Building Blocks for Cognitive Robots:

Embodied Simulation and Schemata in a Cognitive Architecture

Nikolas J. Hemion

Gedruckt auf alterungsbeständigem Papier nach ISO 9706

Building Blocks for Cognitive Robots:

Embodied Simulation and Schemata in a Cognitive Architecture

Nikolas J. Hemion

Der Technischen Fakultät der Universität Bielefeld
vorgelegt zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
Juli 2013

Abstract

Building robots with the ability to perform general intelligent action is a primary goal of artificial intelligence research. The traditional approach is to study and model fragments of cognition separately, with the hope that it will somehow be possible to integrate the specialist solutions into a functioning whole. However, while individual specialist systems demonstrate proficiency in their respective niche, current integrated systems remain clumsy in their performance. Recent findings in neurobiology and psychology demonstrate that many regions of the brain are involved not only in one but in a variety of cognitive tasks, suggesting that the cognitive architecture of the brain uses generic computations in a distributed network, instead of specialist computations in local modules. Designing the cognitive architecture for a robot based on these findings could lead to more capable integrated systems.

In this thesis, theoretical background on the concept of *embodied cognition* is provided, and fundamental mechanisms of cognition are discussed that are hypothesized across theories. Based on this background, a view of how to connect elements of the different theories is proposed, providing enough detail to allow computational modeling. The view proposes a network of generic building blocks to be the central component of a cognitive architecture. Each building block learns an internal model for its inputs. Given partial inputs or cues, the building blocks can collaboratively restore missing components, providing the basis for *embodied simulation*, which in theories of embodied cognition is hypothesized to be a central mechanism of cognition and the basis for many cognitive functions. In simulation experiments, it is demonstrated how the building blocks can be autonomously learned by a robot from its sensorimotor experience, and that the mechanism of embodied simulation allows the robot to solve multiple tasks simultaneously.

In summary, this thesis investigates how to develop cognitive robots under the paradigm of embodied cognition. It provides a description of a novel cognitive architecture and thoroughly discusses its relation to a broad body of interdisciplinary literature on embodied cognition. This thesis hence promotes the view that the cognitive system houses a network of active elements, which organize the agent's experiences and collaboratively carry out many cognitive functions. On the long run, it will be inevitable to study complete cognitive systems such as the cognitive architecture described in this thesis, instead of only studying small learning systems separately, to answer the question of how to build truly autonomous cognitive robots.

Contents

1

Introduction

Intelligent robots as tireless helpers, aiding us in whatever situation is demanding, dangerous, stressful, or simply unpleasant for humans. This vision has existed for almost a century and has become ubiquitous in modern science fiction (see Figure 1.1). Actual attempts to construct intelligent machines began with the dawn of artificial intelligence research in the 1950s, and lead around the year 1970 to the first mobile robot that was capable of moving around in its environment (the robot "Shakey" at Stanford University, see Nilsson, 1969). Yet, we are still far from understanding how to build truly autonomous robots that can be employed in our everyday situations. It has rather become clear that this is an extremely ambitious aim, and that we do not even know for certain in what way we should approach it.

While early work in artificial intelligence was already devoted to the goal of building intelligent robots, the focus was placed almost entirely on the "thinking": Endowing computational systems with the capabilities to plan, to reason, to deliberate, etc. The "acting", i.e. using a physical body to manipulate and move around in the real world, was more or less put aside and considered to be an independent problem that could be solved once the thinking was ready. However, this approach has turned out to be problematic: Today's "intelligent" machines perform well, in some cases with supra-human performance, but only as long as the human designer of the machine is able to provide a suitable abstract description of the machine's task. A combination of ever increasing computing power and efficient algorithms that process huge amounts of data has allowed these machines to become proficient for specific purposes. Famous examples are IBM's two computer systems "Deep Blue" and "Watson": Deep Blue won a chess match against world champion Garry Kasparov, and Watson competed in the game show *Jeopardy!* against two human contestants, both former winners of the game, and placed first. However, such machines remain *disembodied* devices that can only passively process information and are limited to the very task they were designed for. Robots (i.e. *acting* machines) on the other hand remain scarce up to today, and the few that are available on the consumer market have rather restricted capabilities. Their most limiting factor still seems to be their inability to freely move around and act in unstructured environments: Vacuum-cleaning robots are restricted to planar surfaces;

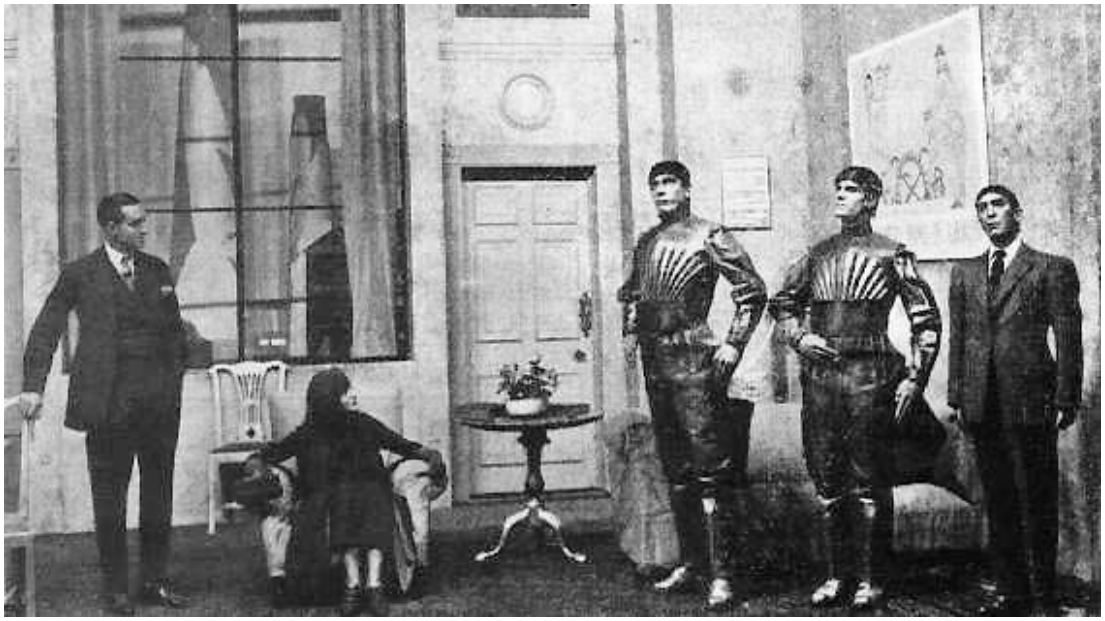


Figure 1.1: A scene from Karel Čapek's 1920 science fiction play "R.U.R.", showing three robots.

flying robots require enough free space for floating, so that they can largely avoid critical encounters with the solid parts of the world; and robots that do have legs and arms thus far do not make very dexterous use of them, and instead rather resemble toys (for kids and scientists alike) instead of actual helpers. "Moravec's paradox" probably captures this imbalance in our understanding of different aspects of cognition best:

> "In hindsight, this dichotomy is not surprising, since the first multi-celled animals appeared about a billion years ago, survival in the fierce competition over such limited resources as space, food, or mates has often been awarded to the animal that could most quickly produce a correct action from inconclusive perceptions. Encoded in the large, highly evolved sensory and motor portions of the human brain is a billion years of experience about the nature of the world and how to survive in it. The deliberate process we call reasoning is, I believe, the thinnest veneer of human thought, effective only because it is supported by this much older and much more powerful, though usually unconscious, sensorimotor knowledge. We are all prodigious olympians in perceptual and motor areas, so good that we make the difficult look easy. Abstract thought, though, is a new trick, perhaps less than 100 thousand years old. We have not yet mastered it. It is not all that intrinsically difficult; it just seems so when we do it" (Moravec, 1988, pp. 15–16).

Insights such as this have recently led researchers to propose a different way of thinking: The clearcut separation between the thinking and the acting, which was

assumed in earlier work in artificial intelligence, is argued to be entirely misguiding, according to this new school. Instead, it is proposed that these apparently distinct capabilities of a cognitive agent are just different manifestations of the same underlying mechanisms. True cognition is said to be *embodied*, meaning that it is entirely linked to the sensorimotor capabilities of the cognitive agent. Deliberating about the real world is in many ways supported by the large amount of sensorimotor knowledge that an agent has acquired through physical interaction with its environment. The problem with current "intelligent" robots is that they do not posses a comparable background of sensorimotor knowledge but operate on abstract task representations, which allows them to seem incredibly adept at solving complex abstract tasks, but at the same time they are hopelessly outperformed even by very young infants when it comes to natural real-world tasks.

In response, a new way of conceiving robots has been proposed, as machines that need to *learn* and *develop*, instead of being programmed by the human designer (see Asada et al., 2001; Weng et al., 2001). Researchers have begun to identify possible mechanisms that allow cognitive capabilities of an agent to *emerge* from its dynamic interaction with the environment, and to demonstrate the capability of computational models to learn through sensorimotor experience. However, to advance from building computational models of individual phenomena to constructing complete cognitive robots, we are still lacking an understanding of how to integrate the various models, or in short, an answer to the question: What is the *cognitive architecture* that supports autonomous mental development in a robot?

1.1 Research Goals and Contributions of this Thesis

The overarching goal of this thesis is to advance the current understanding of how cognitive robots could be developed under the paradigm of *embodied cognition*. More specifically, it seeks to **identify basic elements from which a cognitive architecture for a robot might be built**. To this end, the interdisciplinary setting of the study of cognitive robotics is taken into account: As part of the cognitive sciences, cognitive robotics shares many close links with other disciplines, including psychology, neuroscience, linguistics and philosophy (cf. Miller, 2003). For example, most computational models of cognitive mechanisms are based on empirical findings from the other disciplines, and in return allow to make new predictions and thus encourage new empirical studies. In spite of that, also on a more theoretical level there are many connecting factors, which cannot be ignored. While computational models of individual phenomena can provide convincing accounts for how these phenomena might emerge from sensorimotor interaction, it also needs to be taken into consideration how they relate to comprehensive theories of cognition. In the traditional approach to artificial intelligence, this was not so problematic, since it was assumed that the brain's operation was like that of a computer program. Consequently, individual faculties of cognition, such as visual processing, language understanding, or planning, would be integrated into a whole system by using the metaphor of interfaces via which informa-

tion is passed between components. Thus, successfully developed specialist solutions to sub-problems of cognition would be directly transferrable to the final cognitive system. However, this is not so easily possible if cognition is assumed to be an emergent system property: Existing computational models of emergent cognitive phenomena differ substantially on a methodological level, and a system that demonstrates the emergence of one phenomenon cannot simply be assumed to also support the emergence of another.

Therefore, to be successful beyond individual phenomena, common grounds need to be identified. The approach that was pursued in the scope of this thesis was to collect prominent theories of embodied cognition across disciplines and to identify points of agreement among them, with the goal to **establish a theoretically sound basis that is backed by empirical data, to ground the computational modeling of generic principles for a cognitive system**. Thus, while computational models of empirically observed phenomena use specifically tuned mechanisms to demonstrate a desired system behavior, and are evaluated by comparing the results obtained from robot experiments with observations related to the respective phenomenon, here it is tried to develop *generic* methods for the development of cognitive systems.

To demonstrate the validity of the proposed methods, example applications for learning and control in a robot system will be used for evaluation purposes. Nevertheless, the focus of the modeling lies not on finding specific solutions for the example applications, but on the goal to **develop a generic building block and its mechanisms for the design of a cognitive architecture, supporting the autonomous acquisition of knowledge from sensorimotor experience**. Empirical evidence suggests that the mammalian cortex is composed of repeating neural structures (Mountcastle, 1997), thus pointing to the conclusion that also the cognitive architecture of the brain is based on generic mechanisms that are employed to process information domain-independently (e.g. Melchner et al., 2000; O'Leary, 1989). In addition, basing the design of a cognitive architecture on the use of generic building blocks is beneficial also from an engineering point of view: It limits the design effort for the creation of a cognitive system to the specification of connections in between building blocks and the system's sensory inputs and motor outputs. Since the internal mechanics of the building block (i.e. the mechanisms for learning and operation) are fixed by design, it would not be necessary to implement individual specialist components and in principle provides a possibility for open-ended learning.

To conclude, this thesis pursues the following research goals:

G0 (overarching goal) Identify basic elements from which a cognitive architecture for a robot might be built.

G1 Establish a theoretically sound basis that is backed by empirical data, to ground the computational modeling of generic principles for a cognitive system.

G2 Develop a generic building block and its mechanisms for the design of a cognitive architecture, supporting the autonomous acquisition of knowledge from sensorimotor experience.

1.2 Outline

Chapter 2 introduces the general topic of cognitive architecture, and gives an overview of current modeling approaches and existing computational cognitive architectures. For the development of any cognitive architecture, fundamental modeling choices need to be made with far reaching consequences. To appreciate these, it is helpful to understand the theoretical motivation of the modeling choices in the interdisciplinary context of the cognitive sciences. Therefore, the overview of cognitive architectures is embedded in a description of the currently predominant theoretical paradigms for understanding cognition, which are *cognitivism*, *behavior-based robotics*, *connectionism* and *dynamicism*, accompanied by a discussion of their benefits and drawbacks.

In Chapter 3, the concept of *embodied cognition* is introduced with the goal to establish that it is a suitable theoretical basis to motivate the modeling of a cognitive architecture. An overview of prominent theories and arguments in favor of an embodied view on cognition is given. Subsequently, a view is proposed that coherently connects several theoretical approaches and is concrete enough to support computational modeling. Based on these considerations, a new cognitive architecture is proposed, which uses a network of generic building blocks as its main component.

Following the overall description of the cognitive architecture, Chapters 4 and 5 are concerned with the description of concrete mechanisms of the generic building block. These are related to the questions of how to learn from sensorimotor experience, how to implement distributed decision making and allow the system to solve multiple tasks simultaneously, and how to decide what training samples are relevant for individual building blocks without relying on labeled training data.

Finally, in Chapter 6 the work is summarized, and the cognitive architecture and its mechanisms are discussed in relation to the state of the art in cognitive architecture, as well as in relation to the interdisciplinary literature of embodied cognition. A conclusion is drawn and suggestions for future work and improvements are given.

2

Cognitive Architecture: Overview of Theoretical Paradigms and Computational Models

In many scenarios where robots are applied it is possible, perfectly reasonable, and sometimes also desirable, to engineer complex robotic systems based on the knowledge that the human designer has of the robot's tasks, the premise being that it can be sufficiently well described in advance. Example scenarios where this approach has been applied with some success include residential service robots (e.g. Wachsmuth et al., 2010) or robots acting as a museum guide (Thrun et al., 1999). In both of these examples it is possible to pre-specify a range of possible situations which the robot might face, including interactions with humans, which allows to prepare the reactions of the robot in these situations. Also the behavior of a carefully engineered system is predictable for the designer, which is desirable in cases where a system fault entails high cost, as for example in space robots such as the Mars Exploration Rovers (Biesiadecki et al., 2007).

However, when the environment is difficult to model and can change unpredictably, these systems tend to break down (Asada et al., 2001). The alternative is to develop a system that is not highly specialized to mainly perform well in pre-specified situations, but one that is capable of *general intelligent behavior*, which is the goal of work in the research field of cognitive architecture (Laird et al., 1987). To this end, a cognitive architecture implements a scientific hypothesis about what aspects of cognition are independent of task (Howes and Young, 1997), that is to say, it is explored whether a single theory of what is common among many cognitive behaviors can support all of cognition (Lehman et al., 1996). This is different from developing systems that perform particularly well in a specific task, as it is rather the goal to develop an integrated system that can cope with *many* situations.

The definition of the generic underlying mechanisms of a cognitive architecture is

tied to subscribing to one of the paradigms of cognition that have been proposed in the cognitive science literature. Each of these paradigms takes a significantly different stance on the nature of cognition, what processes and structures are underlying cognition, and how a cognitive system should be analyzed. The traditional and still popular paradigm in cognitive science is the so-called *cognitivism*. It assumes that "higher" cognition is a form of computation, operating on abstract mental representations of the world, and has served as the motivation for classical artificial intelligence. While cognitivism prevails until today, it has been attacked on several grounds by the proponents of a broader class of paradigms, which collectively are referred to as *emergentism*. While the individual emergentist paradigms differ in several important ways, their common notion is that it is a fallacy to assume higher cognition to be categorically different from "low-level" processes, such as neural activation dynamics or reflexive motor behavior.

With the objective to survey computational models of cognitive architecture, in this chapter the different paradigms of cognition will be reviewed, each followed by a description of computational models that subscribe to that paradigm by the choice of their modeled mechanisms. It is not easily possible to disentangle the paradigms of cognition that have been proposed in the literature. Instead, in many ways, different approaches are based on similar intuitions, make comparable assumptions or have developed an overlap in their characteristics. Nevertheless, this chapter will be organized by introducing the following paradigms, which are often described in the literature as full-fledged theories, or at least autonomous research programs: *Cognitivism*, *behavior-based robotics*, *connectionism*, and *dynamicism*.

Section 2.2 will begin with describing *cognitivism*, which assumes that cognition is a rule-based manipulation of structured mental representations. Cognitivism is the predominant view in cognitive science since the middle of the last century and strongly related to traditional artificial intelligence. In Section 2.2.2, so-called "hybrid" architectures will be described, which combine the cognitivist approach with aspects of other approaches, trying to benefit from the strengths of both while mitigating their weaknesses.

In Section 2.3, the *behavior-based robotics* movement will be described, which from the perspective of cognitive architecture in robotics has been the first strong challenger of the traditional approach to designing intelligent robots. Behavior-based robotics is an attempt to model intelligent behavior without resorting to the use of an internal representation of the external world, and is strongly related to a concept from cognitive neuroscience, the "motor schema", which in turn is based on the more general concept of "schema" from psychology, both of which will be described later on in Sections 3.2.1 and 3.1.3, respectively.

In cognitive science, *connectionism* (Section 2.4) has been held as the rivaling view against cognitivism since around the early 1980s. Its main characteristic is a strong commitment to artificial neural networks as a method and mechanisms of spreading activation for the description of cognitive processes. Connectionism argues that representations should not be structured and explicit in the system, as in cognitivism, but implicitly encoded in graded activation values of neurons.

More recently, *dynamicism* (Section 2.5) has been introduced in cognitive science and has gained attention as a third view appart from cognitivism and connectionism. Dynamicism argues that the dynamic interaction of the agent with its environment should be the prime subject of investigation, and that the question of representation should not be investigated in isolation, but should rather be discussed using the terminology of a dynamical system, such as attractor points and instabilities. As such, dynamicism has to some extent resemblance with the behavior-based robotics approach, which also tries to eliminate internal representations from the system as much as possible. At the same time, a broad class of artificial neural networks also are a form of dynamical system, and could therefore qualify as methods for investigation under the flag of dynamicism. Thus, a subset of connectionist approaches also share strong resemblance with dynamicism.

By no means, this short list of interdependencies can be claimed complete. Completely missing are the relations to ideas that have already been formulated in the field of philosophy, some several centuries ago, and also to work in psychology before the dawn of cognitive science in the middle of the last century. However, further examination, although interesting, would not serve the purpose of this discussion but make it lengthy.

As some of the paradigms make strong commitments to how the hypothesized mechanisms are implemented in the brain, the chapter will begin with a very brief introduction to the structure of the cortex, to provide the necessary background.

2.1 Structure of the Cortex: A Brief Introduction

The cerebral cortex is the main site of information processing in the mammalian brain. It plays a key role in many functions, including perception and action, but also attention, memory, learning, thought and language. On a coarse level it is already possible to tell a certain structure with the naked eye, as fissures divide the whole cortex into a left and a right hemispheres, and each hemisphere further into four lobes: The frontal lobe in the front, the parietal lobe in the upper part and the temporal lobe in the lower part of the brain, and the occipital lobe in the back (see Figure 2.1). Connections

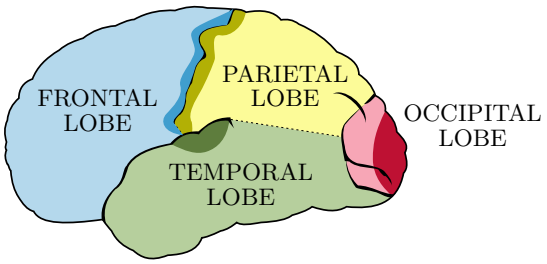


Figure 2.1: Schematic view of the brain, left side. Locations of primary sensory and motor cortices are indicated by darker shading: Primary motor cortex is shown in blue, primary somatosensory cortex in yellow, primary auditory cortex in green, and primary visual cortex in red. Modified drawing from (Gray, 1918).

from and to the senses arrive at so-called primary sensory areas. Visual information is processed in the primary visual cortex, which is located in the occipital lobe. Auditory and somatosensory information arrive at the primary auditory cortex, located in the temporal lobe, and the primary somatosensory cortex, located in the parietal lobe, respectively. Similarly, motor information is directed to and from the primary motor cortex, located in the frontal lobe. Based on anatomical considerations and observations in behavioral studies, the cortex can be further divided into regions, which can be distinguished functionally, although there are no clear anatomical demarcations (for an extensive overview of the literature, see Mesulam, 1998). There are different proposed anatomical subdivisions of the cortex, or "cortical maps", of which Brodmann's proposal is the most commonly used (Brodmann, 1909), which divides the cortex into 52 regions.

Neurons in the cortex are not randomly connected to each other, but connection patterns can be found on several levels. Work on brain mapping has demonstrated the existence of at least two kinds of intra-cortical connections (Sporns and Zwi, 2004): On the one hand, neurons lying close together form densely interconnected clusters with short connections between individual neurons. Neurons lying inside such a cluster often share common dynamical properties in their firing patterns. On the other hand, there is a sparse inter-clustered connectivity via longer connections, to allow information exchange between neurons that are further apart from each other in the cortex. Starting from the primary sensory and motor cortices, where uni-modal information arrives at separate locations across the cortex, a hierarchical organization of the cortical regions can be found (see e.g. Scannell et al., 1995). Reciprocal connections send information from the primary sensory and motor regions to so-called association areas that fuse information from several cortical regions, first in modality-specific association areas that integrate information from the same modality, then from there on to higher association areas, where information becomes increasingly multi-modal the further one goes up the hierarchy (see Figure 2.2).

A prominent example of the hierarchical organization of parts of the cortex is described by the "two-streams hypothesis". According to this widely accepted view, visual information is processed in two separate "streams" (Goodale and Milner, 1992; Mishkin and Ungerleider, 1982; Schneider, 1969), see Figure 2.2. The *dorsal pathway* on the one hand, also known as the "where stream", is hypothesized to extract information from the visual input on the position of objects in space, ultimately to guide the direction of action. The *ventral pathway*, or "what stream", on the other hand is said to extract the visual appearance of objects, independent of the stimulus location on the retina[1]. Both of these pathways begin at the primary visual cortex in the occipital lobe, where information from the retina arrives. From there, the dorsal pathway follows connec-

[1]The name "dorsal" originates from Latin *dorsalis*, meaning "relating to the back", and "ventral" originates from Latin *ventralis*, meaning "abdomen". In anatomical terms these denominate relative locations in terms of orientation inside the body of an animal. In the brain, dorsal refers to regions lying in the upper part, whereas ventral refers to regions lying in the lower part. In most animals, this corresponds to the dorsal-ventral direction of the whole body, but in humans the orientation of the brain has changed evolutionary due to the upright body posture.

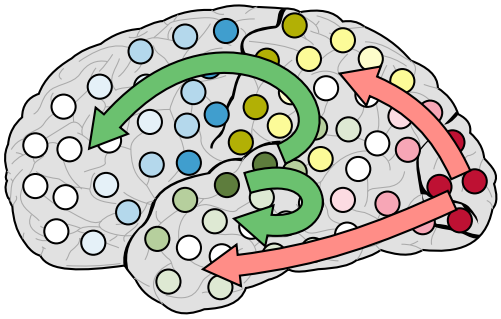


Figure 2.2: Schematic drawing of the hierarchical organization of the cortex. Circles represent clusters of neurons (not to be taken as anatomically precise demarcations), color shading indicates hierarchical level: Darker shaded circles represent clusters of neurons that are closer to the primary cortical regions, brighter shaded circles represent higher association areas. Lower level regions process uni-modal information, whereas higher cortical regions combine more and more multi-modal information.
Arrows indicate hypothesized forward streams of information processing: Red arrows show dorsal and ventral streams of visual information processing (upper and lower arrow, respectively), green arrows show dorsal and ventral streams of auditory information processing. Note that the direction of arrows indicates a hypothesized hierarchical organization and should not be understood as uni-directional information transfer: Cortico-cortical connections are mostly reciprocal, also along the shown pathways (see text for explanation).

tions between regions of the cortex until reaching the parietal lobe, whereas the ventral pathway describes connected regions that reach from the occipital lobe downwards to the temporal lobe.

Analogously to the two-streams hypothesis of visual information processing, a similar model of auditory information processing has been proposed. Here, a dorsal stream is thought to run from the primary auditory cortex in the temporal lobe via the parietal lobe forward to the frontal lobe, and a ventral stream is thought to go from the primary auditory cortex forward into an association cortex in the temporal lobe (Hickok and Poeppel, 2007). In this model, the dorsal stream is said to be involved in a sound-to-motor mapping, whereas the ventral stream associates sounds with meaning (see Figure 2.2).

Note that connections between cortical regions along the "streams" are not uni-directional, but mostly reciprocal. That is, an interpretation according to which information is passed only in a feed-forward manner in one direction seems to be uninformed (Goldman-Rakic, 1988).

2.2 Cognitivism

As mentioned earlier, cognitivism is the traditional and still popular view in cognitive science that asserts that higher cognition, including faculties such as reasoning, memory, planning and language, is essentially a form of computation. It is assumed that the architecture of the mind is composed out of many functional modules (Fodor,

1983), each of which is domain-specific and operating on a certain kind of input. The assumption of modularity is based on observations that localized cortical damage can lead to the loss or impairment of specific cognitive abilities. For example, Broca's area, which is located in the lower part of the frontal lobe (roughly in Brodmann's areas 44 and 45), is traditionally associated with language processing, since damage to this region has been reported to cause an almost complete loss of the ability to speak (for a review, see Dronkers et al., 2007). Combined with the observation mentioned above that sensory input is represented in localized places across the cortex, which also gives the sensorimotor parts of the cortex a module-like character, it gives rise to the hypothesis that all of the cortex is made up of functionally distinct modules, which are inter-wired in a complex cognitive architecture.

Thus, the cognitive architecture is thought to be very much like that of a modular computer program. "High-level" cognitive functions, such as reasoning, planning, memory and language, are thought to be the result of abstract computations, based on the representation of physical entities and events as "symbols" in the cognitive system, a cognitive code on which operations are carried out and from which the behavior of the system is determined (Pylyshyn, 1986). This fundamental idea of cognitivism is captured in the hypothesis that any cognitive system capable of general intelligent action is a so-called "physical symbol system" (PSS, Newell and Simon, 1976). A PSS, as defined by Newell and Simon, consists of a set of entities, called the "symbols", and certain operations that can be performed to manipulate these entities, which are the instantiation, destruction, copying and modification of symbols. A cognitive system that implements a PSS thus operates by handling a set of "symbol tokens" (instantiations of the symbols), and manipulates the symbols according to a set of rules to come up with a plan of how to act. Some of the symbols can be directly related to our intuitive understanding of the task domain of the system, and thus be "transparent" for our interpretation of the system's performance, as for example a chess playing program might use symbols for knight, E2, castling and checkmate (cf. Clark, 2001).

When a PSS is used in a cognitive architecture for a robot or some other embodiment (such as in ambient intelligence), it needs to deal with the problem of relating the symbol tokens that exist inside the system to objects in the external world, which is known as the "symbol grounding problem" (Harnad, 1990; Searle, 1980). If we consider a chess-playing robot instead of a chess-playing computer program, it is obvious that the robot needs to know what parts of the environment (for example in terms of image regions in its camera input) correspond to the symbol tokens it uses for its computation, for example where the actual chess piece is located that it plans to move. Thus, in any cognitive architecture that uses a PSS, it needs to be accompanied by a sensorimotor system that reliably "grounds" the symbol tokens in sensorimotor states, see Figure 2.3.

The apparent functional separation of sensory information processing in the brain, as in the two-streams hypothesis of visual information processing, where spatial information and information on the visual appearance are processed separately from each other, could be seen as support for a modular view of the organization of the brain. The interpretation would be that the processing is implemented in different "pipelines",

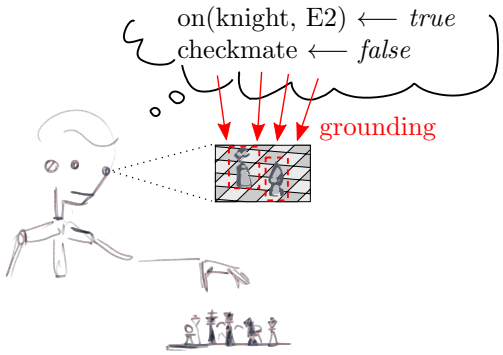


Figure 2.3: A chess-playing robot based on a physical symbol system: Its high-level functions, such as planning the next move, are based on the manipulation of abstract symbolic representations inside the robot's program, based on sets of rules that should reflect the objective structure of the robot's world. The robot needs to keep its internal representation of the world in sync with external reality, by grounding the symbols it uses in sensorimotor states, for example by detecting the chess pieces in the camera image.

where information is passed from one distinct processing stage to the next, and in the end, the results of the different pipelines are fused into a representation of the visual perception. This view would be in line with the PSS hypothesis: The sensory systems of the brain perform complex computations to extract and ground a compact, symbolic representation of the environment, which is then used for abstract computations that implement higher cognitive functions.

An important methodological assumption underlying the cognitivist paradigm is that a clear distinction between different levels of analysis can be made (Griffiths et al., 2010; Marr and Poggio, 1976): The problems faced by the system are studied on a "computational level" in an abstract way (for example, find an object in a visual scene, plan a journey from your home to Notre Dame in Paris, understand a spoken command), algorithms to solve the problems are studied on an "algorithmic level", while the way that the algorithms are actually realized is studied on the "hardware level". Importantly, it is argued that it is possible to decouple the different levels of analysis, which should allow the study of cognition entirely independently of considering the way it is implemented, for example in the neural hardware of the brain. This separation is reflected in the way that cognition is thought to be realized in a PSS: The set of symbols and operations that determine the system are entirely abstract definitions, and as long as the symbol tokens are reliably grounded, there would be no difference in the functioning of the system on the computational level of analysis when it is implemented in different embodiments (for example in two robots with different sensors and actuators).

2.2.1 Computational Models

In the cognitivist paradigm, the definition of a cognitive architecture typically entails a description of the long-term and short-term memories, the representations that are

contained in these memories, how these representations are combined into larger-scale representational structures, and the mechanisms that operate on the representations (Langley et al., 2009). In the following, two candidates of the cognitivist paradigm will be described, *Soar* and *ACT–R*. While there exist many more in the literature, these two architectures are arguably the most prominent, and a description of their functioning will give a clear enough understanding of the general mechanisms underlying any cognitive architecture in the cognitivist paradigm, to allow a discussion of the strengths and importantly the unsolved problems of the paradigm, which follows in Section 2.2.2.

Both Soar and ACT–R are at their core a so-called *production system*, which is a set of processes called productions (Simon, 1975) that operate on the symbolic representations in the system. Each production is an *IF-THEN* rule, the *IF* portion specifying a condition in terms of symbols and their values, and the *THEN* portion defining what the system should do when the associated *IF* portion "fires", i.e. evaluates true. This can either be the execution of an action or a manipulation of the symbols inside the system. For example, a typical production rule in a service robot might look something like "*IF goal* is *bring-user-coffee* and *have-container* is *false THEN* set *goal* to *find-favorite-coffee-mug*". The productions are said to be the content of long-term memory, while the symbol tokens that they operate on are held in working memory. Production systems can differ in terms of how the productions are selected for execution, for example whether multiple productions can fire simultaneously or if only a single production at a time is allowed to fire.

Soar

The idea behind the Soar Cognitive Architecture (Laird and Rosenbloom, 1996; Laird et al., 1987; Lehman et al., 1996; Nuxoll and Laird, 2004) is to separate domain-specific content from domain-general mechanisms, to allow the mechanisms to be applicable across domains and to limit the need for programing to the definition of domain-specific production rules. Soar uses two memories, a long-term memory that stores the production rules, and a working memory in which the current world state and the system's goal in terms of a goal state are represented in symbolic form. Thus, Soar uses its productions for goal-directed symbolic reasoning: The architecture implements a set of mechanisms that search through and apply the production rules in long-term memory to find a sequence of actions that will produce the goal state. These mechanisms are free from domain-specific assumptions, while domain knowledge is provided by the designer to the system in the form of the production rules, which are grouped together to define different problem spaces.

While the production rule is the basic element of long-term memory, Soar's working memory uses a larger representation, which is called the "goal context", consisting of a description of the goal state, which problem space (set of rules) to use, the current state, and an "operator" that defines the system's next action. Processing in Soar is performed in two repeating phases: An "elaboration" phase and a "decision" phase. In the elaboration phase, the system repeatedly compares the current state of the working memory with the *IF*-portions of the productions and fires all matching rules

simultaneously. As some of the productions in Soar extend and manipulate the symbols in working memory, this results in an augmentation of the current state representation with associative knowledge from long-term memory, while other productions propose operators for executions. The elaboration phase continues iteratively until there are no more rules that fire, and is followed by the decision phase. The purpose of the decision phase is to select a single operation for execution, as it is not guaranteed that only a single operator is proposed by the productions. For this purpose, Soar makes use of further production rules, which can tell the decision mechanism to favor one or another operation, based on the current state of working memory. If an impasse arises, i.e. multiple operations remain that are equally favored, the decision mechanism generates a new goal context to extend the working memory with knowledge from another problem domain. Processing continues in this new goal context in the same manner, through an elaboration phase and a decision phase, which can either result in resolving the impasse in the first goal context or can again trigger a new goal context. This process continues until a single operation is favored over all others, which is then executed by the system, for example by telling the motor system to execute an action.

Soar traditionally supports only one form of learning, which is called "chunking": Whenever an impasse arises and is successfully resolved by the addition of knowledge from other problem domains, Soar creates a new production rule that associates the state in which the impasse arose with the outcome of the decision mechanism. Chunking should prevent the system from having to resolve the same impasse more than once. More recently however (Laird, 2008, 2012), Soar was extended with sub-symbolic learning mechanisms to support a slightly more graduated decision making mechanism, by associating to each of the production rules a numeric value. These values are trained through reinforcement learning, using an intrinsic "arousal" signal for training. The values should provide the decision mechanism with further information on which rules have previously led to a favorable outcome.

ACT–R

"Adaptive control of thought–rational" (ACT–R, Anderson et al., 2004; Anderson and Lebiere, 1998), as compared to Soar, is more tightly linked to brain anatomy, as it tries to provide a theory that is able to explain data from studies in cognitive neuroscience, for example from brain imaging experiments. As such, ACT–R is based on the assumption that the brain is composed out of functional modules that are largely encapsulated (Fodor, 1983), an assumption which is widely accepted by proponents of the cognitivist paradigm (see above). ACT–R focuses on how information is passed between these modules. Like Soar, ACT–R has been developed over a longer time period and is still being extended. As of the version ACT–R 5.0, it consists of five implemented modules, each responsible for processing a different kind of information: A visual module identifies objects, a manual module provides an interface for controlling the hands, a declarative module retrieves declarative information from long-term memory, a goal module manages the system's goals during problem solving, while a central production system module contains procedural knowledge in the form of production rules and is

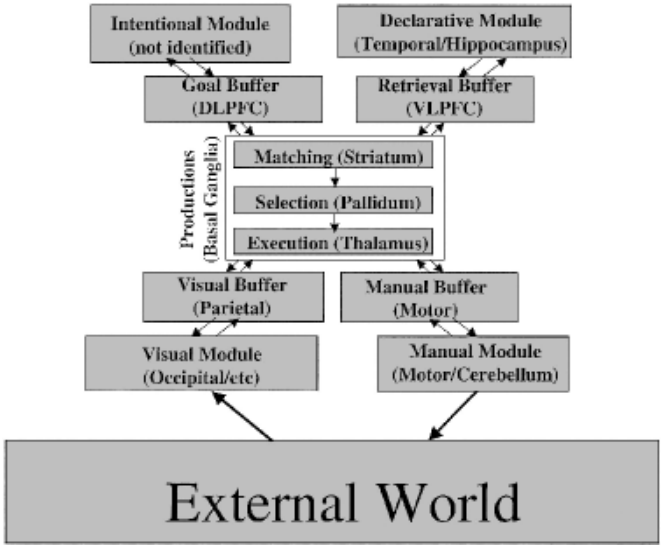


Figure 2.4: A schematic overview of ACT–R 5.0, reproduced from (Anderson et al., 2004). See text for description.

responsible for coordinating the behavior of all other modules. Figure 2.4 is a schematic overview of ACT–R 5.0.

Information is passed between the modules and the central production system through "buffers" that intendedly can only hold a limited amount of information, called "chunks", which are lists of attribute-value pairs. As many other design choices behind ACT–R, the decision to limit the buffer size to a limited amount of information is motivated by available data from psychological and neuroscientific studies. For example, the visual system only provides information on a single object at a time, in the form of an identity chunk and a location chunk (to model the ventral and the dorsal pathway of visual information processing in the brain, cf. Section 2.1), as it is assumed that the human brain also only processes information from a single attended visual stimulus at a time (apart from low-level visual signal processing, which is known to be massively parallel in nature).

The central production system matches the content of the buffers against the *IF*-portions of its stored productions, and selects a single production for execution. This is one of ACT–R's main differences to the Soar cognitive architecture, which allows the parallel firing of all matching productions. The selection mechanism of the single production to fire is based on a computation of expected utility of a production for the current goal and the state of the buffers. This computation comprises an estimation of how likely it is that the current goal will be achieved after selecting the production, an estimate that the system learns from experience in a Bayesian way. Thus, processing in ACT–R happens cyclically, with the production system reading the content of the buffers in the beginning of each cycle, then selecting a production and finally executing it at the end of the cycle. One cycle is assumed to take approximately 50 ms to complete,

while the processing in the other modules runs in parallel and asynchronously. Firing productions change the content of the buffers, by which the processing is coordinated across the modules.

The sensorimotor modules of ACT–R, i.e. the visual module and the manual module, are not implemented as to support actual sensory information processing or motor control, but rather simply simulate the approximate timing of the perceptual and motor systems and only process information at the interface level between the production system and the sensorimotor systems. On the one hand, the motor system is assumed to take as input symbolic command descriptions for actions to be executed. The visual system on the other hand is assumed to provide information on an attended stimulus in symbolic form as chunks, i.e. feature lists. The production system can influence the selection of the attended stimulus by providing top-down cues, such as "color: red" or "vertical: top".

2.2.2 Hybrid Architectures

The distinguishing feature of the cognitivist approach is that it assumes cognition to be based on the processing of symbolic representations. As a consequence, there is a clear separation of cognition into low-level sensorimotor processes and "higher cognition" (reasoning, planning, and the like). The latter is implemented as a set of programs that manipulate symbolic representations of the situations, and search through a collection of fixed association rules that are stored in long-term memory to find sequences of actions that will produce a goal state. Thus, a stable symbolic representation of the environment is necessarily required, therefore in this approach it is the responsibility of the sensory system to extract relevant information from raw sensory input into crisp symbolic representations, and to ground the symbols in the input signals. The motor system is thought to provide a set of basic actions, which are also represented in a symbolic format (for example "open gripper" or "say *apple*") to support symbolic planning.

The focus of cognitivism clearly lies on "higher cognition", which is studied in isolation from sensorimotor processes on an abstract level, whereas motor control and sensory processing are only marginally considered. Chiefly this is the case because cognitivist approaches mainly try to reach human-level performance in problem solving tasks, and are not primarily intended to control robots in the first place. The implicit belief behind this approach is therefore that limitations of current systems are to a large extent due to our insufficient understanding of the symbol-generating and symbol-grounding sensorimotor processes, but once satisfying solutions to these problems have been found, we can "plug in" the symbol manipulation systems for higher cognition and will have a robot with general intelligence.

The earliest attempts to construct robots with general-purpose problem solving capabilities go back to the work in the 1960s at Stanford University, where the robot "Shakey" was developed (Nilsson, 1969). Shakey was equipped with a television camera, a range finder and a bump detector, and had a mobile base so it could drive around. Its programming was based on a decomposition of the task into the three stages of

sensing, planning and acting, where the sensing stage generated an internal world model in a symbolic format, the planning stage used the world model to generate an action sequence that would achieve a given goal, and the acting stage would execute the action sequence. Shakey's architecture and later approaches that are based on a similar problem decomposition (e.g. Albus et al., 1989) are collectively referred to as beloning to the "sense-plan-act paradigm" (see e.g. Orebäck and Christensen, 2003).

However, the sense-plan-act paradigm has been found to be problematic, as planning in real-world scenarios can take long and the robot would be blocked while waiting for the planning to complete, and executing plans in a dynamic world without involving the sensors is dangerous (Kortenkamp and Simmons, 2008). To deal with this inherent difficulty of the sense-plan-act paradigm, it has become widely accepted among proponents of the cognitivist approach that the planning, or "deliberative", system needs to be accompanied by an efficient sensorimotor system that should not only provide an interface to the external world for the deliberative system, but can also reactively control the robot on its own (Orebäck and Christensen, 2003). The sensorimotor system is allowed to operate to a large degree independently of the deliberative system to ensure that the robot remains responsive at all times. Thus, the strict separation between the three stages of sensing, planning and acting is blurred, and instead the deliberative system only takes "top-down influence" on the sensorimotor system.

The deliberative sub-system is responsible for high-level functions, such as reasoning, planning, dialog management, etc., and is implemented using different symbolic representations for each of these functions. For the sensorimotor sub-system on the other hand, sub-symbolic representations for engineering sensorimotor responses are adopted that initially were developed outside the cognitivist paradigm, such as behaviors (see Section 2.3) or neural networks (see Section 2.4), as they are better suited for processing sensorimotor inputs and outputs under real-time constraints. Because of this marriage between symbolic and sub-symbolic components from different paradigms in one system, these architectures are called "hybrid" architectures.

The deliberative sub-system can be module-based, implementing the individual high-level capabilities in distinct modules. This has the advantage that specially tailored solutions to individual problems can be used: As the field of artificial intelligence has fragmented into many sub-disciplines (such as computer vision, speech processing, etc.), which each has developed its own specialist representations (Wyatt and Hawes, 2008), a modular system design can benefit from directly incorporating the individual solutions from the sub-disciplines. Module integration and inter-module communication is implemented using a dedicated software, which is commonly called the "middleware" (e.g. Fitzpatrick et al., 2008; Fritsch and Wrede, 2007).

One example of a hybrid architecture design is "3T" (Bonasso et al., 1997). It is composed out of three interacting layers or "tiers" (therefore the name): A reactive control layer, a sequencing layer, and a deliberative layer, see Figure 2.5. The reactive control layer is directly communicating with the robot's sensor and motor hardware and comprises a collection of simple behaviors, each of which tightly connects sensor input with motor output. Here, an important concept is "situated behavior" (Kortenkamp

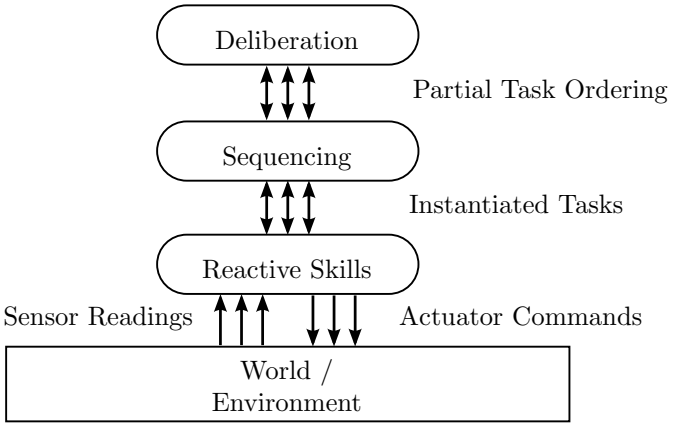


Figure 2.5: Schematic overview of 3T, a hybrid architecture design. Reproduced from (Bonasso et al., 1997). See text for description.

and Simmons, 2008): Many behaviors only successfully achieve a given short-range goal when activated in the correct context. For example, a behavior that should drive a robot down a hallway is only appropriate when the robot actually is situated in a hallway. Therefore, the next layer in the architecture, the sequencing layer, combines the behaviors into action sequences and attaches context information, such that behaviors are selected depending on the current context. Additionally, each action sequence descriptor provides information on the change in state that it achieves, if all behaviors execute successfully. This allows the third layer, the deliberative layer, to search through the library of action sequences and to compile plans.

2.2.3 Implications

The cognitivist paradigm to cognitive architecture is motivated by the assumption that cognition is based on the manipulation of symbolic representations and proposes a clear separation between high-level cognitive processes and low-level sensorimotor processes. An important aspect of cognitivist architectures is that they search through a library of associations, stored in a symbolic format in long-term memory (for example in the form of production rules), to find sequences of actions that will transform the current situation into a target situation. The paradigm has spawned many robots that have been successfully applied in a variety of tasks, especially in pursuing the hybrid architectural approach.

However, even though the goal of cognitive architecture is to create systems that are able to cope with *many* situations (as mentioned at the beginning of this chapter), the cognitivist paradigm has not been able to satisfactorily demonstrate this quality to date, despite its continuous development for several decades. As it was noted for example by Christensen, intelligent behavior needs to be very flexible, as the same behavior can lead to significantly different outcomes depending on the context (Christensen, 2004), a

fact that has also been recognized by the cognitivist community (cf. Section 2.2.2). To deal with context when applying its rule-based knowledge, a cognitivist system has to be extended with more rules that describe how a given context influences an outcome. However, a system that would exhibit a certain flexibility in its behavior would need to store a vast number of rules in its long-term memory. This casts an inherent difficulty for symbol-based cognitive systems, which is what one interpretation of the infamous "frame problem" states (cf. Clark, 2002): It would be computationally intractable to search through the entire database of rules to form a plan. What a general intelligent system working on symbolic representations would need, so it seems, would be a way to know what is relevant in a given situation. Some approaches have tried to deal with this by adding explicit symbolic knowledge about situations, such as lists of relevant items, for example by detailing out that balloons and cakes are relevant to birthday parties (Minsky, 1974). Yet, this still faces the problem that the system needs a way to determine what situation knowledge currently applies, which is again not a trivial task. This unsolved problem of systems using symbolic representations limits their applicability to rather sterile laboratory environments (cf. MacDorman, 1999), where the robot only faces situations that are sufficiently well covered by its database of rules, which in turn is not extensively large due to the limited scenario size.

The nature of flexible behavior seems to be more related to the ability to recognize and make use of spontaneous "opportunities to interact" with the environment (Christensen, 2004; Gibson, 1977), instead of relying on a tremendous amount of rules that apriori determine what is relevant and what not. To achieve this, the system needs to be in a much closer relationship of co-ordinated interaction with the environment. Instead of trying to construct an internal model of the world and using this copy to decide on what to do, a robot should use the world itself as the primary source of information (Brooks, 1991b; Dreyfus, 1972, 2007), thus removing the burdon of having to maintain an up-to-date representation of the required information and instead directly using the sensors where information is available. In the same direction, hinting to the evaluation that the complexity of a symbol-based system seems to be out of balance with the problem complexity when it comes to physical interaction, Pfeifer and Scheier argue that only in "ecologically balanced" systems a successful co-ordination with the environment can occur (Pfeifer and Scheier, 1994). A system is ecologically balanced, if there is a match between the complexity of the sensory and the motor system, while it would be out of balance to have a very sophisticated visual system (for example with the goal to establish a symbol grounding) when ultimately only simple motor acts are performed by the robot.

Other significant problems that symbol-based approaches face are related to the fact that the knowledge of the system has to be provided by the human designer in advance, for example in the form of production rules in the case of Soar and ACT–R. While this has the apparent benefit of producing "transparent" systems, as the symbols that the system uses are "meaningful" to the human designer who can thus interpret the steps that the system took in solving a problem, at the same time it can be argued that it is a limiting factor as the programmer-dependent representation biases the system

and constrains it to an idealized description (Vernon et al., 2007). Furthermore, the amount of knowledge that has to be provided to the system increases dramatically with the complexity of the robot's task, which quickly renders it unfeasible to formulate the necessary knowledge in advance (Stoytchev, 2006; Weng et al., 2001).

Thus, on a practical level the symbol-based approach to cognitive architecture faces the problem that it only performs well in small-scale scenarios, in which the human designer has envisioned all possible situations that the robot will face and has carefully prepared the system in advance, which is the exact opposite of the target of creating "general intelligence". Additionally, also on a more theoretical level, a growing number of criticisms of the cognitivist view have been formulated over the last decade. Essentially the argument is that cognition cannot be accounted for by symbolic computation, which instead is merely an idealized description of emergent properties of the dynamic sensorimotor processes that actually amount to cognition, and that consequentially an alternative research program is needed in cognitive science to replace cognitivism (e.g. Barsalou, 1999, 2008; Beer, 2000; Clark, 2001; Dreyfus, 2007; McClelland et al., 2010). Candidates in the current cognitive science literature are connectionism and dynamicism, which will be described in Sections 2.4 and 2.5, respectively. Apart from these, the "behavior-based robotics" movement has proposed an alternative to cognitivist modeling in robotics, which will be described next.

2.3 Behavior-based Robotics

As an approach to programming robots that is radically different from the cognitivists' symbol manipulation approach, the "behavior-based robotics" paradigm was introduced in the 1990s, championed by Brooks and his group at the MIT (e.g. Brooks, 1991a). Here it was demonstrated that it is possible to build robots that solve real-world problems without any form of "high-level" cognitive processing. Several simple behaviors are designed and programmed that operate at all times in parallel, instead of making the system rely on a loop that alternates between perception, abstract computation and action. The behaviors compete against each other, and the winning one gains control over the hardware.

The behavior-based robotics paradigm is intended by its proponents to replace the cognitivist paradigm for building robots that can act in the real world (Brooks, 1986; Pfeifer and Scheier, 1994). It is argued that instead of studying complex systems in simple environments before scaling up the complexity of the environment, one should begin with studying simple systems in the real world and then begin to scale up the complexity of the system (Brooks, 1991a). The main motivation behind this paradigm is to make robots "reactive" to environmental cues, by trying to minimize the time between using the sensors and sending a command to the motors. Instead of making a central planning stage responsible for generating all kinds of behaviors, every behavior that the robot can produce is implemented in its own sub-component of the system. Behaviors are usually implemented in ways that generate motor commands as a simple function of the sensory input, for example by interpreting the readings of a range

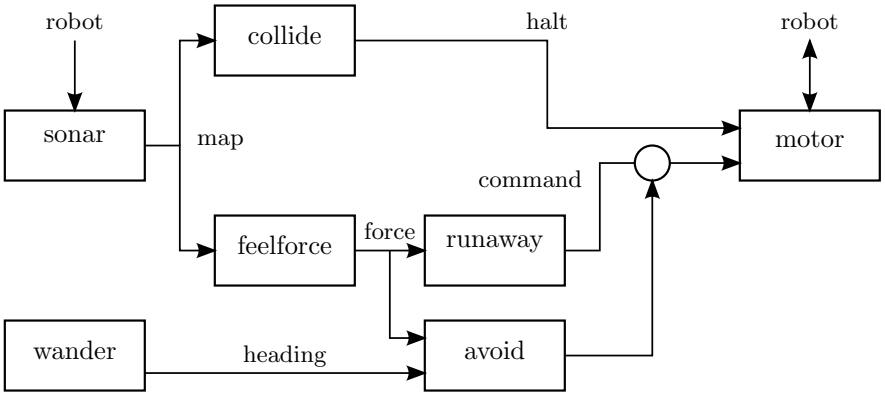


Figure 2.6: Example of the subsumption architecture for a robot that can wander around and avoid obstacles. In this example, the "avoid" module subsumes the "runaway" module, meaning that the former, when it is active, can overwrite the output of the latter. Drawing reproduced from (Brooks, 1986).

finding sensor as a "force" that pushes the robot away from obstacles (Arkin, 1989). Examples for typical behaviors are *approach-the-object* or *avoid-bumping-into-obstacles*. By providing the system with a set of these simple behaviors it is ensured that the most basic functionality, such as not bumping blindly into obstacles, is always at work.

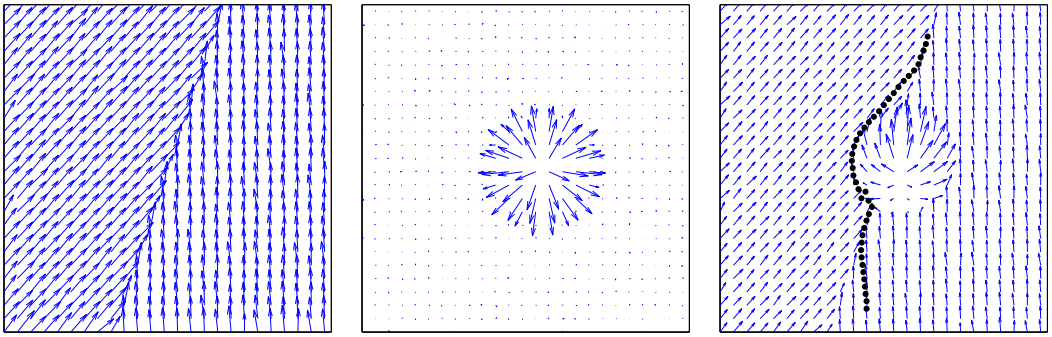
2.3.1 Computational Models

The Subsumption Architecture

The first and most influential architecture that was proposed in the behavior-based tradition was Brooks' "subsumption" architecture (Brooks, 1986). Behaviors are implemented as modules that read values from a set of inputs and generate output values, which can be sent to the motors or can be used by other behaviors. The architecture is composed out of layers of behaviors, where behaviors in higher layers "subsume" behaviors in lower layers, meaning that they can inhibit their inputs and overwrite their outputs in situations where the lower level behaviors are uninformed or to implement longer term goals (for example, if the object in front of the robot is its charging station and the robot needs to charge up, then avoiding a collision with the station should be inhibited and instead a docking maneuver should be initiated). This design principle ensures that only a single behavior will send its output to the motors at any time, as higher-level behaviors will overwrite the outputs of lower-level behaviors, thus there will be no conflicts between behaviors that produce outputs for the same motors. Figure 2.6 shows a schematic for an example of the subsumption architecture.

AuRA

Arkin's "Autonomous Robot Architecture (AuRA)" (Arkin, 1989), another example of a behavior-based robot architecture, implements behaviors entirely as potential fields



(a) (b) (c)

Figure 2.7: An example of how AuRA uses potential fields in a navigation task. (a) The task to navigate along a path is implemented as an attracting force with a constant drift in the direction of travel. (b) When an obstacle is sensed, a repellent force is added in its immediate surrounding to avoid a collision. (c) The combined potential field lets the robot follow a path, here shown as black dots, that follows the desired path but avoids the obstacle.

of forces: Obstacle-avoidance behaviors are repellent force fields, navigation behaviors are attracting force fields. The rational behind using the potential field formulation is that multiple behaviors can be combined simply by adding their forces together, which results in a combined potential field. The robot monitors its environment and instantiates new behaviors on the fly, for example when it senses an obstacle with its range finders. As long as the obstacle is close, a repellent force is added in the calculation of the current motor command. Figure 2.7 is an example of how AuRA performs in robot navigation.

The potential field method is very specialized for navigation tasks and is prone to local minima, as for example the robot can easily get trapped in simple dead-end constelations of obstacles (Koren and Borenstein, 1991).

2.3.2 Implications

Behavior-based robots can cope easily with dynamic environments, as they continuously monitor their sensors and almost immediately react with motor responses. The behavior-based paradigm is known for producing robots with insect-like behavior. However, it has become clear that the design of purely behavior-based systems that should achieve long-range goals and of behaviors that are more complex than "just" insect-like is very difficult (Kortenkamp and Simmons, 2008). Also, behaviors are pre-defined by the designer of the robot, and the learning of new behaviors is not addressed.

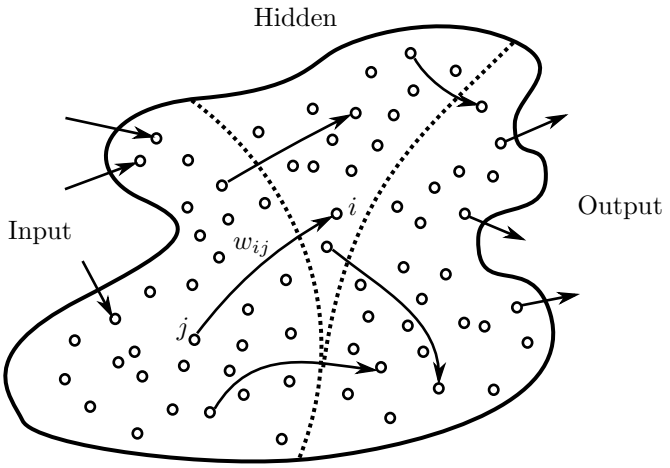


Figure 2.8: Schematic drawing of a connectionist system. Reproduced drawing from (Rumelhart and Todd, 1993). See text for description.

2.4 Connectionism

Connectionism is the study of artificial neural networks and other networks of neuron-like units as information processing devices (Medler, 1998). Connectionist systems operate by the parallel computation of numeric activation values of their units as a response to the presentation of input. They store information in a distributed sub-symbolic format, instead of explicit logic rules. While artificial neural networks had already been studied earlier (most notably the perceptron, see Rosenblatt, 1962), the idea was made popular in cognitive science and other disciplines in the 1980s by Rumelhart and McClelland and the PDP group ("parallel distributed processing", McClelland and Rumelhart, 1986; Rumelhart and McClelland, 1986). Connectionist systems touch important aspects of cognition that had been handled poorly by cognitivist approaches, such as learning, generalization capabilities, graceful degradation and content-addressable memory (cf. Medler, 1998). These systems are based on the parallel operation of many simple processing units that are connected as a network, and store information implicitly in the form of numeric values, the connection weights, which are assigned to the connections between the units. Information is processed by a parallel computation of activation values for the units. A subset of the units, the "input units", are allowed to be externally activated, while the activation values of another subset of units, the "output units", determines the output of the system, see Figure 2.8.

Knowledge is not given to the system explicitly, as in cognitivism in the form of logic rules, but is learned by the system through the processing of training samples. Learning methods extract statistical information from sample pairs of input and output values, and the network adapts its connection weights to approximate the training data. Thus, when the connectionist methodology is applied to a given task, it needs to be decided on the network structure on the one hand and the learning method on the other, and suitable training data has to be collected.

Example connectionist models from the cognitive science literature include networks that can categorize words according to their lexical role (Elman, 1991), networks that learn and implicitly represent structural similarity as in semantic maps (McClelland and Rogers, 2003; Rumelhart and Todd, 1993), and networks that select action primitives in routine tasks (Botvinick and Plaut, 2004; Cooper and Shallice, 2000), to name only a few. In machine learning, neural networks are commonly applied as a method for function approximation and classification tasks (see Bishop, 2006). In robotics, a common example is the learning of the forward and inverse kinematics functions (i.e. how the angular positions of the robot's joints determine the positions of its parts) from self-generated examples (e.g. D'Souza et al., 2001; Herbort et al., 2010; Jordan and Rumelhart, 1992; Reinhart and Steil, 2009).

All of the just mentioned works solve a particular task with a dedicated connectionist model. Thus, these works show how *specialist* systems can be built that are flexible in solving a specific task, but ultimately are limited to that single task. The goal of cognitive architecture however, as already mentioned, is to develop a system with more *general* capabilities, which can perform well in many tasks. It seems unreasonable to assume that constructing a system as a bag of specialist components, following a "one net, one task" policy, would lead to a success, and it is also implausible to argue that it would suffice to simply increase the complexity (such as the number of units) of a given model, as the success of these models strongly depends on a careful selection of valid training data (cf. Clark, 2001, ch. 4). Instead, approaches to cognitive architecture in the connectionist paradigm introduce a second level of structure, which defines how individual sub-components are interconnected so as to produce a coherent system. In the following, candidate connectionist architectures from the literature will be described.

2.4.1 Computational Models

Brain-Based Devices

The group around Krichmar and Edelman has developed a series of neural-network-based cognitive architectures for mobile robots, which they call "brain-based devices", following the research methodology to begin with developing a simple but functioning system based on a neural network model, and continuously increasing the complexity of the model by adding more details, such as neuromodulation, to iteratively produce a more and more realistic implementation of an artificial brain (e.g. Fleischer and Krichmar, 2007; Krichmar and Edelman, 2002, 2005; McKinstry et al., 2006). One example is the robot "DARWIN VII", which can learn to visually distinguish two types of objects (Krichmar and Edelman, 2002). The robot's system is composed of a set of behaviors on the one hand, and several interconnected neural networks on the other, and the goal of the robot is to learn about the properties of the different objects by physically interacting with them. The layout of the system, in terms of which neurons in which neural networks are connected how as well as which neurons activate a behavior or receive activation from the sensors, is specified by design, while the weights of all

connections are learned online. After being activated, the robot's behavior system lets it drive around and approach the objects that are scattered on the ground. The robot uses a gripper to "taste" the objects with a sensor in the gripper that measures their conductivity: Striped objects are strongly conductive and "tasty", and dotted objects are weakly conductive and "disgusting". At first, the robot has to approach and "taste" every object. An "innate reflex" is triggered when a bad tasting object is gripped, initiating an avoidance behavior. After several trials, the system's neural networks have learned to associate the visual pattern of the dotted objects with the bad taste, and activation from the visual input is propagated via the learned associative connections to motor representations for the avoid behavior. This lets the robot avoid bad-tasting objects instead of approaching and gripping them. Thus, the system's trained neural networks extend innate behaviors by associating new sensory inputs with the motor responses.

In models that are composed of several interconnected neural networks with different types of pre-specified inter-network connections, such as the one of Krichmar and Edelman (connections in their model can be either static excitatory, plastic excitatory, or inhibitory, and can depend on a special "value" input or not), the result of training depends on the exact layout of the whole system, i.e. which networks are how connected. Furthermore, artificial neural networks have a limited capacity for how much information they can store. If they are continuously adapted to new inputs, this can lead to a situation where previously trained information is lost as the network adapts to the new inputs, a problem that is known as "catastrophic interference" (French, 1999). Thus, it is a problem to extend an already trained system with a new skill. This is especially problematic in the case of cognitive robots that should continuously learn new skills by interacting with their environment.

Epigenetic Robotics Architecture

Morse et al. have proposed the "Epigenetic Robotics Architecture" (ERA, Morse et al., 2010) with the research goal to provide modeling constraints for a cognitive architecture. They identify three main requirements for a cognitive architecture: (i) It should not be the goal for the design of a system to make it perform perfectly in a designated niche, but it should rather display *ongoing development* through interaction with the environment. (ii) The system's knowledge, gathered throughout its development and emergent from its interaction with the environment, should be organized in *recognizable conceptual structures*, making it *transparent* for investigation. Morse et al. argue that otherwise, if the acquired knowledge is not transparent, we will ultimately not learn much about the nature of human conceptual knowledge, even when the system has achieved it. (iii) Lastly, the system should not be bound to a single domain of cognitive performance, but should be *scalable* and allow to *integrate* a wide range of phenomena.

ERA is intended to provide "modeling guidelines" for the modeling of a cognitive architecture, meaning that their formal implementation is not to be seen as a finalized description of a cognitive architecture. It should rather be seen as a framework, which they claim is consistent with emergentist theories of cognition, and can support further

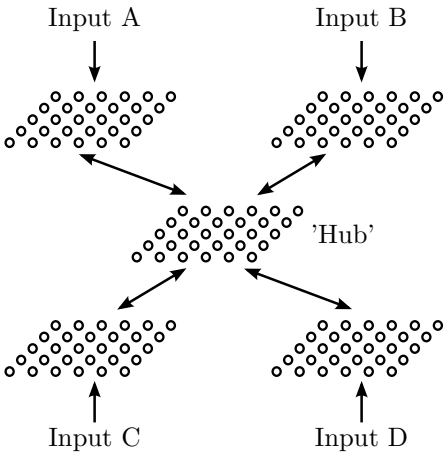


Figure 2.9: The "basic ERA unit": Multiple SOMs are activated by different inputs and are associated with each other via bi-directional connections to a dedicated "hub" SOM. Connection strengths are trained in a Hebbian way by strengthening connections between winning units in the different SOMs. Drawing reproduced from (Morse et al., 2010).

extensions without overly constraining the models (Morse et al., 2010).

The architecture is defined as a network of building blocks, the "basic ERA unit", which itself is composed of several interconnected neural networks: An ERA unit is a structured association of multiple self-organizing maps (SOMs; Kohonen, 1982), where one of the maps is declared a "hub", see Figure 2.9. All other maps are reciprocally connected to the hub, connecting each neuron in the hub with all other neurons, and Hebbian learning is applied to learn the connection weights. ERA commits to the theory of "conceptual spaces" in the question of how information is represented, proposed by Gardenfors, which states that entities are represented as points in high-dimensional feature spaces, along dimensions such as size, color, weight, and so on, and concepts correspond to regions in these spaces (Gardenfors, 2004). Hierarchy in ERA is implemented by associating the hub SOMs of several ERA units with another SOM, which is then the hub of an ERA unit on the next hierarchical level. Information processing is implemented as a mechanism of spreading activation along learned associative connections.

The main function of an ERA unit in the simulation experiments described by Morse et al. is that of "priming": By allowing activation to spread via the reciprocal connections to units that have been previously co-activated, units become active also when there is currently no direct input to them. Morse et al. exploit this property to implement a model of the behavior observed in a label-learning study with 18- to 24-month-old infants, where it was demonstrated that the locations at which objects have been previously presented is an important cue for infants when learning a new label for the objects, and can even be a stronger cue than the visual appearance of the target object (Smith and Samuelson, 2010). Morse et al. demonstrate that by driving the robot into a body posture and thus "priming" the color feature that was previously

associated with that posture, a label can be associated to the object in its absence, and report that the performance of their model, in terms of how often the label is associated with the correct object across trials, is similar to that of infants as it was observed in the original study (Morse et al., 2010).

As ERA is inteded as a framework for modeling cognitive architectures, the specification of the ERA unit (SOMs with associative connections) leaves quite much room for adjustment. Many issues remain unaddressed, especially in relation to motor control, so that more specific mechanisms would have to be defined and implemented. For example, if we assume a more complex scenario in which more than just three SOMs are connected to each other, and there are many associative connections between them, then the simple mechanism of spreading activation that was used in Morse et al.'s experiment would quickly lead to an explosion of activation across the whole system, where the majority of neurons would become active after just a few iterations, even if only a few neurons had been activated by actual input. In an implementation of a complete cognitive architecture that is based on ERA as a framework, one should expect a substantial redefinition and extensions of the underlying mechanisms.

2.4.2 Implications

Connectionist models focus on learning knowledge through the discovery of statistical regularities in training examples instead of hardcoding rules into structured representations. In turn, the success of the learning is dependent on the employed learning algorithm and the selection of training examples. In most connectionist models, training samples are selected by the designer and only contain relevant and informative samples, which is a requirement for learning methods to work, as they can handle "noise" only up to a certain degree. However, this cannot be assumed for a cognitive architecture that should be able to learn in natural settings. For example, a common method to let a robot learn how it can move its own hand is to let it fixate on said hand with its cameras while randomly moving its arm around (e.g. Metta et al., 1999). During this process, the system can generate training examples by associating each arm posture with the corresponding position of its hand, as measured with its cameras. This can then be used as training data for a neural network. However, this only works as long as the training samples are only collected during times where the robot indeed fixates on its own hand. If it tracks something else in between and includes the data as training samples, this adds as noise to the training process. To prevent this from happening, the system requires some way of maintaining that only valid training samples are used (this issue will be discussed in more detail in Chapter 5, where also a solution will be proposed), for example by hardcoding a hand detector. However, assuming that the system knows apriori which training samples are relevant equates to assuming that it is fixed by design what needs to be learned, which would open up the system for the same kind of criticism as that against cognitivist systems. The described connectionist architectures circumvent this problem, as the scenarios in which they are evaluated are so limited in size that nothing irrelevant ever occurs.

2.5 Dynamicism

For now more than a decade, dynamicism is a "third contender" (Eliasmith, 1996) as a theory for cognition next to cognitivism and connectionism. Dynamicism advocates the view that cognitive agents are complex dynamical systems (van Gelder, 1998), not only in the evident sense as they dynamically control their body, but in the sense that they actually are made up of interconnected dynamical systems. According to this view, it is the dynamic interplay and the interaction of these coupled dynamical systems from which overt behavior of the agent emerges (Barsalou et al., 2007; McClelland et al., 2010; Thelen and Smith, 1996; van Gelder, 1998). This includes neural networks in the agent's brain, the agent's body as a (bio-)mechanical dynamical system, as well as the environment (Beer, 2000). All of these processes are intrinsically intertwined and let cognition unfold continuously in time as the result of internal and external forces (Erlhagen and Bicho, 2009). In this sense, the brain is "just another participant" (Clark, 1997, p. 479) in a complex process from which cognition and behavior emerges.

Dynamicism rejects the assumption that cognition is based on mapping external reality onto an internal representation (van Gelder, 1998). This is certainly at odds with the cognitivists' view of cognition as a process of symbol manipulation. Moreover, while many artificial neural networks are as a matter of fact dynamical systems, some dynamicists also feel unsettled with connectionism as a research program and criticize it for still being "too much like traditional cognitive theory" (Thelen and Smith, 1996, p. 42), as many connectionist works aim at demonstrating symbol-like capabilities in neural network models. Inputs are processed to establish a pattern of activation across the network units as an internal representation of some external state of affairs. In contrast, input to dynamical systems models rather serves as a perturbation of the intrinsic dynamics of the system (Beer, 2000).

The dynamical systems approach to cognition assumes that the cognitive system and the environment, both being complex dynamical systems, are tightly coupled with each other in interaction. In contrast to purely reactive agents however, as for example behavior-based robots designed according to the subsumption architecture (see Section 2.3.1), the dynamical systems hypothesis maintains that the cognitive system has its own internal state, which influences its next action. Thus, the system is not merely reacting to external cues, but can initiate behavior and organize future action in anticipation of expected environmental circumstances. It can react differently to the exact same environmental circumstance, as its state can be different, and the trajectory of its state can depend on its past experience (Beer, 1995, 2003).

As dynamicism is a relatively new development in cognitive science, most theories can be said to currently still be case studies rather than sophisticated accounts, and computational models are provided for "minimally cognitive agents" (Beer, 2003) rather than robots. One notable exception is the "dynamic field theory", which is currently gaining momentum both in empirical work, as well as in computational modeling in robotics. Therefore, in the next section, the dynamic field theory will be described as a prime example of a dynamical systems theory of cognition. Computational models

of cognitive architecture in the dynamicist paradigm, including, but not limited to, models of dynamic field theory follow in Section 2.5.2.

2.5.1 Dynamic Field Theory

The dynamic field theory (Erlhagen and Schöner, 2002; Johnson et al., 2008; Schöner et al., 1997; Smith and Thelen, 2003; Thelen et al., 2001) has originally been developed as a theory of how movement preparation is realized in the brain, and provides insights into how the integration of information and decision making are realized in an embodied format. As an example, consider the questions of how reaching movements or saccades are prepared (Schöner et al., 1997): The target location for a reaching movement can be specified by visual cues, auditory cues, somatosensory cues, but also from memory. And in the presence of ambiguous information, for example when several objects are present which could potentially be reached for, a single target location has to be picked (Schöner et al., 1997). Moreover, when acting under time pressure, a decision might need to be made under uncertain circumstances (i.e. before all available information could be carefully processed). In psychophysical studies it has been demonstrated that in a task where subjects had to produce cued movements where there was too little time to properly react to the given cue, the subjects would produce a guessed movement according to which cue could be expected (see Erlhagen and Schöner, 2002). If one of the cues was more likely to occur than the others, subjects would tend to produce the movement corresponding to that cue more frequently than the other movements.

In the cognitivist view, all of these processes would happen in the cognitive processing stage: After sensory information from all modalities was extracted, all available information would be integrated into an abstract world model, on the basis of which a decision for an action would be made. This action would then be executed by the motor stage. In contrast, the dynamic field theory proposes a model which can account for these aspects on the basis of dynamic sensorimotor processes alone.

The theory is built around a specific type of neural network model, which was originally proposed by Amari (Amari, 1977). It is a model of a neural field, i.e. not simulating the behavior of individual neurons, but rather of a piece of neural tissue to describe the firing pattern of populations of neurons. This kind of model is motivated by the fact that the number of neurons and synapses in even a small piece of cortex is so immense, that it can be meaningfully studied in the limit case, where space is continuous (individual neurons are not modeled) and an activity landscape describes the mean firing rates of neurons in the tissue (Coombes, 2006). The specific model of Amari assumes that neurons are laterally connected in two ways: Neurons lying close to each other are mutually excitatory, while neurons lying at range from each other are mutually inhibitory (Amari, 1977) (a more detailed discussion of Amari's model will follow in Section 4.2.2). This connectivity pattern yields a number of interesting dynamical properties in the activation pattern of the neural field, most important of which, in the context of the dynamic field theory, is a competition-like behavior: When input is presented to the neural field at different locations, the dynamics of the field will eventually select a single location in the field which will remain active, while all other

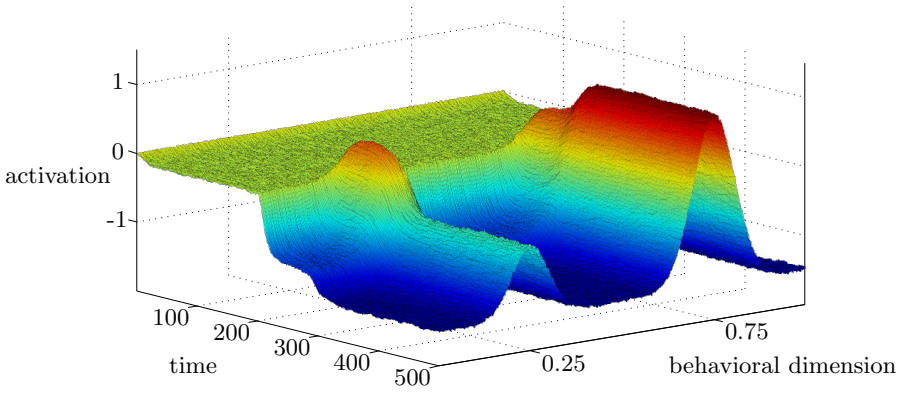


Figure 2.10: Example of the evolution of activity in a one-dimensional dynamic neural field over time. Upon initialization, the field activity goes to a resting level, which is homogeneous across the whole field, with only little differences in activation due to random noise. At time 200, two equally strong inputs are presented to the field at locations 0.25 and 0.75 in the behavioral dimension. The field reacts by first raising activation of neurons near the candidate locations, thus being "uncertain". At around time 300, the input at location 0.75 has won the competition due to minimal differences caused by random noise, and dominates the activation in the form of a single strong peak of activation above threshold (in this case zero-level).

locations are inhibited by the winning group of neurons. The dynamics of the neural field continuously updates the activation of the field, depending both on the current input as well as on the current state of the field itself, therefore the activation in the field evolves over time. When one location in the field has a strong enough activation level, it can spontaneously develop into a self-sustaining peak of activation, which will suppress activation at other locations in the field (Amari, 1977; Erlhagen and Schöner, 2002), see Figure 2.10 for an example.

The dynamic field theory assumes that any movement is represented by a number of parameters, such as the spacial direction of the movement, the peak velocity, the level of force needed, and so on, which are referred to as "behavioral dimensions" (Erlhagen and Schöner, 2002; Schöner et al., 1997). Independently of how exactly the movement is executed and controlled, the specification of a particular movement can be seen as the assignment of particular values to these parameters (Erlhagen and Schöner, 2002). In the dynamic field theory, parameter values are represented as activations in dynamic neural fields that are spanning the behavioral dimensions for a movement. For example, to represent the target for a reaching movement, a dynamic neural field is used in which the topographical organization corresponds to the Cartesian space of possible target locations. The parameter spaces that are spanned by the behavioral dimensions are continuous (Erlhagen and Schöner, 2002), as parameters describing actions normally come from continuous domains (e.g. Cartesian space for target locations, a continuous

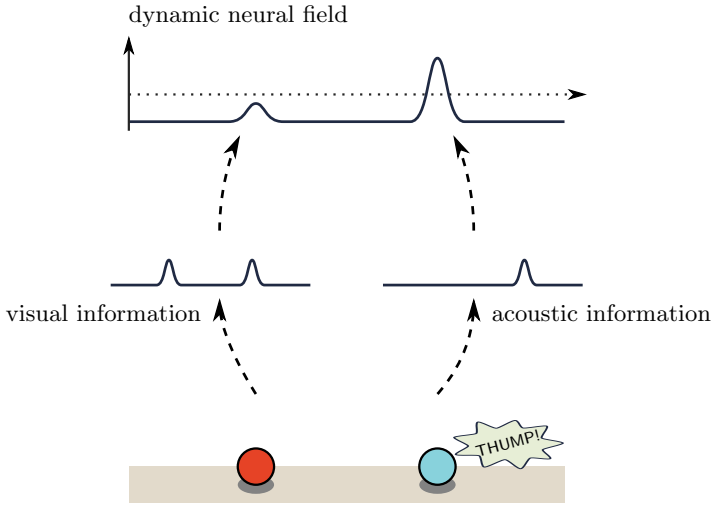


Figure 2.11: Example of integration of information in a dynamic neural field. Two target objects, a red and a blue ball, represent candidate locations for a grasping action. The blue ball additionally produces a sound. Visual perception of the scene generates two activation peaks as input for the neural field, auditory information only produces a single peak of activation. Both inputs are presented to the dynamic neural field. The field dynamics responds with a single strong peak of activation at the location of the blue ball, since the superposition of the two inputs produces a stronger input at that location.

scale of force, and so on). Neural fields therefore can naturally be used as a representation, since their topographic organization describes a continuum of values. Although the layout of actual cortical tissue is limited to two dimensions, an extension of neural fields to higher-dimensional spaces is mathematically speaking straightforward[1]. Thus, also parameters such as a location in three-dimensional space, or the joint position in angular coordinates of a robot arm can be represented in high-dimensional neural fields.

The input to the dynamic neural field can come from different sources of information. For example in the case of a field for reaching movement preparation, low-level visual and auditory information could be used as an input for the field (seeing objects lying somewhere, or hearing a characteristic sound of an object from some direction), see Figure 2.11 for an example. Integration of several inputs to a dynamic neural field is modeled as a superposition of the input signals. Other than coming from sensory sources of information, input to a field can also represent for example task knowledge, i.e. knowledge about locations where objects typically are (Schöner et al., 1997). Thus, integration happens at the level of movement parametrization, where information from different sources are combined to form candidate movement parametrizations.

When the activation level of a peak becomes strong enough to exceed a certain

[1]This does not mean that high-dimensional neural field representations cannot exist in the cortex. The brain would only have to solve a mapping from a higher-dimensional layout onto a two-dimensional tissue using neural wiring.

threshold, then a movement is initiated (Erlhagen and Schöner, 2002). This way, the field dynamics implements the decision-making of the system on the sensorimotor level of representation. Erlhagen and Schöner propose to regard the threshold that determines when a movement is initiated as variable (Erlhagen and Schöner, 2002): Under normal circumstances, the threshold is set relatively high, so that movements are only initialized when a strong peak of activation has evolved in the field, thus representing a clear decision for a parameter value. However, if the system is under some kind of pressure to react (as for example when in a study a subject is instructed to perform a movement upon hearing a signal tone), the threshold could be artificially lowered, so as to trigger a readout of a parameter also when there is no clear winning location in the field. Since the activation in the field evolves over time, this might result in a situation in which a parameter value is read out shortly after a change to the inputs of the field has occurred, and the field is still transitioning between states.

Thus, the dynamic neural field model of movement preparation has various dynamical properties that determine the executed movement, depending both on the spatial layout of locations in the input signals (due to the lateral interactions in the field), as well as on timing. The interesting observation that makes the dynamic field theory attractive, is that these dynamical properties correspond in several ways to the behavior of subjects in psychophysical studies. For example, Erlhagen and Schöner used a dynamic neural field model to reproduce the results of a study in which participants were instructed to produce a certain amount of force with their index finger at a specific point in time (after hearing a metronome click several times) (Erlhagen and Schöner, 2002). The amount of force they had to produce, which was either a low, medium or high amount of force, was determined by one of three visual cues that was presented to them shortly before the time when they had to perform the action. Thus, there was a short interval of time in which participants could mentally prepare the movement. Importantly, participants had to guess the movement if this interval was too short to react. What was observed in the study was that, if the time interval was too short (less than 125 ms), participants tended to produce an amount of force that was at the center of the force scale, as a sort of "best guess". This was evident as the responses formed a wide distribution centered around the middle of the force scale. In contrast, if participants had enough time to react (more than 250 ms), the distribution was narrow and was centered around the target level of force. The surprising finding of the study was that the shift from the wide distribution around the center of the scale to the narrow distribution around the target levels did not occur abruptly, so that the one pattern would be found if the time was too short to react while the other would be found if reacting was possible, but the shift occurred continuously with the center of the distribution wandering off *continuously* from the center to the target value. Thus, if participants had a short amount of time to process the visual cue (less than 250 ms but more than 150 ms), they tended to produce amounts of force that were lying *in between* target values. These experimental results cannot be explained by a rule-based cognitivist approach, but can be naturally accounted for by assuming the involved processes to be dynamical systems. Erlhagen and Schöner propose that the

observed pattern of behavior is related to a property of dynamic neural fields, by which the peak of activation gradually wanders off to a new location of high input, if it lies close to the current location of the peak (for detail, see Erlhagen and Schöner, 2002).

The dynamic field theory has also been utilized to propose a different view than the standard view on the processes behind cognitive development. Development is usually seen as the continuous acquisition of new skills as being added to an existing repertoire, where having learned one skill can enable the learning of others. Smith and Thelen have proposed to instead think about development by regarding cognition as a complex dynamic system, and development as change to its state space (Smith and Thelen, 2003). They illustrate their idea through a dynamic field model of the so-called "A-not-B error" (Thelen et al., 2001), a behavior pattern that is found in 8- to 10-month-old infants (Piaget, 1997 [1953]): In an experimental setup, where there are two lids placed in front of the infant, the experimenter presents a toy to the infant and then hides it under one of the two lids (the one at "location A"). When the infant is then allowed to reach for the lids, he or she will most likely reach for the one where the toy was just hidden to reveal the toy. After this is repeated for several times, the experimenter hides the toy under the other lid (at the "B location"). Even though the infant saw the toy being hidden under the B-lid, he or she will likely still reach for the A-lid where he or she had found the toy in the previous trials. Infants of more than 12 months of age however will correctly lift the B-lid. This qualitative change in behavior between the 10-th and 12-th month has traditionally been accounted for by the development of an "object-permanence" concept, the understanding that objects continue to exist when they are out of sight.

The model proposed by Thelen et al. gives an alternative account for the "error" (Thelen et al., 2001). It is a dynamic neural field model similar to that of Erlhagen and Schöner, but with an additional neural field that has a slower dynamics: Peaks of activation in this field remain active for a longer period of time when input to them is removed (the field has a slower decay). This "memory field" is reciprocally connected to the motor field, it thus receives as input the activation in the motor field. This gives the system a kind of memory of which decisions it has made across the last few trials. The activation in the motor field however is volatile so that the time in between trial is enough to let the activation in that field go to its resting level. In the simulation of the A-not-B experiment, the memory field builds up a peak of activation at the A-location across the initial A-trials. This serves as a kind of pre-activation, or "habituation", of the A-location for the system. Then, in the crucial B-trial, the hiding event produces a transient strong input at the B-location, but as this input disappears, the fast dynamics of the motor field "forgets" about the hiding event at the B-location and is dominated by the input at the A-location coming from the memory field, which causes the system to produce the A-not-B error (Smith and Thelen, 2003; Thelen et al., 2001). When interpreting the neural field model as an actual model for the processes in the infant's brain, then the infant's producing the error is not due to a missing object-persistency concept, but timing is crucial for the error to occur: If the time interval between the hiding event and the reaching movement of the infant is reduced, then the transient

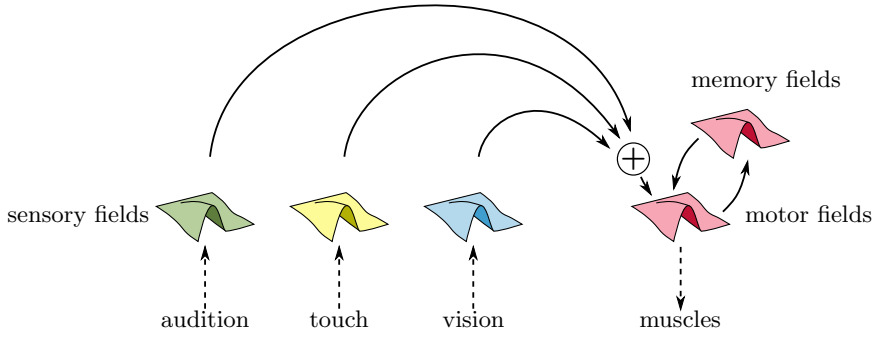


Figure 2.12: A rough illustration of how dynamic neural fields interact when combining the models from the literature on dynamic field theory. Modality-specific dynamic neural fields represent parameter spaces, on the sensory side for example of visually perceived locations or colors, and on the motor side for the preparation of movements, for example as target locations or force levels. Integration of information coming from multiple sources happens as a superposition of input signals, here exemplarily depicted as the integration of information from multiple modalities for movement preparation. Additionally, fields that do not receive direct input from the senses exist, such as in the case of memory fields that are reciprocally connected to other fields.

input of the hiding even should not yet have disappeared from the activation of the fields. Thus, the model predicts that when the time interval is reduced, i.e. infants are allowed more quickly to reach for the lids, then also the 8- to 10-month-old infants should not produce the A-not-B error, which is indeed the case (Smith and Thelen, 2003). Thus, Smith and Thelen argue that the development that occurs between the 10-th and the 12-th month of age in infants should not be seen as the acquisition of a single new mechanism, but that the change is to be found in the dynamics of a complex, distributed dynamic system (Smith and Thelen, 2003).

All of the work on dynamic field theory that has been cited so far was related to motor and perceptual tasks. It describes a detailed dynamic neural model of how information is integrated and decisions are formed on the basis of low-level sensorimotor representations in the neural fields. As such, the theory provides helpful insights into what processes operating on sensorimotor representations might be involved in perception, decision making, and working memory. It assumes representations in topographically organized neural field, to which input is mapped from different sources. The neural fields code information as continuous activation landscapes, where hypotheses are constantly competing against each other. Hypotheses are formed from inputs coming from the senses, as well as from connections between neural fields, for example in the case of memory fields that build up memory traces of past motor responses. Figure 2.12 is a schematic representation that should illustrate how the dynamic field theory describes the interconnections between neural fields, as well as in- and outputs.

From what is known about the structure of the cortex, dynamic neural field representations would most likely be found in the sensorimotor cortices, since topographically

organized representations have been identified in those regions (e.g. Amari, 1980). And indeed, the fact that predictions made by the theory about how changes to stimuli influence overt behavior and responses of subjects could be tested and confirmed in follow-up studies, seems to support the model. This, and the fact that work on dynamic field theory concentrates around sensorimotor aspects of cognition, could lead to the conclusion that the dynamic field theory can only explain data related to low-level sensory and motor information processing. More recently however, Johnson et al. have shown that the same principles (and parameters) of dynamic neural field models can be applied across different tasks, not only in the motor domain (Johnson et al., 2008). They provided simulation results of models for spatial visual working memory, as well as for color and shape representations, which reproduced results from psychophysical studies. Johnson et al. use these results to argue that the dynamic neural field may be a universal mechanism across all levels of cognition, as a kind of building block for cognitive architecture.

The dynamic field theory sees cognition as intrinsically integrated and as the result of a complex, dynamic system in a close coupling with the environment. It is thus to be contrasted with the approach to model and test sub-systems of cognition separately before integrating them into a whole system. However, by building dynamic neural field models of sub-systems that allow to make measurable predictions, it allows for "rigorous hypothesis testing" (Johnson et al., 2008). On the one hand, this is a strength of the dynamic field theory, but on the other hand it still is also its weakness: In its current level of specification, the theory only covers small patches of cognition and leaves many issues unaddressed. The models are hand-crafted to prove certain points, inputs are specified beforehand and learning is not addressed[1].

2.5.2 Computational Models

Cognitive Architectures Based on the Dynamic Field Theory

Appart from the computational models of the dynamic field theory that have already been mentioned above, the methodological framework of the theory has also been applied to several robot studies, where it has in particular been used to implement planing, decision making, information integration and working memory capabilities (Bicho et al., 2010; Erlhagen et al., 2006; Sandamirskaya and Schöner, 2010). All of these implementations use a set of interconnected dynamic neural fields to implement the main functionality of the model. The fields interface to sensory and motor modules, which provide input to the fields from the sensors and readout parameters from the

[1]This statement only applies to computational models that are directly linked to the dynamic field theory. Learning between (non-dynamic) neural fields in general (for example in the form of Hebbian learning between self-organizing maps) has of course been studied in the literature (e.g. Li et al., 2004; Morse et al., 2010; Westerman and Miranda, 2002), however it has not been applied to cognitive architectures and computational models based on the dynamic field theory. Notably, Gläser et al. demonstrated that also the lateral (inhibitory and excitatory) connections inside a dynamic neural field can be learned in a data-driven way (Gläser et al., 2009; Gläser, 2012).

fields for the execution of actions. The behavior of the robot is therefore governed by the intrinsic activation dynamics of the coupled dynamic neural fields.

For example, Erlhagen et al. utilize three interconnected dynamic neural fields in their model for joint action tasks (Erlhagen et al., 2006). Two simulated mobile robots are situated in an arena with two objects and two goal locations, and the task of the robots is to bring both objects to their respective goal locations. To be effective, the robots have to coordinate their actions, such that both objects are handled simultaneously. The three fields represent the current locations of the two objects, a short-term anticipation of the position of the other robot, and a movement target for the position of the robot itself, respectively. Inputs are provided to the fields in the form of peaks of activation, thus for example the field representing the locations of the objects has as input an activation landscape with two peaks (similar to the field shown in Figure 2.10). The activation dynamics of the field representing the movement target of the robot is tuned to implement a winner-take-all mechanism to select among multiple candidate targets. It receives excitatory input from the field representing the locations of the two objects, and inhibitory input from the anticipated location of the other robot. Thus, when the other robot approaches one object, the input to the field of movement targets at the location of this object becomes weaker due to the inhibitory input, which leads the activation dynamics of the field to select the other location.

Similarly, to implement collaborative behavior for human-robot interaction in a joint assembly task, Bicho et al. utilize a model with several interconnected dynamic neural fields to select among multiple primitive actions of the robot, such as pointing to a part or picking up a part and handing it over to the human collaboration partner (Bicho et al., 2010). The primitive actions are represented in a one-dimensional dynamic neural field along an abstract behavioral dimension. Input is provided to the field at eleven equidistant locations, representing the robot's eleven available action primitives. Each action primitive receives input from different sources, for example when the robot parses a spoken command from the human interaction partner or when it sees him reach for a certain part. In this way, rule-like functionality is implemented in the model via the connections linking the different fields together (for example, if the human reaches for a bolt, the robot's primitive action to give him a nut receives input). Bicho et al. argue that the direct link between sensory inputs (such as observed actions of the collaboration partner) and action execution, combined with the dynamic decision making in the neural fields, avoids the need to couple a high-level deliberative layer with a low-level reactive layer (Bicho et al., 2010), as in hybrid control architectures (see Section 2.2.2).

Sandamirskaya et al. propose an alternative implementation for rule-like functionality in a dynamic neural field model. They utilize an additional dynamic structure, which they call an "elementary behavior", consisting of three reciprocally coupled neurons (Sandamirskaya et al., 2011). The activation of the neurons is governed by a dynamics similar to the Amari dynamics used in dynamic neural fields. Each elementary behavior is tied to a primitive action of the robot, such as "close gripper". The neurons of the elementary behaviors explicitly represent a rule-like structure, with one

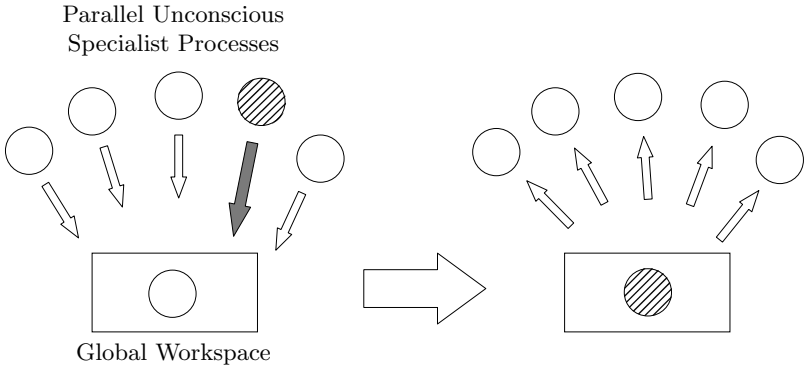


Figure 2.13: Organization principle of brain processes as proposed by the global workspace theory. Many specialist processes operate in parallel and compete for writing access to the global workspace. Winning specialist processes fill the global workspace with their reply, which is then broadcast to all specialist processes. Drawing reproduced from (Shanahan, 2006).

of the neurons representing a precondition for the triggering of the linked primitive action, and another one representing the intended outcome of the action (e.g. "gripper closed"). Thus, by providing their system with elementary behaviors for each primitive action and coupling the respective neurons coding for preconditions and intended outcomes, Sandamirskaya et al. endow their model with a kind of serial order for action execution (Sandamirskaya et al., 2011).

The Global Workspace Architecture

Shanahan proposes a cognitive architecture and its implementation in a dynamical system (Shanahan, 2006; Shanahan and Connor, 2008; Shanahan, 2008, 2012) based on the "global workspace" theory of consciousness (Baars, 2002; Shanahan and Baars, 2005). It proposes that conscious experience is the result of the process of communication between a wide range of sub-conscious brain processes. The theory suggests that information exchange between specialist processes is realized in the brain through a dedicated memory, the global workspace, which can only hold one consistent entry at a time. The specialist processes compete for privileged access to this limited capacity storage. All processes are allowed to read from the global workspace, but only a winning coalition of processes is allowed to fill the memory with new content. According to the theory, the serial nature of concious thought arises from the massively parallel operation of the brain through this system of information exchange: While many processes are at operation in parallel, the global workspace can only hold a single thread of information. Figure 2.13 is a schematic drawing of the organization principle behind the global workspace architecture.

While similar concepts of a shared memory for the communication between processes have already been proposed in the field of artificial intelligence, the global workspace theory assumes a dynamic, sub-symbolic mechanism behind the memory.

Shanahan argues that it is based on cortical circuits with recurrent connections, which implement a dynamical system that defines an attractor landscape, and perceptual categories become its attractors (Shanahan, 2006). When the global workspace has reached a stable state in one of the attractors, the specialist processes read the information and evaluate if it is relevant to them, in which case they try to gain access to the global workspace to broadcast their reply by driving it into a new stable state.

In a proof-of-concept simulation experiment, Shanahan demonstrates the functioning of the architecture in a scenario where a robot has to decide for one of three possible actions based on the current sensory input (Shanahan, 2006). A mobile robot faces three differently colored objects, with the one in the middle being the only in its field of view. The robot can either turn left to face the object on the left, turn right to face the object on the right, or move forward to approach the object that it is currently facing. Only one of the actions will produce a reward. Shanahan demonstrates that the system, after having learned a forward model for each action, is able to simulate the action outcomes via the global workspace and to select the rewarding action. In a followup study, Shanahan demonstrated that a global workspace can be implemented in a biologically plausible spiking neural network model (Shanahan, 2008). The model is a simulation of the firing pattern of several cortical clusters, some of which represent an abstraction of a forward model, and others corresponding to the global workspace. The clusters belonging to the global workspace exhibit a dynamics of activation that synchronizes their activation patterns. Each forward model sends its output to one of the clusters of the global workspace, which can trigger an instability and drive the whole set of connected global workspace clusters into a new stable state, which corresponds to the output of the forward model.

The global workspace architecture envisions two kinds of modules to be central to the organization of the cognitive architecture. On the one hand, the main functionality of the system is based on a set of forward models of primitive actions, which can predict the next stable state of sensory input that would result from executing the associated primitive action. On the other hand, a distributed set of clusters of neurons implements the global workspace, which acts as a shared memory and as the basis of communication between the forward models and other processes. The architecture is thus composed of two sensorimotor loops: One, which is closed externally through the effectors and sensors of the robot and its interaction with the environment, and one which is closed internally through which the system can simulate its actions and evaluate their outcome prior to executing them (Shanahan, 2006). The selection of an action is partly due to the importance that the individual forward models themselves report for their selection, and partly due to a value-based selection mechanism. The forward model that currently has the highest importance gains access to the global workspace and simulates the outcome of its execution. The predicted sensory input that is the result of this simulation is then evaluated by a value system, which either promotes the importance of the action, in which case it becomes subject to actual execution by the robot's effectors, or demotes the importance of the action, in which case another forward model will have the highest importance and is allowed to broadcast its simulation of predicted sensory input.

2.5.3 Implications

Dynamicism sees cognition as the emergent result of the interaction of several complex dynamical systems, including the agent's (artificial) central nervous system, its body and the environment. In its strongest form, dynamicism completely rejects the view that cognition is based on an internal representation of some external state of affairs, be it in the form of symbols, or be it in the form of distributed activation patterns (Thelen and Smith, 1996; van Gelder, 1998). Instead, it is argued that the internal dynamics is influenced by environmental forces, for example in the form of sensory input. Thus, studies of dynamicist computational models primarily focus on how the behavior of situated dynamical systems unfolds over time through the dynamic interaction between the systems and their environment.

The presented cognitive architectures demonstrate how various cognitive functions can be explained by the intrinsic dynamics of a system of multiple interconnected dynamical systems, without relying on any form of high-level processing. Both the global workspace model as well as the models based on the dynamic field theory show the capability to decide for one of several possible courses of action. In the case of the global workspace architecture, this is the selection of one out of several action primitives, which are represented by forward models as distinct units of knowledge with no lateral interaction. In the case of the dynamic field theory, the selection can either be that of a specific value for a parameter of an action, such as the target of a movement, or the selection of one of several primitive actions, which are arranged along an abstract dimension. In either case, knowledge is stored in the dynamic field theory architectures by the way that input is directed to the fields, for example by mapping possible movement targets onto the action parametrization for approaching them, or by associating discrete events with responses. In contrast to connectionist models, existing cognitive architectures based on the dynamicist paradigm do not account for the learning of the knowledge, but in the experiments it is assumed that the system has already acquired it somehow[1]. In effect, knowledge is provided by the designer to the system in these cases, in a more or less explicit form. This is either the mapping of a sensory input onto a motor space, or a dynamical systems version of discrete rules: If some precondition is satisfied, then reinforce an attractor point in the system's state space that represents the appropriate motor response (e.g. if facing a red object, turn left; if the human interaction partner reaches for the bolt, hand over the nut; etc.)

This implementation of distinct rule-like structures can be seen as a limiting factor of current cognitive architectures in the dynamicist paradigm. While the dynamicist theories advance the provocative view that cognition is the emergent result of how the internal state of a system is dynamically and flexibly influenced, bent and changed under environmental forces, the current computational models cannot be said to provide an adequate account to reflect this view. They boil down to a dynamical systems implementation of one particular decision-making heuristic for choosing one solution among a set of discrete candidates. The overall flexibility of the system is then again

[1] Which of course does not mean that learning in a dynamic neural field representation is not possible (see e.g. Gläser, 2012, ch. 3).

determined by how flexible the primitive action controllers and event classifiers have been implemented by the designer. Those models that do not make use of discrete action primitives, such as the model used by Erlhagen et al. for the collaborative robot scenario, implement simple mappings from inputs onto the state space of the dynamical system (in their case mapping positions of objects and of the robot collaboration partner as attractors states onto a space of movement targets, both represented in Cartesian space), producing similar results as behavior-based robotics.

What makes the dynamicist theories attractive is their ability to elegantly explain subtle effects of timing and continuity in behavior, but this characteristic is insufficiently presented in current computational cognitive architectures.

2.6 Discussion

This chapter gave an overview of current cognitive architectures in robotics. As the definition of each architecture is based on specific assumptions about the nature of cognition, the description was embedded into an introduction to the five currently predominant paradigms in the field of computational models of cognition: Cognitivist and hybrid architectures on the one hand, and emergentist architectures (behavior-based robotics, connectionism, and dynamicism) on the other.

Architectures based on the cognitivist paradigm, and hybrid architectures along with them, assume that cognition is a form of computation operating on a symbolic representation of the world. High-level cognitive function is seen as a process separate from low-level sensorimotor functions. This allows to treat cognition as an abstract problem and to employ algorithmic solutions, which can produce impressive results as long as the environment and the problem are sufficiently well modeled by the designer. However, as already noted earlier, this approach leads to difficult problems regarding flexibility, complexity and the question of how to determine what is relevant in a given context, which limits the applicability of current cognitivist systems either to very specific tasks, or to rather sterile laboratory environments (Christensen, 2004; Clark, 2002; MacDorman, 1999; Stoytchev, 2006; Weng et al., 2001).

In contrast, emergentist approaches pursue a bottom-up approach to modeling cognition by proceeding from low-level sensorimotor processes. Behavior-based robotics directly addresses the problem of cognitivist systems in coping with dynamically changing real-world situations by tightly coupling motor responses to sensory cues. The problem of relevance is solved in this way, as each behavior of the robot is provided with a dedicated detection device for low-level sensory input that determines whether the behavior is relevant or not. However, the design of behavior-based robots has proven to be difficult, which effectively limits the approach to robots with insect-like behavior. Connectionism on the other hand emphasizes the importance of learning in a cognitive system. Here, knowledge is acquired by the system by finding statistical regularities in training examples, rather than being explicitly provided by the designer. While this can in principle be regarded as a necessary and powerful system property, also current connectionist architectures seem to be limited to small-scale scenarios, as

they too are affected by the problem of relevance detection (see Section 2.4.2). Finally, dynamicism argues that the dynamic interaction between the system, its body and the environment is key to cognition, rather than mapping external reality onto internal representations. Dynamicist architectures demonstrate how cognitive tasks can be solved by coupled dynamical systems, which renders them equally adept at handling dynamic environments as the behavior-based architectures while promising to be better transferable to more complex task settings. However, current dynamicist architectures rely on hand designed, rule-like knowledge, which also limits their applicability to small-scale scenarios.

A fundamental problem in the endeavor to model cognition is to find the right level of abstraction for the model. Among the current approaches, connectionism assumes the lowest level of abstraction as its models are based on what is known about the electrophysiology and neurochemistry of the brain. Still, this is by no means the lowest possible level of abstraction, as the standard mathematical model of a neuron is already a very strong abstraction of the biological neuron, which is indeed far more complex. Dynamicism is based on the study of low-level processes as well, studying the cognitive system as a composition of coupled dynamical systems, but abstracts to a large extent the details of neural implementation away (even though most dynamicist models still use dynamic neural network models). Further up the scale of abstraction comes behavior-based robotics, which does not necessarily aim at building a model of the biological brain, but is rather an engineering approach that addresses concrete difficulties in the modeling of intelligent robots. Finally, at the highest level of abstraction, cognitivism studies cognition entirely as an abstract problem and supposes the brain to be an arbitrary implementation of an abstract problem solving machine.

If there was a way to objectively measure and correlate task complexity, generality and level of abstraction for cognitive architectures, we should expect to find two trends for current cognitive architectures: A negative correlation between feasible task complexity and generality, as well as a negative correlation between level of abstraction and generality (see Figure 2.14). Cognitivist architectures, on the upper extreme of the level of abstraction, can benefit from hand-crafted task knowledge provided by the human designer. This gives them an advantage over the other paradigms when it comes to task complexity, so that they currently are applicable to the most complex tasks as compared to the other approaches. However, the resulting systems tend to be very specialized to the scenario that they were prepared for and demonstrate poor task generality. On the lower extreme, connectionist systems commit to generating all knowledge entirely by learning from observations. Although this approach promises to be extremely task general, powerful enough learning methods first need to be found. Current systems only perform well if the training examples that are provided for learning come from a limited domain, while learning in truly natural settings is not yet feasible. While most neural network models use homogeneous network layouts apart from defining an input layer, an output layer and a number of hidden layers, connectionist approaches to cognitive architecture seem to agree that an additional level of structure is necessary to address the complexity of the task (although it has also been argued that hierarchical

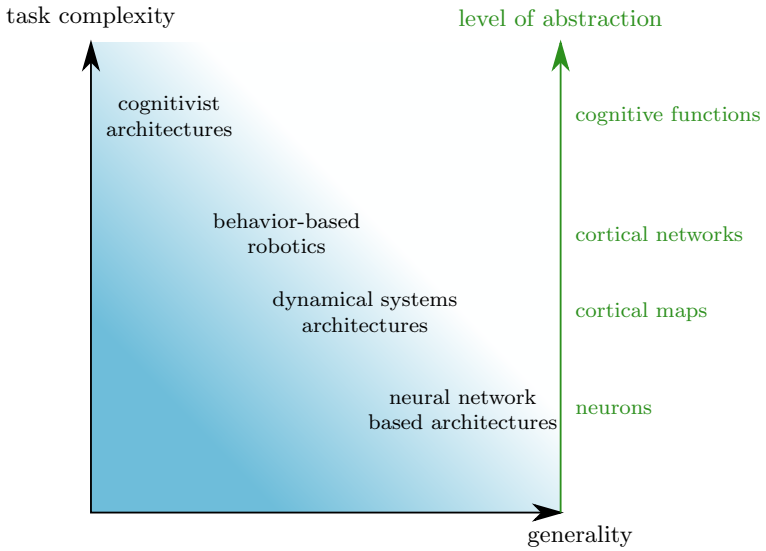


Figure 2.14: Visualization of a hypothetical space measuring task complexity, generality and level of abstraction for cognitive architectures. Current cognitive architectures are restricted to the blue region, meaning that the more general an approach is, the less complex the feasible task complexity is. Note that this is not claimed to be a theoretical necessity, but is merely an observation of the current state of cognitive architectures across the different paradigms.

structure can emerge implicitly in layered neural networks, Botvinick and Plaut, 2004; Yamashita and Tani, 2008). The definition of this additional level of structure can either be a complete layout of the whole system, as in the case of brain-based devices, or in the form of a generic building block, as in the case of ERA. Either case adds to the level of abstraction as composed to a general neural network architecture. The former gives a higher immediate gain in tractable task complexity whereas the latter should in principle provide a more general system. In between cognitivism and connectionism in both trends would lie behavior-based robotics and dynamicism. While dynamicist theory describes a less structured and more general cognitive system, computational models in the paradigm use hand-defined knowledge, which renders them more structured and task-specific (see Section 2.5.3).

A successful cognitive architecture should both be general and cope well in complex tasks. Thus, independently of the paradigm chosen as motivation for research in cognitive architecture, making progress would equate to improving either (or both) task complexity or generality as compared to existing approaches, without regressing in the respective other. With the goal to make such progress and to address some of the difficulties of current cognitive architectures, in the next chapter it will be proposed to use recent developments in Psychology as inspiration, where some accounts regard cognition as inherently relying on sensorimotor processes in the brain and thus subscribe to the emergentist view, but at the same time propose a particular mechanism for the representation of knowledge in a more structured way than connectionism and

dynamicism assume. This mechanism is based on an active element of the cognitive system, which is claimed to be in line with recent findings in neuroscience. Thus, this view assumes an intermediate level of abstraction while remaining flexible enough to support a more general applicability. Later on in this work, it will be demonstrated that a system can acquire these active elements through its own sensorimotor experience, and a possible way to address the problem of relevance in the learning process will be proposed.

3

A New Cognitive Architecture Based on Embodied Simulation

The last chapter gave an introduction to the different currently predominant paradigms to studying cognition in general, and cognitive architecture in robotics in particular. On the one hand, cognitivism is based on the use of explicit symbolic representations and posits a clear-cut separation between low-level sensorimotor processes and high-level cognition. On the other hand, emergentist views postulate that it can be misleading to study the problem of cognition on an abstract level, i.e. separately from its implementation and its embodiment. Instead, the study of low-level processes in natural systems should be the starting point. Overt behavior is seen as the result of covert dynamical processes. By exploring what is possible in models of these low-level processes and seeing limitations as useful scaffolding constraints for the modeling, cognitive functions are explained as the emergent result of the cooperation of simpler functional elements.

The different emergentist approaches commit to certain methodologies for the modeling of system components: Behaviors, neural networks or dynamical systems. Independent of this choice, it remains an open research question, how to design an emergentist system on the architectural level (i.e. the connectivity between subsystems). There are two different approaches: Either the subsystem inter-connectivity is specified by design, which however limits what the system can learn as it cannot discover contingencies that are not covered by its fixed connections. Or a generic building block is introduced, which can be regarded as the more general solution, as it opens the system up to open-ended learning at least in principle. From the architectures that were described in the last chapter, behavior-based robotics, brain-based devices and the dynamic field theory models can be counted to the former class, while ERA and the global workspace architecture can be counted to the latter class. However, as already mentioned in the discussion of these architectures, there are several limitations to the way they implement building blocks: On the one hand, mechanisms in ERA are intentionally underspecified, as it is meant to be a guideline for modeling instead of a proper cognitive architecture on its own. This has the consequence that it cannot be directly transferred to many scenarios, as for example problems related to motor con-

trol remain unaddressed, and scalability is most likely an issue. On the other hand, the global workspace architecture relies on distinct forward models as its building blocks, with no lateral interaction at all, and all communication between building blocks happens via the global workspace as a shared memory. This has the consequence that the system implements a very strict winner-take-all competition, where only a single building block is active at a time, for example to produce a motor command in response to a sensory cue (although Shanahan mentions that a "winning coalition of processes" (Shanahan, 2006, p. 438) could broadcast information to the global workspace instead of only single winning models, however this feature is neither present in the implementation of the architecture, nor explained in more detail). This is disadvantageous, as it limits the flexibility of the system to that of its individual motor primitives, a general issue with regards to the integration of building blocks in a cognitive architecture, which will be discussed in detail in Chapter 4.

In this chapter, the concept of "embodied simulation" will be introduced, with the goal to establish that it is a suitable mechanism for a cognitive architecture, which supports the modeling of a generic building block that can address some of the above mentioned difficulties and grounds the model in a comprehensive body of theoretical, as well as empirical work on cognition. The concept of embodied simulation has recently been proposed in the psychology and neuroscience literature (see Barsalou, 2008; Gallese, 2005), and has been used in explaining various cognitive functions, such as memory, representation, categorization, integration of multimodal information, decision making, concept formation, or language, in an emergentist framework. It is part of arguments in favor of an embodied view of cognition, which represents the idea that the body is not simply the acting organ of the cognitive system, but that cognition is directly influenced in many fundamental and important ways by the body and its situatedness in the environment (Barsalou, 1999; Beer, 2000; Clark, 1997; Pfeifer and Scheier, 2001; Varela et al., 1991). Theories of embodied cognition seek to explain all of cognition as closely tied to bodily states, and therefore strongly reject the cognitivist view according to which sensorimotor processes and cognitive functions are separated in the brain (cf. Section 2.2). There is agreement among these theories that descriptions of cognitive functions as the manipulation of symbolic representations can at most be understood as idealized views on actual brain processes. Instead it is highlighted that many cognitive functions do not necessitate any form of abstract processing, but that instead they can be explained on the basis of simpler sensorimotor processes. Therefore, theories of embodied cognition either postulate that higher cognitive functions inherently rely on making use of sensorimotor representations in the brain, or that there is no such thing as higher cognitive function and all of cognition emerges from interacting low-level processes.

The behavior-based robotics paradigm has played a pivotal role in the development of the embodied cognition thesis, as here it was demonstrated that physically capable robots could be constructed that did not rely on an abstract representation of the world or were reasoning about their situation on an abstract level. Also, as the dynamicist paradigm assumes that the environment perturbs the dynamical cognitive system and

thus helps shape the process of cognition, dynamicism too can be said to be an embodied view of cognition (and is therefore sometimes also referred to as "embodied dynamicism", cf. Thompson, 2007).

In pursuing the goal of this thesis to identify basic elements from which a cognitive architecture for a robot might be built, this chapter will focus on the question: What are the building blocks of cognition, and what are their nature? Important concepts that will come up are "schemata" and "simulators", for which it will be tried to develop a clear enough understanding to support their modeling in a computational system. Therefore, in the following, first some of the prime empirical evidences in favor of an embodied view of cognition will be introduced in Section 3.1, followed by a description of the "convergence-divergence model" in Section 3.1.1, which is often referred to in the context of theories of embodied cognition as a neuroscientifically plausible model of cortical organization to support embodied simulation. In Section 3.1.2, representative accounts from the psychology and neuroscience literature for conceptual processing on the basis of embodied simulation will be presented. Many of the ideas that are used in these theories are borrowed from a concept that has a longstanding history in psychology, which is the concept of schema. This is especially worth mentioning as several existing computational models in the fields of artificial intelligence and robotics are based on that concept. Therefore, in Section 3.1.3 a short introduction to the historical development of the concept of schema will be given and a comparisons to the theories of embodied cognition will be drawn. Section 3.1.4 summarizes the discussed theories and provides a compact description of the core properties of what constitutes a basic element of the cognitive system. Finally, Section 3.3 proposes to combine theoretical views of embodied simulation and of the dynamic field theory in a way that the two can be mutually supportive, and describes a new cognitive architecture based on a building block that incorporates these ideas. Section 3.3.3 compiles the theoretical considerations into a list of concrete system properties, which directly guides the computational modeling described in the following chapters.

3.1 Theoretical Background on Embodied Cognition

Most of the arguments put forward in favor of an embodied view of cognition are based on the discovery that sensorimotor areas in the brain are also activated in conceptual tasks that do not demand sensory processing or a motor act. This is against any prediction one would make when assuming a strict correspondence between cortical regions and cognitive functions. Why would a mathematical task activate a brain region that is responsible for controlling hand movement? However, exactly this kind of response pattern is found in the brain (Zago et al., 2001), along with a multitude of other effects where sensorimotor regions are active in conceptual tasks. One of the most striking findings in this direction is that language processing, long having been seen as the flagship cognitive ability, with its syntax and symbolic rules inherently conceptual, also activates sensorimotor regions. For example, Pulvermüller and colleagues demonstrated in brain imaging studies that in the processing of action words, such as "lick", "pick"

and "kick", activation can be found not only in regions in the auditory cortex and brain regions that have long been associated with language processing, such as Broca's area, but also in regions in the motor cortex that are otherwise used in motor performance of the mouth, hands and legs, respectively (Pulvermüller, 2005).

Other studies have shown that viewing and naming pictures of man-made tools involved activation in the premotor cortex (Chao and Martin, 2000). These findings point to the interpretation that the brain uses a very distributed neural representation to process entities or events, both when actually perceiving or acting, as well as when conceptualizing in their absence (Gallese and Lakoff, 2005). Furthermore, this distributed representation seems not to rely on cortical regions that are designated to represent and manipulate in an abstract format, but it seems to inherently rely on sensorimotor regions of the cortex, namely those regions that are involved in sensorimotor experience of that entity or event (Barsalou, 1999). This view is further supported by studies on mental imagery, where it was found that when subjects imagined performing an action, it took them a comparable amount of time as when they actually performed the action physically (Decety et al., 1989), and in brain imaging studies it could further be confirmed that parts of the motor cortex were both active during mentally imagining the execution of an action as when physically performing it (Decety et al., 1994).

Based on these empirical evidences, theories of embodied cognition formulate accounts for cognitive functions as products of sensorimotor processes, rather than of abstract manipulations of symbolic representations. Therefore they attack cognitivism with Occam's Razor (Gallese and Lakoff, 2005): Positing the existence of abstract symbolic representations and mechanisms that manipulate them is not required to explain cognitive functions. Instead, accounts can be formulated that only involve sensorimotor processes. Thus, since the sensorimotor processes are necessary in any cognitive system (also symbol manipulating cognitive agents would need sensorimotor processes for their motor control), postulating mechanisms that manipulate abstract symbolic representations for cognitive functions that the sensorimotor processes already provide *without* symbolic representations would be redundant. Therefore, while it might be difficult up to impossible to completely refute theories that see cognition as the manipulation of abstract symbolic representations, embodied views on cognition offer simpler and more elegant explanations (Beer, 2003).

3.1.1 The Convergence-Divergence Model

Damasio's *convergence-divergence model* (Damasio, 1989; Meyer and Damasio, 2009) is one of the earliest contemporary theories to propose that sensorimotor processes in the brain do the work of what would otherwise have been accounted to a module of cognition. To establish how processes like memory and high-level perception can be supported by the physiological structure provided by the cortex, Damasio emphasizes that connections between regions of the cortex follow a direction, and that there are not only connections going from "bottom" to "top", i.e. from the primary sensory and motor cortices to association areas, but just as importantly also the other way around. He looked at evidences from several cases of disorders caused by localized damage to

association areas in the cortex and symptoms that they caused. For example, a patient was well able to describe the contents of a photograph, for example one taken at a birthday party, in very much detail and indistinguishable from the performance that one would expect from a healthy person. What the subject was not able to do however was to recognize that the picture was actually one taken at his *own* birthday party. Damasio concluded from this that what had been damaged in the patient was not the ability to perceive, act and reason in the world on a *generic* level, but it was the recall of *specific* entities or events that was hindered (Damasio, 1989).

This performance of the patients is difficult to explain with cognitivism's information processing metaphor of the brain's organization, according to which the integration of sensory information happens higher "upstream" in the cortical hierarchy, since these areas were damaged. Damasio proposed a different view, positing that the rich perceptual experience of entities and events that we have in our everyday life does not require that information has passed the complete sensory processing machinery of the brain. He argues that the recording of activity in integrative cortical regions cannot be said to correspond to the conscious experience of perceiving some entity or event (Damasio, 1989). The subjective experience of an integrated sensory and motor perception, for example when touching, seeing and smelling an apple, does not rely on the transformation of sensorimotor information into an "integrated" neural representation in an association area. Instead, he proposes that a distributed activation at different locations in the sensory and motor areas of the cortex gives rise to the experience. What integrates the different modalities is the *simultaneity* of the activation, in a "time-locked co-activation of geographically separate sites of neural activity within sensory and motor cortices" (Damasio, 1989, p. 39). Because the visual perception of the apple occurs at the same time as the haptic perception, we perceive the two as belonging together to the event of us touching an apple.

Nonetheless, the association areas that are found across the cortex do extract information about correlations in the firing of neurons in "earlier" regions (meaning topographically closer to the primary sensory and motor cortices) that they are connected to. However, according to Damasio's model, this does not serve the purpose of creating a multi-modal, integrated representation of entities and events in the world in the form of abstract symbols, but instead their function is to support *recall* of specific instances. For example, an association area in the visual cortex could learn about the usual color of an object that has the shape of an apple. However, information does not flow uni-directionally towards integrative cortical areas to integrate sensory information, but the reverse direction is just as essential. It is used during recall to reactivate, or *reconstruct*, the activation in the early sensory and motor areas. The activation in association areas therefore does not *represent* in place of the sensorimotor areas. Activation of a group of neurons or even single cells in higher cortical areas does not induce a memory or the experience of perceiving something by themselves, it has to be accompanied by the activation of sensory and motor areas. In other words, there is no such thing as the infamous "grandmother cell", that all by itself represents all the memory of the own grandmother, but a whole network of interconnected regions needs

to be in place to store the memories. Also, much of what we know about specific entities or events (our grandmother, or our own birthday party, etc.) relies on our generic knowledge that is encoded in regions closer to the sensorimotor regions (what elderly women look like, etc.), as demonstrated in the case of the subject who could still reason on a generic level about the photograph of its own birthday party, despite the damage to an association area. More distal regions are responsible for bringing together specific activation patterns across the sensorimotor regions, to represent memories of specific entities and events, but their activation always needs to be accompanied by activations in the sensorimotor regions. In fact, their purpose is to reactivate sensorimotor regions in a coherent way.

This view establishes a model of cortical organization that reaffirms the hierarchical structure via association areas, but stresses the importance of *top-down* connections. It further proposes a different function for association areas than in the information processing metaphor. Damasio posits structures in association cortices, which he calls "convergence-divergence zones (CDZ)", that store records about how to combinatorially arrange knowledge fragments in earlier cortices so as to represent an object or an event comprehensively (Damasio, 1989; Meyer and Damasio, 2009). The CDZs exist across all levels of cortical organization, first inside the sensory and motor cortices as modality-specific CDZs, and on top of these, higher association areas house cross-modal CDZs that allow the integration of more and more multi-modal information. See Figure 3.1 for a schematic drawing of the model.

3.1.2 Embodied Concepts and Embodied Simulation

The implemented models in the behavior-based robotics domain and of the dynamic neural field theory might seem to suggest that in an emergentist's view, there is no need for concepts: Behavior can be generated using close couplings between sensors and actuators, information can be integrated and decisions be made without the need to extract abstract representations from the sensory input. However, Damasio's convergence-divergence model does hint at a way in which also concepts can be understood more "embodied", i.e. closely related to bodily states: Damasio proposed, that entities and events are "re-presented" during memory recall by reactivating the same sensorimotor regions that were active during actual perceptual stimulation. This hypothesized mechanism, the reactivation of patterns in sensorimotor regions from higher-level regions, has been borrowed by several theories of embodied cognition as a central feature to implement concept-like function. It is commonly referred to as "embodied simulation" (or equivalently, "mental simulation", "internal simulation" or just "simulation"), in the sense that the input to sensorimotor regions of the cortex that produces activation patterns during actual experience is *simulated* by top-down activation from higher-level cortical regions to create a similar activation pattern in the absence of actual input.

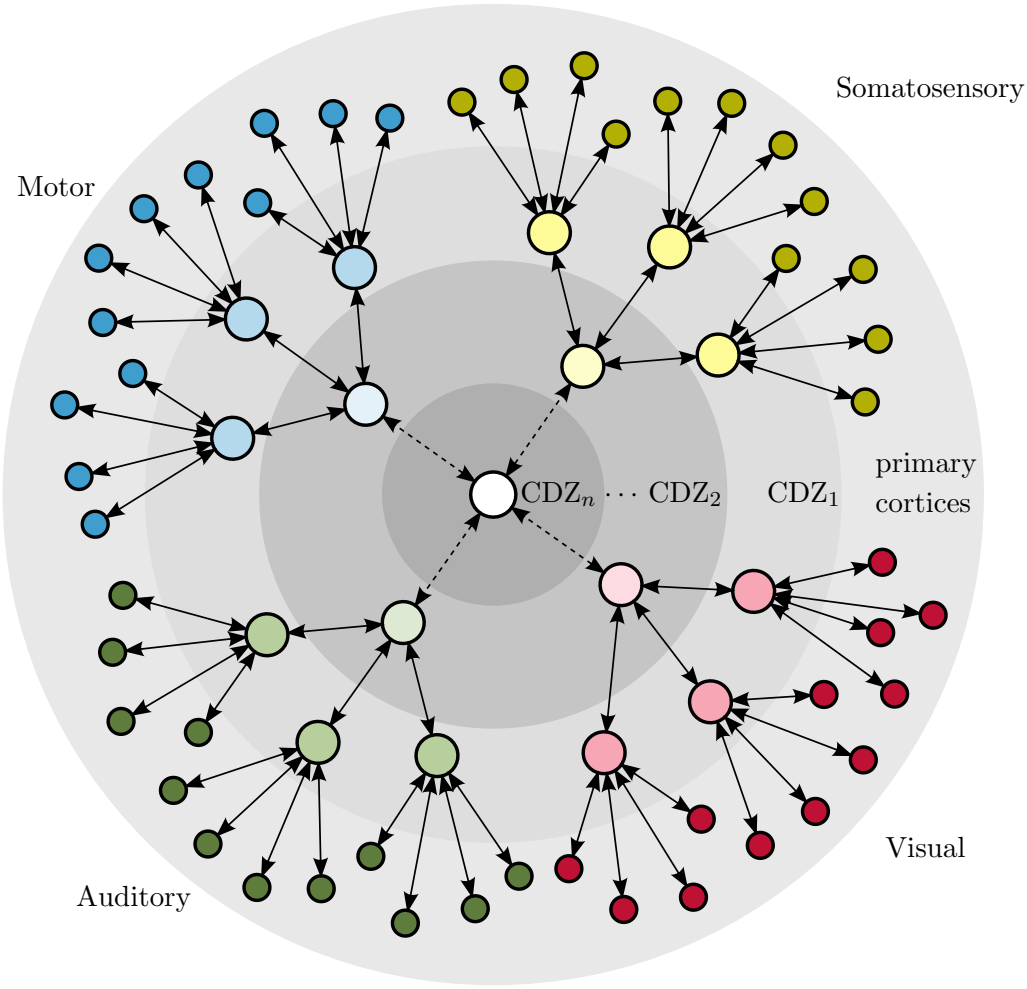


Figure 3.1: Schematic drawing of cortical organization according to the convergence-divergence model. Information is directed from and to the senses and muscles via the primary cortices, here depicted on the outermost hierarchical level. From there it is first directed to modality-specific CDZs (here CDZ_1 and CDZ_2), and from there onwards to cross-modal CDZs (here depicted generically as CDZ_n). Information flows both up the hierarchy, for example during perception and learning, as well as down the hierarchy towards the primary cortices, for example during recall. Drawing based on (Meyer and Damasio, 2009).

Perceptual Symbol Systems

An exemplary theory of cognition that utilizes such a mechanism is Barsalou's theory of *Perceptual Symbol Systems* (Barsalou, 1999). He argues that conceptual processing, the work that traditionally has been ascribed to an "amodal" symbol system, i.e. mechanisms based on abstract symbolic representations, is actually done by "perceptual symbols" that reside in the sensorimotor regions. Perceptual symbols are temporary, embodied representations in the form of activation patterns in the sensorimotor regions, which are controlled by a complex network of neural structures, which he calls "simulators". In Barsalou's theory, simulators correspond (i.e., *do the work of*) concepts: There is a *car*-simulator to represent and process cars, a *cup*-simulator for cups, and so on. Barsalou uses Damasio's framework of the convergence-divergence model as a vehicle for his theory, arguing that simulators are cortical networks in the association areas, i.e. in the convergence-divergence zones.

Simulators are formed though experience. For example, when one sees a car, the visual input activates a set of neurons in the visual cortex, which account for the actual sensation of seeing the car. The assumption is that the perceptual similarity of different instances of cars is reflected in the similarity of these activation patterns in the sensory regions during perception. Thus, after having encountered many instances of cars, each of which activated a similar set of neurons, the brain establishes in an association area a structure, the simulator for the category *car*, that has remembered which neurons are typically active during the perception of a car. The simulator can then be used to reactivate those neurons in the absence of sensorial input, to simulate the perception. The simulations are used as perceptual symbols in conceptual processing, for representation, categorization and categorical inference.

To represent an entity or event in its absence, for example when one imagines seeing a car, the simulator produces a pattern of activation in the sensorimotor regions, matching a pattern of activation that an actually perceived car could have produced. The simulator however does not only produce a single simulation, but can produce a variety of simulations, accounting for the variety of perceptually different actual cars (big and small cars, red and green cars, sports cars and utility vans, and so on).

Categorization, i.e. treating entities and events as belonging to one kind or another, is a key feature of the conceptual system. In a perceptual symbol system, categorization is done by the simulators: To establish if an object in the environment is a member of a category, the simulator representing that category produces a simulation of the perception, which is compared to the actual input. If the simulated and the actual input are sufficiently similar, then the object is classified as belonging to that category ("the thing I see must be a car, because it looks much like what I could imagine a car would look like"). Thus, the activation in the sensory areas is not matched against a symbolic "type" represented somewhere far down a processing hierarchy by a special categorization module, but categorization happens directly inside the sensorial area by using simulation.

Barsalou proposes that simulators are organized around a *frame*-structure. Frames, a concept proposed in A.I. by Minsky, are structured representations of knowledge

(Minsky, 1974). They define a concept as a set of "attributes" or "slots" (for example, a *car*-frame has a slot for a door), which can be assigned values or "fillers" (the particular door of that car) and constraints on the values (a car door must not be wooden). Furthermore, frames are recursive, as slots can take other frames as fillers (the door itself being a frame with a slot for a window, etc.). The original description of Minsky assumed frames to be abstract and symbolic, thus *disembodied* by definition. However, Barsalou argues that embodied, frame-like processing can be implemented by simulators and perceptual symbols. He sees frames as structures that are learned from experience, and the slots formed by selective attention. Without being specific about the details, he proposes that frames hold the information about value constraints, so that the values for the individual slots are mutually constraining, and competing against, each other. For example, when producing a specific simulation of a car, the overall shape of the car that is simulated might constrain the specific simulation of the tires (a sports car does not have the wheels of a tractor).

As another important property of Barsalou's interpretation of frames, he argues that they represent spatial and content information separately, motivated by the two-streams hypothesis (see Section 2.1). Thus, frames represent on the one hand volumetric regions according to their spatial layout (for example, where in the object-centered reference frame of a car the windshield is located), and on the other hand contents of these sub-regions are represented as specializations. Through experience, perceptual symbol systems learn spatial layouts not only of objects, but also in generic ways. For example, after many encounters of objects being above other objects, a generic frame with a spatial layout of two locations, one being above the other, is learned, to represent the *above*-concept. It can then be used productively in the generation of infinitely many simulations of objects that are in an *above* relation to each other. And by further including the recursion property of frames, complex simulations can be generated, for example of one object being above another object being to the left of another.

Embodied Simulation and Motor Control

As simulation is the process of activating the same neural regions that are active during actual experience, in action this means activating on the one hand neurons in the motor regions of cortex, which code for the movement, and on the other hand neurons in sensory regions, which code for the visual, auditory and somatosensory perception that is tied to the execution of the movement. This process is closely related to *internal models*, a concept that has been thoroughly studied in the literature (see below in Section 4.1; D'Souza et al., 2001; Jordan and Rumelhart, 1992; Wolpert and Kawato, 1998). An internal model is used for the transformation between sensory and motor representations, for example how a movement of the arm will affect the position (and hence the perception) of the hand.

Since internal models perform a kind of simulation, the existence of internal models could be seen as indirect evidence for the existence of simulators, and thus by extension also for the embodied cognition hypothesis. Evidence for the existence of internal

models in the brain comes from psychophysical studies, where it can be found that the brain is able to swiftly adapt to changes in the environment in the way that movements are performed (for a review, see Kawato, 1999). For example, if an external force is applied to the arm (something keeps pushing the arm to the right, for example), this has a strong effect on the outcome of a movement. The brain can learn to compensate the force within just a few trials, so that the movement is executed as if there was no external force. If the force is then removed from one trial to the next, it results in an "over-compensation", which the brain reverses again within a few trials. If it was assumed that the brain controls movements by directly using an error signal from sensor readings (for example the visually perceived distance of the hand to a target point), this behavior could not be accounted for. In contrast, if the brain uses an internal model during control to produce error signals, the over-compensation can be interpreted by assuming that the internal model adapts to new sensory information and temporarily produces wrong signals (Kawato, 1999).

However, *action concepts* are used for more than "just" the control of movements and the prediction of sensory feedback. Particularly, in social cognition they are involved in the understanding of the behaviors and goals of others. Traditionally, this ability is accounted to one's having a "theory of mind", i.e. acknowledging that others have beliefs, knowledge and desires that are different from one's own. For example, from seeing you reach for a glass of water and bringing it to your mouth, I can infer that you have the desire to drink and the belief that the glass is filled with something to drink. The theory of mind is often also described as a uniquely human trait, which distinguishes the human species from other primates and animals (for a discussion, see Barrett et al., 2007). In the modular view of the organization of the brain, a "theory of mind module" would be assumed in the cognitive processing stage, which is responsible for the decoding of the actions of others, operating on an abstract representation of the sensory observations. This would also encompass inferring which action was performed by the other by mapping sensory information onto an abstract action representation, i.e. an action *concept*. In contrast, an embodied cognition account for social cognition proposes that the understanding of the observed behavior of others and their goals depends on a more direct mapping of the perception of their action execution onto one's own action representations. In that sense, decoding actions of others in social interaction, including speech, gestures or facial expressions, means activating one's own action representations, i.e. *simulating* the action. The idea is that the brain can employ the forward models of its available actions, each of which generates predictions about the unfolding of sensory information. One of the forward models will produce the best approximation of the event sequence that the action of the observed other produces, which allows the observer to interpret the observation as the action that is associated with the best matching forward model (Wolpert et al., 2003). This way, a very direct link between the sensory observation and the own action representation (through simulation of the forward model) can be established, without the need for a theory of mind module.

This view is supported by empirical evidences from experimental neuroscience,

where the so-called "mirror neurons" were discovered in the cortex of the macaque monkey. Mirror neurons are reported to fire both when a monkey executes a movement as well as when it observes someone else executing the same or a similar movement, and brain-imaging experiments with human subjects provide evidence for the existence of a similar mechanism in the human brain (see Rizzolatti and Destro, 2008). In initial experiments, mirror neurons were mostly found for grasping actions, for example for the action of grasping a nut from a tray. These neurons were found to be very selective in their firing, such that for example both the observed action and the action performed by the monkey had to use a precision grip, and using a tool to grasp a nut would not trigger any response (Gallese et al., 1996). The firing pattern of mirror neurons during the observation of actions of others allows for the interpretation that the same action is automatically re-enacted by the observer through embodied simulation (Gallese, 2003). Apart from grasp-related mirror neurons, also "audiovisual mirror neurons" have been found that are sensitive to the sound of an action (Kohler et al., 2002): These neurons fire when the monkey cracks a peanut, observes someone else crack a peanut, and also when it only hears a peanut being cracked. Thus, it seems that the observation of an action performed by someone else triggers a complex simulation of the event, which could allow the observer to understand the goals of the other (Gallese, 2003). Put more generally, representations of action, of perception and concepts seem to be all based on the same widespread, multi-modal networks of cortical regions that operate together and in close coupling with the environment to support the control of action and the processing of sensory information, up to "cognitive" tasks, such as the inference of the goals of others.

3.1.3 The Concept of Schema

It should be noted that there is in many ways a substantial similarity between aspects of embodied cognition as described in the more recent literature and the literature on a more classic concept from the literature in psychology of how the brain organizes knowledge, which is called the *schema*. This is also reflected by the terminologies used in the literature. For example, Barsalou emphasizes that he sees perceptual symbols as *schematic* in nature (Barsalou, 1999), and Gallese and Lakoff make a direct reference by calling their embodied version of the "grasp" concept a *grasp schema* (Gallese and Lakoff, 2005). Additionally, also on the level of the functions that are ascribed to the mechanisms of embodied cognition and the schema, there is lots of similarity that is worth mentioning.

The notation of "schema" is often associated with its use in the context of artificial intelligence (see below in Section 3.2.1), as it was described by Minsky in his work on the frame representation (Minsky, 1974). The terminology of "schema" really goes back to the writings of Kant (Kant, 1781), although the theoretical roots of the concept of schemata, as it is conceived of in today's literature in the Cognitive Sciences, can be found in the work of Bartlett (Bartlett, 1932). Bartlett extended on ideas that were proposed earlier by Head and Holmes, who studied the cases of several patients with lesions of the cortex (Head and Holmes, 1911).

Head and Holmes describe, among other things, the ability of the patients to recognize changes to their body posture, both in voluntarily produced, as well as in passive movements, and their ability to localize the position of a stimulus on the own body. These abilities seem to be dissociated in the brain, as a patient with a deficit in the one ability is not necessarily impaired in the other. For example, Head and Holmes reported that one of their patients, when touched on some spot on the hand, was able to indicate on a diagram or on another person's hand the exact position of the spot that had been touched, while he was unable to tell the location in space where his own hand was lying. On the other hand, a different patient could tell that he had been touched, but was unable to specify the location on his body where he had been touched.

These findings led Head and Holmes to argue that the human brain has a means to arrange incoming stimuli into existing models of the own body, which they termed *schemata*. A schema, in their description, is a plastic model that is constantly being updated. All perceptual stimuli automatically are put into relation to the schema representation and are measured as a relative change. They postulated the existence of at least two such schemata, one that represents a model of the body posture, and another one as a model of the surface of the body. According to Head and Holmes, these schemata are constantly being updated with every movement that one does or that is being elicited. Thus, the schema representing the body posture (i.e. the "body-schema") is the knowledge about the current posture of the own body, and with every incoming stimulus this schema is updated to match the new information, before the relative change comes into consciousness: Schemata "modify the impressions produced by incoming sensory impulses in such a way that the final sensations of position, or of locality, rise into consciousness charged with a relation to something that has happened before." (Head and Holmes, 1911, p. 189)

Head and Holmes account all ability to recognize postures and locality of stimulation to the schemata. Also the recognition of locations beyond the limits of the own body, as for example the tip of a tool, is in their account a function of the schema. The plasticity of the schema allows to incorporate changes to the posture, even to such an extent that objects (and tools) that we carry become part of the schema. Lesions of the schemata render individuals unable to make use of the incoming stimuli.

Bartlett took up this idea of schemata and extended it to a "theory of remembering" (Bartlett, 1932), contrasting it to the metaphor of the cortex as a "storehouse" of "memory traces" that predominated the literature of the time, whereby memorizing was the recording of events and reactions as static structures, and the act of remembering was seen as the retrieval of suitable traces. According to Bartlett, the human memory is not such a collection of static entries, but similar past experiences and reactions are organized into a "unitary mass," that is, a plastic representation has been formed that incorporates all past reactions of a certain kind. He picks up the notion of schema as a description for this idea, as the representation he describes performs a similar function as what has been described for the body schema by Head and Holmes earlier. The schemata are used by the organism to perceive a new situation in a way that it is related to the past experiences of that organism. An example that Bartlett gives

is that of performing a motor task: Swinging a tennis racket. Performing it anew requires not simply to retrieve a textbook version of the swing but to adapt it to the current situation, the available visual queues about the trajectory of the ball, the balancing of the body, and so on. From today's point of view this may seem rather obvious, as any motor controller in a robotic system that has been designed to achieve a certain motor task implements a dynamical system, inherently relying on sensory input. However, Bartlett applies this functioning also to higher-level cognitive tasks, such as remembering a story or a scene seen on a photograph. In his view, it is not a listing of elements and their features that is remembered, but mainly an overall "impression" of the situation (Bartlett speaks of an "attitude" towards the thing to be remembered). When a situation is perceived, it is matched against the organism's schemata, which are then adapted to the new information, just as the body schema is adapted to the movement of a limb, or the state of the motor controller is adapted to the sensory input. Remembering the situation is then not the process of fitting together a list of elements, but to recall the overall "impression" by fitting the schemata to bits of information that can be restored. All details are then filled in from there. This seems to be Bartlett's main point, that the act of remembering is not one of retrieval but one of reconstruction, based on the schema as "an active organised setting" (Bartlett, 1932, p. 201).

These are already strong parallels between the embodied cognition hypothesis and the concept of schemata: Bartlett's view of memory recall as a *reconstructive* process falls very well in line with Damasio's description of the basic functioning of the cortex, according to which the process of recall is the retro-activation of sensorimotor regions. If we take for example the task to remember the layout of a certain room, say an office that we have seen earlier, according to Bartlett this will involve first recalling an overall, very unspecific, impression of the room. As our brain works on recalling the appearance of the room, more and more details will be filled in. Some of the details will be based on actual details that we remember, because we attended them (cf. Barsalou, 1999), but others will be induced from our generic knowledge about the layout of offices. Thus, where we cannot recall the details, our *schema knowledge* about offices fills in the rest. In the light of Damasio's theory, we can interpret this process in terms of the flow of activation in the cortical hierarchy. When we stand in an office, sensory information first is extracted from the scene as we attend the furniture, the objects lying around, the windows, and so on. In the course of processing the sensory stimuli along the pathways of the cortex, modality-specific convergence-divergence zones bundle information, for example about object shapes and texture in the visual cortex. From there, in cross-modal convergence-divergence zones the information from the different modalities is combined, so that ultimately a large network of distributed activation patterns is responsible for the sensory experience of the room. When considering Barsalou's view of simulators, this process is largely determined by our past experience, as we will for example classify the desk as such, because our brain has established a *desk*-simulator that becomes active in a cross-modal convergence-divergence zone, just as an overall *office*-simulator will become active as we acknowledge

that we are standing in an office.

Then, when we are asked to recall the layout of the room later on, our brain will make use of the same network of distributed representations, by employing the *office*-simulator, and along with it other simulators for the objects that we encountered. This will give rise to a retro-activation of the sensorimotor regions of the cortex, which will allow us to remember various aspects of the room. Just as in Bartlett's account for recall, also the embodied simulation account assumes that much of what is recalled (i.e. simulated) will not match the exact details of the actual room, but will correspond to a generic embodied simulation of the office simulator, that is ultimately, activation patterns of neurons in the sensorimotor regions as they usually occur when we are in an office and attend the objects in it.

Also for the body-schema there exist parallels to modern views in embodied cognition. Head and Holmes propose that it is an internal representation of the own body, which is continuously updated to match new sensory information (Head and Holmes, 1911). On the one hand, this is the case during action, as one moves the own body and processes the sensory feedback, for example about the visual position of the hands. On the other hand, Head and Holmes argue that the body-schema is also involved in the recognition of postures, thus ascribing the function of recognizing the perceptual feedback of own actions to the same underlying processes as the recognition of postures in others, which, although arguably in a stronger sense, is also the fundamental principle behind the mirror neuron hypothesis (cf. Section 3.1.2): The automatic re-enactment of observed actions by making use of the same representations that are involved in one's own motor performance.

Some years after Bartlett, but before the introduction of the term "schema" to the field of artificial intelligence by Minsky, Piaget coined the term in his theory of cognitive development (Piaget, 1997 [1953]). Piaget describes schemata as the cognitive basis for both physical and mental action. He considers intelligence in line with the problem faced by biological systems needing to persist in their environment. In his argumentation, cognitive development can be regarded analogous to how adaptation appears on the level of biochemical organization. As he argues, biological organisms necessarily need to *adapt* to their dynamically changing environment, as in the example of a simple biological organism. What defines the organism are the physiochemical and kinetic *processes*, that are active again and again during the life of the organism. For example, the organism could ingest a substance from the environment, transforming it into another substance, and thus fueling another process. Through the coupling with the environment, the various processes of the organism orchestrate a cycle, where the result of one process triggers another, of which the result triggers again the next, and so on, until the first process again becomes active. The organism thus uses what the environment has to offer by using its processes. Piaget says, it is with the environment in a relationship of *assimilation*, which means that it unites the environmental elements with its existing processes as they are, in such a way as to incorporate them into the cycle. It uses its processes to deal with environmental inputs. As long as the cycle is stable, the organism functions in its environment.

As the environment changes, the organism necessarily needs to adapt to these changes, otherwise its cycle is interrupted and it ceases to function. Assume for example, one of the substances on which the organism relies changes in the environment. If the corresponding process in the organism is not able to handle the changed substance properly, that is, the environmental input cannot be assimilated, the process needs to be adapted. Thus, upon successful adaptation, the responsible process is transformed into a modified variant, which is capable to handle the new kind of input. Piaget calls this *accommodation*. Concluding, Piaget says that "adaptation is an equilibrium between assimilation and accommodation" (Piaget, 1997 [1953], p. 6).

The same principles of assimilation and accommodation apply also to intelligence, according to Piaget. He sees intelligence merely as another form of adaptation to the environment, and that its function is to "structure the universe just as the organism structures its immediate environment" (Piaget, 1997 [1953], p. 4). On the level of sensorimotor intelligence, the processes determining the organism are the schemata. Here, adaptation means structuring the environment: Reducing the incoming information to the known. When an infant has learned a sensorimotor program (i.e. a schema) to grasp a ball, but finds a different object, say a toy cup, it can readily employ the available schema to grasp the cup. It *assimilates* the object *into the grasp schema*, using the schema to interpret the world. However, since grasping a cup is slightly different from grasping a ball, although the learned schema may succeed in grasping the object, the sensory experiences of performing the action differ from those when grasping a ball. Thus, in order to adapt well to her environment, the infant will use the new impressions to extend her knowledge of the world, by *accommodating* the schema to better represent the new information.

Piaget's ideas of the process of assimilation and accommodation underlying the schema are often used as motivation for various computational models (see below in Section 3.2.1). However, as for example Klahr notes, this part of Piaget's theory is not sufficiently constrained to allow modeling (Klahr, 1995), so that models that use it as motivation are often only vaguely related to the concept of schema. Piaget's theory is also often criticized in the literature, as he describes cognitive development as organized in distinct stages, with qualitative changes in the child's behavior across the different stages (Piaget, 1997 [1953]). However, this is a too simplified description of the progress of cognitive development and does not account well for experimental data (Mandler, 1992; Thelen and Smith, 1996). For example, Piaget argues that infants demonstrate the A-not-B error (see Section 2.5.1) in an earlier developmental stage, but as they enter the next stage, they will solve the task correctly. However, Thelen and Smith have demonstrated that this monolithic change posited by Piaget is wrong, as the "complexity and messiness of cognitive development" (Thelen and Smith, 1996, pp. 21–22) becomes evident when tasks are altered. For example, as the timing of the experiment is changed and waiting times for the infant are shortened, it becomes less likely that the error is observed also with younger infants (cf. Section 2.5.1).

The properties of schemata of assimilation and accommodation can also be found in Barsalou's description of simulators (Barsalou, 1999). Assimilation is the process of

making use of existing mental structures to interpret the world: In Barsalou's terms, this corresponds to the categorization of stimuli by comparing them to perceptual symbols. Accommodation is the process of incorporating the new information by slightly changing the existing representation, schema or simulator.

While the above cited works on the concept of schema focus on explaining how the schema as a knowledge structure can enable and support cognitive function, Mandler takes the viewpoint of developmental psychology and aims at addressing the question of how schema knowledge can be acquired during cognitive development (Mandler, 1992). Mandler derives her view from work in cognitive linguistics, where the so-called "image schemata" are seen as the conceptual basis also in adults (Johnson, 1987; Lakoff, 1987; Lakoff and Johnson, 1980). Image schemata represent concepts that can directly be observed in perceptual structures, such as the *path* on which an object travels, or the *containment* of one object in another. According to Lakoff and Johnson, these are primitive representations that combine into more complex and abstract concepts by means of conceptual metaphors: To understand abstract meanings, image schemata are employed to attach actual experiences to the concept, as for example evident in sayings such as "life is a journey", or "argument is war" and "attacking an argument", and so on (Lakoff and Johnson, 1980). Mandler argues that many image schemata, such as *animate motion* and *caused inanimate motion*, are acquired very early on in development and become an integral part of a wide range of conceptual knowledge. As Mandler states, perceptual similarity provides one level for concept formation, but importantly also image schemata are employed as the infant learns concepts such as "agency". In a process of "perceptual analysis", perceptual stimuli are transformed into the perceptual format of image schemata to support the formation of new concepts, according to Mandler (Mandler, 1992). This can be compared to how Head and Holmes describe perception to be automatically transformed or augmented by the cognitive system, as stimuli are compared to existing schema representations as they are processed.

Taken together, it can be said that "embodied" ideas of how the brain organizes knowledge can already be found in the literature of the beginning of the last century. The concept of schemata, as it was originally conceived of, has many similarities with modern theories of embodied cognition. With the idea of schemata being the motivation behind several knowledge representations in the field of artificial intelligence, a schema is now sometimes also interpreted to be a very abstract, static knowledge representation that has been hand-crafted for a computer program, and thus would fall in line with the brain-is-a-computer metaphor. However, the original description of Bartlett points quite in the opposite direction, and seeing a schema as a static knowledge structure can be said to be a misinterpretation of his work (Pfeifer and Scheier, 1994).

3.1.4 Summary

In sum, the embodied cognition hypothesis draws a fundamentally different picture of cognitive architecture than proposed by the cognitivist view. One of the main arguments is that a substantial part of cognitive processing, not only that related directly to

perception and control, is implemented as, or inherently supported by, processes in the sensorimotor regions of cortex. A hierarchy is formed of convergence-divergence zones (Damasio and Meyer, 2008; Damasio, 1989) or simulators (Barsalou, 1999, 2008; Simmons and Barsalou, 2003), which represent multi-modal information in a distributed manner and are used in different ways depending on the task: During learning, information about the simultaneous activation of neurons across regions are stored in association regions (Damasio, 1989) and simulators are formed (Barsalou, 1999). For recall and conceptualization, activation in early sensorimotor regions is reactivated to simulate the perceptual experience (Barsalou, 1999; Damasio, 1989). Categorization is the process of matching simulated and real perceptual activation patterns (Barsalou, 1999). Cross-modal integration happens early on inside sensorimotor regions via inter-regional connections, on the level of activation in neural fields, where neural dynamics can be accounted for decision making in movement preparation (Erlhagen and Schöner, 2002; Schöner et al., 1997; Thelen et al., 2001) as well as in tasks related to sensory information (Johnson et al., 2008). It is hypothesized that the cortex is a complex dynamical system, composed of basic elements that implement function uniformly across the whole system in a distributed way (McClelland et al., 2010), such as simulators (Barsalou, 1999) or dynamic neural fields (Johnson et al., 2008). Sensory perceptions of for example objects, as well as actions, are represented and controlled by distributed representations across multiple clusters of neurons (Barsalou, 1999; Damasio, 1989; Gallese, 2008; Gallese and Lakoff, 2005; Glenberg and Gallese, 2011) (hypothetically in the form of topographically organized neural fields), each of which codes for a specific aspect. For example, in visual perception, different clusters of neurons code for shape features and color features of an object, and the clusters are combined into a cortical network via convergence-divergence zones (Damasio, 1989). For action, clusters code for parameters of the action, such as locations towards which the action is directed, or amplitudes and forces (Gallese and Lakoff, 2005; Smith and Thelen, 2003; Thelen et al., 2001). These clusters are then associated with sensory representations, so that internal models are formed to support prediction and planning (Wolpert and Kawato, 1998), as well as to form the basis for the understanding of the actions of others (Wolpert et al., 2003), on the basis of embodied simulation (Gallese, 2003). Representations higher up in the cortical hierarchy structure lower-level representations, so as to form for example representations of goal directed actions (Gallese, 2008). The cortical regions on higher hierarchical levels are then responsible to coordinate the activation across associated lower level regions (Damasio, 1989). Furthermore, as the activation in some of the regions determine the way in which an action is performed, for example by specifying the target position of a reaching action in ego-centric space, they can be seen as providing a kind of *argument structure* to the action (Gallese, 2008; Gallese and Lakoff, 2005). Thus, the result of the activation in one area, for example corresponding to the action of grasping, is modulated by the activation in another area, for example determining *where* to grasp. This is related to the frame-like organization, which Barsalou hypothesizes for the assemblage of simulations (Barsalou, 1999).

Embodied cognition is in many important ways compatible with the concept of

schema. A central property, which both views have in common and put an emphasis on, is the active nature of the stored knowledge. Given a noisy or incomplete sensory cue, it is proposed that the cognitive system interprets this input in a best-fit manner by matching it against its existing knowledge structures. In the vocabulary of Bartlett's schema theory, this is described as reconstructing information based on schema knowledge (Bartlett, 1932), which is similar to Damasio's view that information spreads via convergence-divergence zones to reconstruct a multimodal stimulus based on learned associations (Damasio, 1989). Likewise, Piaget proposes that sensory stimuli are assimilated by the existing schemata (Piaget, 1997 [1953]), and Barsalou argues that stimuli are categorized by comparing them with embodied simulations (Barsalou, 1999). Thus, when a partial stimulus is matched with a particular simulator, missing information can be substituted by a multimodal simulation of the matched simulator.

When the embodied cognition hypothesis and the concept of schema are used as motivation for the modeling of a robot cognitive architecture, several modeling constraints can be formulated:

- The cognitive system is based on active elements, which are in concert responsible for a wide range of cognitive functions. The hypothesized elements are termed differently in the various theories outlined above (such as "schemata" or "simulators") and differ in the details of their description, but they share important properties across the theories. In the following, the term "schema" will be adopted for referring to this active element of the cognitive system, even though some of the here outlined properties are more pronounced in theories that use another designation.

- Schemata store knowledge about patterns of information, and the correlation of patterns across modalities. As they come to represent sensorimotor patterns, schemata become *activated* by the perception of these patterns. The interpretation that the cognitive system has of a situation corresponds to the collection of its currently active schemata.

- Based on the information that they represent, schemata form self-organized representations in a data-driven manner. These new levels of representation provide the basis for a hierarchical structure of the cognitive system. As the information that is represented by schemata on higher levels of the hierarchy covers wider and wider parts of the sensorimotor input and output space of the system, schemata represent increasingly multimodal and abstract knowledge upwards the hierarchy.

- An important property of the cognitive system is its ability to *reconstruct* sensorimotor situations based on the knowledge stored by the schemata. The process of reconstruction can involve single schemata, or collaborations of several schemata, which mutually influence each other and the outcome of the reconstruction. The format of the reconstruction is the same as that of low-level perceptual input.

- Schemata compete against each other in the processing of information. Importantly, schemata that have won this competition, and thus have become activated

by an input pattern, will adapt to this newly available information.

- Schemata enable the understanding of observed actions of others through an automatic covert reenactment of the same action, as an embodied simulation of the own action representation.

3.2 Related Computational Models

3.2.1 Models Based on the Concept of Schema

As the concept of schema has been interpreted in many different ways, computational models that use it as motivation are also very different from each other. The earliest models have been proposed in symbolic A.I. research, and propose the schema to be a structured representation.

Symbolic Implementations of Schemata

In the 1970s, researchers in A.I. felt that there was a need for a larger scale representation that gave more structure to the knowledge of a computational agent, as systems based on rather simple symbolic rules as atomic representations were facing several difficulties. One was related to the computational complexity of search algorithms in reasoning applications, as they had to handle large knowledge bases of symbolic rules (see Section 2.2.3); another one was related to the problem of language understanding, where the information that is explicitly given in a piece of text is often insufficient to make proper judgements (Minsky, 1974; Schank, 1972). In both cases, so was the intuition of researchers, a network of larger scale representations should provide the agent with a kind of common-sense knowledge: In the former case to limit a search to relevant bits of knowledge, in the latter case to extend a given bit of information by the required implicit knowledge.

As Bartlett's work describes the property of the faculty of memory to condense past experiences into schematic representations that only capture the relevant information, it became the motivation for several A.I. researchers to propose networks of structured representations as a symbolic knowledge base for computational agents, most notable being Minsky's "frames" (Minsky, 1974) and Schank and Abelson's "scripts" (Schank and Abelson, 1977), both of which are similar in their implementation but aim at different goals. Minsky's frames were meant as a means to reflect the structure of external reality in a symbolic representation, in terms of the relationships between entities, and how parts are made up out of sub-parts. Minsky describes as an example how cubes are composed out of faces and edges, each of which are in a certain relationship with the others, such as one face being parallel to another, or an edge connecting two perpendicular faces. Another example is that of a chair, having parts such as a certain number of legs and a backrest, as well as some defined properties such as a given height and weight. Minsky proposes that this sort of information about the relationship between pieces of knowledge should be stored in what he calls a frame, essentially being

a structured list of properties and their values. Each value could be assigned a default value, in case no specific information is given (for example, the default number of legs for a chair might be four). Schank and Abelson's scripts on the other hand are meant to serve the purpose of representing information about the sequential order of events as they occur and what actions become relevant when. For example, the overall event of going out to dine in a restaurant is made up of a sequence of situations and possible actions that can be pursued: Upon entering a restaurant, a possible action is to find a waiter to ask to be seated; after having eaten a main course, common actions are asking for the bill or for the desert menu, and so on.

Later on, Drescher formulated a cognitivist interpretation of Piaget's schema theory (see Section 3.1.3) (Drescher, 1991), which is structurally closely related to Minsky's frames and Schank and Abelson's scripts. It groups together symbolic descriptions of a context, an action and a result into a data structure, which Drescher uses for problem solving mechanisms, for example by "chaining" these schemata to describe a sequence of action and how it will transform a situation (similar to how cognitivist architectures implement planning, cf. Section 2.2.1): Departing from the description of a given situation, the schema information can be used to compute the change made by some action to this situation, producing the description of another situation, which again can be manipulated by other schemata, and so on. Reaching a goal then means to search through the library of schema representations to find a sequence of actions that will transform the original situation into a goal situation.

However, as already noted further above, the symbolic implementations of the concept of schema hardly capture the essential point of the concept, of it being a flexible structure that adapts to sensory information and provides a "best-fit" interpretation by coordinating the activation of dynamic sensorimotor representations.

Localist Networks with Global Constraint Satisfaction

Shortly after the symbolic interpretations of the concept of schema had been proposed, and with the dawn of connectionism in the 1980s, Rumelhart proposed a more dynamic model of schemata as a process of global constraint satisfaction in a localist neural network representation (Rumelhart, 1984; Rumelhart et al., 1986). In this kind of network, individual network nodes represent hypotheses about the presence of a certain feature in the input, and connections between nodes represent constraints that the hypotheses pose upon each other. For example, if one feature is an indicator for another feature, then there should be an excitatory connection from the node representing the former to the node representing the latter. The other way around, if one feature is almost certainly not present if another feature has been detected, then an inhibitory connection between the two responsible nodes would encode this constraint. Such a network implements a dynamical system that, when presented with an input activation, after several iterations converges to a stable state in which a set of nodes is active that corresponds to a consistent set of hypotheses (Hinton and Sejnowski, 1986). A helpful example for a constraint satisfaction network is presented in Figure 3.2.

In the interpretation of Rumelhart et al., "schemata are like models of the outside

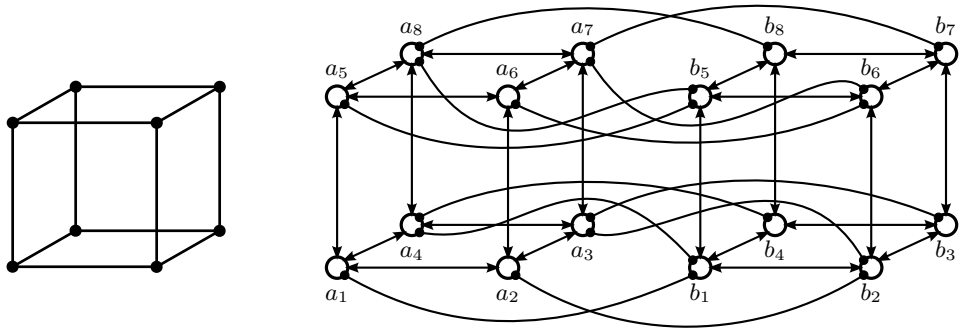


Figure 3.2: (a) Necker cube, (b) a constraint satisfaction network that has the two possible interpretations of the Necker cube as attractor points, where one attractor point is the network state in which nodes a_i are active and nodes b_i are inactive, and the other vice versa. Connections shown with arrow markers are excitatory, connections with dot markers are inhibitory (not all connections shown). Drawing based on (Rumelhart et al., 1986).

world", and processing information with the use of a schema corresponds to discovering a "consistent configuration of schemata [...] which, in concert, offer the best account for the input. This configuration of schemata constitutes the *interpretation* of the input" (Rumelhart et al., 1986, p. 18, their emphasis). This interpretation of the concept of schema naturally fits into the language of constraint satisfaction networks. The model of the world is composed out of the features represented by the network nodes and constraints for valid configurations are stored in the connection weights. In contrast to the symbolic schema implementations described above (see Section 3.2.1), according to which the schema is an explicit representation of variables and their default values, the formulation as a constraint satisfaction network offers a view in which the schema is not explicitly present in the system, but is stored in a distributed code in the connection weights. When an input is presented, the network relaxes to an interpretation of the input, or in Piaget's terms, assimilates the input.

As an example, Rumelhart et al. present a network that should encode schematic knowledge about different types of rooms. It is a fully connected single-layer network with 40 units, each of which represents a room feature in a localist fashion, such as *telephone*, *desk*, *refrigerator* and *bed* (Rumelhart et al., 1986). The network weights are set to reflect the pairwise probabilities of co-occurrences of the room features, computed from descriptions that were collected for 80 different rooms (offices, living rooms, kitchens, bathrooms and bedrooms). This resulted for example in a strong excitatory connection between the *telephone* and *desk* nodes, since both items frequently co-occur in offices, and a strong inhibitory connection between the *refrigerator* and *bed* nodes, since the two rarely can be found in the same room. The network dynamics then drives the activation from any initial state into one of several attractor states. For example, by clamping the *oven* and *ceiling* nodes to a high activity and initially setting all other

nodes to zero-activity, the network will transition into a state in which also nodes such as *cupboard*, *toaster*, *refrigerator*, *sink* and *coffee-cup* are activated.

Rumelhart et al. interpret the behavior of the network as schema-like, arguing that the structure provided by a schema can be implicitly encoded in the connection weights in a global constraint satisfaction network (Rumelhart et al., 1986). One example for this is how attributes and values of a schema can be explained in terms of the network's properties[1]: While some of the units in the network share strong connections, and thus almost always become co-activated as the network settles into a stable state, others are more loosely connected and may or may not become activated. Again other sets of units can be connected via inhibitory connections, such that either one or another unit will be activated as the network settles, but they will never be activated together. Rumelhart et al. argue that this kind of behavior can be interpreted as the filling of attributes with values (Rumelhart et al., 1986): The strongly connected units form the core of a schema, which will always be active as a coalition. The weakly connected units represent attributes of the schema, which can either be present or not, and if there are different values possible then the network dynamics will select one of them.

Another implementation of the concept of schema in a form of global constraint satisfaction network is Cooper and Shallice's "interactive activation model" of routine action selection (Cooper and Shallice, 2000, 2006). Here, a hierarchy of nodes represents the actions of an agent and action goals at different levels of granularity, in their example for the routine action of preparing instant coffee. At the bottom of the hierarchy are nodes representing the primitive actions of the agent (*pick up*, *tear*, etc.), followed by nodes representing the goals which can be attained by these actions (*hold*, *open*, etc.). Nodes in higher hierarchical levels represent compositional actions and goals (e.g. *add sugar from packet*, *sugar into coffee*), which can include several alternative possibilities (taking sugar from a packet, or taking it from a bowl). Activation in the network flows down the hierarchy along pre-specified connections, from the overall action goal down to the primitive actions. When there are alternative means to achieve a sub-goal (such as using a packet of sugar or a bowl of sugar), there is lateral inhibition between the corresponding nodes on the same level of the hierarchy, which forces the activation dynamics of the network to select one among the alternatives (similar to how the network of Rumelhart et al. can chose among one of several conflicting attractor states, see above). Furthermore, environmental affordances influence the activation of the network, such that actions receive activation when the current state of the environment allows for their execution (for example, *pick up* receives activation when there is something to be picked up and the agent's hand is currently unoccupied). Cooper and Shallice report on the one hand that their model is able to generate sequences of actions that correctly produce the overall goal, and on the other hand argue that

[1]While the idea that schemata are structures with attributes and their values certainly comes from symbolic accounts of the concept of schema (Brewer and Nakamura, 1984; Minsky, 1974; Rumelhart, 1984; Schank and Abelson, 1977), also more recent accounts, although explicitly denying schemata to be of symbolic nature, assume there to be some form of variable-like property (Barsalou, 1999; Gallese and Lakoff, 2005), for example in the form of variable firing patterns in clusters of neurons coding for a schema.

it will also eventually produces errors (such as omitting a step) in a similar way as found in human performance (Cooper and Shallice, 2000). As compared to the model proposed by Rumelhart et al., which is focused on explaining how perception takes place in a constraint satisfaction network as a static input results in a stable final state, Cooper and Shallice's model demonstrates that a similar network can also be applied in sequential tasks, where it traverses several states.

In contrast to the symbolic implementations of schemata, these models allow to interpret the concept of schema in a more dynamic way, which seems to be more compatible with the theories described in Sections 3.1.1 to 3.1.3. It should also be noted that more recently, a growing number of connectionist models have been proposed that subscribe to the same underlying mechanism of global competition between alternative hypotheses as models for various cognitive phenomena (see Maia and Cleeremans, 2005). However, the form of network models described in this section has the drawback that it is based on the localist assumption, so that each network node represents a distinct concept. This only allows for binary combination of different representations (such as a room with or without a bed depending on whether or not the *bed* node is active), but not for a graded integration of concepts (such as a car which is like a truck but the size of a compact car). While this might be a criticism on a rather theoretical level, the same argument does also apply for a more practical limitation of localist models in the realm of robotics: When it comes to the representation of action knowledge, the nodes in a localist model correspond to basic action units, or primitive action controllers (as in Cooper and Shallice's interactive activation model for primitive actions, such as *pick up*, *tear*, etc.). The only form of integration of several primitive actions that is possible in such a model is the sequencing of actions. However, in a robotic system it would be desirable to allow for a more flexible integration on the sub-action level, i.e. blending sub-actions together. For example, a flexible robot should be able to reach for an object with one hand *while* keeping another object steady with the other hand. Such flexible integration is only possible if it is realized below the level of distinct action primitives, an issue which will be further explored in Chapter 4.

Schema Models Related to Behavior-Based Robotics

Another approach to modeling schemata, which is also strongly related to the behavior-based robotics movement, is based on Arbib's schema theory (Arbib, 1998). Arbib's account for schemata is concerned with explaining the neural basis for behavior and argues that schemata are modular neural structures that each encode the agent's knowledge about some meaningful aspect of the environment. These knowledge structures are separated into two kinds of components: Perceptual schemata detect whether a certain feature is present in the current sensory input, and motor schemata encode behavioral responses to respond to these features. The motor schemata are connected to the perceptual schemata via excitatory or inhibitory connections, thus implementing behavioral rules ("if A is present, then activate behavior B"). In an example, Arbib proposes a schema-based model of a frog, which uses two perceptual schemata, *detect-all-moving-objects* and *detect-large-moving-object*, and two motor schemata, *snap*

and *avoid*. The *snap* behavior is excited by *detect-all-moving-objects* and inhibited by *detect-large-moving-object* ("snap if there is a moving object that is not large"), while the *avoid* behavior is excited by the *detect-large-moving-objects* ("avoid large moving objects"). This corresponds to the behavioral responses of snapping and avoiding of frogs as observed in studies (Arbib, 1998). Arbib argues that schemata not necessarily need to be linked to unique neural structures in the brain. Instead, the schema can be seen as a useful abstraction from neural implementation details, but more directly linked to explaining behavior than symbolic representations are. Similarly to the latter, Arbib argues that schemata can be instantiated, for example as several instances of the *detect-all-moving-objects* schema to represent two or more moving objects at a time.

For a computational model of schemata this equates to building a modular system, much in the sense of a behavior-based architecture (see Section 2.3), where individual components implement on the one hand detection devices for relevant environmental cues, and on the other hand the appropriate motor responses to react to the presence of sensory cues. The internal mechanics of these components can be arbitrarily implemented, but the details of how the components interact are fixed by design. In Arbib's schema model, perceptual schemata encode in their activation value how certain they are about the presence of a sensory cue. Their activation level is directly transfered to motor schemata, either in excitation or inhibition. If a motor schema's activity level exceeds a pre-defined threshold, the corresponding motor program is sent to the actuators.

In later work, Oztop and Arbib have proposed a schema-based model of the mirror neuron system in the premotor cortex of the macaque monkey (Oztop and Arbib, 2002). It includes perceptual schemata for the recognition of hand shapes (to distinguish different types of grasps) and of hand motion, as well as of object features, object location and of object affordances. The motor schemata encode the motor programs for reaching and for grasping. Additionally, a dedicated mirror system schema learns to associate motor programs for grasping with object affordances and observed hand states. When the agent itself performs grasping actions, both the perceptual schemata for detecting object affordances and recognizing the own hand state are active, as well as the motor schema for grasping, allowing the mirror system schema to learn the appropriate associations. It can then be activated in two circumstances, on the one hand when the agent itself performs a grasp, but also when the associated perceptual schemata are active but the grasp motor schema is not. This is the case when the agent observes someone else perform a grasping action, given that the perceptual schema detects both the own hand and another's hands in the same way.

Similar to Arbib's approach, Arkin's AuRA architecture (see Section 2.3.1) implements a model of schemata that assumes that schemata are instantiated when needed: When the robot encounters an obstacle, a new schema instance is created to generate a repellant force at the position of said obstacle, and the instance is maintained as long as the obstacle is in range (Arkin, 1998). In contrast to Arbib's models however, there is no separation between perceptual and motor schemata.

Another computational model that does use perceptual and motor schemata in Ar-

bib's sense is Pezzulo and Calvi's "Artificial Knowledge Interface for Reasoning Applications" (AKIRA, Pezzulo and Calvi, 2006a,b, 2007; Pezzulo et al., 2005). In contrast to Arbib's model, schemata in AKIRA additionally implement forward models and their activity depends on how well each schema is able to predict the sensory data. Pezzulo and Calvi describe a computational model of the behavior of a praying mantis (Pezzulo and Calvi, 2006a). It uses perceptual schemata such as *detect prey*, *detect predator* or *detect obstacle*, and motor schemata such as *chase*, *escape* and *avoid obstacle*. Motor commands generated by multiple schema instances are integrated through fuzzy command fusion, which on the one hand holds benefits as compared to vector-based combination (Safiotti, 1997) as it is used for example in AuRA, but on the other hand has difficulties in scaling up to higher-dimensional problems (Sala et al., 2005) (motor commands in Pezzulo and Calvi's simulations are one-dimensional, for turning and advancing on a two-dimensional plane).

Furthermore, Gläser et al. proposed a neural-network-based computational model of schemata, which they say implements "generic behaviors", meaning that the activation of one behavior, across different initial situations, should always result in the same final situation, which corresponds to the *goal* of that behavior (Gläser et al., 2009). Gläser et al.'s implementation of schemata is intended to model four important properties of the concept of schema: A forward model should allow the simulation of the outcome of an action execution; an inverse model is used to enable goal-directed action; a third internal model, which they call a "schemata recognizer", provides the basis for the recognition of observed actions by directly matching them with the own action repertoire; and finally, a self-organized neural field provides a new level of representation that could be used for the development of hierarchical systems. Each neuron in this neural field, which they call a "schemata map", represents a goal situation of the according behavior. In the examples provided by Gläser et al., this corresponds to bringing the hand and the gaze of a simulated robot to target locations, so that one neuron would for example represent the goal to bring the hand to the upper left corner and the gaze to the center of the workspace of the robot. The field is learned in a topography-preserving way, to ensure that similar goals are represented by neighboring neurons in the field, which allows the model to blend between prototypic solutions. The representation of goal situations in the schemata map is used to modulate the other networks in the model: The activation of a certain goal will result in the inverse model generating motor commands for reaching that goal, and in the forward model producing estimates of the sensory feedback in the following time steps on the trajectory towards the goal. Furthermore, the schemata recognizer transforms sensory observations into estimated goal situations of observed action, by activating the corresponding neurons in the schemata map. Therefore, the same neurons in the schema map are active both when the robot performs an action itself, as well as when it observes an action execution. Thus, appart from Oztop and Arbib's model, also Gläser et al.'s model can be said to implement mirror neuron functionality.

All of the just described models fall into the category of behavior-based robotic systems, and as such are subject to the same concerns: It remains a difficult problem to

design the inter-module connectivity for systems that should behave not only reactively and insect-like, but should exhibit longer-term planning capabilities (see Section 2.3).

Schemata in a Robot Control Framework

The computational models presented so far directly aim at modeling schemata. Another model, which is primarily intended as a framework for designing robust robot control programs rather than modeling schemata, has also been related to a liberal interpretation of Piaget's account for schemata by its authors (Hart and Grupen, 2011; Hart et al., 2008b), that is, to the processes of accommodation and assimilation (see Section 3.1.3). The model makes use of a basis of rigorously designed controllers (Coelho and Grupen, 1997; Huber, 2000; Huber and Grupen, 1997), which guarantee stable motor responses and provide an event-based discretization of continuous sensorimotor streams. Each of the robot's controllers (i.e. primitive actions) is defined to be at any time in one of several distinct states, such as being inactive, currently running, or having converged to a target. Through this mechanism a discretization is achieved, which allows the application of combinatorial search algorithms and reinforcement learning methodology to find rewarding sequences of actions (i.e. subsequently executed primitive actions), based on a mechanism of intrinsic motivation (Hart and Grupen, 2011; Hart et al., 2008a). In intentionally restricted training scenarios, in which a robot only uses a subset of its available control programs to discover rewarding action sequences (for example only employing head-related actions or only using one arm) and where the environment is prepared for the learning session (for example by repeatedly placing a single object at different locations in front of the robot), Hart and Grupen demonstrate that a robot can discover policies for actions, such as searching and tracking salient objects, or reaching for and touching an object with the gripper.

In a subsequent step, the discovered control policies are generalized to new situations by extending the robot's exploration. This is accomplished by declaring different *types* of input and output variables for the system (comparable to typing in functional programming), and defining for each controller a list of input and output types that it supports (Hart and Grupen, 2011; Hart et al., 2008b). For example, a control scheme that is designed to bring an effector close to a target location can be instantiated in the model with any output of the type *configuration variable*, independent of whether the variable designates configurations of the left arm, right arm or both arms, and any input of the type *Cartesian position*, independent of whether the position has been found using stereo triangulation or using a laser range finder. The system uses this to transform action policies that it has previously discovered into an abstract representation, in which the sequential order of primitive actions remains but the specific inputs and outputs are replaced by their *types*. For example, such an abstract representation of the action to reach and grasp would specify that a Cartesian position is necessary as a target for the action and that the output of the control program is a configuration variable, as well as provide the learned sequential structure for contingencies such as: If the target position is not very close to the gripper, then reach instead of grasp. The system then employs different instantiations of the abstract representation, for exam-

ple by sometimes using the left arm and sometimes the right arm, to discover similar actions that also produce reward.

Hart and Grupen relate their system to Piagetian schemata in the following way. As *accommodation* describes how an organism discovers new ways to adapt itself to its environment, Hart and Grupen argue that the first stage in their system (using reinforcement learning to find rewarding action sequences) corresponds to the process of accommodation; and *assimilation* describes how existing knowledge structures are fit to new situations, which Hart and Grupen relate to the second stage in their system of instantiating and evaluating different actions from an abstract representation based on *types* (Hart and Grupen, 2011; Hart et al., 2008b). Other than these two rather lose relations however, Hart and Grupen's model does not account for any other properties of the concept of schema.

3.2.2 Models of Embodied Simulation

Computational models of embodied simulation usually take the "simulation hypothesis" (Hesslow, 2002) as motivation, which states that sensorimotor structures can be activated as in normal action but without causing any overt movement. In the process, internal estimates of sensory feedback are generated, resembling sensory information that would have occurred if the action had been physically carried out. This anticipated feedback can be internally used to "close the loop" by again activating the sensorimotor structures and producing further anticipations, thus simulating a sensorimotor trajectory of events as they might be. This is a more specific form of embodied simulation than that proposed by others, as described in Section 3.1.2.

The short-term anticipation of sensorimotor events through embodied simulation has been demonstrated to be beneficial in several ways (Butz et al., 2007b), including shorter reaction times in sensorimotor control tasks (Mehta and Schaal, 2002), improved learning by comparing anticipated and actual results of actions (which will be the topic of Chapter 5; see also Wolpert et al., 2011), and the ability to evaluate sequences of actions in terms of their outcome before actually performing them (Shanahan, 2006).

In the study of anticipatory systems (for an overview of recent developments, see Butz et al., 2003, 2007b), several models have been proposed that are based on this idea. The overall idea, as it was outlined above, is well captured by Möller's "perception through anticipation" approach (Möller, 1999). Möller proposes that the process of anticipation provides the functional basis for perception. As it allows to augment sensory information with value information by anticipating possible valuable situations that can be reached by performing some action, the problem of having to determine relevance in sensory input (cf. Section 2.2.3) can be avoided (Möller, 1999). Instead of detecting features in the sensory input to generate an internal representation of the situation, and to use this representation for deciding on the next action, the anticipative system transforms the current sensory situation into possible future situations that can be reached by means of the agent's actions. In a computational model of the approach, Hoffmann and Möller use a multilayer perceptron to learn a forward model, which allows a mobile robot to associate the sensory input, paired with a motor command for

an action, with the expected outcome of the action. By concatenating the same forward model into a chain of several forward models (by using the output of one network as the sensory input signal for the next network), the model can perform a simulation of short action sequences and evaluate the anticipated outcome (Hoffmann and Möller, 2004). Hoffmann and Möller propose to use an optimization method to select the motor commands for the action sequence. Their experiments show that the model is capable of planning short action sequences to reach given goals, for example to plan a path to move away from scattered objects. In a more recent adaptation to the experiment, Schenck et al. replaced the optimization method with an inverse model that directly maps sensory inputs onto motor commands for a desired behavior. Training samples for the learning of the inverse model are generated by letting the system simulate the outcome of different action sequences, starting from real situations. This process greatly speeds up the training, as the internal simulation of sensorimotor trajectories is much more feasible than actually performing each action sequence, which is necessary as the training of the inverse model requires many samples (Schenck et al., 2012).

Appart from being used for short-term planning, anticipation through embodied simulation can also be used to dynamically select suitable controllers among multiple candidates. This has first been proposed by Wolpert and Kawato in the "Modular Selection and Identification for Control (MOSAIC) model", which is a model of how motor control is realized in the brain (Haruno et al., 2001; Wolpert and Kawato, 1998; Wolpert et al., 2003). It is intended as an account for how the brain copes with different contexts in control strategies, for example in the case of lifting objects of different weights, where more force has to be invested for heavier objects than for light objects. Two ways of implementing the control system are possible: Either a single, very complex controller should be able to cope with all possible contexts, or a modular approach is chosen where multiple controllers for the same action but in different contexts are used (Haruno et al., 2001). The MOSAIC model is of the latter kind, and it is argued that the brain selects the controllers by employing both a feedforward as well as a feedback strategy. In the feedforward selection of controllers, sensory information is evaluated, for example by using visual cues to guess the weight of the object. More importantly however, feedback information can be used to re-evaluate the selection, for example when the object turns out to be heavier than expected. This decomposition is achieved in MOSAIC by combining a forward model and an inverse model, as well as a "responsibility predictor" into a building block of the model. All building blocks receive the same inputs, which are contextual information from the sensors, a desired state that should be achieved by controlling the motors (for example the height to which the object should be lifted), an efferent copy of the last motor command, as well as feedback information for the current state. The building blocks operate in parallel, using their responsibility predictors to produce a-priori estimates of how well they will perform in the current context, and producing motor commands for the desired state. Most importantly however, as feedback information becomes available (after having started to lift the object), each building block uses this information to provide more accurate evaluations of their responsibilities, thus allowing the system to dynamically

change and improve its control strategy. The final motor command is generated by forming a linear combination of the individual motor commands that are provided by the building blocks, using the normalized responsibility signals as coefficients.

Similarly to MOSAIC, also the AKIRA model (see Section 3.2.1) uses the process of anticipation through forward simulation to compute the relevances for the system's components in the current situation. The activation of each schema in the system is not only computed based on a static evaluation strategy of the sensory input, as in Arbib's model (see Section 3.2.1), but also on how well the schema can anticipate future sensory inputs (Butz et al., 2007b; Pezzulo and Calvi, 2007; Pezzulo et al., 2007). For example in the model of the praying mantis, multiple perceptual schemata are used for the detection of prey, which are specifically tuned for different types of prey (slow moving and fast moving insects). The schema that best anticipates future locations of the prey gains most activation and determines the system's behavior.

Finally, also Shanahan's global workspace architecture implements embodied simulation as the forward simulation of the agent's actions (see Section 2.5.2; Shanahan, 2006). In contrast to the works described above, Shanahan's model implements a discrete time forward simulation, in which forward models predict the final outcome of the execution of an action.

While all of the just described approaches offer important insights into one relevant aspect of embodied simulation, that of internally rehearsing possible actions and thus anticipating their effects on the world, there are many more aspects to embodied simulation as the concept is used by theories of embodied cognition (as described above in Section 3.1.2, see also Svensson et al., 2009). Importantly, the core assumption behind the anticipatory systems described above is that processing begins with the current sensory situation (Möller, 1999), which is transformed in the process of simulation for anticipation. This only allows the system to plan for brief episodes into the future, over the time span of single actions or short action sequences, for example to anticipate the result of moving a small distance and turning left or right. However, one of the most important properties of the human conceptual system and key to its flexibility, as described for example in the works of Bartlett, Damasio or Barsalou, is the ability to *reconstruct* a distal sensorimotor situation from fragments of information, based on the knowledge that the system has acquired.

3.3 A Cognitive Architecture Based on Embodied Simulation

As one of the main contributions of this thesis, a new cognitive architecture will now be formulated in this section. In Section 2.6, a hypothetical scale to correlate the generality of a cognitive architecture with its level of abstraction and the task complexity that it can feasibly cope with has been described. Independently of what paradigm is chosen as a motivation behind a new cognitive architecture, progress can be defined as adding to both generality and task complexity as compared to existing models, or at least to make an improvement in one dimension without regressing in the other (i.e., moving

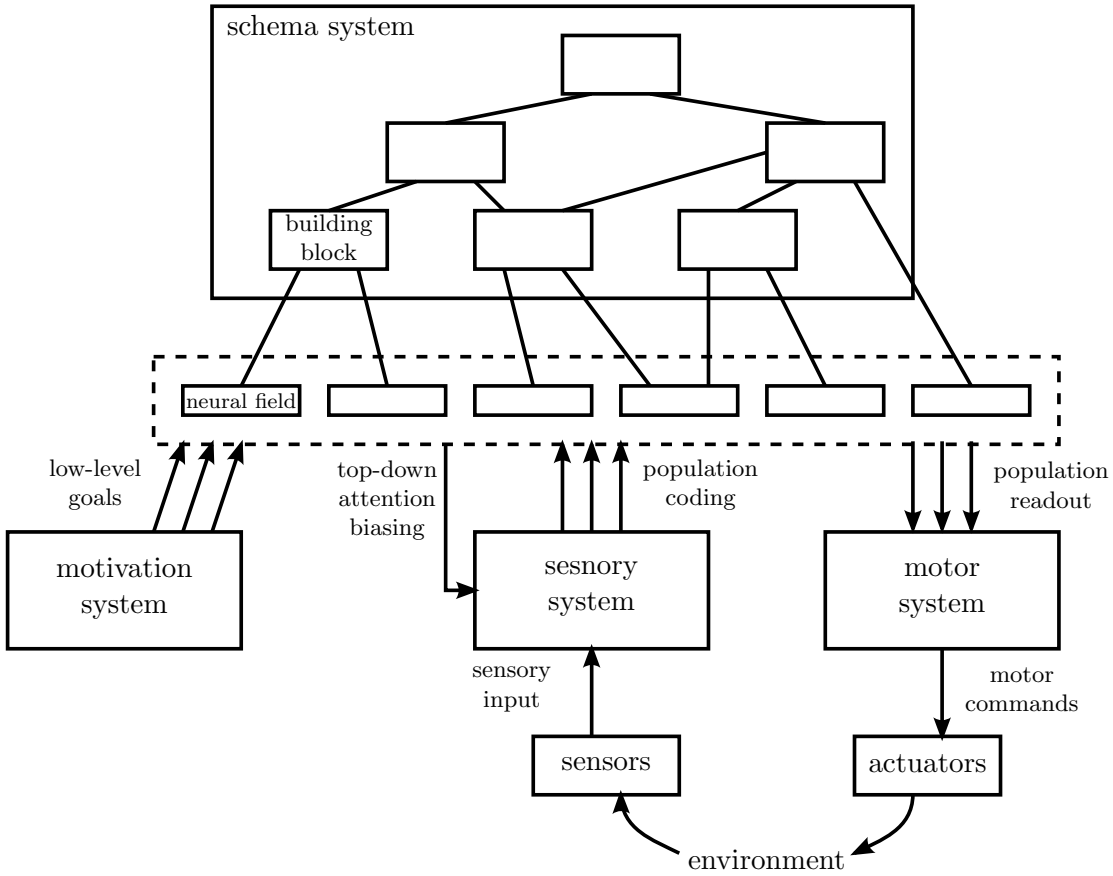


Figure 3.3: Overview of the proposed cognitive architecture.

up and to the right in Figure 2.14, so to say).

In this work, a cognitive architecture will be proposed, as an attempt to make such progress. It draws on the embodied cognition hypothesis as a motivation and is aimed at extending and improving existing approaches to cognitive architecture in the connectionist and dynamicist paradigms. For this, in the following it will be elaborated on compatibilities between, and common grounds among, the different views on embodied cognition. It will then be tried to identify general principles that can guide the design of a computational model. In the following sections, an overview of the proposed architecture will be given, and its main properties and how they relate to the picture drawn by the embodied cognition hypothesis will be described.

Figure 3.3 depicts an overview of the main components of the architecture and their interaction. The core is a *schema system*, which houses a network of structurally identical building blocks and implements the main functionality of the system, such as learning, action selection, decision making, and planning. It receives input from, and produces output for, the other systems in the architecture, which are the *motor system*, the *sensory system* and the *motivation system*.

3.3.1 The Schema System

A pervasive feature of most theories of embodied cognition is the assumption that some sort of building block is in place, which enables cognitive processing: Concepts that have been proposed in the literature are schemata (Arbib, 1998; Bartlett, 1932; Gallese and Lakoff, 2005; Piaget, 1997 [1953]), convergence-divergence zones (Damasio, 1989; Meyer and Damasio, 2009), simulators (Barsalou, 1999, 2008) and dynamic neural fields (Erlhagen and Schöner, 2002; Johnson et al., 2008; Smith and Thelen, 2003; Thelen et al., 2001), among others (e.g. Fuster, 2009). The schema system as the architecture's central component is composed of a network of structurally identical building blocks, which implement the functionality of the system, including learning, decision making and planning.

The use of a generic building block in a cognitive architecture is appealing for several reasons. From an engineering point of view, it limits the design effort for the creation of a cognitive system to the specification of connections in between building blocks and the system's sensory inputs and motor outputs. Since the internal mechanics of the building block (i.e. the mechanisms for learning and operation) are fixed by design, it would not be necessary to develop individual specialist components. Furthermore, in principle it should be possible to let the system self-organize the interconnections between building blocks in a data-driven way, which would be a possibility for open-ended learning.

In addition, also from an empirical point of view it is reasonable to assume that cognition can be implemented as a network of functionally equivalent modules, as it has been argued that the organization of the cortex is best described as composed out of uniform "columns" (repeating structures that can be found across the cortex, where the firing of neurons seems co-determined inside a column, but largely independent across neighboring columns; Mountcastle, 1997). Furthermore, while the adult cortex is found to consist of different areas that can be distinguished from one another (see Section 2.1), this layout seems to epigenetically develop to a large extent (for example as the result of afferent input) rather than be fixed by genetics, as early on in development the structure of the cortex is much more uniform and areas cannot be distinguished (O'Leary, 1989). This assumption is also supported by experimental findings, as it has been shown in ferrets that an artificial redirection of retinal projections to brain regions that normally develop into auditory areas results in the development of retinotopic maps in these regions (Melchner et al., 2000). Thus, it is not predetermined for a region what its function will be, but the same region of the neonatal cortex can develop to either process auditory information or visual information, depending on what afferent inputs it receives. This is strong evidence supporting that information processing is implemented similarly at different locations in the cortex, and consequently that specializations of cortical regions are the result of the history of information that the regions have processed.

Also some of the existing cognitive architectures make use of a generic building block: The ERA as a connectionist architecture proposes connected SOMs and a mechanism of spreading activation as the "ERA unit" (see Section 2.4.1), and the architec-

tures based on the dynamic field theory rely on the dynamic neural field as a building block (see Section 2.5.2). However, as already noted in the discussions of these architectures, both have their limitations: The mechanism of spreading activation used in the ERA unit seems not suitable for many tasks, for example it remains unaddressed how motor control can be reliably achieved, and in a larger scenario where more than just a few associations need to be learned by the networks, an over-activation of network units across the whole system can be expected due to the unconstrained nature of the propagation rule that is used. The dynamic neural field offers as a building block important functionalities, such as local decision making between competing hypotheses, but current computational models rely on hand-specified connections between fields, and learning is not addressed.

Building blocks in the schema system of the here proposed architecture are modeled after the idea that the concepts of simulators and dynamic neural fields may be combined in a coherent manner, supported by the structural framework provided by the convergence-divergence model, as it will now be described.

The simulator is an appealing concept as a candidate for a building block, as Barsalou's theory of perceptual symbol systems, which is based on the concept, provides an account for many aspects of cognition, ranging from low-level attention and categorization to language and other high-level capabilities (see Section 3.1.2; Barsalou, 1999). However, as compared for example to the detailed description of dynamic neural fields, it remains rather unspecific concerning the processes underlying the account, for example how frames are produced, how they are learned or how complex simulations are generated. Barsalou's descriptions, as he admits himself, leave many questions open for further consideration, but are rather meant to give a general idea about the processes he imagines to be involved (Barsalou, 1999, pp. 582, 590). Barsalou also does not specify, how a simulator selects specific simulations that are produced, given that it can supposedly produce infinitely many different simulations.

The dynamic field theory on the other hand is formulated in great detail, but remains rather specific about the processes that are modeled, which are mainly the integration of sensorimotor information and decision making. If we assume that both simulators and the processes underlying the dynamic field theory are in place, we could argue that the same mechanisms that are responsible for the decision making in motor tasks (cf. Section 2.5.1) could also be accounted for making the decision of which specific simulation to produce. In this interpretation, simulators, located in association regions (or convergence-divergence zones, cf. Section 3.1.1), would produce input via top-down cortical connections to the dynamic neural fields in sensorimotor regions, corresponding to pre-activations of neurons for possible simulations. The field dynamics could then spontaneously select one specific simulation, possibly in the light of multiple simultaneous top-down activations (i.e., other simulations) or current afferent input (see Figure 3.4 for a schematic representation of this idea). For example, an *apple*-simulator might be located in an association area where, among others, visual shape, color and size properties are combined. The *apple*-simulator would be rather specific about the shape property, but some variability in the size and color domains

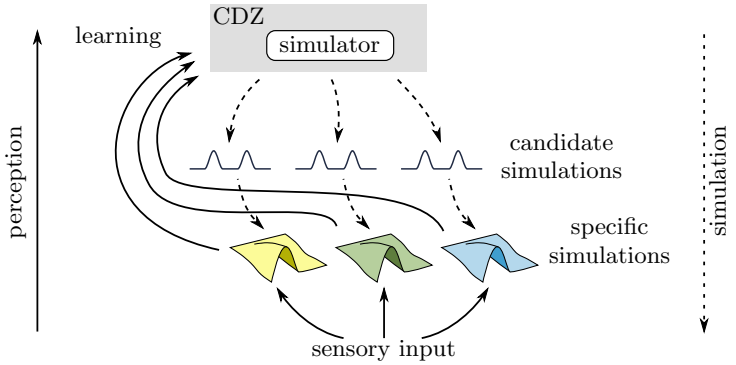


Figure 3.4: Schematic representation of how the dynamic field theory and embodied simulation may combine. Dynamic neural fields form the basis of representation and receive sensory input during perception (solid arrows). When sensory input is present, information is passed on to convergence-divergence zones (CDZs), where correlations in the input are learned and simulators are formed. During simulation (dashed arrows), simulators produce candidate simulations in the form of activation landscapes as input for the neural fields. The activation dynamics in the neural fields spontaneously develops single peak activations which correspond to the specific simulations. The decision dynamics can be influenced by the concurrent activation of several simulators, or by sensory cues that produce bottom-up influence.

would be possible. When a simulation is produced, different neurons in the dynamic neural fields representing the shape, color and size properties would be activated via the top-down connections of the simulator. A specific simulation of an apple would be the result of the field dynamics selecting one of the alternatives, for example due to priming or memory effects (cf. Section 2.5.1) or neural noise.

If we extend this idea by further assuming that simulators become arranged by self-organization in topographical maps, we can imagine a generic connectivity pattern between simulators and dynamic neural fields to exist throughout the cortical hierarchy. An example for such self-organization is the structural organization of action representations in the primary motor cortex and the premotor cortex: Whereas the primary motor cortex is involved in performing primitive actions, such as flexing and extending the fingers, turning the wrist, flexing and extending the elbow, and so on (Gallese, 2008), the premotor cortex structures these primitives into more complex, coordinated, goal-directed actions, for example bringing the hand to the mouth, or bringing the hand to specific points in ego-centric space, independent of the initial situation (Graziano et al., 2002). Thus, in the primary motor cortex, the somatotopic arrangement in neural fields can lend its topography directly from the actual layout of the body. However, the topographical arrangement of the representation of goal-directed movements in premotor cortex needs to form in a process of self-organization. Here, neurons for example responsible for reaching with the hand towards locations in the workspace are arranged so as to form ego-centric maps, where the distance between neurons resembles the distance between target locations (Graziano et al., 2002). The relation between

movements in the joints and the positions of the hand that are obtained is non-linear and not trivial, so the topography of this representation likely needs to be learned by integrating sensory feedback on the hand location. Thus, if we assume that an internal model, i.e. a simulator, is responsible for the control of these goal-directed movements and the association of the motor representation with sensory representations, then this internal model (or the internal models if more than one are involved) provides a new topographically organized representation, which makes information available to higher levels in the hierarchy in a more compact format. It reliably represents the action of reaching for some location in ego-centric coordinates, without the specific details of how to reach for that location on the level of which joints are involved, and how they are moved. It is not unreasonable to assume that similar self-organizing representations can also be found in other modalities. For example, it has been proposed by Haxby et al. that an "object form" topography can be found upstream the visual "what" pathway in the ventral temporal cortex, which presumably reflects how the more complex attributes of visual appearance of objects and faces are related visually, structurally, or semantically (Haxby et al., 2001).

Thus, topographically arranged representations can be found on higher levels of the cortical hierarchy just as in the primary cortical regions, and topographic distances reflect more and more complex and multi-modal relationships the higher the representation is in the hierarchy. What is important here is that also the high-level representations can be represented in topographically arranged neural fields, where the lateral connectivity could implement the integration of information and decision making as proposed by the dynamic field theory. This allows to propose a view, in which a cascade of building blocks composed of simulators and dynamic neural fields would be at work, with higher-level simulators generating coarse simulations which are then refined more and more as the simulation propagates down the hierarchy. For example, a simulator on the "object form" level could generate coarse multi-modal candidate simulations of a car, from which the associated dynamic neural fields select one candidate through dynamic competition (for example a compact car instead of a van). The information about this selection is propagated downward to the next lower hierarchical level, where simulators and dynamic fields appropriately fill in more details (for example about the shape). In this way, the simulation is propagated downwards in the hierarchy, and simulations on higher levels of the hierarchy dynamically constrain or select the simulators on lower levels, a process which is repeated in a cascade until the sensorimotor regions of the cortex are reached.

Conversely, during the processing of bottom-up information, simulators become activated to a degree that reflects how well their simulations match the available data. This is in line with Barsalou's view that simulation is the process behind categorization, as stimuli are categorized according to a measure of similarity with simulations (see Section 3.1.2; Barsalou, 1999). It also resembles the activation mechanism behind the MOSAIC model (see Section 3.2.2), which has been used by its authors to explain data from psychophysical studies (Wolpert and Kawato, 1998; Wolpert et al., 1998). As the simulators have become topographically organized in a neural field, dynamic compe-

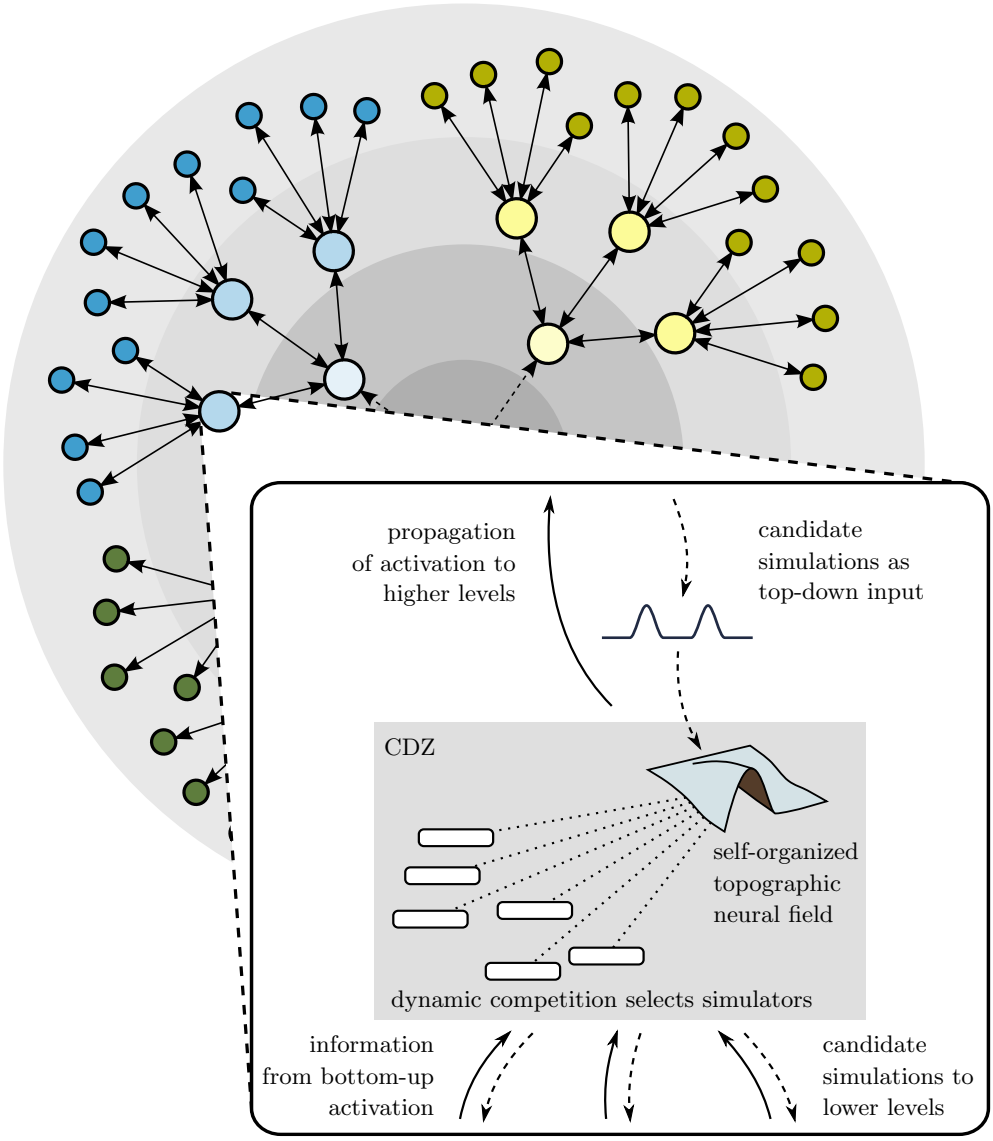


Figure 3.5: The proposed view on the integration of simulators and dynamic neural fields in a framework for embodied cognition. In a process of self-organization, simulators become arranged in a topographically organized dynamic neural field. In the face of bottom-up activation, simulators become activated in a way reflecting the degree to which their simulations match the current bottom-up information. Dynamic neural competition selects the winning simulators and inhibits all others. The winning simulators are adapted to the new information in a process of learning. Also, the activation of the winning simulators is propagated to higher hierarchical levels. In the process of simulation, top-down connections propagate activation as candidate simulations. The field dynamics selects from the candidates the winning simulators, which then produce candidate simulations for lower levels in the hierarchy.

tition selects among them the simulators that best describe the input, as it has the highest input activation. In the process of continuous adaptation to new information, the winning simulators are trained to incorporate the new information (or, in Piaget's terms, accommodate the input). Also, the activation of the winning simulators is further propagated to higher hierarchical levels where the process repeats analogously. Figure 3.5 is an adaptation of Figure 3.1 and summarizes the structure and flow of information as proposed by this view.

The use of a generic building block in a system requires the definition of a common data format, which is used to exchange information between the building blocks (Gepperth et al., 2008). As mentioned above, it is assumed here that topographically organized neural maps represent information at all levels of the cortical hierarchy. In the proposed architecture, and borrowed from the dynamic field theory, information therefore is passed in between building blocks using population coding in multidimensional neural maps: Each neuron in a neural map responds selectively for a certain stimulus with a bell-shaped tuning curve (i.e. each neuron has a preferred stimulus to which it responds strongest, and activation decreases the more the input deviates from the preferred stimulus). Originally identified in visual cortex (Hubel and Wiesel, 1962), population coding neurons are known to be used everywhere in the cortex and their processing has been demonstrated to be robust against noise (see Deneve et al., 2001). In contrast to vector coding, which is typically used in the robotics literature, population coding in neural fields allows for a simultaneous representation of multiple values at the same time, and the representation of no value, which both will be a crucial feature for the integration of building blocks, as it will be discussed below and in detail in Chapter 4.

The proposed view of information processing in a network of generic building blocks offers to understand the concept of schema in a similar, but refined way as compared to Rumelhart et al.'s description (Rumelhart et al., 1986): As a dynamical system that is attracted by a state in which as many constraints as possible are satisfied. However, while (Rumelhart et al., 1986)'s model is limited to a selection of binary features (cf. Section 3.2.1), the here proposed view provides a tangible interpretation of how a schema representation with slots for attributes and their values can be found in a neurodynamic model: Neural fields throughout the cognitive system represent domains of values, initially in modality-specific domains (such as color, line orientation, pitch, amplitude of force, etc.), and later on in more abstract domains (such as an "object form" topography, Haxby et al., 2001). Building blocks learn about the combinatorial arrangement of input values that occurred synchronously (Damasio, 1989). As it will be described in Chapter 4, the building blocks can restore information about what values in other domains correspond to given cues. As the building blocks of the schema system connect to a set of topographic neural fields, they define the "slots" for attributes in their representation: Each connected neural field can be "filled" with a value by means of an embodied simulation, be it in a modality-specific representation (such as color, amplitude, etc.), or be it in abstract cross-modal representations (such as a particular "object-form", which itself would trigger a further embodied simulation

in connected lower-level representations, as described above). As it will be demonstrated later on in Section 4.2, multiple building blocks can collaborate in producing a specific embodied simulation, by mutually constraining their respective decisions. In this way, the retrieval of information based on schematic knowledge can be understood as a distributed process of refinement of incomplete information, as the building blocks collaboratively contribute to the decisions of which simulations to produce in the sensorimotor representations of the system. As the dynamic neural field representations become interconnected via the network of building blocks, their individual decisions become co-dependent, and settle to a state in which the constraints that the system has learned in its schema knowledge are satisfied as much as possible.

3.3.2 The Motor-, Sensory- and Motivation Systems

Information is passed between components of the architecture in the form of population codes in neural fields, both inside the schema system in between building blocks, as well as between the schema system and the other systems. A set of neural fields is defined at the interface between the schema system and the other systems (see Figure 3.3). All of these fields represent unimodal inputs and outputs in domain inherent metrics, as for example retinal location, color, shape or pitch for inputs, or joint spaces for outputs.

The motor system implements low-level motor functions, such as control of the effectors in joint space. The sensory system provides input to the schema system in the form of neural fields that reflect topographies inherent to the sensory domains. Note that values are represented in the neural fields in the form of activations landscapes, thus a single value is represented as a single-peak activation, but also multiple values can be represented simultaneously in a multi-peak activation. In some cases, the input to the fields is naturally restricted to always be in the form of a single-peak activation, as for example in the case of joint positions (the robot cannot have several arm configurations for the same arm at the same time). In other cases however, the input can be in the form of an activation landscape, as for example in the case of a field representing salient image locations.

The motivation system produces goals for the agent, in the form of target values that should be reached for specific inputs. In a more complex agent, the motivation system could be assumed to monitor bodily states, such as energy levels or maybe rewards and pain. However, as the focus of the work in this thesis lies on the definition of the building blocks in the schema system and their underlying mechanics, it is simply assumed that the motivation system selects a target for some input, which the agent should try to achieve.

3.3.3 Mechanics of the Building Blocks

Given the above considerations, the following modeling constraints can be formulated for the building blocks of the schema system:

- Information is passed between building blocks via topographically organized dynamic neural fields.

- Each building block receives as input the activation of several neural fields, based on which it learns a model (or simulator), which codes for the information of which combinations of activation patterns across the neural fields are valid.
- New topographies in neural fields on higher hierarchical levels are formed through self-organization, providing representations of reduced dimensionality.
- Building blocks compete against each other in explaining sensory data; inputs are compared with predictive simulations, and building blocks receive activation according to the goodness of match for accommodating the new information.
- The models learned by the building blocks are employed by the system to produce a plan of how to act to reach a goal.

The last point is especially relevant to explore for a robot cognitive architecture, as it relates to the important question of how motor responses are generated based on the system's goals and afferent inputs. As mentioned above, it is assumed that a motivation system provides a goal input, which has the form of target values for sensory variables. In the following, examples will be given for how the structural constraints of building blocks just described, together with the functionality provided by dynamic neural fields, can let a robot accomplish its goals. While in-depth descriptions of the internal mechanics of the building blocks will follow in the next chapters, especially concerning how internal models are learned and how they are used for the control of the robot's movements, this section should provide an overview of how the overall architecture is envisioned to work.

In using embodied simulation as a means to reach goals, the architecture's mechanics is following the empirically supported view that overt action is preceded by a mental representation of a goal-state, that is, a successfully terminated action pattern (see Gallese and Metzinger, 2003; Herbort and Butz, 2012): Given a goal (i.e. a target value for some input), the schema system uses its stored associations to reconstruct a simulation of a situation in which that goal is met. Actions are then chosen in the attempt to achieve that situation. As opposed to the existing computational models of embodied simulation described in Section 3.2.2, this does not equate to a short-term anticipation of a future state along a trajectory of sensorimotor states, which is bound to originate from the current state. Instead, it allows the agent to simulate arbitrary situations where the robot's goals are met, based on what it has learned about the world, although the current context can be used to influence this process, so as to favor more feasible solutions.

Example 1

As a first example, consider the case of a robot equipped with an arm and a manipulator as shown in Figure 3.6. For it to be able to bring its manipulator to a target position, as for example observed via a camera sensor mounted to the robot's base, it requires the ability to transform between positions in Cartesian coordinates, or camera

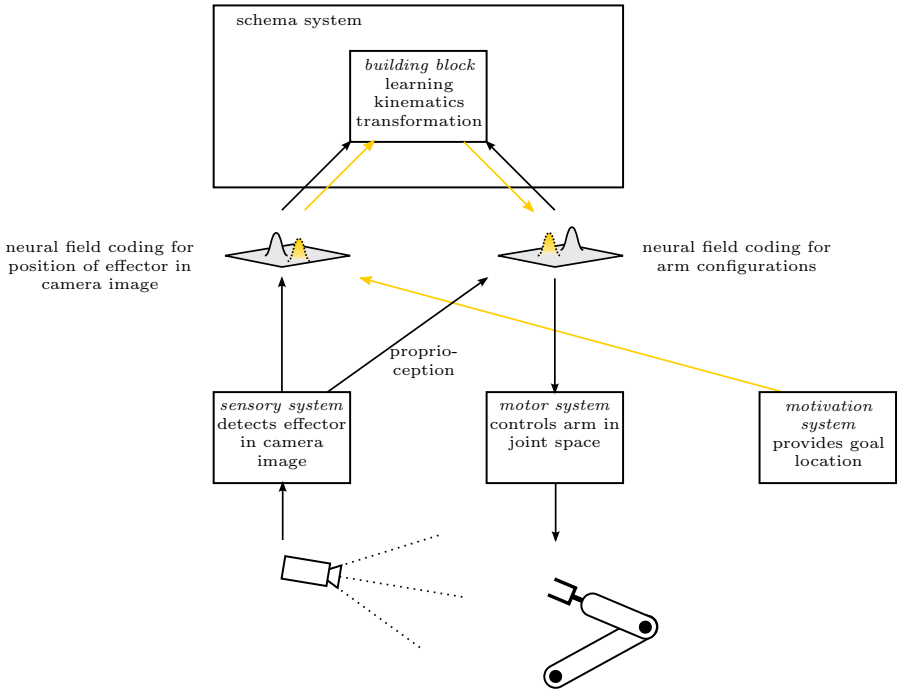


Figure 3.6: Example for a system with only a single building block in its schema system, for the learning of the kinematics transformation. Black arrows show the flow of activation originating from sensory information: As the robot moves its arm, the neural fields are activated to represent the arm posture in joint space, and the position of the effector in image coordinates, respectively. The building block learns to associate co-activated neurons in the two fields. Yellow arrows show the flow of activation as the motivation system provides a target location for the effector. It is transformed via the learned associations into a posture, which is sent to the motor system through population readout.

coordinates, and positions in angular coordinates (known in robotics as the kinematics transformation). This transformation is performed by a building block connecting the two inputs and outputs: A neural field representing the angular position space of the robot arm and a neural field representing the Cartesian positions of the manipulator. The former is the interface to the motor system, which reads out values from activations in the field and translates them into motor commands that are sent to a controller. The latter is the interface to the sensory system, which in this case is assumed to detect the position of the robot's manipulator in a camera image and provide it to the schema system as an activation peak in the neural field (e.g., a peak in the center of the neural field corresponds to the effector being in the center of the camera image).

From experience (i.e. the moving around of the effector), the system learns the valid associations between the activations in the two fields, thus acquiring the kinematic transformation inside the building block. The system could then be given the goal to bring its manipulator to a certain position, which would be represented by a goal activation in the neural field for Cartesian coordinates (see Figure 3.6). Thus, to

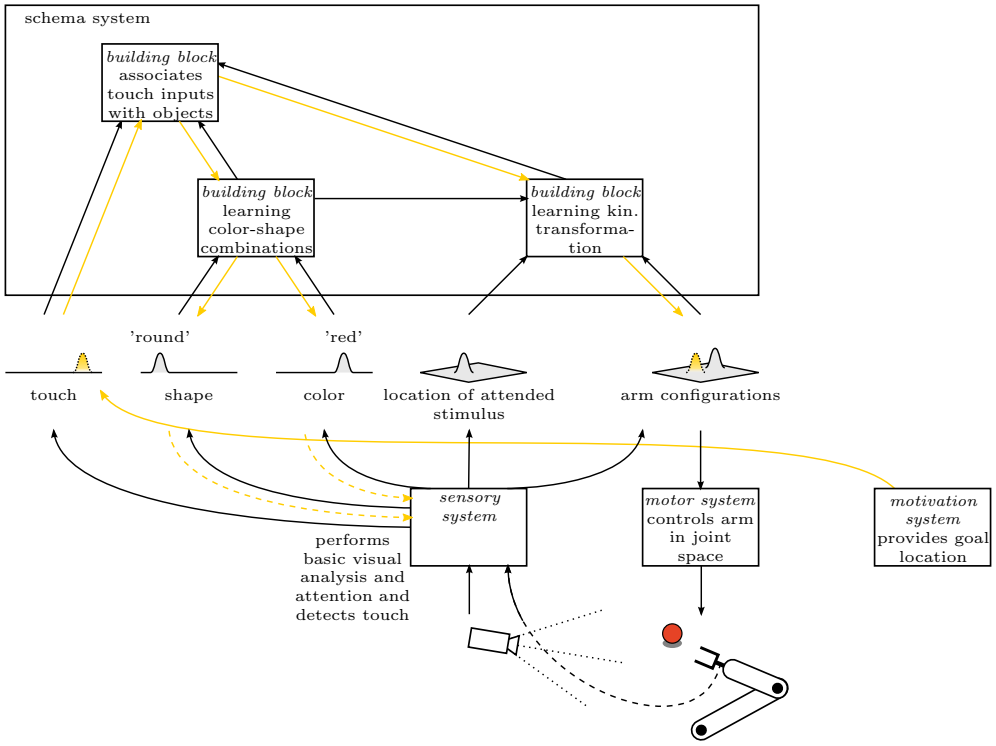


Figure 3.7: Overview of the system described in the second example. Here, activation from the target value in the touch domain is first propagated to the building block that has learned about color-shape combinations of objects, which in turn triggers a simulation of a matching object in the shape and color fields. This activation is used by the sensory system as top-down attentional cues (dashed yellow arrows). In turn, the location of the object is activated, which is transformed into an arm configuration that lets the effector touch the object.

reach this goal, the system needs to determine an arm configuration that will move the effector such that the input coming from the sensory system matches the goal activation in the neural field. Via the associations learned by the building block, the system reconstructs a situation in which this is the case, producing candidate activations in the arm configuration field. The field dynamics then selects one configuration, which is sent to the motor system and effectuated by the arm.

Example 2

In the simple example above with only a single building block in the schema system, the goal of the robot is achieved through a single association between an input and an output. However, the same picture of how a goal is propagated from one neural field via learned associations in the building blocks across to other neural fields, extends to more complex examples. Assume a system as shown in Figure 3.7, that has the capabilities for basic low-level visual analysis and attention mechanisms. It represents

image regions in a shape and a color domain, and can tune its bottom-up attention process to favor specific colors or shapes. Visual analysis maps sensory inputs into neural fields for the color domain and the shape domain, and a visual location field reflects the location of the currently attended region in the camera image (for example, if a red ball in the camera image is visually attended, then neurons corresponding to *red* in the color field and *round* in the shape field are active, as well as the neurons representing the image region in the location field). It also has a touch sensor in its manipulator that can distinguish textures.

With three building blocks in its schema system (see Figure 3.7), the robot can learn several things about its world and utilize this knowledge to achieve its goal in the following way. As it visually inspects different objects, the building block connecting the shape and color property fields can learn that several different activations of shapes and colors appear in combination, which results in a higher-level neural field representation for objects (i.e., color-shape combinations). As in the first example, a building block associating the arm configuration field and the visual location field can learn the kinematic transformation, but in this case it will also learn to associate the visual appearance of its own effector. Finally, the third building block can learn that when a certain object is attended and a reaching action for its location is effectuated, this results in the perception of a specific input to the touch sensor, as the robot's effector touches the attended object. For the moment it is not important how the associations are learned or how they are stored inside the building blocks (which will both be covered in Chapter 4). It is only important that a kind of "pattern completion" is possible: When a partial cue, that is, an activation in a subset of the connected neural fields, is present, then the building block can retrieve possible values for the missing inputs.

Now assume that the agent has (for whatever reason) the goal to have a sensation of a specific touch. This goal cannot be simply associated to a specific bodily state (such as a fixed arm posture as in the last example), as the agent first needs to come up with the visual appearance of an object with the desired texture and then look for such an object and reach for it. Thus, to achieve its goal, the agent needs to use its knowledge to translate between a target value specified in one modality (a specific touching sensation) into an appropriate value in another modality (the control variable for arm configurations, to bring the manipulator to the location of an object, once it has been found), involving several learned associations. The behavior of the system in this case, as it is envisioned in the proposed architecture, can be described as follows. As the goal in the touch input is associated with certain objects via one of the building blocks, the goal activation spreads to the neural field of object representations where the corresponding neurons become active, which results in a simulation of the object in the color and shape fields. This extends the original goal, that of perceiving a certain touch input, to the visual perception of an associated object. The representation of this goal in the color and shape feature maps can be used by the attention system to tune the attention process, such that a matching object would become subject to visual attention, if one is in the field of view. This in turn provides information on the

location of the attended object in the location map. At the same time, as the action of bringing the effector to a target location is also associated with the original goal, the location of the attended stimulus becomes transformed via the kinematics building block into a simulation of candidate arm postures to reach for the object.

In sum, the system would come up with a simulation of a situation in which it is touching with its effector an object, which it had associated with the target tactile stimulus. If such an object was in the robot's field of view, this would trigger a transformation of the generic simulation of seeing and touching the object into the simulation of the specific goal situation for the current context, in which the robot had moved its effector to the position of the visible object. As the activation across the neural fields settled, the motor system would readout the arm posture information from the neural field, and the robot would drive its arm to that posture, where its effector would come into contact with the object and its goal would be reached.

3.3.4 Network Layout in the Schema System

It is important to note that the network layout in the schema system should not be constrained to be a strict hierarchy, in the sense that it would need to be a tree-like structure. Instead, building blocks can arbitrarily connect neural fields across the whole schema system. Thus, in the language of graph theory, a node can have multiple parents. Hierarchical levels can only be defined in terms of the distance to the interface neural maps.

An example where one neural map is connected to several building blocks is a system that associates different sensory inputs to the same motor output. For example, the position of the effector could be associated to the angular configuration of the whole arm, while individual sensors could report the "goodness" of the positions of individual joints, so as to let the system be able to avoid joint limits. In this layout, output maps for the individual arm joints would be connected each to one building block for this joint position sensor, as well as to the building block for the kinematics transformation (see Figure 3.8). This system could have multiple goals at the same time, for example those of bringing the effector to a target position on the one hand, and keeping joints away from limits on the other hand (or even having a joint in its limit, which would be an unreasonable, yet possible goal). Both goals would be mapped via the building blocks onto the same motor output fields. Although in some cases the two goals might coincide in one particular arm configuration, this is unlikely for most cases. Thus, there are conflicting candidate outputs originating from different goals. Since it is avoided in the proposed architecture to rely on supervisory components to determine an optimal solution based on hand-designed heuristics, as for example in hybrid cognitivist architectures (see Section 2.2.2), a locally operating competition needs to be implemented to select an action to be executed. In Chapter 4, a mechanism will be demonstrated that can be accounted for this distributed selection behavior.

Another problematic situation, which is the result of the very generic layout of the network of building blocks in the schema system, is related to the problem of relevance detection, which was mentioned earlier (see Sections 2.2.3 and 2.4.2): Several building

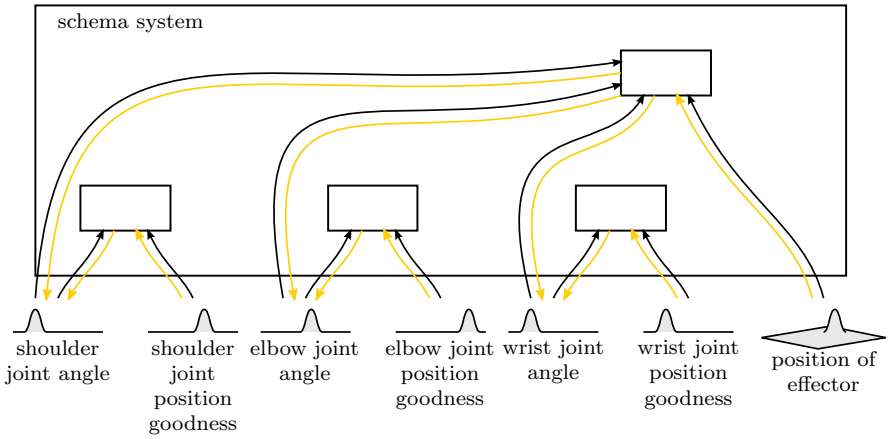


Figure 3.8: An example setup of a schema system, in which multiple building blocks connect to the same fields for motor variables. In such a case, multiple goals can result in conflicting candidate postures. A coherent solution needs to be found through a distributed process of local decision making, since the use of a supervisory component that implements task-specific selection heuristics is avoided.

blocks can use the activation pattern in the same neural field as input, but only a subset of the activation patterns actually corresponds to *relevant* inputs for the individual building blocks. For example, it is often assumed in the literature that a robot can autonomously learn the kinematics mapping by repeatedly moving its arm into different postures while observing the resultant position of the effector. As briefly outlined in the above examples, and covered in detail in Chapter 4, a building block in the schema system can obtain the kinematics transformation by using such a set of observations as training data. However, the building block will only be able to successfully learn the mapping with a reasonable performance, if it only trains on examples from situations in which the robot actually looks at its own effector, and not at some random object in the visual background scene (cf. Section 2.4.2). In the literature it is usually implicitly assumed that the training examples only stem from valid situations. Chapter 5 will elaborate on this problem further and will provide a solution, which allows the building blocks to implement a local competition for input signals, such that inputs are correctly distributed among the building blocks and allows them to successfully learn internal models, without relying on such an implicit assumption.

3.4 Discussion

This chapter was divided into two main parts. In the first part, the embodied cognition hypothesis and related theories were introduced, which provide a promising new paradigm for cognition in the current cognitive sciences literature and is intended as an alternative to the cognitivist view. In many ways, this new school of theories compares to more traditional theories that are based on the concept of schema, as highlighted in the discussion of these theories. Several computational models of the concepts exist, as

then described.

In the second part, motivated by the body of embodied cognition theories thus far outlined, a new cognitive architecture based on embodied simulation as a central mechanism has been proposed. It uses a schema system as its central component, which is composed of a network of generic building blocks and implements the main functionality of the architecture. It is based on an attempt to coherently combine views from several theories of embodied cognition and schemata, and proposes an alternating cascade of internal models (or simulators) and dynamic neural fields.

Although so far the description of the architecture was on an abstract level, leaving details about the implementation open for the descriptions to follow in the next chapters, so much can already be said in relating the architecture to the existing approaches that were described in Chapter 2: As it is motivated by theories of embodied cognition, it shares most similarities with cognitive architectures from the connectionist and dynamicist paradigms. Similarly as in ERA (see Section 2.4.1; Morse et al., 2010), it is argued that a generic building block should be used in an emergentist cognitive architecture instead of fixing the layout on the architectural level of the system by design. The use of a building block introduces a level of abstraction for the modeling which is above that of the simple processing units in standard connectionist models, thus allowing to tackle more complex problems, but retains the same separation of mechanisms from assumptions about the content of the information to be processed: It models generic mechanisms that are the same across the whole system, independently of for example which modality the information comes from that is being processed. The sole cause determining what an individual building block comes to represent is the set of neural fields that it connects to and the history of information that it has processed. As opposed to ERA, the here proposed architecture introduces the important mechanism of local competitive decision making in dynamic neural fields, through which information is passed between building blocks. This way, the spreading of activation in the system, as it uses the learned internal models to determine actions to pursue, is limited to the simulation of a coherent sensorimotor situation and an over-activation across the whole system is prevented. Importantly, the architecture addresses issues that arise as a consequence of using a generic building block in a cognitive architecture, which have not been solved in other cognitive architectures so far. These include the issues mentioned above of combining the outputs of multiple internal models to form motor commands that solve multiple tasks simultaneously, and of the unsupervised learning of different sensorimotor mappings from information coming from a single sensory modality. These issues will be discussed in detail, and solutions presented, in the following chapters.

Embodied simulation is a central process in the proposed architecture, as the simulation of sensorimotor situations, in which the system's goals are met, precedes the execution of overt motor actions, as a form of attractor sensorimotor state that the system strives to reach. As described in Sections 3.1.2 and 3.1.3, in the Psychology literature it is argued that representations supporting embodied simulation are organized around a *frame*-structure (Barsalou, 1999; Gallese and Lakoff, 2005; Rumelhart, 1984), which in some way defines a set of attributes and the range of possible values for these.

In the proposed architecture, a similar view on frames is assumed as the computational models of schemata that are based on a process of global constraint satisfaction: These models propose to encode the knowledge of the system in a set of associations in a network of localist representations, and the processing of the system to be based on a mechanism of relaxation to a state in which as many of the associations are satisfied as possible (see Section 3.2.1). Similarly, processing in the proposed architecture uses the learned associations in the building blocks to propagate a goal, which is specified in one modality, across to representations in other modalities. The decision of which specific sensorimotor situation will be simulated is based on a distributed process across multiple dynamic neural fields, and is contingent on the involved associations. This can be said to be a form of constraint satisfaction, but not relying on a localist representation.

Finally, it is also noteworthy that basing a cognitive architecture on the concept of embodied simulation makes it compatible with theories of how the understanding of observed actions of others is implemented in the brain (cf. Sections 3.1.2, 3.1.3–3.1.4). According to the embodied simulation accounts to action understanding, an observation of someone else's action automatically triggers an embodied simulation of the action using sensorimotor representations (Gallese, 2003). This is in contrast to a more cognitivist view, according to which social interaction involves the decoding of streams of sensory information and mapping them onto an abstract representation of beliefs, desires and intentions of the observed person. Instead, sensorimotor representations that have been learned for the control of the own actions are activated not only for the own performance of an action, but also when seing someone else perform the same or a similar action. As briefly stated in the examples in Section 3.3.3 (and explained in more detail in the following chapters), building blocks in the schema system of the proposed architecture learn to associate the robot's actions with the sensory feedback corresponding to the outcome of the action. During the robot's goal-directed action planning and execution, this knowledge is used to find possible actions that will produce a desired state, by first expanding an initial goal activation with associated sensorimotor states, until a decisive simulation of a sensorimotor situation is achieve, including an activation of the motor program for producing the situation. In complying with the embodied simulation account for social interaction, essentially similar functions of the same representations should underly the understanding of another's actions: When watching someone else's actions, sensory observations that have previously been associated with the own action performance can be mapped by the learned sensorimotor representations in the building blocks onto the motor states corresponding to the actions of the other. Thus, while during the own goal-directed action execution a goal activation (triggered by the motivation system) corresponding to a desired sensorimotor situation is mapped onto an activation of motor representations to produce this situation, during observation of other's actions the same mapping would produce an embodied simulation of the action execution using the own action representations. This property of a the schema-based representation used in the proposed architecture is potentially beneficial for addressing the question of how to let a robot learn from a tutor's demonstration in social interaction. This issue however needs further attention, which is out of the scope of this thesis.

4

Integration of Internal Models by Making Use of Redundancies

Internal models[1] are discussed in the robotics literature as essential tools, for example for movement control, such as kinematics and dynamics models of the robot's own body (Nguyen-Tuong and Peters, 2011). In mathematical terms, they describe mappings between the robot's motor space and a sensory feedback space. In the context of cognitive architecture, as well as in theories of embodied cognition, the term describes a more general concept, as it describes for example the mechanisms underlying the process of embodied simulation (cf. Section 3.1.2). In that sense, internal models describe knowledge structures that capture the relation between a set of representations, not only for sensor-to-motor transformations as in the more specific usage in the robotics literature, but for any set of representations in the cognitive system.

In the last chapter it was proposed that a cognitive architecture should be composed of generic building blocks. These building blocks should learn about patterns that occur in their inputs and outputs, and thus implement internal models. The operation of the system is envisioned to be the result of the collaboration of its building blocks, each performing dynamic local computations, but in concert producing a coherent system response. This principle of organization of a cognitive system yields a fundamental question that any cognitive architecture based on the use of building blocks needs to address: How are the outputs of multiple internal models integrated, for example when several internal models produce candidate values for the same motor output? A simple example scenario where this question needs to be addressed was already outlined at the end of the last chapter (see Section 3.3.4): A robot that learns to associate its postural configurations with several different sensory feedback signals, such as the Cartesian position of the end-effector on the one hand, and a proprioceptive feedback about the goodness of joint positions on the other hand. The robot could be given multiple ambiguous tasks, for example to bring the end-effector to a certain position

[1] In the robotics literature, more commonly simply the term "model" is used. In this work however, the more specific term "internal model" is used to avoid confusion with other terms in the cross-disciplinary literature, such as "model of cognition", etc.

in Cartesian coordinates, and to avoid certain joint positions for the individual joints. As several internal models would each represent a mapping of a target sensory input space onto the same motor output space, each of them produces different output values. The system thus needs to somehow select one of the candidate values to actually send to the motor controllers, preferably one that satisfies multiple of the robot's tasks simultaneously. Furthermore, as this work is aimed at investigating ways to *generically* integrate building blocks in a cognitive architecture, the solution should be free from any task-specific considerations.

This chapter will first give an overview of existing methods for the integration of internal models in Section 4.1, after which a new method will be proposed in Section 4.2, which makes use of the existence of redundant solutions in many tasks. The method is not tied to a specific choice for the representation of internal models, but a concrete implementation using a specific neural network architecture will be described in Section 4.3, and tested using a simulation of the humanoid robot iCub, as will be described in Section 4.4.

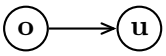
Parts of this chapter (in particular Sections 4.2.2, 4.3.1, 4.3.4 and 4.4) are based on (Hemion et al., 2012).

4.1 Integration of Internal Models in Robotics

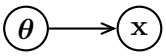
As stated above, internal models describe a mapping between an input domain and an output domain, where for example in the case of a kinematics model, the input domain would correspond to the angular position space of the robot and the output domain would correspond to Cartesian space. Depending on the application, the exact definition of this mapping can vary. The following is a brief summary of the most common types of internal models from the robotics literature. Figure 4.1 also summarizes the different definitions as graph representations.

- **Behaviors** (cf. Section 2.3) are used to implement sensory-driven motor reactions of the robot by mapping a sensory input onto a motor response (Brooks, 1986; Pfeifer and Scheier, 1994), see Figure 4.1(a). Behaviors are either hand-specified by the designer of the robot, or can be learned using optimization techniques, such as evolutionary algorithms (Nolfi and Floreano, 2000), or reinforcement learning methods (Sutton and Barto, 1998). In the case of behaviors, the input domain of the internal model is the space of possible observations for a certain sensory input, for example the space of possible measurements from a laser range finder, and the output domain is the space of commands for a certain motor controller of the robot, for example for controlling the turning speed of the two main wheels of a mobile robot.

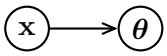
- **Kinematics models** describe the causal relation between body configurations and positions of the robot's parts in space. The input domain of a kinematics model is the space of angular configurations of the robot, and the output domain is either the Cartesian space of positions, or the combined space of positions



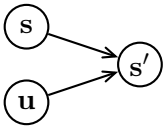
(a) Behaviors, mapping sensory observations $\mathbf{o}$ onto actions $\mathbf{u}$.



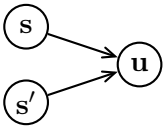
(b) Kinematics forward model, mapping joint configurations $\boldsymbol{\theta}$ onto positions in space $\mathbf{x}$.



(c) Kinematics inverse model, mapping target positions in space $\mathbf{x}$ onto joint configurations $\boldsymbol{\theta}$.



(d) Dynamics forward model (observable case), mapping current state $\mathbf{s}$ and action $\mathbf{u}$ onto a predicted future state $\mathbf{s}'$.



(e) Dynamics inverse model (observable case), mapping current state $\mathbf{s}$ and desired future state $\mathbf{s}'$ onto action $\mathbf{u}$.

Figure 4.1: Overview of the most common forms of internal models in the robotics literature

and orientations. Kinematics models are further divided into forward models and inverse models. Forward models map angular configurations onto resulting positions in space, see Figure 4.1(b), while inverse models are used to obtain a joint configuration that will bring the robot's parts to target positions in space, see Figure 4.1(c). Since each angular configuration of the robot corresponds to exactly one position for each part in space, the forward model is a well-defined function and can be learned using function approximation methods, such as feedforward neural networks, whereas the inverse model can be multivalued and thus requires further attention when being learned using standard machine learning techniques (D'Souza et al., 2001; Jordan and Rumelhart, 1992).

- **Dynamics models** are similar to kinematics models, but describe the relation of forward and inverse models in the case of dynamical systems (Nguyen-Tuong and Peters, 2011). Here, the mapping describes the relation between the robot's actions and the resulting state of the system. In contrast to kinematics models, the outcome of an action not only depends on the action itself but also on the current state of the system. Thus, the output domain of dynamics models corresponds to the space of possible system states, and the input domain additionally covers the space of the robot's actions. In the case of dynamics models, forward models predict a future state of the system (usually the immediate next state in a time series of discrete measurements) when a given action is effected, see Figure 4.1(d), and inverse models determine an action to reach a transition from the current state to a desired future state, see Figure 4.1(e). In many cases in robotics, especially in the control of the own body, it can be assumed that the state of the system can be directly measured via the robot's sensors, whereas in the more general case it has to be assumed that the actual state variable is not

readily available to the robot's sensors, but must be deduced from other sensory observations. Furthermore, Figure 4.1 only shows a simpler case in which the internal model only incorporates the most recent known system state, whereas it is also possible to include several past system states as input, for example the five most recent measurements of the state variable.

In a system with multiple internal models, it is possible that internal models share the same input and output domains (i.e. read data from the same inputs and/or provide data for the same outputs), which makes it necessary to introduce a mechanism for the assignment of data from and to inputs and outputs, to avoid the occurrence of conflicts and impasses. Having multiple internal models share one input, i.e. read and process the same input data, is unproblematic on the output side as long as the outputs of the internal models are different. For example, in a robot equipped with two arms, motor commands could be generated via two internal models for both arms separately by using input from a single camera sensor (but note that here the related problem arises of how multiple internal models can decide which input signals are relevant to them, a topic which will be discussed in Chapter 5). However, when multiple internal models share an output, somehow the data produced by the different internal models needs to be combined and a decision needs to be made. This is related to the problem of allowing a robot to pursue multiple tasks simultaneously, for example when trying to bring the hand of the robot to a target position while also trying to maintain a certain orientation of the hand, or trying to avoid driving joints into their limit positions. As another example, a humanoid robot could be given different target positions for both hands. Bringing the hands to these positions could involve rotating the torso, but rotating it to bring the right hand closer to its target position could move the left hand further away from its respective target position. In all of these cases, different sensory inputs and targets (such as the current position and orientation of the hands, and their respective targets) are transformed via different internal models into commands for the same control spaces (the joint space of one arm, or two joint spaces that both include the torso rotation joints). Also for problems not related to kinematics control the same issue exists, for example in the case of two movement behaviors generating commands for the same motors, such as one behavior for following a track and another behavior for avoiding obstacles.

Existing approaches to the integration of internal models can be classified into three main categories: Those integrating internal models (i) through *serialization*, effectively allowing only a single internal model privileged access to a resource; (ii) by forming a *linear combination* of the values output by multiple internal models; or (iii) by assigning *priorities* to candidate values and selecting among those values with the highest priority. In the following, approaches from the literature will be described and their respective advantages and limitations will be discussed.

4.1.1 Approaches Based on Serialization

The strategy to serialize access to resources, such as motors of the robot, is usually employed by cognitivist approaches. Architectures based on the sense-plan-act approach compute sequences of actions on the basis of an abstract world model, which are then executed by the motors one after another. Similarly, also hybrid architectures (see Section 2.2.2) and other state-of-the-art robot setups often rely on this strategy, as they employ deliberative reasoning methods to ensure that no conflicts between concurrently operating low-level modules occur, or are fixed by design to avoid conflicts, such as several behaviors trying to access the same motor resource. For example, in the work of Gienger et al. (Gienger et al., 2010), the task for a humanoid robot to pick up an object is translated into the execution of an action sequence of the kind "walk to position x in front of the table," "find the object," "determine a good way to grasp and perform it," etc. Most of these basic actions are implemented as *whole-body controllers*, meaning that each internal model takes over exclusive control of the entire body of the robot or at least entire parts of the body, such as limbs. These controllers transform a desired displacement in a task space into a displacement in the space of a control variable (comparable to dynamics models, see Figure 4.1), for example by transforming the direction of movement that would bring the robot's hand closer to a target position into a direction of change in the robot's joint space. To some extent it is possible to include further criteria into this control process, for example to avoid joint limits (e.g. Gienger et al., 2005). These additional criteria are then optimized in the so-called "null-space" of the movement, meaning that they are only effectuated as long as they do not interfere with the main task of the controller of moving the hand to its target position. Some actions can also be performed in parallel by employing two or more controllers simultaneously, for example visual search only requires using the head motors while the head movement is rather negligible for the behavior of walking to some position. However, both the definition of what is the main task and what are additional optimization criteria, as well as the knowledge of which controllers can operate in parallel, have to be carefully implemented by the designer based on task-specific considerations and are put into the system a-priori.

Also some behavior-based architectures (see Section 2.3), for example Brooks subsumption architecture (see Section 2.3.1; Brooks, 1986), effectively serialize the output of internal models: While all behaviors are active in parallel, behaviors from higher layers of the architecture *overwrite* the output of behaviors on lower levels of the architecture and thus entirely take over the control of the effector.

4.1.2 Approaches Based on Linear Combination

In contrast to using serialization to integrate internal models, using a linear combination of outputs allows the system to "blend" between solutions. Here, the values produced by individual internal models for the same output modality are summarized using some form of weighting scheme, which should ensure that those internal models that are most relevant to the current situation should predominate the overall system behavior

by having the strongest coefficient. This form of integration is used for example in approaches that use vector field implementations for behaviors, such as AuRA (see Section 2.3.1; Arkin, 1989). Here, a geometric weighting scheme is used, as the responses of behaviors are tied to distances between the robot and points in the environment, such as the robot's distance to its goal location, or its distance to an obstacle.

Using a linear combination to integrate internal models allows for a cooperation between the internal models, which is not possible when integrating through serialization. The system can blend between solutions instead of always having a single component determine the whole behavior of the robot. However, the vector field methodology is very specialized for the problem of robot navigation and has inherent difficulties, such as running into local minima (cf. Section 2.3.1). Also, the method does not allow for a selection of one of several alternative courses of action to satisfy multiple tasks.

4.1.3 Approaches Based on Prioritization

This third class of approaches to integrate internal models is based on the idea to have on the one hand the internal models provide different candidate values for the output, and on the other hand have an accompanying process that evaluates the different candidates in terms of how well they are suited for the current tasks of the robot.

Wolpert and Kawato's MOSAIC model (see Section 3.2.2; Wolpert and Kawato, 1998; Wolpert et al., 2003) for example employs a set of paired forward and inverse models, all representing the same action of for example lifting an object, but with different parametrization of the action. The responses of individual inverse models are assigned priorities based on how well the respective forward models predict the sensory input data, and the final response of the system is computed as a weighted sum of the individual responses, by using the priorities as coefficients. This allows for the control of a system of which the state has a hidden component, such as the weight of the object that should be lifted. Through the dynamic adaptation of the coefficients, the hidden component of the control system is thus implicitly estimated. MOSAIC however only proposes a method for integrating several instantiations of the same internal model, each with a different parametrization, but not for generically integrating different internal models, and does not allow for a selection of one of different candidate actions to solve multiple tasks.

The global workspace architecture (see Section 2.5.2; Shanahan, 2006) uses embodied simulation of forward models and proposes that a dedicated value system evaluates the outcome of different actions before the robot physically executes them. However, as the system selects an action based on whether the associated value exceeds a certain threshold, and thus only employs one winning internal model for executing an action at a time, also the global workspace architecture effectively implements a serialized integration of internal models.

Rosenblatt's "distributed architecture for mobile navigation" (DAMN; Rosenblatt, 1997) is intended as a framework for the integration of navigation behaviors, such as goal seeking and obstacle avoidance, which should avoid both relying on a binary selection of behaviors, as in Brooks subsumption architecture (see Section 2.3.1; Brooks, 1986), as

well as running into the problems associated with the vector field approach. In DAMN, the output space of the behavior is discretized, and the outcome of each behavior is evaluated for the discrete alternatives. For example, the space of turning actions is discretized into the five actions of moving "hard left", "soft left", "straight ahead", "soft right" or "hard right". Each behavior then votes for the alternative actions, for example an obstacle avoidance behavior would estimate if either of the alternatives would result in a collision and as a result would vote for or against the corresponding discrete actions, and a path following behavior would vote for those alternatives that would remain on the path. The action that has received most votes is executed by the robot. Rosenblatt's approach is only feasible for few action alternatives, as predictions need to be made for all alternatives by all behaviors, which can quickly lead to an explosion in computational cost.

A similar approach is based on fuzzy command fusion (e.g. Pezzulo et al., 2005; Saffiotti, 1997; Yen and Pfluger, 1995) by representing behaviors as fuzzy rules, such as "IF *obstacle at -45°* THEN *turn 5*", which are translated into fuzzy sets (such as triangular shapes around the specified values). This allows to combine the responses of the different behaviors using fuzzy logic, thus assigning a higher value to commands that satisfy multiple behaviors, and to apply different defuzzyfication strategies, which allows for some more design freedom to control the overall system response as compared to Rosenblatt's approach (Yen and Pfluger, 1995). However, as already mentioned in the discussion of the AKIRA model, fuzzy command fusion has problems with scaling up to higher dimensions and only rather simple sets can be feasibly expressed using the language of fuzzy logic (see Section 3.2.1; Sala et al., 2005).

4.2 Making Use of Redundancies for the Integration of Internal Models

We are interested in finding a way to generically integrate internal models as the building blocks of a cognitive architecture. This gives us the following two important constraints: The method should be free from any domain- or task-specific considerations, and it should allow the robot to learn internal models through sensorimotor experience. This section will propose a method for the integration of internal models, which makes use of the existence of redundant solutions in many sensorimotor transformations. It is intended to allow the system to find and favor solutions, which fulfill multiple tasks simultaneously. As such, the method is in some ways similar to the approaches for integrating internal models based on prioritization (see Section 4.1.3), as it assigns higher priorities to solutions that satisfy multiple tasks, and dynamically chooses among the solutions with highest priority. This section introduces the method itself, which is not tied to the use of any particular learning technique. An example implementation of the method using a neural network model will be described in Section 4.3.

Most approaches for the learning and representation of internal models focus on accuracy, on the generalization capability of the learning method from as few training examples as possible, and on its suitability for online-learning. While all of these aspects

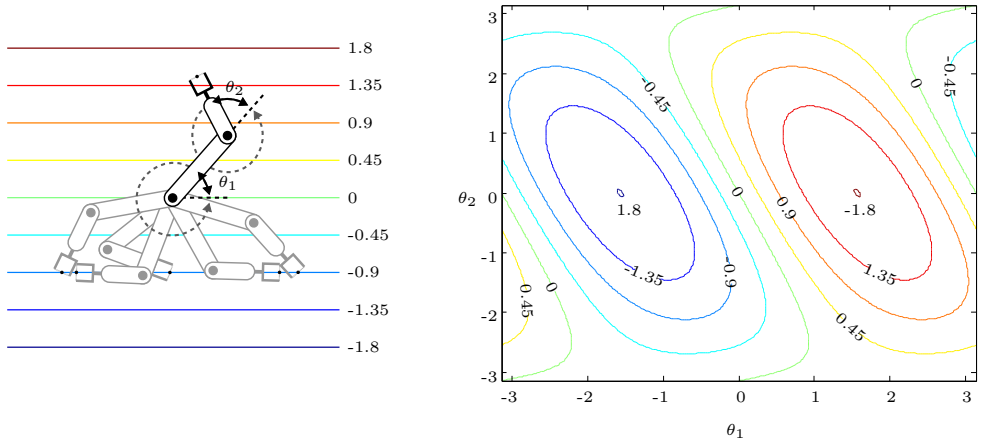
are very important, usually the learning of only one single internal model is studied in a special experimental setup in isolation (e.g. Ijspeert et al., 2003; Kormushev et al., 2010; Montesano et al., 2008; Mülling et al., 2011). Humans constantly perform many tasks in parallel. We can sip from a cup of coffee while walking, we can read in a magazine while stirring in a casserole, we can talk to another person while we are driving a car, etc. This is possible because our bodies are highly *redundant*, which means that we have many ways to solve a single task. When putting the index finger of one hand on a spot on a table, we can still move our elbow around quite freely without having to lift the index finger. Thus, there are several solutions of arm configurations allowing to fulfill the task of keeping a finger on a certain spot on the table. With the other hand we could now still reach most points on the table for an additional task of picking up an object, even if it meant to bend over or step to the side a bit. This redundancy, or flexibility in fulfilling a task, enables us to perform several tasks simultaneously.

Many robots also have redundant kinematic setups, for example humanoid robots mimic the structure of the human body. These robots in principle have the same, or at least some of the flexibility that we have in performing skills. As mentioned above (see Section 4.1.1), in control theory the subspace of the joint-space which allows for reaching a task goal is termed the *null-space* of a task (Liégeois, 1977). Knowledge and control of the null-space offers advantages (Gienger et al., 2005): When controllers are designed by the system developer, null-space control allows for example to avoid self-collisions while reaching. However, the necessary information has to be carefully considered by the human developer beforehand when implementing the controllers.

In contrast, in the machine learning literature redundancy is often discussed as a problem, the *non-convexity problem* (Jordan and Rumelhart, 1992). The problem states that averaging between multiple solutions for one task does not necessarily yield a valid solution (see also Section 4.3 below). Averaging between solutions is done in learning approaches that use function approximation when training from data points that originate from a *multivalued* function. Thus, these methods try to learn a *many-to-one* mapping yielding incorrect solutions where actually a *many-to-many* mapping should be learned. An illustrative example is that of a simple robotic arm with a fixed base and two rotational joints. The robot can bring the end of its arm to most points in its workspace in two ways, which are the "elbow-up" and the "elbow-down" solutions. Learning methods that average between solutions will average these two solutions, ending up with associating a fully extended arm posture to most target points, which is incorrect.

To overcome this problem, "redundancy resolution schemes" are applied to the training set, which sort out training samples to guarantee that effectively only training samples from a single-valued function remain (Rolf et al., 2010). One consequence is of course, that the resulting learned mapping only stores a single solution for each target. Thus, there is a great loss of information and the robot cannot know how to move around in the null-space.

Instead of removing information to adjust the learning problem to standard machine learning approaches, here it is instead investigated how multivalued functions can be



(a) (b)

Figure 4.2: Example of a sensorimotor task with redundant solutions. (a) A planar robot with two rotational joints and a fixed shoulder is given the task to bring its hand to a certain vertical position, disregarding the horizontal position. That is, each colored line corresponds to equally valid solutions to the same task specification: All arm configurations shown in light gray represent examples for valid solutions for the task of bringing the hand to the vertical position $y = -0.9$. (b) The two-dimensional joint space of the robot, spanning the interval $[-\pi, \pi]$ in both dimensions. Colored lines represent the sets of solutions to different task specifications, corresponding to the lines shown in (a). Thus, each point on one of the lines corresponds to a solution to the respective task specification.

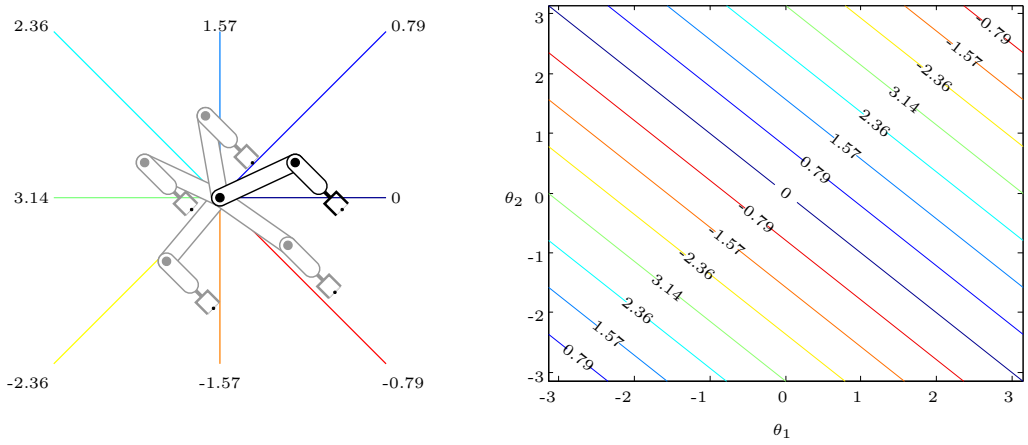
learned and the valuable information about redundancies preserved. Furthermore, it will be shown how multiple internal models that are learned separately from each other can be integrated to find solutions that comply with several tasks at once, by making use of the learned knowledge about redundancy in tasks.

4.2.1 Redundancy in Sensorimotor Tasks

In mathematical terms, redundancy can be interpreted in terms of a *many-to-many* mapping: For a given task specification, such as a target position for the hand in an inverse kinematics model, there can be many possible values for the output variable (postural configurations in this case) that are a solution for the task. Thus, the mapping is not restricted to be a function (i.e., a *many-to-one* mapping). As a simple example[1], consider the case of a planar robot with two rotational joints and the task to control the vertical position of its end-effector, disregarding its horizontal position. For each target vertical position (apart from the upper and lower extremes), the robot has infinitely many configurations to solve its task, see Figure 4.2.

Formally, the forward kinematics of the robot is given by the function (i.e., many-

[1] example adapted from (Rolf et al., 2010)



(a) (b)

Figure 4.3: A second task with redundant solutions for the robot shown in Figure 4.2. (a) The robot should obtain a certain orientation of its hand, that is, all configurations shown correspond to solutions to the task of obtaining the orientation $\alpha = -0.25\pi$. (b) The robot's joint space, showing sets of redundant solutions corresponding to directions indicated by the lines in (a).

to-one mapping)

$$f \;:\; \Theta \subset \mathbb{R}^2 \;\longrightarrow\; Y \subset \mathbb{R}, \tag{4.1}$$

where the input domain Θ is the set of possible (two-dimensional) joint configurations of the robot and is mapped onto the output domain Y, which is the interval of vertical positions that the robot can reach. In contrast, the inverse kinematics of the robot is given by a many-to-many mapping

$$g \;:\; Y \;\longrightarrow\; \mathcal{P}(\Theta), \tag{4.2}$$

where $\mathcal{P}(\Theta)$ denotes the power set of Θ, such that

$$\forall \quad \boldsymbol{\theta}^i, \boldsymbol{\theta}^j \;\in\; g(y),\; y \in Y \;:\quad f(\boldsymbol{\theta}^i) \;=\; f(\boldsymbol{\theta}^j), \tag{4.3}$$

meaning that $g(y)$ is the subset of Θ of angular configurations $\boldsymbol{\theta}$ that will bring the robot to the same vertical positon y. In Figure 4.2(b), each colored line represents the set of solutions $g(y)$ for one particular $y \in Y$, that is, any point along one line is mapped by the forward kinematics function $f(\cdot)$ onto the same value y. Thus, to reach its task to bring its end-effector to a certain vertical position, the robot can choose any one solution that lies on the corresponding line, representing the set of redundant solutions for that task.

As a second task that the simple robot in Figure 4.2 might have, assume that it should control the orientation of its end-effector. That is, we introduce another inverse

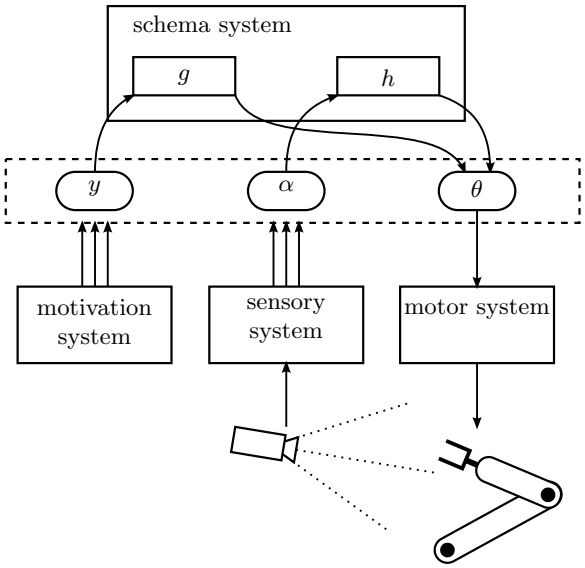


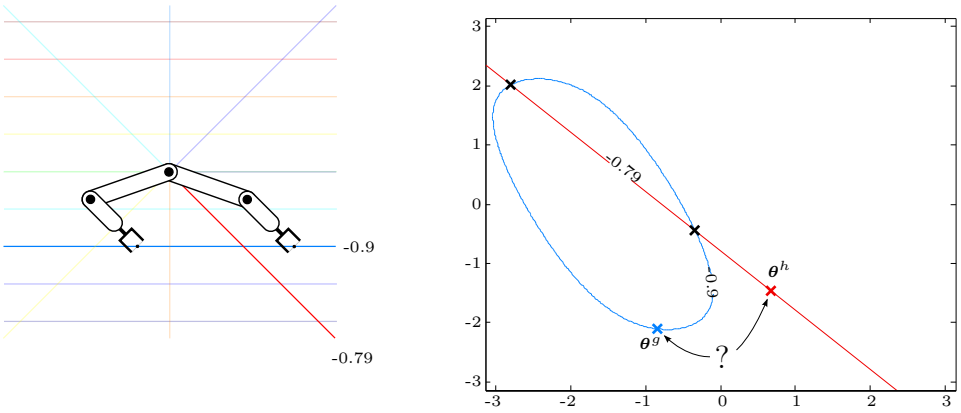
Figure 4.4: Layout of a system with two internal models, both transforming sensory information and target values into arm configurations $\boldsymbol{\theta}$. When the robot is given task specifications for both y and α, the system needs to select a single solutions from multiple candidate solutions.

mapping, which captures the relation between end-effector orientations $\alpha \in [-\pi, \pi]$ of the robot arm and angular configurations $\boldsymbol{\theta}$,

$$h \;:\; [-\pi, \pi] \;\longrightarrow\; \mathcal{P}(\Theta)\,. \tag{4.4}$$

The robot has infinitely many configurations for all target end-effector orientations, which also makes this second inverse mapping a many-to-many mapping, see Figure 4.3.

Now assume a system with the layout shown in Figure 4.4, that has acquired two internal models, corresponding to the two inverse mappings g and h, respectively. It could be given two tasks, on the one hand to bring its hand to a certain vertical position y, and on the other hand to obtain a certain orientation α of the hand. If the two internal models of the system would both independently produce a single solution, $\boldsymbol{\theta}^g$ and $\boldsymbol{\theta}^h$ respectively, these two solutions would most likely differ from each other. None of the methods for the integration of internal models that were described above in Section 4.1 would be suited to produce a solution that satisfies both tasks simultaneously: Using a serialized integration, the robot could only perform one task after the other by switching between the two arm configurations $\boldsymbol{\theta}^g$ and $\boldsymbol{\theta}^h$; using some form of linear combination $\boldsymbol{\theta}^* = a \cdot \boldsymbol{\theta}^g + b \cdot \boldsymbol{\theta}^h$ would not yield a valid solution to either task, as the two internal models describe entirely different sensorimotor mappings, see Figure 4.5. Only if both internal models can restore the knowledge about redundancies in the task can the system decide on a solution that solves both tasks simultaneously. The approaches for the integration of internal models using fuzzy command integration



(a) (b)

Figure 4.5: (a) The two arm configurations that solve the two tasks of bringing the hand to the vertical position $y = -0.9$ and obtaining a hand orientation of $\alpha = -0.25\pi$ simultaneously. (b) The robot's joint space, showing the sets of redundant solutions corresponding to the respective tasks. If the system employed its two internal models independently from each other, thus producing two uninformed candidate solutions $\boldsymbol{\theta}^g$ and $\boldsymbol{\theta}^h$, shown as a blue and a red cross respectively, neither of them would satisfy both tasks. Also a linear combination of the two would not yield a solution for both tasks, but in most cases would produce an arm configuration that would satisfy neither task instead. Only the two configurations that lie in the intersections of both sets, shown as black crosses, are solutions to both tasks.

(see Section 4.1.3) pursue a similar idea, as in their cases each internal model produces a fuzzy set as response, instead of a single vector. However, it is not easily possible to represent non-trivial sets of solutions, such as the set of solutions $g(-0.9)$ shown in blue in Figure 4.5(b), using fuzzy Sets.

Therefore, it is here proposed to integrate internal models in a generic way by relying on a learning method that can maintain the information about redundancy in the training data, and a form of representation that allows to retrieve not only single solutions, but *arbitrary sets of solutions* upon a query. We then want to apply a prioritization scheme for the integration of internal models, which favors those solutions that lie in the intersection of as many solution sets as possible. In Figure 4.5(b), exactly two solutions exist that lie in the intersection of both solution sets for the two tasks. These solutions correspond to the arm configurations that can be seen in Figure 4.5(a). Furthermore, in most sensorimotor tasks, gradual changes in the target value also correspond to gradual changes in the associated variables. Thus, when for example the target position or orientation for the robot's hand is changed gradually, also the set of solutions moves continuously, not abruptly. Therefore, the selection of new solutions as the task changes over time should be dependent on the current solution, and the system should not abruptly select entirely different solutions, which would result in an

unstable behavior of the robot. Finally, one of the tasks could have a higher priority for the robot than others. For example, actually brining the hand to the target position might be crucial, while obtaining a certain orientation of the hand might be optional. Thus, the former task should be given a higher priority than the latter.

4.2.2 Dynamic Selection of Solutions Using Dynamic Neural Fields

In Chapter 3, it was already proposed that dynamic neural fields should form the interface for information exchange between building blocks in a cognitive architecture. As we will now see, this choice is not only motivated by theoretical considerations, as discussed in Section 3.3.3, but also has a pragmatic reason: The differential equation that governs the activation dynamics of a dynamic neural field can be tuned in such a way that the properties of a decision making mechanism for the integration of internal models, as outlined in the last section, can be implemented using a dynamic neural field model. This section will demonstrate that internal models can be integrated by using dynamic neural fields, and that a single solution can be selected that is in the intersection of multiple sets of solutions, which were retrieved from the internal models. Furthermore, it will be shown that the selection is stable in the light of changes to the task.

In a dynamic neural field model, it is assumed that neurons are topographically arranged and are laterally connected to their respective neighborhood in two ways: Connections are excitatory for short distances and inhibitory for longer distances between neurons (comparable to a "mexican hat" function). The field dynamics is based on a dynamic update rule, which was first studied by Amari (Amari, 1977) and is therefore also referred to as the "Amari dynamics". Amari investigated the properties of such fields with respect to dynamic pattern formation and stability. One simple, but in the context of this work important case is the formation of a single peak solution in the neural field, such that the inhibitory connections prevent the formation of any other activation peaks. The peak remains stable as long as it is provided with input, is robust against noise as the Amari dynamics implements a low-pass filtering of the input signal, and can also follow the input signal when the location of strong input is shifted gradually. Thus, the Amari dynamics nicely complies with the requirements for the decision making process that was stated above.

The dynamic neural field model that is used in this work follows that of Erlhagen and Schöner as a discrete time implementation (Erlhagen and Schöner, 2002), which was also adapted by Toussaint (Toussaint, 2006). The differential equation governing the activation u_i of the neurons in the neural field is

$$\tau \dot{u}_i = -u_i + h + S_i + \sum_{j=1}^{m} w_{ij} \varphi(u_j), \tag{4.5}$$

where τ is the time constant of the dynamics, h is a parameter for global self-inhibition and specifies the resting level, S_i is the input to the i-th unit in the neural field, w_{ij} is a distance weighting that effectively implements the excitation and inhibition property,

and φ is a non-linearity. Here, a "ramp" function is used,

$$\varphi(u) = \begin{cases} 0 & u \leq 0 \\ u & 0 < u < 1 \\ 1 & u \geq 1 \end{cases}, \tag{4.6}$$

which at the same time also serves as the output function for the field. For the distance weighting w_{ij} a Gaussian function shifted by a constant w_I is used, to achieve inhibition across the whole neural field instead of an inhibition that is restricted to a local neighborhood,

$$w_{ij} = w_E \cdot \exp\left(-\frac{d(i,j)^2}{2\sigma_E^2}\right) - w_I. \tag{4.7}$$

Here, $d(i,j)$ is the Euclidean distance between the units i and j in the neural field, σ_E determines the excitatory range and w_E determines the strength of the excitation.

To use the field dynamics as a decision making mechanism for the integration of internal models, the outputs of the internal models must be translated into an input activation pattern S_i for the field neurons. For this, population coding (Deneve et al., 2001) is used, such that each field unit responds selectively for a certain region of the input space with a bell-shaped tuning curve, that is, the field units are represented by radial basis functions with centers $\mathbf{m}_i$. Given an input vector $\mathbf{x}$, the population code is computed as

$$S_i = G(\mathbf{x} - \mathbf{m}_i), \tag{4.8}$$

where $G(\cdot)$ is a Gaussian function according to

$$G(\mathbf{d}) = \exp\left(-\nu \cdot \frac{\mathbf{d}^T\mathbf{d}}{2\sigma^2}\right). \tag{4.9}$$

The receptive fields of the units have to cover the input space, such that all possible values of $\mathbf{x}$ can be represented. For this it is possible to self-organize the topology of the neural field and the distribution of receptive field centers $\mathbf{m}_i$ in a data-driven manner (Gläser et al., 2008). However, for reasons of simplicity, a predefined topology and a static distribution of receptive field centers is used in this work. The connection pattern of the field units corresponds to a multidimensional regular grid, and the centers $\mathbf{m}_i$ of the receptive fields are evenly distributed to cover the domain of the input vector $\mathbf{x}$.

Using population coding, it is not only possible to represent a single value of the input vector $\mathbf{x}$, but also to represent multiple values, or no value at all (i.e., an absence of input). Thus, sets of solutions for a task, such as the ones shown in Figure 4.5(b), can be simultaneously represented as an activation landscape for the input S_i to the dynamic neural field, by combining population codes for all values of $\mathbf{x}$ in the set of solutions $\mathcal{X}$. A simple way to combine population codes is by using the maximal response for each unit, as

$$S_i = \max_{\mathbf{x} \in \mathcal{X}} \{\ G(\mathbf{x} - \mathbf{m}_i)\ \}. \tag{4.10}$$

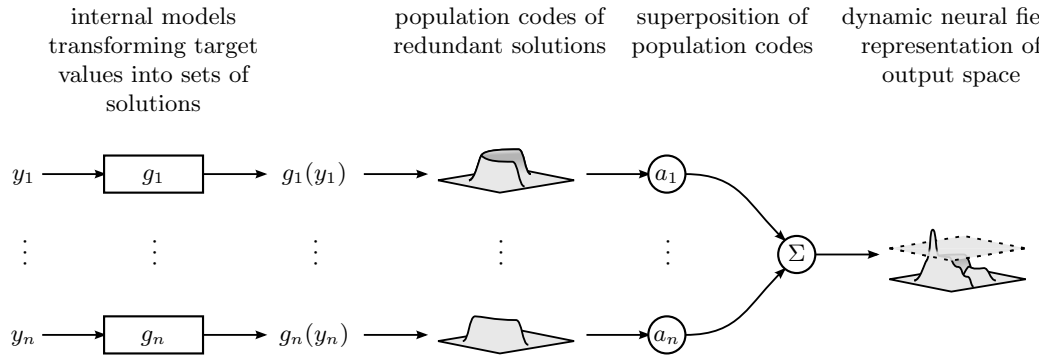


Figure 4.6: Overview of the proposed method for the integration of internal models. It is assumed that internal models can produce sets of solutions for tasks, instead of single solutions, which are transformed into population codes. These are then combined as a superposition, which is provided as input to a dynamic neural field. The field dynamics selects a single value for the output variable.

To combine the outputs of multiple internal models, $S_i^1, \ldots, S_i^n$, each corresponding to a set of solutions $\mathcal{X}_k$, a superposition is used,

$$\widehat{S}_i = \sum_{k=1}^{n} a_k \cdot S_i^k, \tag{4.11}$$

where the factors a_k are normalized so that their sum is below or equal to 1. This is done, because the activation dynamics of dynamic neural fields is very dependent on the choice of parameters (see Equation 4.5), especially in terms of how the network responds to different amplitudes in the input activation. By restricting inputs $\widehat{S}_i$ to values from the interval $[0, 1]$, it is more feasible to tune the parameters (note that by using a Gaussian function as the activation function, as defined in Equation 4.9, it is ensured that also the individual population codes S_i only have values from that interval). Figure 4.6 gives a schematic overview of the proposed method.

Figure 4.7 shows the response of a dynamic neural field, when receiving as input a population code for solutions to the two tasks used in the above example of a planar robot (cf. Figure 4.5). It can be seen that the field behavior complies well with the definition of properties of a decision making mechanism for the integration of internal models as stated in Section 4.2.1, in the following way. If there is no input at all, then the field activation goes to a resting level below the lower threshold of the output function, thus there will also be no output from the dynamic neural field. If there is input to the field, coming from internal models, the summation in Equation 4.11 serves as the prioritization scheme for the decision making mechanism: When solutions from different internal models lie close together in the output space, i.e. both lie close to the center of the receptive field of one field unit, this unit receives a stronger input than units

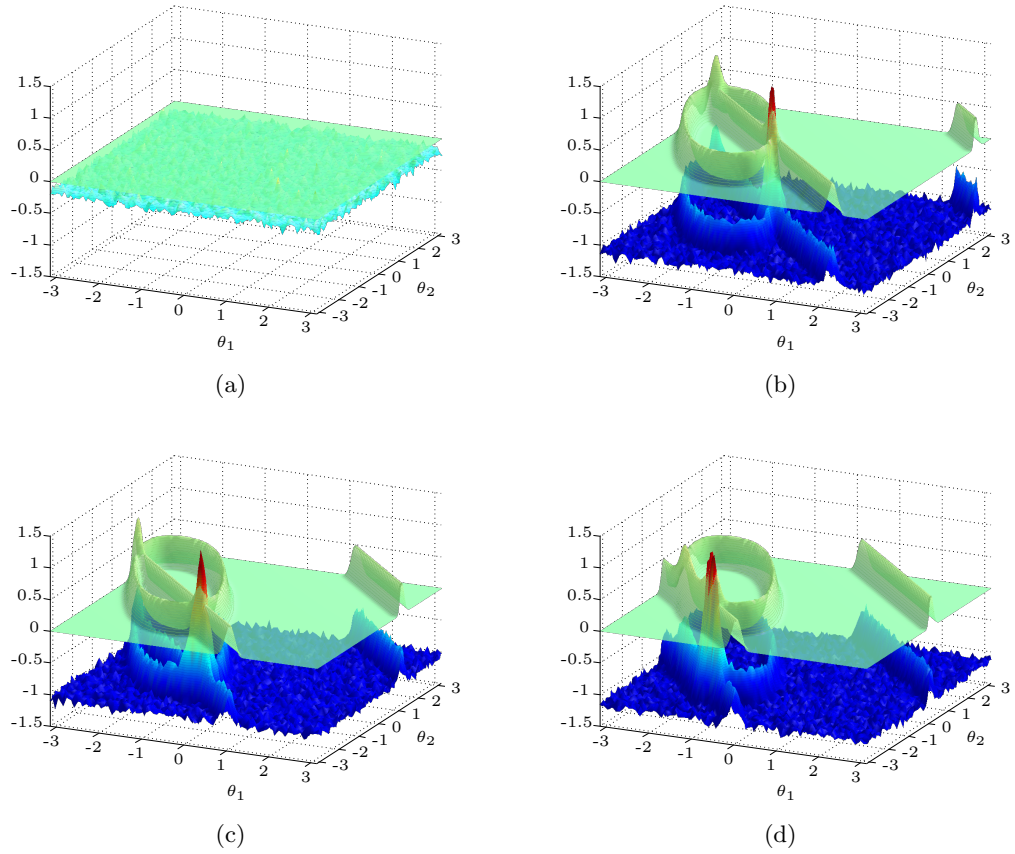


Figure 4.7: Example for how a dynamic neural field can be used as a decision making mechanism for the integration of internal models. Field inputs are shown as activation landscapes in light green, and the resulting activation of the dynamic neural field is shown in blue. Peaks of activation above threshold (zero-level), corresponding to the output of the dynamic neural field, are shown in red. (a) If no input is present, the field activation goes to a resting level below threshold. (b)–(d) The two tasks of bringing the hand of the robot to a target position and obtaining a target orientation of the hand are transformed by internal models into sets of solutions, here to be seen as a circular shape for the former and the two diagonal lines for the latter task, cf. Figure 4.5(b). Both inputs are combined into a single activation landscape through a summation with equal coefficients, as explained in the text. The field responds with a single peak of activation above threshold, lying at one of the intersections of the two sets. When the task of obtaining a target orientation is moved gradually, the solution set also moves gradually in the output space. As the peak of activation moves towards maximal input amplitude in its local neighborhood, it dynamically follows the moving input.

with solutions of only one internal model inside their receptive fields. The activation of the dynamic neural field consequently develops into an activation landscape where only a single peak at an intersection of the solution sets lies above threshold. If the target value for one internal model now gradually changes, in this case the target orientation for the hand, the peak of activation dynamically follows the input, i.e. the set of solutions. These properties derive from known properties of the neural field dynamics (Amari, 1977), as peaks of activation only develop in the field where there is input present, and peaks move in the direction of increasing input amplitude, searching for the maximum, as long as it is within the range of excitation.

Assigning priorities to tasks can easily be achieved in the model by controlling the factors a_k in Equation 4.11. If there exists a point where all solution manifolds intersect, this point will have the highest activation value in the neural field. If however there is no such point, the manifold with the higher priority will have a higher activation value in the superposition and will thus be selected by the field dynamics.

4.2.3 Distributed Decision Making in Co-ordinated Dynamic Neural Fields

Above it was described, how multiple responses of internal models can be integrated by making use of the existence of redundant solutions, and using dynamic neural fields. The proposed method allows to simultaneously solve multiple tasks that depend on the same outputs (e.g., require the use of the robot's arm). However, a complete cognitive system not only has to select a solution in a single output domain, but needs to make many decisions in parallel, some of which might be interdependent. For example, when an agent should find and pick up a fruit, it needs to first decide on a particular kind of fruit, maybe in the light of additional criteria, then find an instance of the fruit and transform its location into motor commands to reach for it. In Section 3.3.3, it was already outlined, how a distributed decision making process should take place in the cognitive architecture that was proposed in the last chapter. Now, it will be described, with the aid of a simulation experiment, how a distributed decision making process can be implemented by co-ordinating the activation in multiple dynamic neural fields.

For this, consider again the planar robot with two degrees of freedom (see Figure 4.2). In the above descriptions for the integration of internal model responses it was assumed, that the joint space of the robot is represented by a single dynamic neural field, of which the output corresponds to the decision of the system. Several internal models generate sets of redundant solutions in this joint space, which are combined in the input to the dynamic neural field. However, in a system with multiple dynamic neural fields, each operating independently, but coupled with each other via the learned internal models, it can become necessary that the decisions of the individual fields are co-ordinated, in the sense that the overall behavior of the system is only coherent if the decisions of the different fields are mutually constrained. As an example, consider a system that represents the joint spaces for the two joints of the robot individually, each as a one-dimensional dynamic neural field (cf. Figure 3.8). Such a separation makes sense, as different internal models can generate solutions for different sub-sets

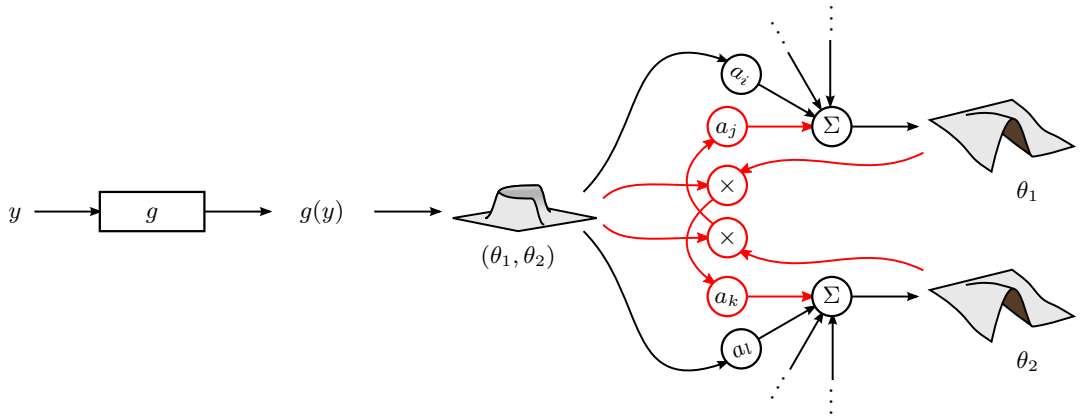


Figure 4.8: An extension of the method for integrating internal models, allowing a distributed decision making process by co-ordinating multiple dynamic neural fields. See text for description.

of degrees of freedom. For example, another internal model would generate solutions for only one degree of freedom, when the task is to lock a single joint into a certain angular position. As a more complex example, imagine one internal model for balancing a humanoid robot, which makes use of the whole body, while another internal model transforms egocentric object coordinates into commands for reaching with the arm, thus only making use of the degrees of freedom related to the arm. In such cases, the decisions made by the individual dynamic neural fields, each for a different degree of freedom (or sub-set of degrees of freedom), need to be co-ordinated, as otherwise the overall system behavior would not solve the tasks that depend on multiple degrees of freedom.

To achieve such a co-ordination of multiple dynamic neural fields, an extension is introduced to the integration method proposed above, as shown in Figure 4.8. The two degrees of freedom of the robot's arm are represented separately by two one-dimensional dynamic neural fields, which by themselves each take as input a combination of model responses for integration (as previously described, cf. Figure 4.6). Additionally however, for internal models that rely on combinations of multiple output spaces, as here for example an internal model generating solutions in the two-dimensional space of joint configurations of the arm, mutual feedback connections are introduced that link the dynamic neural fields (sown in Figure 4.8 in red). Via these feedback connections, the *outputs* (i.e., the decisions) of the dynamic neural fields are transformed into an additional input to each of the respective other involved dynamic neural fields (this transformation is indicated as crosses in Figure 4.8). This way, the information about the current decisions of the other fields is available at the input of each individual dynamic neural field, which makes a co-ordinated decision making process possible.

More precisely, the solutions given in the output of the internal model, which are rep-

resented in the combined higher-dimensional space (in the example the two-dimensional joint space), are projected into the component dimensions, and the projections are used as input to the individual dynamic neural fields. The feedback (i.e., the output of the dynamic neural fields) that is given in response is transformed by first discarding from the output of the internal model all solutions but those that agree with the current decision of the dynamic neural field, and then projecting only the remaining solutions into the component dimensions. For the example of the two-dimensional robot arm this would mean that the output of the dynamic neural field representing the first joint would be translated into all those solutions for the second joint, which agree with the current decision of the first joint (and vice versa for the other feedback connection).

Figure 4.9 shows simulation results of the system for the example of the planar robot arm. Two one-dimensional dynamic neural fields are used to represent the two joint angles of the robot, θ_1 and θ_2, respectively. The robot is given the task to bring its end-effector to a certain vertical position, which is translated by an internal model into sets of redundant solutions, represented using population coding in a two-dimensional (non-dynamic) neural field. Additionally, a second internal model is introduced for the task to keep the first joint in a target angular position. Thus, both dynamic neural fields receive as inputs on the one hand the output of the first internal model, projected into the respective dimensions that the fields represent. On the other hand they receive the mutual feedback inputs, as described above. Furthermore, the dynamic neural field representing the first joint receives the additional input from the second internal model.

The target vertical position for the end-effector is continuously shifted from the lower-most, upwards to the higher-most. The target angular position for the first joint initially remains constant. Figure 4.9 shows the evolution of the activation of both one-dimensional dynamic neural fields over time. The solutions for the target vertical position correspond to plateaus of equal activation in the inputs to the two fields. The current decision of the respective other field, translated via the feedback connections, arrives at the inputs as "bumps" in these plateaus. The target angular position for the first joint is present as an additional peak in the input to the corresponding dynamic neural field. After a few time steps, both one-dimensional fields have co-ordinated their decisions, such that the arm reaches its target vertical position of the end-effector, however not having the first joint in the target angular position at first, since the two tasks cannot be solved simultaneously. Thus, the peak corresponding to the target angular position remains below threshold. Figure 4.9(a) shows for the time step marked with (a) the set of solutions to the task in the two-dimensional joint space of the robot (i.e., the output of the first internal model), along with the result of the distributed decision-making process as a peak at the location in the two-dimensional space, which corresponds to the two individual decisions. This peak coincides with the set of solutions, meaning that a valid system behavior is achieved for solving the first task. As the target vertical position is moved upwards, the set of solutions changes, and the decisions of the two fields continuously track the solutions, until at the time marked with (b), the target vertical position is at a level that allows the robot to also achieve the second task of bringing the first joint into its target angular position. Consequently, the

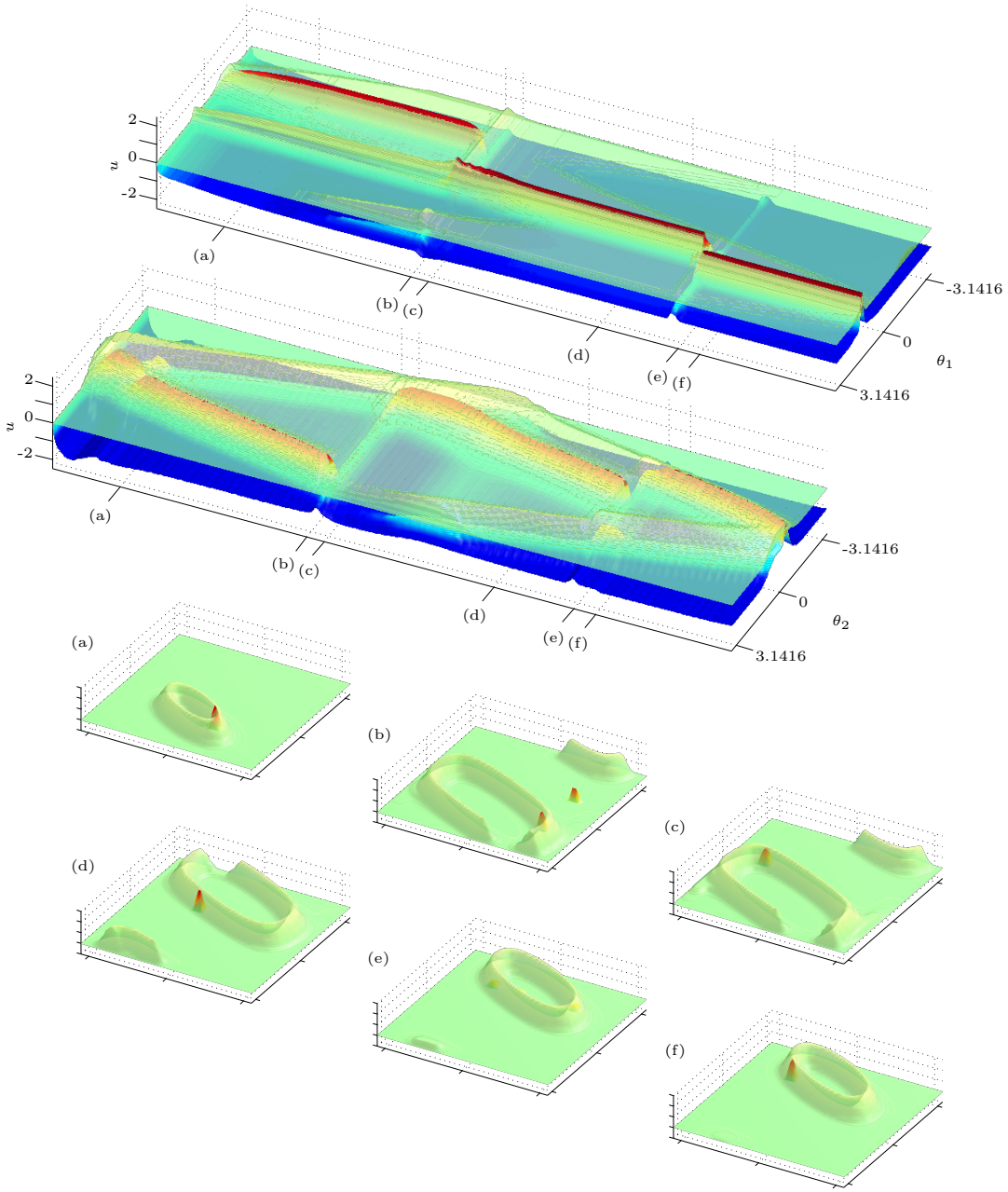


Figure 4.9: Simulation results showing the system behavior for distributed decision making in two co-ordinated dynamic neural fields. See text for description.

dynamic neural field corresponding to the first joint transitions to this single solution that solves both tasks, as it is now the strongest input to the field. This transition briefly produces an overall decision of the system that is incompatible with the first task, as the second dynamic neural field is still in a state of co-ordination corresponding to the previous decision of the first field, which can be seen in Figure 4.9(b) as a new peak begins to form at a location that is not part of the set of solutions. However, shortly after this transition in the first field, the second field again co-ordinates with the new decision in the first field and a new valid solution is picked, as can be seen in Figure 4.9(c), and this new solution is tracked by both fields, see Figure 4.9(d). At the time marked with (e), a new target angular position for the first joint is given, which again triggers a transition in the first dynamic neural field, see Figure 4.9(e), again shortly after followed by a co-ordination with the second field, see Figure 4.9(f).

4.2.4 Summary

In this section it was proposed to integrate internal models by making use of redundant solutions in sensorimotor tasks. For this, it is assumed that internal models store and represent knowledge in such a way that information about redundancy remains intact and can be retrieved upon query. That is, given a target value for an input variable, the internal model generates a set of solutions in the output domain, instead of only a single solution. For a robot with multiple tasks, some of which might depend on the same output variables (such as multiple tasks that can be solved using the same effector), solutions should be selected that satisfy as many tasks as possible. Here it is proposed to use dynamic neural field representations of the output domains, and to use a superposition of population codes of the sets of solutions as input for the dynamic neural field.

Furthermore, it was shown that multiple dynamic neural fields, each representing different output domains, can be co-ordinated to achieve in concert a coherent system response. This allows for the integration of internal models with partly overlapping output domains, as it was shown with the example of a robot arm and separate dynamic neural fields for the individual joints. Splitting the decision making process into multiple dynamic neural fields, as opposed to using a single high-dimensional dynamic neural field, not only allows for a more flexible integration of internal models, but also is beneficial in terms of computational complexity: The number of operations necessary for the computation of the next state of a dynamic neural field grows exponentially with the number of input dimensions. Thus, the use of a high-dimensional dynamic neural field can be computationally demanding, whereas using multiple low-dimensional dynamic neural fields only linearly increases the computational cost.

Note that in the description above it is assumed that the internal model response is first transformed into a population code in a (non-dynamic) neural field representing the combined higher dimensional space. This obviously still entails having to hold a large amount of activation values in memory, growing exponentially with the number of dimensions covered by the neural field representation. However, also this can be circumvented if the query of the internal model supports not only to specify a target value

for the input, but also to include constraints for components of the output variabel. In this case, the representation of the result of a single query in a high-dimensional neural field can be replaced by multiple queries, of which the results are each represented by lower-dimensional neural fields. Later on in Section 4.3.3, this will be explained in more detail on the basis of an example implementation of the proposed method for integrating sensorimotor mappings.

The presented method depends on the assumption that internal models can store and represent redundant solutions, i.e. can learn many-to-many mappings instead of many-to-one mappings. In the following, a neural network model will be presented as one possible way to learn and store many-to-many mappings, before the method is experimentally evaluated in Section 4.4 using a simulation of a humanoid robot.

4.3 Using Networks of Sigma-Pi Units for the Learning and Query of Redundant Mappings, and for Robot Control

In the last section, a method for the integration of internal models was proposed that exploits the existence of redundant solutions. It therefore requires a representation of internal models that is able to restore not only a single solution to a task, but sets of redundant solutions. In this section, a method will be proposed to learn many-to-many mappings from training data, retaining information about redundancies.

A standard approach to the learning of internal models is to use function approximation methods, such as the multilayer-perceptron (Jordan and Rumelhart, 1992). However, since a function approximation method maps each input value onto exactly one output value, these methods cannot store information about redundancies. When using function approximation, trying to learn an internal model from a training set that consists of redundant samples will produce invalid results without further attention, see Figure 4.10 for an example. To overcome this difficulty, usually the learning procedure is adapted in such a way that redundant samples are sorted out, either prior to or during the training (Rolf et al., 2010). Thus it is guaranteed that effectively only training samples from a single-valued function remain. However, dismissing redundant samples from the training set results in a great loss of information, and the robot cannot learn about redundant solutions to a task.

Some learning methods exist that can also deal with multivalued functions. One approach is to train not a single, but many estimators, each of which learns the relationship between input and output values only for a local region (D'Souza et al., 2001; Lopes and Damas, 2007). This approach relies on the assumption that complex non-linear mappings can be approximated as nearly-linear, when only looking at a small subspace. Thus, many simple linear estimators are trained from non-redundant (because locally restricted) training samples. In a different approach, Reinhart and Steil have proposed to associatively combine input and output variables into a single vector, and to train a reservoir network using the resulting higher-dimensional training samples. Reinhart

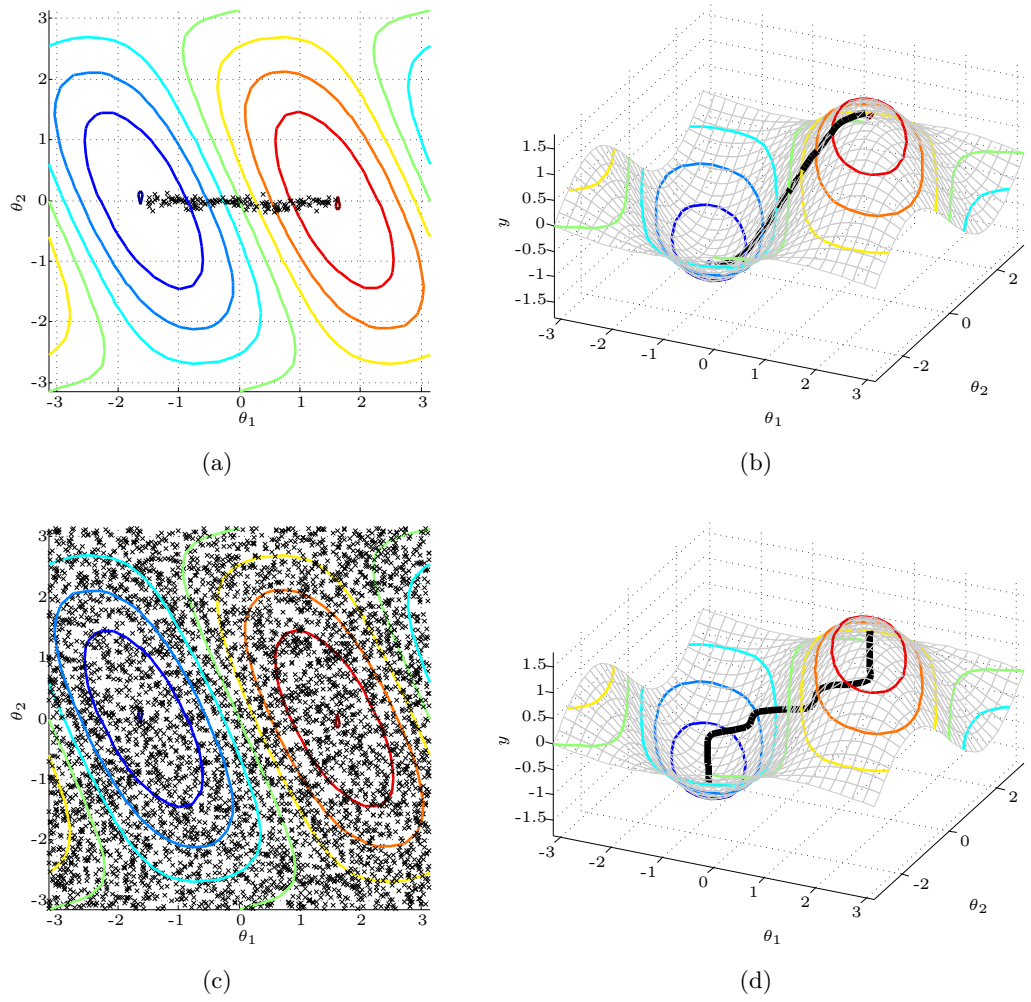


Figure 4.10: An example demonstrating the so-called "non-convexity problem" (Jordan and Rumelhart, 1992) for the case of learning the inverse kinematics mapping g for the planar robot, as defined in Section 4.2.1. When only drawing minimally redundant examples while sampling the motor space of the robot, a multilayer perceptron (MLP) is able to acquire the desired mapping. A valid training set consists only of training samples that cover each manifold of redundant solutions in a single point, as for example the 150 samples shown in (a). Using these samples to train an MLP with five hidden units in 10.000 training epochs results in the internal model shown in (b) that produces valid outputs for all targets, which can be seen as all outputs of the MLP (black line) lie on the actual manifold (shown in grey). In contrast, training the network from randomly drawn training samples, even when using as many as 5.000 training samples as shown in (c), results in an invalid mapping. The learning method regards multiple samples with the same value for the output variable as noise in the input variable, and averages between samples. This results in a mapping that produces invalid outputs for almost every input, as shown in (d).

and Steil have shown that after training, when presenting the network with an input in which the output components of the combined vector are clamped to target values, the network is attracted to a state in which the input components of the vector correspond to valid associations (Reinhart and Steil, 2011). Both approaches provide as output only a single vector as a solution. However, as described in the previous section, we require here a method that can restore entire sets of redundant solutions.

A straightforward way to learn and represent redundant mappings is to learn associative connections between population codes of input and output variables, which allows to associate a single value in one map with multiple values (or no value) in other maps. Since the method for the integration of internal models that has been proposed in Section 4.2 relies on the use of dynamic neural fields, which also require input to be in the form of population codes, using associative learning between population codes for the internal models would constitute a very intuitive implementation of the method.

While most connectionist models today use vectorized inputs, early attempts have been made by Rumelhart and McClelland to model the synaptic basis of associative learning in cerebral cortex using networks of "Sigma-Pi units" (Mel and Koch, 1989; Rumelhart and McClelland, 1986), which store information about co-activations of input units in the connection weights, allowing to restore multiple values as activation landscapes in response to a query. More recently, the use of sigma-pi units has been picked up by Weber and Wermter, who used a SOM-like learning algorithm to learn invariance in input signals in an unsupervised fashion (Weber and Wermter, 2007). They used their model to learn a self-organized representation of a head-centric coordinate system, based on pairs of inputs consisting of the angular configuration of the neck and the retinal position of a stimulus. After training, Weber and Wermter's sigma-pi SOM can reproduce for any point in head-centric coordinates co-activations of sets of solutions in the neural fields for angular configurations and retinal positions.

A neural network architecture that is essentially similar to networks of sigma-pi units has been used by Butz et al. in their computational model of arm movement control, the "sensorimotor, unsupervised, redundancy-resolving control architecture" (SURE_REACH; Butz et al., 2007a; Herbort et al., 2010), for restoring redundant solutions to kinematics tasks. Using a simulation of a planar arm with three rotational joints, Butz et al. show that their model can be used to restore all redundant arm configurations for a target hand location that the robot has encountered during training, by using an associative memory between population coding neurons, which represent on the one hand the space of arm configurations, i.e. joint positions, and on the other hand the position of the hand in Cartesian coordinates. When querying the network for a target hand location by activating the according population coding neurons for Cartesian coordinates, the network in response activates the associated neurons coding for arm configurations, resulting in an activation landscape representing the known solutions (similar to the input activation for the dynamic neural field shown in Figure 4.7). Butz et al. use dynamic programming to transform this activation landscape into an activation of the whole space of arm configurations, such that a gradient of activation points from any arm configuration to the nearest configuration that brings

the hand to the target location. Furthermore, Butz et al. use an inhibition of the activation of units representing arm configurations to constrain the selection of the final arm posture, for example by inhibiting all units but those that correspond to configurations in which one joint has a certain angle. Also, the positions of obstacles can be transformed from Cartesian coordinates into the arm's joint coordinates via the kinematic transformation stored in the network weights, which allows to use Butz et al.'s network for planning a collision-free trajectory of the hand by inhibiting all neurons that correspond to postures in which the hand would be colliding with an obstacle (Butz et al., 2007a).

An obvious drawback of using networks of sigma-pi units in conjunction with population coding in regular neural fields is that the number of possible synaptic connections in the network increases exponentially with the number of input dimensions. There are various ways possible to address this issue, for example by introducing self-organization in the neural fields, or by using more complex receptive fields with adaptive parametrization. However, any such steps would shift the focus of research, away from the question of how to integrate internal models, towards developing more efficient learning tools. This would also add to the complexity of simulation results by introducing parameters that are inherent to the learning method, making an interpretation more difficult, while qualitatively not changing the properties of the integration method. Therefore, to remain true to the research topic of developing a cognitive architecture based on dynamic building blocks, as proposed in Chapter 3, the following sections will describe and evaluate an implementation of the method for integration introduced in Section 4.2, using networks of sigma-pi units. To make the implementation capable of coping with a level of complexity that is more interesting, another strategy is followed to reduce computational cost, without changing the network architecture. It is based on the observation that while the number of *possible* synaptic connections will grow exponentially with the number of input dimensions, so does the number of zero-weights. Thus, the number of *necessary* synaptic connections is much lower. Therefore, the network naturally lends itself to an implementation using a sparse representation, in which only existing (i.e. non-zero) synaptic connections are simulated. A similar strategy can also be found in the brain: Neurons in the young infant's brain initially are massively interconnected, before a large amount of connections and neurons become pruned as they are not used in experiencing the environment, and only the necessary connectivity is kept (Edelman, 1987). Using a sparse implementation of networks of sigma-pi units allows to use the model in more complex settings, such as controlling the arm of the iCub humanoid robot, which has been used for the simulation experiments as will be described in Section 4.4.

4.3.1 Networks of Sigma-Pi Units

Figure 4.11 shows a schematic view of a network of sigma-pi units. For every input variable x_i there is a set of t_i input units with receptive fields covering the domains of the inputs, which receive their activations $u_{i,j}$, $j \in \{1, \ldots, t_i\}$ through population coding, as in Equation 4.8. Synaptic connections with associated sigma-pi weights

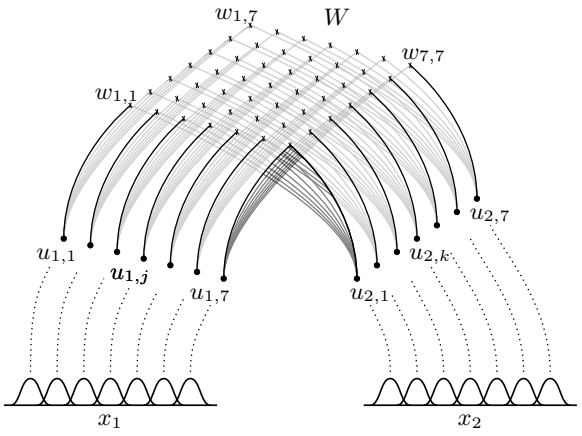


Figure 4.11: Schematic view of a network of sigma-pi units, for simplicity in the case of two one-dimensional inputs. Two sets of input neurons have their receptive fields in the domains of the inputs x_1 and x_2, and receive activations $u_{1,j}$ and $u_{2,k}$, respectively. Possible synaptic connections with associated sigma-pi weights $w_{j,k}$ are shown as a grid of crosses. Each unit in each set of input neurons can be connected to every other unit in the remaining sets of input units. Note that there are no lateral connections, and crosses in the grid only represent possible connections for which the network can store sigma-pi weights.

combine exactly one neuron from every set of input units. Thus, if $S_i = \{1, \ldots, t_i\}$ is the set of indexes for input neurons corresponding to the i-th input, then each element s in the Cartesian product S of the sets S_i,

$$S = S_1 \times \cdots \times S_n \tag{4.12}$$

$$= \{s = (s_1, \ldots, s_n) \mid s_i \in S_i\}, \tag{4.13}$$

corresponds to a possible synaptic connection w_s between units $u_{1,s_1}, \ldots, u_{n,s_n}$.

Networks of sigma-pi units belong to the class of "higher order" neural networks, as the simple additive units of linear feed-forward neural networks are extended by multiplicative connections. Thus, whereas in first-order neural networks the net input to a unit is given by

$$net_i = \sum_j w_{ij} u_j, \tag{4.14}$$

for the sigma-pi units the activation function includes the multiplication of inputs,

$$net_{i,j} = \sum_{s \in S,\ s_i = j} w_s u_{1,s_1} \ldots u_{n,s_n} \tag{4.15}$$

$$= \sum_{s \in S,\ s_i = j} w_s \prod_{m=1}^{n} u_{m,s_m}. \tag{4.16}$$

The introduction of these multiplicative connections allows units to *gate* one another (Rumelhart and McClelland, 1986): If one unit has zero activation, then the activation of other units in the multiplicative connection have no effect.

In the implementation used in this work, the sum and product operators were replaced by the max and min operators, respectively, to avoid the need for normalization of network responses. Thus, the modified net input to a unit is given by

$$net_{i,j} = \max_{s \in S,\ s_i = j} \left(w_s \cdot \min_{m=1}^{n} (u_{m,s_m}) \right). \tag{4.17}$$

We want the network weights $w_s, s \in S$, to reflect the amount to which the input units determined by s tend to be co-active during training. Thus, if the neurons are always activated together, then the weight should adopt a high value, and if one or more units are never active along with the others during training, then the weight should be zero (i.e. there is no connection). If whenever one of the neurons was active, only half of the time also all the other neurons determined by s were also co-activated, then the connection weight should have a value around 0.5.

To achieve this, we can use a simple Hebbian learning rule. Given a training set of tuples of input vectors $(x_1, \ldots, x_n)$, the input neurons are activated according to Equation 4.8. The resulting activations $u_{i,j}$ are used to compute the network activation for all $s \in S$ (cf. Equation 4.16), which is then used to update the network weights as

$$\delta w_s = \lambda \cdot \prod_{m=1}^{n} u_{m,s_m} \tag{4.18}$$

with learning rate λ. Again, for the implementation used in this work, the product operator was replaced with the min operator, giving

$$\delta w_s = \lambda \cdot \min_{m=1}^{n} u_{m,s_m}. \tag{4.19}$$

Furthermore, one-shot learning from training samples can be realized by omitting λ in the update rule (i.e., setting $\lambda = 1$), and updating the weights according to

$$w_s^t = \max(w_s^{t-1}, \delta w_s^t), \tag{4.20}$$

where w_s^{t-1} is the weight before, and w_s^t is the weight after processing the t-th training sample.

After training, we want to query the network by specifying a task description in terms of target values for one or several input variables, and want to retrieve possible values for the remaining variables in the form of population codes representing sets of redundant solutions. We specify which variables to constrain by defining a set $Q \subseteq \{1, \ldots, n\}$, corresponding to the indexes of respective input domains. Given the notation of S in Equation 4.13 and the net input in Equation 4.16, we can formulate the network query as

$$\widetilde{u}_{i,j} = \sum_{s \in S,\ s_i = j} w_s \prod_{m \in Q} u_{m,s_m}, \tag{4.21}$$

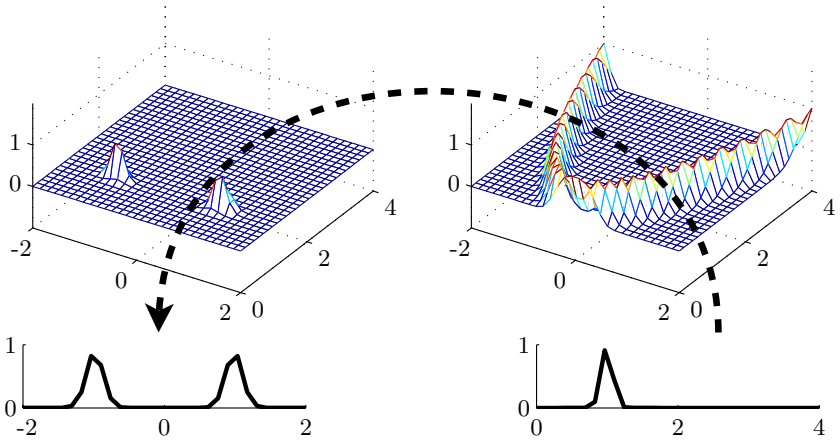


Figure 4.12: Graphical interpretation for the network query using the example of a network that has learned the function $y = x^2$ in the interval $x \in [-2, 2]$. To retrieve all solutions for $x^2 = 1$, the population of input neurons in the y domain are activated for $y = 1$, see bottom right plot. This activation is fed into the network and after the multiplication with the synaptic weights corresponds to an activation along the synaptic connections that can be seen in the top left plot. The net input to each unit in the readout is computed by accumulating this activity from all points on the grid to which it is connected.

where $u_{i,j}$ is the input activation that was specifies for the query for all units in the sets of input units given by Q, and $\widetilde{u}_{i,j}$ is the activation value that was retrieved from the network in response to the query. The activation of a unit is a sum over the activation of all elements s of the Cartesian product S in which that unit is itself a member, i.e. $s_i = j$. For all of these elements we compute the product of the activations of the units that were specified in the query, weighted by the connection weight w_s. Again, in the implementation used for this work a modified version was used, which is

$$\widetilde{u}_{i,j} = \max_{s \in S \;\; s_i = j} \left(w_s \cdot \min_{m \in Q} \left(u_{m,s_m} \right) \right). \tag{4.22}$$

There is an intuitive graphical interpretation for this query, see Figure 4.12. In the simple case where there are only two sets of input neurons, the network weights can be arranged in a planar grid, such that each knot in the grid represents the multiplication of two neurons (cf. also Figure 4.11). Thus, all knots in the grid together represent the Cartesian product of the sets of input neurons. If in a query we specify activations of input neurons in one domain and compute the product of the activations with the associated weights, we get an activation along the synaptic connections, such that we have one value for each of the knots in the grid. In the next step, we accumulate for each unit all the values along the line of knots that are connected to that unit, which gives us the retrieved activation value of the unit as the response of the query. The same picture can easily be extended to the higher dimensional case, in which there are more than two inputs or where the inputs have more than a single dimension. Here, the

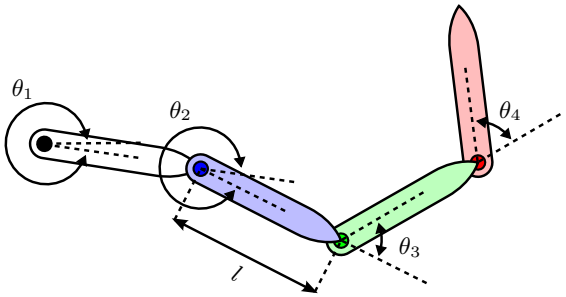


Figure 4.13: Schematic drawing of the kinematic chain used for the analyzes described in the text. The chain is composed of n segments, with $n \in 2, 3, 4$, ending with either the blue, green or red segment, respectively. All segments are of equal length l, with a total length of the chain of 1.

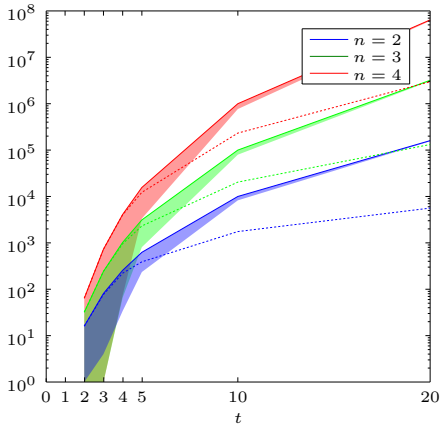
network weights are arranged in a hyper-cube instead of a grid, and the line of knots corresponds to a slice through the hyper-cube.

4.3.2 Evaluation of the Sparsity in Networks of Sigma-Pi Units when Learning Kinematics Models

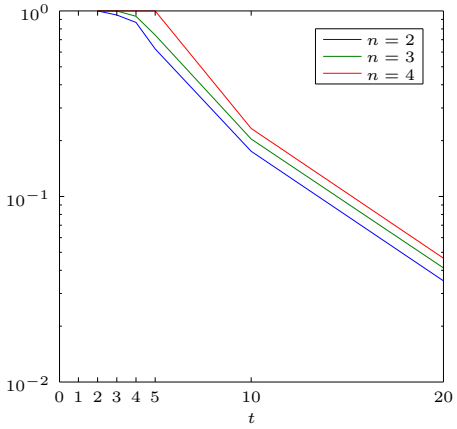
To investigate the number of synaptic connections that receive non-zero weights in the training, a simulation of a simple kinematic chain was used, see Figure 4.13. A number of equally long segments is connected via rotational joints. The length of the segments is set to $l = \frac{1}{n}$, where n is the number of degrees of freedom (i.e. segments) in the kinematic chain. Each joint is allowed to move freely into any angle from 0° to 360°, without considering any form of self-collision.

The space of joint configurations $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$, $n \in \{2, 3, 4\}$, was represented by a field of population coding neurons, with receptive field centers distributed uniformly in the n-dimensional interval of values for $\boldsymbol{\theta}$ with $\theta_i \in [-\pi, \pi]$. The Cartesian space of positions for the end-point of the kinematic chain was represented by a two-dimensional neural field, also with receptive field centers uniformly distributed to cover the domain of $\mathbf{x} = (x_1, x_2)$, $x_1, x_2 \in [-1, 1]$, which covers the region of locations that the end-point of the kinematic chain can reach. The number of neurons used in each dimension was varied, taking values $t \in \{2, 3, 4, 5, 10, 20\}$. Thus, in the case of for example $n = 3$ and $t = 5$, the neural field coding for joint configurations consisted of $t^n = 125$ neurons, and the neural field coding for end-point positions consisted of $t^2 = 25$ neurons. The number of possible synaptic connections in the corresponding network of sigma-pi was $t^{n+2} = 125 \cdot 25 = 3125$. For $n = 4$ and $t = 20$, the total number of possible synaptic connections is 64.000.000.

To keep the number of connections low, the Gaussian kernel for the population coding (see Equation 4.9) was replaced by a transformation of the input vector into barycentric coordinates of the surrounding hypercube of receptive field centers. This was done to ensure that an input would not activate an unnecessarily large number of



(a) Total number of possible synaptic connections (solid lines) and actual number of non-zero weights after training (shown both as dashed lines, and as colored regions below solid lines).



(b) Ratio between the number of non-zero weights after training and the total number of possible synaptic connections.

Figure 4.14: Comparison between the total number of possible synaptic connections in a network of sigma-pi units, and the number of non-zero weights after training. Blue, green and red graphs correspond to the end-point of the second, third and fourth segment, respectively (cf. Figure 4.13). Solid lines in (a) show the total number of possible synaptic connections, dotted lines show the number of non-zero weights after training. Colored regions under the solid lines also correspond to the percentage of non-zero weights, to visualize the relationship between zero and non-zero weights in the trained networks.

neurons due to a too large kernel size, but instead would activate either 2^n neurons, or only one, if the input coincided exactly with a receptive field center. The mathematical details for the transformation into barycentric coordinates are given in Appendix A.

To train the connection weights of the sigma-pi units, the n-dimensional space of configurations $\boldsymbol{\theta}$ was sampled, input units activated and connection weights updated according to Equation 4.20. The posture space was sampled for 30 values along each dimension, thus yielding training sets of 30^n samples (900, 27.000 and 810.000 samples for 2, 3 and 4 joints, respectively).

Figure 4.14(a) shows for all combinations of n and t the total number of possible synaptic connections, and the proportions of zero and non-zero weights after training. It can be seen that while the number of non-zero weights does increase exponentially with the number of input dimensions, it only takes up a fractional amount of the total number of possible connections, when 10 or more neurons are used to cover each dimension. This can also be seen in Figure 4.14(b), which shows the ratio between the total number of possible connections and the number of non-zero weights.

To see how many neurons are required in every dimension to represent the kinematic function to a sufficient degree of precision, each network was queried for the forward kinematics. Since the forward kinematics is a proper function, mapping each posture $\boldsymbol{\theta}$ onto exactly one end-point position $\mathbf{x}$, it was not necessary to select among sets of

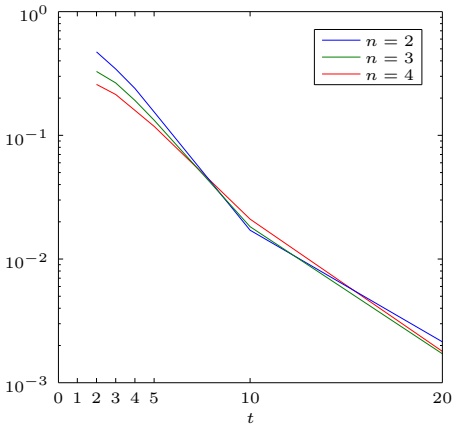


Figure 4.15: Mean squared errors obtained with different network sizes and numbers of segments in the kinematic chain. See text for explanation.

solutions. Thus, the responses of the trained networks were directly used to readout estimations of end-point positions for different postures. The posture space was again sampled with 30 values in each dimension. The estimated end-point position for each posture was compared with the true end-point position, and the mean squared error was computed for all samples as

$$\mathrm{MSE}_{n,t} = \frac{1}{30^n} \sum_{k=1}^{30^n} \|\widetilde{\mathbf{x}}_{k,t} - \mathbf{x}_k\|^2 \,, \tag{4.23}$$

where $\mathbf{x}_k$ is the true end-point position for the k-th sample configuration, and $\widetilde{\mathbf{x}}_{k,t}$ is its estimate, which was readout from the query of the corresponding network. Figure 4.15 shows the obtained values for the mean squared error for the different combinations of n and t. It can be seen that the mean squared error behaves similarly for different values of n, starting off high and decreasing exponentially with the number of neurons used in each dimension. With 5 or less neurons, the error remains rather high (above 0.1, which corresponds to 10% of the total length of the kinematic chain). With 10 neurons in each dimension, the error lies between 0.01 and 0.03 for all values of n, which can be deemed acceptable. Note that these results would correspond to open-loop control, while more precise results can be obtained through online adjustment of the posture using sensory feedback, which will be discussed later in Section 4.3.4.

Table 4.1 summarizes the results of the analyzes for all combinations of t and n.

Networks of sigma-pi units represent a straightforward way to implement internal model learning with the possibility to restore redundant solutions. Inputs are transformed into population codes and associations learned between co-activated input neurons. This allows to train networks for any form of internal model (cf. Section 4.1), by simply representing all inputs and outputs of the internal model as population codes. However, networks of sigma-pi units suffer from the exponential growth of the number of possible connections with the number of input dimensions. Using a sparse imple-

	Total number of possible synaptic connections			Number of non-zero synaptic weights (ratio between non-zero weights and number of possible connections)		
t	$n = 2$	$n = 3$	$n = 4$	$n = 2$	$n = 3$	$n = 4$
2	16	32	64	16 (1)	32 (1)	64 (1)
3	81	243	729	77 (0.951)	243 (1)	729 (1)
4	256	1.024	4.096	222 (0.867)	958 (0.936)	4.022 (0.982)
5	625	3.125	15.625	390 (0.624)	2.316 (0.741)	12.282 (0.786)
10	10.000	100.000	1.000.000	1.752 (0.175)	20.404 (0.204)	232.656 (0.233)
20	160.000	3.200.000	64.000.000	5.616 (0.035)	131.908 (0.041)	2.976.312 (0.047)

Table 4.1: Summary of results for the analyzes of the relationship between the total number of possible synaptic connections and the number of non-zero weights after training in a network of sigma-pi units.

mentation makes the simulation of larger networks more feasible, since the percentage of non-zero weights in a trained network decreases exponentially with the number of neurons used to cover each dimension, as was demonstrated in the last section.

Still, also the number of non-zero weights increases exponentially with the number of input dimensions. To make the network computationally more efficient, changes to the network properties would have to be made, for example by allowing for variable receptive fields, to cover larger regions in high-dimensional spaces where the mapping to be learned is uniform in character, e.g. close to linear. This would render the network more similar to learning methods that are based on the use of many locally-linear models (as described in the beginning of Section 4.3), and would allow to substantially reduce the number of necessary units.

Internal models are represented in networks of sigma-pi units by locally generalizing radial basis functions (Mel and Koch, 1989), meaning that generalization from training samples only happens within the radius of the receptive fields of input neurons. Therefore, the number of training samples that are necessary to train the network has to increase with the number of input neurons used, since otherwise (if the resolution of input neurons is higher than the resolution of training samples) the mapping would not be represented for spaces in between training samples. In the example application of learning a kinematic transformation, which was used for the analyzes above, a good trade-off between performance and cost would be to use around $t = 10$ neurons for each input dimension, which gives a rather low mean squared error, while not requiring to draw intractably many training samples for an accurate mapping to be learned.

4.3.3 Using Multiple Queries for Distributed Decision Making

In Section 4.2.4 it was discussed that the presented method for co-ordinating multiple dynamic neural fields is computationally more efficient than using a single high-dimensional dynamic neural field, which for the computation of an update would have a cost in time and space growing exponentially with the number of input dimensions. In contrast, the cost of using multiple one-dimensional dynamic neural fields only grows linearly with the number of input dimensions, both in time and space. For the gen-

eral case where ne pre-assumptions about the internal model are made other than the requirement that it can restore information about redundancies, a high-dimensional (non-dynamic) neural field is needed to represent the retrieved set of solutions from a query (cf. Figure 4.8). However it was also noted, that also this burdon can be lifted if the query of the internal model allows to incorporate output feedback in the form of the decisions of individual low-dimensional dynamic neural fields. This should allow to retrieve low-dimensional outputs from multiple queries of the internal model, instead of retrieving a high-dimensional output which is then projected into its component dimensions. This section will now demonstrate that networks of sigma-pi units belong to a class of internal models that allow exactly this kind of query, which incorporates the output feedback from the low-dimensional dynamic neural fields. Thus, overall the cost of the presented method for integrating internal models becomes linear in relation to the number of input dimensions, both in time and space.

The network query, as defined by Equations 4.21 and 4.22, is based on selecting a subset of dimensions from the input- and output-domains and providing target activations for the respective neurons. As such, it is not restricted to specifying targets only in dedicated "input" dimensions, but arbitrary combinations of dimensions can be chosen for a given query. To incorporate the output feedback of the lower-dimensional dynamic neural fields equivalently to how the feedback was used in the method presented in Section 4.2.3, transforming the decision of each dynamic neural field into corresponding candidate decisions for the respective other dynamic neural fields, the network query is slightly adapted as

$$\widetilde{u}_{i,j} = \sum_{s \in S,\ s_i = j} \left[w_s \cdot \left(a_0 + \sum_{m \in O} a_m u^{DNF}_{m,s_m} \right) \cdot \prod_{m \in Q} u_{m,s_m} \right], \tag{4.24}$$

or analogously using the max and min operators,

$$\widetilde{u}_{i,j} = \max_{s \in S,\ s_i = j} \left[w_s \cdot \left(a_0 + \sum_{m \in O} a_m u^{DNF}_{m,s_m} \right) \cdot \min_{m \in Q} u_{m,s_m} \right]. \tag{4.25}$$

Here, additionally to the set $Q \subseteq \{1, \ldots, n\}$ corresponding to the indexes of the respective input domains as in Equations 4.21 and 4.22, the set $O \subseteq \{1, \ldots, n\}$, $O \cap Q = \emptyset$ specifies input domains for which the decisions of low-dimensional dynamic neural fields should be incorporated in the query. Values u^{DNF} are the output activations of the corresponding dynamic neural fields. Weights a_i are used for individual inputs (cf. Figure 4.8), where a_0 is the weight associated to the query formulated for the input domain, and each a_m corresponds to the output of one dynamic neural field. Figure 4.16 demonstrates the result of using this adapted query in a toy example.

As already mentioned above, directly incorporating the decisions of low-dimensional dynamic neural fields into the query allows to retrieve the input for individual dynamic neural fields (cf. Section 4.2.3) without the need to first store an activation landscape in a high-dimensional (non-dynamic) neural field, which would then have to be projected into the respective component dimensions. Thus, instead of using a single query with

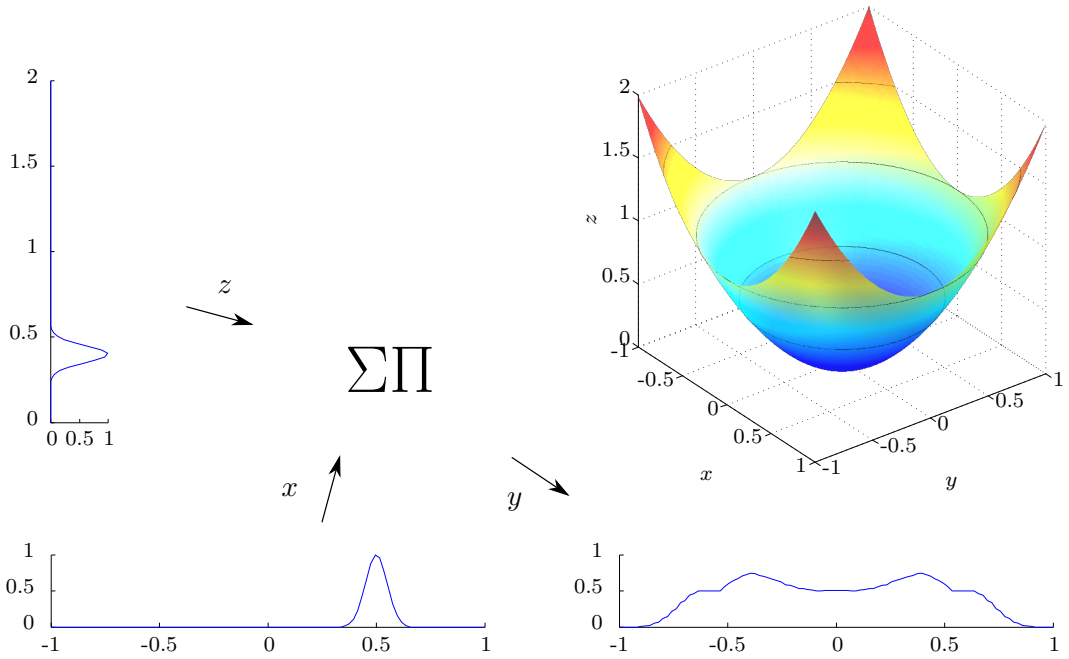


Figure 4.16: Example of using the adapted query, incorporating the decision of a low-dimensional dynamic neural field. A network of sigma-pi units (indicated as "ΣΠ") has learned the mapping shown in the upper right. A target value for the z-domain is specified, shown as the plot in the upper left. Using this input to query the network for solutions in the x-y-plane would result in a ring of activation in a two-dimensional (non-dynamic) neural field. This ring would then be projected into its two component dimensions, giving plateaus of equal activation, which would consecutively be amended to incorporate the feedback from individual dynamic neural fields (see Section 4.2.3). However, using the adapted query presented in this section, it is possible to directly incorporate the output of a dynamic neural field representing the x-domain, here shown as the plot in the lower left. The result of the adapted query is an activation landscape for the y-domain, shown in the lower right, where a plateau of equal activation corresponds to values for y that are candidate solutions to achieve the target value for z, and two "bumps" in the plateau correspond to the solutions that would co-ordinate the dynamic neural field representing the y-domain with the decision of the other dynamic neural field. Thus, the same result of retrieving solutions for individual output domain is possible without the use of a high-dimensional neural field as an intermediate step. Weights were chosen as $a_0 = 0.5$ for the input query, and $a_1 = 0.25$ for the output of the dynamic neural field.

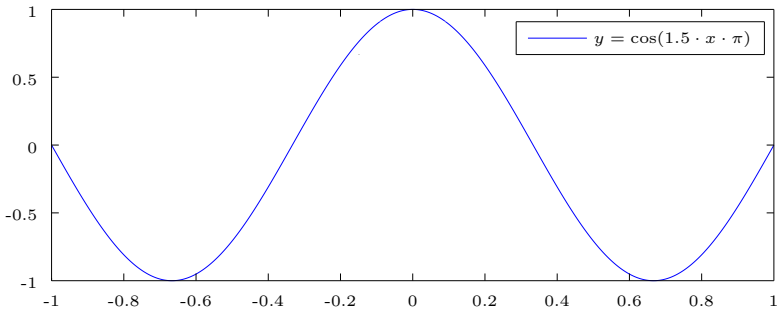


Figure 4.17: Plot of the function that is used in the description of the method for controlling sensorimotor systems using networks of sigma-pi units.

a computational cost in space growing exponentially with the number of input dimensions, the adapted version allows to query the network once for each required output, thus yielding a linear increase in complexity both in time and space.

4.3.4 Using Networks of Sigma-Pi Units for Accurate Robot Control

So far it was described in the last sections, how internal models can be integrated by using population codes for redundant solutions in conjunction with dynamic neural fields, and how internal models can be learned and redundant solutions restored using networks of sigma-pi units. Section 4.3.2 showed the accuracy of querying a trained network for an estimation of the forward kinematics, which corresponds to open-loop control. Employing sensory feedback and closed-loop control allows to solve tasks with higher precision, by transforming errors between target and actual values of input variables into changes for output variables. In networks of sigma-pi units this can be approximated as explained in the following (a toy example will be used for the description, for reasons of simplicity; an application to kinematic control of the arm of a humanoid robot, the iCub, will be described further down in Section 4.4).

Sigma-pi units do not explicitly code for directions in the connection weights, thus it is not possible to directly compute a mapping from desired changes to an input variable onto changes of an output variable. However, it is possible to use the population codes of the target value and of the current input to compute a difference in activation, and to transform this via the sigma-pi weights into an attractor landscape in the output space, which can be used to iteratively compensate for an initial control error.

As an example for the description, consider a network that should learn the function shown in Figure 4.17,

$$y = f(x) = \cos(1.5 \cdot x \cdot \pi), \tag{4.26}$$

and should be used to retrieve a value for x, given a target value for y. To represent values, population codes in one-dimensional neural fields are used for both variables with activations $u_{x,i}$ and $u_{y,i}$, respectively, according to Equation 4.8. To achieve a precise control of x to reach the target y^*, now two network queries are performed and

the responses combined, the first network query being as described above to select one among several redundant solutions but not being sufficiently accurate, the second query allowing for a local adjustment.

Thus, for the first query, the target value y^* is population coded as $\mathbf{u}_y(y^*)$ and is used for a network query, as described in Section 4.3.1, to generate the input $\widetilde{\mathbf{u}}_x$ for the dynamic neural field. The output of the dynamic neural field, $\widehat{\mathbf{u}}_x$, is a localized activation pattern of the units in the domain of x, such that when computing a target for x from this output as a linear combination of centers $m_{x,i}$, using activation levels as coefficients,

$$\widetilde{x} = \sum_{i=1}^{t_x} \widehat{u}_{x,i} \cdot m_{x,i} \tag{4.27}$$

will produce a result that lies close to, but is not necessarily equal to the target value y^*, due to imprecision in the learned internal model.

To compensate for this error, a second network query is performed using the difference of activations of population codes for the target value y^* and the current value y,

$$\mathbf{u}_y^{err} = \mathbf{u}_y(y^*) - \mathbf{u}_y(y), \tag{4.28}$$

which will generate an activation landscape $\widetilde{\mathbf{u}}_x^{err}$. If the current x is not close to a value where $f(x) = y^*$, this activation landscape corresponds to a positive activation of all redundant solutions, as in the network query described in Section 4.3.1, but with a negative activation for all values of x where the current value for y is achieved. However, when x is close to a solution (i.e. y almost equals y^*), the positive and the negative activations are overlapping, such that those parts of the positive activation that correspond more to the current value of y become inhibited, and only the parts of the positive activation that lie in the direction of y^* remain strong. By only using these activation values in the local neighborhood of the current x for a population readout, a local optimization can be achieved which allows to accurately reach the target y^*. Thus, we compute a second target for x as

$$\widetilde{x}^{err} = \sum_{i=1}^{t_x} H(u_{x,i}(x)) \cdot R(\widetilde{u}_{x,i}^{err}) \cdot m_{x,i}, \tag{4.29}$$

where $R(\cdot)$ is the ramp function

$$R(u) = \begin{cases} 0 & u < 0 \\ u & u \geq 0 \end{cases}, \tag{4.30}$$

and $H(\cdot)$ is the Heaviside step function

$$H(u) = \begin{cases} 0 & u < \epsilon \\ 1 & u \geq \epsilon \end{cases}, \tag{4.31}$$

which is used to ensure that only coefficients are used where the population code of the current value for x has activation values above a small threshold ϵ. Thus, only the

coefficients in the local neighborhood of the current value of x are used to compute a correction.

The two responses $\widetilde{x}$ and $\widetilde{x}^{err}$ should be combined in such a way, that the system uses the former to get close to the selected solution, and switches to the latter as soon as it is close to the solution, to perform a local error correction. This is modeled by computing the amount of overlap between activations in the output of the dynamic neural field and the population code for the current value of x. An overlap between the two activations indicates that x is close to the solution, and that it is safe to perform local error compensation. The overlap is computed using a sigmoid function as

$$\mathrm{sig}_x(\eta_x) = \frac{1}{1 + \exp\left(-b \cdot (\eta_x - \mu)\right)}, \tag{4.32}$$

where η_x is computed as

$$\eta_x = 1 - \frac{\sum_i H(\widehat{u}_{x,i}) \cdot u_{x,i}}{\sum_i u_{x,i}}, \tag{4.33}$$

such that $\mathrm{sig}_x(\eta_x)$ assumes values close to 1 as long as there is no overlap between activations, and values close to 0 when the activation of the population code for x is largely overlapping with the output activation of the dynamic neural field. The parametrization of the sigmoid function was chosen as $b = 50$ and $\mu = 0.9$, to get a steep slope near 1, so that the system quickly switches to a local error correction as soon as it is near to the selected solution.

The two outputs of the queries are combined using a simple linear combination that switches between the two outputs,

$$x^* \;=\; \mathrm{sig}_x(\eta_x) \cdot \widetilde{x} \;\;+\;\; (1 - \mathrm{sig}_x(\eta_x)) \cdot \widetilde{x}^{err}. \tag{4.34}$$

A change to the value of x that will reduce the current error can be computed as

$$dx = v \cdot \frac{x - x^*}{\|x - x^*\|}, \tag{4.35}$$

where the amplitude of change v is chosen to depend on the current discrepancy between y and y^*, again modeled as the amount of overlap, η_y, between activations (analogous to Equation 4.33), as

$$v = \gamma \cdot \max(\mathrm{sig}_y(\eta_y), \mathrm{sig}_x(\eta_x)), \tag{4.36}$$

where sig_y is parametrized using $b = 5$ and $\mu = 0$ and is shifted and scaled to assume values between -1 and 1, and γ is a constant scalar. Making v depend both on the overlap in the x- and y-domain assures that the amplitude of change is large as long as y deviates from y^*, but does not become slow if a solution is crossed that is different from the one chosen by the dynamic neural field. Applying the method to other scenarios and higher-dimensional cases is straightforward and possible without adaptation.

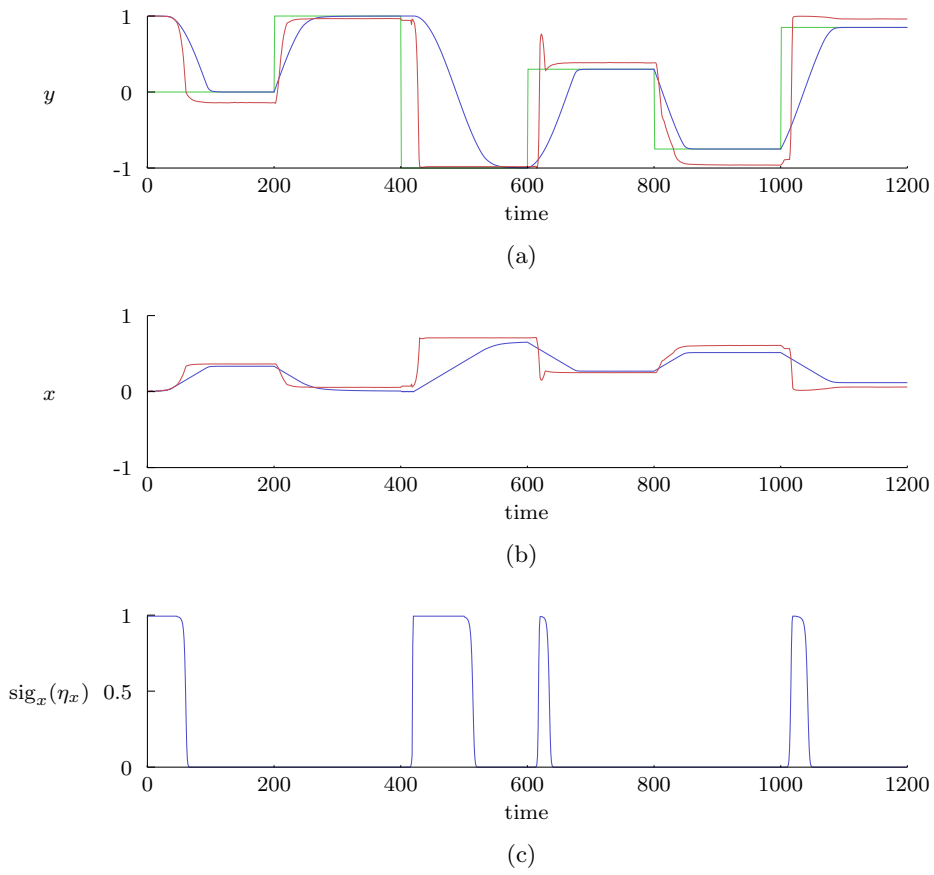


Figure 4.18: System behavior for the control of a simple simulated sensorimotor system. See text for description.

Evaluation

To evaluate the proposed method for local error compensation, two neural fields were used for population coding x and y, using $t_x = t_y = 15$ neurons in both fields. The output x was controlled using Equation 4.36, with $\gamma = 0.005$. A network of sigma-pi units was trained to learn the mapping f, as defined in Equation 4.26 and plotted in Figure 4.17, using 100 samples that were equally distributed in the interval $x = [-1, 1]$. The dynamic neural field was parametrized with $w_E = 4.5$, $\tau = 5$, $h = -0.027$ and $w_I = 1$, with a distance between neighboring neurons i and j in the neural field was $d(i, j) = 1$ (cf. Equation 4.5). A new target y^* was given to the system every 200 time steps.

In Figure 4.18, the evolution of x and y is shown when applying the described method for error correction, as opposed to directly using the output of the dynamic neural field for the control of x. It can be seen that the system behavior for y, shown in Figure 4.18(a) as a blue line, reliably reaches the target values y^* that are provided,

shown in the plot as a green line. In contrast, if the output of the dynamic neural field was directly used, the plateaus of the red line would be reached, which in most cases do not coincide with the target values (a value for x was readout from the output activation, and $y = f(x)$ was directly computed, without using a control scheme, which is why the red line approaches its stable states more quickly that the blue line). The corresponding values of x are shown in Figure 4.18(b). Additionally, $\text{sig}_x(\eta_x)$ is plotted in Figure 4.18(c), to show at which times the system uses the dynamic neural field output to get close to a selected solution, and at which times it performs local error correction. It can be seen that, if the dynamic neural field picks a new solution which is close to the last one, the system can directly apply local error correction to reach the new solution. Only if a new solution is not in the local excitatory neighborhood of the peak activation in the dynamic neural field, and a new peak develops at a different location in the field instead of the old peak wandering off to the new solution, then the system response first approaches the new solution before the local error correction is initiated.

Figure 4.19 shows both the result of the first network query, together with the activation of the dynamic neural field (Figure 4.19(a)), and the result of the second network query (Figure 4.19(b)) over time. The effect of local error correction can be seen as the peaks in Figure 4.19(b) slightly wander off from their initial positions, to compensate for errors, before diminishing when the error approaches zero, whereas the peaks in the activation of the dynamic neural field (Figure 4.19(a)) remain at their stable positions.

4.4 Simulation Experiment with the iCub Humanoid Robot

To demonstrate the applicability of the proposed methods for the integration of internal models (see Section 4.2.2) in a robotic setup, the method was implemented using networks of sigma-pi units as a learning technique (see Sections 4.3.1) together with the method for accurate robot control using networks of sigma-pi units (see Section 4.3.4). For the evaluation, a simulation of the iCub humanoid robot (Tikhanoff et al., 2008) was used.

The kinematics of the robot's right arm (i.e. shoulder and elbow joints) should be learned by the network, corresponding to a four-dimensional space of angular positions $\boldsymbol{\theta}$. This space was sampled along a regular grid by placing in each dimension 10 equally distributed grid points, resulting in $T = 10000$ configurations. The robot's arm was moved into the joint configurations and pairs of vectors $\left(\boldsymbol{\theta}^t, \mathbf{x}^t\right)$ was recorded as training samples.

To represent the Cartesian input vect , thusresultingin or of end-effector coordinates, a three-dimensional neural field of $42 \times 36 \times 41$ neurons was used. The receptive fields centers were initialized on a regular grid of positions that were equally distributed and covered the whole workspace of the robot. Similarly, a four-dimensional neural field of $10 \times 10 \times 10 \times 10$ neurons with receptive fields arranged on a grid that covered the domain of the arm angular position vector was used.

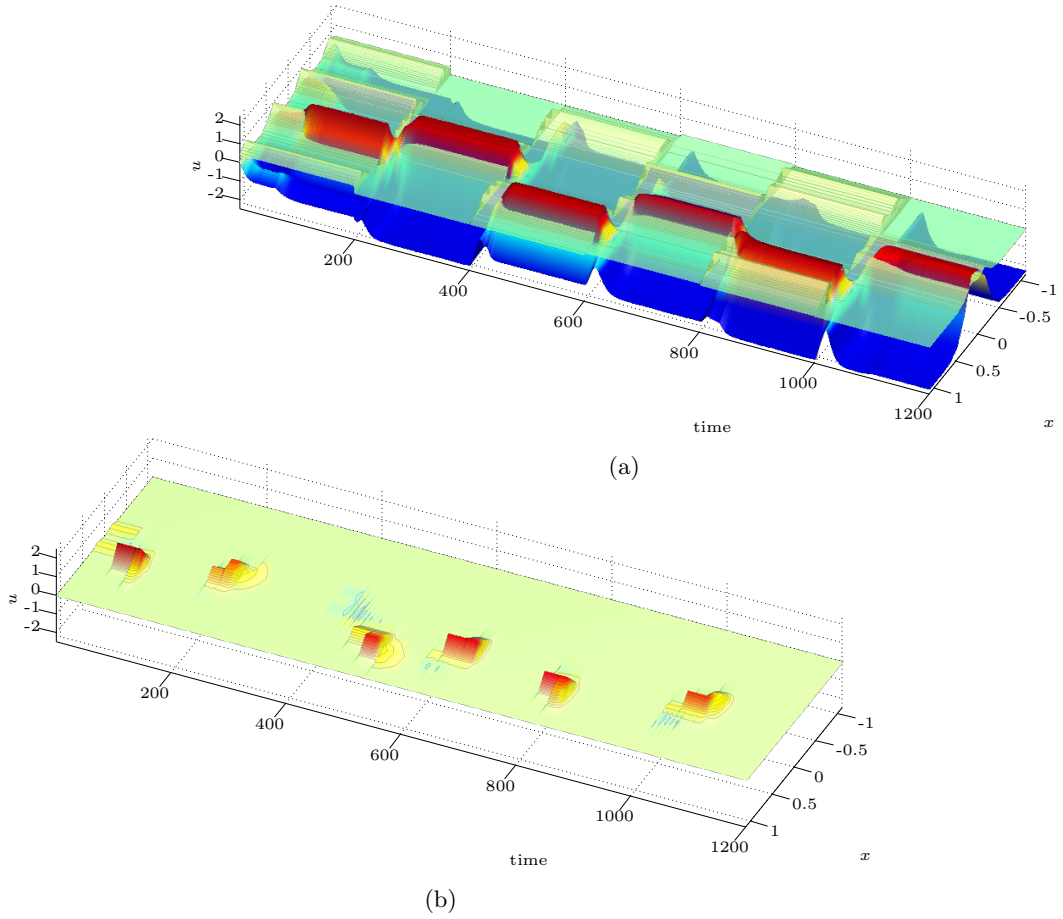


Figure 4.19: Results of querying the trained network of sigma-pi units, (a) as input to a dynamic neural field to select a solution, and 4.19(b) to compensate for control errors. See text for description.

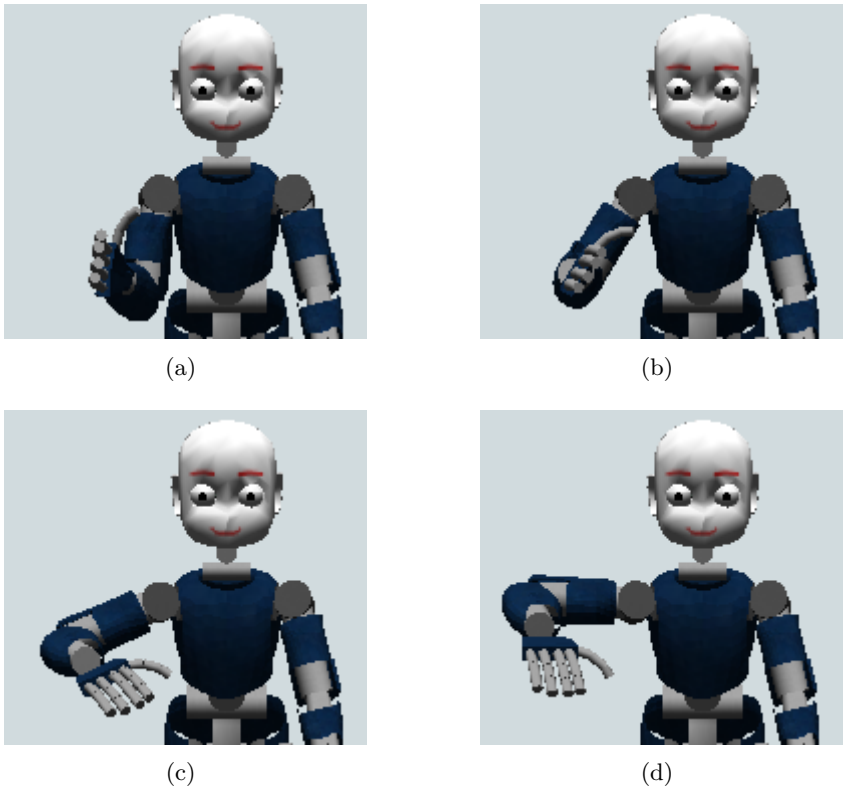
A network of sigma-pi units was used to learn associations between joint configuration vectors and Cartesian end-effector coordinates, as described in Section 4.3.1. A dynamic neural field representing the domain of arm configurations was initialized with the set of parameters $\tau = 15$, $h = -0.1$, $w_E = 1.26$, $w_I = 0.004$ and $\sigma = 0.6$, where the distance between two neighboring neurons i and j in the dynamic neural field was $d(i, j) = 1$ (cf. Equation 4.5).

To test the integration of the kinematics model with another internal model, a second network of sigma-pi units was introduced that should learn the mapping

$$f(\boldsymbol{\theta}) = \theta_2, \tag{4.37}$$

which simply maps every posture $\boldsymbol{\theta}$ onto the angular position of the second shoulder joint. Thus when querying the learned mapping for candidate postures that use a given

angular position for the second shoulder joint, the network response will be an activation representing the axis-parallel hyper plane for all postures along θ_2. A one-dimensional input map of 10 neurons with receptive fields covering the possible angular positions for the joint was used. Figure 4.20 shows example postures that were generated by giving the system the task to reach for a position in front of the robot, while keeping the second shoulder joint in different target angular positions. The system was able to generate solutions that satisfied both of the goals that it was given, i.e. it successfully moved the end-effector to the target point while staying in the specified angular position with the second shoulder joint. When the target angular position of the second shoulder joint was set such that the robot could not reach the target point with its hand with the given constraint, the system had to pick a solution for either the first or the second task. In this case, the second task was selected and the system drove the robot's second shoulder joint into the target angular position, while moving the end-effector away from the target position, as seen in Figure 4.20(d).



(a) (b)

(c) (d)

Figure 4.20: Several postures that were retrieved by the system for the task of bringing the robot's end-effector to one point while constraining one of the shoulder joints. The position of iCub's end-effector is at the center of its palm, which is kept at the target point in the queries shown in (a)–(c) In each new query, the constrained postural value for the shoulder joint is increased, until in the final query that is shown in (d) iCub cannot reach the target point with the constrained posture, because it has reached its joint limits.

4.5 Discussion

This chapter was concerned with the question of how to integrate internal models, to allow for a flexible selection of solutions that can satisfy multiple task specifications simultaneously. This research question was motivated by the goal to find a way to generically integrate building blocks in a cognitive architecture, without relying on using task- or domain-specific knowledge by design. More specifically, it was investigated how the outputs of multiple internal models can be combined for the generation of the system's behavior, in a way that solutions are favored to satisfy multiple tasks of the robot in parallel. This is in contrast with existing methods for the integration of internal models that serialize outputs of internal models to avoid conflicts or use linear combinations to average outputs, and instead follows the idea to assign priorities to candidate outputs, giving those outputs that satisfy the most tasks the highest priority. While other approaches from the literature pursue a similar idea, they are limited to low-dimensional problems (only one-dimensional output spaces have been used). The method presented in this chapter was demonstrated to also work with higher-dimensional problems, as for example in the control of the arm of the humanoid robot iCub.

The presented method is based on the use of dynamic neural fields as a winner-take-all selection mechanism, using population codes of redundant solutions as inputs. It was demonstrated that multiple low-dimensional dynamic neural fields can be coordinated for a distributed decision making process, which allows on the one hand for a more flexible integration of internal models, and on the other hand provides better scalability to higher-dimensional problems as compared to using a single, high-dimensional dynamic neural field. As population codes of candidate solutions are used as inputs to the dynamic neural fields, the method relies on a representation of internal models that is capable of learning and restoring redundant solutions. Here, one possible implementation was presented using networks of sigma-pi units, which is probably the most straight forward way of implementing the proposed method. An obvious drawback of this modeling choice is the fact that networks of sigma-pi units do not scale well with the number of input dimensions. Nevertheless, using a sparse implementation in which only connections with non-zero weights are simulated, it is possible to use the model also in non-trivial cases, as it was demonstrated that a trained model can be used to integrate two internal models for the control of iCub's arm, to select solutions for solving two tasks simultaneously.

With the proposed implementation it was demonstrated that a robot can learn internal models from sensorimotor experience, preserving information about redundancies. Both the proposed method for integrating internal models, as well as the specific implementation using networks of sigma-pi units are free from task- or domain-specific assumptions, and can thus be seen as a general model for the learning and integration of internal models. The model is neurally plausible and compatible with neurophysiological data (cf. Section 2.5.1; Mel and Koch, 1989).

A neural network architecture similar to networks of sigma-pi units has also been

used by Butz et al. in their model of arm movement control (Butz et al., 2007a). However, the usage of the network in this work differs from Butz et al.'s work, in several ways. First of all, Butz et al. use the network with the intention to model human arm movement control, correctly arguing that the representation of redundancies allows for a more flexible control, whereas here it is used as a means to learn and represent redundant solutions to sensorimotor tasks, for providing a concrete implementation of the proposed method for the integration of internal models. Second, Butz et al.'s model relies on transforming the retrieved activation landscapes of redundant solutions into a gradient of activation in the complete space of arm configurations, which requires to hold values for all neurons in memory at the same time. This is contradictory to the idea of using a sparse implementation for improving computational performance. Butz et al. rely on this step, because they model arm motor control as a dynamical system and use the angular configuration of the arm as control variable, which requires a series of motor commands along a planned trajectory to reach a target configuration, whereas in this work target arm configurations are directly used as the output for a controller. To enable the model to plan sequences of actions while keeping computational cost low would probably require to represent the input and output spaces at different levels of granularity, using coarser representations for planning while using a finer level for actual control. Furthermore, using a self-organizing representation for input and output spaces (e.g. Gläser et al., 2008) would also reduce the number of necessary connections. Finally, the usage of the network of sigma-pi units is extended in this work by a method to accurately compensate for initial control errors, which are due to imprecision in the learned internal models. Herbort et al. have proposed to use an additional control structure alongside the network of sigma-pi units, which shifts an internal copy of the target location for the hand in the opposite direction of the difference to compensate the error, and using the displaced internal copy of the target location as input to the network (Herbort et al., 2010). The model presented in this work does not rely on an additional control structure and provides a solution for local error correction that seems less ad-hoc. The here presented method for local error compensation could be seen as a model for the role of the cerebellum in fine control of limb movements: It is known that patients with cerebellar lesions suffer from dysmetria, causing them to overshoot or undershoot goal-directed movements (Pschyrembel and Dornblüth, 1975). In the presented model, this corresponds to the behavior of the system when not using the local error compensation (cf. Figure 4.18).

One positive aspect of using networks of sigma-pi units for learning is their ability of one-shot learning, allowing to store single observations with a simple update step. In contrast, condensed representations using fewer units allow for a decreased computational cost for storage and for querying the network. However, the training effort for such representations is usually dramatically increased, as learning a consolidated representation requires presenting inputs many times, to iteratively adapt the representation to the statistics of the input values. As proposed for example by Gläser, both rapid one-shot learning and slow statistical learning might be part of a single intertwined learning process in the brain, where observations first enter a short-term memory in

which a model for the input is learned via one-shot learning, which is then used to internally generate many training samples for a slow statistical learning method of a more efficient representation in long-term memory (Gläser, 2012, pp. 63–64).

However, it should be noted that the proposed method for the integration of internal models is independent of the choice of method used for the learning and representation of internal models, as long as it is capable of restoring sets of redundant solutions. Using networks of sigma-pi units on the one hand easily complies with the use of dynamic neural fields, which use population coded inputs, and on the other hand provide a neurally plausible model, even if it is computationally not very efficient on current standard computer hardware. For a computationally more efficient implementation, learning methods with locally linear models (D'Souza et al., 2001; Lopes and Damas, 2007) or Gaussian mixture models (Rasmussen, 2000) could be used.

5

Self-Organized Learning of Multiple Internal Models

In the preceding chapters, a cognitive architecture based on the use of internal models as generic building blocks was introduced, and methods were described for the learning of the building blocks and for their integration, enabling the system to solve multiple tasks simultaneously. However, there was an implicit assumption made for the learning of internal models: That training samples can be generated for each internal model through sensorimotor experience, by observing the consequences of actions. This cannot always be immediately assumed, as in a cognitive architecture a limited number of inputs necessarily needs to be shared among multiple internal models. This means, that the sensory feedback used for training the individual internal models arrives temporally interleaved at the inputs, without a predefined structure determining a priori, which feedback signals should be used to train which internal models.

As an example, consider a robot with an arm and a movable camera head. For it to be able to control its effectors, it should acquire at least an internal model for visuomotor control of the hand (i.e., how to reach for visually perceived target positions), as well as an internal model for its gaze control (i.e., how to direct its gaze to the positions of target visual stimuli, or to track moving objects). In the examples used in the last chapter, it was suggested that learning should be guided by sensorimotor exploration, by executing actions and observing the outcomes in the sensory feedback. However, how can the system know if an observed change in the environment has been caused by a movement of the arm, by a movement of the head, or both? Without a mechanism for assigning sensory feedback signals to individual internal models during training, successful learning is not possible in a natural setup. Imposing artificial constraints on the exploration process could be used to circumvent this problem in some cases, for example by only moving either the arm or the head, while keeping the respective other still. This would remove ambiguity for the assignment of changes in the sensory feedback to actions, but would be a very task-dependent and ad-hoc approach.

Since this work is aimed at developing a cognitive architecture based on the use of generic building blocks, instead of relying on a decomposition into specially tailored

modules, a solution would be desirable that is free from task- or domain-specific considerations and does not impose artificial constraints on the process of sensorimotor exploration. This chapter will therefore introduce a learning method that allows the learning of multiple internal models from a shared input to be self-organized. The relationship between actions and outcomes (i.e., which action was the cause for the observed outcome) is governed by a latent variable in the learning process. To estimate this latent variable, the proposed method follows the idea to put involved internal models in competition with each other, and assigns training samples (i.e., observed action-outcome pairs) among the internal models based on how well they can predict the outcome.

Using such a competition to implicitly estimate a latent variable has also been done by Wolpert and Kawato in the MOSAIC model (see Section 3.2.2), which learns multiple instances of the same action, each adapted to a different context (Wolpert and Kawato, 1998). During learning, each internal model in the MOSAIC model receives as input an afferent copy of the next motor command to be issued, as well as the current sensory feedback signal, and produces an estimation of the next sensory input. These estimates are compared with the actually observed sensory feedback after sending the command to the motors. Using the new input for gated error-driven learning, by adapting internal models to an amount relative to how good they predicted the sensory feedback, Wolpert and Kawato show that each instance of the internal model specializes to one particular context. For example, if the action to be learned is that of lifting an object, multiple instances could each specialize for a certain weight of the object to produce an appropriate amount of force during the execution of the action.

This chapter extends the approach of using a competitive learning mechanism for the estimation of a latent variable into the domain of multiple different internal models: While the MOSAIC model demonstrates that multiple specialized instances of the same action can be trained, by implicitly estimating a parameter of the action (such as the amount of force needed to lift an object with unknown wight), here it will be shown, that a learning method similar to the one employed in MOSAIC can be used to solve the problem of assigning outcomes to actions. This allows to train multiple different internal models using data from a single input, at which the sensory feedback for individual internal models arrives interleaved. The method will first be explained in Section 5.1 using a toy example. As an example application, in Section 5.3 the learning of two mappings related to the "body-schema" for a robot will be used and the method evaluated in simulation experiments with the humanoid robot iCub.

Parts of this chapter (in particular Sections 5.1 and 5.3) are based on (Hemion et al., 2011).

5.1 Bootstrapping the Learning of Internal Models by Exploiting Preliminary Model Predictions

Figure 5.1 shows a system in which two internal models each associate an input, x_1 or x_2 respectively, with a common feedback signal y. To learn the two internal models,

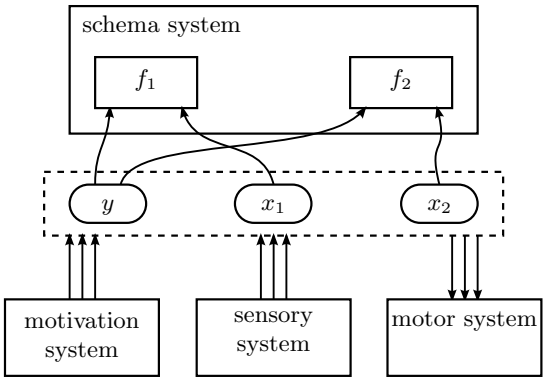


Figure 5.1: Example of a system in which multiple internal models, here f_1 and f_2, use a common sensory feedback signal, in this case y.

the system has to processes combinations of values for the three variables x_1, x_2 and y. However, as stated above, we assume that the system does not know a-priori whether it should associate a given y with x_1 or with x_2. Instead, it is assumed that values for y corresponding to either the one or the other internal model arrive temporally interleaved at the input.

For example, in the case of the robot that should learn its gaze control and its visuo-motor control, angular configurations of the head and the arm (the outputs of the system) have to be associated with head-centric positions (the common input). When gathering information by performing head and arm movements and monitoring the subsequent change in the head-centric position of an attended stimulus, the system does not know whether this new information should be used to train the internal model for gaze control, or the internal model for visuo-motor control. The former would be the case, if the robot had been attending an object in the background scene: Shifting its gaze direction transforms the head-centric position of background objects, while the robot's arm movement does not have any influence. If however the system had been attending its own hand, both the head movement and the arm movement would have influenced the observed change. Thus, at times when the system attends a background object, input data is related to (and should be used for training the internal model for) gaze control, whereas at times when the system attends its own hand, input data is related to visuo-motor control. Instances of the former and the latter case are temporally interleaved: As the system switches attention from one stimulus to another, subsequently arriving input data will correspond to either of the two cases, depending on whether the now attended stimulus is the own hand or not.

To self-organize the assignment of input data to individual internal models, we will exploit the fact that each internal model can only predict certain instances of incoming data well: Those that can be explained on the basis of the inputs that the internal model uses. For example, the internal model for gaze control is uninformed about arm movement commands and thus cannot predict how the hand will move in the visual field.

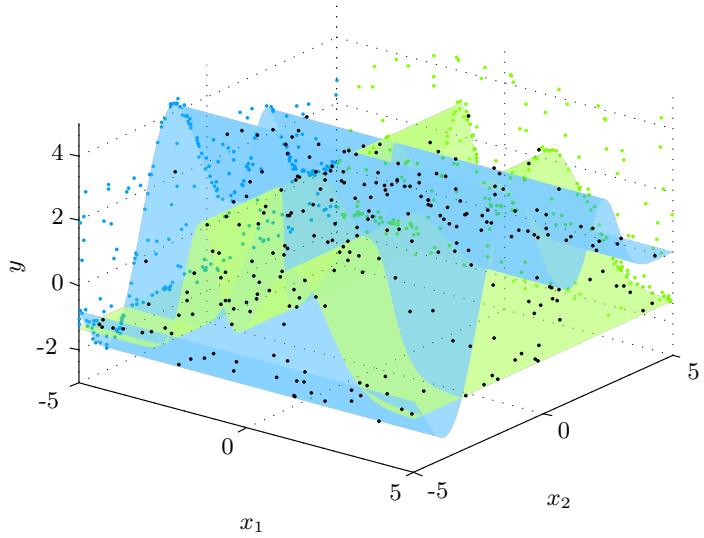


Figure 5.2: Visualization of the example used in the text, of two arbitrary mappings f_1 and f_2, shown in green and blue respectively. Samples drawn for a training set based on a latent variable from either of the two mappings are shown as black dots. Projecting the samples into the two-dimensional sub-spaces (x_1, y) and (x_2, y) results in the samples shown as green and blue dots, respectively, with a one-to-one correspondence between black, blue and green dots.

Only the internal model for visuo-motor control has the necessary information available for predicting these instances of input data. Each internal model should therefore be better at predicting certain classes of input data than the respective other internal models. Even if predictions will initially be rather inaccurate, we should expect them to be better than chance. Thus, if we assign samples from a training set respectively to the internal model that best predicted them, we expect internal models to receive more valid than invalid training samples, and thus improve their ability to predict subsequent training samples. Additionally, training samples that one internal model is predicting well are removed from the inputs to other internal models through the competition, which amounts to a reduction of noise. Thus, when iterating the competitive training process, we would expect each internal model to converge to the mapping that should be learned, and thus the assignment of training samples to self-organize.

Consider a simple example in three dimensions (see Figure 5.2), where the system should estimate two arbitrary mappings,

$$\begin{aligned} f_1 &: \mathbb{R} \to \mathbb{R}, \\ f_2 &: \mathbb{R} \to \mathbb{R}. \end{aligned} \tag{5.1}$$

We generate a training set by consecutively drawing two random inputs $x_1^t, x_2^t \in \mathbb{R}$, from some distribution (e.g. a uniform distribution over the interval $[-5, 5]$ in this example). We then compute y^t for each sample, based on a latent random variable

either as $y^t = f_1(x_1^t)$ or as $y^t = f_2(x_2^t)$ and combine the values as training samples for the internal models, as

$$\begin{array}{rcl} S_1 & = & \{\ (x_1^t, y^t)\ \mid\ t = 1, \ldots, T\ \} \text{ and} \\ S_2 & = & \{\ (x_2^t, y^t)\ \mid\ t = 1, \ldots, T\ \}, \end{array} \tag{5.2}$$

respectively. By omitting one of the input dimensions and thus projecting the training samples into two-dimensional sub-spaces, those training samples that were generated as $y^t = f_1(x_1^t)$ retain the necessary information for learning in the training set S_1 while amounting to random noise in the training set S_2, and vice versa for training samples that were generated as $y^t = f_2(x_2^t)$. Thus, based on the latent variable that was used for deciding whether f_1 or f_2 should be used to generate y^t, the training samples corresponding to the two individual mappings are interleaved in the training sets S_1 and S_2.

Using a competition between internal models allows to self-organize the assignment of the training samples in the following way. Estimates for the two mappings are trained as

$$\begin{array}{rcl} \widetilde{y}_1^t & = & \phi_1(w_1, x_1^t) \text{ and} \\ \widetilde{y}_2^t & = & \phi_2(w_2, x_2^t), \end{array} \tag{5.3}$$

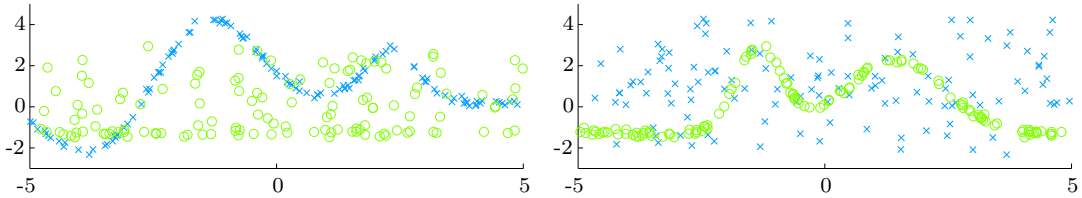
with some learning technique ϕ_i and corresponding model parameters w_i, for example using a feedforward neural network with associated connection weights. Initial sets of training samples, S_1^0 and S_2^0, are used to produce the first estimates. Then the following steps are iterated. New training sets S_1^k and S_2^k are generated as described above, and the two estimates are used to compute predictions $\widetilde{y}_1^t$ and $\widetilde{y}_2^t$ for all samples in the training sets. These predictions are compared with the actually observed outcomes y^t and squared errors are computed as

$$\mathrm{SE}_i^{t,k} = \|y^t - \phi_i(w_i^k, x_i^t)\|^2, \quad \forall\ \ (x_i^t, y^t) \in S_i^k, \tag{5.4}$$

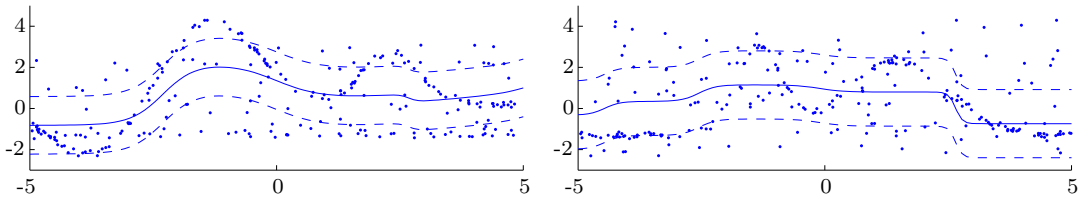
where w_i^k are the model parameters after iteration $k-1$. Then a binary weighting scheme is used for updating the estimates, to only use those training samples for which the estimate has produced the lowest squared error, according to

$$I_i^{t,k} = \begin{cases} 1 & \text{if } \forall\ j,\ i \neq j\colon \mathrm{SE}_i^{t,k} \leq \mathrm{SE}_j^{t,k}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

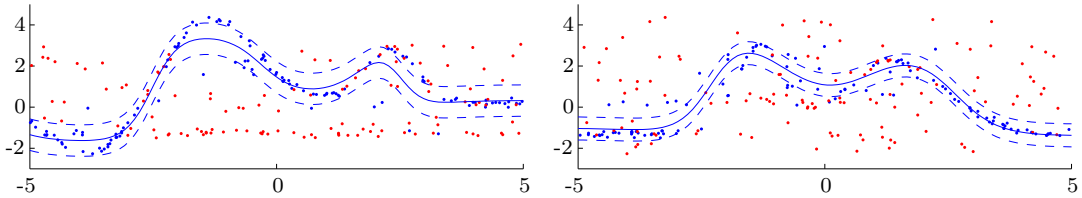
Figures 5.3 shows a visualization of a simulation of the method, using multilayer perceptrons as learning method, with 5 units in a single hidden layer. The two plots in Figure 5.3(a) show an example training set, where green circles correspond to one kind of observation and blue crosses correspond to another kind of observation. Note that there is a one-to-one correspondence of points in the two plots, as both plots correspond to projections from a three-dimensional data set into two two-dimensional subspaces. It can be seen that the green circles are random noise (with some non-trivial distribution in the vertical dimension) in the left plot, whereas the blue crosses are random noise in the right plot. Figure 5.3(b) shows the initialization of the estimates from the initial



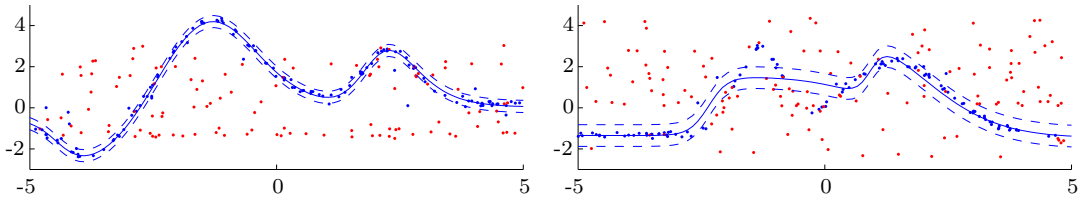
(a) Example data set



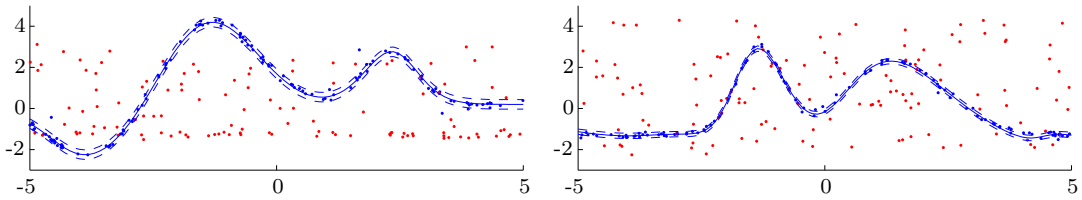
(b) Initialization



(c) Iteration 1



(d) Iteration 2



(e) Iteration 3

Figure 5.3: Simulation results for the example of learning estimates for two arbitrary functions, using the competitive learning method proposed in this chapter. See text for description.

training sets S_1^0 and S_2^0 as a solid blue line, with dashed lines indicating the mean squared errors of the estimates. Figures 5.3(c)–(e) show the results of the first three iterations of the described method. Using the estimates from the respective preceding iteration, training samples are weighted according to Equation 5.5. Training samples that received a weight of 1 and 0 are shown in blue and red, respectively, along with the result of updating the estimates using the weighted training samples. As it can be seen in Figure 5.3(c), already the initial estimate allows the weighting scheme to significantly reduce the amount of noise in the training sets. And after only three iterations, both estimates are very precise and only few training samples are wrongly assigned.

5.2 Handling Noise

In the above example it was assumed, that all observations originated from either of two functions, both of which could be learned by internal models. In the case of an actual robot based on the proposed cognitive architecture, this would amount to the assumption that the robot traines internal models for all observations that it encounters. For example, in the case of the robot that should learn its visuomotor control and gaze control, the two internal models would only be able to learn to predict all observations if the robot is situated in a static environment, where nothing but its own arm is moving: Shifting the own gaze while tracking a moving stimulus obviously produces different results than when tracking a static stimulus. Thus, for the system to be able to distinguish moving stimuli from static ones, it would have to train further internal models, which would have to learn to predict the different kinds of movement of stimuli. This is by no means an unreasonable assumption, as very young infants are already capable of distinguishing different forms of animate motion, which has lead researchers to suggest that concepts such as animacy, inanimacy and agency are among the earliest and most fundamental concepts that are acquired by infants during development (cf. Section 3.1.3; Mandler, 1992). Furthermore, the ability to predict the environment well results in a minimization of surprise, which has been proposed to be a driving force for learning in adaptive agents, as agents that do not minimize their surprise will sooner or later encounter potentially harmful surprising events (Friston et al., 2006; Kiebel et al., 2008).

Nevertheless, it is interesting to see how the method performs if observations are noisy, i.e. in the presence of observations that the system cannot learn to predict on the basis of its available information. To test this, the previous example was extended by drawing an additional set of samples $x_3^t \in \mathbb{R}$, which however were not provided as input to the system. Observations y^t were computed either as $f_1(x_1^t)$, $f_2(x_2^t)$ or $f_3(x_3^t)$, based on a latent variable. However, since the system was only provided with inputs x_1^t and x_2^t, but not with x_3^t, it could not learn to predict the values of y^t that were computed as $f_3(x_3^t)$, which instead added additional noise to the training input for both internal models.

For the system to handle noise, each internal model compares its predictions with actual observations and rejects all samples that deviate too much. Appart from this,

the previously described competition between internal models remains the same. To make the rejection adaptive to the current level of performance of each internal model, a threshold for rejection is used for each internal model that is proportional to the current mean squared error of the internal model. Thus, based on all samples that have been assigned to an internal model for training, the mean squared error is computed as

$$\mathrm{MSE}_i^k = \frac{1}{\sum_{t=1}^{T} I_i^{t,k}} \sum_{t=1}^{T} I_i^{t,k} \cdot \|y^t - \phi_i(w_i^k, x_i^t)\|^2, \quad (x_i^t, y^t) \in S_i^k, \tag{5.6}$$

and a threshold for rejection is updated for the next iteration as

$$\rho_i^{k+1} = \lambda \cdot \mathrm{MSE}_i^k, \tag{5.7}$$

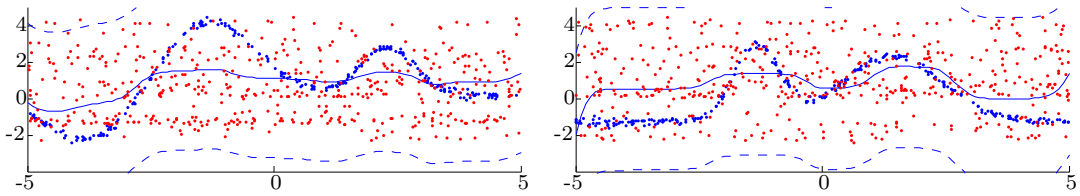
where $\lambda \in \mathbb{R}^{>0}$ is a parameter of the method. The weighting scheme is adapted to take the threshold for rejection into account as

$$I_i^{t,k} = \begin{cases} 1 & \text{if } \mathrm{SE}_i^{t,k} \leq \rho_i^k \text{ and } \forall\ j,\ i \neq j \colon \mathrm{SE}_i^{t,k} \leq \mathrm{SE}_j^{t,k}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.8}$$

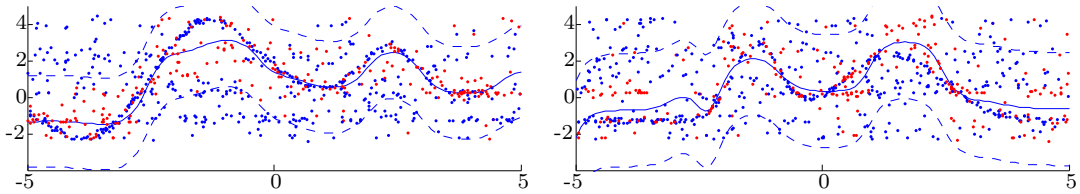
As before, for the initialization of the model parameters for both internal models, all samples from initial training sets S_1^0 and S_2^0 are used, without any form of assignment.

To test the method, two multilayer perceptrons were used as learning technique, each with 10 units in a single hidden layer. Before each iteration, 250 samples were generated using each $f_1(x_1^t)$, $f_2(x_2^t)$ and $f_3(x_3^t)$, resulting in a total of $T = 750$ training samples in each training set S_1^k and S_2^k. Figure 5.4 shows an example run of the method, using $\lambda = 7$. In Figure 5.4(a), the result of initializing the multilayer perceptrons using the initial training sets S_1^0 and S_2^0 is shown. Blue dots correspond to correctly assigned samples, and red dots correspond to wrongly assigned samples (on the one hand false negatives, i.e. samples that were rejected but should have been accepted, and on the other hand false positives, i.e. samples that were accepted but should have been rejected). The mean squared error of both internal models is indicated by dashed lines as a belt around the estimates. Figures 5.4(b)–(e) show the situation after iterations 1, 5, 10 and 25, respectively. It can be seen that the method is able to learn both mappings despite the additional noise, although convergence to the final solution is slower than when there is no additional noise (cf. Figure 5.3). Furthermore, while the learned estimates of both internal models are already very precise after iteration 10 (see Figure 5.4(b)), still some samples are wrongly assigned. However, the wrongly assigned samples correspond to "noisy" samples (i.e. samples that were generated using an input that is not known for the internal model) that happen to be similar to the function to be learned. For example, some of these samples will have been generated as $y^t = f_3(x_3^t)$, but by chance this happens to be close to the value of $f_1(x_1^t)$. Thus, even though these samples are de facto wrongly assigned samples, they only add a negligible amount of noise to the assigned training samples.

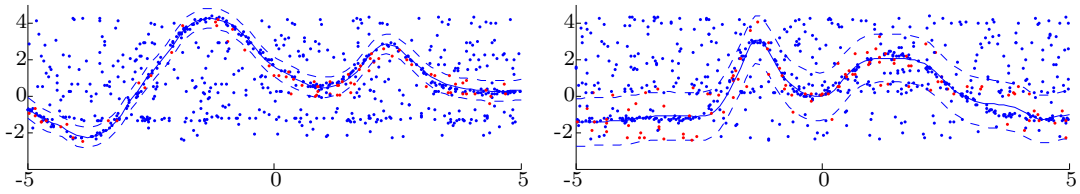
To evaluate the influence of the parameter λ on the learning performance, the training was systematically repeated for different values, using $\lambda \in \{1, \ldots, 13\}$. For each



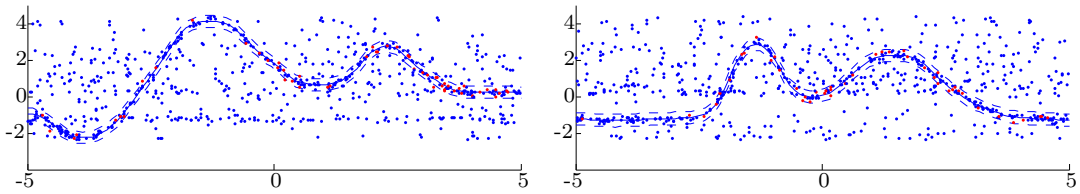
(a) Initialization



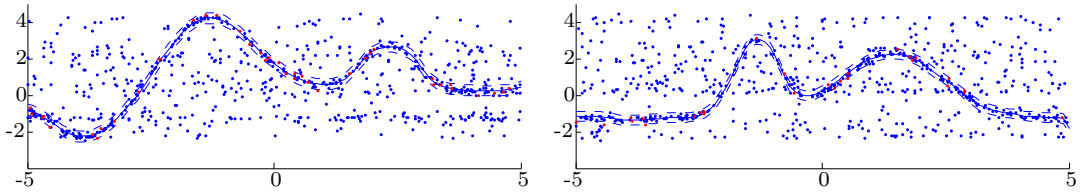
(b) Iteration 1



(c) Iteration 5



(d) Iteration 10



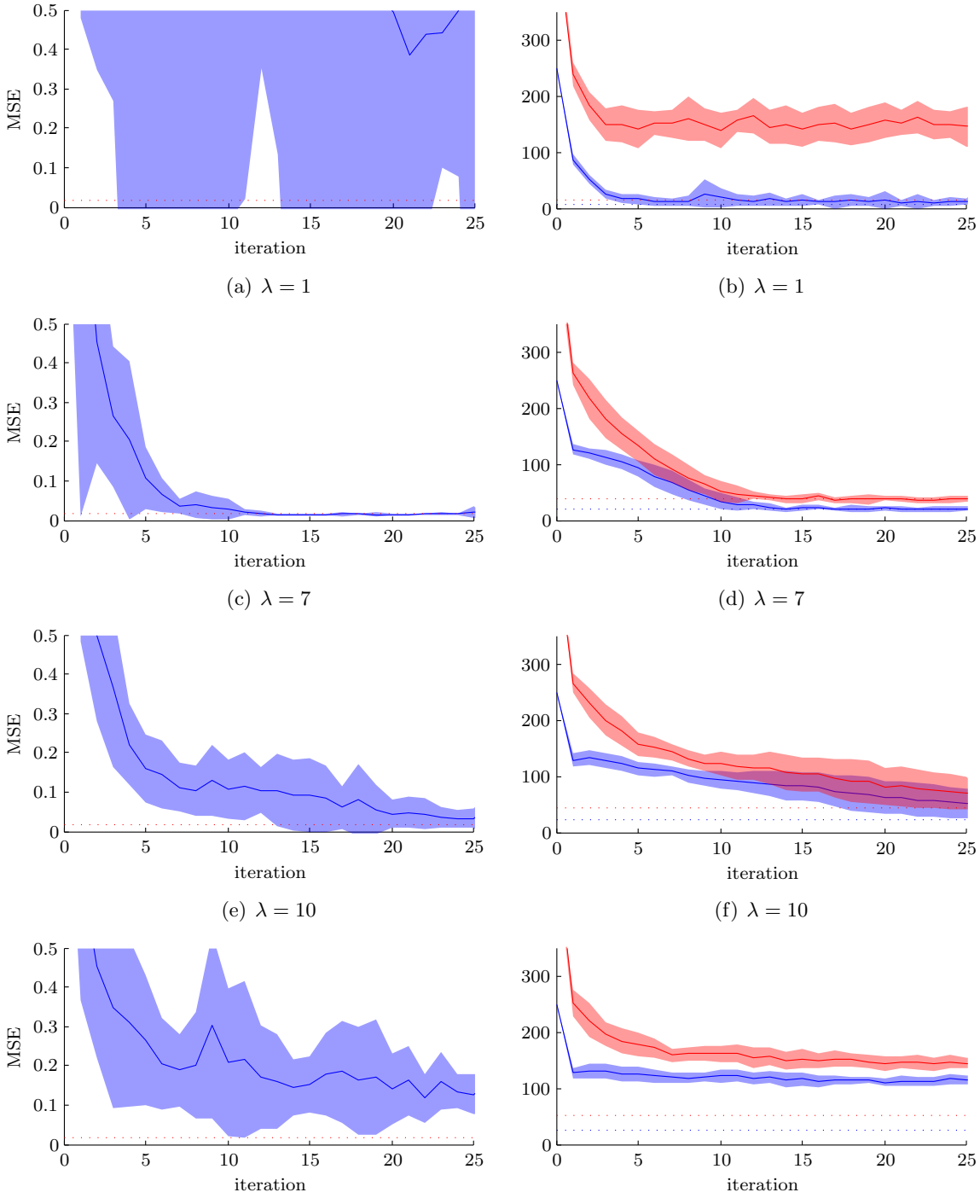
(e) Iteration 25

Figure 5.4: Simulation results for the example of learning estimates for two arbitrary functions, using the competitive learning method proposed in this chapter. See text for description.

value, 15 independent training trials were performed, each for 25 iterations. Figure 5.5 shows the mean learning performance of the internal model learning f_1, exemplarily for the values 1, 7, 10 and 13. Figures 5.5(a), (c), (e) and (g) show the development of the mean squared error, with standard deviation indicated as a colored belt around the mean value. For comparison, 50 feedforward networks (with the same network architecture as the ones used for the internal models) were trained using only correct training samples, and their mean performance was computed to obtain a reference performance level, which is shown as a dotted red line in the plots. Figures 5.5(b), (d), (f) and (h) show the number of noise samples (i.e. ones that were computed as $y^t = f_3(x_3^t)$) that were wrongly accepted by the internal model in blue, and the total number of wrong decisions for the internal model (both false negatives and false positives) in red, again showing the mean as solid line and the standard deviation as a colored belt. As stated above, samples that do not belong to the respective target function are randomly distributed in the input space, with some non-trivial distribution in the y-dimension. However, there is a certain probability that samples will fall onto, or lie close to, the target function. These samples are by chance similar to the target function, and thus cannot be distinguished from correct samples by the system. The expected numbers of samples for which this happens[1] are shown as dotted lines, on the one hand for samples originating from f_3 in blue, and on the other hand for samples originating from either f_2 or f_3 in red. Thus, for an optimally performing system, the solid blue line should coincide with the dotted blue line, meaning that the mean number of wrongly accepted noise samples equals the expected number of noise samples that are close to the target function, and the solid red line should coincide with the dotted red line, meaning that the mean total number of false positives and false negatives equals the expected total number of samples that do not correspond to the target function but happen to lie close to it.

Several things can be observed: First of all, the best performance is achieved with the choice of $\lambda = 7$, where after around 15 iterations the mean model performance (in terms of the mean squared error) has reached the reference performance level and training samples are optimally assigned. Choosing a too small value of λ results in a situation in which the models reduce the threshold for rejection too much, eventually producing a positive feedback loop of overfitting: As the mean squared error of the models gets lower during training, fewer training samples are accepted in the following iterations, causing an overfitting of the models due to the low amount of training samples, which in turn further reduces the amount of accepted training samples. Figure 5.6 shows an example, where the choice of $\lambda = 1$ has caused such a situation. This is mirrored in Figures 5.5(a)–(b), where on the one hand it can be seen that the internal model does not converge to a good performance, and on the other hand only few noise

[1]The expected number of samples that are close to the target value was numerically estimated by keeping track of the number of times that this happend across all training trials. During the generation of training samples, normally distributed noise with variance σ_ϵ^2 was added to the y^t. Thus, optimally performing internal models should produce a mean squared error of MSE $= \sigma_\epsilon^2$. Therefore, in each trial the number of times that $\|f_1(x_1^t) - f_2(x_2^t)\|^2 < \lambda \cdot \sigma_\epsilon^2$ and $\|f_1(x_1^t) - f_3(x_3^t)\|^2 < \lambda \cdot \sigma_\epsilon^2$ were counted, providing an estimate of the expected number of samples for which this is the case.



(a) $\lambda = 1$ (b) $\lambda = 1$

(c) $\lambda = 7$ (d) $\lambda = 7$

(e) $\lambda = 10$ (f) $\lambda = 10$

(g) $\lambda = 13$ (h) $\lambda = 13$

Figure 5.5: (a), (c), (e), (g) Mean model performance; (b), (d), (f), (h) mean number of wrongly accepted noise samples (blue) and mean number of false negatives and false positives (red). See text for description.

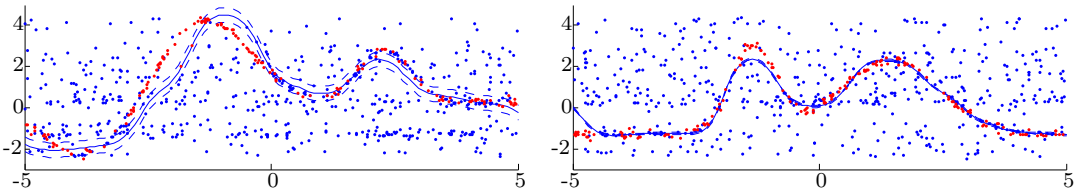


Figure 5.6: Example where the choice of a too low value for the parameter $\lambda = 1$ has led to an overfitting of the internal models. The internal model on the left is close to the target function in some regions of the input space, which allows it to maintain a relatively low mean squared error. However in other regions it has drifted away from the target function, thus producing many false negatives, which keeps the internal model from finding the target function. The internal model on the right hand has found a sub-optimal solution, rejecting too many samples, also causing an overfitting.

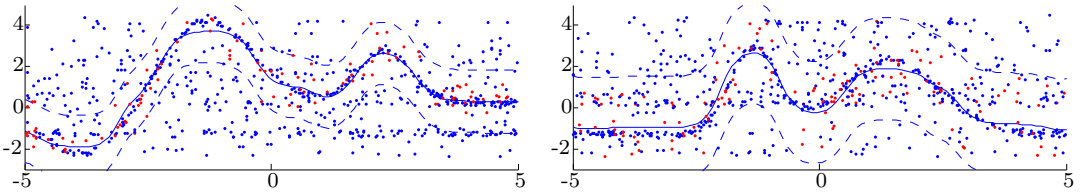


Figure 5.7: Example where the choice of a too large value for the parameter $\lambda = 13$ keeps the internal models from obtaining optimality. Many false positives retain large amounts of noise in the training sets for the internal model, causing them to demonstrate sub-optimal learning performance.

samples are accepted, but at the same time the number of false negatives is very high. In contrast, choosing a too large value of λ causes the model to continue to produce large amounts of false positives. As it can be seen in Figures 5.5(g)–(h) for the case of $\lambda = 13$, the performance of the model does improve over the number of iterations, but never reaches optimality, and the number of accepted noise samples remains large. Figure 5.7 shows an example result of a training for this choice of parametrization.

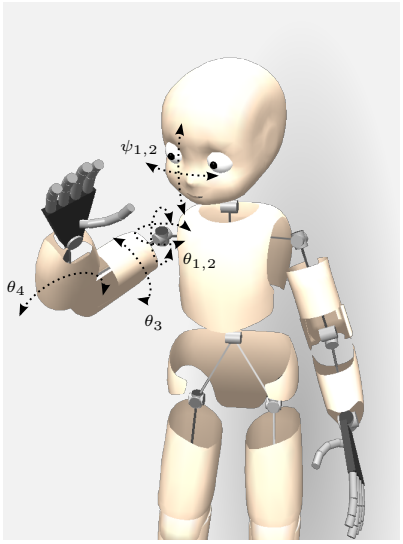
In conclusion it can be said, that with the right choice for the parameter λ (in this case $\lambda = 7$) the proposed method can successfully self-organize the learning process also in the presence of large amounts of noise. Overall it would be possible to improve the performance of the model, if the global threshold for rejection that each internal model uses was replaced by locally varying thresholds. This could be achieved by estimating the model performance not by a single value (here the mean squared error in conjunction with the model parameter λ), but estimating it for subregions of the input space separately. This would allow to prevent situations in which an internal model that in average produces good estimates for large amounts of the input space is not able to approach the target functions in other regions where it produces many false negatives instead.

5.3 Example Application of Acquiring a Body-Schema

In the last Section, a toy example was used to give a proof of concept for the proposed method. In the following, an application in robotics will be used to evaluate the method on a larger scale. Already at the beginning of the chapter, an example was described for a robot that should learn two internal models, one for its gaze control and one for its visuo-motor control. In the robotics literature, the acquisition and representation of this kind of knowledge is called a "body-schema" (Hoffmann et al., 2010), inspired by Psychology (cf. Section 3.1.3). Most existing approaches to learning a body schema as a mapping between motor commands and sensory feedback use a setup with a robot arm and a camera, fixed or movable, observing a scene in which the robot's own arm is also located. The body-schema is learned by fitting a model onto training data using standard machine learning techniques (e.g. Gaskett and Cheng, 2003; Metta et al., 1999; Rolf, 2012). These works make an implicit assumption about the training scenario: That the robot is already able to detect the position of its hand, for example by giving it an easily detectable color, which otherwise does not occur in the background scene (Gaskett and Cheng, 2003; Metta et al., 1999), or by fixing markers to the robot's hand (Rolf, 2012). This way it is of course ensured that only training data for the desired mapping is collected, and it is not considered how the model acquisition could be done in a more generic setup, where the robot should for example learn more than just a single internal model or there is no such intervention by the designer possible.

To loosen this limitation, several methods have been proposed to let the robot "self-detect" its end-effector in the camera input, other than using an easily detectable color or other kind of marker. Mostly these methods use some combination of movement detection and temporal contingency estimation between the sending of motor commands and the detection of movements. The methods can be prepended to the learning system to provide the necessary input, i.e. the position of the end-effector. For example, Stoytchev demonstrated how the characteristic delay between sending a motor command and observing change in the image can be estimated using temporal contingencies (Stoytchev, 2011). Fitzpatrick and Arsenio proposed to use a mechanism of correlating changes in arm postures with changes in pixel values, while performing rhythmic movements to locate the visual region of the end-effector in the camera image (Fitzpatrick and Arsenio, 2004). Kemp and Edsinger used mutual information to estimate the amount to which the robot's actions had control over image patches of different appearances and showed that their robot could learn the appearance of its end-effector and that it was controllable (Kemp and Edsinger, 2006). Gold and Scassellati trained a graphical model to use probabilistic reasoning over time to classify image regions as belonging to the robot or being animate or inanimate "other" (Gold and Scassellati, 2009).

In contrast to these approaches, which again introduce very task-specific demands to the overall learning method, this section will demonstrate that the proposed method for self-organized assignment of training samples to competing internal models can successfully learn internal models related to the body-schema of a robot, without the



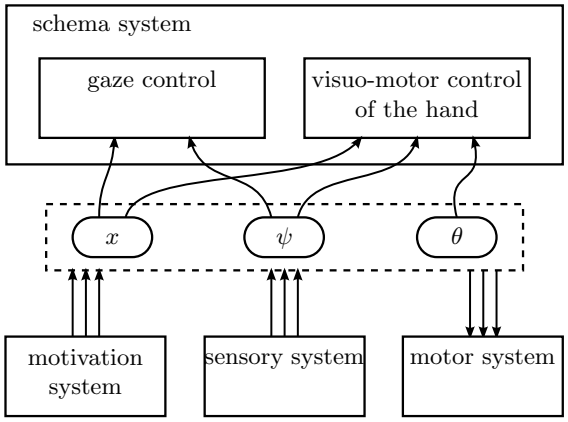


Figure 5.8: Overview of the system used in the simulation experiments.

need for prior knowledge about the appearance of the own hand, or the use of any other form of detection method. For this evaluation, a simulation of the humanoid robot iCub was used (Tikhanoff et al., 2008).

The system that was used for the evaluation consisted of two internal models, see Figure 5.8. The position of the head of the robot was controlled in the horizontal (turning) and the vertical (raising and lowering) direction, with angular positions $\psi \in \mathbb{R}^2$. The configuration of the right arm of the robot was controlled using the three shoulder joints and the elbow joint, with angular positions $\theta \in \mathbb{R}^4$.

Visual input to the system is provided as the location of one object in head-centric Cartesian coordinates, $x \in \mathbb{R}^3$. Thus, it is assumed that the vision problem is solved by the sensory system insofar as it is abstracted from raw camera input to the coordinate of an attended visual stimulus, which can either be in the background or on the hand of the robot, and that the stimulus is visually tracked while the robot moves its head and arm. However, the system does not know whether the attended stimulus corresponds to its own hand or not, which introduces the latent variable to the training process. To emulate the behavior of an attention mechanism, the system was set to attend either its own hand or randomly placed distractor objects with a given probability.

To learn its gaze control, the system should acquire the mapping

$$x(t_c) = f_{gaze}(\psi(t), \psi(t_c), x(t)), \tag{5.9}$$

describing how moving the head from posture $\psi(t)$ to posture $\psi(t_c)$ transforms the head-centric position $x(t)$ of a stimulus into a new position $x(t_c)$. Here, t denotes the point in time when new target postures for head and arm are issued to the controllers, and $t_c \geq t$ denotes the point in time when the controllers have converged.

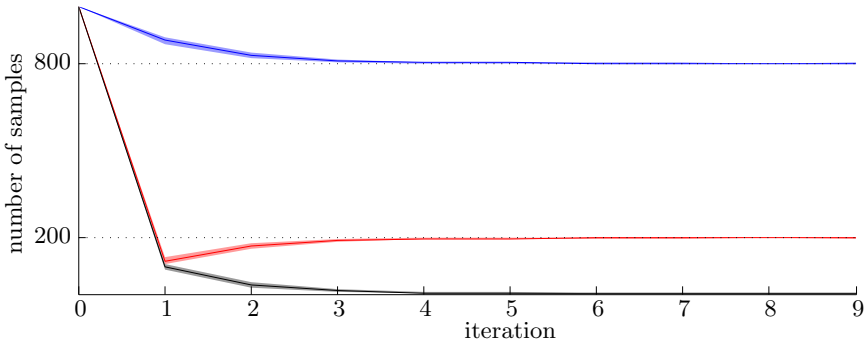


Figure 5.9: Number of training samples assigned to gaze model (red), visuo-motor model (blue), and total number of wrongly assigned samples (black). Mean values are shown as solid lines with standard deviations indicated by shaded regions.

Similarly, to learn its visuo-motor control, the system should acquire the mapping

$$x(t) = f_{vm}(\psi(t), \theta(t)), \tag{5.10}$$

describing the relationship between head and arm configurations and corresponding positions of the hand in head-centric coordinates.

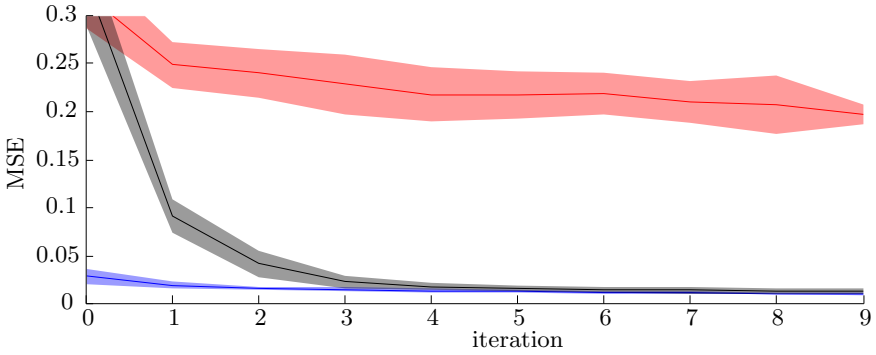
Training samples for the two internal models were generated by letting the robot simultaneously drive both its head and arm into random postures, and recording angular positions before and after the movement. Furthermore, before each new movement was initiated, a distractor object was placed at a random position in the workspace of the robot (the distractor object was always static, thus there were no moving background stimuli). The system was led to attend the attractor object with an 80% chance and its own hand with a 20% chance, and the position of the attended stimulus in head-centric coordinates was recorded before and after the movement. Thus, the training sets for the two internal models were

$$\begin{aligned} S_{gaze} &= \left\{ \left(\psi(t^i), \psi(t_c^i), x(t^i), x(t_c^i)\right) \mid i = 1, \ldots, N \right\} \text{ and} \\ S_{vm} &= \left\{ \left(\psi(t^i), \theta(t^i), x(t^i)\right) \mid i = 1, \ldots, N \right\} , \end{aligned} \tag{5.11}$$

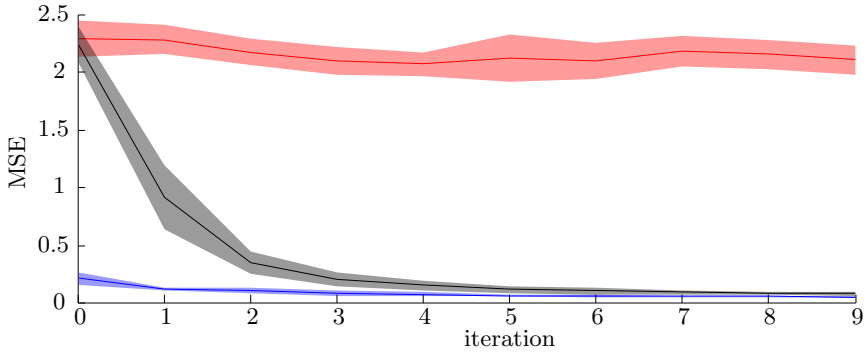
with a total of N training samples for each iteration of the learning method.

To test the proposed mechanism using standard machine learning methods, two multilayer perceptrons were used for learning, each with 30 neurons in a single hidden layer. The proposed learning method was iterated 9 times, and 1000 samples for training sets S_{gaze} and S_{vm} were generated for each iteration. After the assignment of training samples in each iteration, the weights of each multilayer perceptron was updated by training the networks for 50 epochs. The whole procedure was done in 15 separate trials to acquire some statistical information about the learning performance.

Figure 5.9 shows the number of training samples that were assigned to the two internal models, along with the total number of wrongly assigned training samples. For the initialization of the internal models, all 1000 training samples are assigned to



(a) Gaze control



(b) Visuo-motor control

Figure 5.10: Development of the mean squared error of internal models in the case of (a) gaze control, and (b) visuo-motor control. Solid black lines show the mean value of the mean squared error across all 15 training trials, with the standard deviation indicated by shaded regions around the lines. For comparison, learning performance when continuing to use all training samples for both networks is shown in red, and learning performance when using supervised learning is shown in blue.

both internal models. This is mirrored in Figure 5.9 as all training samples are being counted as wrongly assigned (training samples that correspond to situations where the robot looked at its own hand were also wrongly assigned to the internal model for gaze control, and samples corresponding to situations where the robot looked at the background were also wrongly assigned to the internal model for visuo-motor control). Over the next few iterations of the method, the process stably approaches a correct assignment of training samples with only very few miss-assignments (a mean of 2.8 wrongly assigned samples in the ninth iteration, with a standard deviation of 2.61, corresponding to a recognition rate of 99.72%).

Figure 5.10 shows the development of the performance of both internal models when using the proposed training method, on the one hand compared with using all training samples for the training of both internal models (as a lower bound for the system performance), and on the other hand compared with supervised learning, where the

system is told which training samples belong to either class (as an upper bound for the system performance). It can be seen that, after starting with an initially high error, both internal models quickly reach the performance level of the internal models trained with supervised learning. Also, the standard deviation of the model performances is quite low, which shows that the proposed method stably converges to a correct solution. The remaining residual errors, which are observed for both the internal models trained with the proposed method, as well as the internal models trained with supervised learning, are due to the choice of model parameters: A total number of 1000 training samples, with 800 for the gaze model and 200 for the visuo-motor control model, can be deemed rather low for the number of input dimensions for both internal models, and requires the multi-layer perceptrons to generalize between samples. Increasing both the number of training samples as well as the number of hidden units would allow the networks to perform better. However, this error is an artifact of the choice of learning method and parametrization, and is not linked to the proposed method for self-organizing the assignment of training samples.

5.4 Discussion

This chapter investigated the problem of generically integrating building blocks in a cognitive architecture from the learning side. Specifically, an issue was highlighted that arises as the consequence of having multiple internal models share the same sensory input channels: If there are different *kinds* of sensory feedback signals arriving at the same input, and one internal model should be trained for each kind of feedback, there has to be some way of dividing these inputs into separate streams of information for the individual internal models. As a concrete example, the problem of learning two internal models related to the body schema of a robot was used: One for gaze control, and one for visuo-motor control, both of which use as sensory feedback the Cartesian position of a stimulus, corresponding either to an object or to the robot's own hand, respectively. For the learning to be successful, each of these two kinds of inputs needs to be assigned to one or the other internal model. In the literature, this problem is either solved through designer intervention (using easily detectable cues and hard-coding a cue detection for the selection of inputs), or by implementing dedicated detection methods and prepending them to the learning process. However, both of these kinds of solutions make task-specific assumptions, whereas for the integration of building blocks in a cognitive architecture a method needs to be task- and domain-general.

As a solution, a method was proposed that puts involved internal models in competition with each other. Inputs are assigned to those internal models that are most adept to these inputs, in terms of how well they predict the inputs. It was shown, both in a toy example by way of illustration, as well as in a learning scenario using a simulation of the iCub robot, that this method is capable of self-organizing the learning process without any form of designer intervention or supervision, and to correctly assign inputs to corresponding internal models.

To allow for a better evaluation of the performance of the method, standard machine

learning methods were used for the learning of internal models. However, it should be noted that similarly to the method for integrating internal model responses presented in the last chapter, also this method does not commit to a specific learning method. Instead, it only requires the learning method to support some metric for measuring the similarity between predicted and actual inputs. In the simulation experiments that were presented, multilayer perceptrons were used as learning method, and the Cartesian distance in the space of input vectors as the metric for comparing prediction errors, but also using population codes for the representation of inputs (as required by the definition of the cognitive architecture proposed in Chapter 3) instead of vector-based representations is possible: Instead of using a Cartesian metric, the amount of overlap between activations in population codes can be compared, with a perfect overlap corresponding to an accurate prediction. This way, the method can also be implemented when using for example networks of sigma-pi units as a learning method.

The competition between internal models was described in this chapter as a binary weighting of training samples. To put it into a closer relation with the formulation of the cognitive architecture that is proposed in this thesis, the following extension to the method can be described. In Section 3.3.1 it was stated that simulators become arranged in self-organized topographies in dynamic neural fields, to form representations on higher levels of the hierarchy in the schema system. We could imagine the two internal models used in the simulation in this chapter to become part of such a self-organized representation of movement types of visual stimuli: On the one hand for static objects, and on the other hand for the movement of the own body. Additionally, further internal models, for example ones learning to predict moving objects, could also be integrated into this self-organized representation. The competition mechanism proposed in this chapter can then be described in the language of theories of embodied cognition and of the concept of schema, where it is argued that active elements of the conceptual system (simulators or schemata) absorb, and become activated by, perceptual events, so that the combination of activated elements represents the agent's interpretation of the situation (cf. Sections 3.1.2–3.1.3). For example, Barsalou argues that simulators produce embodied simulations, which are compared with observations, and simulators that produced the best matching simulations become activated (Barsalou, 1999). This process is congruous with the method used in this chapter, as here internal models produce estimations of observations (forward simulations for actions in the case of learning a body schema), and are selected if there is a good enough match. If we assume the internal models to be arranged in a dynamic neural field (as described in Section 3.3.1), the competition that is needed for the method described in this chapter can be understood as a result of internal models receiving activation during the processing of perceptual information: Each internal model accumulates activation as it produces estimations for the perceptual input, to a degree corresponding to how good the estimations match the observations. As the internal models are in a topographic relation with each other in the dynamic neural field, ones that are strongly activated during this process will inhibit others (cf. Section 4.2.2). By further assuming that activated internal models are adapted using the available information, i.e. they accom-

modate to the input (cf. Section 3.1.3; Piaget, 1997 [1953]), the method described in this chapter can be naturally explained as a function of the hypothesized active elements of the conceptual system. Furthermore, using dynamic neural fields for the competition between internal models should render the method robust against noise, as the Amari dynamics implements a low-pass filtering of the input signal (cf. Section 4.2.2; Amari, 1977). This can be especially beneficial if the internal models process sequential input, for example trajectories of different types of dynamic object movements: As a moving stimulus is tracked, the system might try to decide whether it is an animate movement (i.e. self-propelled), or if it is a caused movement (e.g. a rolling ball). Two internal models would learn to predict these different kinds of movements of stimuli, and their predictions would be matched with the observed trajectory. From one time step to the next, the decision of which internal model predicts the trajectory better would probably be rather unstable. However, accumulated over time and low-pass filtered in the dynamic neural field, the decision would become stable, allowing the winning internal model to gather the new information for learning.

6

Conclusion

6.1 Summary

It is not enough to develop many seperate computational models that each explain individual cognitive phenomena. After all, we would only end up with a pile of disconnected models, but without a clue on how to build a cognitive robot. With this intuition in mind, this thesis was concerned with the research question of how to build a cognitive architecture for a robot under the paradigm of embodied cognition. Motivated by a survey of the interdisciplinary literature in Chapters 2 and 3, and as mentioned in the introduction, the overarching goal of this thesis (**G0**, see Section 1.1) was to identify basic elements from which a cognitive architecture for a robot might be built. The approach that was taken was to consult prominent comprehensive theories of cognition from the literature and to compare the hypothesized basic mechanisms, from which cognition should emerge, according to these theories. As a result, important properties that can be used to guide the modeling of a cognitive archtecture have been identified (see Section 3.1.4). Most importantly, across theories it is argued for the existence of an active element, of which the cognitive system is composed, both on the structural level, as well as in terms of how knowledge is organized in the system. Therefore, with the working hypothesis that cognitive function can be implemented in a distributed way by basing a cognitive system on the use of such active elements, in Chapter 3 a new cognitive architecture was introduced with a schema system as its central component. The schema system houses a network of generic building blocks, that each perform only local computations based on their respective inputs, but in concert are responsible for the overall behavior of the system. To guide the modeling and in complying with research goal **G1** of the thesis to establish a theoretically sound basis for the modeling that is backed by empirical data, in Section 3.3.1 a view was proposed that combines several theories of embodied cognition and schemata. It was argued that a highly interconnected cascade of internal models and dynamic neural fields is at work, with the former being responsible for learning and the generation of candidate simulations, and the latter for decision making and integration. The view is congruous with Damasio's convergence-divergence model (Damasio, 1989) in terms of network structure and in-

formational flow, and is therefore also empirically backed by neuroscientific evidence. Furthermore, as it is based on the use of dynamic neural fields for decision making, it is also compatible with empirical findings from the dynamic field theory (Erlhagen and Schöner, 2002; Smith and Thelen, 2003; Thelen et al., 2001). Finally, by subscribing to a view of cognition as put forward by the concept of schema (Bartlett, 1932; Gallese and Lakoff, 2005; Piaget, 1997 [1953]) and by theories of embodied simulation (Barsalou, 1999; Gallese, 2003), a comprehensive view of how cognition emerges from the interplay of the active elements can be described. Thus, the proposed view is both concrete enough for computational modeling, as well as provides a coherent perception of how the modeling can be guided towards a more complete cognitive system.

Chapter 4 was concerned with several important questions related to the proposed cognitive architecture: How the building blocks support the acquisition of internal models from sensorimotor experience, how they can be used for accurate motor control in a closed-loop sensorimotor system, and most importantly how the outputs of multiple building blocks can be combined to produce a coherent system response. To solve this last question, a method was proposed that exploits the fact that redundant solutions exist in many sensorimotor problems, and lets the system select solutions that satisfy multiple tasks simultaneously. To be able to do so, the method requires the use of a learning technique that is able to retain information about redundancies and can restore this information upon a query, which most standard machine learning techniques are not capable of. As an example implementation of the method, networks of sigma-pi units were used as a learning technique, as an associative memory between population coding neural fields. To make the implementation applicable also in higher-dimensional domains despite the problems of scalability related to networks of sigma-pi units, a sparse implementation was used, and the network functionality was extended to support local error corrections in closed-loop control to compensate for inaccuracies in the information encoded in the sigma-pi weights. Furthermore, it was demonstrated that multiple low-dimensional dynamic neural fields can be used to substitute for a high-dimensional dynamic neural field, which is beneficial in terms of computational cost.

Chapter 5 introduced a method that solves the important problem of how to distinguish inputs that originate from different situational contexts, without any prior knowledge in the system. The example application of learning a body schema was used, where it was demonstrated that the system was able to separate inputs from the contexts "looking at own hand" and "looking at background scene", without relying on prior knowledge of for example the appearance of the own hand, or any other kind of pre-specified detection strategy. Instead, the learning can be bootstrapped by using preliminary model predictions, generated from the internal models themselves, while they are trained. For the development of a cognitive architecture based on generic building blocks, the method allows information to be directed to building blocks in a self-organized manner, without the need to introduce any supervisory mechanism on the architectural level. Both the methods for integrating outputs of building blocks and for separating input signals treat information independently from its content, and are therefore domain- and task-independent, contributing to research goal **G2** of the thesis

to develop a generic building block and its mechanisms, supporting the autonomous acquisition of knowledge from sensorimotor experience.

6.2 Discussion in Relation to Machine Learning and the Field of Cognitive Architecture

For the design of a cognitive architecture that should support the autonomous acquisition of knowledge from sensorimotor experience it obviously should be taken into account, under what conditions current machine learning methodology can operate successfully. For example, the camera input to a system could be handled by first selecting one salient image region and further applying filtering techniques to extract information from this region on color distribution, line orientations, principal components, and so on. This can be argued to be biologically plausible, since it is known that the brain processes input from the eyes in a similar way (cf. Section 2.1). On the other hand, such a representation is also necessary for a system to be operable with currently available machine learning methodology. Representations in vector formats are predominantly expected by current machine learning methods.

However, while it seems immediately clear that available methodology should constrain the design of a cognitive architecture, it is important to also consider the inverse. Learning methods are normally evaluated in the literature in terms of their accuracy, their generalization capability from as few training examples as possible, their suitability for online-learning, and so on, thus in how well they perform as a learning system by themselves. This is insufficient when aspiring to build cognitive robots: For this goal, the learning methodology that is employed needs to be capable of working well in a complex system, where *many* learning sub-systems are combined. Thus, aside from trying to develop learning methods that are powerful by themselves, research in machine learning should also consider the question of how the goal to build cognitive robots constrains the design of new learning methods. One example for such a constraint was described in Sections 4.1–4.2, where it was argued that a learning method should be capable to handle and restore information about redundancies for a flexible integration. Note that considering how to combine multiple learning systems can also be beneficial beyond the immediate goal to build an integrated system: For example, it is well known that a combination of several simple classifiers outperforms a single complex classifier in many classification tasks (see Polikar, 2009).

6.2.1 Comparison with Other Cognitive Architectures

It is not easily possible to evaluate one cognitive architecture against another, because they differ in too many ways and their respective capabilities are scattered across the vast spectrum of cognition. No current architecture demonstrates prowess in more than just a few aspects of cognition. Therefore, in Section 2.6 a hypothetical scale was introduced to approximately capture the current state of cognitive architectures in a way that allows to compare them, on the one hand in terms of the generality

of the underlying methodology, and on the other hand in terms of the tractable task complexity. Current cognitive architectures can be said to roughly follow a trend of negative correlation along the two dimensions. Progress in the field of cognitive architecture, as proposed earlier, can be defined as advancing in either one dimension, without regressing in the respective other.

The cognitive architecture that was proposed in this thesis shares most similarity with current connectionist and dynamicist architectures, thus it makes most sense to compare it with the existing cognitive architectures from these paradigms. Firstly, it is related to architectures based on the dynamic field theory, given the modeling choice of using dynamic neural fields for the integration and exchange of information between building blocks. These architectures however do not address the question of how learning is implemented, but instead focus on the integration and decision making capabilities of the system. In contrast, the cognitive architecture that was proposed in this thesis puts an explicit focus on how internal models in building blocks can be learned from sensorimotor experience and therefore allows the system to autonomously learn new sensorimotor transformations. Furthermore, while the architectures based on the dynamic field theory introduce decision making to select one among several possible actions (in most cases discrete action units) to solve a single task, the here proposed architecture extends this capability by allowing the system to solve multiple tasks simultaneously, as described in Chapter 4. Secondly, the proposed architecture is related to Morse et al.'s ERA (Morse et al., 2010) in several ways. Both the here proposed architecture and ERA argue for the use of generic building blocks in a cognitive system. For passing information between building blocks, both architectures make use of population coding neural fields. ERA uses self-organizing maps to cover the domain of input signals, which allows for a more efficient encoding as compared to the static neural fields that were used in the example implementations for simulations in Chapters 4 and 5. However, extending the here proposed architecture with the functionality of self-organizing maps would be straightforward. Furthermore, ERA uses Hebbian learning between self-organizing maps, which is essentially similar to using networks of sigma-pi units. In contrast, the architecture that was proposed in this thesis does not subscribe to one particular learning technique, but instead defines constraints on how information is integrated and passed in between building blocks. The use of dynamic neural fields at the interface between building blocks, and the way that information is combined allows the system to select favorable solutions to tasks. In contrast, if only Hebbian learning and a simple propagation of activation is used for the integration of building blocks, and an output is selected for example by selecting the neuron with strongest activation in each neural field, the system would be highly prone to undesired effects due to noise and optimal solutions would rarely be picked (cf. Figure 4.5). Moreover, the propagation of activation as it is used in the here proposed architecture limits activation across the system to one coherent sensorimotor situation, as in embodied simulation (see Section 3.3.3), whereas in ERA activation propagates unconstrained along learned connections, which can be expected to cause problems in more complex scenarios due to an over-activation of units across the whole system.

Additionally, the method presented in Chapter 5 for separating input data to train multiple internal models from a shared input is relevant for any learning system, not only in the context of cognitive architectures. It allows to train multiple internal models also when their training samples are not separated in advance, for example because they use different inputs. This or similar assumptions have been made in existing cognitive architectures (cf. Section 2.4.2), as well as in many other learning systems, such that in their limited evaluation scenarios the problem of relevance detection was circumvented by only presenting the learning method with valid samples. The method that was presented in this thesis allows the system to autonomously learn from experience also in more natural scenarios.

In sum, the proposed cognitive architecture together with the mechanisms presented in Chapters 4 and 5 extends the capabilities of existing cognitive architectures by addressing problems related to the integration of internal models and to learning. It solves problems that any cognitive architecture that is based on the use of building blocks faces (cf. Section 3.3.4). Furthermore, the proposed methods are not intrinsically tied to the use in a cognitive architecture, but can be employed in general for the integration of sensorimotor models on the one hand, and for the separation of input signals on the other.

6.3 Discussion in Relation to Embodied Cognition and the Concept of Schema

The presented cognitive architecture was motivated by theories of embodied cognition and by the concept of schema, as described in Chapter 3. As such, it should also be seen as a computational model of these concepts, even though the simulation experiments described in Chapters 4 and 5 were primarily intended to demonstrate the functionality of the system on a more technical level, and are not in a direct relation to psychophysical studies. However, in the light of the theories of embodied cognition and of the concept of schema that were described in Chapter 3, a few conclusions can be drawn related to the nature and function of the concepts. For this, it first needs to be made clear in what ways the computational models presented in this thesis can be related to the concepts of embodied cognition and schema.

A common hypothesis that is central to many theories related to embodied simulation and the concept of schema is that cognition is an emergent property in a system that is composed of basic active elements (see Sections 2.5.1, 3.1.1–3.1.4). This assumption is obviously mirrored in the proposed cognitive architecture as it is based on the use of generic building blocks, which each only perform local computations but in concert produce the overall system behavior. In the aforementioned theories, the active element is hypothesized to have several important functionalities. Most importantly, it allows to produce simulations of perceptual events in sensorimotor regions of the brain, a central mechanism through which many cognitive functions are realized (Barsalou, 1999; Gallese, 2003). Whereas existing computational models of embodied simulation are restricted to a context-dependent forward simulation of sensorimotor

states (see Section 3.2.2), the presented cognitive architecture represents a model of embodied simulation that is in a closer relationship with the way that the concept is used in the above mentioned theories. Many cognitive functions are hypothesized to be based on embodied simulations of *distal* sensorimotor situations. As one example, it is argued that language understanding in the brain is realized through an embodied simulation of the content of a perceived utterance (Glenberg and Gallese, 2011): Person A understands person B's utterance "the girl gives the horse an apple" by covertly reenacting a sensorimotor situation in which person A is watching a girl give an apple to a horse. It would be unreasonable to assume that this process is based on a forward simulation of a sensorimotor trajectory that originates from the current sensorimotor situation (such as imagining a girl and a horse walk into the office in which person A is currently sitting). Instead, related to how memory retrieval is thought to work based on schema knowledge (Bartlett, 1932), available cues (such as the words in the utterance) are used to reconstruct a situation that matches the information, augmented by schematic knowledge about other aspects that can be expected to match the information (for example being at a place where one could expect to find a horse, such as a horse stable). The process of reconstructing a distal sensorimotor situation can be thought to be achieved analogously to how information from multiple internal models is combined in the computational model presented in Chapter 4: Knowledge structures (internal models, simulators or schemata) that are related to the available information become activated and produce candidate simulations for sensorimotor regions, where the information becomes integrated in dynamic neural fields. For example, if it is assumed that the cognitive system has acquired an internal model of the appearance of what is referred to as "horse", it would generate unspecific activations in neural fields across modalities. Similarly as how a kinematics transformation can encompass redundant solutions (cf. Section 4.2.1), also the transformation between the linguistic label "horse" and a visual perception encompasses abundant amounts of redundancy: Horses can have diverse coat colors, can be tall or short, can have markings or not, can be viewed from the front and from the side, and so on. Thus, as in the mechanism for finding solutions to sensorimotor tasks described in Chapter 4, an embodied simulation of the visual perception of a horse could likewise be based on first activating large amounts of neurons in sensorimotor regions via top-down connections from associative cortical regions, corresponding to activating "sets of solutions", and then selecting a specific simulation through a competition in dynamic neural fields (cf. Section 3.3.1). Furthermore, if more information is available, for example that the horse is brown, the simulation is further restricted analogously to how multiple goals in sensorimotor tasks were used in this work to constrain the selection of the arm posture for a robot. Based on the resulting embodied simulation of a distal sensorimotor situation, the complete utterance of "the girl gives the horse an apple" could then be understood by using forward models to simulate a sensorimotor trajectory originating from the initially constructed simulation (cf. Glenberg and Gallese, 2011).

Thus, the cognitive architecture presented in this thesis subscribes to a view according to which information is combined in self-organized representations, which are again

subject to being combined with other representations, as part of a complex network in which more and more abstract representations are formed (first in modality-specific associative regions, and later in regions fusing cross-modal information). Importantly, this network of active elements of the cognitive system is not thought to be in the form of a strict tree-like hierarchy, in which each region would be only connected to a single higher-level region, but instead multiple regions can be connected to the same higher-level region, and there can also be lateral connections among connected lower-level regions. To some extent this is reflected in the simulation experiments described in Section 4.2.3, where multiple dynamic neural fields are co-ordinated to integrate information across representations, allowing to combine constraints that involve different groups of neural fields (in the simulation example corresponding on the one hand to kinematic constraints, which involve all degrees of freedom of the arm, and on the other hand constraints for individual joints). This hints at how a schema- or frame-like representation with slots for attributes and their values, which is hypothesized to be the basis for knowledge organization in the human conceptual system (cf. Sections 3.1.2–3.1.3; Barsalou, 1999; Gallese and Lakoff, 2005; Rumelhart, 1984), are to be found in the proposed architecture: Neural fields throughout the cognitive system represent domains of values, initially in modality-specific domains (such as color, line orientation, pitch, amplitude of force, etc.), and later on in more abstract domains (such as an "object form" topography, see Section 3.3.1; Haxby et al., 2001). Internal models learn about the combinatorial arrangement of input values that occurred synchronously (see Sections 3.3.1, 4.3; Damasio, 1989). Given some cues or target values in one or several domains, internal models can restore this information about what values in other domains correspond to the given cues, in the form of activation landscapes in the corresponding population coding neural fields (see Sections 4.2.2, 4.3.1). Together, this structure implements the combination of slots for attributes, in the form of connected neural fields, and values with their constraints, via internal models mapping values in between neural fields. With the mechanism for co-ordinating the activation of interconnected dynamic neural fields presented in Section 4.2.3, this suggests to understand the retrieval of information based on schematic knowledge in a similar way as proposed by Rumelhart et al. in their global constraint satisfaction network model (see Section 3.2.1; Rumelhart et al., 1986): Mutual constraints between values in input domains let the system settle to a state in which as many constraints as possible are met. While Rumelhart et al.'s model is based on a localist representation, variables can here assume continuous values as activation peaks in the neural fields. As shown in simulation in Section 4.2.3, each dynamic neural field separately selects a value based on the available input constraints. As one field settles on a decision, the input to other connected dynamic neural fields is amended and thus their respective decisions are influenced. Over time, the system settles to a state in which the decisions of the dynamic neural fields are co-ordinated, thus "filling" the "slots" with a matching set of values.

Finally, in respect to parallels between the proposed cognitive architecture and the concept of schema, it should also be noted that the mechanism for separating inputs described in Chapter 5 can also be described using Piaget's terminology of

"assimilation" and "accommodation" of schemata (Piaget, 1997 [1953]): Piaget argues that schemata are used to reduce incoming information to the known (see Section 3.1.3). The cognitive system tries to match, i.e. assimilate, the current situation with its schema knowledge. Whenever a schema assimilates stimuli, it takes in this new information and accommodates to it, for the cognitive system to stay adept to its environment. These assumptions are also paralleled in Barsalou's theory of perceptual symbol systems, where he argues that simulators are used for categorization by producing embodied simulations, which are compared with perceived stimuli (Barsalou, 1999). An analogous process is used in the computational model presented in this thesis (see Section 5.1): As internal models produce estimates for inputs in a form of embodied simulation, they stand in competition against each other and are selected if they can account sufficiently well for the available data. This process can be described as the internal models "assimilating" input data, or inputs being categorized as Barsalou describes. Each new data point is then used for training the respective winning internal model, thus it is "accommodated" to the new information.

6.4 Outlook

With the methods presented in Chapters 4–5, basic functionality of the generic building block of the architecture has been demonstrated, as it was described in Sections 3.3.1–3.3.3. However, not all properties could be demonstrated by an implementation in simulation experiments, which is out of the scope of this thesis. One important property that was not covered by the presented implementations is that of self-organized forming of topographical representations on higher hierarchical levels in the schema system. Extending the functionality of the building blocks with this property would allow to use representations of reduced dimensionality and to form a hierarchical network structure. As one possible way to implement this property, an approach from the literature could be adapted: Weber and Wermter have demonstrated that networks of sigma-pi units can be trained with a SOM-like learning rule, which allows the network to discover topographies by unsupervised learning (Weber and Wermter, 2007). Their "Sigma-Pi SOM" is similar to the network of sigma-pi units described in this work, but uses an additional neural field as an output layer. After training, the topographical arrangement of neurons in this output field should reflect a topography inherent to the input data. As an example, Weber and Wermter use the learning of a head-centric coordinate representation: Since the same visual stimulus is located at different positions in the image depending on the current angular configuration of the neck, the network needs to discover what pairs of input values (of angular positions and positions in the camera image) correspond to the same head-centric location. The system can discover this information by visually tracking an object and recording its position in the image before and after a head movement. The input before the movement is used to select a winning neuron in the output layer, which is then trained along with its local neighborhood using the input after the movement. After iterating this process many times, the topography of the output neurons corresponds to a head-centric coordinate

representation.

It should however be noted that one can expect to encounter known problems of the SOM learning rule, such as folding. Also, a Sigma-Pi SOM has positive weights for most synaptic connections early on during training, before the majority of connections dies out as the network settles, which however limits the network to low-dimensional problems due to the number of synaptic connections increasing exponentially with the number of input dimensions in networks of sigma-pi units. Similar as already discussed in Section 4.5, this limitation could be lifted if a learning method was used that represents the input space more efficiently. Whereas sigma-pi units have fixed receptive fields, other learning methods allow to adapt the receptive fields of units as needed, for example to let one unit represent larger regions of the input space where the input data distribution is more or less uniform (e.g. D'Souza et al., 2001; Lopes and Damas, 2007; Rasmussen, 2000).

Networks of sigma-pi units, including (Weber and Wermter, 2007)'s model, have the nice property that they directly support the retrieval of redundant solutions. For example, in the case of the head-centric coordinate representation that is learned by (Weber and Wermter, 2007)'s network, clamping one neuron in the output field to a high activity in turn produces activations in the input fields that correspond to all pairs of inputs (head configurations and positions in the input image) that correspond to the head-centric coordinate encoded by the clamped output neuron. This kind of behavior is exactly that described in Section 3.3.1 for the hierarchical production of an embodied simulation: Neural fields on higher hierarchical levels topographically organize "simulators", which upon activation produce candidate simulations for lower-level fields. In the case of the head-centric coordinate representation, neurons on the higher-level neural field correspond to simulators for different locations. Analogously, one could imagine the output field to encode for example an object form topology. Activating a region in this field would result in candidate simulations in lower-level representations corresponding to the encoded object form.

The implementation of the building block described in Chapter 4 is able to retrieve solutions to tasks as embodied simulation of sensorimotor situations in which the goal is achieved. This is sufficient in cases where a single action can produce the simulated situation. For example, in the case of kinematic tasks, a suitable body posture is retrieved and can be directly effectuated by the motors. However, for the more general case it needs to be considered that several actions can be necessary to achieve a goal. For example, a mobile robot might need to plan several movements for reaching a goal from its current location. Preliminary simulation results suggest that this can be achieved by using a method of propagating activation along the sigma-pi connections, to produce a gradient of activation that points towards the selected solution. It is then sufficient to produce outputs along the gradient, originating from the current sensorimotor situation. A similar method has been used by Butz et al., but instead of using a propagation along the synaptic connections, a gradient of activation is generated in the input space, in the case of Butz et al.'s model corresponding to arm configurations (Butz et al., 2007a). This has the consequence that resulting trajectories are always linear in the space of arm

configurations, whereas when using the space of synaptic connections for generating the gradient of activation, arbitrary trajectories could be generated, for example to achieve a linear movement with the hand in Cartesian space. Toussaint has also proposed to use a gradient of activation to plan trajectories (Toussaint, 2006). Toussaint's model first builds up a topographic representation of the input space using a type of growing neural gas algorithm, and then uses the learned topographical representation for the generation of a gradient of activation for planning trajectories. This has the additional advantageous property that the learned representation covers the input space more efficiently than the fixed layout of networks of sigma-pi units does.

As already discussed in Section 4.5, a representation of the input space using fewer units is desirable, but achieving it always comes with the cost of increased learning effort. The network of sigma-pi units that was described in this thesis is capable of one-shot learning, and thus can rapidly be trained from input samples. In contrast, learning a consolidated representation requires presenting inputs many times, to iteratively adapt the representation to the statistics of input values. A possibility to benefit both from one-shot learning as well as from a more efficient representation would be to combine both learning methods: As for example suggested by Gläser, rapid one-shot learning could first be used to store samples during online operation in short-term memory, and subsequently a more efficient representation could be trained by using the short-term memory model to produce many training samples for the slow learning of a consolidated representation (Gläser, 2012, pp. 63–64).

A complete implementation of the proposed cognitive architecture, including the extensions to the implementation of the building block that were just outlined, could be used in more complex scenarios, and would allow to study interesting properties of the cognitive system, as well as to test hypotheses from other disciplines about the function of embodied simulation. A first scenario that would be easily tractable is the example described in Section 3.3.3 of an agent that can reason about object properties by means of embodied simulation. And since the representational format in the cognitive architecture is compatible with theories of embodied simulation and the concept of schema, it also provides the necessary means to study social interaction and the understanding of the actions of others through embodied simulation, as outlined in Section 3.1.4. On the long run, it will be inevitable to study complete cognitive systems such as the cognitive architecture described in this thesis, instead of only studying small learning systems separately, to answer the question of how to build truly autonomous cognitive robots.

References

J.S. Albus, H.G. McCain, and R. Lumia. NASA/NBS standard reference model for telerobot control system architecture (NASREM). Technical report, National Institute of Standards and Technology, Gaithersburg, MD, 1989. 18

Shun-ichi Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 27(2):77–87, 1977. 30, 31, 103, 107, 153

Shun-Ichi Amari. Topographic organization of nerve fields. *Bulletin of Mathematical Biology*, 42(3):339–364, 1980. 36

J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. *Psychological Review*, 111(4):1036–1060, 2004. 15, 16

J.R. Anderson and C.J. Lebiere. *The atomic components of thought*. Lawrence Erlbaum, 1998. 15

Michael A. Arbib. Schema theory. In *The handbook of brain theory and neural networks*, pages 993–998. MIT Press, 1998. 67, 68, 69, 73, 75

Ronald Arkin. *Behavior-based robotics*. MIT Press, Cambridge Mass., 1998. 68

Ronald C. Arkin. Motor schema-based mobile robot navigation. *The International Journal of Robotics Research*, 8(4):92–112, August 1989. 22, 96

Minoru Asada, Karl F. MacDorman, Hiroshi Ishiguro, and Yasuo Kuniyoshi. Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous Systems*, 37(2–3):185–193, November 2001. 3, 7

Bernard J. Baars. The conscious access hypothesis: origins and recent evidence. *Trends in Cognitive Sciences*, 6(1):47–52, January 2002. 38

Louise Barrett, Peter Henzi, and Drew Rendall. Social brains, simple minds: does social complexity really require cognitive complexity? *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1480):561–575, April 2007. 54

L. W. Barsalou. Perceptual symbol systems. *Behavioral and Brain Sciences*, 22(04): 577–660, 1999. 21, 46, 48, 52, 53, 55, 57, 59, 60, 61, 62, 66, 73, 75, 76, 78, 88, 152, 156, 159, 161, 162

REFERENCES

Lawrence W. Barsalou. Grounded cognition. *Annual Review of Psychology*, 59(1): 617–645, January 2008. 21, 46, 61, 75

Lawrence W Barsalou, Cynthia Breazeal, and Linda B Smith. Cognition as coordinated non-cognition. *Cognitive Processing*, 8(2):79–91, June 2007. 29

F. C. Bartlett. *Remembering*. Oxford University Press, Oxford, England, 1932. 55, 56, 57, 58, 60, 62, 63, 73, 75, 156, 160

Randall D. Beer. A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, 72(1–2):173–215, January 1995. 29

Randall D. Beer. Dynamical approaches to cognitive science. *Trends in Cognitive Sciences*, 4(3):91–99, March 2000. 21, 29, 46

Randall D. Beer. The dynamics of active categorical perception in an evolved model agent. *Adaptive Behavior*, 11(4):209 –243, December 2003. 29, 48

Estela Bicho, Luís Louro, and Wolfram Erlhagen. Integrating verbal and nonverbal communication in a dynamic neural field architecture for Human–Robot interaction. *Frontiers in Neurorobotics*, 4(5), 2010. 36, 37

Jeffrey J. Biesiadecki, P. Chris Leger, and Mark W. Maimone. Tradeoffs between directed and autonomous driving on the mars exploration rovers. *The International Journal of Robotics Research*, 26(1):91–104, January 2007. 7

C.M. Bishop. *Pattern recognition and machine learning*, volume 4. Springer, New York, 2006. 25

R. Peter Bonasso, R. James Firby, Erann Gat, David Kortenkamp, David P. Miller, and Mark G. Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2-3):237–256, 1997. 18, 19

Botvinick and David C Plaut. Doing without schema hierarchies: A recurrent connectionist approach to normal and impaired routine sequential action. *Psychological review*, 111:395—429, 2004. 25, 43

W. F. Brewer and G. V. Nakamura. The nature and functions of schemas. *Handbook of social cognition*, 1:119–160, 1984. 66

Korbinian Brodmann. *Vergleichende Lokalisationslehre der Grosshirnrinde in ihren Prinzipien dargestellt auf Grund des Zellenbaues*. Johann Ambrosius Barth Verlag, Leipzig, Germany, 1909. 10

R. Brooks. A robust layered control system for a mobile robot. *IEEE journal of robotics and automation*, 2(1):14–23, 1986. 21, 22, 92, 95, 96

R. A. Brooks. How to build complete creatures rather than isolated cognitive simulators. In *Architectures for intelligence: the twenty-second Carnegie Mellon Symposium on Cognition*, page 225–239, 1991a. 21

Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1–3): 139–159, January 1991b. 20

Martin V. Butz, Olivier Sigaud, and Pierre Gérard. Anticipatory behavior: Exploiting knowledge about the future to improve current behavior. In Martin V. Butz, Olivier Sigaud, and Pierre Gérard, editors, *Anticipatory Behavior in Adaptive Learning Systems*, number 2684 in Lecture Notes in Computer Science, pages 1–10. Springer Berlin Heidelberg, January 2003. 71

Martin V. Butz, Oliver Herbort, and Joachim Hoffmann. Exploiting redundancy for flexible behavior: Unsupervised learning in a modular sensorimotor control architecture. *Psychological Review*, 114(4):1015–1046, 2007a. 114, 115, 133, 163

Martin V. Butz, Olivier Sigaud, Giovanni Pezzulo, and Gianluca Baldassarre. Anticipations, brains, individual and social behavior: An introduction to anticipatory systems. In Martin V. Butz, Olivier Sigaud, Giovanni Pezzulo, and Gianluca Baldassarre, editors, *Anticipatory Behavior in Adaptive Learning Systems*, number 4520 in Lecture Notes in Computer Science, pages 1–18. Springer Berlin Heidelberg, January 2007b. 71, 73

Linda L. Chao and Alex Martin. Representation of manipulable man-made objects in the dorsal stream. *NeuroImage*, 12(4):478–484, October 2000. 48

Wayne Christensen. Self-directedness: A process approach to cognition. *Axiomathes*, 14(1):157–175, 2004. 19, 20, 41

A. Clark. Local associations and global reason: Fodor's frame problem and second-order search. *Cognitive Science Quarterly*, 2(1), 2002. 20, 41

Andy Clark. The dynamical challenge. *Cognitive Science*, 21(4):461–481, October 1997. 29, 46

Andy Clark. *Mindware : an introduction to the philosophy of cognitive science*. Oxford University Press, New York, 2001. 12, 21, 25

Jefferson A. Coelho and Roderic A. Grupen. A control basis for learning multifingered grasps. *Journal of Robotic Systems*, 14(7):545–557, 1997. 70

Stephen Coombes. Neural fields. *Scholarpedia*, 1(6):1373, 2006. 30

R. Cooper and T. Shallice. Contention scheduling and the control of routine activities. *Cognitive Neuropsychology*, 17(4):297–338, 2000. 25, 66, 67

R. P Cooper and T. Shallice. Hierarchical schemas and goals in the control of sequential behavior. *Psychological review*, 113(4):887–916, 2006. 66

REFERENCES

Antonio Damasio and Kaspar Meyer. Behind the looking-glass. *Nature*, 454(7201): 167–168, July 2008. 61

Antonio R. Damasio. Time-locked multiregional retroactivation: A systems-level proposal for the neural substrates of recall and recognition. *Cognition*, 33(1-2):25–62, November 1989. 48, 49, 50, 52, 57, 61, 62, 73, 75, 80, 155, 161

J. Decety, D. Perani, M. Jeannerod, V. Bettinardi, B. Tadary, R. Woods, J. C. Mazziotta, and F. Fazio. Mapping motor representations with positron emission tomography. *Nature*, 371(6498):600–602, 1994. 48

Jean Decety, Marc Jeannerod, and Claude Prablanc. The timing of mentally represented actions. *Behavioural Brain Research*, 34(1–2):35–42, August 1989. 48

S. Deneve, P. E. Latham, and A. Pouget. Efficient computation and cue integration with noisy population codes. *Nature Neuroscience*, 4(8):826–831, August 2001. 80, 104

Gary Drescher. *Made-up minds : a constructivist approach to artificial intelligence*. MIT Press, Cambridge Mass., 1991. 64

Hubert L. Dreyfus. *What computers can't do: a critique of artificial reason*. Harper & Row, 1972. 20

Hubert L. Dreyfus. Why heideggerian AI failed and how fixing it would require making it more heideggerian. *Philosophical Psychology*, 20(2):247–268, 2007. 20, 21

N. F. Dronkers, O. Plaisant, M. T. Iba-Zizen, and E. A. Cabanis. Paul broca's historic cases: high resolution MR imaging of the brains of leborgne and lelong. *Brain*, 130 (5):1432–1441, May 2007. 12

A. D'Souza, S. Vijayakumar, and S. Schaal. Learning inverse kinematics. In *2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2001. Proceedings*, volume 1, pages 298–303 vol.1. IEEE, 2001. 25, 53, 93, 112, 134, 163

Gerald M. Edelman. *Neural Darwinism: the theory of neuronal group selection*. Basic Books, New York, 1987. 115

Chris Eliasmith. The third contender: A critical examination of the dynamicist theory of cognition. *Philosophical Psychology*, 9(4):441–463, 1996. 29

Jeffrey L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7(2):195–225, 1991. 25

W. Erlhagen, A. Mukovskiy, E. Bicho, G. Panin, C. Kiss, A. Knoll, H. van Schie, and H. Bekkering. Goal-directed imitation for robots: A bio-inspired approach to action understanding and skill learning. *Robotics and Autonomous Systems*, 54(5):353–360, May 2006. 36, 37, 41

Wolfram Erlhagen and Estela Bicho. Dynamic field theory (DFT): applications in cognitive science and robotics. White paper for the euCognition network, University of Minho, Portugal, 2009. 29

Wolfram Erlhagen and Gregor Schöner. Dynamic field theory of movement preparation. *Psychological Review*, 109(3):545–572, 2002. 30, 31, 33, 34, 61, 75, 103, 156

Paul Fitzpatrick and Artur Arsenio. Feel the beat: using cross-modal rhythm to integrate perception of objects, others, and self. In *Proceedings of the Fourth International Workshop on Epigenetic Robotics*, Genoa, Italy, 2004. Lund University Cognitive Studies. 147

Paul Fitzpatrick, Giorgio Metta, and Lorenzo Natale. Towards long-lived robot genes. *Robotics and Autonomous Systems*, 56(1):29–45, January 2008. 18

Jason G. Fleischer and Jeffrey L. Krichmar. Sensory integration and remapping in a model of the medial temporal lobe during maze navigation by a brain-based device. *Journal of Integrative Neuroscience*, 06(03):403–431, September 2007. 25

Jerry A. Fodor. *Modularity of Mind*. MIT Press, Cambridge, Mass., 1983. 11, 15

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, April 1999. 26

Karl Friston, James Kilner, and Lee Harrison. A free energy principle for the brain. *Journal of Physiology-Paris*, 100(1–3):70–87, July 2006. 141

Jannik Fritsch and Sebastian Wrede. An integration framework for developing interactive robots. In Davide Brugali, editor, *Software Engineering for Experimental Robotics*, volume 30 of *Springer Tracts in Advanced Robotics*, pages 291–305. Springer Berlin / Heidelberg, 2007. 18

Joaquín M. Fuster. Cortex and memory: Emergence of a new paradigm. *Journal of Cognitive Neuroscience*, 21(11):2047–2072, November 2009. 75

Vittorio Gallese. The manifold nature of interpersonal relations: the quest for a common mechanism. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):517–528, March 2003. 55, 61, 89, 156, 159

Vittorio Gallese. Embodied simulation: From neurons to phenomenal experience. *Phenomenology and the Cognitive Sciences*, 4(1):23–48, March 2005. 46

Vittorio Gallese. Mirror neurons and the social nature of language: The neural exploitation hypothesis. *Social Neuroscience*, 3:317–333, September 2008. 61, 77

Vittorio Gallese and George Lakoff. The brain's concepts: the role of the sensory-motor system in conceptual knowledge. *Cognitive Neuropsychology*, 22(3):455, 2005. 48, 55, 61, 66, 75, 88, 156, 161

Vittorio Gallese and Thomas Metzinger. Motor ontology: the representational reality of goals, actions and selves. *Philosophical Psychology*, 16(3):365–388, 2003. 82

Vittorio Gallese, Luciano Fadiga, Leonardo Fogassi, and Giacomo Rizzolatti. Action recognition in the premotor cortex. *Brain*, 119(2):593 –609, April 1996. 55

Bjorn Peter Gardenfors. *Conceptual Spaces: The Geometry of Thought*. MIT Press, 2004. 27

C. Gaskett and G. Cheng. Online learning of a motor map for humanoid robot reaching. In *Proc. 2nd Int. Conf. Computational Intelligence, Robotics and Autonomous Systems*, 2003. 147

A. Gepperth, J. Fritsch, and C. Goerick. Cross-module learning as a first step towards a cognitive system concept. In *Proceedings of the International Conference on Cognitive Systems*, Karlsruhe, Germany, 2008. 80

JJ Gibson. The concept of affordances. *Perceiving, acting, and knowing*, page 67–82, 1977. 20

M. Gienger, H. Janssen, and C. Goerick. Task-oriented whole body motion for humanoid robots. In *2005 5th IEEE-RAS International Conference on Humanoid Robots*, pages 238–244. IEEE, December 2005. 95, 98

Michael Gienger, Marc Toussaint, and Christian Goerick. Whole-body motion planning – building blocks for intelligent systems. In Kensuke Harada, Eiichi Yoshida, and Kazuhito Yokoi, editors, *Motion Planning for Humanoid Robots*. Springer, 2010. 95

Arthur M. Glenberg and Vittorio Gallese. Action-based language: A theory of language acquisition, comprehension, and production. *Cortex*, 2011. 61, 160

C. Gläser, F. Joublin, and C. Goerick. Learning and use of sensorimotor schemata maps. In *IEEE 8th International Conference on Development and Learning*, pages 1–8, 2009. 36, 69

Claudius Gläser. *Making sense of words through the eyes of a child : a computational framework for the acquisition of world meanings*. PhD thesis, Bielefeld University, 2012. 36, 40, 133, 134, 164

Claudius Gläser, Frank Joublin, and Christian Goerick. Enhancing topology preservation during neural field development via wiring length minimization. In Véra Kůrková, Roman Neruda, and Jan Koutník, editors, *Artificial Neural Networks*, volume 5163, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. 104, 133

Kevin Gold and Brian Scassellati. Using probabilistic reasoning over time to self-recognize. *Robotics and Autonomous Systems*, 57(4):384–392, April 2009. 147

P S Goldman-Rakic. Topography of cognition: Parallel distributed networks in primate association cortex. *Annual Review of Neuroscience*, 11(1):137–156, 1988. 11

Melvyn A. Goodale and A.David Milner. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15(1):20–25, January 1992. 10

Henry Gray. *Anatomy of the human body*. Lea & Febiger, Philadelphia, 1918. 9

Michael S.A Graziano, Charlotte S.R Taylor, and Tirin Moore. Complex movements evoked by microstimulation of precentral cortex. *Neuron*, 34(5):841–851, May 2002. 77

Thomas L. Griffiths, Nick Chater, Charles Kemp, Amy Perfors, and Joshua B. Tenenbaum. Probabilistic models of cognition: exploring representations and inductive biases. *Trends in Cognitive Sciences*, 14(8):357–364, August 2010. 13

Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42 (1–3):335–346, June 1990. 12

S. Hart and R. Grupen. Learning generalizable control programs. *Autonomous Mental Development, IEEE Transactions on*, PP(99):1, 2011. 70, 71

S. Hart, S. Sen, and R. Grupen. Intrinsically motivated hierarchical manipulation. In *IEEE International Conference on Robotics and Automation 2008*, pages 3814–3819, 2008a. 70

S. Hart, S. Sen, and R. Grupen. Generalization and transfer in robot control. In *Proceedings of the 8th International Conference on Epigenetic Robotics*, 2008b. 70, 71

Masahiko Haruno, Daniel M. Wolpert, and Mitsuo Kawato. MOSAIC model for sensorimotor learning and control. *Neural Computation*, 13(10):2201–2220, October 2001. 72

J. V. Haxby, M. I. Gobbini, M. L. Furey, A. Ishai, J. L. Schouten, and P. Pietrini. Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, 293(5539):2425–2430, September 2001. 78, 80, 161

Henry Head and Gordon Holmes. Sensory disturbances from cerebral lesions. *Brain*, 34(2-3):102–254, 1911. 55, 56, 58, 60

Nikolas J. Hemion, Frank Joublin, and Katharina J. Rohlfing. A competitive mechanism for self-organized learning of sensorimotor mappings. In *Proceedings of the IEEE International Conference on Development and Learning (ICDL)*, Frankfurt am Main, August 2011. IEEE. 136

Nikolas J. Hemion, Frank Joublin, and Katharina J. Rohlfing. Integration of sensorimotor mappings by making use of redundancies. In *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI)*, Brisbane, Australia, June 2012. IEEE. 92

Oliver Herbort and Martin V. Butz. Too good to be true? ideomotor theory from a computational perspective. *Frontiers in Cognition*, 3:494, 2012. 82

Oliver Herbort, Martin Butz, and Gerulf Pedersen. The SURE_REACH model for motor learning and control of a redundant arm: From modeling human behavior to applications in robotics. In Olivier Sigaud and Jan Peters, editors, *From Motor Learning to Interaction Learning in Robots*, volume 264 of *Studies in Computational Intelligence*, pages 85–106. Springer Berlin / Heidelberg, 2010. 25, 114, 133

Germund Hesslow. Conscious thought as simulation of behaviour and perception. *Trends in Cognitive Sciences*, 6(6):242–247, June 2002. 71

Gregory Hickok and David Poeppel. The cortical organization of speech processing. *Nature Reviews Neuroscience*, 8(5):393–402, May 2007. 11

Geoffrey E. Hinton and Terrence J. Sejnowski. Learning and relearning in boltzmann machines. In David E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, pages 282–317. MIT Press, Cambridge, MA, USA, 1986. 64

H. Hoffmann and R. Möller. Action selection and mental transformation based on a chain of forward models. In *From animals to animats 8: Proceedings of the International Conference on Simulation of Adaptive Behavior*, page 213, 2004. 71, 72

M. Hoffmann, H. Marques, A. Arieta, H. Sumioka, M. Lungarella, and R. Pfeifer. Body schema in robotics: A review. *Autonomous Mental Development, IEEE Transactions on*, 2(4):304–324, 2010. 147

Andrew Howes and Richard M. Young. The role of cognitive architecture in modeling the user: Soar's learning mechanism. *Hum.-Comput. Interact.*, 12(4):311–343, December 1997. 7

D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154.2, January 1962. 80

Manfred Huber. *A hybrid architecture for adaptive robot control*. PhD thesis, University of Massachusetts Amherst, 2000. 70

Manfred Huber and Roderic A. Grupen. Learning to coordinate controllers - reinforcement learning on a control basis. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997. 70

Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, page 1523–1530. MIT Press, Cambridge, MA, 2003. 98

Jeffrey S. Johnson, John P. Spencer, and Gregor Schöner. Moving to higher ground: The dynamic field theory and the dynamics of visual cognition. *New Ideas in Psychology*, 26(2):227–251, August 2008. 30, 36, 61, 75

Mark Johnson. *The Body in the Mind*. The University of Chicago Press, Chicago, 1987. 60

Michael I Jordan and David E Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16(3):307–354, July 1992. 25, 53, 93, 98, 112, 113

Immanuel Kant. *Critik der reinen Vernunft*. Hartknoch, Riga, 1781. 55

Mitsuo Kawato. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9(6):718–727, December 1999. 54

Charles C. Kemp and Aaron Edsinger. What can i control? a framework for robot self-discovery. In *Proceedings of the Sixth International Conference on Epigenetic Robotics*, Paris, France, 2006. 147

Stefan J. Kiebel, Jean Daunizeau, and Karl J. Friston. A hierarchy of time-scales and the brain. *PLoS Comput Biol*, 4(11), November 2008. 141

D. Klahr. Computational models of cognitive change: the state of the art. *Developing cognitive competence: New approaches to process modeling*, pages 355–373, 1995. 59

E. Kohler, C. Keysers, M.A. Umiltà, L. Fogassi, V. Gallese, and G. Rizzolatti. Hearing sounds, understanding actions: Action representation in mirror neurons. *Science*, 297(5582):846–848, August 2002. 55

Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982. 27

Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1398 –1404 vol.2, April 1991. 23

P. Kormushev, S. Calinon, R. Saegusa, and G. Metta. Learning the skill of archery by a humanoid robot iCub. In *10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 417–423, 2010. 98

David Kortenkamp and Reid Simmons. Robotic systems architectures and programming. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 187–206. Springer Berlin Heidelberg, 2008. 18, 23

Jeffrey L. Krichmar and Gerald M. Edelman. Machine psychology: Autonomous behavior, perceptual categorization and conditioning in a brain-based device. *Cerebral Cortex*, 12(8):818–830, August 2002. 25, 26

REFERENCES

Jeffrey L. Krichmar and Gerald M. Edelman. Brain-based devices for the study of nervous systems and the development of intelligent machines. *Artificial Life*, 11 (1-2):63–77, January 2005. 25

John E. Laird. Extending the soar cognitive architecture. In *Proceeding of the 2008 conference on Artificial General Intelligence*, pages 224–235. IOS Press, 2008. 15

John E. Laird. *The Soar Cognitive Architecture*. MIT Press, April 2012. 15

John E. Laird and Paul S. Rosenbloom. The evolution of the soar cognitive architecture. In David Steier, Tom M. Mitchell, and Allen Newell, editors, *Mind matters: A tribute to Allen Newell*, pages 1—50. Lawrence Erlbaum Associates, Mahwah, NJ, 1996. 14

John E. Laird, Allen Newell, and Paul S. Rosenbloom. SOAR: an architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, September 1987. 7, 14

George Lakoff. *Women, Fire, and Dangerous Things*. The University of Chicago Press, Chicago, 1987. 60

George Lakoff and Mark Johnson. *Metaphors We Live By*. The University of Chicago Press, Chicago, 1980. 60

Pat Langley, John E. Laird, and Seth Rogers. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160, June 2009. 14

Jill Fain Lehman, John Laird, and Paul Rosenbloom. A gentle introduction to soar, an architecture for human cognition. *Invitation to Cognitive Science*, 4, 1996. 7, 14

Ping Li, Igor Farkas, and Brian MacWhinney. Early lexical development in a self-organizing neural network. *Neural Networks*, 17(8–9):1345–1362, October 2004. 36

Alain Liégeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man and Cybernetics*, 7 (12):868–871, December 1977. 98

M. Lopes and B. Damas. A learning framework for generic sensory-motor maps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 1533–1538, San Diego, USA, 2007. IEEE. 112, 134, 163

Karl F. MacDorman. Grounding symbols through sensorimotor integration. *Journal of the Robotics Society of Japan*, 17:20–24, 1999. 20, 41

T. V Maia and A. Cleeremans. Consciousness: Converging insights from connectionist modeling and neuroscience. *Trends in Cognitive Sciences*, 9(8):397–404, 2005. 67

Jean M. Mandler. How to build a baby: II. conceptual primitives. *Psychological Review*, 99(4):587–604, 1992. 59, 60, 141

D. Marr and T. Poggio. From understanding computation to understanding neural circuitry. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, May 1976. 13

J. L. McClelland and D. E Rumelhart. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 2: Psychological and Biological Models*. MIT Press, 1986. 24

James L. McClelland and Timothy T. Rogers. The parallel distributed processing approach to semantic cognition. *Nature Reviews Neuroscience*, 4(4):310–322, April 2003. 25

James L. McClelland, Matthew M. Botvinick, David C. Noelle, David C. Plaut, Timothy T. Rogers, Mark S. Seidenberg, and Linda B. Smith. Letting structure emerge: connectionist and dynamical systems approaches to cognition. *Trends in Cognitive Sciences*, 14(8):348–356, August 2010. 21, 29, 61

Jeffrey L. McKinstry, Gerald M. Edelman, and Jeffrey L. Krichmar. A cerebellar model for predictive motor control tested in a brain-based device. *Proceedings of the National Academy of Sciences of the United States of America*, 103(9):3387–3392, February 2006. 25

David A. Medler. A brief history of connectionism. *Neural Computing Surveys*, 1(2): 18—72, 1998. 24

Biren Mehta and Stefan Schaal. Forward models in visuomotor control. *Journal of Neurophysiology*, 88(2):942–953, August 2002. 71

Bartlett W. Mel and Christof Koch. Sigma-pi learning: On radial basis functions and cortical associative learning. *Advances in Neural Information Processing Systems*, 2: 474–481, 1989. 114, 122, 132

Laurie von Melchner, Sarah L. Pallas, and Mriganka Sur. Visual behaviour mediated by retinal projections directed to the auditory pathway. *Nature*, 404(6780):871–876, April 2000. 4, 75

M. M. Mesulam. From sensation to cognition. *Brain*, 121(6):1013–1052, June 1998. 10

G. Metta, G. Sandini, and J. Konczak. A developmental approach to visually-guided reaching in artificial systems. *Neural Networks*, 12(10):1413–1427, December 1999. 28, 147

Kaspar Meyer and Antonio Damasio. Convergence and divergence in a neural architecture for recognition and memory. *Trends in Neurosciences*, 32(7):376–382, July 2009. 48, 50, 51, 75

George A Miller. The cognitive revolution: a historical perspective. *Trends in Cognitive Sciences*, 7(3):141–144, March 2003. 3

Marvin Minsky. A framework for representing knowledge. In P. H. Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, 1974. 20, 52, 53, 55, 58, 63, 64, 66

Mortimer Mishkin and Leslie G. Ungerleider. Contribution of striate inputs to the visuospatial functions of parieto-preoccipital cortex in monkeys. *Behavioural Brain Research*, 6(1):57–77, September 1982. 10

L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory–motor coordination to imitation. *Robotics, IEEE Transactions on*, 24(1):15–26, 2008. 98

Hans Moravec. *Mind Children: The Future of Robot and Human Intelligence*. Harvard University Press, 1988. 2

A.F. Morse, J. de Greeff, T. Belpeame, and A. Cangelosi. Epigenetic robotics architecture (ERA). *Autonomous Mental Development, IEEE Transactions on*, 2(4):325 –339, December 2010. 26, 27, 28, 36, 88, 158

V. B. Mountcastle. The columnar organization of the neocortex. *Brain*, 120(4):701–722, April 1997. 4, 75

R. Möller. Perception through Anticipation—A behaviour-based approach to visual perception. *Understanding Representation in the Cognitive Sciences: Does Representation Need Reality?*, page 169, 1999. 71, 73

Katharina Mülling, Jens Kober, and Jan Peters. A biomimetic approach to robot table tennis. *Adaptive Behavior*, 19(5):359 –376, October 2011. 98

Allen Newell and Herbert A. Simon. Computer science as empirical inquiry: symbols and search. *Commun. ACM*, 19(3):113–126, March 1976. 12

Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive Processing*, 12(4):319–340, 2011. 91, 93

Nils J. Nilsson. A mobile automaton: An application of artificial intelligence techniques. In *Proceedings of the International Joint Conference on Artificial Intelligene*, pages 509—520, Washington, D.C., 1969. 1, 17

Stefano Nolfi and Dario Floreano. *Evolutionary robotics : the biology, intelligence, and technology of self-organizing machines*. MIT Press, Cambridge, Mass., 2000. 92

A. Nuxoll and J. E Laird. A cognitive model of episodic memory integrated with a general cognitive architecture. In *International Conference on Cognitive Modeling*, page 220–225, 2004. 14

Dennis D. M. O'Leary. Do cortical areas emerge from a protocortex? *Trends in Neurosciences*, 12(10):400–406, 1989. 4, 75

Anders Orebäck and Henrik I. Christensen. Evaluation of architectures for mobile robotics. *Autonomous Robots*, 14(1):33–49, 2003. 18

Erhan Oztop and Michael A. Arbib. Schema design and implementation of the grasp-related mirror neuron system. *Biological Cybernetics*, 87(2):116–140, 2002. 68, 69

G. Pezzulo and G. Calvi. Toward a perceptual symbol system. In *Proceedings of the Sixth International Conference on Epigenetic Robotics*, volume 118, 2006a. 69

G. Pezzulo and G. Calvi. A schema based model of the praying mantis. In *From Animals to Animats 9*, volume 4095 of *Lecture Notes in Computer Science*, pages 211–223. Springer, Berlin, Heidelberg, 2006b. 69

Giovanni Pezzulo and Gianguglielmo Calvi. Schema-based design and the AKIRA schema language: An overview. In *Anticipatory Behavior in Adaptive Learning Systems*, number 4520 in Lecture Notes in Computer Science, pages 128–152. Springer Berlin Heidelberg, 2007. 69, 73

Giovanni Pezzulo, Dimitri Ognibene, Gianguglielmo Calvi, and Daniela Lalia. Fuzzy-based schema mechanisms in AKIRA. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'06) - Volume 02*, pages 146–152. IEEE Computer Society, 2005. 69, 97

Giovanni Pezzulo, Gianluca Baldassarre, Martin V. Butz, Cristiano Castelfranchi, and Joachim Hoffmann. From actions to goals and vice-versa: Theoretical analysis and models of the ideomotor principle and TOTE. In Martin V. Butz, Olivier Sigaud, Giovanni Pezzulo, and Gianluca Baldassarre, editors, *Anticipatory Behavior in Adaptive Learning Systems*, number 4520 in Lecture Notes in Computer Science, pages 73–93. Springer Berlin Heidelberg, January 2007. 73

R. Pfeifer and C. Scheier. From perception to action: the right direction? In *From Perception to Action Conference, 1994., Proceedings*, pages 1 – 11, September 1994. 20, 21, 60, 92

Rolf Pfeifer and Christian Scheier. *Understanding Intelligence*. MIT Press, 2001. 46

Jean Piaget. *The origin of intelligence in the child*. Routledge, London; New York, reprint of the 1953 edition, 1997 [1953]. 34, 58, 59, 62, 64, 65, 70, 71, 75, 80, 153, 156, 161, 162

Robi Polikar. Ensemble learning. *Scholarpedia*, 4(1):2776, 2009. 157

Willibald Pschyrembel and Otto Dornblüth. *Klinisches Wörterbuch mit klinischen Syndromen*. de Gruyter, Berlin ; New York, 252., durchges. u. verb. aufl edition, 1975. 133

Friedemann Pulvermüller. Brain mechanisms linking language and action. *Nat Rev Neurosci*, 6(7):576–582, July 2005. 47, 48

Zenon W. Pylyshyn. *Computation and Cognition*. MIT Press, February 1986. 12

Carl Edward Rasmussen. The infinite gaussian mixture model. In *Advances in Neural Information Processing Systems*, volume 12, pages 554–560, Denver, Colorado, USA, 2000. MIT Press. 134, 163

Rene Felix Reinhart and Jochen Jakob Steil. Neural learning and dynamical selection of redundant solutions for inverse kinematic control. In *2011 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 564–569. IEEE, October 2011. 112, 114

R.F. Reinhart and J.J. Steil. Reaching movement generation with a recurrent neural network based on learning inverse kinematics for the humanoid robot iCub. In *9th IEEE-RAS International Conference on Humanoid Robots, 2009. Humanoids 2009*, pages 323 –330, December 2009. 25

Giacomo Rizzolatti and Maddalena Destro. Mirror neurons. *Scholarpedia*, 3(1):2055, 2008. 55

M. Rolf, J.J. Steil, and M. Gienger. Goal babbling permits direct learning of inverse kinematics. *Autonomous Mental Development, IEEE Transactions on*, 2(3):216–229, 2010. 98, 99, 112

Matthias Rolf. *Goal Babbling for an Efficient Bootstrapping of Inverse Models in High Dimensions*. Doctoral thesis, Faculty of Technology, Bielefeld University, Bielefeld, Germany, 2012. 147

Frank Rosenblatt. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Spartan Books, 1962. 24

Julio K. Rosenblatt. DAMN: a distributed architecture for mobile navigation. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2-3):339–360, 1997. 96, 97

D. E. Rumelhart. Schemata and the cognitive system. In *Handbook of social cognition*, volume 1, pages 161–188. Lawrence Erlbaum Associates, 1984. 64, 66, 88, 161

D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*. MIT Press, 1986. 24, 114, 117

D. E. Rumelhart and P. M. Todd. Learning and connectionist representations. *Attention and performance XIV: Synergies in experimental psychology, artificial intelligence, and cognitive neuroscience*, pages 3–30, 1993. 24, 25

D. E. Rumelhart, P. Smolensky, J. L. McClelland, and G. E. Hinton. Schemata and sequential thought processes in PDP models. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 2: Psychological and Biological Models*, page 7–57. MIT Press, Cambridge, MA, USA, 1986. 64, 65, 66, 67, 80, 161

A. Saffiotti. The uses of fuzzy logic in autonomous robot navigation. *Soft Computing*, 1(4):180–197, December 1997. 97

A. Safiotti. Fuzzy logic in autonomous robotics: behavior coordination. In *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems*, volume 1, pages 573–578, 1997. 69

Antonio Sala, Thierry Marie Guerra, and Robert Babuška. Perspectives of fuzzy systems and control. *Fuzzy Sets and Systems*, 156(3):432–444, December 2005. 69, 97

Y. Sandamirskaya, M. Richter, and G. Schoner. A neural-dynamic architecture for behavioral organization of an embodied agent. In *2011 IEEE International Conference on Development and Learning (ICDL)*, volume 2, pages 1 –7, August 2011. 37, 38

Yulia Sandamirskaya and Gregor Schöner. An embodied account of serial order: How instabilities drive sequence generation. *Neural Networks*, 23(10):1164–1179, December 2010. 36

J. W. Scannell, C. Blakemore, and M. P. Young. Analysis of connectivity in the cat cerebral cortex. *The Journal of Neuroscience*, 15(2):1463–1483, February 1995. 10

Roger C. Schank. Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, 3(4):552–631, October 1972. 63

Roger C. Schank and Robert P. Abelson. *Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures*. Artificial Intelligence Series. Lawrence Erlbaum Associates, Hillsdale, NJ, July 1977. 63, 64, 66

W. Schenck, H. Hasenbein, and R. Möller. Detecting affordances by mental imagery. In Alessandro Di Nuovo, Vivian M. De La Cruz, and Davide Marocco, editors, *Proceedings of the Workshop on Artificial Mental Imagery in Cognitive Systems and Robotics*, pages 15–18, Odense, Denmark, 2012. University of Plymouth Press. 72

Gerald E. Schneider. Two visual systems. *Science*, 163(3870):895–902, February 1969. 10

Gregor Schöner, Klaus Kopecz, Wolfram Erlhagen, Pietro Morasso, and Vittorio Sanguineti. The dynamic neural field theory of motor programming: Arm and eye movements. In *Self-organization, Computational Maps, and Motor Control*, volume Volume 119, pages 271–310. North-Holland, 1997. 30, 31, 32, 61

REFERENCES

John R. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(03): 417–424, 1980. 12

M. Shanahan. A cognitive architecture that combines internal simulation with a global workspace. *Consciousness and Cognition*, 15(2):433–449, 2006. 38, 39, 46, 71, 73, 96

M. Shanahan and D. Connor. Modeling the neural basis of cognitive integration and consciousness. *Artificial Life*, 11:553, 2008. 38

Murray Shanahan. A spiking neuron model of cortical broadcast and competition. *Consciousness and Cognition*, 17(1):288–303, March 2008. 38, 39

Murray Shanahan. The brain's connective core and its role in animal cognition. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1603):2704–2714, October 2012. 38

Murray Shanahan and Bernard Baars. Applying global workspace theory to the frame problem. *Cognition*, 98(2):157–176, December 2005. 38

W. Kyle Simmons and Lawrence W. Barsalou. The similarity-in-topography principle: Reconciling theories of conceptual deficits. *Cognitive Neuropsychology*, 20:451–486, May 2003. 61

Herbert A Simon. The functional equivalence of problem solving skills. *Cognitive Psychology*, 7(2):268–288, April 1975. 14

Linda B. Smith and L. Samuelson. Objects in space and mind: From reaching to words. In Kelly S. Mix, Linda B. Smith, and Michael Gasser, editors, *The Spatial Foundations of Language and Cognition*. Oxford University Press, Oxford, England, 2010. 27

Linda B. Smith and Esther Thelen. Development as a dynamic system. *Trends in Cognitive Sciences*, 7(8):343–348, August 2003. 30, 34, 35, 61, 75, 156

Olaf Sporns and Jonathan Zwi. The small world of the cerebral cortex. *Neuroinformatics*, 2(2):145–162, 2004. 10

A. Stoytchev. Five basic principles of developmental robotics. In *NIPS 2006 Workshop on Grounding Perception, Knowledge and Cognition in Sensori-Motor Experience*, 2006. 21, 41

Alexander Stoytchev. Self-detection in robots: A method based on detecting temporal contingencies. *Robotica*, 29(Special Issue 01):1–21, 2011. 147

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998. 92

Henrik Svensson, Anthony F. Morse, and Tom Ziemke. Neural pathways of embodied simulation. In Giovanni Pezzulo, Martin V. Butz, Olivier Sigaud, and Gianluca Baldassarre, editors, *Anticipatory Behavior in Adaptive Learning Systems*, number 5499 in Lecture Notes in Computer Science, pages 95–114. Springer Berlin Heidelberg, January 2009. 73

Esther Thelen and Linda B. Smith. *Dynamic Systems Approach to the Develop.* MIT Press, January 1996. 29, 40, 59

Esther Thelen, Gregor Schöner, Christian Scheier, and Linda B. Smith. The dynamics of embodiment: A field theory of infant perseverative reaching. *Behavioral and Brain Sciences*, 24(1):1–34, 2001. 30, 34, 61, 75, 156

Evan Thompson. *Mind in Life: Biology, Phenomenology, and the Sciences of Mind.* Harvard University Press, 2007. 47

S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. MINERVA: a second-generation museum tour-guide robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 1999–2005, 1999. 7

V. Tikhanoff, A. Cangelosi, P. Fitzpatrick, G. Metta, L. Natale, and F. Nori. An open-source simulator for cognitive robotics research: the prototype of the iCub humanoid robot simulator. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, PerMIS '08, page 57–61, New York, NY, USA, 2008. ACM. 129, 148

Marc Toussaint. A sensorimotor map: Modulating lateral interactions for anticipation and planning. *Neural Computation*, 18(5):1132–1155, May 2006. 103, 164

Tim van Gelder. The dynamical hypothesis in cognitive science. *Behavioral and Brain Sciences*, 21(05):615–628, 1998. 29, 40

Francisco J. Varela, Evan T. Thompson, and Eleanor Rosch. *The Embodied Mind: Cognitive Science and Human Experience.* MIT Press, 1991. 46

D. Vernon, G. Metta, and G. Sandini. A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *IEEE Transactions on Evolutionary Computation*, 11(2):151–180, 2007. 21

Sven Wachsmuth, Frederic Siepmann, Denis Schulze, and Agnes Swadzba. ToBI - team of bielefeld: The human-robot interaction system for RoboCup@Home 2010. Technical report, CITEC, Bielefeld University, June 2010. 7

Cornelius Weber and Stefan Wermter. A self-organizing map of sigma-pi units. *Neurocomputing*, 70(13-15):2552–2560, August 2007. 114, 162, 163

REFERENCES

Juyang Weng, James McClelland, Alex Pentland, Olaf Sporns, Ida Stockman, Mriganka Sur, and Esther Thelen. Autonomous mental development by robots and animals. *Science*, 291(5504):599–600, January 2001. 3, 21, 41

Gert Westerman and Eduardo Reck Miranda. Modelling the development of mirror neurons for auditory-motor integration. *Journal of New Music Research*, 31(4):367–375, 2002. 36

D. M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7-8):1317–1329, October 1998. 53, 61, 72, 78, 96, 136

Daniel M Wolpert, R.Chris Miall, and Mitsuo Kawato. Internal models in the cerebellum. *Trends in Cognitive Sciences*, 2(9):338–347, September 1998. 78

Daniel M. Wolpert, Kenji Doya, and Mitsuo Kawato. A unifying computational framework for motor control and social interaction. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):593–602, March 2003. 54, 61, 72, 96

Daniel M. Wolpert, Jörn Diedrichsen, and J. Randall Flanagan. Principles of sensorimotor learning. *Nature Reviews Neuroscience*, 12(12):739–751, October 2011. 71

J. Wyatt and N. Hawes. Multiple workspaces as an architecture for cognition. In *Proceedings of AAAI 2008 Fall Symposium on Biologically Inspired Cognitive Architectures*, November 2008. 18

Yuichi Yamashita and Jun Tani. Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment. *PLoS Computational Biology*, 4(11), November 2008. 43

J. Yen and N. Pfluger. A fuzzy logic based extension to payton and rosenblatt's command fusion method for mobile robot navigation. *IEEE Transactions on Systems, Man and Cybernetics*, 25(6):971–978, 1995. 97

Laure Zago, Mauro Pesenti, Emmanuel Mellet, Fabrice Crivello, Bernard Mazoyer, and Nathalie Tzourio-Mazoyer. Neural correlates of simple and complex mental calculation. *NeuroImage*, 13(2):314–327, February 2001. 47

Appendix A

Additional Mathematical Formulations

Transformation of Input Vectors into Barocentric Coordinates of a Hypercube of Receptive Field Centers

Given an input vector $\mathbf{x} = (x_i, \ldots, x_n)$, determine the 2^n receptive field centers $\mathbf{m}^j$ surrounding $\mathbf{x}$. Let m_i^{floor} and m_i^{ceil} be the i-th coordinates of centers $\mathbf{m}^j$, such that $x_i \geq m_i^{floor}$ and $x_i \leq m_i^{ceil}$. We transform $\mathbf{x}$ into normalized coordinates inside the surrounding hypercube by defining

$$p_i = \frac{x_i - m_i^{floor}}{m_i^{ceil} - m_i^{floor}}. \tag{A.1}$$

Let $\mathbf{c}^j = (c_1^j, \ldots, c_n^j)$ be the normalized coordinates of hypercube vertices. We then compute the activations u_j of the neural field units corresponding to the hypercube vertices as

$$u_j = \prod_{i=1}^{n} c_i^j p_i + (1 - c_i^j)(1 - p_i). \tag{A.2}$$

Given activations u_j and corresponding receptiv field centers $\mathbf{m}_j$, the input $\mathbf{x}$ can be restored as

$$\mathbf{x} = \sum_{j=1}^{2^n} u_j \cdot \mathbf{m}_j. \tag{A.3}$$