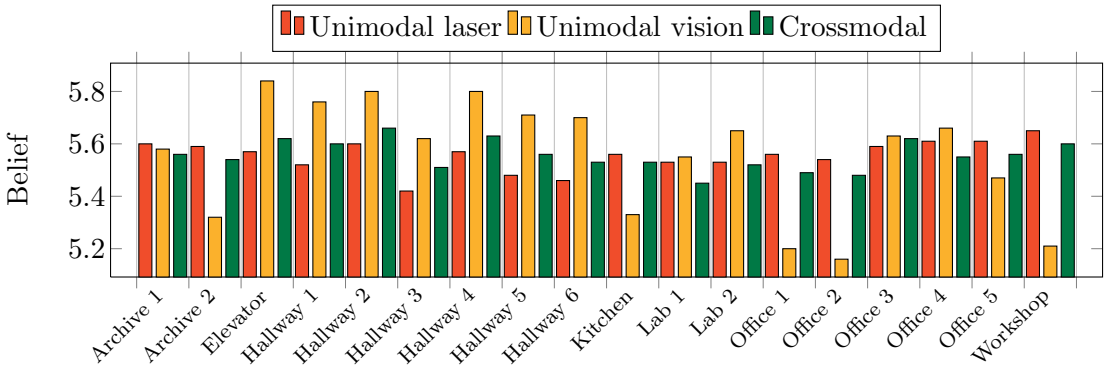


Figure 8.12.: The robot's belief if triggered with pattern *Hallway 3*.

8.3.2. Classifying Rooms with Variations

The data acquisition was accomplished across different dates and day-times. Figures 8.14(a)–8.14(c) illustrate that the training data possess variations in the robot's position, lighting and arrangements of the scene. By using image features such as edges, most of the variations

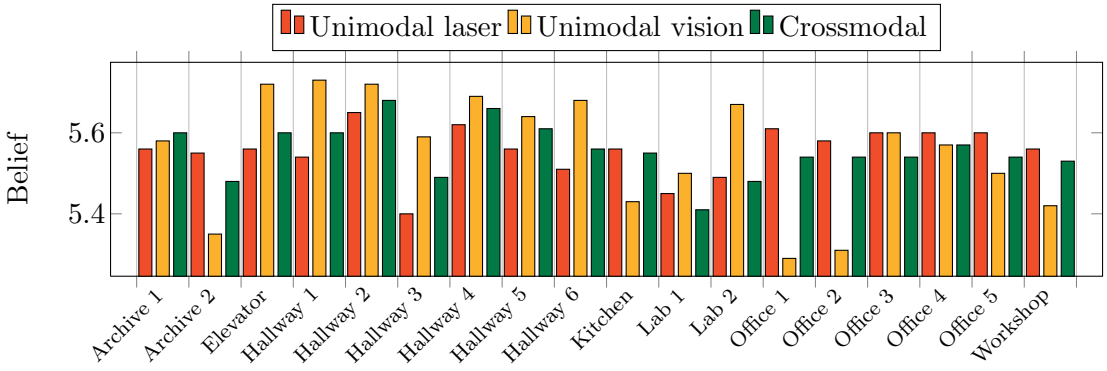


Figure 8.13.: The robot's belief if triggered with pattern *Hallway 5*.

in the scene's illumination conditions become irrelevant[5].

Note that image sampling follows Equation 8.2 and the angle of the first slice correlates to the angle of the robot during data acquisition. Thus, the image orientation is compensated while the robot's position slightly varies across the different images of the test set. Looking at the trigger image shown in Figure 8.14(d) reveals that several persons are located in the office, too. Nonetheless, the robot's location can be successfully determined by its past experience of the room (cf. Figure 8.15). Such particular examples show the high flexibility of a multi-SDM robot localisation.

Other triggers, for instance with an open door, also showed that the multi-SDM is suitable to locate the robot's position correctly although the doors were closed in all of the training patterns. It can be concluded that the multi-SDM provides a flexible mechanism for robot localisation whose accuracy can be improved through the integration of different modalities.

8.3.3. Separability of Particular Rooms

Several rooms show similar characteristics with respect to their size and the amount of furniture. The latter leads to laser range scans that comprise major and sudden changes in the measured depth values. Compared to all the other rooms of the office environment, the laboratories and the workshop are relatively large rooms. The offices are of comparably small size. The multi-SDM mostly provides comparable beliefs for rooms that look similar with respect to their proportions, e.g. for hallways, laboratories and the offices. But what may be the reason for this?

In some cases, the multi-SDM computes comparatively high beliefs for a group of rooms, e.g. offices in Figure 8.15. It can be seen that most of the offices' belief values exceed the values of all remaining rooms. This effect is considerably large for *Lab 1*. Figure 8.16 shows that the robot's belief of being in one of the large rooms with a lot of free space and less furnishing considerably exceeds the low belief levels for the remaining rooms of the office environment. This illustrates that the multi-SDM is able to find certain high-level relations within low-level sensory data patterns without explicitly teaching them.

[5]Small variations of the images' brightness are manageable. Please note that this statement does not include vigorous changes, e.g. a dark room at night without any room illumination as opposite to a bright room during a sunny day.

(a) (b)

(c) (d)

Figure 8.14.: Source images of the training set for the visual modality for *Office 1* (a-c) and source image of the trigger for the multi-SDM classification (d).

8.3.4. In Search of Crossmodal Effects

The multimodal localisation experiment presented in this chapter was carried out to check whether crossmodal effects occur in a multi-SDM prediction like in human perception. It was anticipated that a crossmodal belief exceeds the beliefs achieved by the unimodal perceptions, thus yielding a supramodal effect. According to the operational definition given in Section 8.1, the research question of whether crossmodal prediction can be advantageous for memory-based robot localisation has been answered. Figure 8.6 shows that the crossmodal prediction predominantly reveals a partially facilitatory effect with respect to the unimodal classifications. However, a supramodal effect in the current configuration of the memory

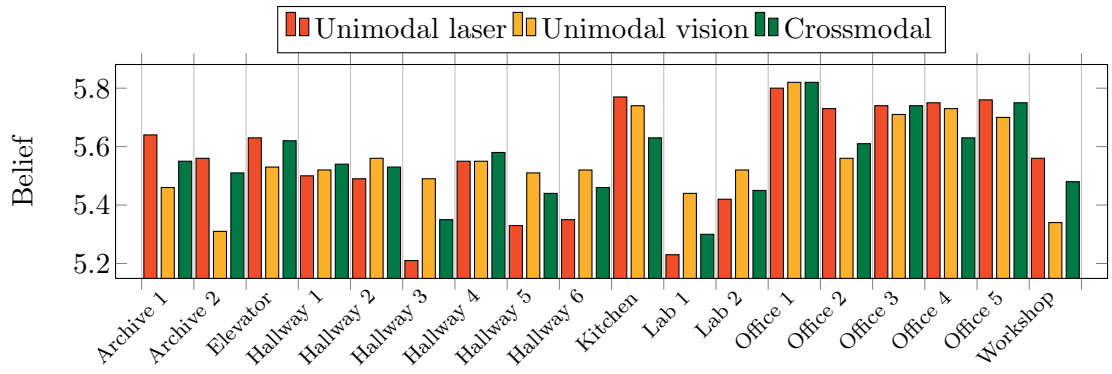


Figure 8.15.: The robot's belief if triggered with pattern *Office 1*.

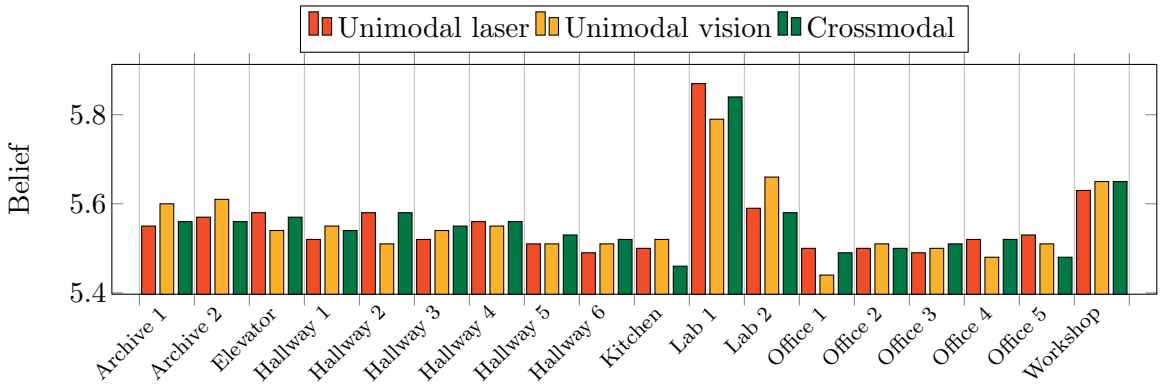


Figure 8.16.: The robot's belief if triggered with pattern *Lab 1*.

system can be as little observed as totally inhibitory effects. The particular belief figures demonstrate that a crossmodal prediction forms a kind of average of both of the unimodal predictions.

Let us consider the crossmodal input vector of a multi-SDM again. As already mentioned, the laser-based portion of the vector is 20 times larger than the visual-based portion. Thus, when comparing any crossmodal input vector to memory hard locations, the laser-based perception of a room has a stronger influence on the errors of the distance metric. An empirical analysis was conducted to determine the average influence of each modality on the error rates of a multi-SDM prediction. It reveals that the laser-based error is 130 times higher on average than the vision-based error. Considering the circumstance that the laser-based vector portion is 20 times larger than the vision-based vector portion, the former portion causes 6.5 times more errors on average than the latter portion. Paradoxically, it was empirically observed that weighting the vision-based portion by a factor of 60 yields a proper localisation behaviour for the robot. But currently it is not clearly evident why a weighting factor of 6.5 does not perform comparably well.

However, this experiment clarifies that weighting particular modalities in a crossmodal classification task has a considerable influence on the computed results. It would be of great interest for this study to know which senses dominate in human crossmodal perception[6]. More thorough investigations on cognitive neuroscience are necessary to resolve the modality-specific weighting in crossmodal perception. The outcomes of such studies may help to determine appropriate weighting factors for further studies on multi-SDM-based crossmodel perception, particularly if further sensors are involved in future.

8.4. Discussion

Previous chapters considered the implementation of the SDM and multi-SDM concept for unimodal robotic experiments. The experiments of this chapter reveal the feasibility of using a multi-SDM architecture for multimodal studies in artificial cognitive systems. The multi-SDM was used to achieve a localisation of a robot within an office environment based on past experience. The integration of various sensoric modalities led to a more reliable localisation than achievable by unimodal SDM-based classification. These experiments proved that the multi-SDM architecture is suitable to deal with input vectors that integrate several unimodal percepts into a unified crossmodal representation of the world at a particular time. The multi-SDM offers a computational method for unification-based crossmodal interactions[7] that exhibit partially facilitatory effects. Future research may close the gap between natural intelligent and artificial intelligent systems by further developing the proposed system.

The robustness of a perception can be increased by the combination and integration of multiple sources of sensory information. Using multiple sensory sources helps to achieve disambiguation of similar perceptions. The above-mentioned experiment outlined that a multi-SDM crossmodal perception can obtain disambiguation. The proposed architecture

[6] Indeed, it can be assumed that the visual modality dominates in human perception under consideration of the size of the modality-specific regions in the brain. The visual cortex is considerably larger than all the brain regions used for other sensoric modalities. This is not only caused by the complexity of the respective modality.

[7] Note that it is not claimed that the proposed method replicates human intelligence regarding crossmodal interactions.

constitutes an appropriate platform for additional studies on crossmodal perception and integration.

Future work may involve the integration of further modalities or microfeatures of the world. Such features could comprise a colour histogram, lines, corners and so forth. Several computed beliefs vigorously compete for the winning prediction, e.g. the robot's crossmodal beliefs of being either in *Hallway 6* or *Hallway 5* when triggered with the pattern presented in Figure 8.10. Future work may comprise studies where competitive predictions yield a fuzzy robot localisation such that the multi-SDM indicates the top three classifications as most probable. Examining different permutations of several involved modalities[8] followed by a comparison and majority decision regarding the sets of predictions may lead to more reliable predictions. It has been shown that the multi-SDM provides a flexible architecture for studying the crossmodal fusion of low-level sensory-based patterns.

[8] If more than just two modalities are concerned in crossmodal classification.

9 Conclusion

Memory itself is an internal rumour.
(Jorge Augustín Nicolás Ruiz de Santayana, American philosopher, 1863–1952. The Life of Reason or The Phases of Human Progress)

Contents

9.1. Summary

In this work, an autobiographical associative memory system has been developed for the purpose of learning and predicting low-level sensor and actuator patterns of an autonomous service robot. The following topics have been addressed in the course of this work:

- Development of an associative memory structure based on a connectionist approach to learn and predict robot arm motion sequences for an autonomous task execution.
- Association of new input patterns to already learnt contexts with the goal to generate an appropriate output even when modified or noisy contexts are presented to the memory.
- Development and analytical investigation of alternative information encoding methods for an optimisation of the memory model.

- Comparison of the memory model's performance when transferred to other robotic modalities and domains, such as vision-based robot navigation.
- Extension of the architecture and evaluation for the purpose of learning, generalising and classifying interactive manipulation sequences. Human instructors with different skill levels demonstrate complex manipulation tasks to the robot via a telemanipulation system.
- Comparison of uni- and multimodal perception for rough robot localisation in an office domain based on the developed autobiographical robot memory.

Learning is part of intelligent behaviour and is based on building and operating on an internal model of the world. Besides just observing and learning physical aspects of the world through sensors, a system also acts within the world and learns from its interactions. Learning to perform actions relies on learning to reproduce sequences of motion patterns. The predictive power of a memory is given by its ability to retrieve sequences and to generalise. The presented work has described a memory system to store an autobiographical past of a robot system. It memorises sequences of discrete robot motion patterns in an SDM. Even if not intuitively compelling, Kanerva's mathematical-statistical SDM approach made a major contribution by exploring the nonintuitive properties of high-dimensional binary space. Patterns representing the current moment can be used to address and retrieve consequences of similar moments in the past. Retrieving consequences of similar moments from a memory of autobiographical experiences constitutes an important aspect of an autonomous, mentally developing robot system to choose appropriate actions, e.g. to avoid danger, to seek reward and to re-use solution strategies for similar tasks.

Several modifications have to be applied to the SDM model to learn non-random sensor-based perceptions of a robot. The drawback of an *a priori* selected and non-modifiable address space has been eliminated by using a randomised reallocation algorithm. Memory locations are dynamically added based on the observed data rather than being restricted to a fixed address space. New hard locations are allocated randomly in the neighbourhood of an input address if the data cannot be stored into enough existing hard locations. Further, the memory capacity of the autobiographical robot memory has been increased by replacing the counters of the contents matrix by single bits. As a result of the replacement the efforts for computing the counter's values could be reduced and the operational speed of the entire system was increased.

The bitwise implementation with the Hamming metric does not perform very well, and thus, alternative coding schemes were developed. The performance of three encoding modes have been compared and analysed in Section 5.3 and showed that the model can be significantly increased when operated with an arithmetic mode and Euclidean distance metric. The sum code mode exhibited a robust behaviour, too. Its performance corresponds approximately to the arithmetic mode, but at the expense of a longer runtime due to the increased vector size and corresponding distance computation. Note that the increase in runtime only occurs when the SDM is implemented in software rather than in a parallel hardware architecture. Since this work is primarily concerned with an implementation for robots whose control PC usually operates at performance limits, runtime and the required memory space cannot be disregarded. In Chapter 5 it has been shown that robot arm motion sequences can be learnt by a modified SDM. One of the main advantages for a robot system is that the property of iterative prediction of consequences for certain situations enables the robot

to re-execute sequences autonomously. The problem of predicting higher than first order associations has been partially solved by using the folding concept. Additional SDMs also represent second-, third- and up to k^{th}-order state transitions, however, at the expense of higher computational costs. Within the multi-SDM architecture, the context of a sequence state has been considered by weighting the preceding sequence predictions. Nevertheless, temporal dependencies still remain a significant issue in sequence learning, especially in the case of long-range dependencies.

Chapter 6 showed that the developed memory model also supports other sensory modalities. It was empirically shown that the three proposed encoding modes yield a similar performance according to the predictive property of the memory even when applied to different fields, such as vision-based navigation and manipulation. The arithmetic mode achieved the most robust behaviour in the domain of view sequence-based robot navigation, too. It further became clear that the memory model is suitable for working with considerably larger memory spaces than presented in the manipulation cases of this work and a lot of the preceding work on the SDM.

The developed memory model was mainly used as a flexible pattern recognition and prediction system. Different sensor-based and actuator-based patterns, captured during interactive manipulation task demonstrations performed by several human teleoperators, had to be processed by the memory system. It was therefore very important to develop a memory that is content-addressable, such as in human memory. A linkage of various low-level sensory percepts and several high-level semantic task descriptions has been achieved by an extension referred to as multi-SDM architecture, detailed in Section 7.4. This linkage, in parts, refers to the symbol grounding problem (Harnad, 1993; Coradeschi and Saffiotti, 2003).

Various robot arm motion trajectories of different tasks have been learnt and generalised by the multi-SDM architecture. The memory model proved its ability to relate yet unknown and vague motion trajectories to high-level tasks and to predict consequences of given situations based on past experience, cf. Section 7.5. The multi-SDM can infer high-level events from sequences of states that describe the world at a definite point in time. By means of a detailed analysis it has been shown that the proposed system is able to classify vague and non-identical trajectories correctly with high accuracy. It has to be emphasised that the multi-SDM model obtains comparable results regardless of the teacher's skill level. The system does not show a significant difference when trained by a single expert or a set of laymen. The generalisation and local Hebbian learning property together with the sparse and distributed memory characteristic yields a good possibility to learn from numerous teachers. Moreover, the autobiographical robot memory is able to compensate for memory damage and spontaneous loss of arbitrary memory locations through its redundant information storage mechanism.

Chapter 8 showed the feasibility of using a multi-SDM architecture for multimodal studies in artificial cognitive systems. The proposed architecture constitutes an appropriate platform for additional studies on crossmodal perception and integration. Training the multi-SDM architecture with unification-based crossmodal percepts led to partially facilitatory crossmodal effects with respect to a robot that has to localise itself in an office environment based on its past perceptions.

Kanerva (1988) concludes his seminal work with the hypothesis that SDMs are particularly suitable for robotics, which has been partially proven by this study. Nevertheless, some criticism still arose during the implementation proposed in this work. Since the SDM is a parametric model, the learning rate and activation radii have to be determined with

respect to the application purpose. When closely related values are stored in an SDM, large activation radii yield high error rates. Decreasing the activation radii to gain better results reduces the distributive characteristic of the model and thus constitutes a conflict between generalisation of input patterns and reproduction accuracy of sequences.

For encoding a world in an SDM, the representative features heavily depend on the available sensors and the particular domain. An SDM always represents the entire context of the world, regardless of whether each entity is of particular interest or not. The entire context is treated equally. Chapter 8 shows that any weighting of involved modalities requires empirical analysis. The SDM always relates input patterns to the closest stored concepts. Defining decision functions for detecting *new* concepts are again domain-specific. As all PDP models, the SDM is not transparent to the user concerning the representation of the stored data.

9.2. Contributions of this work

It was shown that the SDM model as a mathematical model of certain apparent properties of real neural systems yields a good mechanism to deal with high-dimensional problems of low-level robotics. This work presents the first transfer of the SDM concept to mobile robot manipulation and further robotic applications. Moreover, this work presents the first application of an SDM-based multimodal integration for a catadioptric vision system and laser range finders. It demonstrates that the SDM model can be applied to several of the current challenges in robotics but also reveals certain practical problems when transferring the SDM's elaborated theory to robotic applications. Nevertheless, the presence of noise in sensor data is nearly inevitable and a challenging problem when dealing with sensors. The developed model enhances a robot with elementary cognitive processes to determine proper response functions and actions to perceived situations by finding the best corresponding output to an imperfect input vector. The resulting autobiographical memory provides the robot with simple storage and retrieval mechanisms for maintaining subjective experiences. The distributed data storage and retrieval yield a more robust information storage with respect to failure and damage of individual memory locations. Although heavily dependant on an appropriate selection of parameters, as shown in Section 5.3, the model provides good capabilities for trajectory learning based on simple learning rules.

This project has been part of the CINACS Graduate Research Training Group on Cross-modal Interactions in Natural and Artificial Cognitive Systems[1]. The main contributions of this work are as follows.

9.2.1. Robotics

Content-addressable memories such as the developed autobiographical robot memory bear great potential to access stored information based on the information itself. The main achievement of this work is to show that a context-based and self-organising memory with localised Hebbian learning can be successfully applied as a content-addressable autobiographical robot memory for past events. The implementation is suitable for both one-shot online and offline learning and furthermore supports natural forgetting.

Another important characteristic shown is the memory's robustness against damage and loss of individual locations. It was shown that a robot equipped with an SDM does not nec-

[1] `http://www.cinacs.org`

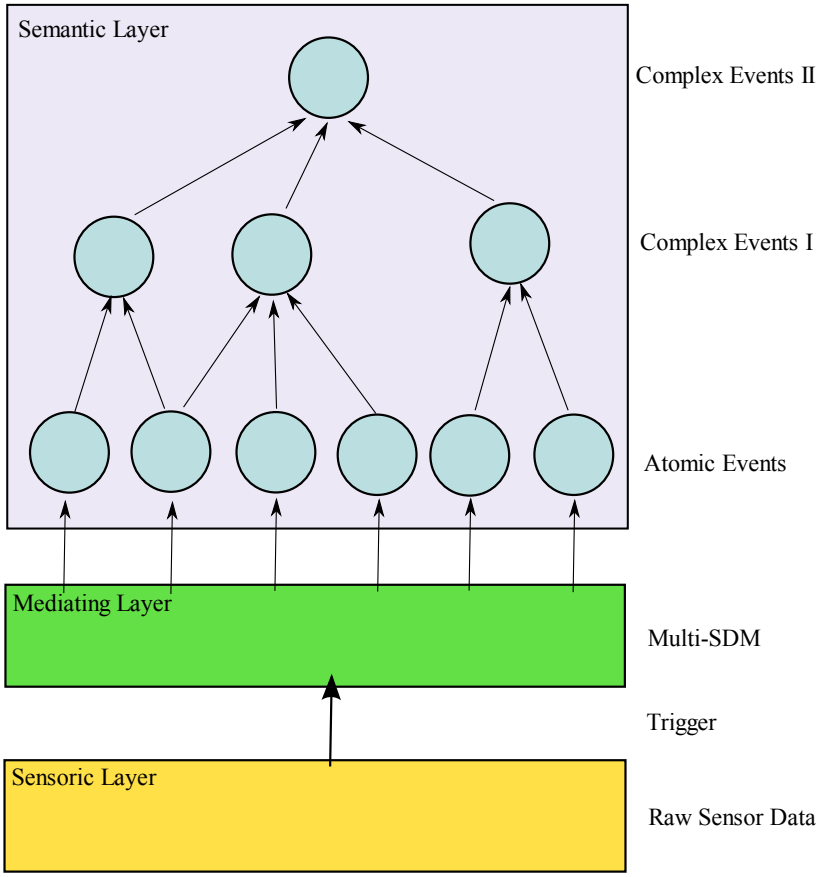


Figure 9.1.: Putting SDM into a frame: Taxonomy of an entire perception system for robots. The multi-SDM is suitable to mediate between low-level sensory-based and high-level semantic layers.

essarily lose information irrevocably when some memory locations are injured, and remains capable of retrieving action consequences. The autobiographical memory enables a robot to learn on the fly and to dynamically adapt its model of the world. The developed system has been successful in retrieving appropriate actions from its past according to similar situations. The robot is able to autonomously complete a motion sequence by retrieving the associated actions of a current situation from its experience. The proposed model provides a valuable alternative for robot learning-by-demonstration and is particularly attractive because of its simple learning and generalisation mechanism as well as its integrated memory storage and retrieval procedures.

A multi-SDM can be integrated in the framework of a comprehensive perception system of a robot (see Figure 9.1). A still challenging problem of cognitive robotics is the sensorimotor transduction problem, the representation of sensory information within a symbol system. A solution to this problem is a prerequisite for a mapping of perceptions to semantic knowledge (Hertzberg and Saffiotti, 2008). Numerous research activities in robotics and computer vision approach this problem, e.g. Chella et al. (2000); Coradeschi and Saffiotti (2000); Coradeschi et al. (2001); Coradeschi and Saffiotti (2003); Gärdenfors (2004).

Most approaches model the mapping of sensor values to symbols manually, which is any-

thing but simple. Obviously, the perception of the world is totally different in robots and humans. It is a promising approach to base this mapping on the subjectively learnt experience of a robot. In contrast to the above-mentioned approaches, the developed multi-SDM mediates between the sensory and semantic layer as follows: (a) The mapping of sensor-based patterns to semantic entities is learnt by the robot based on its experience while acting in its environment. (b) Learning is online and can be carried out for a robot's entire lifecycle. Changes in the world cause changes in the behaviour of the multi-SDM, and thus of the robot's behaviour. This gives rise to a plasticity analogous to the one accomplished by natural cognitive systems.

Figure 9.1 summarises the integration of a multi-SDM model within the perception system of a robot. Features of an arbitrary input, so-called microfeatures, are extracted within a sensoric layer. Those microfeatures constitute the SDM address vectors. Triggering the multi-SDM with such address vectors yields a classification of (sequential) input data to symbols represented in the semantic layer. Furthermore, currently perceived and classified sequences can be learnt online as new experiences to influence future behaviour of the multi-SDM.

9.2.2. Modelling Cognitive Functionalities & Multimodal Integration

High-dimensional spaces and randomness constitute valuable concepts concerning the modelling of cognitive mechanisms. Hyperdimensional representations may explain some illusions in the future that let us doubt our perception. Mathematical considerations can suggest how circuits need to work to achieve certain cognitive functionalities.

Neuroscience can benefit from mathematical approaches to representation and perception. Experimental psychology and neuroscience emphasise the crucial influence of multimodal perception and representations though focusing mostly on unimodal perception in the past. Further studies on connectionist approaches in close collaboration with experimental psychology and neuroscience have a high potential for contributing to the decipherment of mechanisms of multimodal cognitive processes. The analogous and associative nature of the brain to retrieve solutions to similar previous problems from memory rather than to compute new solutions has to be transferred to robot applications to gain genuinely intelligent behaviour. To be able to constantly adapt and extend a world model, a robot with a grounded memory needs plasticity for processing dynamic events and a possibility to learn online.

The multi-SDM presents a first bottom-up approach to assess simple storage and recall mechanism to provide robots with mathematically inspired but biologically plausible memories. Though not entirely solved, the developed autobiographical memory is a step towards grounded memories for robots based on biological plausible mechanisms. Rather than focusing on single modalities that highlight different aspects of the world, different sensors can jointly encode particular aspects of it. The sparse and distributed memory concept offers a good foundation to study such jointly encoded aspects. The developed system provides a basis for further investigations on cognitive influences based on multimodal robot perception.

9.3. Directions for Future Work

The following options for extension and improvement to this model of memory have been identified.

9.3.1. Potential Improvements Concerning Task Representations

Transforming the input signal into a form useful for modelling the world is still a challenging problem. The encoding problem plays a crucial role when dealing with connectionist models. It already arises in the initial developing stages with design decisions such as using binary or continuous feature descriptors. Furthermore, it is crucial to decide which features should be used to represent the world, concepts, tasks and objects appropriately. In case of motion trajectories, Wu (2008) developed signature descriptors based on Euclidean differential invariant features to describe raw trajectories. Using such invariant features may improve the task descriptions within the robot's autobiographical memory. Other approaches for deriving motor primitives and atomic actions have been proposed by Jenkins and Mataric (2004); Williams et al. (2006).

The developed multi-SDM architecture links an entire motion sequence to a high-level semantic task descriptor. An improvement would be to represent a complex task by a set of atomic actions (Weser, 2009). The multi-SDM system can then be used with quantitative predicates and information in the following way: each SDM instance represents an atomic robot action, e.g. bend arm, unbend arm, open hand, twist wrist clockwise, and so forth, rather than representing complex tasks. Thus, the multi-SDM can categorise portions of more complex robot arm motion trajectories online over time. An additional hierarchical layer may record the temporal occurrences and relations of robot actions and perceptions as a sequence of high-level events. Such a recording mechanism has already been developed in the context of an preliminary study on episodic robot memory called EPIROME in Jockel et al. (2007b, 2008b). Apart from that, another SDM can be implemented as a higher cognitive layer that deals with symbolic descriptions to represent higher-order concepts, e.g. analogous to a hierarchical composition of complex actions.

9.3.2. Potential Improvements Concerning the Overall System

The developed SDM and its extension referred to as autobiographical robot memory have been implemented in software. For pragmatic reasons, the robot's memory is maintained on a workstation within a distributed software architecture. This saves processing resources for major control tasks of the robot. But, as shown by many authors, the SDM concept is eminently eligible for being realised in a parallel hardware architecture that can be directly connected to the robot. A workstation would thereby become superfluous and only few resources of the robot's control PC would be needed to maintain the memory inputs and outputs. Implementation in hardware would also considerably speed up the processing time of the memory, which is still challenging in a software implementation with growing memory size. Since the multi-SDM extension deals with an arbitrary number of separate SDMs, it necessitates either to have a number of SDM hardware modules or to extend the addressing mode and the content matrix to provide a number of virtual SDMs, e.g. identified by a unique index number.

Inclusion of all available robot sensors and actuators would yield a multimodal and more detailed world model that provides a comprehensive context database for more reliable computation of certain actions. The proposed system is ideally suited for classifying vague and noisy low-level sensor and actuator patterns with respect to their context. Adding more than a single layer as proposed in the multi-SDM architecture would yield a hierarchical network that may enable the system to achieve a symbolic description of a perceived situation. With

appropriate feature detectors, the systems may also be used for the arbitrary association of high-level symbols (cf. Figure 9.1).

9.3.3. Open Issues According to Sequence Learning

In sequence learning, temporal dependencies are still a significant issue, at least in case of non-Markovian sequences. Most current models have problems to handle dependencies that are not just related to what has happened right before a current situation, but might have happened a long time ago. Long-range dependencies are hard to learn for neural network models as well as heuristic models and even harder for reinforcement learning (Sun and Giles, 2001).

Although Kanerva proposed the k-folded SDM to overcome temporal dependencies, it remains a challenge to apply this idea to long-range dependencies of sequence elements. However, the reported experiments focused on motion trajectories with a limited sequence length. Experiments showed (cf. Section 5.2.3) that most of the prevailing dependencies became manageable. Nevertheless, long-range dependencies cannot be completely overcome with the folding concept due to the resulting expansion of storage capacity and increase of prediction time, and thus, improved approaches are required.

Another issue is hierarchical structuring of sequences. A sequence may consist of numerous subsequences which in turn consist of subsubsequences and so forth. The problem is to identify the boundaries of such subsequences. If found, hierarchical structuring might help to reduce temporal dependencies.

Even if this work does not solve the hierarchical structuring issue completely, the developed multi-SDM architecture provides a hierarchical extension to the SDM concept. The SDM is used as an autobiographical robot memory to recognise interactive and noisy motion trajectories of teleoperators. By combining several labelled SDMs, it becomes possible to ground or assign physical properties gathered via sensors to symbols or semantic units.

A

Signs, Symbols and Acronyms

Sign or symbol	Denotation
$O(r, x)$	Circle of radius r and centre x
A	Address matrix of an SDM
C	Content matrix of an SDM
$d(x, y)$	Distance of two points x and y
E	Energy of a Hopfield network
K	Storage capacity
k	Number of folds, *see* k-folded memory
n	Dimension of a space
2^n	Number of points in space
N	High-dimensional space, here $N = \{0, 1\}^n$, also referred to 2^n
N'	Physical subspace of N
0	Origin
S	State
s	Number of current and past sensory inputs
x	Point in space N, a memory address represented by an n-tuple
$'x$	Complement of x
x'	A *hard-location*
$\|x\|$	Norm—number of ones—of a vector
$x : y : z$	A point y between points x and z
ξ^μ	Binary pattern
w_{ij}	Weight between two synapses
Z_2	Boolean space $Z_2 = \{0, 1\}$

Acronym	Denotation
ANN	Artificial neural network
BC	Binary code
DC	Direct-current electricity
DHT	Denavit-Hartenberg-Transformation
DoF	Degrees of freedom
GC	Gray code
HN	Hopfield network
LAN	Local area network
LbD	Learning by demonstration
LTD	Long-term depression
LTM	Long-term memory
LTP	Long-term potentiation
MHI	Mitsubishi Heavy Industries, Ltd.
NBC	Natural binary code
NN	Neural network
PA10-6C	Model number of used robot arm by Mitsubishi Heavy Industries, Ltd.
RCCL	Robot-Control-C-Library
RL	Reinforcement learning
SC	Sum code
SDM	Sparse distributed memory
STM	Short-term memory
TCP	Tool centre point
TCP/IP	Transmission control protocol/internet protocol
UDP/IP	User datagram protocol/internet protocol
UTF	Unicode transformation format
WAN	Wide area network
WLAN	Wireless local area network
WM	Working memory
WN	Willshaw network

B

Technical Details: The Telemanipulation System

Some more technical details of the telemanipulation system will be presented in this chapter. Further details can be found in Bruder (2009). First, some details of the communication protocol between client and server are given. After this, the gradient descent optimisation is outlined.

B.1. Transmission Protocol

The quite simple and flexible *transmission protocol* that is used for server–client communication is based on UDP/IP. In quick succession the server sends $\vec{s}_t$-packages to notify a client about the current state of its hardware. The structure of these $\vec{s}_t$-packages is as follows.

After the usual IP and UDP headers, the first two bytes of the own header give information about the package type and carry the *id* of the manipulator. This *id* is used to keep the system as flexible as possible for future use with respect to alternative manipulator hardware. The following three integers inform about the amount of information within a package and hold the timestamp in seconds and microseconds. The timestamp is used to re-identify the ordering of the packages and to discard outdated information. Next, the payload data with a set of 4-byte floating point numbers carries the information of the system state. This structure is illustrated in Figure B.1.

Such a package structure is easy to handle and the variables of $\vec{s}_t$ can directly be put into the data section without any conversion. By switching the first byte of the own header, the protocol can be used to transmit the command vectors $\vec{c}_t$ from the client to the server.

B.2. Gradient Descent Optimisation

Normally, a task specification for an arbitrary trajectory of a manipulator is given in Cartesian space while the controllers work in joint space. The mapping of the joint space onto the Cartesian space is commonly known as *kinematics*. If the Cartesian space is mapped onto the joint space it is referred to as *inverse kinematics*. When working with kinematically underconstrained, redundant manipulators, the inverse kinematics provides an infinite number

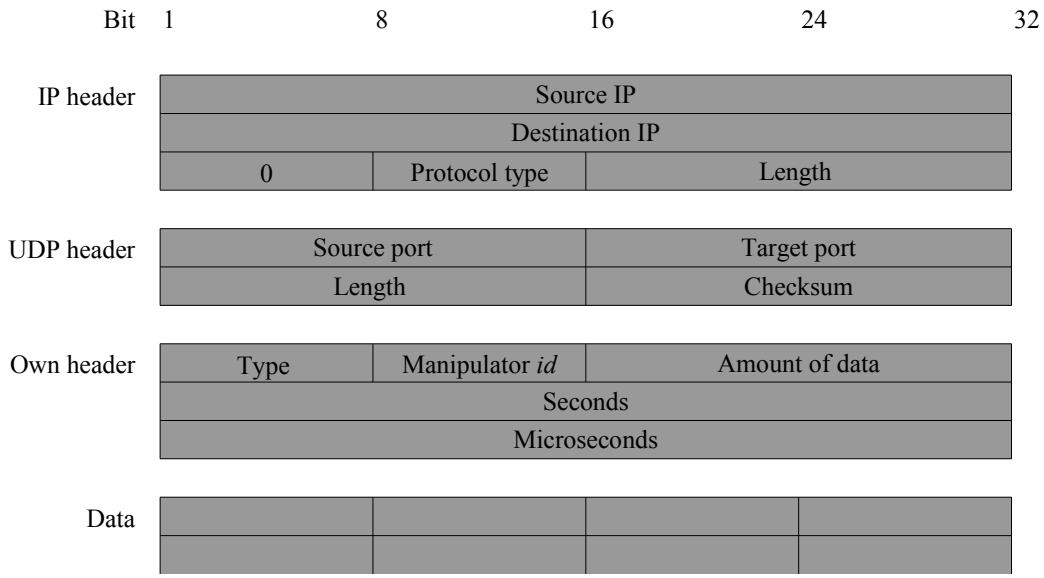


Figure B.1.: Package structure of the client-server transmission protocol for the telemanipulation system. By courtesy of Jan Bruder.

of solutions for the mapping between both spaces. To solve the redundancy, it is necessary to choose one of the provided solutions by some criterion.

The *Gradient descent*, also known as *steepest descent*, is a optimisation method for redundancy resolution. It is a first-order optimisation algorithm to find a local minimum of a cost function. It is an interactive method that uses the derivate of a function to reduce the value of it iteratively by taking steps proportional to the negative of the gradient.

During a heuristic search all joint angles j_i of a manipulator will be consecutively improved by the gradient of the error-function $\vec{\nabla}e$. The following equation where t refers to the iteration number illustrates this procedure:

$$\vec{j}_{t+1} = \vec{j}_t + c\vec{\nabla}e(\vec{j}_t) \tag{B.1}$$

To allow for continuation in case of $\vec{\nabla}e(\vec{j}_t) = 0$, a pseudo-random number $r_t \in [-\epsilon, \epsilon]$ with $\epsilon \in \mathbb{R}$ is introduced such that:

$$\vec{j}_{t+1} = \vec{j}_t + c\vec{\nabla}e(\vec{j}_t) + r_t \tag{B.2}$$

The value $c \in \mathbb{R}$ defines the speed of the descent. The goal is to minimise the abberation of the position and orientation by $e(\vec{j}) = dist(\mathbf{DHT}(\vec{j}) - \mathbf{T}_{goal}(\mathrm{R}, \vec{p}))$ with the Denavit-Hartenberg-Transformation $\mathbf{DHT}(\vec{j})$ and the orientation and position of the tool centre point, described by $R \in \mathbb{R}^{3\times3}$ and $\vec{p} \in \mathbb{R}^3$ respectively. The Denavit-Hartenberg-Transformation $\mathbf{DHT} : \mathbb{R}^n \mapsto \mathbb{R}^{4\times4}$ maps the vectors of the joint space into a three-dimensional Cartesian space.

To improve the speed of convergence, values c and ϵ can be modified depending on the iteration step t or $\vec{\nabla}e(\vec{j}_t)$. To stop at a stable solution for $\vec{\nabla}e(\vec{j}_t) = 0$ must $\epsilon = 0$. To search for a global minimum the method of simulated annealing would be used when ϵ remains variable.

The fact that a gradient descent method can take many iterations to converge to a local minimum is one of the disadvantages of such an algorithm. Furthermore, a dynamic computation of the step size rather than choosing a fixed one can entail high computational costs.

C

Multi-SDM Predictions

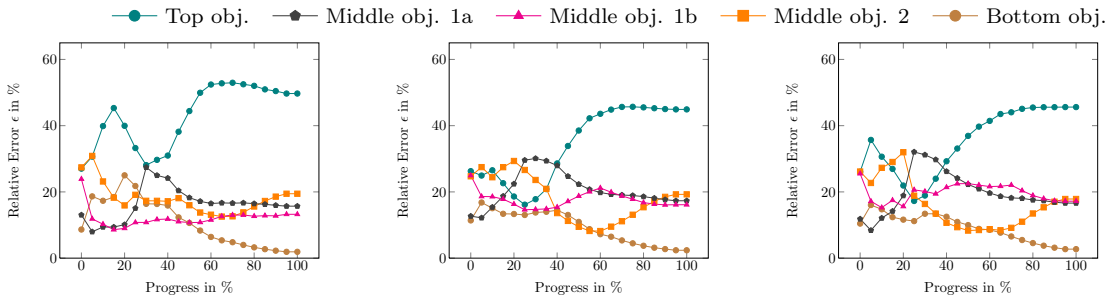
This appendix complements Chapter 7 by presenting the diagrams of all predictions made by the multi-SDM architecture that have been used for evaluation. This appendix is subdivided in consideration of the experimental setup as described in Sections 7.5.1–7.5.5. As a reminder, the particular descriptions of the experiments are given once more.

The set of results of each experiment are portioned into two main figures. One presents all successful classifications, while the other figure shows all failed task classifications of the multi-SDM architecture. Due to the potentially large number of tested trajectories, figures comprise a large number of subfigures. The main figure caption will then be given at the end of the figure that may range over several pages.

C.1. Experiment A: Skilled Teacher–Unskilled Users

All task executions of the experienced user are considered as training set for the multi-SDM system. The robot arm motion trajectories of the four remaining, inexperienced users are used as test set. This yields a training-to-test ration of 100:100 manipulation trajectories with a well-balanced training of the various tasks. The teacher–user relation can be described as logical `XOR`. This experiment studies how well the system classifies unknown trajectories when taught by a single expert. Figure C.1 shows all successful classifications while Figure C.2 summarises all failed classifications.

C.1.1. Successful Classifications



C.1.1: S–U_1; Bottom obj.(1); 55%

C.1.2: S–U_1; Bottom obj.(2); 60%

C.1.3: S–U_1; Bottom obj.(3); 60%

C.1.4: S–U_1; Bottom obj.(4); 70%

C.1.5: S–U_1; Bottom obj.(5); 70%

C.1.6: S–U_1; Top obj.(1); 10%

C.1.7: S–U_1; Top obj.(2); 15%

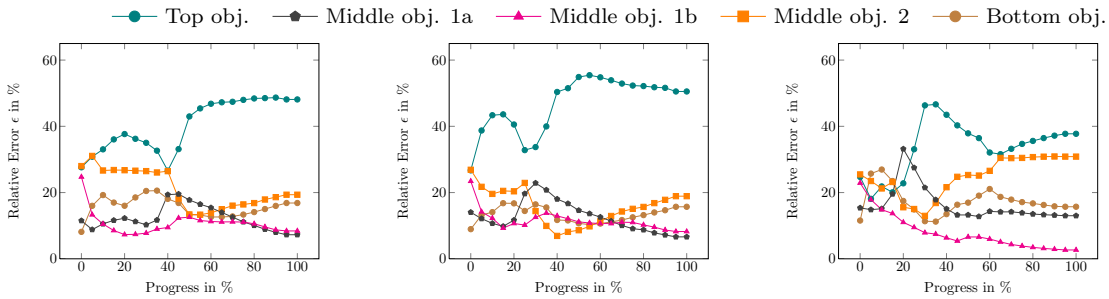
C.1.8: S–U_1; Top obj.(3); 10%

C.1.9: S–U_1; Top obj.(4); 15%

C.1.10: S–U_1; Top obj.(5); 30%

C.1.11: S–U_1; Middle obj.1a(1); 65%

C.1.12: S–U_1; Middle obj.1a(3); 75%



C.1.13: S–U_1; Middle obj.1a(4); 80%

C.1.14: S–U_1; Middle obj.1a(5); 70%

C.1.15: S–U_1; Middle obj.1b(1); 10%

C.1.16: S–U_1; Middle obj.1b(2); 40%

C.1.17: S–U_1; Middle obj.1b(3); 40%

C.1.18: S–U_1; Middle obj.1b(4); 30%

C.1.19: S–U_1; Middle obj.1b(5); 15%

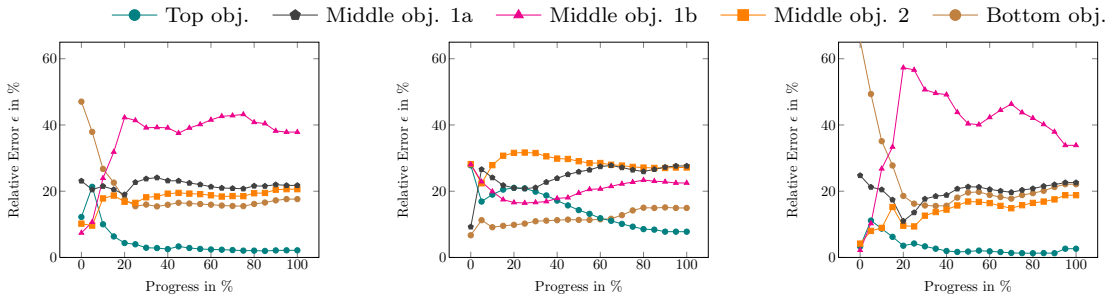
C.1.20: S–U_2; Bottom obj.(1); 55%

C.1.21: S–U_2; Bottom obj.(2); 85%

C.1.22: S–U_2; Bottom obj.(3); 40%

C.1.23: S–U_2; Bottom obj.(4); 65%

C.1.24: S–U_2; Bottom obj.(5); 85%



C.1.25: S–U_2; Top obj.(1); 10%

C.1.26: S–U_2; Top obj.(2); 65%

C.1.27: S–U_2; Top obj.(3); 15%

C.1.28: S–U_2; Top obj.(4); 40%

C.1.29: S–U_2; Top obj.(5); 30%

C.1.30: S–U_2; Middle obj.2(1); 45%

C.1.31: S–U_2; Middle obj.2(2); 25%

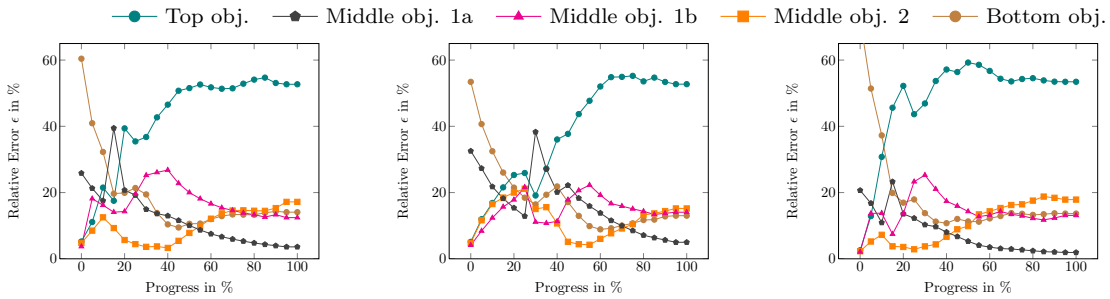
C.1.32: S–U_2; Middle obj.2(3); 30%

C.1.33: S–U_2; Middle obj.2(4); 25%

C.1.34: S–U_2; Middle obj.2(5); 35%

C.1.35: S–U_2; Middle obj.1a(1); 50%

C.1.36: S–U_2; Middle obj.1a(2); 40%



C.1.37: S–U_2; Middle obj.1a(3); 55%

C.1.38: S–U_2; Middle obj.1a(4); 75%

C.1.39: S–U_2; Middle obj.1a(5); 45%

C.1.40: S–U_2; Middle obj.1b(1); 70%

C.1.41: S–U_2; Middle obj.1b(2); 70%

C.1.42: S–U_2; Middle obj.1b(3); 60%

C.1.43: S–U_2; Middle obj.1b(5); 40%

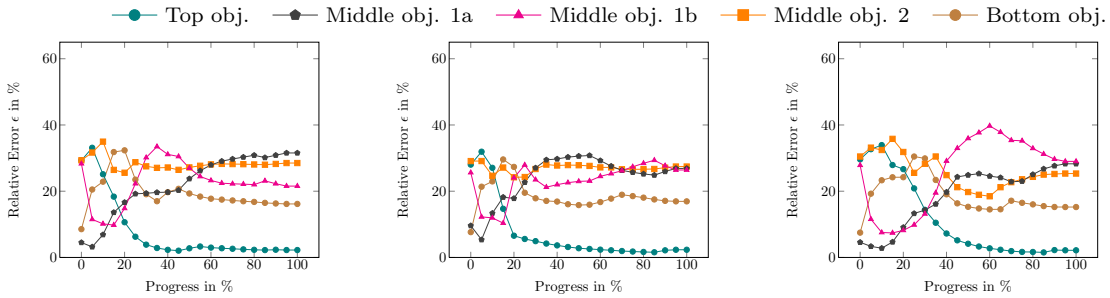
C.1.44: S–U_3; Bottom obj.(1); 45%

C.1.45: S–U_3; Bottom obj.(2); 30%

C.1.46: S–U_3; Bottom obj.(3); 45%

C.1.47: S–U_3; Bottom obj.(4); 35%

C.1.48: S–U_3; Bottom obj.(5); 0%



C.1.49: S–U$_3$; Top obj.(1); 20%

C.1.50: S–U$_3$; Top obj.(2); 20%

C.1.51: S–U$_3$; Top obj.(3); 35%

C.1.52: S–U$_3$; Top obj.(4); 35%

C.1.53: S–U$_3$; Top obj.(5); 55%

C.1.54: S–U$_3$; Middle obj.2(1); 40%

C.1.55: S–U$_3$; Middle obj.2(2); 45%

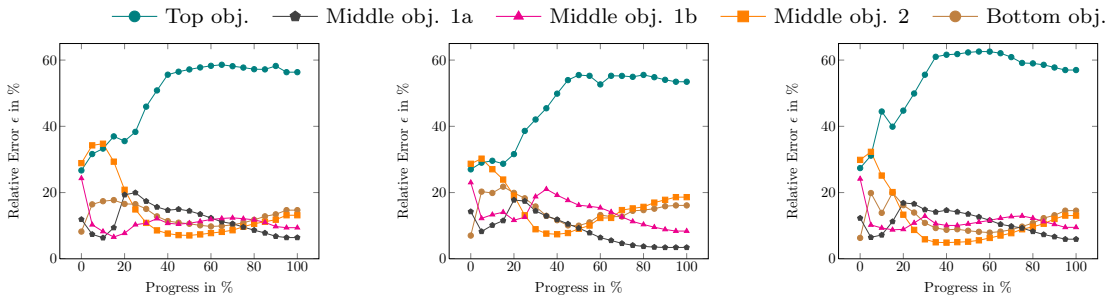
C.1.56: S–U$_3$; Middle obj.2(3); 50%

C.1.57: S–U$_3$; Middle obj.2(4); 30%

C.1.58: S–U$_3$; Middle obj.2(5); 45%

C.1.59: S–U$_3$; Middle obj.1a(1); 35%

C.1.60: S–U$_3$; Middle obj.1a(2); 80%



C.1.61: S–U_3; Middle obj.1a(3); 80%

C.1.62: S–U_3; Middle obj.1a(4); 55%

C.1.63: S–U_3; Middle obj.1a(5); 80%

C.1.64: S–U_3; Middle obj.1b(1); 25%

C.1.65: S–U_3; Middle obj.1b(2); 30%

C.1.66: S–U_3; Middle obj.1b(3); 20%

C.1.67: S–U_3; Middle obj.1b(4); 10%

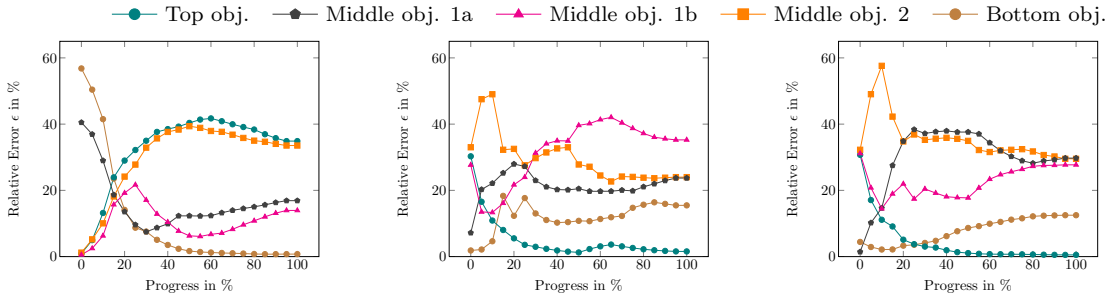
C.1.68: S–U_3; Middle obj.1b(5); 25%

C.1.69: S–U_4; Bottom obj.(1); 45%

C.1.70: S–U_4; Bottom obj.(2); 30%

C.1.71: S–U_4; Bottom obj.(3); 70%

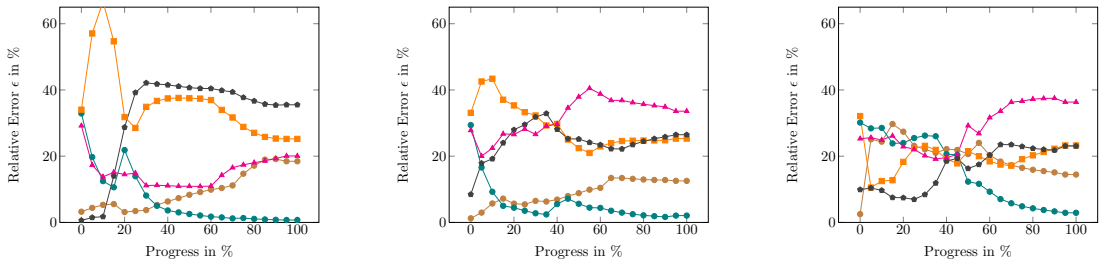
C.1.72: S–U_4; Bottom obj.(4); 30%



C.1.73: S–U_4; Bottom obj.(5); 35%

C.1.74: S–U_4; Top obj.(1); 15%

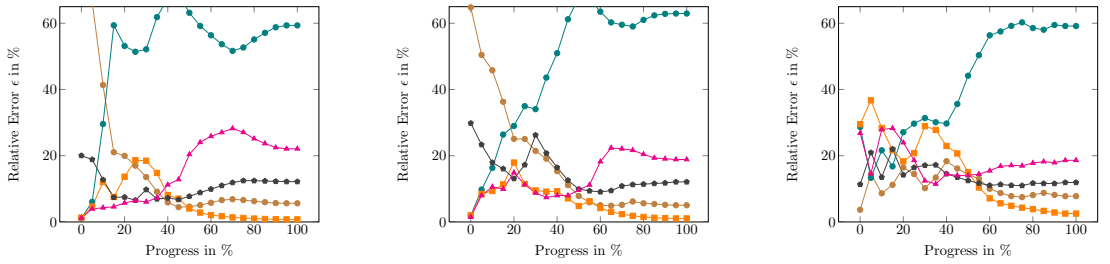
C.1.75: S–U_4; Top obj.(2); 30%



C.1.76: S–U_4; Top obj.(3); 35%

C.1.77: S–U_4; Top obj.(4); 15%

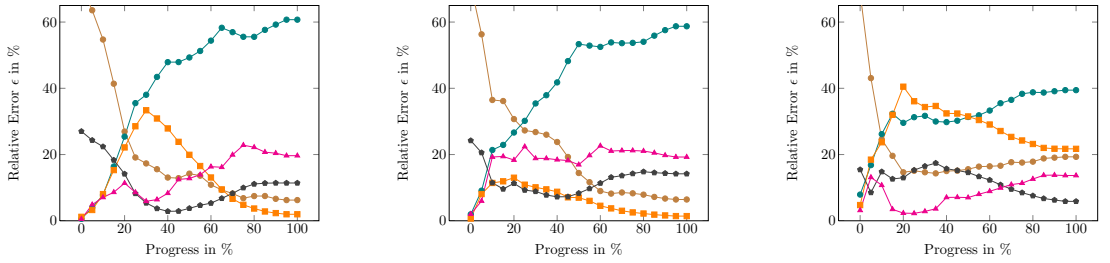
C.1.78: S–U_4; Top obj.(5); 50%



C.1.79: S–U_4; Middle obj.2(1); 50%

C.1.80: S–U_4; Middle obj.2(2); 60%

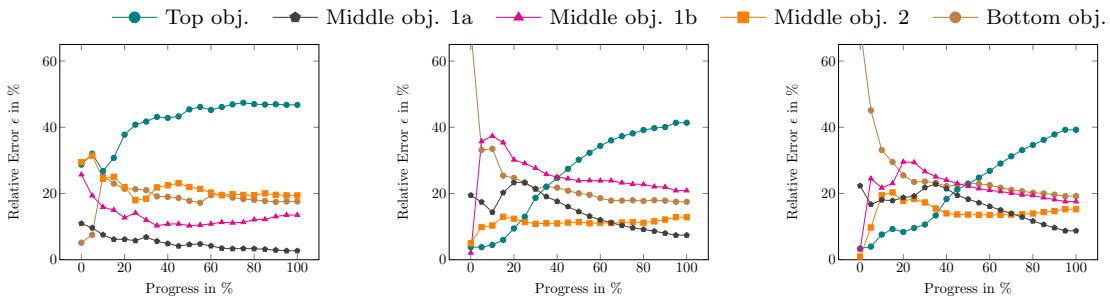
C.1.81: S–U_4; Middle obj.2(3); 55%



C.1.82: S–U_4; Middle obj.2(4); 70%

C.1.83: S–U_4; Middle obj.2(5); 50%

C.1.84: S–U_4; Middle obj.1a(1); 70%



C.1.85: S–U_4; Middle obj.1a(2); 10%

C.1.86: S–U_4; Middle obj.1a(3); 70%

C.1.87: S–U_4; Middle obj.1a(4); 75%

C.1.88: S–U_4; Middle obj.1a(5); 55%

C.1.89: S–U_4; Middle obj.1b(1); 80%

C.1.90: S–U_4; Middle obj.1b(2); 25%

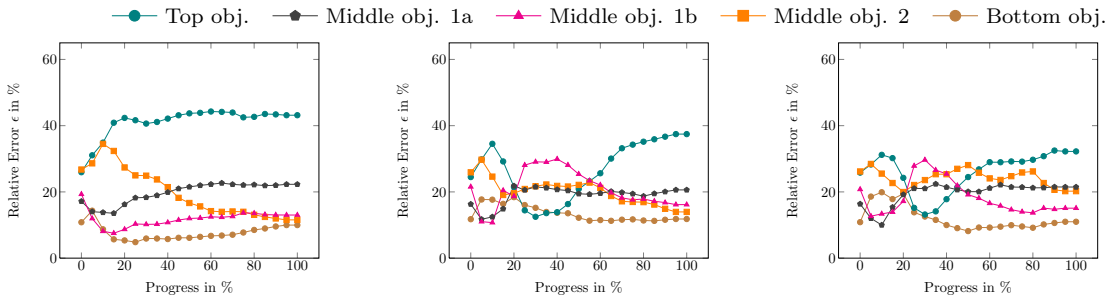
C.1.91: S–U_4; Middle obj.1b(3); 20%

C.1.92: S–U_4; Middle obj.1b(4); 20%

C.1.93: S–U_4; Middle obj.1b(5); 30%

Figure C.1.: All *successful* classifications of the multi-SDM architecture for the skilled teacher – unskilled user setup (S–U_i). The subfigure captions give additional information about (a) the teacher–user setup with respect to the particular user (subscript); (b) the task and index of the trigger cue; and (c) the duration until correct classification of the test sequence.

C.1.2. Failed Classifications



C.2.1: S–U_1; Middle obj.2(1); failed

C.2.2: S–U_1; Middle obj.2(2); failed

C.2.3: S–U_1; Middle obj.2(3); failed

C.2.4: S–U_1; Middle obj.2(4); failed

C.2.5: S–U_1; Middle obj.2(5); failed

C.2.6: S–U_1; Middle obj.1a(2); failed

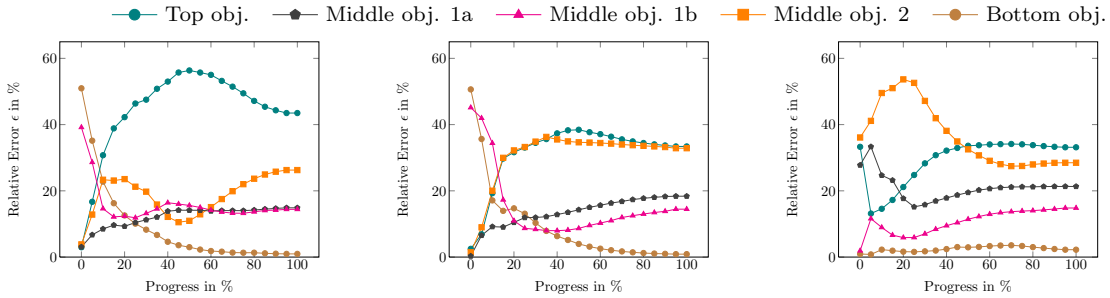
C.2.7: S–U_2; Middle obj.1b(4); failed

Figure C.2.: All *erroneous* classifications of the multi-SDM architecture for the skilled teacher – unskilled user setup (S–U_i). The subfigure captions give additional information about (a) the teacher–user setup with respect to the particular user (subscript); (b) the task and index of the trigger cue; and (c) the duration until correct classification of the test sequence.

C.2. Experiment B: Skilled Teacher–Skilled User

A randomly chosen subset, 75%, of task executions of the experienced user are considered as training set for the multi-SDM system. The remaining 25% of the set of robot arm motion trajectories from the same, experienced user are applied as test set. The motion trajectories provided by the inexperienced users are neglected within this experiment. The training-to-test ration is about 75:25 manipulation trajectories. The teacher–user relation can be described as logical `AND`. This experiment studies the ability of the multi-SDM system to predict the intention of a user when he initially provides some training. Figure C.3 shows all successful classifications, none of the test trajectories failed in this experiment.

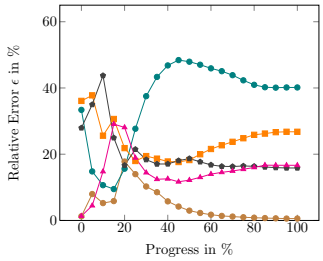
C.2.1. Successful Classifications



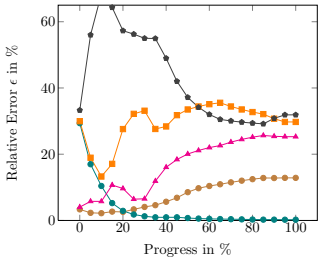
C.3.1: Bottom obj.(7); 25%

C.3.2: Bottom obj.(15); 25%

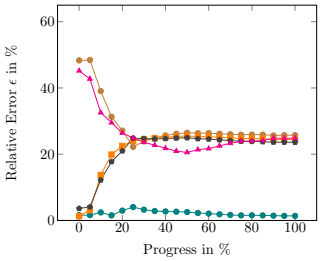
C.3.3: Bottom obj.(16); 0%



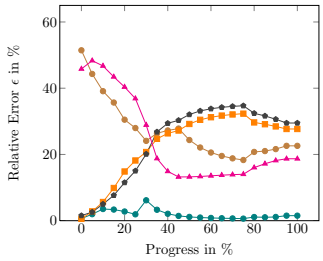
C.3.4: Bottom obj.(20); 25%



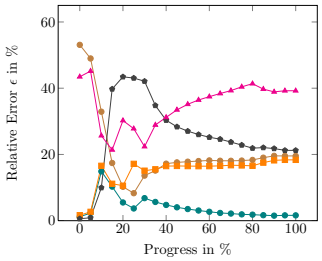
C.3.5: Top obj.(7); 25%



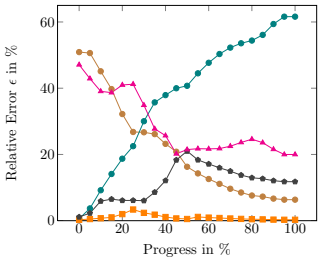
C.3.6: Top obj.(15); 5%



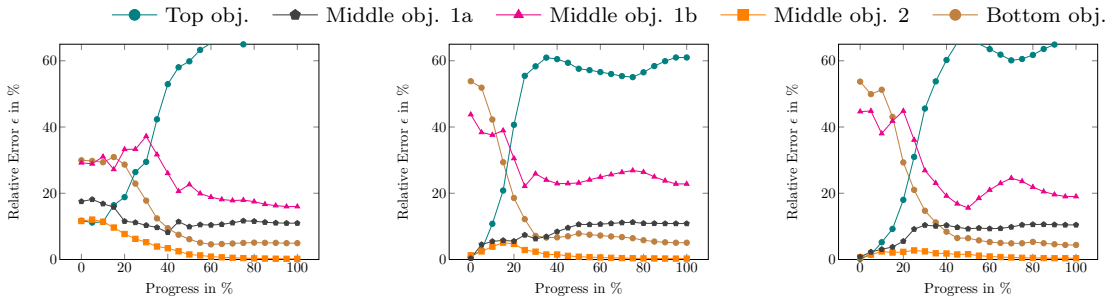
C.3.7: Top obj.(16); 5%



C.3.8: Top obj.(20); 15%



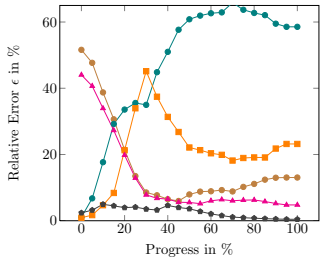
C.3.9: Middle obj.2(7); 5%



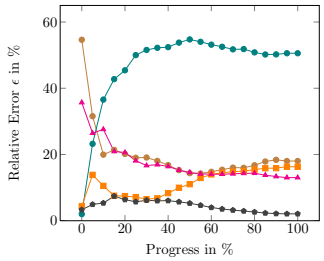
C.3.10: Middle obj.2(15); 15%

C.3.11: Middle obj.2(16); 5%

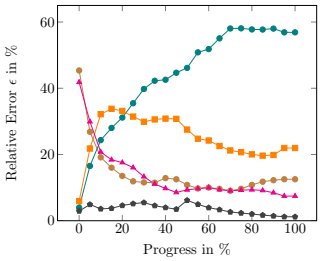
C.3.12: Middle obj.2(20); 5%



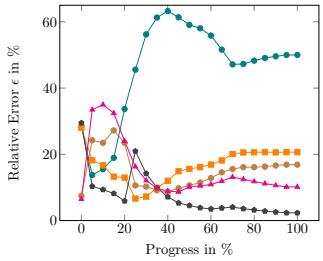
C.3.13: Middle obj.1a(7); 15%



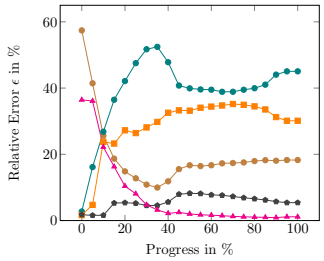
C.3.14: Middle obj.1a(15); 20%



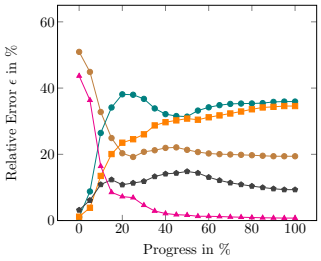
C.3.15: Middle obj.1a(16); 0%



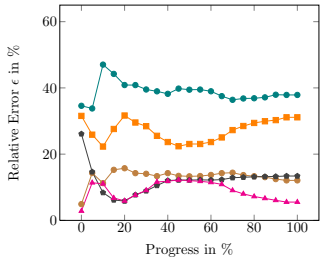
C.3.16: Middle obj.1a(20); 40%



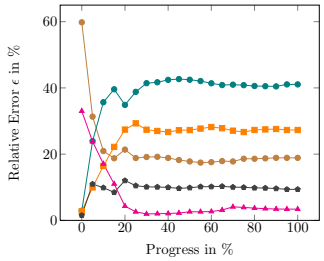
C.3.17: Middle obj.1b(7); 35%



C.3.18: Middle obj.1b(15); 15%



C.3.19: Middle obj.1b(16); 55%



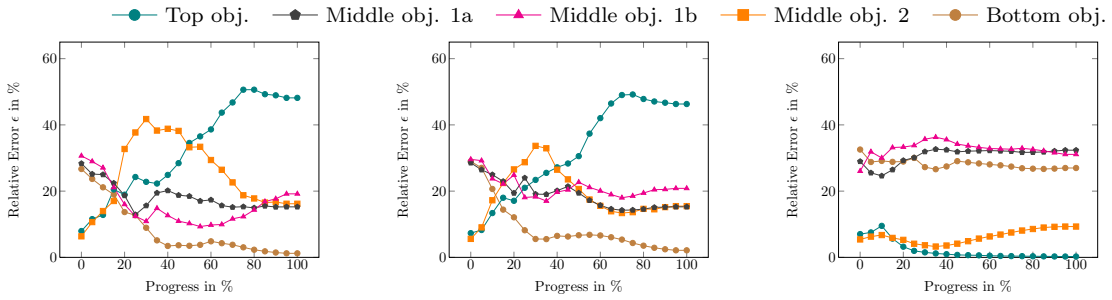
C.3.20: Middle obj.1b(20); 20%

Figure C.3.: All *successful* classifications of the multi-SDM architecture for the skilled teacher – skilled user setup (S–S). The subfigure captions give additional information about (a) the task of the trigger cue; and (b)the duration until correct classification of the test sequence.

C.3. Experiment C: Unskilled Teachers–Unskilled Users

A randomly chosen subset, 60%, of the task executions provided by the four inexperienced users are considered as training set for the multi-SDM system. For this purpose three of the five executions per task of each inexperienced user are randomly chosen to gain a balanced training of each SDM instance. The remaining 40% of the set of robot arm motion trajectories from the inexperienced user are applied as test set. The motion trajectories of the experienced user are not considered in this experiment. The training-to-test ration is about 60:40 manipulation trajectories. The teacher–user relation can be described as logical `AND`. The purpose of this experiment is to study the system's predictive behaviour when undergoing a community-based training without any experts. Figure C.4 shows all successful classifications while Figure C.5 illustrates the only failed classifications.

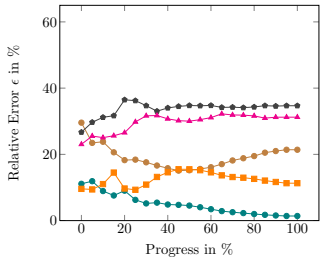
C.3.1. Successful Classifications



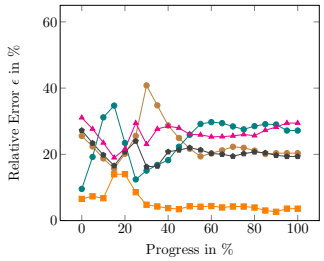
C.4.1: U–U$_1$; Bottom (3); 30%

C.4.2: U–U$_1$; Bottom (4); 15%

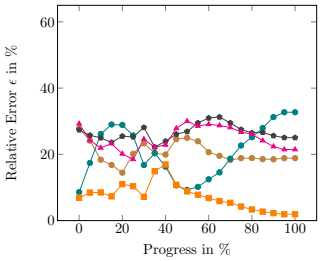
C.4.3: U–U$_1$; Top obj.(2); 20%



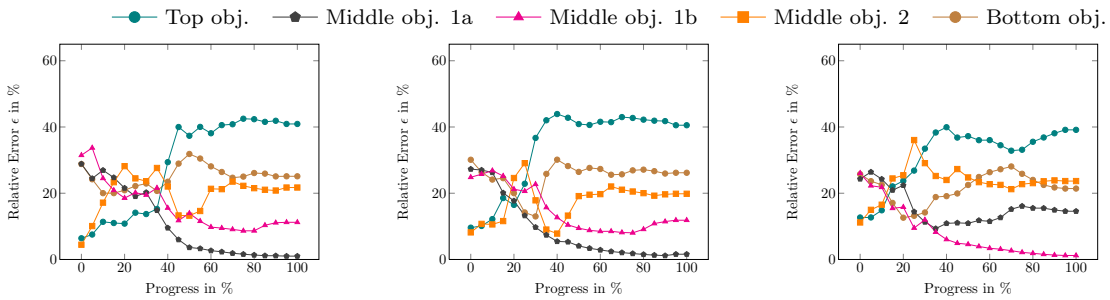
C.4.4: U–U$_1$; Top obj.(3); 10%



C.4.5: U–U$_1$; Middle obj.2(3); 0%



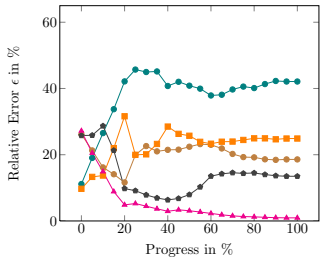
C.4.6: U–U$_1$; Middle obj.2(4); 50%



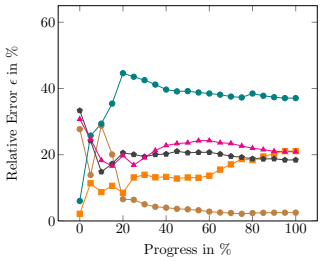
C.4.7: U–U_1; Middle obj.1a(4); 35%

C.4.8: U–U_1; Middle obj.1a(5); 25%

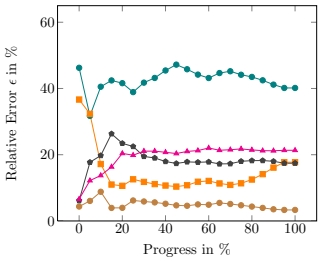
C.4.9: U–U_1; Middle obj.1b(2); 35%



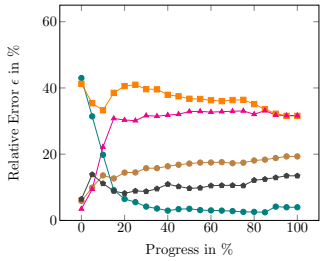
C.4.10: U–U_1; Middle obj.1b(3); 15%



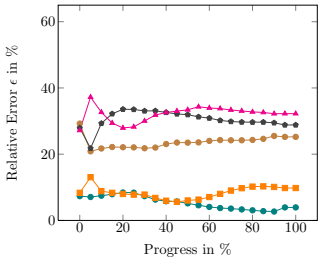
C.4.11: U–U_2; Bottom obj.(4); 20%



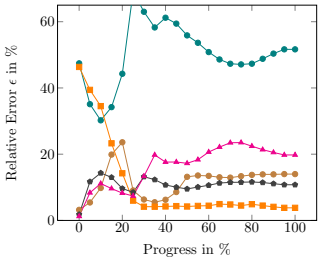
C.4.12: U–U_2; Bottom obj.(5); 0%



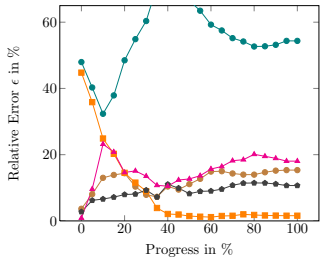
C.4.13: U–U_2; Top obj.(1); 20%



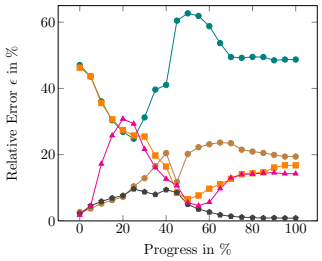
C.4.14: U–U_2; Top obj.(2); 50%



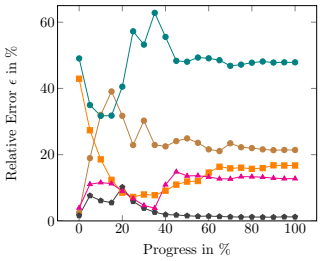
C.4.15: U–U_2; Middle obj.2(2); 25%



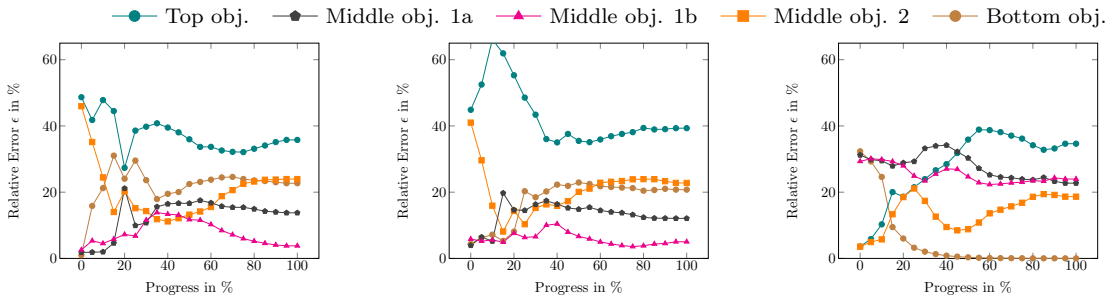
C.4.16: U–U_2; Middle obj.2(3); 35%



C.4.17: U–U_2; Middle obj.1a(4); 45%



C.4.18: U–U_2; Middle obj.1a(5); 25%



C.4.19: U–U_2; Middle obj.1b(3); 50%

C.4.20: U–U_2; Middle obj.1b(4); 20%

C.4.21: U–U_3; Bottom obj.(1); 15%

C.4.22: U–U_3; Bottom obj.(2); 15%

C.4.23: U–U_3; Top obj.(2); 5%

C.4.24: U–U_3; Top obj.(3); 10%

C.4.25: U–U_3; Middle obj.2(3); 50%

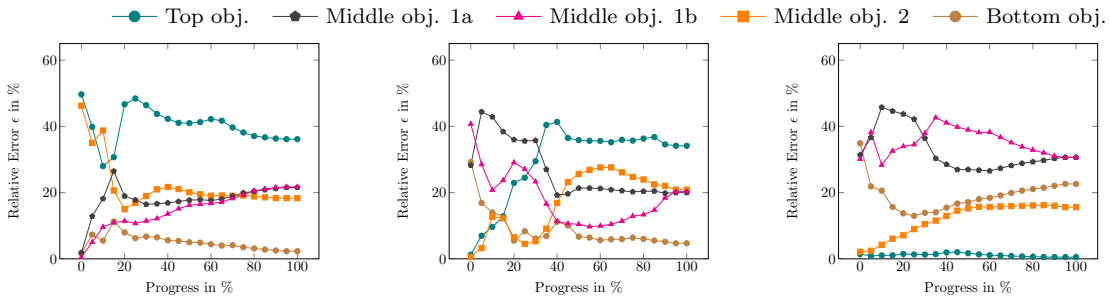
C.4.26: U–U_3; Middle obj.2(4); 40%

C.4.27: U–U_3; Middle obj.1a(3); 15%

C.4.28: U–U_3; Middle obj.1a(4); 50%

C.4.29: U–U_3; Middle obj.1b(2); 15%

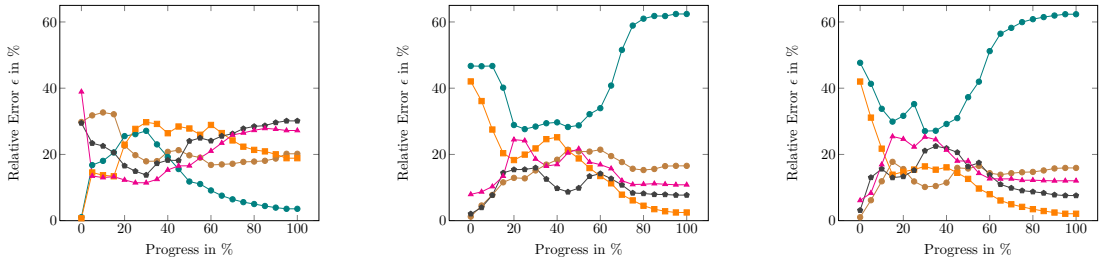
C.4.30: U–U_3; Middle obj.1b(3); 15%



C.4.31: U–U$_4$; Bottom obj.(2); 20%

C.4.32: U–U$_4$; Bottom obj.(3); 45%

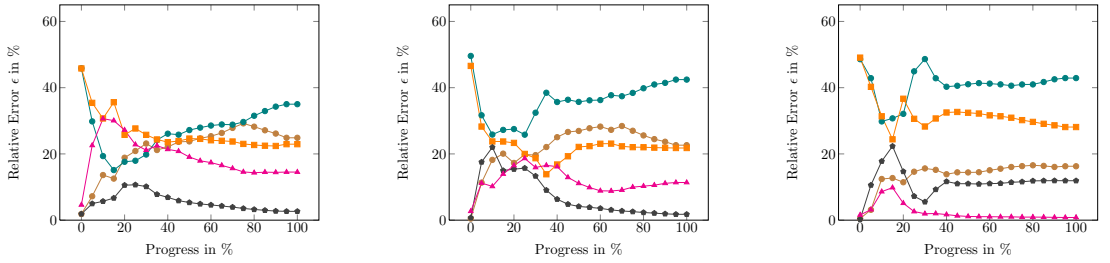
C.4.33: U–U$_4$; Top obj.(4); 0%



C.4.34: U–U$_4$; Top obj.(5); 0%

C.4.35: U–U$_4$; Middle obj.2(4); 60%

C.4.36: U–U$_4$; Middle obj.2(5); 45%



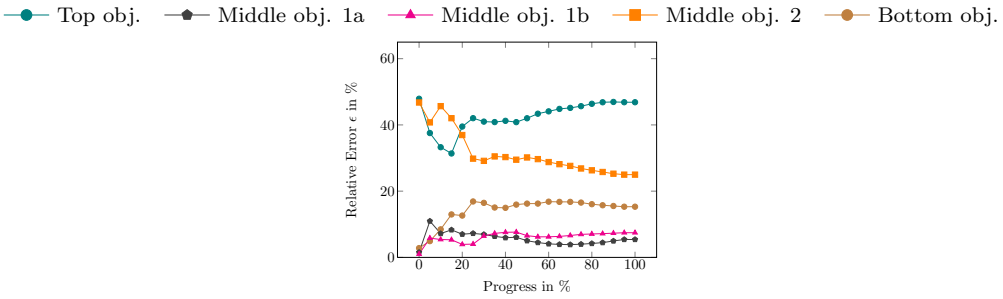
C.4.37: U–U$_4$; Middle obj.1a(4); 5%

C.4.38: U–U$_4$; Middle obj.1a(5); 20%

C.4.39: U–U$_4$; Middle obj.1b(4); 10%

Figure C.4.: All *successful* classifications of the multi-SDM architecture for the unskilled teacher – unskilled user setup (U–U$_i$). The subfigure captions give additional information about (a) the teacher–user setup with respect to the particular user (subscript); (b) the task and index of the trigger cue; and (c) the duration until correct classification of the test sequence.

C.3.2. Failed Classifications



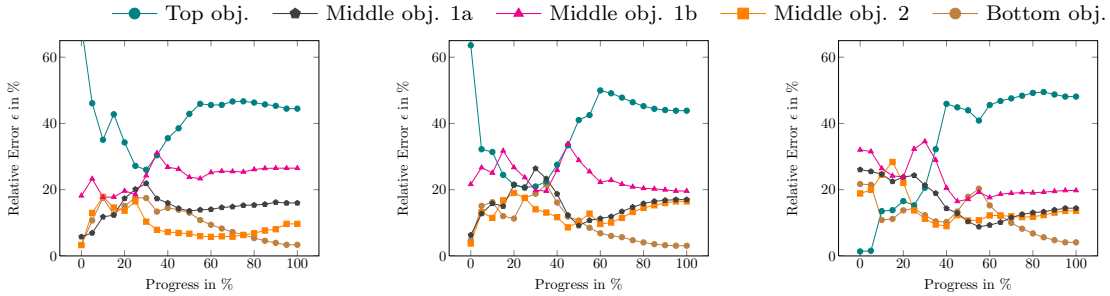
C.5.1: U–U_4; Middle obj.1b(5); failed

Figure C.5.: All *erroneous* classifications of the multi-SDM architecture for the unskilled teacher – unskilled user setup (U–U_i). The subfigure captions give additional information about (a) the teacher–user setup with respect to the particular user (subscript); (b) the task and index of the trigger cue; and (c) the duration until correct classification of the test sequence.

C.4. Experiment D: Unskilled Teachers–Skilled User

All task executions of the four inexperienced users are considered as training set for the SDM system. The robot arm motion trajectories of the remaining, experienced user are used as test set. This yields a training-to-test ration of 100:100 manipulation trajectories. The teacher–user relation can be described as logical `XOR`. This experiment studies how well the intention of an expert, that may have specialised task solving strategies, can be predicted when the system is trained by a community of laymen. Figure C.6 shows all successful classifications while Figure C.7 summarises all failed classifications.

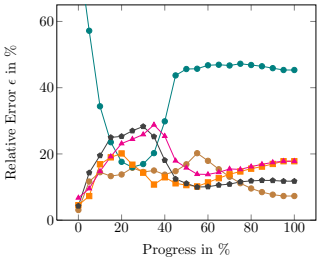
C.4.1. Successful Classifications



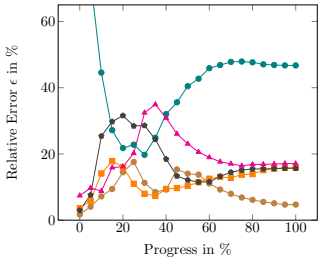
C.6.1: Bottom obj.(1); 80%

C.6.2: Bottom obj.(2); 55%

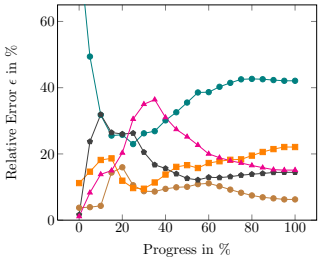
C.6.3: Bottom obj.(3); 70%



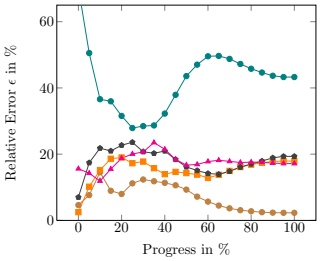
C.6.4: Bottom obj.(4); 75%



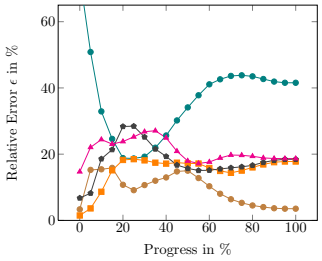
C.6.5: Bottom obj.(5); 60%



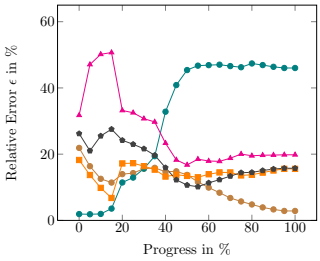
C.6.6: Bottom obj.(6); 30%



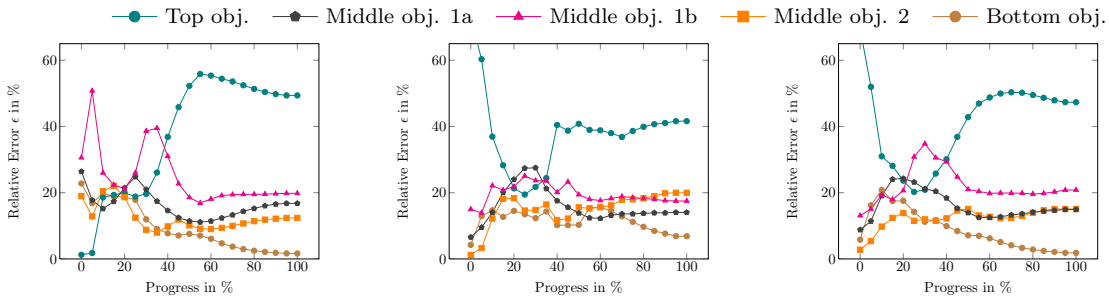
C.6.7: Bottom obj.(7); 15%



C.6.8: Bottom obj.(8); 20%



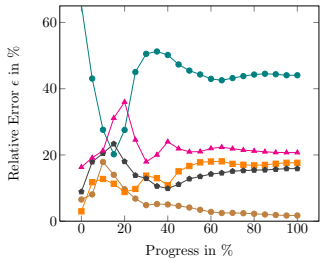
C.6.9: Bottom obj.(9); 60%



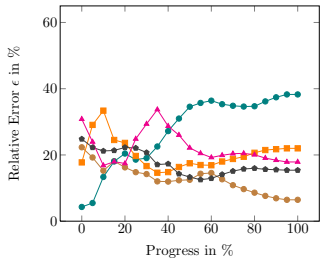
C.6.10: Bottom obj.(10); 40%

C.6.11: Bottom obj.(11); 70%

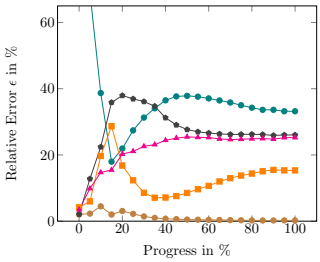
C.6.12: Bottom obj.(12); 40%



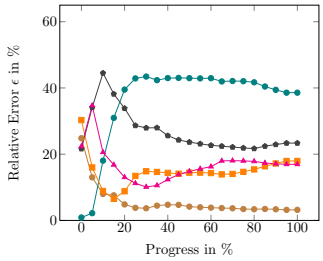
C.6.13: Bottom obj.(13); 25%



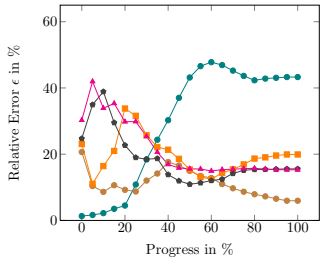
C.6.14: Bottom obj.(14); 65%



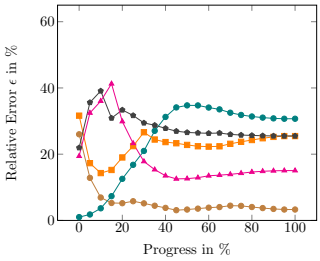
C.6.15: Bottom obj.(15); 5%



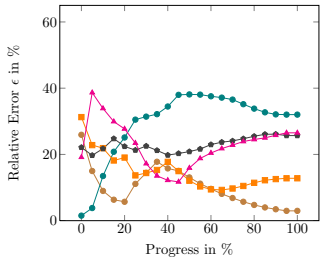
C.6.16: Bottom obj.(16); 20%



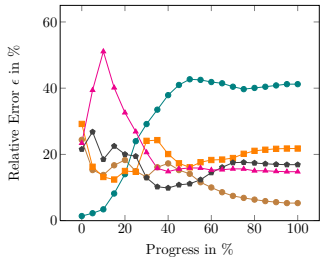
C.6.17: Bottom obj.(17); 65%



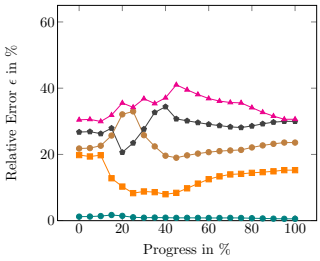
C.6.18: Bottom obj.(18); 15%



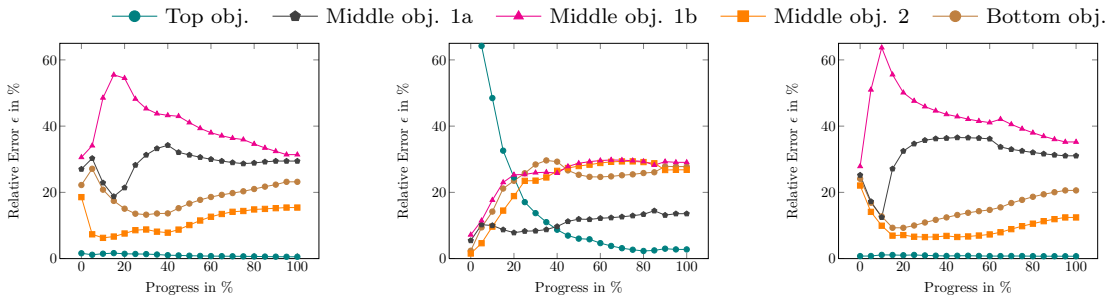
C.6.19: Bottom obj.(19); 65%



C.6.20: Bottom obj.(20); 55%



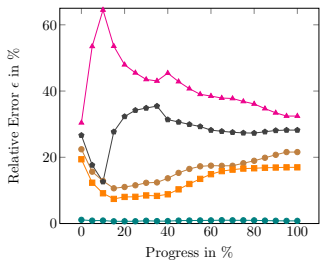
C.6.21: Top obj.(1); 0%



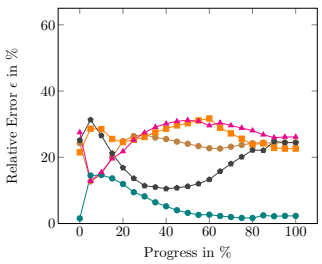
C.6.22: Top obj.(2); 0%

C.6.23: Top obj.(3); 40%

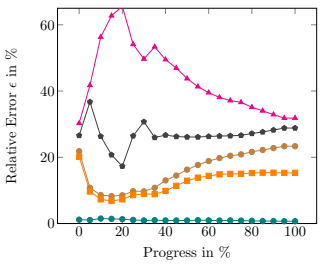
C.6.24: Top obj.(4); 0%



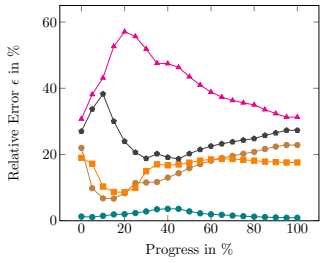
C.6.25: Top obj.(5); 0%



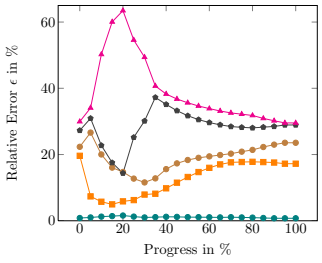
C.6.26: Top obj.(6); 15%



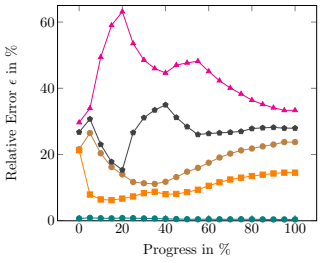
C.6.27: Top obj.(7); 0%



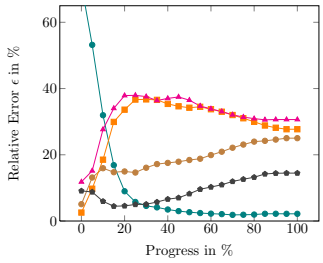
C.6.28: Top obj.(8); 0%



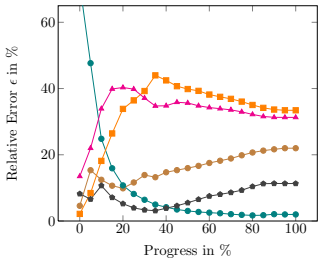
C.6.29: Top obj.(9); 0%



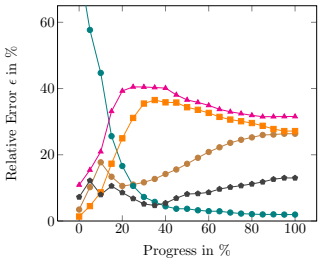
C.6.30: Top obj.(10); 0%



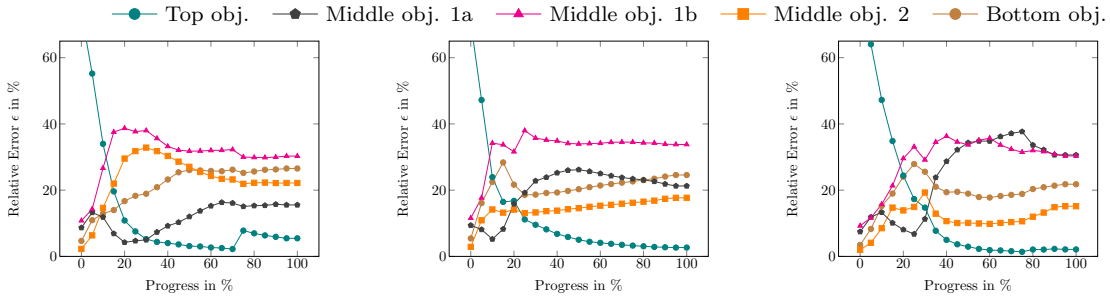
C.6.31: Top obj.(11); 35%



C.6.32: Top obj.(12); 45%



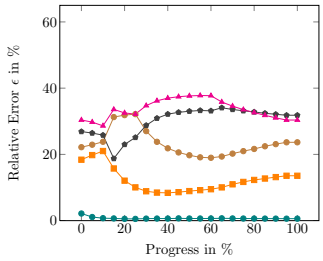
C.6.33: Top obj.(13); 40%



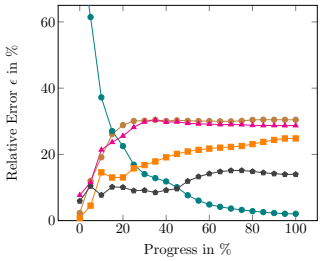
C.6.34: Top obj.(14); 35%

C.6.35: Top obj.(15); 25%

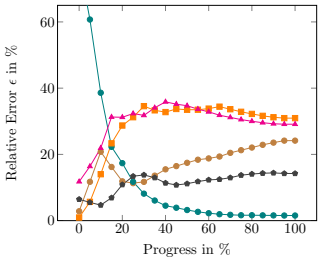
C.6.36: Top obj.(16); 35%



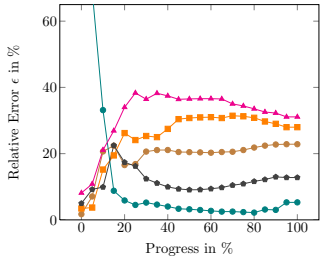
C.6.37: Top obj.(17); 0%



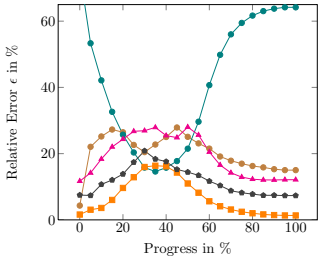
C.6.38: Top obj.(18); 50%



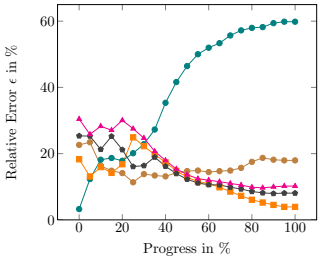
C.6.39: Top obj.(19); 30%



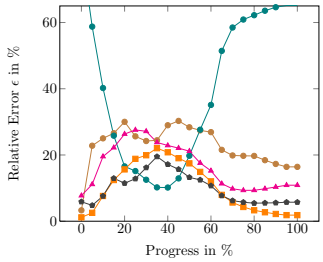
C.6.40: Top obj.(20); 15%



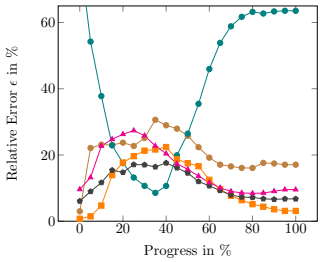
C.6.41: Middle obj.2(2); 45%



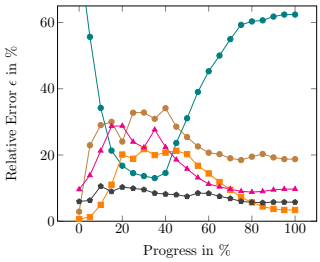
C.6.42: Middle obj.2(3); 65%



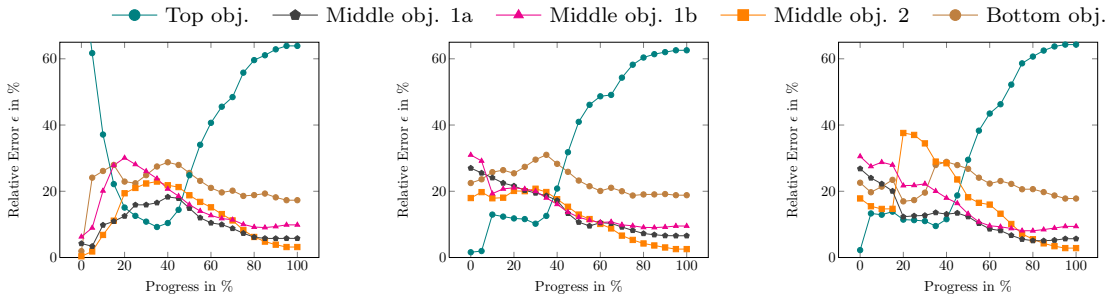
C.6.43: Middle obj.2(4); 70%



C.6.44: Middle obj.2(5); 75%



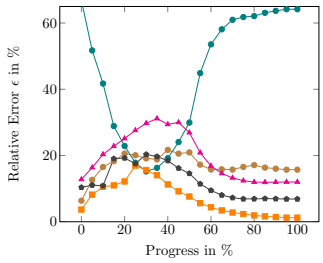
C.6.45: Middle obj.2(6); 85%



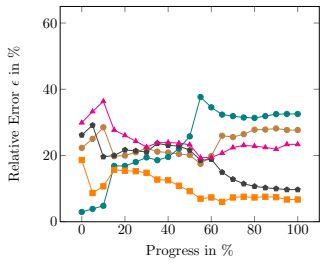
C.6.46: Middle obj.2(7); 85%

C.6.47: Middle obj.2(8); 65%

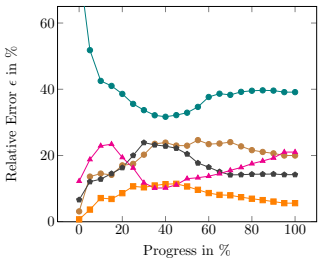
C.6.48: Middle obj.2(10); 85%



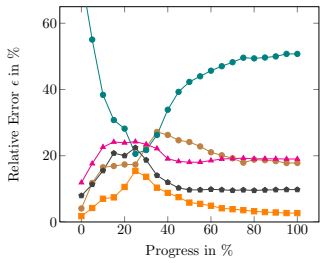
C.6.49: Middle obj.2(11); 35%



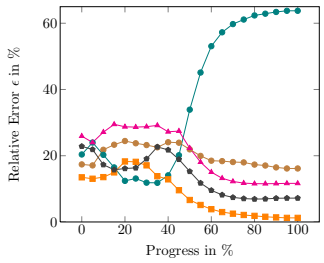
C.6.50: Middle obj.2(12); 15%



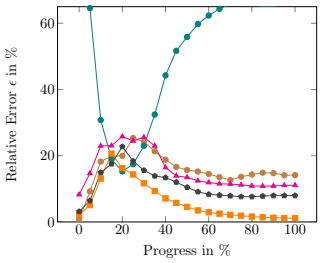
C.6.51: Middle obj.2(13); 50%



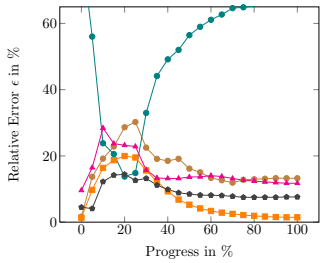
C.6.52: Middle obj.2(14); 0%



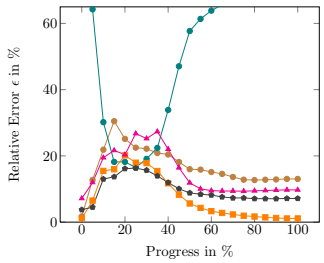
C.6.53: Middle obj.2(15); 40%



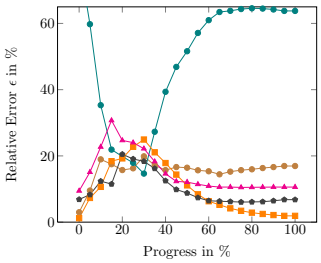
C.6.54: Middle obj.2(16); 25%



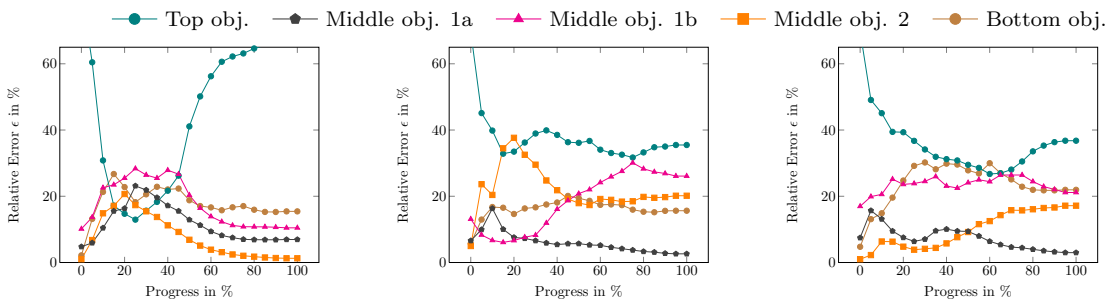
C.6.55: Middle obj.2(17); 40%



C.6.56: Middle obj.2(18); 45%



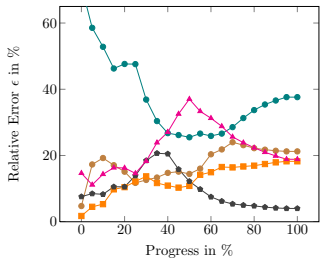
C.6.57: Middle obj.2(19); 65%



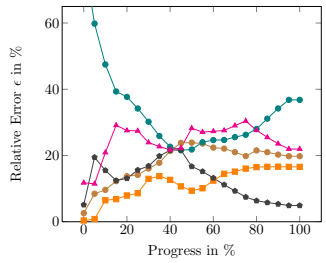
C.6.58: Middle obj.2(20); 35%

C.6.59: Middle obj.1a(1); 30%

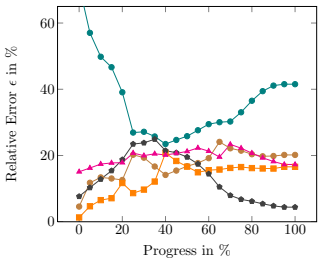
C.6.60: Middle obj.1a(2); 55%



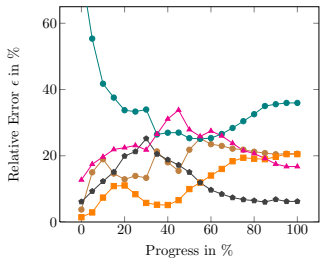
C.6.61: Middle obj.1a(3); 55%



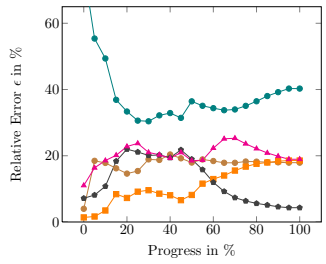
C.6.62: Middle obj.1a(4); 65%



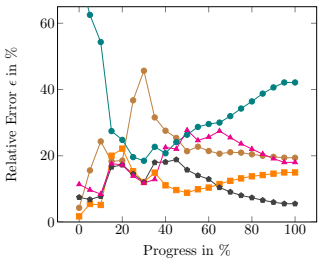
C.6.63: Middle obj.1a(5); 60%



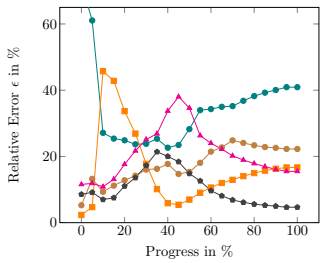
C.6.64: Middle obj.1a(6); 60%



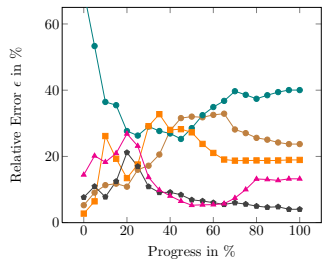
C.6.65: Middle obj.1a(7); 60%



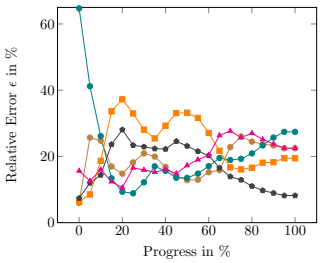
C.6.66: Middle obj.1a(8); 65%



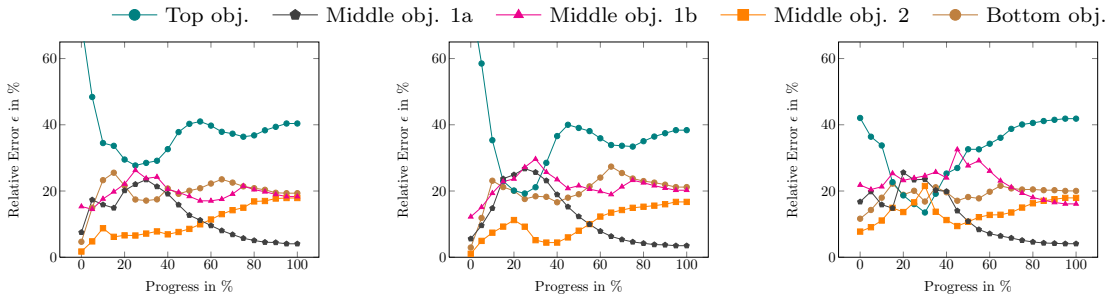
C.6.67: Middle obj.1a(9); 60%



C.6.68: Middle obj.1a(10); 65%



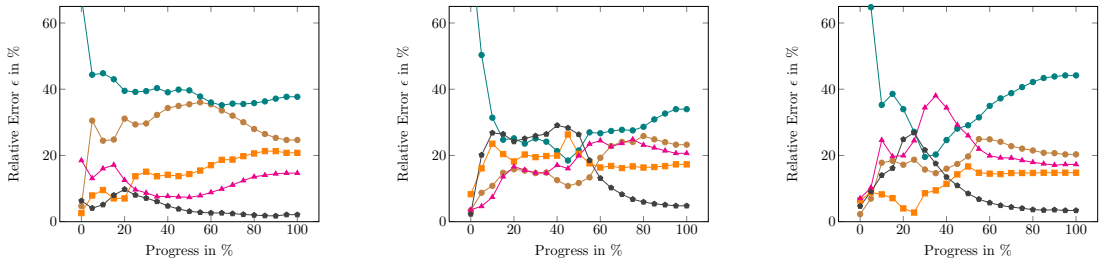
C.6.69: Middle obj.1a(11); 70%



C.6.70: Middle obj.1a(12); 60%

C.6.71: Middle obj.1a(13); 60%

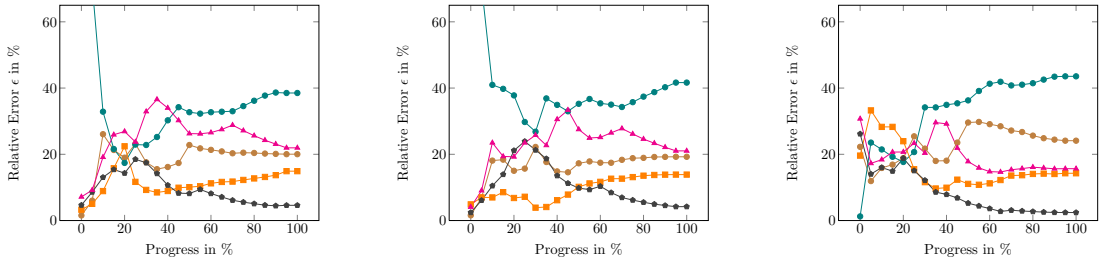
C.6.72: Middle obj.1a(14); 55%



C.6.73: Middle obj.1a(15); 25%

C.6.74: Middle obj.1a(16); 60%

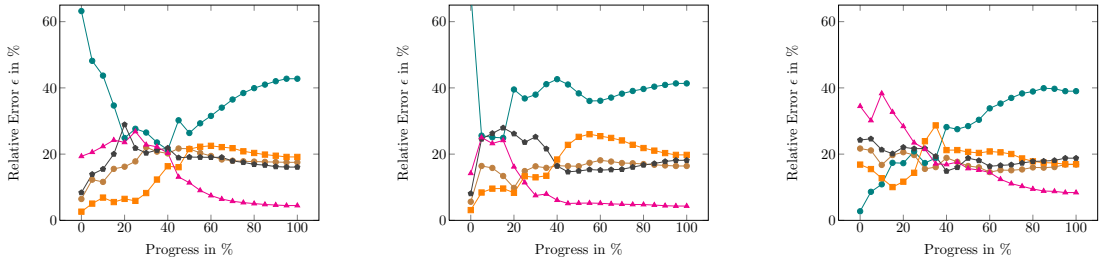
C.6.75: Middle obj.1a(17); 45%



C.6.76: Middle obj.1a(18); 45%

C.6.77: Middle obj.1a(19); 50%

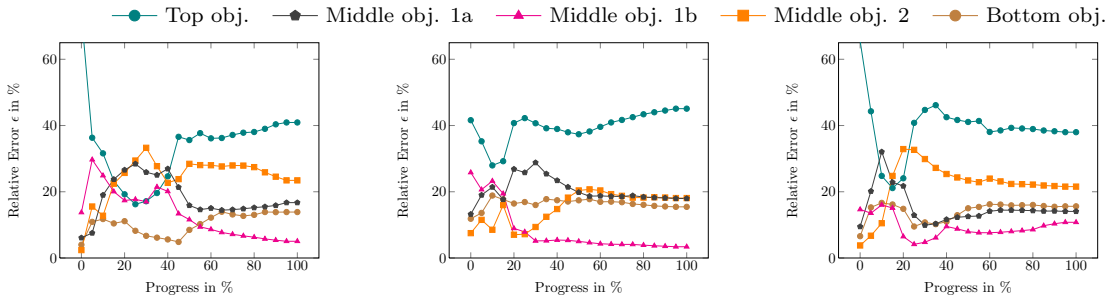
C.6.78: Middle obj.1a(20); 35%



C.6.79: Middle obj.1b(1); 45%

C.6.80: Middle obj.1b(4); 25%

C.6.81: Middle obj.1b(5); 65%



C.6.82: Middle obj.1b(8); 55%

C.6.83: Middle obj.1b(11); 30%

C.6.84: Middle obj.1b(12); 15%

C.6.85: Middle obj.1b(13); 35%

C.6.86: Middle obj.1b(14); 70%

C.6.87: Middle obj.1b(16); 10%

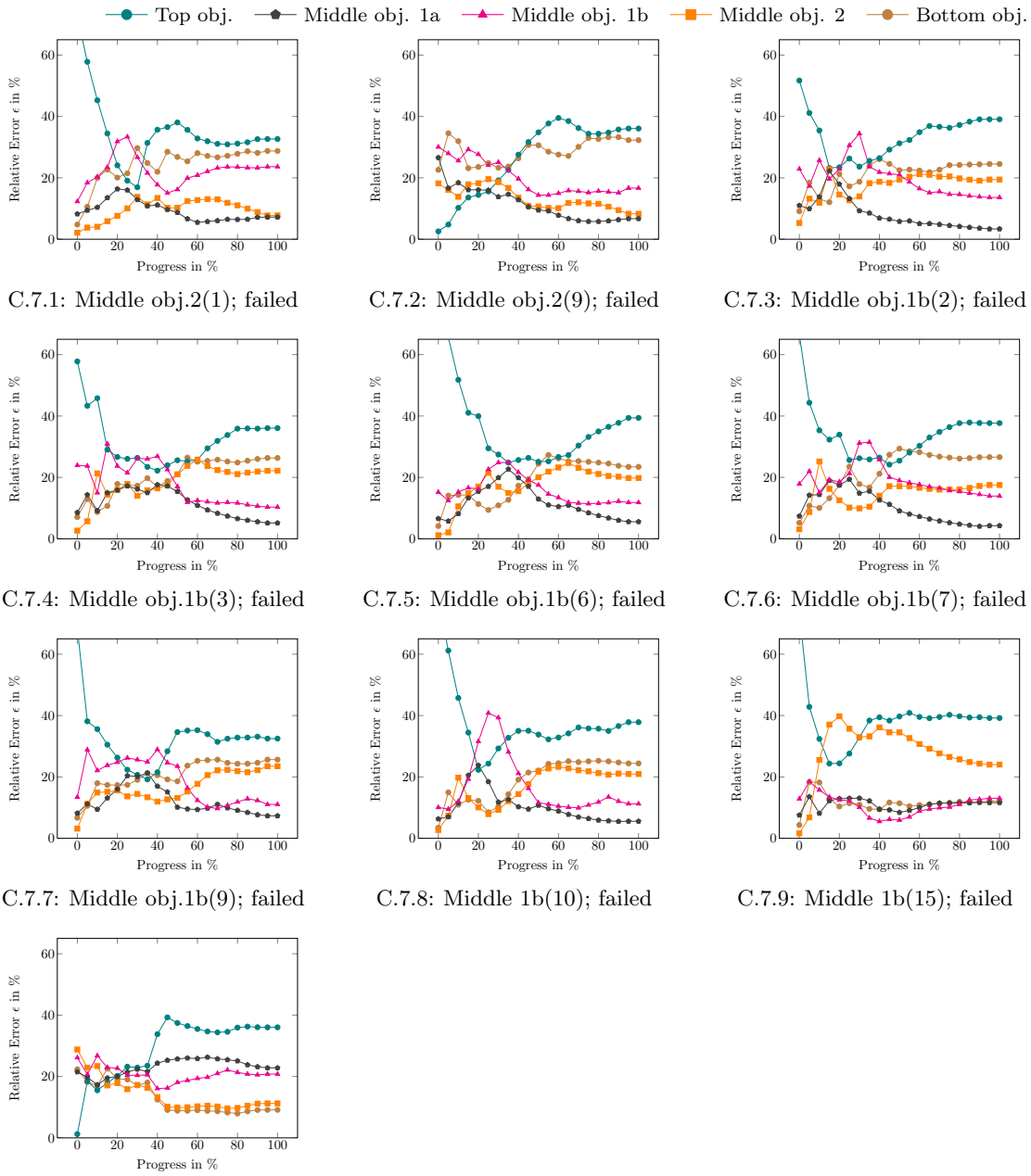
C.6.88: Middle obj.1b(17); 15%

C.6.89: Middle obj.1b(18); 55%

C.6.90: Middle obj.1b(20); 20%

Figure C.6.: All *successful* classifications of the multi-SDM architecture for the unskilled teacher – skilled user setup (U_{all}–S). The subfigure captions give additional information about (a) the task and index of the trigger cue; and (b) the duration until correct classification of the test sequence.

C.4.2. Failed Classifications



C.7.1: Middle obj.2(1); failed

C.7.2: Middle obj.2(9); failed

C.7.3: Middle obj.1b(2); failed

C.7.4: Middle obj.1b(3); failed

C.7.5: Middle obj.1b(6); failed

C.7.6: Middle obj.1b(7); failed

C.7.7: Middle obj.1b(9); failed

C.7.8: Middle 1b(10); failed

C.7.9: Middle 1b(15); failed

C.7.10: Middle 1b(19); failed

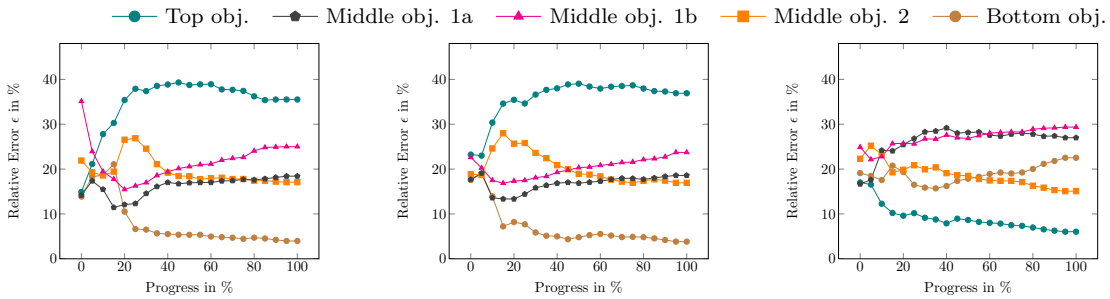
Figure C.7.: All *erroneous* classifications of the multi-SDM architecture for the unskilled teacher – skilled user setup (U_{all}–S). Subfigure captions inform about the task of the trigger cue and classification result of the test sequence.

C.5. Experiment E: Unskilled Teacher–Unskilled Users (Mutually Exclusive)

In this experiment, four evaluations are made where each inexperienced user is considered as trainer for the SDM system separately. The respective unskilled teacher does not contribute any trajectories to the test set. Several robot arm motion trajectories of the remaining inexperienced users are used as test set. Each user, except for the trainer, contributes 2 trajectories per task to the training set. This yields a training-to-test ration of 25:30 manipulation trajectories per run. The difference to the above-mentioned experiment is that the trainer–user relation is mutually exclusive and thus can be described as logical `XOR` rather than a logical `AND`.

Due to the potentially large number of diagrams obtained from this experiment, the resulting figures are summarised into two classes according to the preceding sections. Figure C.8 illustrates the successful classifications while Figure C.9 shows all misclassifications. Note that the particular indices indicate which unskilled user acted as teacher and user respectively. The total number of multi-SDM predictions for this test consists of 120 trajectories (4 runs × 30 test trajectories) that have been used as trigger cue.

C.5.1. Successful Classifications



C.8.1: U_1–U_2; Bottom obj.(4); 20%

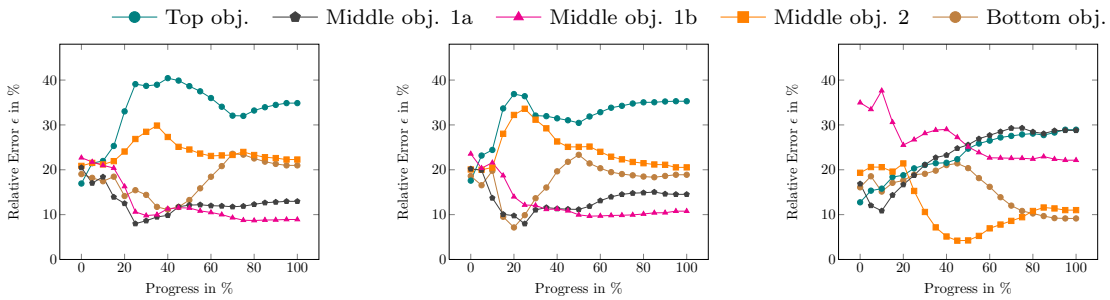
C.8.2: U_1–U_2; Bottom obj.(5); 15%

C.8.3: U_1–U_2; Top obj.(1); 10%

C.8.4: U_1–U_2; Top obj.(2); 50%

C.8.5: U_1–U_2; Middle obj.1a(4); 65%

C.8.6: U_1–U_2; Middle obj.1a(5); 45%



C.8.7: U_1–U_2; Middle obj.1b(3); 50%

C.8.8: U_1–U_2; Middle obj.1b(4); 45%

C.8.9: U_1–U_3; Bottom obj.(1); 80%

C.8.10: U_1–U_3; Top obj.(2); 15%

C.8.11: U_1–U_3; Top obj.(3); 40%

C.8.12: U_1–U_3; Middle obj.1a(3); 15%

C.8.13: U_1–U_3; Middle obj.1a(4); 55%

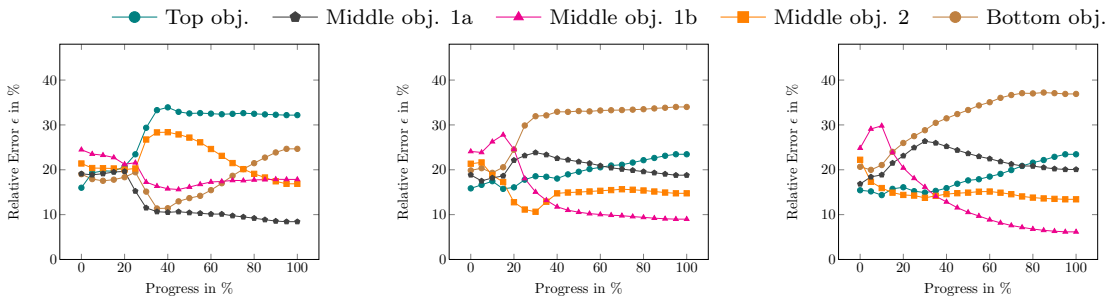
C.8.14: U_1–U_3; Middle obj.1b(3); 40%

C.8.15: U_1–U_4; Bottom obj.(2); 85%

C.8.16: U_1–U_4; Top obj.(4); 30%

C.8.17: U_1–U_4; Top obj.(5); 20%

C.8.18: U_1–U_4; Middle obj.1a(4); 55%



C.8.19: U_1–U_4; Middle obj.1a(5); 25%

C.8.20: U_1–U_4; Middle obj.1b(4); 40%

C.8.21: U_1–U_4; Middle obj.1b(5); 35%

C.8.22: U_2–U_1; Bottom obj.(3); 75%

C.8.23: U_2–U_1; Bottom obj.(4); 10%

C.8.24: U_2–U_1; Top obj.(2); 15%

C.8.25: U_2–U_1; Top obj.(3); 0%

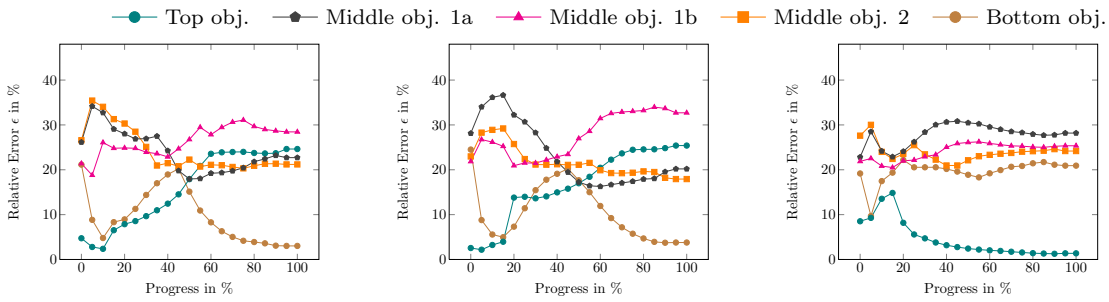
C.8.26: U_2–U_1; Middle obj.2(4); 90%

C.8.27: U_2–U_1; Middle obj.1a(4); 60%

C.8.28: U_2–U_1; Middle obj.1a(5); 80%

C.8.29: U_2–U_1; Middle obj.1b(2); 40%

C.8.30: U_2–U_1; Middle obj.1b(3); 20%



C.8.31: U_2–U_3; Bottom obj.(1); 50%

C.8.32: U_2–U_3; Bottom obj.(2); 55%

C.8.33: U_2–U_3; Top obj.(2); 0%

C.8.34: U_2–U_3; Top obj.(3); 20%

C.8.35: U_2–U_3; Middle obj.2(3); 95%

C.8.36: U_2–U_3; Middle obj.2(4); 55%

C.8.37: U_2–U_3; Middle obj.1a(3); 85%

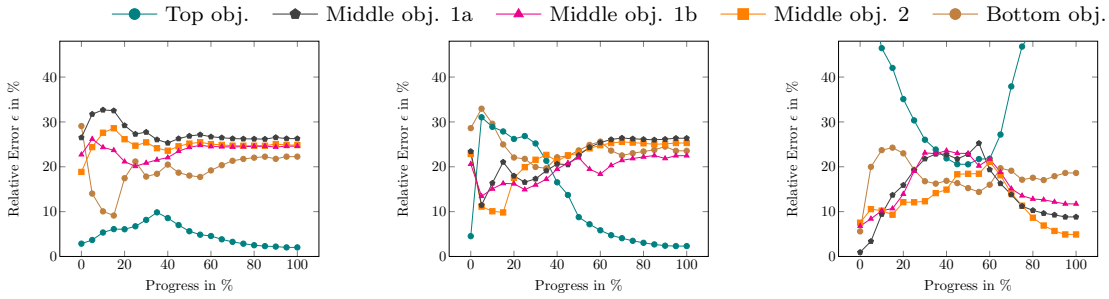
C.8.38: U_2–U_3; Middle obj.1a(4); 35%

C.8.39: U_2–U_3; Middle obj.1b(2); 30%

C.8.40: U_2–U_3; Middle obj.1b(3); 45%

C.8.41: U_2–U_4; Bottom obj.(2); 30%

C.8.42: U_2–U_4; Bottom obj.(3); 45%



C.8.43: U_2–U_4; Top obj.(4); 0%

C.8.44: U_2–U_4; Top obj.(5); 40%

C.8.45: U_2–U_4; Middle obj.2(4); 80%

C.8.46: U_2–U_4; Middle obj.2(5); 75%

C.8.47: U_2–U_4; Middle obj.1a(4); 50%

C.8.48: U_2–U_4; Middle obj.1a(5); 45%

C.8.49: U_2–U_4; Middle obj.1b(4); 70%

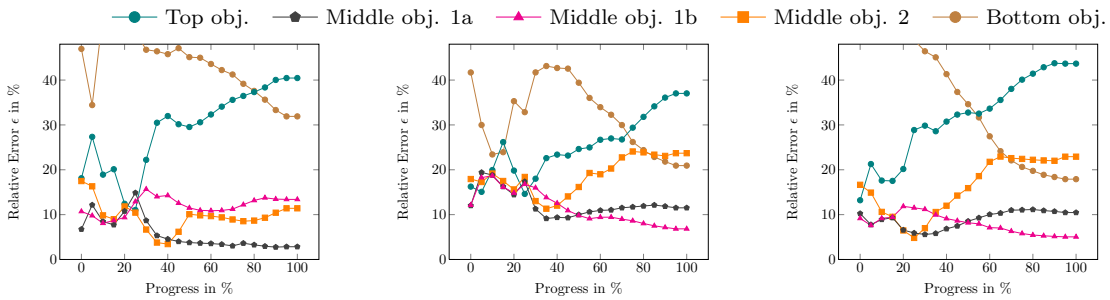
C.8.50: U_2–U_4; Middle obj.1b(5); 75%

C.8.51: U_3–U_1; Bottom obj.(4); 95%

C.8.52: U_3–U_1; Top obj.(2); 10%

C.8.53: U_3–U_1; Top obj.(3); 10%

C.8.54: U_3–U_1; Middle obj.1a(4); 50%



C.8.55: U_3–U_1; Middle obj.1a(5); 45%

C.8.56: U_3–U_1; Middle obj.1b(2); 50%

C.8.57: U_3–U_1; Middle obj.1b(3); 50%

C.8.58: U_3–U_2; Bottom obj.(4); 80%

C.8.59: U_3–U_2; Top obj.(1); 10%

C.8.60: U_3–U_2; Top obj.(2); 35%

C.8.61: U_3–U_2; Middle obj.1a(4); 45%

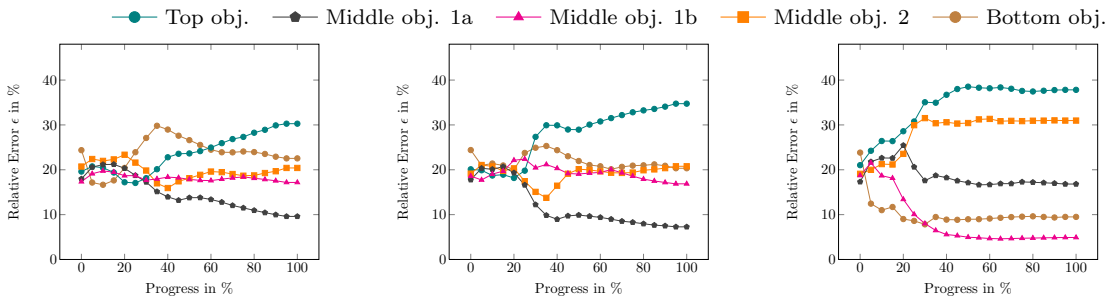
C.8.62: U_3–U_2; Middle obj.1a(5); 25%

C.8.63: U_3–U_4; Bottom obj.(2); 10%

C.8.64: U_3–U_4; Bottom obj.(3); 50%

C.8.65: U_3–U_4; Top obj.(4); 20%

C.8.66: U_3–U_4; Top obj.(5); 45%



C.8.67: U_3–U_4; Middle obj.1a(4); 30%

C.8.68: U_3–U_4; Middle obj.1a(5); 25%

C.8.69: U_3–U_4; Middle obj.1b(4); 35%

C.8.70: U_3–U_4; Middle obj.1b(5); 15%

C.8.71: U_4–U_1; Bottom obj.(3); 95%

C.8.72: U_4–U_1; Bottom obj.(4); 90%

C.8.73: U_4–U_1; Top obj.(2); 0%

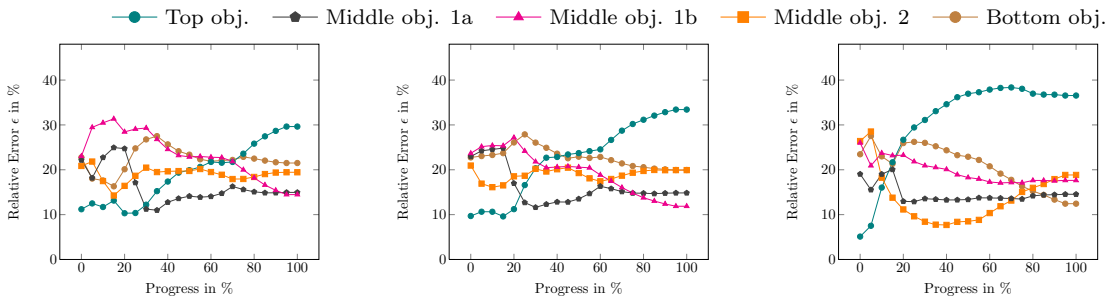
C.8.74: U_4–U_1; Top obj.(3); 0%

C.8.75: U_4–U_1; Middle obj.2(3); 85%

C.8.76: U_4–U_1; Middle obj.2(4); 75%

C.8.77: U_4–U_1; Middle obj.1a(4); 50%

C.8.78: U_4–U_1; Middle obj.1a(5); 35%



C.8.79: U_4–U_1; Middle obj.1b(2); 95%

C.8.80: U_4–U_1; Middle obj.1b(3); 80%

C.8.81: U_4–U_2; Bottom obj.(4); 90%

C.8.82: U_4–U_2; Top obj.(1); 85%

C.8.83: U_4–U_2; Top obj.(2); 35%

C.8.84: U_4–U_2; Middle obj.2(2); 30%

C.8.85: U_4–U_2; Middle obj.2(3); 40%

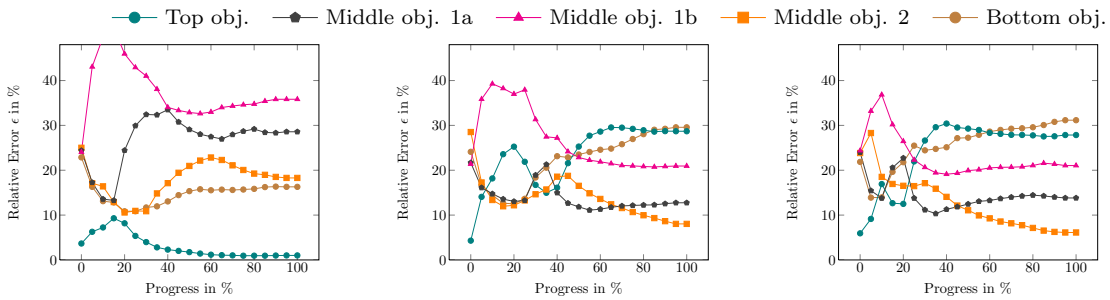
C.8.86: U_4–U_2; Middle obj.1a(4); 65%

C.8.87: U_4–U_2; Middle obj.1a(5); 45%

C.8.88: U_4–U_3; Bottom obj.(1); 10%

C.8.89: U_4–U_3; Bottom obj.(2); 15%

C.8.90: U_4–U_3; Top obj.(2); 0%



C.8.91: U_4–U_3; Top obj.(3); 0%

C.8.92: U_4–U_3; Middle obj.2(3); 70%

C.8.93: U_4–U_3; Middle obj.2(4); 50%

C.8.94: U_4–U_3; Middle obj.1a(3); 25%

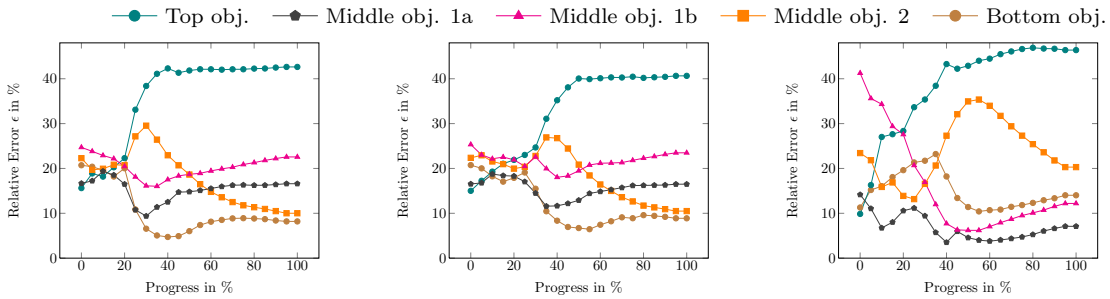
C.8.95: U_4–U_3; Middle obj.1a(4); 60%

C.8.96: U_4–U_3; Middle obj.1b(2); 65%

C.8.97: U_4–U_3; Middle obj.1b(3); 80%

Figure C.8.: All *successful* classifications of the multi-SDM architecture for the mutually exclusive unskilled teacher – unskilled user setup (U_i–U_j, with $i \neq j$). The subfigure captions give additional information about (a) the teacher–user setup with respect to the particular teacher and particular user; (b) the task and index of the trigger cue; and (c) the duration until correct classification of the test sequence.

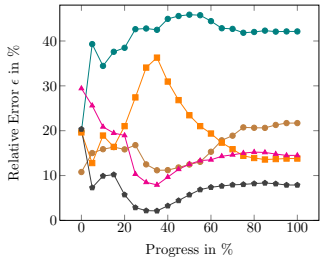
C.5.2. Failed Classifications



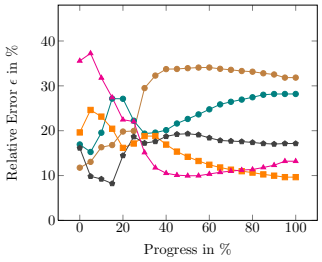
C.9.1: U_1–U_2; Middle obj.2(2)

C.9.2: U_1–U_2; Middle obj.2(3)

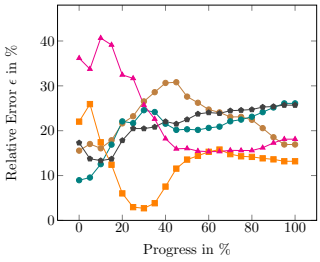
C.9.3: U_1–U_3; Middle obj.2(3)



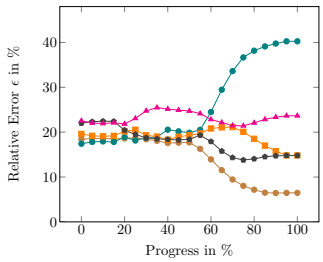
C.9.4: U_1–U_3; Middle obj.2(4)



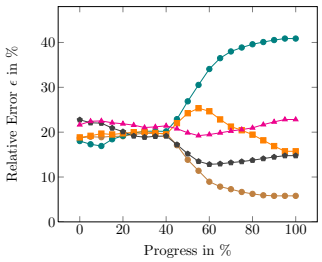
C.9.5: U_1–U_3; Middle obj.1b(2)



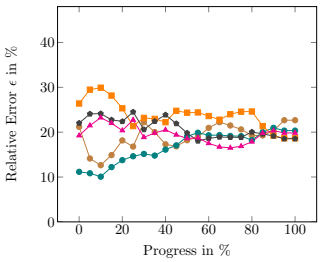
C.9.6: U_1–U_4; Bottom obj.(3)



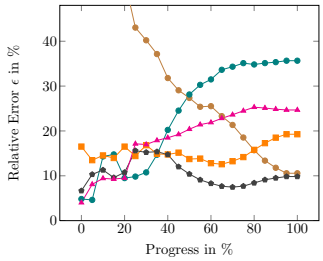
C.9.7: U_1–U_4; Middle obj.2(4)



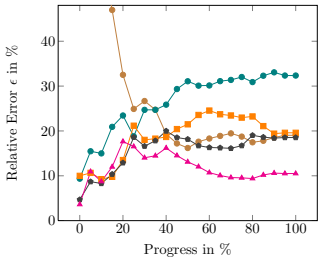
C.9.8: U_1–U_4; Middle obj.2(5)



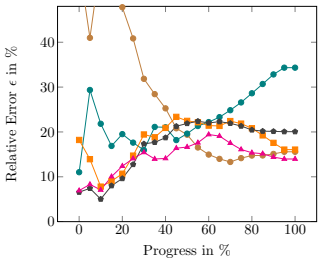
C.9.9: U_2–U_1; Middle obj.2(3)



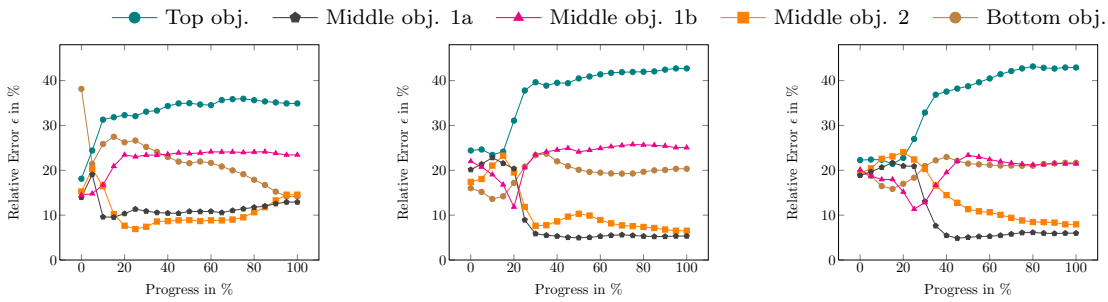
C.9.10: U_3–U_1; Bottom obj.(3)



C.9.11: U_3–U_1; Middle obj.2(3)



C.9.12: U_3–U_1; Middle obj.2(4)



C.9.13: U_3–U_2; Bottom obj.(5)

C.9.14: U_3–U_2; Middle obj.2(2)

C.9.15: U_3–U_2; Middle obj.2(3)

C.9.16: U_3–U_2; Middle obj.1b(3)

C.9.17: U_3–U_2; Middle obj.1b(4)

C.9.18: U_3–U_4; Middle obj.2(4)

C.9.19: U_3–U_4; Middle obj.2(5)

C.9.20: U_4–U_2; Bottom obj.(5)

C.9.21: U_4–U_2; Middle obj.1a(4)

Figure C.9.: All *erroneous* classifications of the multi-SDM architecture for the mutually exclusive unskilled teacher – unskilled user setup (U_i–U_j, with $i \neq j$). The subfigure captions give additional information about (a) the teacher–user setup with respect to the particular teacher and particular user; (b) the task and index of the trigger cue.

D

Crossmodal Robot Localisation

This appendix complements Chapter 8 by presenting the remaining diagrams for the multi-SDM-based crossmodal robot localisation. The diagrams illustrate the robot's belief of where it is located in an office environment. The localisation is based on past experience stored in a multi-SDM architecture.

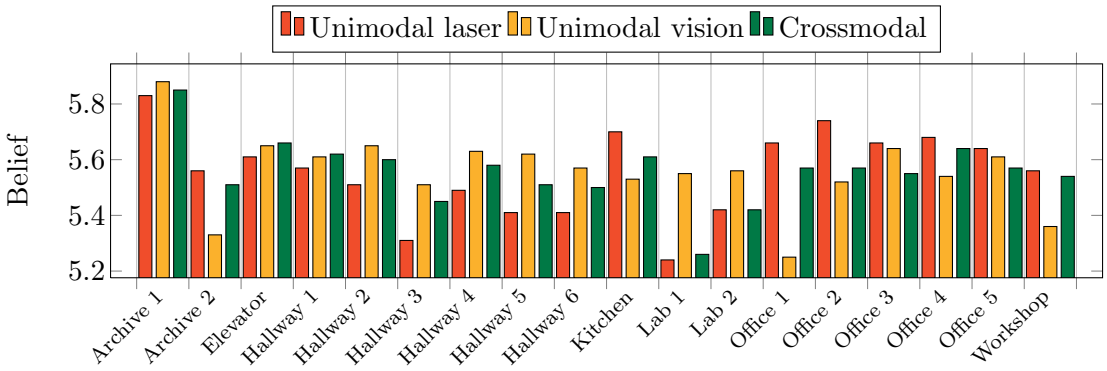


Figure D.1.: The robot's belief if triggered with pattern *Archive 1*.

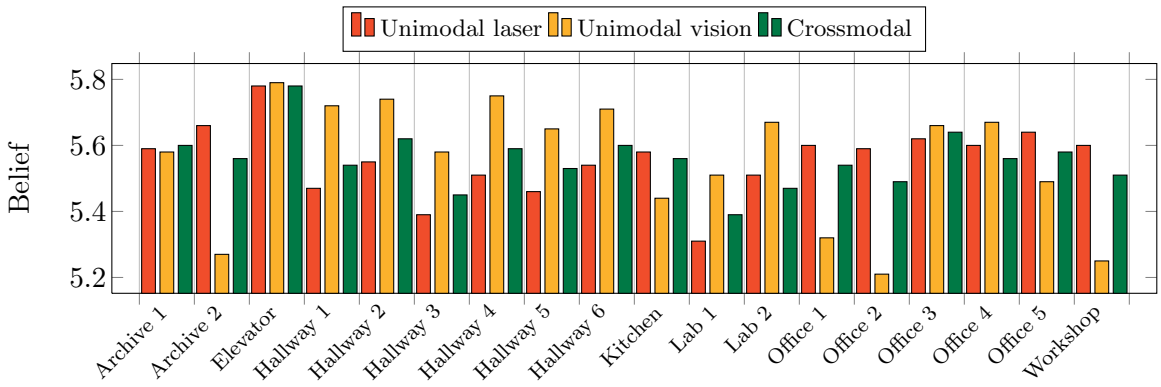


Figure D.2.: The robot's belief if triggered with pattern *Elevator*.

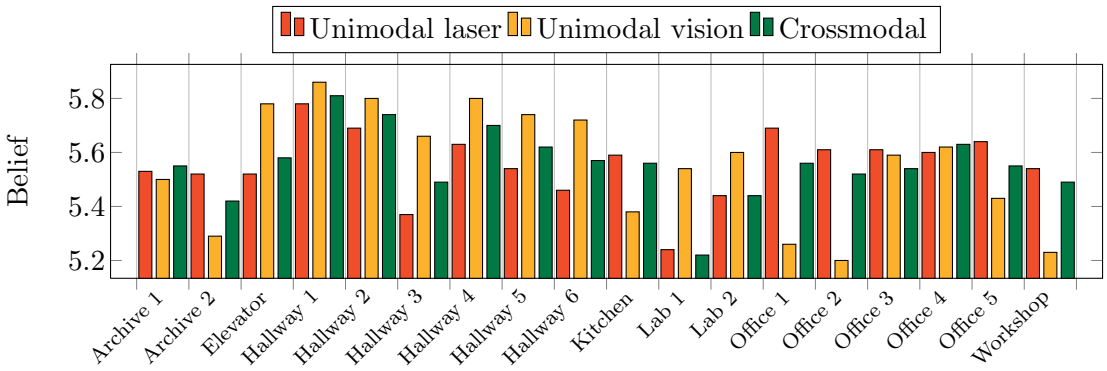


Figure D.3.: The robot's belief if triggered with pattern *Hallway 1*.

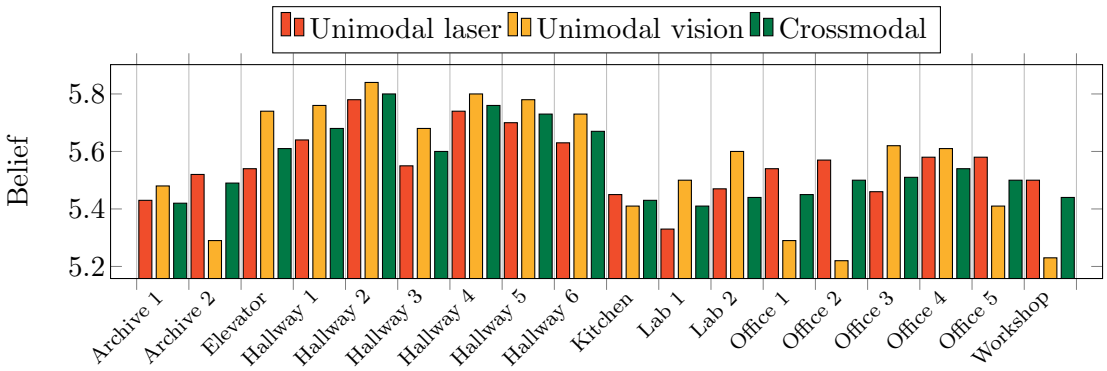


Figure D.4.: The robot's belief if triggered with pattern *Hallway 2*.

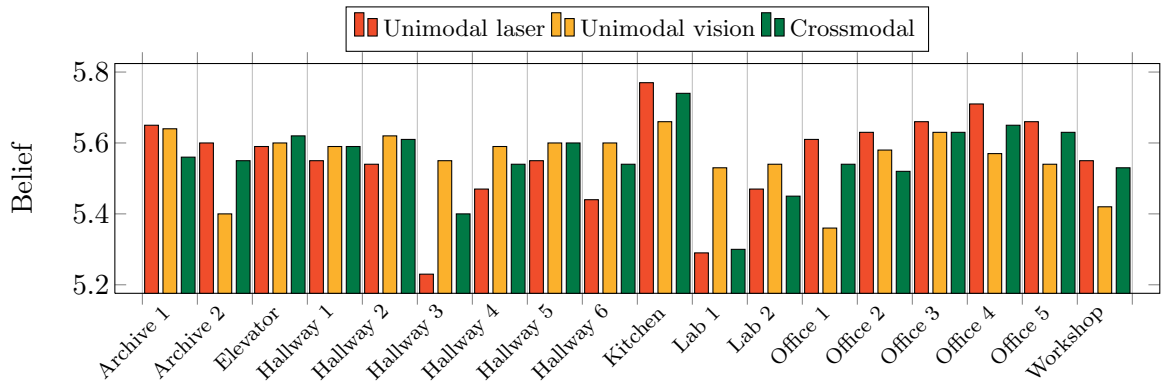


Figure D.5.: The robot's belief if triggered with pattern *Kitchen*.

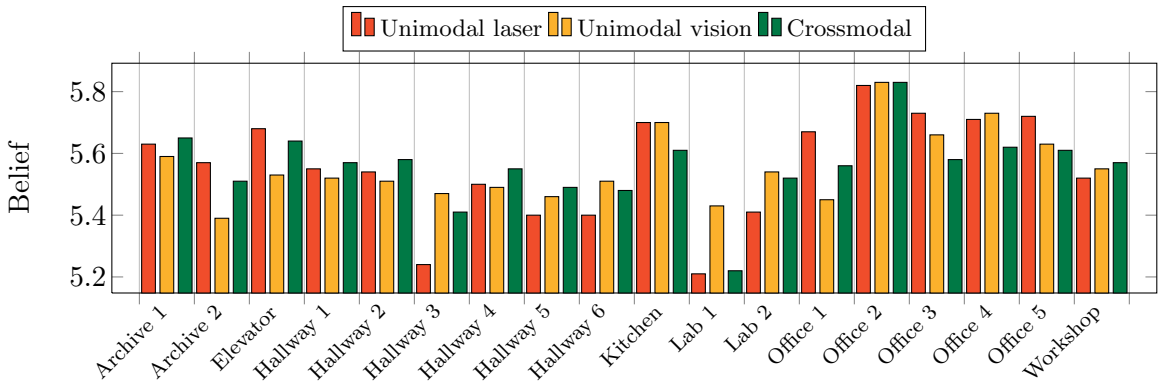


Figure D.6.: The robot's belief if triggered with pattern *Office 2*.

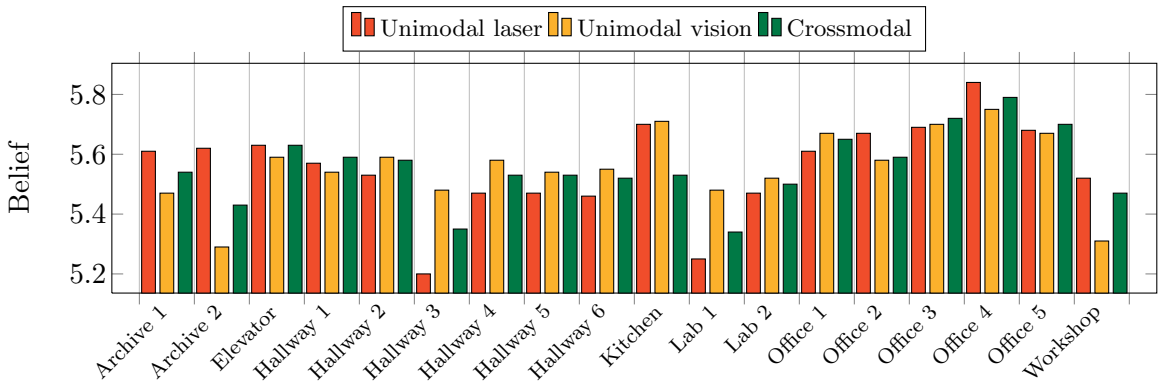


Figure D.7.: The robot's belief if triggered with pattern *Office 4*.

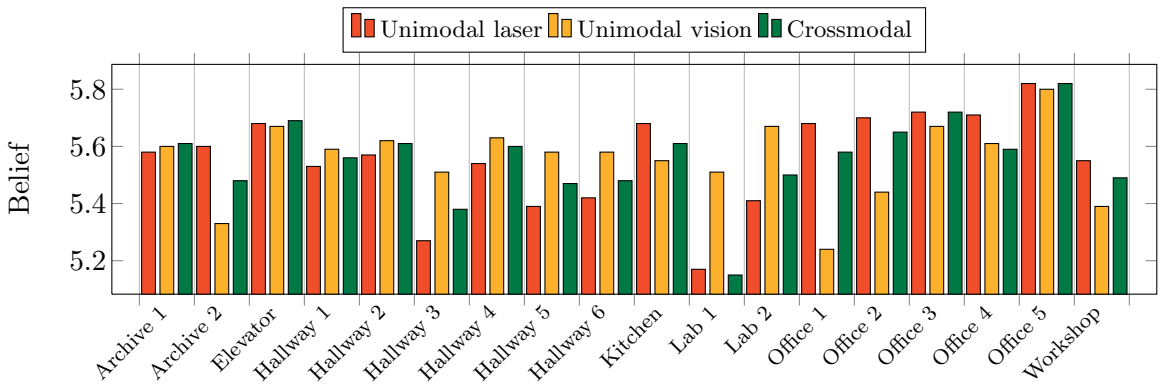


Figure D.8.: The robot's belief if triggered with pattern *Office 5*.

E

Acknowledgements

Completing a doctoral thesis in three years is truly a marathon event, and I would not have been able to complete this journey without the aid and support of countless people over the past years.

I would like to express my gratitude to my supervisor, Prof. Dr. Jianwei Zhang, for all his encouragement during my graduation, for giving me visions and helping me to enter the research community. Moreover, I would like to thank Prof. PhD. Bernd Neumann for all the inspiring discussions and feedback that helped me sharpening my work tremendously. Alessandro Saffiotti and Guosong Liu, thank you both for the nice discussions, fruitful comments and for raising my excitement.

I am deeply indebted to Mateus Mendes who encouraged me through many discussions to realise my studies on SDM. I doubt that I will ever be able to convey my appreciation fully to Felix Lindner. Felix, your unfailing assistance made the thesis what it is. I loved how we shared our expertise and different backgrounds and it helped me tremendously shaping my work. Furthermore, without the excellent contribution of Jan Bruder, most of the presented telemanipulation experiments would not have been possible.

I will miss all CINACS mates, in particular Martin Weser, Pat McCrae, Cengiz Acarturk, Tian Gan, Mario Maiworm, Christian Graf. I will never forget the wonderful culinary journey ceremonies we have had. Great appreciation goes out to all my colleagues from the TAMS group for providing professional support and advice while never losing sight of the funny parts of academic research and life. You never hesitated to serve as subjects for my experiments. Thus, special thanks goes to Lu, Norman, Andreas, Hannes, Denis, Houxiang, Bernd, Tim, Daniel, Markus and Mohammed.

Above all, I would like to thank all members of my family. I cannot adequately acknowledge all the ways in which they have contributed to my work. Similarly, I thank my flatmates Sarah and Christian and all my friends for their support, especially during the last year. Very special thanks are addressed to Kerstin Koch, who helped me sacrificially during various stages of this work and my life, also in unpleasant times.

In conclusion, I recognise that this research would not have been possible without the financial support of the German Research Foundation (DFG) in the framework of the Research Training Group 1247 on Crossmodal Interactions in Natural and Artificial Cognitive Systems (CINACS). Hence, sincere thanks are given to all German taxpayers.

Last but not least, I have to emphasise my poignant grief to leave the CINACS coffee machine behind. Awaken from years of retirement in a dusty wardrobe it provided us the finest black brew from the first days of the project. I am sad to say that the "who makes the strongest coffee ever" competition will have an end now, but on the other hand my stomach might thank for that.

List of Figures

List of Tables

List of Algorithms

Bibliography

Aarno, D. and Kragic, D. (2008). Motion intention recognition in robot assisted applications. *Robotics and Autonomous Systems*, 56(8):692–705.

Agre, P. E. and Chapman, D. (1990). What are plans for? In Maes, P., editor, *Designing autonomous agents: Theory and practice from biology to engineering and back*, pages 17–34. The MIT Press, Cambridge, MA, USA.

Albus, J. (1993). A reference model architecture for intelligent systems design. In Antsaklis, P. and Passino, K., editors, *An introduction to intelligent and autonomous control*. Kluwer Academic Publishers.

Albus, J. S. (1971). A theory of cerebellar function. *Mathematical Biosciences*, 10(1/2):25–61.

Albus, J. S. (1975). A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement, and Control*, 97:220–227.

Albus, J. S. (1981). *Brains, behavior and robotics*. McGraw-Hill Inc., New York, USA.

Amat, J., Frigola, M., and Casals, A. (2001). Virtual exoskeleton for telemanipulation. In *Experimental Robotics VII (ISER)*, pages 21–30, London, UK. Springer.

Amedi, A., Kriegstein, K., Atteveldt, N. M., Beauchamp, M. S., and Naumer, M. J. (2005). Functional imaging of human crossmodal identification and object recognition. *Experimental Brain Research*, 166(3):559–571.

Amit, D. J., Campbell, C., and Wong, K. Y. M. (1989). The interaction space of neural networks with sign-constrained synapses. *Journal of Physics A: Mathematical and General*, 22(21):4687–4693.

Anderson, J. A. (1977). Neural models with cognitive implications. In LaBerge, D. and Samuels, S., editors, *Basic Processes in Reading, Perception and Comprehension*, Hillsdale, NJ. Lawrence Erlbaum Associates.

Anderson, J. R. (1990). *The adaptive character of thought*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Anderson, J. R. (2007). *Kognitive Psychologie*. Spektrum Akademischer Verlag, Heidelberg, 6 edition.

Anwar, A. (2005). Cognitive experiments with SDMSCue (sparse distributed memory for small cues). In *Proceedings of the International Conference on Cognitive Systems (ICCS)*, New Delhi.

Anwar, A., Dasgupta, D., and Franklin, S. (1999). Using genetic algorithms for sparse distributed memory initialization. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, volume 2, pages 1043–1050, Washington, DC.

Apps, R. and Garwicz, M. (2005). Anatomical and physiological foundations of cerebellar information processing. *Nature Reviews Neuroscience*, 6:297–311.

Araujo, A. and Vieira, M. (1998). Associative memory used for trajectory generation and inverse kinematics problem. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 3, pages 2057–2062. IEEE.

Arbib, M. A., editor (2002). *The handbook of brain theory and neural networks*. The MIT Press, Cambridge, MA, USA.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483.

Atkinson, R. C. and Shiffrin, R. M. (1971). The control of short-term memory. *Scientific American*, 224:82–90.

Baddeley, A. and Hitch, G. (1974). Working memory. *The Psychology of Learning and Motivation: Advances in Research and Theory*, 8:47–89.

Baddeley, A. D. (2001). Is working memory still working? *American Psychologist*, 56(11):851–864.

Baier, T., Hüser, M., Westhoff, D., and Zhang, J. (2006). A flexible software architecture for multi-modal service robots. In *Proceedings of the Multiconference on Computational Engineering in Systems Applications (CESA)*, Beijing, China.

Baier-Löwenstein, T. (2008). *Lernen der Handhabung von Alltagsgegenständen im Kontext eines Service-Roboters*. PhD thesis, Department of Informatics, University of Hamburg, Germany.

Bapi, R. S. and Doya, K. (2001). Multiple forward model architecture for sequence processing. *Sequence Learning—Paradigms, Algorithms, and Applications*, pages 308–320.

Barakova, E. I. and Lourens, T. (2005). Spatial navigation based on novelty mediated autobiographical memory. In *Mechanisms, Symbols, and Models Underlying Cognition*, volume 3561 of *Lecture Notes in Computer Science (LNCS)*, pages 356–365, Berlin / Heidelberg. Springer.

Barreto, G. and Araujo, A. (1998). Competitive and temporal hebbian learning for production of robot trajectories. In *Brazilian Symposium on Neural Networks*, pages 96–101, Los Alamitos, CA. IEEE Computer Society.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41(1):164–171.

Bertelson, P. (1999). Ventriloquism: A case of crossmodal perceptual grouping. In Achersleben, G., Bachman, T., and Müsseler, J., editors, *Cognitive contributions to the perception of spatial and temporal events*. Elsevier, Amsterdam.

Billard, A. (2002). Imitation. In Arbib, M. A., editor, *Handbook of Brain Theory and Neural Networks*, pages 566–596, Cambridge, MA, USA. The MIT Press.

Billard, A., Calinon, S., Dillmann, R., and Schaal, S. (2008). Robot programming by demonstration. In Siciliano, B. and Khatib, O., editors, *Handbook of robotics*, pages 1371–1394. Springer.

Billard, A. and Matarić, M. J. (2001). Learning human arm movements by imitation: Evaluation of biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, 941(941):1–16.

Bistry, H., Pöhlsen, S., Westhoff, D., and Zhang, J. (2007a). Development of a smart laser range finder for an autonomous service robot. In *Proceedings of the IEEE International Conference on Integration Technology (ICIT)*, Shenzhen, China.

Bistry, H., Westhoff, D., and Zhang, J. (2007b). A smart interface-unit for the integration of pre-processed laser range measurements into robotic systems and sensor networks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 358–363, San Diego, CA.

Blumenthal, L. M. and Menger, K. (1970). *Studies in geometry*. W. H. Freeman, San Francisco, CA.

Bose, J. (2003). A scalable sparse distributed neural memory. Master thesis, Department of Computer Science, University of Manchester, UK.

Bose, J., Furber, S., and Shapiro, J. (2005). A spiking neural sparse distributed memory implementation for learning and predicting temporal sequences. *Artificial Neural Networks: Biological Inspirations (ICANN) and Lecture Notes in Computer Science (LNCS)*, 3696:115–120.

Bouchard-Côté, A. (2004a). A faster implementation of a sparse distributed memories architecture. Technical report, McGill University, Montreal, Quebec, Canada.

Bouchard-Côté, A. (2004b). Sparse distributed memories in a bounded metric state space: Some theoretical and empirical results. Technical report, McGill University, Montreal, Quebec, Canada.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23.

Brooks, R. A. (1991). Intelligence without representation. In *Artificial Intelligence*, volume 47, pages 139–159.

Brown, A. S. (2008). Why hands matter. *Mechanical Engineering Magazine*.

Bruder, J. (2009). *Praktische Realisierung einer haptischen Telerobotik-Steuerung für eine interaktive Nutzung*. Diploma thesis, Department of Informatics, University of Hamburg, Germany.

Chella, A., Frixione, M., and Gaglio, S. (2000). Understanding dynamic scenes. *Artificial Intelligence*, 123(1-2):89–132.

Chou, P. A. (1989). The capacity of the Kanerva associative memory. *IEEE Transactions on Information Theory*, 35(2):281–298.

Clarke, T., Prager, R., and Fallside, F. (1991). The modified Kanerva model: Theory and results for real-time word recognition. *IEE Proceedings of Radar and Signal Processing*, 138(1):25–31.

Cohen, N. and Squire, L. (1980). Preserved learning and retention of pattern-analyzing skill in amnesia: Dissociation of knowing how and knowing that. *Science*, 210(4466):207–210.

Coradeschi, S., Driankov, D., Karlsson, L., and Saffiotti, A. (2001). Fuzzy anchoring. In *Proceedings of the 10th IEEE Conference on Fuzzy Systems*, pages 111–114.

Coradeschi, S. and Saffiotti, A. (2000). Anchoring symbols to sensor data: Preliminary report. In *Proceedings of the 17th AAAI Conference*, pages 129–135, Menlo Park, CA. AAAI Press.

Coradeschi, S. and Saffiotti, A. (2003). An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2-3):85–96.

Craik, F. and Lockhart, R. (1972). Levels of processing. A framework for memory research. *Journal of Verbal Learning and Verbal Behaviour*, 11:671–684.

Danforth, D. G. (1990). An empirical investigation of sparse distributed memory using discrete speech recognition. Technical report 90.18, Research Institute for Advanced Computer Science, NASA Ames Research Center.

Dawkins, R. (1976). *The selfish gene*. Oxford University Press, UK.

D'Mello, S. and Franklin, S. (2009). Computational modeling/cognitive robotics compliments functional modeling/experimental psychology. *New Ideas in Psychology*, pages 1–11.

D'Mello, S., Franklin, S., Ramamurthy, U., and Baars, B. J. (2006). A cognitive science based machine learning architecture. In *AAAI Spring Symposium Series*, Palo Alto, CA, USA. Stanford University, American Association for Artificial Intelligence.

D'Mello, S., Ramamurthy, U., and Franklin, S. (2005). Encoding and retrieval efficiency of episodic data in a modified sparse distributed memory system. *Proceedings of the 27th Annual Meeting of the Cognitive Science Society*.

Driver, J. and Noesselt, T. (2008). Multisensory interplay reveals crossmodal influences on sensory-specific brain regions, neural responses, and judgments. *Neuron*, 57(1):11–23.

Driver, J. and Spence, C. (2000). Multisensory perception: Beyond modularity and convergence. *Current Biology*, 10(20):R731–R735.

Edelman, S. and Vaina, L. M. (2001). David Marr (a short biography). In Smelser, N. J. and Baltes, P. B., editors, *International Encyclopedia of the Social & Behavioral Sciences*, Oxford. Elsevier.

Fan, K.-C. and Wang, Y.-K. (1997). A genetic sparse distributed memory approach to the application of handwritten character recognition. *Pattern Recognition*, 30(12):2015–2022.

Field, D. J. (1994). What is the goal of sensory coding? *Neural Computation*, 6(4):559–601.

Floreano, D. and Mattiussi, C. (2008). *Bio-inspired artificial intelligence: Theories, methods, and technologies*. The MIT Press, Cambridge, MA, USA.

Fowler, D. (1991). *A neural network as an instrument of prediction*. PhD thesis, Graduate College, University of Nebraska, Lincoln, USA.

Franklin, S. (1995). *Artificial minds*. The MIT Press, Cambridge, MA, USA.

Franklin, S. (2005). Perceptual memory and learning: Recognizing, categorizing, and relating. In *Symposium on Developmental Robotics, American Association for Artifical Intelligence (AAAI)*, Palo Alto, CA, USA.

Furber, S. B., Bainbridge, W. J., Cumpstey, J. M., and Temple, S. (2004). Sparse distributed memory using n-of-m codes. *Neural Networks*, 17(10):1437–1451.

Furber, S. B., Brown, G., Bose, J., Cumpstey, J. M., Marshall, P., and Shapiro, J. L. (2007). Sparse distributed memory using rank-order neural codes. *IEEE Transactions on Neural Networks*, 18(3):648–659.

Fuster, J. M. (1998). Distributed memory for both short and long term. *Neurobiology of Learning and Memory*, 70(1-2):268–274.

Gabora, L. (1996). A day in the life of a meme. *Philosophica*, 57:901–938.

Gabora, L. (2002). Cognitive mechanisms underlying the creative process. In *Proceedings of the 4th Conference on Creativity & Cognition*, pages 126–133, New York, USA. ACM.

Gallese, V. and Lakoff, G. (2005). The brain's concepts: The role of the sensory-motor system in conceptual knowledge. *Cognitive Neuropsychology*, 22(3-4):455–479.

Garalevicius, S. J. (2007). Memory-prediction framework for pattern recognition: Performance and suitability of the Bayesian model of visual cortex. In Wilson, D. and Sutcliffe, G., editors, *Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference*, pages 92–97, Key West, Florida. AAAI Press.

Gärdenfors, P. (2004). *Conceptual spaces: The geometry of thought*. The MIT Press, Cambridge, MA, USA.

Golomb, D., Rubin, N., and Sompolinsky, H. (1990). Willshaw model: Associative memory with sparse coding and low firing rates. *Physical Review A*, 41(4):1843–1854.

Haikonen, P. O. A. (2009). The role of associative processing in cognitive computing. *Cognitive Computation*, 1(1):42–49.

Hämäläinen, T. (1996). Sparse distributed memory implementations on tree shape parallel neurocomputer. In *Proceedings of the 6th IEEE Signal Processing Society Workshop: Neural Networks for Signal Processing*, pages 539–548.

Harnad, S. (1990). The symbol grounding problem. *Physica D*, 42:335–346.

Harnad, S. (1993). Grounding symbols in the analog world with neural nets. *THINK*, 2:12–78.

Hassoun, M. H. (1993). *Associative neural memories: Theory and implementation*. Oxford University Press, UK.

Hawkins, J. (2005). *On Intelligence*. Owl Books, NY, USA.

Hayward, V. and Richard, P. P. (1986). Robot manipulator control under Unix RCCL: A robot control C library. *International Journal on Robotics Research*, 5(4):94–111.

Hebb, D. O. (2002). *The organization of behavior: A neuropsychological theory*. Lawrence Erlbaum Associates, new edition.

Hely, T., Willshaw, D., and Hayes, G. (1997). A new approach to Kanerva's sparse distributed memory. *IEEE Transactions on Neural Networks*, 8(3):791–794.

Hely, T. A. (2006). Sparse distributed memory. *Encyclopedia of Cognitive Science*, pages 101–108. University of Edinburgh, UK.

Hendin, O., Horn, D., and Usher, M. (1991). Chaotic behavior of a neural network with dynamical thresholds. *International Journal of Neural Systems*, 1(4):327–335.

Hendrich, N. (1996). *Mustererkennung mit neuronalen Associativspeichernetzen*. Number 137 in DISKI. infix, Sankt Augustin, Germany.

Hertzberg, J. and Saffiotti, A. (2008). Using semantic knowledge in robotics. *Robotics and Autonomous Systems*, 11:875–877.

Hinton, G. E., McClelland, J. L., and Rumelhart, D. E. (1987). Distributed representations. In Rumelhart, D. E., McClelland, J. L., et al., editors, *Parallel Distributed Processing: Volume 1: Foundations*, pages 77–109. The MIT Press, Cambridge, MA, USA.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence*. The MIT Press, Cambridge, MA, USA. 1st edition: 1975, University of Michigan Press, Ann Arbor.

Hong, Y.-S. and Chen, S.-S. (1991). Character recognition in a sparse distributed memory. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3):674–678.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Science*, volume 79, pages 2554–2558.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. (1998). The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 256–265, Madison, WI, USA. Morgan Kaufmann.

Hüser, M., Baier, T., and Zhang, J. (2005). Imitation learning of grasping skills through a multimodal service robot. In *2005 IEEE ICRA Workshop on The Social Mechanisms of Robot Programming by Demonstration*, Barcelona, Spain.

Hüser, M., Baier, T., and Zhang, J. (2006). Learning of demonstrated grasping skills by stereoscopic tracking of human head configuration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2795–2800.

Hüser, M., Baier-Löwenstein, T., Svagusa, M., and Zhang, J. (2007). Natural demonstration of manipulation skills for multimodal interactive robots. In *Universal Access in Human-Computer Interaction. Ambient Interaction*, pages 888–897. Springer Berlin / Heidelberg.

Ido, J., Shimizu, Y., Matsumoto, Y., and Ogasawara, T. (2009). Indoor navigation for a humanoid robot using a view sequence. *International Journal of Robotics Research*, 28(2):315–325.

Ito, M. and Tani, J. (2004). On-line imitative interaction with a humanoid robot using a dynamic neural network model of a mirror system. *Adaptive Behaviour*, 12:93–115.

Jaeckel, L. A. (1989). An alternative design for a sparse distributed memory. RIACS report 89.28, Research Institute for Advanced Computer Science, NASA Ames Research Center.

Jenkins, O. C. and Mataric, M. J. (2004). Performance-derived behavior vocabularies: Data-driven acquisition of skills from motion. *International Journal of Humanoid Robotics*, 1(2):237–288.

Jockel, S., Baier-Löwenstein, T., and Zhang, J. (2007a). Three-dimensional monocular scene reconstruction for service-robots: An application. In *Proceedings of 2nd International Conference on Computer Vision Theory and Applications (VISAPP)*, volume Special Sessions, pages 41–46, Barcelona, Spain. INSTICC Press.

Jockel, S., Lindner, F., and Zhang, J. (2008a). Sparse distributed memory for experience-based robot manipulation. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1298–1303, Bangkok, Thailand.

Jockel, S., Mendes, M., Zhang, J., Coimbra, A., and Crisóstomo, M. (2009). Robot navigation and manipulation based on a predictive associative memory. In *Proceedings of the 8th IEEE International Conference on Development and Learning (ICDL)*, Shanghai, China.

Jockel, S., Weser, M., Westhoff, D., and Zhang, J. (2008b). Towards an episodic memory for cognitive robots. In *Proceedings of 6th Cognitive Robotics Workshop at 18th European Conference on Artificial Intelligence (ECAI)*, pages 68–74, Patras, Greece. IOS Press.

Jockel, S., Westhoff, D., and Zhang, J. (2007b). EPIROME–A novel framework to investigate high-level episodic robot memory. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1075–1080, Sanya, China.

Joglekar, U. D. (1989). Learning to read aloud: A neural network approach using sparse distributed memory. RIACS report 89.27, Research Institute for Advanced Computer Science, NASA Ames Research Center.

Jonides, J., Lewis, R. L., Nee, D. E., Lustig, C. A., Berman, M. G., and Moore, K. S. (2008). The mind and brain of short-term memory. *Annual Review of Psychology*, 59(1):193–224.

Kahana, M. J. (1 September 2002). Associative symmetry and memory theory. *Memory & Cognition*, 30:823–840(18).

Kandel, E. R., Schwartz, J. H., and Jessell, T. M., editors (1996). *Neurowissenschaften: Eine Einführung*. Spektrum Akademischer Verlag, Heidelberg.

Kanerva, P. (1988). *Sparse distributed memory*. The MIT Press, Cambridge, MA, USA.

Kanerva, P. (1993). Sparse distributed memory and related models. In Hassoun, M., editor, *Associative Neural Memories: Theory and Implementation*, pages 50–76, New York. Oxford University Press.

Kanerva, P. (1994). The spatter code for encoding concepts at many levels. In Marinaro, M. and Morasso, P. G., editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 226–229, London. Springer.

Kanerva, P., Sjödin, G., Kristoferson, J., Karlsson, R., Levin, B., Holst, A., Karlgren, J., and Sahlgren, M. (2001). Computing with large random patterns. In Uesaka, Y., Kanerva, P., and Asoh, H., editors, *Foundations of Real-World Intelligence*, pages 251–312. CSLI Publications.

Karlsson, R. (1995a). Evaluation of a fast activation mechanism for the Kanerva SDM memory. Report R95:10, RWCP Neuro SICS Laboratory, Kista, Sweden.

Karlsson, R. (1995b). A fast activation mechanism for the Kanerva SDM memory. In *Proceedings of the RWC Symposium*, pages 69–70, Tokyo. Also RWC Technical Report 95001.

Keeler, J. D. (1988). Comparison between Kanerva's SDM and Hopfield-type neural networks. *Cognitive Science*, 12(3):299–329.

Kelley, R., Tavakkoli, A., King, C., Nicolescu, M., Nicolescu, M., and Bebis, G. (2008). Understanding human intentions via hidden Markov models in autonomous mobile robots. In *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction*, pages 367–374, Amsterdam, Netherlands.

Kemp, C., Edsinger, A., and Torres-Jara, E. (2007). Challenges for robot manipulation in human environments: Grand challenges of robotics. *IEEE Robotics and Automation Magazine*, 14(1):20–29.

Klimentjew, D., Stroh, A., Jockel, S., and Zhang, J. (2008). Real-time 3d environment perception: An application for small humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 354–359, Bangkok, Thailand.

Kohonen, T. (1972). Correlation matrix memories. *Transactions on Computers*, C(21):353–359.

Kohonen, T. (1987). *Content-addressable memories*. Springer, New York, USA.

Kohonen, T. (1989). *Self-organization and associative memory*. Springer, New York, USA.

Kristoferson, J. (1995). Some comments on the information stored in sparse distributed memory. Research report 95:11, Swedish Institute of Computer Science, Kista, Sweden.

Kristoferson, J. (1998). Some results on activation and scaling of sparse distributed memory. In de Padua Braga, A. and Ludermir, T., editors, *Proceedings of the 5th Brazilian Symposium on Neural Networks*, pages 157–160, Belo Horizonte, Brazil. IEEE Computer Society.

Kuniyoshi, Y., Yorozu, Y., Inaba, M., and Inoue, H. (2003). From visuo-motor self learning to early imitation–a neural architecture for humanoid learning. In *Proceedings of the IEEE International Conference on Robotic and Automation (ICRA)*, pages 3132–3139.

Levesque, H. and Lakemeyer, G. (2007). *Cognitive robotics*. Elsevier.

Lloyd, J. E. and Hayward, V. (1992). Multi-RCCL user's guide. Technical report, Computer Science Department, University of British Columbia, Vancouver, Canada.

Marr, D. (1969). A theory of cerebellar cortex. *Journal of Physiology*, 202:437–470.

Matsumoto, Y., Sakai, K., Inaba, M., and Inoue, H. (2000). View-based approach to robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1702–1708.

McCulloch, W. S. and Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysic*, 5:115–133.

McGurk, H. and MacDonald, J. (1976). Hearing lips and seeing voices. *Nature*, 264(5588):746–748.

Mendes, M. (2009). Robot navigation using a sparse distributed memory. PhD thesis, unpublished.

Mendes, M., Coimbra, A., and Crisóstomo, M. (2007). AI and memory: Studies towards equipping a robot with a sparse distributed memory. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1743–1750, Sanya, China.

Mendes, M., Coimbra, A., and Crisóstomo, M. (2009). Assessing a sparse distributed memory using different encoding methods. In *Proceedings of the World Congress on Engineering (WCE)*, volume 1, London, UK.

Mendes, M., Crisóstomo, M., and Coimbra, A. (2008). Robot navigation using a sparse distributed memory. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 53–58, Pasadena, CA, USA.

Newell, F. N. (2004). Cross-modal object recognition. In G., C., C., S., and B.E., S., editors, *The handbook of multisensory processes*, pages 123–139, Cambridge, MA, USA. The MIT Press.

Nicolescu, M. N. (2003). *A framework for learning from demonstration, generalization and practice in human-robot domains*. PhD thesis, Faculty of the Graduate School, University of South California, Los Angeles, USA.

Olshausen, B. A. and Field, D. J. (2004). Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14:481–487.

Palm, G. (1980). On associative memory. *Biological Cybernetics*, 36(1):19–31.

PDP Research Group, C. (1986a). *Parallel distributed processing: explorations in the microstructure of cognition*, volume 1: Foundations. The MIT Press, Cambridge, MA, USA.

PDP Research Group, C. (1986b). *Parallel distributed processing: explorations in the microstructure of cognition*, volume 2: Psychological and biological models. The MIT Press, Cambridge, MA, USA.

Plate, T. A. (1994). *Distributed representations and nested compositional structure*. PhD thesis, Department of Computer Science, University of Toronto, Canada.

Prager, R. (1993). Networks based on Kanerva's sparse distributed memory: Results showing their strengths and limitations and a new algorithm to design the location matching layer. *IEEE International Conference on Neural Networks*, 2:1040–1045.

Prager, R. W. and Fallside, F. (1989). The modified Kanerva model for automatic speech recognition. *Computer Speech and Language*, 3(1):61–81.

Preusche, C., Reintsema, D., Landzettel, K., and Hirzinger, G. (2006). Robotics component verification on ISS ROKVISS–Preliminary results for telepresence. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4595–4601.

Raaijmakers, J. and Shiffrin, R. (1981). Search of associative memory. *Psychological Review*, 88:93–134.

Rao, R. and Fuentes, O. (1998). Hierarchical learning of navigational behaviors in an autonomous robot using a predictive sparse distributed memory. *Machine Learning*, 31(1-3):87–113.

Ratitch, B., Mahadevan, S., and Precup, D. (2004). Sparse distributed memories in reinforcement learning: Case studies. In de Farias, D. P., Mannor, S., Precup, D., and Cochairs), G. T. P., editors, *Papers from the AAAI Workshop on Learning and Planning in Markov Processes: Advances and Challenges*, pages 85–90.

Ratitch, B. and Precup, D. (2004). Sparse distributed memories for on-line value-based reinforcement learning. *Lecture Notes in Computer Science (LNCS)*, 3201:347–358.

Reinhart, R. F. and Steil, J. J. (2008). Recurrent neural associative learning of forward and inverse kinematics for movement generation of the redundant PA-10 robot. *ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems*, pages 35–40.

Ritter, F. E., Shadbolt, N. R., Elliman, D., Young, R., Gobet, F., and Baxter, G. D. (2003). Techniques for modeling human and organizational behaviour in synthetic environments: A supplementary review. Wright-Patterson Air Force Base, Human Systems Information Analysis Center.

Rogers, D. (1988). Kanerva's sparse distributed memory: An associative memory algorithm well-suited to the connection machines. RIACS report 88.32, Research Institute for Advanced Computer Science, NASA Ames Research Center.

Rogers, D. (1990). Predicting weather using a genetic memory: a combination of Kanerva's sparse distributed memory with Holland's genetic algorithms. *Advances in Neural Information Processing Systems*, 2:455–464.

Rojas, R. (1996). *Neural networks: A systematic introduction*. Springer.

Rößler, P. (2009). *Telepräsente Bewegung und haptische Interaktion in ausgedehnten entfernten Umgebungen*. PhD thesis, Fakultät für Informatik der Universität Fridericiana zu Karlsruhe, Germany.

Schacter, D. L. (1987). Implicit memory: History and current status. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 13(3):501–518.

Scherer, T. (2004). *A mobile service robot for automatisation of sample taking and sample management in a biotechnological pilot laboratory*. PhD thesis, Faculty of Technology, University of Bielefeld, Germany.

Schrempf, O. C., Albrecht, D., and Hanebeck, U. D. (2007). Tractable probabilistic models for intention recognition based on expert knowledge. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1429–1434, San Diego, CA, USA.

Serrien, D. J. (2009). Functional connectivity patterns during motor behaviour: The impact of past on present activity. *Human Brain Mapping*, 30(2):523–531.

Sheridan, T. B. (1992). *Telerobotics, automation, and human supervisory control*. The MIT Press, Cambridge, MA, USA.

Siciliano, B. and Khatib, O., editors (2008). *Handbook of robotics*. Springer.

Sjödin, G. (1995). Convergence and new operations in SDM. Report 95:13, RWCP Neuro SICS Laboratory, Kista, Sweden.

Sjödin, G. (1998). The Sparchunk code: A method to build higher-level structures in a sparsely encoded SDM. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 2, pages 1410–1415.

Squire, L. R. (1986). Mechanisms of memory. *Science*, 232(4758):1612–1619.

Squire, L. R. (2004). Memory systems of the brain: A brief history and current perspective. *Neurobiology of Learning and Memory*, 82(3):171–177.

Squire, L. R., Knowlton, B., and Musen, G. (1993). The structure and organization of memory. *Annual Review of Psychology*, 44:453–495.

Strube, G., editor (1996). *Wörterbuch der Kognitionswissenschaft*. Klett-Cotta, Stuttgart.

Sun, R. and Giles, C. L. (2001). Sequence learning: From recognition and prediction to sequential decision making. *IEEE Intelligent Systems*, 16(4):67–70.

Suppes, P. (1994). Representations and models in psychology. *Annual Review of Psychology*, 45:517–544.

Sutton, J. (2008). Memory. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. The Metaphysics Research Lab, Stanford University, CA, USA.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning. An introduction*. The MIT Press, Cambridge, MA, USA.

Sutton, R. S. and Whitehead, S. D. (1993). Online learning with random representations. In *Proceedings of the 10th International Conference on Machine Learning*, pages 314–321. Morgan Kaufmann.

Tanji, J. (2001). Sequential organization of multiple movements: Involvement of cortical motor areas. *Annual Review of Neuroscience*, 24:631–651.

Tham, C. K. (1995). Reinforcement learning of multiple tasks using a hierarchical CMAC architecture. *Robotics and Autonomous Systems*, 15(4):247–274.

Townsend, W. (2000). The BarrettHand grasper–programmably flexible part handling and assembly. *Industrial Robot: An International Journal*, 27(3):181–188.

Tulving, E. (1972). Episodic and semantic memory. In Tulving, E. and Donaldson, W., editors, *Organization of Memory*, New York. Academic Press.

Turk, A. and Görz, G. (1995). Kanerva's sparse distributed memory: An object-oriented implementation on the connection machine. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 473–479.

Wallace, M. (2004). The development of multisensory processes. *Cognitive Processing*, 5:69–83.

Wasserman, P. D. (1993). *Advanced Methods in Neural Computing*. John Wiley & Sons Inc., New York, USA.

Weser, M. (2009). *Hierarchical memory organization of multimodal robot skills for plan-based robot control*. PhD thesis, Department of Informatics, University of Hamburg, Germany.

Weser, M., Westhoff, D., Hüser, M., and Zhang, J. (2006). Multimodal people tracking and trajectory prediction based on learned generalized motion patterns. In *International Conference on Multisensor Fusion and Integration (MFI)*, pages 541–546, Heidelberg, Germany.

Westhoff, D., Stanek, H., and Zhang, J. (2006). Distributed applications for robotic systems using Roblet-Technology. In *Proceedings of the 37th International Symposium on Robotics and Deutsche Fachtagung Robotik 2006*, Munich, Germany. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik.

Whitehead, S. D. and Ballard, D. H. (1991). Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45–83.

Williams, B. H., Toussaint, M., and Storkey, A. J. (2006). Extracting motion primitives from natural handwriting data. In Kollias, S. D., Stafylopatis, A., Duch, W., and Oja, E., editors, *Proceedings of 16th International Conference on Artificial Neural Networks (ICANN)*, volume 4132 of *Lecture Notes in Computer Science (LNCS)*, pages 634–643, Athens, Greece. Springer.

Willshaw, D. (1981). *Parallel models of associative memory*, chapter Holography, associative memory, and inductive generalization, pages 83–104. Lawrence Erlbaum Associates.

Willshaw, D. J., Buneman, O. P., and Longuet-Higgins, H. C. (1969). Non-holographic associative memory. *Nature*, 222(5197):960–962.

Wu, S. (2008). *Signature descriptions for motion trajectory representation, perception, recognition, and reproduction*. PhD thesis, Department of Manufacturing Engineering and Engineering Managment, City University of Hong Kong, China.

Xu, X., Hu, D., and He, H.-G. (2002). Accelerated reinforcement learning control using modified CMAC neural networks. *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP)*, 5:2575–2578.

Yu, W., Dubey, R., and Pernalete, N. (2004). Telemanipulation enhancement through user's motion intention recognition and fixture assistance. In *Proceedings of the 17th Florida Conference on the Recent Advances in Robotics (FCRAR)*.

Zhang, J. and Knoll, A. (2003). A two-arm situated artificial communicator for human-robot cooperative assembly. *IEEE Transactions on Industrial Electronics*, 50(4):651–658.

Zhao, L. (2004). Is sparse and distributed the coding goal of simple cells? *Biological Cybernetics*, 91(6):408–416.

Index

M

N

O

P

R

S

T

U

V

W