Dissertation

Internal Visuomotor Models for Cognitive Simulation Processes

Der Technischen Fakultät
der Universität Bielefeld
vorgelegt von

Alexander Kaiser

zur Erlangung des Grades eines
Doktors der Ingenieurwissenschaften.

Tag der Disputation: 19. Mai 2014

Gedruckt auf alterungsbeständigem Papier gemäß ISO 9706

Abstract

Recent theories in cognitive science step back from the strict separation of perception, cognition, and the generation of behavior. Instead, cognition is viewed as a distributed process that relies on sensory, motor and affective states. In this notion, internal simulations—i.e. the mental reenactment of actions and their corresponding perceptual consequences—replace the application of logical rules on a set of abstract representations. These internal simulations are directly related to the physical body of an agent with its designated senses and motor repertoire. Correspondingly, the environment and the objects that reside therein are not viewed as a collection of symbols with abstract properties, but described in terms of their action possibilities, and thus as reciprocally coupled to the agent.

In this thesis we will investigate a hypothetical computational model that enables an agent to infer information about specific objects based on internal sensorimotor simulations. This model will eventually enable the agent to reveal the behavioral meaning of objects. We claim that such a model would be more powerful than classical approaches that rely on the classification of objects based on visual features alone. However, the internal sensorimotor simulation needs to be driven by a number of modules that model certain aspects of the agents senses which is, especially for the visual sense, demanding in many aspects. The main part of this thesis will deal with the learning and modeling of sensorimotor patterns which represents an essential prerequisite for internal simulation.

We present an efficient adaptive model for the prediction of optical flow patterns that occur during eye movements: This model enables the agent to transform its current view according to a covert motor command to virtually fixate a given point within its visual field. The model is further simplified based on a geometric analysis of the problem. This geometric model also serves as a solution to the problem of eye control. The resulting controller generates a kinematic motor command that moves the eye to a specific location within the visual field. We will investigate a neurally inspired extension of the eye control scheme that results in a higher accuracy of the controller. We will also address the problem of generating distal stimuli, i.e. views of the agent's gripper that are not present in its current view. The model we describe associates arm postures to pictorial views of the gripper. Finally, the problem of stereoptic depth perception is addressed. Here, we employ visual prediction in combination with an eye controller to generate virtually fixated views of objects in the left and right camera images. These virtually fixated views can be easily matched in order to establish correspondences. Furthermore, the motor information of the virtual fixation movement can be used to infer depth information.

Acknowledgements

First of all, I would like to thank my supervisors Prof. Dr. Ralf Möller and Dr. Wolfram Schenck for their endless support, stimulating discussion, and their plentiful ideas. I am also very grateful to Prof. Dr. Martin V. Butz for accepting to review this thesis. Furthermore, I would like to thank Prof. Dr. Barbara Hammer and Dr. Carsten Gnörlich for joining my thesis committee.

I would also like to thank the whole team of the AG Technische Informatik, Birthe Babies, Angelika Deister, Dario Differt, David Fleer, Lorenz Hillen, Michael Horst, Annika Hoffmann, Tim Köhler, Martin Krzykawski, Klaus Kulitza, Frank Röben, Wolfram Schenck, and Constanze Schwan, who contributed to the nice and fruitful working atmosphere in our group.

I would especially like to thank Alexander Spiertz who supported me during the collection of the training examples for the study in Chapter 4, and Jean Saydo who introduced me to the Blender 3D modeling software with which I created some of the figures used in this thesis.

Last, but not least, I would like to thank my friends and my parents for their great support and backing during the period of writing this thesis.

Contents

1. Introduction

Classical theories of cognition made a strict separation between perception and the generation of behavior. This separation is based on the assumption that the brain extracts abstract representations from the perceived stimuli which are then used to trigger appropriate actions based on the intentions of the agent (e.g. picking up a fruit item prior to eating). The advances in related research fields, especially propositional logic, computer science and artificial intelligence (AI) during the 60s and 70s, seemingly supported the validity of this hypothesis (for a critical review, see Pfeifer and Scheier (2001)).

Based on the initial success of classical AI, the brain became thought of as an object manipulating system (e.g. Fodor (1975); Newell and Simon (1961)). Higher cognitive abilities like memory, categorization and language were explained in terms of symbols and their corresponding manipulation. These symbols have an amodal character which means that their assignment to modal sensory sensation is arbitrary and synthetic. For example the sensory event of seeing a chair is facilitated in this notion by the perceptual system assigning the label (or symbol) 'chair' to the corresponding pixels within the perceived image. Any higher level cognitive process can readily access these symbolic descriptions.

This symbolic paradigm, often referred to as *cognitivism*, seems to strongly contradict recent findings in neurophysiology (see Barsalou (2008) for a review). Many studies in recent years have shown that cognition is a distributed process that involves areas of the brain which have been previously attributed to perception and action planning. Based on these findings, the use of amodal symbols in the brain seems more and more implausible. In consequence of this, cognitive scientists have developed new theories that step back from the cognitivist view that perception, cognition and the generation of behavior (action) are strictly separated (e.g. Barsalou (1999); Grush (2004); Hesslow (2002); Möller (2000); Wilson (2002)). These new theories can largely be subsumed under the term *grounded cognition* (Barsalou, 2008). A common theme of these novel theories is that higher cognitive abilities are grounded in affective, sensory and motor processes. The renunciation from the idea that the brain assigns amodal symbols to experienced stimuli has lead to a reformulation of the theory of how the brain stores and processes information. Symbols are now thought of to be modal—i.e. directly related to the way they are sensed. This gives rise to the conception of the brain as an perceptual symbol system (Barsalou, 1999) in which dynamic sensorimotor simulation take over the role previously assigned to symbols and rule based reasoning.

Robotics has benefited to a certain degree from the advances in artificial intelligence, but was also always limited by the shortcomings of classical AI. The main problem is that a robot is typically immersed in an environment that is variable and rich of stimuli. The manipulation of amodal symbols is native to a digital computer and poses little problems. But the assignment of symbols to experienced stimuli (i.e. through a camera) is a rather hard task because these kind of data are noisy and often reside in high dimensional spaces.

Therefore, we advocate the shift from amodal representations to sensorimotor simulations and that rely on modal representations as it has been undertaken in theories of cognition already (Clark and Grush, 1999; Pezzulo et al., 2011). Because sensorimotor simulations make use of representations that are directly linked to the sensors and robot kinematics, novel computational methods are needed. Theses methods have to cope with the high dimensionality of certain modalities (especially vision) and yet be computationally efficient.

The aim of this thesis is twofold: First of all, we will investigate a computational model for simulated object interactions, and in the later chapters will we deal with the various problems in connection with visual data, and propose corresponding solutions. The computational model is based on the principals of grounded cognition and related theories which will be outlined in the following.

1.1. Theoretical Foundations

1.1.1. Grounded Cognition

Recent theories of cognition reject the presence of amodal symbols and semantic memory in the brain (Barsalou, 1999, 2008; Grush, 2004; Pezzulo et al., 2013; Wilson, 2002). In this new stance, cognition is grounded in sensory, affective and motor processes. A large body of experimental findings supports this new view. The main prerequisites for grounded cognition is an embodied agent who is situated in a certain environment. Therefore, the term *embodied cognition* or simply embodiment is often used to describe this new theory (Wilson, 2002).

The term embodied cognition however, i.e. the way how an agent's body with certain characteristics (i.e. sensors and motor repertoire) influences and shapes its mind, is too narrow to describe all aspects of cognition. Current bodily states are important, but in many cases not necessary for cognition (Barsalou, 2008; Wilson, 2002). One prime example is mental imagery.

Mental imagery is the act of mentally simulating previous experienced sensory and motor states, and the generation of novel states based on memory (Kosslyn et al., 1997). Thus, imagery is clearly decoupled from the current sensory and motor situation. Although recent theories and neurophysiological findings support that mental imagery is grounded in sensorimotor processing, it is performed purely off-line (Barsalou, 2008; Wilson, 2002).

Another example is abstract reasoning. While the connection between mental imagery and sensorimotor states is obvious, the claim that reasoning is in a way grounded is more difficult to grasp. However, there is a large body of evidence indicating that even abstract forms of thinking rely on sensorimotor processing

(Barsalou, 2008; Wilson, 2002). These findings suggest that the brain forms and stores bodily or modal metaphors of abstract concepts rather than utilizing amodal symbols and logical calculus as classical theories suggest (Lakoff and Johnson, 1980). Wilson (2002) gives the example of the abstract concept 'communication' that might be encoded by the metaphor of physically moving matter from one person to another.

In summary, the basis of grounded cognition is a modal representation of knowledge in a sensorimotor fashion. Reasoning and mental imagery are forms of mental simulations that utilize the very same neural correlates that are active during perception and action execution (Grush, 2004; Hesslow, 2002, 2012). The body and the environment are reciprocally coupled, but the mind can also work in an off-line manner (Barsalou, 2008; Grush, 2004; Hesslow, 2002, 2012; Wilson, 2002). Proponents of grounded cognition have also applied this theory to aspects of language and social cognition (Barsalou, 2008)—a thorough review of all aspects is therefore clearly beyond the scope of this thesis.

1.1.2. Related Concepts

Affordances

Cognitivist theories state that the perceptual system extracts a high-level description from the senses that is presented in a symbolic way to the cognitive processor. In this view, objects are represented as an aggregation of symbols from which a cognitive process infers a certain meaning (the final percept) based on a semantic memory. For example a chair may be characterized by its components: a seat, legs, a backrest, and armrests. Whenever an agent perceives an object assembled that way, the cognitive processor might infer that the agent is facing a chair. The environment in which an agent is typically immersed is however complex, rich of stimuli, and fuzzy. The extraction of high-level symbolic descriptions from the sensory inflow is thus a very hard problem.

In theories of grounded cognition, the agent is reciprocally coupled with its environment (embodied), and consequently all objects that reside therein. Furthermore, the shape of the agent's body and its current intention influence the meaning of certain objects (to the agent). Therefore, the symbolic approach to characterize objects can no longer be applied.

The ecological theory of (visual) perception (Gibson, 1979/1986) offers a more suitable way of characterizing the environment which is more in line with current grounded (or embodied) theories of cognition. Instead of subdividing the perceived sensation into finer chunks (or symbolic primitives), ecological psychologists describe the environment based on the action possibilities it offers. The action possibilities, termed *affordances*, are directly linked to the bodily capabilities of the perceiving agent. For a human observer, a chair for instance may offer the possibility for sitting down to get some rest while a cat may perceive the chair as a look-out for spotting potential prey.

The ecological theory can be characterized as a form of situated cognition, because the brain is granted only a minor role in the perception process. The environment itself already contains most information. The picking-up of affordances is only vaguely described as a resonance between the perceptual environment and the brain (Gibson, 1979/1986).

Grounded theories of cognition regard the picking-up of affordances as an internal sensorimotor simulation based on previous experience that is triggered by experienced stimuli (Svensson et al., 2009; Möller and Schenck, 2008). Therefore, we will only borrow the concept affordance from ecological psychology with the above definition.

Sensorimotor Contingencies

Another concept which is closely related to affordance is that of *sensorimotor contingency* (O'Regan and Noë, 2001). The concept is rooted in the *theory of active perception* that is at its core linked to ecological psychology. Sensorimotor contingencies are the law-like relations between actions and their sensory consequences. Gibson (1979/1986) refers to theses laws as *ecological physics*, because they bear certain similarities to the physicals laws that govern the environment. In contrast to the classical concept of physics, sensorimotor contingencies are inseparably connected to the body and the senses.

According to O'Regan and Noë (2001), perception and visual conciousness emerges through mastery of sensorimotor contingencies. They motivate this by claiming that, to the brain in its early stages (in a developmental sense), the many sensory inputs and motor outputs form seemingly idiosyncratic patterns. The ability to perceive and to attend to things (e.g. objects in the environment) emerges through a learning process during which the brain starts to recognize certain regularities within these patterns. Of course, the learning of theses patterns, the sensorimotor contingencies, requires active exploration (O'Regan and Noë, 2001).

In the context of visual perception, O'Regan and Noë (2001) identify two kinds of sensorimotor contingencies: those that are related to the oculomotor system and those that are related to the visual attributes within the environment. The first kind is relatively clear: it describes the influence of eye movements onto the visual sensation. This kind of sensorimotor contingency lets us for example perceive a straight line as straight, although it appears curved on the retina due to the inhomogeneous distribution of photoreceptors. The second kind encompasses the changes of visual attributes such as shape when moving around. These contingencies are therefore influenced by various external factors such as distance and illumination. Furthermore, when moving around, certain portions of the object that were previously occluded may come into sight. Still, the perceived object stays the same (i.e. the perceived quality of the object is invariant under these transformations).

We won't go any deeper into this theory and its implications for perception at this point as we just wanted to clarify one of its key concept—namely that of sensorimotor contingency—which is a useful term that will reoccur during later sections. We furthermore note that the theory of sensory motor contingencies is at certain points incompatible with the grounded view that we are going to pursue during this thesis. For example, the authors deny the existence of representations (O'Regan and Noë, 2001)—but representations, in our view, are, to a certain degree, important for internal simulations, and thus the basis for cognition.

In the next section, we will introduce the notion of an internal model which originates from theories of motor control. A specific internal model, the so-called forward model, that mimics the forward characteristics of a plant (e.g. the oculomotor system) represents a way of modeling sensorimotor contingencies. While we conceive a sensorimotor contingency as a law-like relation which is inherently present in a certain perceptual apparatus or the environment, we conceive an internal model as a device to capture and implement such a relation within the cognitive architecture of the agent.

Internal Models

The notion of an internal model is a concept from computational theories of motor control (Ito, 2008; Karniel, 2002; Kawato and Wolpert, 1998; Wolpert et al., 1995, 1998; Wolpert and Kawato, 1998). Internal models are crucial for movement planning and are hypothesized to play an important rule in internal simulations (see Section 1.1.3 below). Because of their formal nature, internal models can be readily implemented in computational frameworks. Large portions of this thesis will be concerned with the efficient implementation of internal models.

In the following, we will highlight the main characteristics of internal models, and will only go as deep into the theory of motor control as needed to explain where internal models originate from. The body of neurophysiological findings that support the existence of such models is overwhelmingly large and will not be considered as we will mainly focus on their functional role (see e.g. Wolpert et al. (1998)).

In general, as the name suggest, an internal model mimics the characteristics of a certain plant (e.g. musculoskeletal system in the context of motor control) in either inverse or forward direction (Kawato and Wolpert, 1998). Thus, the literature dichotomously divides internal models into inverse and forward models. The role of inverse models that act as motor controllers is obvious: in order to e.g. move a limb from an initial position to a goal position, an appropriate motor command needs to be generated. The controller (inverse model) takes as inputs the current limb position (through kinesthesis) and information about the goal location and issues an appropriate motor command that moves the limb such that the goal location is reached (subject to a certain bias and error).

This seemingly simple control scheme suggests that limb movements are ballistic (i.e. neglecting any positional information during the movement), and thus require the inverse model to compute a complex motor plan beforehand (Grush, 2004). Many experimental findings contradict this assumption and postulate that motor control relies on continuous feedback from the musculoskeletal system. In such a feedback control scheme, the controller (inverse model) receives an error signal which is the deviation between the current position and the goal position and which it tries to gradually minimize. Such a control scheme does not require a sophisticated motor plan beforehand as the optimal control signal (i.e. that transitions the limb from initial to goal position) emerges smoothly over time (Grush, 2004).

The closed-loop feedback control scheme requires the continuous evaluation of the current limb position through kinesthesis. This feedback causes a large delay which exceeds the timings of certain movements, and thus fails to explains the corresponding observations in motor control. An early explanation for these effects was given by von Holst and Mittelstaedt (1950) through their notion of the efference copy. An efference copy is generally a copy of the motor command sent to the muscle that is fed into the controller. On the basis of this efference copy the controller is enabled to anticipate the corresponding results of the motor command, and may operate on this anticipated information rather than feedback from the musculoskeletal system (von Holst and Mittelstaedt, 1950).

However, the motor command received through the efference copy likely to be in a different format than the feedback received through kinesthesis (Kawato and Wolpert, 1998). The original theory of efference copy failed to explain how the motor command is transformed. Recently, researchers suggested the notion of forward models to remediate this shortcoming of the original proposal. A forward model mimics (emulates) the forward characteristics of a certain plant (subject to small errors and biases). Its purpose is therefore to transform, in a predictive way, motor commands into the corresponding format required to generate a proper error signal. In the case of motor control, the forward model predicts the kinesthetic information that would be sensed if a certain motor command would be executed.

Besides motor control, researchers hypothesized that forward models are involved in many other contexts including the prediction of sensory consequences of self-induced actions, and even cognition (Grush, 2004; Ito, 2008; Karniel, 2002; Wolpert et al., 2003). One prominent effect that can be easily observed is the fact that most people cannot tickle themselves. A hypothesis states that a somatosensory forward model predicts the self-induced tickling sensation and consequently suppresses the real stimulus sensed by the skin (Blakemore et al., 2000).

A similar finding suggest that even the visual sense is equipped with a mechanism that anticipates the sensory effect of eye movements (Duhamel et al., 1992). This mechanism might explain the phenomenon of visual stability, i.e. that the visual impression seems to be stable despite the permanent execution rapid eye movements (saccades) which induce shifts of visual stimuli on the retina. A study on single cell recordings in monkeys revealed that the parietal cortex performs such a predictive remapping of receptive fields in anticipation to eye movements (Duhamel et al., 1992). The direction of the remapping directly corresponds to the location on retina on which a certain stimulus appears after the eye movement.

1.1.3. Simulation and Emulation Theories

As mentioned above, the paradigm of internal sensorimotor simulation is a fundamental paradigm in grounded cognition that distinguishes it from theories of purely embodied and situated cognition. Furthermore, for the derivation of computational models, we need a clear formulation of internal simulation. Therefore, we will now take a closer look at several simulation theories that were proposed by various authors in recent years.

Möller (2000) proposed that perception is the result of an internal simulation of actions between an agent and the environment and the subsequent evaluation thereof. In this paradigm—*perception through anticipation*—the agent generates a tree-like structure of possible interactions based on its current sensory input. The agent remains passive throughout the perception process and manipulates the current sensory situation based on previously learned sensorimotor patterns or sensorimotor contingencies.

In the context of perception of space and shape, Möller (2000) gives an intuitive example of an agent, a robot equipped with an image sensor, that faces an arrangement of cylindrical obstacles. In this example, the agent has to determine whether the obstacles form a passage that can be crossed or a dead-end. The criterion that determines whether the arrangement is passable is reciprocally coupled to the size of the agent. In order to separate his theory from others, Möller (2000) mentions three different possible ways for implementing such a capability. The cognitivist way would consist in a reasoning step that determines if the arrangement is passable based on an analysis of the initial sensory situation, neglecting any motor information. This agent would have to deal with invariances caused by the perspective and distance towards obstacles. Therefore it would have to be equipped with a sophisticated perception module. On the other hand, a reactive agent, equipped with a simple collision avoiding strategy, would actively explore the arrangement. The reactive agent has the disadvantage that it might get stuck during the exploration, and would consequently not find a passage at all. Finally, the anticipating agent would explore the arrangement "mentally" based on the initial sensory situation during multiple alternative exploration runs. The internal simulations are subsequently evaluated to determine whether the arrangement is passable or not based on motor information. Thus, the anticipating agent is a reactive agent that is capable of simulating actions based on experience.

In the context of consciousness, Hesslow (2002) postulates three assumptions that are the basis for his *simulation theory*: (i) motor structures involved in overt action execution are also active during simulation of behavior, (ii) the sensory cortex is involved in internal simulation of percepts, and (iii) simulated and executed actions both "can elicit perceptual simulation of their normal consequences". The third assumption implies the ability to anticipate the sensory effect of covert actions.

The simulation theory is basically in line with the behaviourist view that a certain stimuli trigger specific actions. (This corresponds to the reactive agent mentioned above.) The action is not necessarily executed but may be suppressed, and the sensory consequences are anticipated. As in the perception through anticipation hypothesis, this results in chains of simulated action–response pairs (Hesslow, 2002).

Anticipation of internal sensory states is a central idea shared by both theories. The neural basis for anticipation might emerge during the developmental stages of an organism through an *anticipatory drive* (Butz, 2008). This metaphorical drive might facilitate the formation of representations that are especially suited for forward predictions. These forward predictions are the basis for internal simulations in which they are responsible for generating internal future sensory states.

However, there are also two striking differences between Hesslow's simulation theory and perception through anticipation. The simulation theory is about conscious thought: that is, the anticipation is initiated and evaluated through consciousness. On the other hand, perception through anticipation is an unconscious process that results in a percept that is subsequently transferred to consciousness. Thus, the anticipation process itself is not directly accessible by consciousness. The second difference lies in the representation of sensory states and possible structures involved in the anticipation process. The simulation theory is very general and does not require a specific representation. In the perception through anticipation hypothesis, sensory states are directly linked to the low-level representation of the sensory modalities.

In a later article, Möller and Schenck (2008) presented a computer simulation of the robot experiment mentioned above: a simulated robot that can move in the 2D plane, and equipped with an omnidirectional sensor, is facing an obstacle arrangement similar to the thought experiment. The robot is equipped with forward and inverse models that predict the optical flow of the objects as if the robot moved, and generate appropriate motor commands, respectively. These internal models are used to drive the internal simulation, based on an initial sensory situation. The notion an internal model plays a crucial role in this instantiation of the perception through anticipation paradigm. In contrast, Hesslow explicitly avoids this notion in his simulation theory, stating that a general association mechanism is involved in the anticipatory generation of covert sensory states (Hesslow, 2012).

Internal models—termed emulators in this context—also play a crucial role in Grush's *emulation theory of representation* (Grush, 2004). One of the goals of this theory is to provide a formal theoretic framework which allows to explain aspects of motor control and motor imagery as well as giving prospect of explaining higher cognitive abilities such as reasoning and language. Besides internal models, Grush's framework incorporates the Kalman filter—a mathematical model from the area of signal processing (Grush, 2004). In general, a Kalman filter is a state estimator whose central part lies in recursively updating an estimate (for an unknown state) based on the comparison between a (noisy) measurement and its corresponding prediction. The Kalman filter is used in combination with an "articulated emulator", i.e. an emulator that models the same input–output relation as the plant and is part of the Kalman filter, to reduce the sensory error as long as sensory input is available.

This framework extends first of all the feedback control scheme mentioned in

Section 1.1.2: the sensory signal emanating from the plant is not entirely replaced by the forward model, but integrated with the forward model's prediction through the Kalman filter. This enables the control loop to run off-line (i.e. in the absence of actual sensory inflow) and online, whereby the Kalman filter refines the sensory signal by taking into account the prediction (Grush, 2004). The weighting strength of the two signals (prediction and actual) is regulated by the so-called Kalman gain. In the case of perception the more reliable signal gets a higher weight. In the case of internal simulation (detached from the current sensory inflow) the corresponding sensors are assigned zero weights.

In case of motor control, the role of the emulator is clear and closely linked to the original notion of a forward model. In the case of visual perception (and visual imagery accordingly), Grush suggest a more complex control scheme. This extended scheme relies on different emulators that account for the anticipation of visual changes due to ego-motion and an amodal environment emulator—amodal in a sense, that this emulator predicts positional information of objects in the environment rather than producing output directly linked to the sensory representation (i.e. visual) (Grush, 2004). The dichotomous separation of ego-motion vs. environment emulators reflects the likewise separation of sensorimotor contingencies (O'Regan and Noë, 2001).

1.1.4. Experimental Evidence

In this section, we will give a short account of experimental evidence from psychology and neurophysiology that underpins the theory that internal sensorimotor simulations are involved in perceptual and cognitive tasks. Many more accounts can be found in the references of the articles in the last section.

The mental rotation experiment of Shepard and Metzler (1971) is recognized as a prime example for an internal simulation taking place and often cited by proponents of these theories. In their experiment, subjects were shown a pair of complex geometric objects. The task was to determine whether the shown objects were the same or mirrored versions of one another. An object was presented in a rotated fashion (to a certain degree) with respect to the reference. The surprising result of the experiment was, that the time a subject needed to draw a conclusion, is proportional to the angle of rotation. This gives rise to the assumption, that the brain mentally rotates the object until it matches the orientation of the reference object. The recent advances in imaging technology have enabled researchers to peek into the brain (albeit on a quite coarse scale) and to reveal the spatial location of neural activity patterns. Such an analysis of subjects engaged in the mental rotation task revealed that, besides visual areas, motor areas are also involved (Lamm et al., 2001). This gives rise to the assumption that mental rotation is indeed an act of sensorimotor simulation.

In the context of mental imagery, a long-standing debate of how mental images are represented, has been going on among cognitive scientists (see e.g. Pylyshyn (1981)). Proponents of the picture hypothesis claim that mental images are represented in an analogue fashion and are thus picture-like (Kosslyn et al., 1997). Finke and Kosslyn (1980) conducted a psychophysical imagery experiment in which subjects should judge the resolution of a specific dot pattern in the peripheral visual field either directly or by introspection (imagery). The human retina has a inhomogeneous layout of photoreceptors that decreases towards the borders of the visual field, thus making the central part of the retina (the so-called fovea) more accurate than the periphery. A correlation between judgements of resolution in the two conditions (direct vs. imagined) would give rise to a similar representation of mental images (with respect to retinal acuity). Prior to the experiment subjects were separated into two groups: vivid imagers and non-vivid imagers through an assessment of the Vividness of Visual Imagery Questionnaire (Finke and Kosslyn, 1980). The result of this study is, that vivid imagers showed a similar judgement of peripheral resolution in both task conditions. Furthermore, the study gives rise to the hypothesis that mental images are not abstract descriptions, but directly linked to the sensory (i.e. retinal) representation.

In later decades, research on mental imagery mainly centered around imaging studies by functional magnetic resonance imaging (fRMI) and positron emission tomography (PET). These studies revealed that the visual cortex is indeed involved during imagery which enforced the assumption that mental images share the same neural basis as perception (Kosslyn et al., 1993, 1997). This finding gives rise to the validity of the picture hypothesis.

Similar findings were obtained by studies on motor imagery and specifically mental practice (Jeannerod, 1994, 1995, 2001). Mental practice refers to the covert reenactment of certain actions with the goal of improving one's own motor capabilities—a type of training often performed by athletes. During mental practice activates the same motor areas that are active when an individual performs the corresponding task.

In the field of neurolinguistics, in which the neurophysiological origin of language is studied, researchers recently found evidence that sensorimotor processing is involved in the comprehension of words (Pulvermüller and Fadiga, 2010). The findings suggest a strong linkage between semantics and certain motor areas. For example hearing an action-related word like "kick" evokes activity in motor-areas of the brain that are typically associate with the execution actions that involve the corresponding part of the body (Pulvermüller and Fadiga, 2010).

There are many more accounts in the literature that indicate that all levels of cognition are indeed grounded. A thorough review would thus be well beyond the scope of this introduction. In the following, we will switch to the modeling perspective, and give an overview over relevant robotic studies which are largely in line with the grounded theory of cognition.

1.2. Learning of Sensorimotor Interactions

In this section, we are going to review several robotic studies that involve the learning of sensorimotor associations and demonstrate the capabilities of grounded cognition and simulation/emulation theories. This review is not meant to be exhaustive, but to cover a wide variety of different learning approaches, and to give a general overview of the different techniques that have been so far successfully applied.

Tani and Nolfi (1999) present a study in which a robot learns a structurally organized internal representation of the world. They used a mobile robot that traveled through two different rooms, connected by a door. The robot was equipped with an array of 20 range sensor pointing in forward direction. The task was to learn the sensorimotor relations that the robot experiences as it travels through its environment. In a later stage, these learned sensorimotor dynamics should serve as a basis for navigation.

The authors employed a hierarchical recursive neural network (RNN) approach to learn the sensorimotor dynamics of the robot (Tani and Nolfi, 1999). The main focus of the paper is not a specific application (although they mention navigation as a possible one), but to study the dynamics of the hierarchical RRN and the overall feasibility of this approach. They conclude, that the RRN was able to capture the underlying sensorimotor dynamics of the traveling robot, but also discuss the lack of goal-directedness of this approach.

In a similar vein, Ziemke et al. (2005) present a minimal neuronal model that allowed a robot to navigate blindfolded in a previously explored environment. Furthermore, they demonstrate that an internal multi-step prediction (Hesslow, 2002) (i.e. when the predicted sensory input is used instead of actual sensory input over a course of multiple time-steps) based on RNNs as used by Tani and Nolfi (1999) proved unsuccessful during this task. They also used a (simulated) mobile robot equipped with 8 proximity sensors (6 pointing in forward direction, two pointing backwards). The successful architecture they propose consists of two feed forward networks: (i) a controller network that maps sensory information to motor commands, and (ii) a predictive network that integrates sensorimotor information to predict the resulting sensory state (Ziemke et al., 2005). This architecture nicely reflects the dichotomy of inverse models (controllers) and forward models (emulators/state predictors) (Wolpert and Kawato, 1998). Equipped with this architecture, the robot was able to navigate blindfolded for “hundreds of time steps” (Ziemke et al., 2005).

The above study represents an implementation of the simulation theory of concious thought (Hesslow, 2002): The robot learned the sensorimotor relations as it traveled through its confined environment. Based on these relations the agent was able to generate an internal multi-step prediction and navigate blind-folded. However, the internal simulation could only reenact situations within the previously explored environment, and is thus most likely unable to generalize to novel situations.

Hoffmann (2007a) presents a study in which a mobile robot was placed inside a nearly circular arrangement of obstacles of different sizes. The task was to simulate movements towards these objects in order to judge the distance based on corresponding motor information. The experiment was conducted by using a real mobile robot. In contrast to the other two studies presented above, the robot was equipped with an omnidirectional vision sensor. The dimensionality of the sensory input is thus significantly higher. A forward model implemented by a feed-forward network was used to drive the simulation process. The images generated by the forward model, however, became too noisy after a few simulation steps to be further used (Hoffmann, 2007a). Therefore, a de-noising operation was performed after each simulation step. The de-noising consisted in projecting patches of the image onto a previously learned manifold. The manifold was modeled by a Gaussian mixture model (Hoffmann, 2007a). The final forward model was successfully applied to to the distance judgment problem. Furthermore, the robot was able to judge whether an arrangement of obstacles represents a dead-end or if there is a passage through which the robot could escape; the robot was thus able to reveal the functional meaning (i.e. the affordance) through an internal sensorimotor simulation (Hoffmann, 2007a; Möller, 2000).

Recently, Schenck et al. (2012) have put the computational thought experiment on anticipatory dead-end recognition (Möller and Schenck, 2008) to a test in the real world. They used a mobile robot equipped with an omnidirectional vision sensor that was placed inside an arrangement of obstacles. As in the previous though experiment (Möller and Schenck, 2008), the robot had to determine whether an arrangement is passable or represents a dead-end. An adaptive forward model was used to predict the changes of object positions within the camera image based on a given motor command. An inverse model was used to control the simulated movements. Based on an internal simulation of possible movements, the robot could make a judgement whether it could escape the arrangement or not. The robot was successful in most trials.

Both of the above studies represent successful implementations of the perception through anticipation paradigm (Möller, 2000). The agents in both studies learned to associate motor commands and their corresponding sensory effects. Based on these associations the agents were able to perform covert movements and to anticipate their sensory consequences. These covert movement sequences were evaluated to infer information about the environment. In the first study, the simulated motor commands were used to determine the distance towards obstacles. Furthermore, in both studies, the internal simulation enabled the agents to reveal the functional meaning (i.e. the affordance) of their environment, and thus to discriminate dead-ends from passages. The learned sensorimotor relations do not adhere to a specific environment, but represent general knowledge and can thus be applied to novel situations as well.

The first account of a quasi-humanoid robot that planned its actions based on a sensorimotor simulation is that of Murphy (Mel, 1988, 1991)—interestingly predating most corresponding simulation/emulation theories of cognition (Grush, 2004;

Hesslow, 2002) by nearly a decade. Murphy was equipped with a 3-DOF planar arm (with shoulder, wrist, and elbow joints) and a camera. The camera had its image plane aligned in parallel to the planar workspace of the arm. The arm was marked by white spots such that it could be easily tracked by the camera. The task for Murphy was to reach for a goal position (given in sensory coordinates) (Mel, 1988), and, in a later study, to plan its trajectories without colliding with obstacles (Mel, 1991).

Murphy could operate in two modes (Mel, 1988): an overt mode and a covert (simulation) mode. During the overt mode, Murphy learned the sensorimotor associations between joint-angle configurations and the corresponding positions of the white spots within the camera image. During the covert mode, Murphy could optimize its movement trajectories towards a goal location or plan trajectories in order to circumvent obstacles. All inputs and outputs are represented in a discrete binary manner. The images were thresholded with respect to a salient color. The underlying connectionist network consisted of retinotopic and motor maps. The associations were learned by a network of so-called of sigma–pi units that correspond to logical disjunctions of conjunctions (Mel, 1988, 1991). The corresponding units can be either 0 or 1, depending on the assignment of the inputs.

Nishide et al. (2008) present a study in which a robot learned to predict object dynamics through a series of object interactions. They used a humanoid robot that pushed objects with its left arm at two different heights above a table. One pushing sequence consisted of 7 time-steps. The results of the pushing actions (i.e. the appearance of the object at the different time-steps) were recorded by a camera. The authors conducted two experiments: (i) using artificial objects, and (ii) using real-world objects. A recursive neural network with parametric bias (RNNPB) was used to learn the visual dynamics of the object interactions. The parametric bias (PB) space self-organized such that objects that exhibit a similar dynamical behaviour were closely grouped within PB space(Nishide et al., 2008). Furthermore, a hierarchical neural network was used to associate the motor command and the static image of the object to the corresponding parametric bias in order to drive the prediction by the RNNPB.

The results show that the robot was able to predict the dynamical motion of different objects and that the clustering within PB space reflects the possible outcomes of pushing the different objects (e.g. elongated objects fall over, round objects roll). Furthermore, the study on real-world objects indicates that the model is able to generalize and successfully predict the motions of objects not shown during the training (Nishide et al., 2008).

A study presented by Montesano et al. (2008) deals with the learning of affordances which they describe as the "relation between actions objects and effects" (Montesano et al., 2008). They propose a developmental approach that comprises three phases (Montesano et al., 2008). In the first phase, the robot should acquire "basic skills" that encompass basic motor skills and visual object perception. The second phase consists in the learning of world interactions which subsumes the perception of action effects, the improvement of motor skills, the learning of affordances, and the acquisition of prediction and planning skills. At the highest level, the third phase, stands the imitation of observed actions.

The model was implementation on a robot that consisted of an arm and a camera head. The learning of perceptual skills was circumvented by a built-in system that was able to categorize objects based on geometric features (Montesano et al., 2008). The authors chose a probabilistic approach (i.e. a Bayesian network) for the implementation. The key feature of such a probabilistic modeling approach is that it can deal well with uncertainty and noise (Montesano et al., 2008). Their developmental approach enabled the robot to imitate the experimenter in an interactive game that involved the interaction with different objects placed in front of the robot (Montesano et al., 2008).

Marques and Holland (2009) present a minimal computational architecture for functional imagination, i.e. the act of interacting with the environment in the absence of actual sensory inflow (i.e. mental imagery). The authors identify a set of five conditions that such a minimal architecture should fulfill (Marques and Holland, 2009). The first two conditions are strongly interdependent: The agent should be able to generate state-based predictions (as consequences of possible actions), and must consequently be able to represent these alternative sensory states. The agent should furthermore posses an intentional goal that it seeks to fulfilled during the imaginary action sequence. Consequently it needs to be equipped with an evaluation routine that examines if the actions are leading towards the goal. The last condition is the presence of an action selection routine that selects an appropriate action among the set of possible actions.

The proposed architecture was implemented on a complex biomimetic humanoid robot (Marques and Holland, 2009). The task of the robot was two move a stick placed on the table in front of it as far away as possible. The robot could perform two possible actions: grasping the stick or throwing it. Furthermore, it had to select between a red and a blue stick. The task referred to the red stick only. For the internal simulation, the authors used a complete physical model the robot and its environment (Marques and Holland, 2009). Based on this internal model the robot simulated possible actions in order to determine which action moves the stick farthest away, and subsequently execute the most successful action overtly. However, the high complexity of the humanoid robot and discrepancies between the physics simulation based “forward model” rendered the experimental evaluation difficult (Marques and Holland, 2009).

1.3. Towards a Model Architecture for Simulated Object Interaction

In this section, we will develop a computational model for the simulation of object interactions. Such a model could enable an agent to learn the behavioral meaning of objects—which is closely linked to the notion of affordances. We claim that a general understanding of the behavioral meaning of objects (i.e. their affordances)

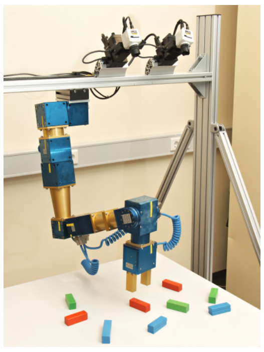


Figure 1.1.: Robotic agent for the study of object interactions consisting of a stereo camera head (2 DOF per camera) and a robotic manipulator (6 DOF) with attached two-finger gripper.

is superior to simple object recognition based on vision alone. In a later step, the learned relations could serve as a basis for the planning of complex motor plans through internal sensorimotor simulations.

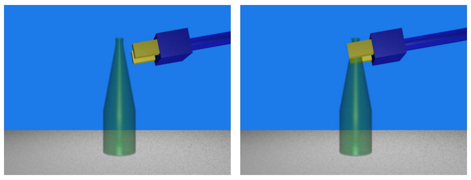
Furthermore, we will outline the problems that have already been solved and will be discussed in depth within the remainder of this thesis. There are nonetheless many open problems that still need to be cast into solid algorithms. As a result, the overall model which we will derive in the following, is not yet been fully lifted onto a level on which it can be implemented. However, the detailed description of the model and its challenges will provide a fruitful ground for further research.

1.3.1. Overview

The proposed model that we will review in detail is composed of a number of sub-models whose purpose will become clear as we introduce the problem that we seek to solve. All considerations are based on the robotic agent depicted in Figure 1.1. The agent is a simplified model of an upper human torso that consists of a stereo camera head and a serial manipulator. The cameras are both mounted on individual pant-tilt units, thus allowing for variable gaze. The robotic manipulator has 6 degrees of freedom (DOF) and an attached two-finger gripper at its end-effector position. The agent is strongly simplified with respect to DOF of the gaze: it has no neck-like structure. Therefore, gaze is only specified by the individual gaze directions of the cameras and does not involve any redundant degrees of freedom (which would be induced by a neck). Furthermore, the focal length of the cameras is fixed, so the agent cannot change its foci through accommodation.

The robot has a total of three sense: (i) binocular vision (allowing for depth perception through stereopsis), (ii) kinesthesis (i.e. the six angles of the current joint configuration), and (iii) a tactile sense that is restricted to a binary value which is based on whether the gripper can be fully closed or not.

Based on these senses, the agent should learn the sensorimotor contingencies that


(a)

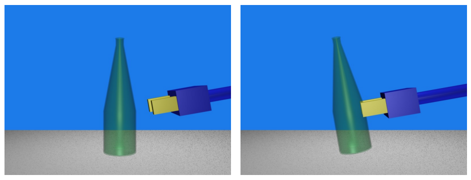

(b)

Figure 1.2.: Example for two possible gripper–object interactions: (a) gripper encloses object, (b) gripper displaces object.

arise during the interaction with objects placed before him. As soon as the agent has learned these sensorimotor contingencies, it is enabled to internally simulate possible object interactions. Here, we pick up the perception through anticipation paradigm in that the evaluation of these internal simulations finally facilitate the perception of object affordances.

Before we start to give a detailed description of the different internal models, we give a small example of how the learning or exploration phase may look like. For this, we consider two differently shaped objects: a elongated cylindrical object (e.g. a glass) and a conical object (e.g. a bottle). During the exploration phase, the agent should learn how to grasp an object. This could be done by selecting a set of potential grasp points on the surface of the object. Let us furthermore assume, that a motor controller that enables the agent to reach for points within its field of view is already available. Now the agent starts to reach for the specified grasp point with its gripper fingers at maximum opening. The visual consequences of this can be either that the object is enclosed by the gripper fingers, thus occluding part of the object, or that the object is displaced, because either the object's width exceeds the opening width of the gripper, or the grasp-point lies beyond the length of the gripper fingers, resulting in a collision between the gripper base and the object. Figure 1.2a depicts the scenario when the gripper approaches a "secure" grasp point, i.e. a not causing a collision. In Figure 1.2b, the grasp point lies within a portion of the object where its size exceeds the maximum opening of the gripper, thus resulting in a collision.

The two latter consequences that result in a displacement are undesirable and the corresponding actions should be avoided. The cases where the gripper encloses the object should trigger a closing of its fingers. After the gripper has closed its fingers, a tactile feedback should indicate that the object could be grasped. In this simplified setting, the gripper is alway oriented in parallel to the ground plane. That means that the agent does not need to decide how the gripper should be oriented prior to a grasping approach. In a more complex scenario, objects with differently oriented handles could be used where the corresponding gripper orientation varies as well. For now, we will restrict our considerations to the above scenario.

The above example shows what sensorimotor contingencies are involved, and lets us cast these into formal descriptions of the corresponding internal models.

Obviously, the agent needs the ability to predict the appearance of its own gripper as it approaches the object, which is similar to an "image emulator" in Grush's (2004) terminology. Such an internal model could be implemented by learning the association between views of the gripper and the corresponding kinesthetic states (as in Mel's (1988) model). Furthermore, the model would need knowledge of the current viewing direction of the cameras. The "environment emulator" that predicts the effects of interactions between the agent and objects is more demanding. Here, we have identified two different results of such an interaction: (i) the gripper encloses the object, and (ii) the object is displaced. The former situation can be solved by means of information already present in the sensory situation plus information from the prediction of the gripper's appearance. The second case, however, might result in portions of the object becoming visible that have be out of view before the interaction. This problem is known as sensory aliasing, and cannot be solved based on the information currently available to the agent. Therefore, we propose a model that only predicts if such a displacement event would occur or not. A third model involved is the tactile associative model that predicts the tactile sensation if the gripper would be closed at a certain position.

The sensory and motor states of the agent can be summarized as follows, and will be assigned mathematical symbols for convenience:

- current sensory input (visual): view of the object O, view of the gripper G; the image containing both, object and gripper will be denoted by $O \cup G$
- kinesthetic arm position $\vec{\psi}$
- viewing direction $\vec{v}$
- tactile information T

Motor commands related to the arm will be denoted by $\Delta\vec{\psi}$, saccadic motor commands (i.e. changes in viewing direction) by $\Delta\vec{v}$. It should be noted that the viewing direction $\vec{v}$ always encodes the individual pan and tilt angles of both cameras. Furthermore, all sensory input relates to a stereo pair of images. This is obviously necessary to encode the spatial position of grasp points/objects in order to guide the (simulated) robot arm towards the object and to drive the internal simulation of object interactions.

In the following, we will focus on the different internal models and the information they rely on. We will furthermore make suggestions on how visual data could be represented and on how the overall interaction between the various models might look like.

1.3.2. Visuomotor Associations

The visuomotor associative network is a function of the type $A : (\vec{\psi}, \vec{v}) \mapsto \hat{G}$ that assigns an estimate view of the gripper $\hat{G}$ to a given arm posture $\vec{\psi}$ as viewed from the viewing direction $\vec{v}$. This model is used to drive the simulated gripper

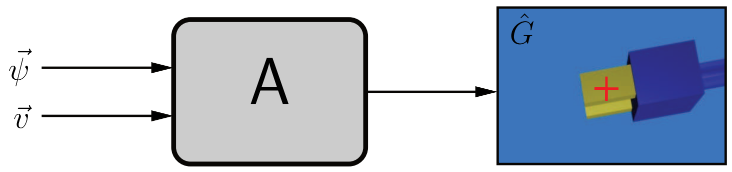


Figure 1.3.: Internal model for the visuomotor association: an image of the gripper, $\hat{G}$, is predicted based on postural information, $\vec{\psi}$, and the current viewing direction, $\vec{v}$.

approach towards the object and its output will be used to determine whether an object interaction takes place.

Figure 1.3 depicts a schema of the visual associative model. The inputs to the left side correspond to the postural variables and the viewing direction. The output on the right represents an (emblematic) example view of the gripper. Note that the gripper need not necessarily appear in the image center (fixated), but can instead appear anywhere (based on the viewing direction $\vec{v}$).

This association could be learned by the cameras observing the gripper as it moves within the manipulator's workspace. In Chapter 4, we will give a detailed description on the implementation and learning for this model. For now, we simply assume that such a model is readily available.

1.3.3. Prediction of Object Interactions

The crucial part of the architecture is the agent/environment emulator that predicts the occurrence of possible changes in the environment based on actions of the agent. In principle, such a model would need knowledge of the laws of physics and thorough information on the geometrical properties of the agent/environment. Therefore, we restrict the object interaction model to predict only results that can be extracted from the given sensory data (i.e. stereoscopic view of the object and gripper, kinesthetic state).

The observation of the different outcomes of a gripper approach towards the object, i.e. non-collision (Figure 1.2a) vs. collision (Figure 1.2b), should be handled differently by the interaction model. The main task of the object interaction model, which we shall denote model B in the following, lies in the extraction of spatial information about the (simulated) gripper in relation to the object. Consider the case, when the gripper is partially occluded by the object as in Figure 1.2a, left. In this case, one gripper finger is located behind the object. If this gripper is closed in this situation, it will experience a tactile signal, indicating that the corresponding grasp point is valid. Moving the gripper further towards the object will eventually lead to collision between the object and the gripper base. These spatial relations need to be extracted from the stereoscopic views of the gripper–object constellation in order to make a corresponding prediction.

Figure 1.4 shows a close-up of the situation when the gripper and gripper has

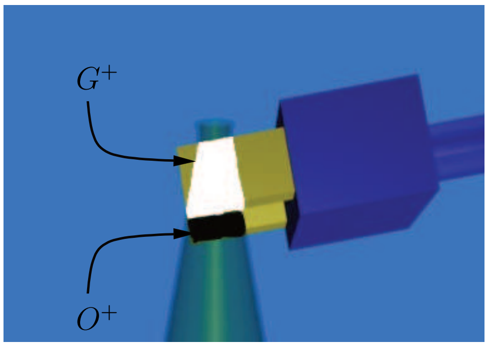


Figure 1.4.: Close-up of the gripper at a grasp point on an example object showing the portion of the gripper that occludes the object, G^+ (white), and the portion of the object occluding the gripper, O^+ (black).

reached a valid grasp point. Gripper and object mutually overlap in this situation: the right gripper finger is occluded by the object (indicated by region O^*), while the left gripper finger occludes the object (region G^+). Based on the stereoscopic appearance of the gripper and the object, the interaction model should predict this situation in order to generate a valid mental image of the situation.

We divide model B into three sub-models: B^+, B^t, and B^*. The simulation is principally driven by model $B^+ : (O, O \cup G_t, G_{t+1}) \mapsto O \cup G_{t+1}$ which predicts the future sensorimotor state $O \cup G_{t+1}$ based on the previous state $O \cup G_t$, the visual state O, and the visually coded motor command in form of the gripper image G_{t+1}. Model B^t receives the same input as model B^+ and predicts a tactile event T. These two models cover the cases when the object is not physically moved by the gripper. Model B^* receives the same input as the former models and predicts whether a collision will occur in time step $t + 1$.

We now make the attempt at a more formal level of the input-output relations of the three sub-models. As stated above, these models mainly operate in the visual domain. Therefore, we assume that all inputs are visual. We note that the motor information about the approach movement of the gripper can be translated into the visual domain by model A. We furthermore assume that the location of the grasp point coincides with the image center, rendering the view of the object independent of the current viewing direction. This last step can be achieved by virtually fixating the object at its current grasp point using a visual forward model that mimics the characteristics of the oculomotor system to predict how the image looks like.

Figure 1.5 depicts a schematic of model B in its entirety. The model receives the current sensorimotor state as a mental image of the current gripper–object configuration, denoted by $O \cup G_t$. It furthermore receives information on the pure sensory situation, O, and the motor command, encoded as the future sensory appearance of the gripper, G_{t+1}. Based on these information, the model generates a prediction of a tactile signal T, that indicates whether the gripper can be fully closed in this situation. It furthermore predicts a gripper–object collision O^*. The information

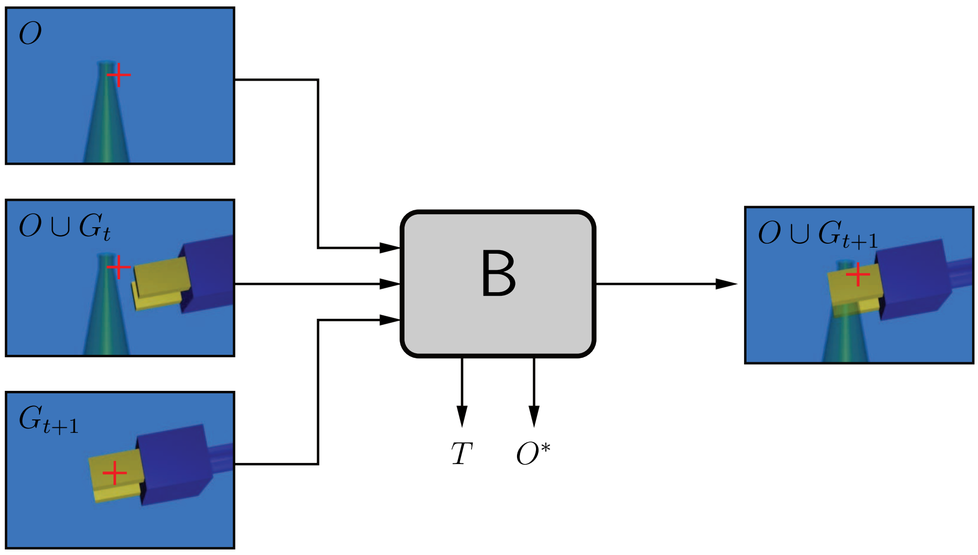


Figure 1.5.: Schematic of the object interaction model. See text for further details.

from the inputs of O and G_{t+1} are integrated to yield a prediction of the combined gripper–object image $O \cup G_{t+1}$. In the following, we will explain the overall simulation process and explain the roles of the different internal states and their dependencies.

1.3.4. Simulation Process

The task of the whole simulation process is to evaluate the possible outcomes of different grasping approaches towards an object. The simulation starts by selecting a number of potential grasp points from within the image. These grasp points are the basis for the simulated approach movements. These movements are performed in an iterative manner by performing a small movement each time-step t. The outcomes of the small movements are constantly predicted by model B.

The simulation process can be summarized as follows:

1. extract grasp-points from object image; each grasp point is given by a pair of 2D coordinates for the left and right image, respectively
2. for each grasp point, generate a fixated view of the object O
3. initiate sensorimotor simulation: generate a sequence of motor commands towards the grasp point
4. generate mental image of the gripper at each time-step t based on the motor command and the viewing direction on the grasp point (model A)
5. predict possible gripper–object interaction based on current sensory input and future gripper image

three possible cases: (i) grasp point not reached, (ii) grasp-point reached, and (iii) gripper–object collision

6. if (i) goto 4; else if (ii) evaluate tactile model B^t and continue with next grasp point; else continue with next grasp point (collision)

The outcome of this simulation process is that each grasp point is associated with a sequence of motor commands and their resulting sensory states, and information about a possible collision or a tactile sensation at the grasp point. These data can be evaluated to gather information about the grasping profile of the object which in turn could facilitate object categorization. Furthermore, the simulation can be used for planning overt grasping movements based on certain criteria, e.g. selecting grasp points based tactile sensation at the grasp point.

However, the overall simulation and its core, model B, is yet to be implemented. Therefore, we can only speculate about its main characteristics and performance in a real-world setting. We assume that the number of training examples needed will be very high (because of the underlying high dimensionality of the visual data). For this reason, an optimal training strategy will be the key to establish should a model. Furthermore, the representation of images (gray-scale, binary or features) and the internal structure of model B need to be clarified more thoroughly.

1.4. Outline

The main part of this thesis will be concerned with the development of efficient models for visuomotor prediction. The handling of visual data is especially delicate, because of its high dimensionality. The models that we will investigate are mostly adaptive and learned through systematic explorations of the sensorimotor space[1]. All models play important roles within the overall architecture (see Section 1.3).

In **Chapter 2** we are going to introduce an adaptive model that captures the forward direction of the oculomotor system (visual forward model). The model is based on a statistical learning approach that captures the optical flow of points within the field of view under certain eye movements. The learned flow fields are interpolated using feed-forward neural networks with radial basis functions (RBFs). In order to predict an image for a given eye movement (motor command), the RBF network must be evaluated at every pixel position which is computationally demanding. For this reason, we will present an efficient evaluation scheme. The aim of this chapter is two-fold: we will test the influence of different basis functions onto the quality of the prediction, and furthermore investigate efficient ways of evaluating radial basis functions for image warping.

In **Chapter 3** we will step back from the adaptive model of Chapter 1 and instead develop a geometric model of the oculomotor system. This geometric model comprises significantly fewer parameters than the fully adaptive model and is thus

[1]We use systematic explorations rather than random (i.e. babbling) strategies mainly due to time considerations.

computationally very simple. We furthermore show that the geometric model can also be used for oculomotor control as well. In this context, we augment the basic geometric model with an adaptive correction network that is inspired by neural mechanism of saccade control. Visual prediction and eye control are key elements within the overall model in which they are used to virtually fixate grasp points (see Section 1.3).

Chapter 4 deals with the learning of associations between kinesthetic and sensory states. We introduce a computational model that allows for the associative learning between low-dimensional inputs and high-dimensional outputs (images). The model is a complete implementation of model A, see Section 1.3.2. The complex task of associating a set of postural variables and a given viewing direction to the corresponding visual appearance of the gripper is tackled by a sequential application of different steps. The model consists of four sub-models: (i) a model that relates kinematic postures to viewing directions (that fixate the gripper), (ii) a model that relates postures to low-dimensional descriptions of the gripper's appearance, (iii) a model that generates a fixated visual view of the gripper, based on the low-dimensional description, and (iv) a visual forward model that transforms the fixated view of the gripper to an arbitrary viewing direction.

In **Chapter 5** we employ the visual prediction mechanism in the context of stereo matching, i.e. the problem of finding correspondences in a pair of stereo images. The method is compared to a descriptor-based approach that has become common in computational vision. We show that our approach is able to deal with severely distorted images which pose a great problem to descriptor-based approaches. In the context of simulated object interactions, stereo matching is important for the extraction of depth information from views of the target object. This depth information is in turn necessary for guiding the simulated robot arm towards specific grasp points during the internal simulation process (see Section 1.3).

Finally, in **Chapter 6**, we relate the results from the different chapters and give an outlook onto future research directions.

2. Visual Prediction by Using RBF Networks

2.1. Introduction

In this chapter, we address the problem of predicting visual changes in camera images as the consequence of camera movements as they occur, for example, when a camera is mounted on a mobile robot or a pan-tilt unit. Visual prediction is here defined as the task of predicting the optic flow induced by camera movements and warping the current input image according to the optic flow field. The movement of the camera needs to be defined by a set of movement parameters.

Visual prediction can be applied to a wide variety of robot vision applications. Visual prediction can for example facilitate covert visual attention shifts (Schenck et al., 2011), i.e. the fixation of objects without the actual execution of eye movements. It has also been used for the perception of spatial arrangements by simulated movements toward the obstacles and subsequent evaluation of the predicted sensory states (Hoffmann, 2007a; Möller and Schenck, 2008). Generally, visual prediction by adaptive learning algorithms is a hard task because the high dimensional nature of visual data (i.e. images) which furthermore necessitates a high number of learning examples that might be—depending on the task at hand—acquired at a high cost. Furthermore, the prediction itself may become computationally costly due to the high complexity of the mapping function. In this chapter, we address both, the learning process in brief and strategies to speed up the prediction in depth.

For the rest of this chapter, we focus on the case where a camera is mounted on a two-axis gimbal or pan-tilt unit (PTU), allowing for horizontal (pan) and vertical (tilt) movements, respectively. Under certain conditions, the change of the camera image depends only on the relative changes of the pan and tilt angles, but not on their current values. First, we propose an adaptive algorithm that learns the relationship between the relative pan and tilt movements and the changes of the camera image (in terms of a sparse optical flow field). We then employ local interpolation techniques in order to predict changes of the whole image for arbitrary angle combinations. We refer to this process as *visual prediction*. We will show that our method works with an uncalibrated camera–PTU assembly, and that it is even able to cope with substantial image distortion.

The proposed method is based on image warping by mapping pixels from the output to pixels inside the input image (reverse mapping). The mapping is performed by a network of radial basis functions (RBFs). We will see in the following that the choice of basis function—or kernel—has significant impact on the quality of the mapping. The basis functions considered in this chapter include the widely used Gaussian kernel (Girosi et al., 1995), the thin plate spline kernel (Duchon, 1976), and some less common kernels found in the machine learning literature (Boughorbel et al., 2005). Furthermore, we introduce a two-stage warping scheme which is based on two separate networks for prediction and image interpolation, respectively, by which we decrease the computational complexity.

Schenck and Möller (2007) approached the problem of visual prediction by a warping via a generalized radial basis functions (GRBF) network of Gaussian units (Moody and Darken, 1989; Girosi et al., 1995). GRBF networks consist of a number of units that is smaller than the number of data points used for training. Such networks can reduce the computational effort drastically, albeit imposing a large degree of regularization (smoothing) on the function approximation. Thus, the interpolation becomes inexact. Furthermore, the number of basis functions has to be chosen carefully in order to assure a small approximation error. "Full" RBF networks, i.e. networks consisting of as many units as there are data points, perfectly approximate the data, but may have poor generalization abilities and a higher computational cost. We demonstrate that these shortcomings can be circumvented by using a two-staged approach that (i) imposes a certain degree of regularization and thus improves the generalization, and (ii) reduces the computational cost if the network has to be evaluated at many points at once (as it is the case for image warping).

RBF networks have been previously applied for image warping in different contexts including morphometrics (Bookstein, 1989), where they are used for modelling biological shape changes, facial expressions (Arad et al., 1994), and image registration (Flusser, 1992), just to name a few. The application of RBF networks for image warping was accompanied by the development of efficient and numerically stable strategies for the adaptation (Dyn et al., 1986; Moody and Darken, 1989; Beatson et al., 1999) and the evaluation (Barrodale et al., 1993; Beatson and Newsam, 1992; Beatson and Light, 1997) of these networks. We will not go into details about these methods and develop a novel efficient evaluation strategy instead.

This chapter is organized as follows: in the following section, the camera setup is introduced, Section 2.3 recapitulates the basic RBF theory upon which we base our approach. In Section 2.4 we introduce the so-called *visual forward model*, the procedure for data acquisition and the different types of implementation. Section 2.5 contains the results of the comparison between the different implementations of the visual forward model, and Section 2.6 concludes this chapter.

2.2. Methods

2.2.1. Setup

Throughout this chapter, we assume that the camera is mounted on a two-axis gimbal, so that it pans by an angle φ and tilts by an angle θ. Furthermore, the camera is assumed to be mounted such that the pan and tilt axes intersect at the

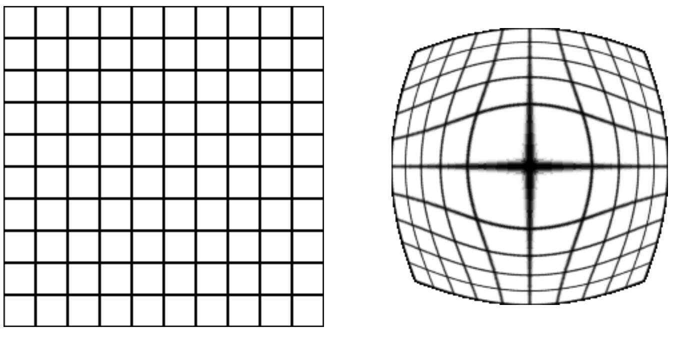

(a) Input grid (b) Warped grid

Figure 2.1.: Effect of the (simulated) radial distortion used in our experiments. A 10×10 regular grid (a) is warped according to (2.1), the result (b) shows the typical "fovea"-effect.

entrance pupil. In this case, the change of the visual information depends only on the relative changes of the pan and tilt angles, denoted by $\Delta\varphi$ and $\Delta\theta$, but not on the current (absolute) angles, φ and θ.

For our tests, we used a camera mounted on a pan-tilt unit (PTU) with two degrees of freedom. This PTU-camera setup deviates slightly from the ideal gimbal-camera setup mentioned above, but it can be shown that the current pan and tilt angles have only a negligible effect (Schenck, 2008).

The camera records color images of 320×240 pixels which were cropped to a size of $\tilde{w}_{\text{in}} = 240 \times \tilde{h}_{\text{in}} = 240$ pixels around the center. Further, the images were warped according to a radial distortion model (see next subsection), thus introducing a high degree of non-linear distortion. After the distortion model has been applied, the images are of size $w_{\text{in}} = 207 \times h_{\text{in}} = 207$ pixels. These images are used for the visual prediction. Note that our approach is purely adaptive, so neither the intrinsic parameters of the camera (including the distortion model) nor the exact kinematics of the PTU need to be known.

2.2.2. Image Distortion Model

The camera images were warped in order to simulate the effect of retinal distortion which is caused by the inhomogeneous distribution of photoreceptors on the human retina. The warping uses a radial mapping of the following form (in polar coordinates):

$$\begin{aligned} \phi_{\text{in}} &= \phi_{\text{out}}, \\ r_{\text{in}} &= \lambda r_{\text{out}}^{\gamma} + (1-\lambda) r_{\text{out}}, \end{aligned} \tag{2.1}$$

where $(\phi_{\text{in}}, r_{\text{in}})$ denotes a point in the (unwarped) input image and $(\phi_{\text{out}}, r_{\text{out}})$ denotes a point in the output image. Note that r_{out} and r_{in} are assumed to lie in the interval $[0;1]$ which can be achieved by normalizing the coordinates. The parameters are $\lambda = 0.8$ and $\gamma = 2.5$. This distortion leads to a "fovea" effect as known from the human visual system: the resolution is much higher in the central region of the image (leading to a large magnification) than in its periphery. Figure 2.1 shows the effect of the radial mapping (2.1) on a regular grid. Note the difference in size between the two images. The region around the warped image which does not contain any information from the input image is filled with black pixels in the actual implementation.

2.2.3. Image Warping

Image warping or geometric transformation is the process of mapping pixels from a source image, denoted by $I_{\text{in}}(x, y)$, onto a destination image, denoted by $I_{\text{out}}(u, v)$ (Wolberg, 1994; Gonzalez and Woods, 2001). The mapping is established via the so-called warping function, denoted by its two components $u = g_u(x, y)$, $v = g_v(x, y)$ (forward mapping) or analogously $x = f_x(u, v)$, $y = f_y(u, v)$ (reverse mapping) (Wolberg, 1994). If the warping function is invertible, both mapping directions are equivalent. In the following, we will assume that the warping is performed by mapping points from the destination (output) image to points in the source (input) image (reverse mapping). In the reverse mapping scheme, the warping function is applied to every destination pixel in the output image to calculate the position of the corresponding source pixel in the input image.

The warping function is typically smooth and continuous, whereas digital images are discrete in their nature. Therefore, applying an interpolation technique is inevitable (Gonzalez and Woods, 2001). Furthermore, in the case of forward mapping, the destination image might suffer from "holes", i.e. regions which were not assigned with a value by the warping function due to quantization effects. These holes require special treatment. This is not the case for reverse mapping which guarantees that a value is assigned to each pixel in the destination image. For our experiments we therefore use reverse mapping with bilinear interpolation (Gonzalez and Woods, 2001).

2.3. Radial Basis Functions

The problem of scattered data interpolation can be stated as the problem of finding a function that passes through a set of reference points and establishes a (smooth) interpolation between these points. Formally, for a set of N data points $\{\vec{x}_1, \ldots, \vec{x}_N | \vec{x}_i \in \mathbb{R}^d\}$ and corresponding function values $\{y_1, \ldots, y_N | y_i \in \mathbb{R}\}$, we seek a function $\hat{f}$ such that

$$\hat{f}(\vec{x}_i) = y_i \qquad \text{for } i = 1, \ldots, N. \tag{2.2}$$

If we furthermore impose smoothness constraints on $\hat{f}$, i.e. in terms of a smoothness functional, the functions minimizing the functional have the form (Girosi et al., 1995)

$$\hat{f}(\vec{x}) = \sum_{i=1}^{N} w_i \kappa(\vec{x}, \vec{x}_i) + \underbrace{\sum_{l=0}^{n} a_l \psi_l(\vec{x})}_{p(\vec{x})}, \tag{2.3}$$

where $\kappa(\cdot,\cdot)$ denotes a so-called radial basis function (RBF) or kernel, $\psi_l(\cdot)$ is a set of n functions, typically polynomials; hence we refer to $p(\vec{x})$ as the polynomial term. Note that this polynomial term is necessary, depending on the choice of RBF, to assure that property (2.2) holds (Girosi et al., 1995). We will only consider kernels that require polynomial terms of degrees 0 ($p(\vec{x}) = a_0$) or 1 ($p(\vec{x}) = a_0 + \sum_{l=1}^{d} a_l (\vec{x})_l$). Here, the operator $(\cdot)_l$ returns the l-th element of its vector-valued argument.

The coefficients w_i and a_j are data-dependent and can be found by solving the linear system (Girosi et al., 1995):

$$\begin{aligned} (K + \lambda I)\vec{w} + \Psi^\top \vec{a} &= \vec{y} \\ \Psi \vec{w} &= \vec{0}, \end{aligned} \tag{2.4}$$

where $(K)_{ij} = \kappa(\vec{x}_i, \vec{x}_j)$, $(\Psi)_{li} = \psi_l(\vec{x}_i)$, $\vec{w} = [w_1, \ldots, w_N]^\top$, $\vec{a} = [a_0, \ldots, a_n]^\top$, $\vec{y} = [y_1, \ldots, y_N]^\top$, and $\lambda \in \mathbb{R}$ is a constant. The matrix K is called kernel matrix and must not be singular. The constant λ acts as a regularization parameter whose choice controls the degree of smoothness of $\hat{f}$. Note that property (2.2) does not hold for $\lambda \neq 0$.

Solving (2.4) iteratively (e.g. by conjugate gradient methods) can become infeasible even for a medium number of RBFs (~ 150) because the system becomes ill-conditioned for most RBFs (Dyn et al., 1986), which leads to a low rate of convergence of the iterative method. Therefore, the system needs to be preconditioned (i.e. the system's coefficient matrix is pre-multiplied with an approximation of its inverse in order to improve the conditioning of the system) in an appropriate manner. Dyn et al. (1986) suggest numerical procedures for solving (2.4) using a conjugate gradient approach.

For our experiments, we used a direct approach for solving (2.4) based on the Bunch-Kaufman algorithm (Bunch and Kaufman, 1977) which is closely related to the $LDL^\top$ factorization of a matrix (Golub and Van Loan, 1996). The $LDL^\top$ factorization has the advantage that it exploits the symmetry of the matrix but does not require positive definiteness. We observe that we can rewrite (2.4) as

$$\underbrace{\begin{pmatrix} K + \lambda I & \Psi \\ \Psi^\top & O \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} \vec{w} \\ \vec{a} \end{pmatrix}}_{\vec{z}} = \underbrace{\begin{pmatrix} \vec{y} \\ \vec{0} \end{pmatrix}}_{\vec{b}}, \tag{2.5}$$

where all symbols are defined as in (2.4), O denotes a zero matrix of size $n \times n$ and $\vec{0}$ is the null vector of length n. This system can then be solved for $\vec{z}$ using the above-mentioned method in order to obtain the RBF coefficients.

2.3.1. Generalized Radial Basis Function Networks

The above formulation of networks of radial basis functions assumes that the number of basis functions equals the number of data points N. Such networks make especially sense in applications where the data is gridded (which is the case for our application). If the data is scattered and noisy, choosing a large number of RBFs eventually leads to overfitting—i.e. the network adapts to the noise and consequently does not generalize well (Bishop, 1995). Explicit regularization is one possibility to circumvent overfitting (see previous section).

However, a more efficient solution is to employ less RBFs than data points which leads to an implicit form of regularization which we will briefly review in the following. Let $k < N$ be the number of basis function, then the network function is of the form (Girosi et al., 1995; Bishop, 1995; Moody and Darken, 1989):

$$\hat{f}(\vec{x}) = \sum_{i=1}^{k} w_i \kappa(\vec{x}, \vec{c}_i) + p(\vec{x}), \tag{2.6}$$

where all symbols are defined as in (2.3), except from $\vec{c}_i$ which denotes the i-th RBF center. We will refer to this type of network as generalized RBF (GRBF) network. Here, the number of RBFs controls the degree of regularization; the more RBFs are used the less smooth $\hat{f}$ becomes.

The training of such a network is a two-step procedure: in the first step the RBF centers are determined in an unsupervised manner (e.g. by running the k-means algorithm multiple times and choosing the best set of centers based on some error criterion (Moody and Darken, 1989)); the second step is analogous to the standard RBF network and involves solving a linear system. As the system in the case $k < N$ is overdetermined (i.e. there are more data points than RBFs), we need to employ least-squares techniques such as the Moore-Penrose pseudo-inverse (Girosi et al., 1995; Bishop, 1995) to calculate the weights. Note that there exist alternative, gradient-based supervised learning techniques for determining the weights (Moody and Darken, 1989; Bishop, 1995) which we will not address here.

A convenient way to calculate the pseudo-inverse is to use a matrix factorization (such as the Cholesky decomposition (Horn and Johnson, 1990)) for solving the system

$$(A^\top A + \lambda I)\vec{z} = A^\top \vec{b}, \tag{2.7}$$

where A denotes the coefficient matrix, $\vec{z}$ the vector of unknowns, $\vec{b}$ denotes the constants, and $\lambda \geq 0$ is an (optional) regularization parameter. Note that the matrix $A^\top A + \lambda I$ is guaranteed to be positive definite if A has full column rank. Furthermore, the additional regularization parameter makes the network less sensitive to k, the choice of the number of RBFs. It is important to note that using the Cholesky decomposition for solving (2.7) can be numerically inaccurate because we need to form $A^\top A$ explicitly (which is not the case if the QR decomposition (Horn and Johnson, 1990) is used), but it has the great advantage that the regularization

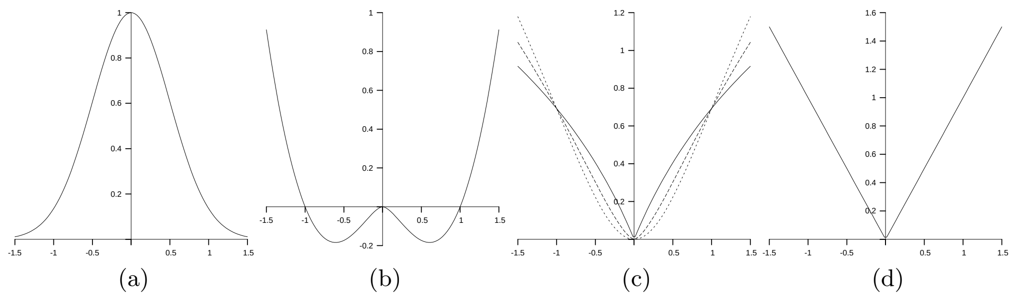


Figure 2.2.: 1D profiles of different kernel functions: Gaussian ($\sigma^2 = 0.25$) (a), thin plate spline ($k = 2$) (b), log-kernel ($\beta = 1, 1.5, 2$) (c), and Euclidean ($\beta = 1$) (d).

matrix λI can be easily incorporated by computing $A^\top A + \lambda I = RR^\top$, where R denotes the Cholesky factor.

In a previous article, Schenck and Möller (2007) applied a GRBF network to the problem of visual prediction. However, we will see that the data produced by their statistical learning approach (which serves as the training set for the network), is in fact gridded and therefore more accurately represented by a standard RBF network.

2.3.2. Positive Definite and Conditionally Definite Kernels

The type of kernel function has a substantial influence on the interpolation $\hat{f}$. Furthermore, the choice of kernel determines whether the system (2.4) has a unique solution at all (Driscoll and Fornberg, 2002).

Positive Definite Kernels

The system is guaranteed to have a unique solution if the kernel function (and consequently the corresponding kernel matrix) is positive definite, i.e.

$$\vec{w}^\top K \vec{w} > 0 \qquad \forall\, \vec{w} \in \mathbb{R}^N \setminus \{\vec{0}\}$$
$$\Leftrightarrow$$
$$\sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \kappa(\vec{x}_i, \vec{x}_j) > 0 \qquad \forall\, w_i, w_j \in \mathbb{R} \setminus \{0\}.$$

In such cases, the system reduces to (2.4, upper equation) (Micchelli, 1986).

The Gaussian kernel (cf. Figure 2.2a)

$$\kappa(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{1}{2}\frac{\|\vec{x}_i - \vec{x}_j\|^2}{\sigma^2}\right),$$

with width parameter σ, is an example of a positive definite kernel. The width parameter controls the smoothness of the approximation and depends on the scaling of the data. Note that, if σ approaches infinity, the system (2.4) becomes ill-conditioned (Driscoll and Fornberg, 2002). Furthermore, σ depends on the spacing of the data points and has to be chosen for each data set appropriately. We do not go into details of how to chose σ in a supervised manner, but refer to the corresponding literature (Bishop, 1995; Moody and Darken, 1989).

A common generalization of the Gaussian kernel, which is strictly speaking not an RBF (because it is not a function of the Euclidean distance and is thus generally anisotropic), is the kernel function (Bishop, 1995)

$$\kappa(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{1}{2}(\vec{x}_i - \vec{x}_j)^\top \Sigma^{-1} (\vec{x}_i - \vec{x}_j)\right), \tag{2.8}$$

where $\Sigma \in \mathbb{R}^{d \times d}$ is a symmetrical and positive definite matrix whose eigenvalues / eigenvectors determine the extension / orientation of the Gaussian surface (Bishop, 1995). This kernel is especially suited for non-gridded data which forms non-uniform clusters, or data which are scaled differently in each dimension. Note that (2.8) is closely related to the probability density function (pdf) of the multivariate normal distribution with mean vector $\vec{\mu} = \vec{x}_j$ and covariance matrix Σ.

Conditionally Definite Kernels

In the following, we will consider an extended class of kernels, namely the conditionally definite kernels. A kernel is called *conditionally (positive) definite*, if it fulfills the following conditions (Dyn et al., 1986; Micchelli, 1986; Girosi et al., 1995):

$$\sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \kappa(\vec{x}_i, \vec{x}_j) \geq 0 \qquad \text{with}$$
$$\sum_{i=1}^{N} w_i p(\vec{x}_i) = 0, \tag{2.9}$$

where $p(\cdot)$ is a polynomial term as defined in (2.3). The second condition (2.9) is enforced by the linear system through (2.4, lower equation). Although this is a weaker form of positive definiteness, it is still guaranteed that the linear system has a unique solution (Micchelli, 1986).

Let $r_{ij} = \|\vec{x}_i - \vec{x}_j\|$. The following kernel functions can be shown to be conditionally definite:

- The *polyharmonic* or *thin plate spline (TPS) kernel* (cf. Figure 2.2b) (Duchon, 1976; Bookstein, 1989):

$$\kappa_{\text{tps}}(\vec{x}_i, \vec{x}_j) = \begin{cases} r_{ij}^k \log r_{ij} & k \text{ even} \\ r_{ij}^k & \text{otherwise,} \end{cases}$$

  where $k \in \mathbb{N}$. For $k = 2$, we obtain the well-known TPS kernel $r^2 \log r$ (Bookstein, 1989). Note that the parameter k determines the degree of the polynomial term (Girosi et al., 1995); for $k = 2, 3$ the polynomial needs to be of degree 1, i.e. a linear (affine) term.

- The *logarithmic* or *log-kernel* (cf. Figure 2.2c) (Boughorbel et al., 2005):

$$\kappa_{\log}(\vec{x}_i, \vec{x}_j) = \log(r_{ij}^{\beta} + 1)$$

- The *power distance* or *Euclidean kernel* (cf. Figure 2.2d) (Boughorbel et al., 2005):

$$\kappa_{\text{pow}}(\vec{x}_i, \vec{x}_j) = r_{ij}^{\beta}$$

  For the special case $\beta = 1$, this kernel leads to a piece-wise linear approximation.

All of these kernel function have in common that they do not rely on a width parameter and are thus independent of the scaling of the data.

We are aware that there exist another widely used class of RBFs, the so called multiquadrics which were introduced by Hardy (Hardy, 1971; Micchelli, 1986). These RBFs share the property of Gaussian RBFs in that they also include a width parameter which depends on the scaling of the data. As we are mainly interested in parameter-less RBFs, we did not include multiquadrics in our tests.

2.3.3. Dual Representation

Besides their representation as a weighted sum of basis functions of the current input $\vec{x}$, in which the coefficients (weights) depend on the target outputs $\vec{y}$, RBF networks can be interpreted equivalently as a weighted sum of the target outputs y_i in which the coefficient (weights) depend on the current input (Girosi et al., 1995).

Let us consider a simplified variant of (2.3) which lacks the polynomial term, i.e. $p(\vec{x}) = 0$ and let us furthermore define $\vec{k}(\vec{x}) = [\kappa(\vec{x}, \vec{x}_1), \ldots, \kappa(\vec{x}, \vec{x}_N)]^\top$, and $\vec{y} = [y_1, \ldots, y_N]^\top$, then we can rewrite (2.3) as

$$\hat{f}(\vec{x}) = \vec{w}^\top \vec{k}(\vec{x}) = ((K + \lambda I)^{-1} \vec{y})^\top \vec{k}(\vec{x}), \tag{2.10}$$

where we used explicit matrix inversion to solve the system (2.4) for $\vec{w}$.

By noting that K is symmetric, and by using the identity $((K + \lambda I)^{-1}\vec{y})^\top = \vec{y}^\top (K + \lambda I)^{-1}$, we can define $\vec{d}(\vec{x}) = (K + \lambda I)^{-1} \vec{k}(\vec{x})$ in order to yield (Girosi et al., 1995)

$$\hat{f}(\vec{x}) = \vec{y}^\top \vec{d}(\vec{x}) = \sum_{i=1}^{N} y_i d_i(\vec{x}), \tag{2.11}$$

which is called the dual representation of (2.10). Note that this result can be easily extended for RBF networks with an additional polynomial term.

The dual representation will be used to derive the two-staged mapping model (see Section 2.4.3) that allows for a faster image warping compared to the other approaches reviewed in this chapter.

2.4. Visual Forward Model

In this section we will fuse the results of the former sections in order to derive a framework for visual prediction, termed "visual forward model". The term forward model stems from the field of motor control and will be used here to describe a predictor which generates a new sensory state based on a current sensory state and a motor command (Karniel, 2002; Schenck and Möller, 2007; Schenck, 2008). Thus, a forward model models the characteristics of a certain plant. Note that this interpretation is one among many others (Karniel, 2002) which is closely related to the state observer in control theory. In our case, the plant is the camera-PTU setup. The current sensory state is the camera image and the motor command is described by $\Delta\varphi, \Delta\theta$, the changes in the setup's pan and tilt angles.

The internal structure of the forward model is two-fold: a so-called mapping model performs the actual prediction by warping the input image, based on $\Delta\varphi, \Delta\theta$; the so-called validator model indicates if a pixel can be faithfully predicted. Pixels that are not present in the input image, but appear in the output image if the motor command would be executed, are classified invalid. This structure is retained from the original implementation by Schenck and Möller (2007).

Formally, the mapping model is defined by a pair of functions $\hat{f}_x(u,\ v,\ \Delta\varphi,\ \Delta\theta)$, $\hat{f}_y(u,\ v,\ \Delta\varphi,\ \Delta\theta)$ which map each pixel of the output image $I_{\text{out}}(u,v)$ onto a pixel in the input image $I_{\text{in}}(x,y)$. The validator model is a discriminator function $\hat{f}_\mathcal{V}(u,v,\Delta\varphi,\Delta\theta) \in \{0,1\}$ which returns 1 in the case that $I_{\text{out}}(u,v)$ is valid (given a certain motor command $(\Delta\varphi,\Delta\theta)$) and 0 otherwise. Therefore, the mapping model can be seen as a regression problem while the validator model performs a binary classification task.

In the following, we will sketch the procedures used for data acquisition as well as two different ways to implement the mapping model. Both methods are based on the combination of warping of the reverse mapping type with RBF networks as warping functions.

2.4.1. Data Acquisition

In order to establish a mapping between all pixel positions before the execution of a motor commando and thereafter, we would in general need to compute a (dense) visual flow field between the images taken before and after the execution of a motor command. Computing such flow fields based on two images is rather delicate and subjected to errors due to mismatches. For this reason, we chose a statistical learning approach which uses a large number of before-after pairs (Schenck and Möller, 2007; Schenck, 2008).

For the learning process, we define a $N_u \times N_v \times N_{\Delta\varphi} \times N_{\Delta\theta}$ regular grid $\mathcal{G}$ in the joint motor-pixel position space. Each grid point, given by the tuple $(u_i,\ v_j,\ \Delta\varphi_k,\ \Delta\theta_l)$, is associated with a so-called cumulator unit C_{ijkl} which corresponds to the input image and is of size $w_{\text{in}} \times h_{\text{in}}$. For every grid position $(i,j,k,l) \in \mathcal{G}$, each (x,y)-position within the corresponding cumulator unit C_{ijkl} is incremented by one if the absolute difference between the pixel intensity at the position (u_i, v_j) in the output image and the intensity at position (x, y) in the input image falls beneath a certain threshold ε_c. For our experiments, ε_c was chosen as 3.5% of the intensity range in an individual color channel. This procedure is repeated for all grid positions and for different initial pan-tilt pairs. After the training, all cumulator units that correspond to valid pixels should contain clear maxima and can be used for the training of the mapping model; invalid cumulator units are only used for the training of the validator model. A validator unit is invalid if there is no maximum or the maximum is lower than a pre-defined threshold.

Note that a similar algorithm can be found in the literature in the context of uncalibrated stereo matching, where it was used to recover the epipolar geometry between two cameras (Wexler et al., 2003).

The above algorithm results in two training sets: the set which contains samples for the mapping model, which is given by

$$\mathcal{T}_m = \{(u_i, v_j, \Delta\varphi_k, \Delta\theta_l, x_{ijkl}, y_{ijkl}) | \, (i,j,k,l) \in \mathcal{G}, \nu_{ijkl} = 1\},$$

where ν_{ijkl} is either 1 or 0, depending on the validity of the corresponding cumulator unit C_{ijkl}, (x_{ijkl}, y_{ijkl}) is the position of the maximum inside C_{ijkl}, and the set which contains samples for the validator model

$$\mathcal{T}_\nu = \{(u_i, v_j, \Delta\varphi_k, \Delta\theta_l, \nu_{ijkl}) \,|\, (i,j,k,l) \in \mathcal{G}\},$$

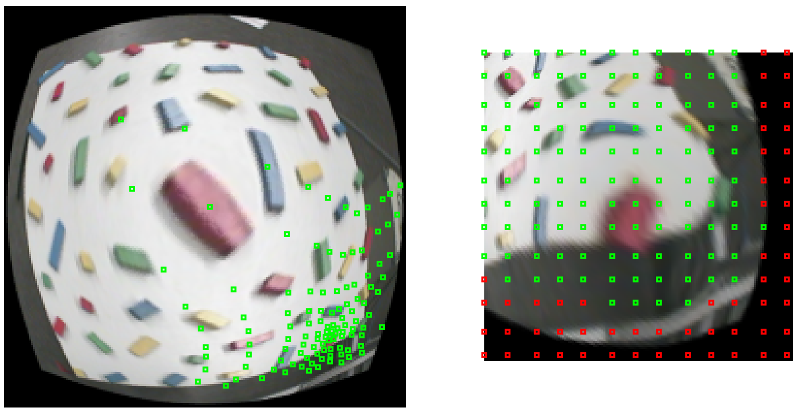
where, as defined above, $\nu_{ijkl} \in \{0, 1\}$ indicates the validity of cumulator unit C_{ijkl}.

For our experiments, we used a $13 \times 13 \times 11 \times 11$-grid of cumulator units. Thus, the training set for the validator model consisted of $20,449$ samples. The training set for the mapping model consisted of $10,708$ samples which corresponds to the number of valid cumulator units.

2.4.2. Full Mapping Model

In the following, the "full" mapping model corresponds to the case in which every grid pixel position in the output image is propagated through the mapping function. The visual forward model which was originally proposed by Schenck and Möller (2007) falls into this category. There, the mapping model was realized by a GRBF network with extended Gaussian kernel functions.

In GRBF networks the number of RBFs is smaller than the number of data points. The importance of this property becomes clear if we consider the problem of scattered data: in this case it is important to impose a certain degree of regularization on the network in order to circumvent overfitting. In our present application, where the data is gridded, we expect that a GRBF network performs worse than a full standard RBF network, because in the former case certain grid points share a common RBF center. The GRBF network employed by Schenck and Möller (2007), for example, consisted of 1500 RBFs—whereas the data set consisted of about $11,000$ data points.



(a) Input image (b) Output image

Figure 2.3.: Example of the two-staged mapping model. Figure 2.3a shows the input image, Figure 2.3b shows the corresponding (warped) output image. Valid grid points $\mathcal{P}_\nu$ are depicted as green squares in Figure 2.3b, red squares symbolize invalid grid points $\mathcal{P}_g \setminus \mathcal{P}_\nu$. Warped valid grid points $\mathcal{P}_w$ are depicted as green squares in Figure 2.3a.

In this chapter, we will focus on "full" standard RBF networks, where each data point from the training set is represented by one RBF. In this case, the associated computational cost, which is consequently very high, becomes an important aspect. Consider the case where the output image is 159×159 pixels in size. Now consider a network with $\sim 11,000$ RBFs—a realistic number for visual prediction tasks (see Section 2.4.1). Performing a single prediction would then require the evaluation of these $\sim 11,000$ RBFs at $25,281$ test points (i.e. pixel positions), which takes, depending on the implementation, up to a minute on current PC hardware. Because of this, we performed all tests using a GPU based implementation of the RBF network evaluation / image warping which reduced the average execution time for the above example to fractions of a second[1].

2.4.3. Two-Staged Mapping Model

The high computational cost of the "full" approach, along with the possibility to view RBF networks in the dual representation, led to the idea of the so-called two-staged mapping model. In this model, the warping is split into a prediction network and an interpolation network. The prediction network (first stage) transforms only a small subset of the total pixel positions in the output image, while the interpolation network (second stage) performs the actual image warping which does not depend on $\Delta\varphi, \Delta\theta$ but on the output of the first stage.

[1] I am very grateful to Wolfram Schenck for providing the GPU implementation of the visual forward model.

This is a computationally more efficient strategy because only the evaluation of the prediction network (first stage) involves non-linear functions (that have to be evaluated for each prediction), while the evaluation of the interpolation network (second stage) corresponds to a linear transformation. Formally the first stage corresponds to equation (2.3), and the second stage corresponds to equation (2.11), where $\vec{y}$ is the output of the first stage and $\vec{d}(\vec{x})$ can be pre-computed for each pixel in the output image.

First Stage

Let $\mathcal{P}_g = \{(u_i, v_j) \,|\, i = 1 \ldots N_u, j = 1 \ldots N_v\}$ denote the set of all grid points, and let $\mathcal{P}_\nu = \{(u_i, v_j) \,|\, (u_i, v_j) \in \mathcal{P}_g, \hat{f}_\nu(u_i, v_j, \Delta\varphi, \Delta\theta) = 1\}$ denote the subset of valid points, given a motor command $(\Delta\varphi, \Delta\theta)$. Figure 2.3b shows an example image, overlaid with the grid. Valid grid points are depicted by green squares, invalid ones by red squares. The prediction network only maps the valid positions onto the warped positions:

$$\mathcal{P}_w = \{(x, y) \,|\, x = \hat{f}_x(u_i, v_j, \Delta\varphi, \Delta\theta), y = \hat{f}_y(u_i, v_j, \Delta\varphi, \Delta\theta), (u_i, v_j) \in \mathcal{P}_\nu\}.$$

Figure 2.3a shows an example input image, overlaid with the warped points, depicted as green squares. Note that the unwarped (input) image contains the warped grid points and the warped (output) image contains the unwarped (regularly spaced) grid points (which is characteristic for a reverse mapping). Furthermore, the prediction network, which is used to transform the grid points, is in principle identical to the mapping network in the “full” approach. The main difference between the “full” approach and the two-staged approach lies in the fact that in the latter only a few grid points are mapped by the RBF network while in the “full” approach each pixel position is mapped.

Second Stage

The second stage, i.e. the interpolation network, is inspired by point-based image warping techniques which are commonly used in the field of morphometrics (e.g. Bookstein (1989)). In point based image warping, the image is warped such that two sets of control points (or landmarks) are brought into accordance: the first set of control points are (source) locations in the input image, the second set of control points are the desired (destination) locations of these points. The warping function has thus no external parameters—in contrast to the first stage—i.e. it maps 2D points onto 2D points.

A commonly used form of kernel function in the context of point-based image warping is the thin plate spline (TPS) kernel; for a thorough treatment of TPS warping, see e.g. Bookstein (1989). In our application of image warping for visual prediction, the source control points correspond to the outputs of the prediction network $\mathcal{P}_w$ and the destination points correspond to the valid grid points $\mathcal{P}_\nu$. The interpolation network is a standard RBF network that has to be evaluated at every pixel position of the output image. We observe that these positions, i.e. the network input, do not depend on the motor command $(\Delta\varphi, \Delta\theta)$; it is only the source control points which depend thereon. Therefore, we can use the dual representations of RBF networks (2.11) in which the network function is a linear combination of the outputs, i.e. the source control points, while the coefficients can be efficiently pre-computed based on the constant inputs.

The above only holds if $\mathcal{P}_g = \mathcal{P}_\nu$, i.e. the motor command has no effect on the validity of any of the grid points. This assumption only holds for small PTU movements and is generally too strict. From (2.10) we see that the regularized inverse of the kernel matrix and the vector $\vec{k}(\vec{x})$ are the only terms that depend on the grid points. Thus, we have to modify these terms according to the set of valid grid points $\mathcal{P}_\nu$. The vector $\vec{k}(\vec{x})$ can be trivially modified by omitting components that depend on invalid grid points. The inverse kernel matrix must be modified accordingly, e.g. by a rank-o down-date, where $o = |\mathcal{P}_g| - |\mathcal{P}_\nu|$ is the number of invalid grid points (see Appendix A for details). Note that the down-dating procedure leads to numerical inaccuracies and that it only makes sense to down-date the inverse matrix by a maximum of $|\mathcal{P}_g| - |\mathcal{P}_\nu| = |\mathcal{P}_g|/2 - 1$ ranks. Down-dating the inverse by o ranks involves solving a linear system of o equations. Therefore down-dating the inverse only makes sense if $|\mathcal{P}_g| - |\mathcal{P}_\nu| < |\mathcal{P}_g|/2$; in all other cases it is more efficient to recompute the inverse directly. We observed during our experiments that the down-dating procedure is sensitive to the values of $(K)_{ij}$. We therefore scaled the pixel positions to lie inside the range $[-1; 1]$ as recommended by Barrodale et al. (1993).

Matrix Form

For the sake of completeness we shall give a detailed description of the two-staged mapping model in its matrix form which matches the concrete implementation that we used for the experiments. Some questions regarding the runtime as well as which parts can be pre-computed can be understood more easily when looking at the matrix equations. To minimize notational clutter, we will redefine some of the symbols used in section 2.3. The matrix form allows us to evaluate the RBF network not only at a single test point $\vec{x}^* \in \mathbb{R}^2$, but at multiple test points at once.

The first stage consists of an RBF network which is evaluated at the valid grid points $(u_i^*, v_i^*) \in \mathcal{P}_\nu$ augmented with the given motor command $(\Delta\varphi^*, \Delta\theta^*)$. In order to evaluate the RBF network, we must pair each element of $\mathcal{P}_\nu \times \{(\Delta\varphi^*, \Delta\theta^*)\}$ with each element of the input part of the training set $\mathcal{T}_m$ by applying the RBF to each pairing. The result of this paring is what we will call the cross-kernel matrix

$$(K_\nu^\times)_{ij} = \kappa((u_i^*, v_i^*, \Delta\varphi^*, \Delta\theta^*)^\top, (u_j, v_j, \Delta\varphi_j, \Delta\theta_j)^\top) \qquad \begin{array}{l} i = 1, \dots, |\mathcal{P}_\nu|, \\ j = 1, \dots, |\mathcal{T}_m| \end{array} \tag{2.12}$$

which is of size $|\mathcal{P}_\nu| \times |\mathcal{T}_m|$.

The cross-kernel matrix is the basis for the RBF network in matrix form. From (2.3) we see that the output of the RBF network is a weighted linear combination of the RBFs. In matrix form this linear combination corresponds to the matrix product

$$L_w = K_\nu^\times W_w,$$

where $K_\nu^\times$ denotes the cross-kernel matrix (2.12), $W_w \in \mathbb{R}^{|\mathcal{T}_m| \times 2}$ denotes the weight matrix, and $L_w \in \mathbb{R}^{|\mathcal{T}_\nu| \times 2}$ contains the resulting warped grid points (i.e. what would be the elements of $\mathcal{P}_w$). The weight matrix W_w is determined in analogous fashion to (2.5) by solving the linear system

$$K_m W_w = L_m, \tag{2.13}$$

where $K_m \in \mathbb{R}^{|\mathcal{T}_m| \times |\mathcal{T}_m|}$ corresponds to the kernel matrix of the input part of $\mathcal{T}_m$, and $L_m \in \mathbb{R}^{|\mathcal{T}_m| \times 2}$ contains the target outputs of $\mathcal{T}_m$. The elements of the matrices are given by

$$(K_m)_{ij} = \kappa((u_i, v_i, \Delta\varphi_i, \Delta\theta_i)^\top, (u_j, v_j, \Delta\varphi_j, \Delta\theta_j)^\top) \qquad i, j = 1, \ldots, N,$$
$$L_m = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_N & y_N \end{bmatrix},$$

where we defined $N = |\mathcal{T}_m|$.

The second stage interpolates between the valid grid points so that every point in the output image potentially gets a corresponding warped point in the input image. This corresponds to evaluating an RBF network for a fixed input (i.e. the pixel positions) while the target outputs (i.e. the outputs L_w of the first stage) are changing according to the motor command $(\Delta\varphi^*, \Delta\theta^*)$. The dual form of an RBF network (2.11) is used for this purpose. In matrix notation, the second stage can be written as

$$L_I = K_I^\times \underbrace{S_\nu K_\nu^{-1} L_w}_{W_I} \tag{2.14}$$
$$= K_I^\times W_I, \tag{2.15}$$

where $K_I^\times \in \mathbb{R}^{w_{\text{out}} \cdot h_{\text{out}} \times |\mathcal{P}_g|}$ denotes the cross-kernel matrix between all pixel positions in the output image and all grid points, whose components are given by

$$(K_I^\times)_{ij} = \kappa((u_i, v_i)^\top, (u_j, v_j)^\top) \qquad (u_i, v_i) \in \{1, \ldots, w_{\text{out}}\} \times \{1, \ldots, h_{\text{out}}\}$$
$$(u_j, v_j) \in \mathcal{P}_g.$$

Furthermore, $S_\nu \in \mathbb{R}^{|\mathcal{P}_g| \times |\mathcal{P}_\nu|}$ denotes a selector matrix (i.e. an incomplete identity matrix) whose $|\mathcal{P}_\nu|$ columns are unit vectors of dimension $|\mathcal{P}_g|$ which correspond to the valid grid points. Multiplying $K_\nu^{-1} L_w$ (from the left) with S_ν has the effect that the rows of $W_I \in \mathbb{R}^{|\mathcal{P}_g| \times 2}$ that correspond to invalid grid points are set to zero. $K_\nu^{-1} \in \mathbb{R}^{|\mathcal{P}_\nu| \times |\mathcal{P}_\nu|}$ is the inverse of the kernel matrix of the valid grid points

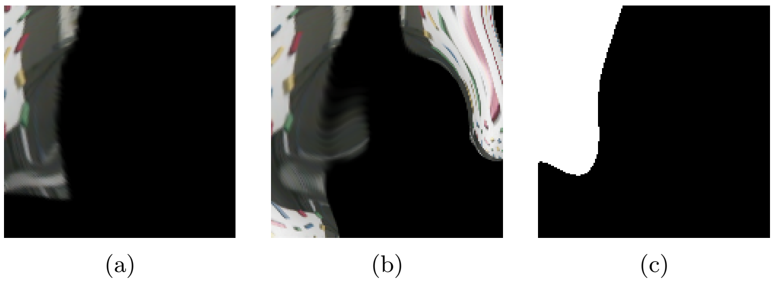


Figure 2.4.: Extrapolation behavior of two different mapping models: (a) RBF network using the log-kernel, (b) generalized RBF network using an anisotropic Gaussian kernel. The latter requires a validator model (c) in order to mask the artifacts due to erroneous extrapolation.

which is calculated from the inverse kernel matrix of all grid points K_g^{-1} using a down-dating procedure (see Appendix A); the elements of K are given by $(K_g)_{ij} = \kappa((u_i, v_i)^\top, (u_j, v_j)^\top)$ with $(u_i, v_i), (u_j, v_j) \in \mathcal{P}_g$. The evaluation of the second stage results in a matrix $L_I \in \mathbb{R}^{w_{\text{out}} \cdot h_{\text{out}} \times 2}$ which contains all warped (i.e. input) pixel positions for each pixel in the output image.

From these equations, it becomes clear that most of the values, i.e. those that do not depend explicitly on the current $(\Delta\varphi^*, \Delta\theta^*)$, can be pre-computed. The matrices that do not depend on the numerical values of the motor command but only depend on the validity of the grid points can be either adjusted by omitting those components that correspond to the invalid grid points (in case of S_ν) or by down-dating strategies (in case of K_ν^{-1}). Furthermore, by introducing the selector matrix S_ν, the "big" cross-kernel matrix $K_I^\times$ need not be modified. This is crucial because even for output images of moderate size, this matrix can become very large: for our experiments we used images of size 159×159 and a grid of size 13×13, thus $K_I^\times$ is of size $159 \cdot 159 \times 13 \cdot 13$. Albeit its size (which poses no problem to current computer hardware) pre-computing $K_I^\times$ circumvents any evaluation of nonlinear functions during the second stage. The whole image warping boils down to a matrix–matrix multiplication (2.15)—or merely two matrix–vector multiplications as W_I has only 2 columns.

2.4.4. Validator Model

In general, the validator model is conceptually simpler than the mapping model because it only has to perform a binary classification task. The validity of every pixel in the output image needs to be checked. The validator model is used to mask regions in the predicted (output) image which contain false information due to erroneous extrapolation of the GRBF network. Figure 2.4b shows the complete (unmasked) output of the visual forward model using the GRBF network; Figure 2.4c shows the corresponding mask generated by the validator model from Schenck and Möller (2007).

In their original publication, Schenck and Möller (2007) proposed an implementation of the validator model by a GRBF network with a large number of RBFs (i.e. 1500). In conjunction with the two-staged mapping model, we used a simpler 3-layer feed-forward network with a hidden layer of only 48 sigmoid units and one sigmoid output unit (Bishop, 1995) (which reflects the binary character of this classification task). The network was trained on the set of the valid–invalid samples $\mathcal{T}_\nu$ using the back-propagation algorithm (Rumelhart et al., 1986). Note, that a small hidden layer imposes a strong degree of regularization on the network, but makes the validator model faster to evaluate than the GRBF network by Schenck and Möller (2007) or a full RBF network. Figure 2.3b shows the influence of the validator model: invalid grid points are depicted as red squares, those points are not used for the warping and have no corresponding counterpart in 2.3a.

Furthermore, we will introduce a heuristic validator model which is used in combination with the full mapping model using RBF networks. The heuristic treats those points as invalid which are mapped to points outside a certain (valid) range, i.e.

$$\hat{f}_\nu(u, v, \Delta\varphi, \Delta\theta) = \begin{cases} 0 \,\text{if}\, (\hat{x}, \hat{y}) \notin [1; w_{\text{in}}] \times [1; h_{\text{in}}] \vee I_{\text{in}}(\hat{x}, \hat{y}) \notin [0; 1] \\ 1 \,\text{else}, \end{cases} \tag{2.16}$$

where $\hat{x} = \hat{f}_x(u, v, \Delta\varphi, \Delta\theta)$, $\hat{y} = \hat{f}_y(u, v, \Delta\varphi, \Delta\theta)$, and w_{in}, h_{in} denote the width and height of the input image. The first condition states that pixel positions which are mapped outside the bounds of the input image should be regarded as invalid; the second condition states that pixels whose values exceed a certain range (in our case $[0; 1]$) should be regarded as invalid. The second condition is important when working with retinal images which might already contain invalid pixels (i.e. the black margin around the actual image, see e.g. Figure 2.3a).

One should note that this heuristic can only produce reasonable results if the mapping function does not map points outside its valid domain onto points that fall into the valid range. Figure 2.4a shows an example that was produced by using a log-kernel network; black regions around the border indicate invalid pixels. Our experiments showed that the log-kernel and the Euclidean kernel are the only RBFs that can be used in this context. All other kernels we have tested showed an erroneous extrapolation behavior, and lead to images akin to that shown in Figure 2.4b. Therefore, a dedicated classifier-based validator model is inevitable. So far, we did not apply any techniques for improving the behavior of RBFs towards the edge of the domain as e.g. described by Fornberg et al. (2002). In the following we will term the heuristic validator model associated with a certain kernel function its "native" validator model.

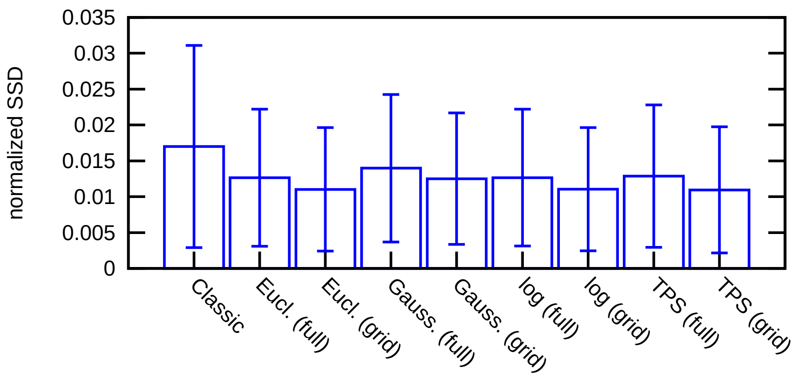


Figure 2.5.: Mean values of the normalized sum of squared differences (SSD) and standard deviations for the different kernel types used by the "full" and the two-staged ("grid") implementation of the visual forward model. "Classic" stands for the GRBF network by Schenck and Möller (2007). All models use the heuristic validator model; see text for further explanations.

2.5. Results

We compare the visual forward model using the "full" and the two-staged "grid" approach with different kernel functions against the original GRBF approach (Schenck and Möller, 2007). In order to quantify the results, we performed a large number of simulated camera movements using an image database. The image before each camera movement was used as input for the visual forward model; the motor command corresponding to the movement $\Delta\varphi, \Delta\theta$ was used to drive the prediction. Thus we can measure the difference between the "real" camera image after the movement and its prediction by the visual forward model. We use the normalized sum of squares error given by

$$E_{\text{SSD}} = \frac{1}{N_c \cdot M} \sum_{c=1}^{N_c} \sum_{u=1}^{w_{\text{out}}} \sum_{v=1}^{h_{\text{out}}} V(u,v) \cdot (I(c,u,v) - \hat{I}(c,u,v))^2, \tag{2.17}$$

where N_c is the number of channels (in our case $N_c = 3$ as we are using RGB images), w_{out} / h_{out} denotes the image width / height, respectively, and I is the original camera image and $\hat{I}$ is the prediction. Furthermore, V denotes a binary image (mask) whose values are either 1 for valid pixels or 0 otherwise, and M is the overall number of 1's in V.

For better comparison of the different methods, we precomputed V for each camera movement based on either the heuristic or the MLP validator model (see Section 2.4.4), respectively, and used these masks in the calculation of (2.17) (if not indicated otherwise). Furthermore, we used an efficient GPU implementation of the "full" approach.

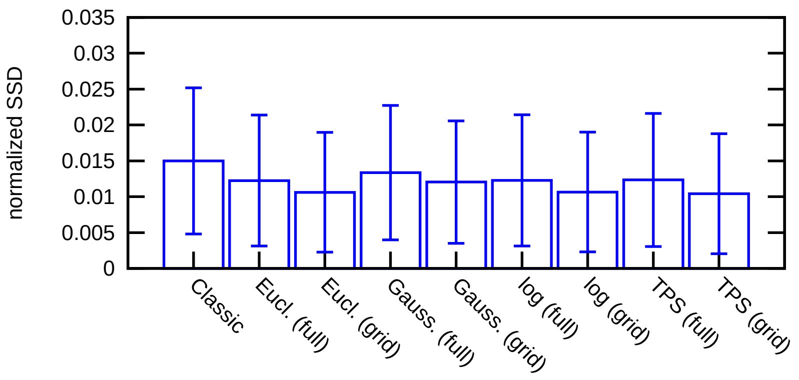


Figure 2.6.: The same as Figure 2.5 except that all models use the MLP classifier as validator model.

In order to provide rich variation in the images, we used different initial viewing directions (φ, θ). The initial pan angle was varied from $-35.14°$ to $-1.46°$ with a step width of $8.42°$, the initial tilt angle was varied from $-23.61°$ to $2.11°$ with a step width of $6.43°$. For each of the 25 initial viewing directions, a certain number of motor commands was executed. The pan movement, $\Delta\varphi$, was varied between approx. $\pm 29.47°$ with a step width of about $4.91°$, the tilt movement, $\Delta\theta$, was varied between approx. $\pm 28.94°$ with a step width of about $4.82°$. This resulted in a 5×5 grid of initial positions, and a (theoretical) 13×13 grid of PTU movements for each initial position. Motor commands that would have resulted in pan and tilt angles outside the valid range of the PTU were excluded from the tests, amounting to a total number of 3717 combinations.

We tested the "full" RBF mapping model and the two-staged "grid" mapping model with the procedure described above, using the masks generated by the heuristic validator model to compute the SSDs for each test position. Figure 2.5 shows the means of the normalized SSDs along with the standard deviation for the different mapping models and the heuristically derived validator masks. In this case, the validator masks are logical conjunctions of the heuristic masks (see Section 2.4.4) generated by each kernel function. A two-way repeated measures analysis of variances (ANOVA) shows that both factors, "grid" vs. "full" and kernel type are significant ($p < 0.001$), as well as the interaction between these factors ($p < 0.001$). Note that albeit the standard deviations are very high (see 2.5), we still get significant results. This is due to the fact that we work with repeated measures. The biggest part of the variance is due to the different initial positions and the camera movements (rather than the different conditions). In repeated measures ANOVA, these sources of variance are eliminated, hence the strong significance of the tests. This also holds for the results of the individual t-tests in the following.

From Figure 2.5 it becomes clear that the two-staged mapping models outperform their "full" counterparts. Therefore, we performed individual t-tests to see if the differences are significant. The two-staged mapping model using the TPS kernel performs best, resulting in a mean SSD of 0.0109 (0.0088), which is significantly different ($p < 0.001$) from the second best, the Euclidean kernel with a mean SSD of 0.0110 (0.0086). The worst two-staged mapping model is the one using the Gaussian kernel with a mean SSD of 0.0125 (0.0091) which is only slightly better than the best "full" model, the one using the Euclidean kernel with a mean SSD of 0.0126 (0.0095), albeit this difference is only significant under the 5% level.

Furthermore, we tested the full GRBF approach by Schenck and Möller (2007) (termed "classic") under the same condition which resulted in a mean SSD of 0.017 (0.0141) and thus performed significantly worse ($p < 0.001$) than the worst "full" model, the one using the Gaussian, that resulted in a mean SSD of 0.014 (0.0102).

The same experiment has been repeated with the MLP classifier as validator model. For this, we precomputed the corresponding validator masks for every combination of $\Delta\varphi$, $\Delta\theta$. Note that the "full" mapping models require a full validator mask, i.e. the validity of each pixel must be checked. Figure 2.6 shows the mean SSDs and the corresponding standard deviations for the different kernels and implementations of the mapping model, respectively.

We performed the same statistical analysis as for the heuristic validator model. The two factors "grid" vs. "full" and kernel type are significant ($p < 0.001$) as well as their interaction ($p < 0.001$).

The best-performing model is the two-staged mapping model using the TPS kernel which yields a mean SSD of 0.0104 (0.0084); the second best is the two-staged mapping model using the Euclidean kernel with a mean SSD of 0.0106 (0.0083) which differs significantly from the best ($p < 0.001$). The best "full" mapping model is the one using the Euclidean kernel with a mean SSD of 0.0122 (0.0091) which differs significantly ($p < 0.001$) from the worst two-staged model, the one using the Gaussian kernel, with a mean SSD of 0.0120 (0.0085).

We also tested the "classic" mapping model in this condition, which yields a mean SSD of 0.015 (0.0101). This value differs significantly ($p < 0.001$) from the worst "full" mapping model, the one using the Gaussian kernel, which yields a mean SSD of 0.0133 (0.0093).

Finally, Figure 2.7 shows the results of the "full" mapping models with their "native" validator models. For the "full" mapping model using the Euclidean and the log kernel, respectively, the native validator model corresponds to the heuristic given by (2.16). The difference between the tests presented in figure 2.5 is that, instead of a logical conjunction, we only used one type of kernel (either the Euclidean or log kernel) to generate the validator masks for the corresponding mapping model. We only consider the Euclidean and the log kernel in this test, because they were the only kernels that produce consistent validator masks (see the discussion in Section 2.4.4). Note that in the case of "Classic" the "native" validator model is a separate GRBF network with a single output, akin to the MLP classifier.

The "full" mapping / validator model using the log kernel yields a mean SSD of 0.0132 (0.0098) and thus perfoms best. Second best is the "full" mapping / validator model using the Euclidean kernel with a mean SSD of 0.0133 (0.0099) which is significantly ($p < 0.001$) worse than the best. Apparently the "Classic"

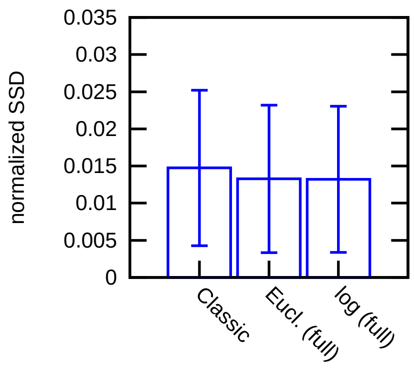


Figure 2.7.: The same as Figure 2.5 except that all models use their "native" validator model.

model model (in combination with its native GRBF validator model) shows the worst performance, yielding a mean SSD of 0.0147 (0.0105), this result also differs significantly ($p < 0.001$) from the one obtained by the Euclidean mapping / validator model.

We compared the performance of the different mapping models / kernel functions using different variants of validator models. Note that not every combination of mapping model / validator model makes sense in practise. For the "full" mapping models, using the "native" heuristic validator model is the most practical choice: its application is possible without any additional computational cost. Using a classifier-based validator model in this context would result in checking the validity of each pixel separately which would introduce a massive computational overhead. The two-staged mapping models on the other hand are based on the prediction of only a few grid points (and subsequent interpolation by the second stage). Thus the validity of only a small number of points needs to be checked. In this case, using a classifier-based validator model would be the most obvious choice.

Furthermore, note that the results depend on the choice of the validator model: for the MLP-based validator model, the mean SSDs are slightly lower compared to the heuristic one. We assume that this effect is caused by the fact that the MLP classifier produces stricter (i.e. tighter) validator masks that cut off larger parts at the margins of the valid portion compared to the heuristic validator model. These margins are prone to erroneous behavior of the mapping models because they lie close to the edge of the domain (where erratic extrapolation might occur). Thus, tightening the validator masks results in smaller SSDs, but also reduces the size of the predicted portion.

Figure 2.8 shows 9 different predictions, using the two-staged mapping model (see Section 2.4.3) with the Euclidean kernel, alongside the original camera images. Only the portions of the images that were classified as valid by the MLP validator model (see Section 2.4.4) are shown (all other pixels are masked as black). The center image is the basis for each prediction.

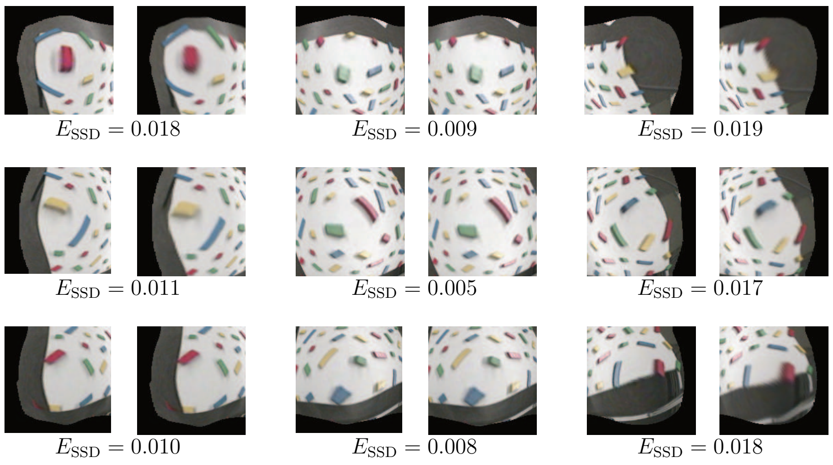


Figure 2.8.: Original–prediction pairs and corresponding SSDs for the 9 different motor commands from $\mathcal{G}_{\text{test}}$ using the two-staged mapping model in conjunction with the Euclidean kernel and the MLP validator model.

The values of the motor commands $\Delta\varphi, \Delta\theta$ were varied within the grid

$$\mathcal{G}_{\text{test}} = \left\{-\frac{1}{2}\Delta\varphi_{\text{max}}, 0, \frac{1}{2}\Delta\varphi_{\text{max}}\right\} \times \left\{-\frac{1}{2}\Delta\theta_{\text{max}}, 0, \frac{1}{2}\Delta\theta_{\text{max}}\right\},$$

where $\varphi_{\text{max}}/\theta_{\text{max}}$ denotes the maximum pan/tilt angle. Note that the position $0, 0$ is also included in the comparison.

At the center position $0, 0$ we see two effects: (i) there are no invalid pixels (i.e. the validator mask consists of all ones), and (ii) the SSD between the original camera image and the prediction is minimal (compared to all other positions). If a pure tilt movement is predicted (i.e. $\Delta\varphi$ is fixed at 0), the SSDs remain in the same order as at the center position. For pure pan movements (i.e. $\Delta\theta = 0$) the SSD increases slightly. For the prediction of combined pan-tilt movements the SSD increases even more for at least 3 positions. (Note that the SSD at the lower left position is very small which is most likely due to the large "white" portion in the images.) We conclude that this effect is due to the slight deviation of the PTU from a perfect gimbal, i.e. the axes of rotation do not intersect in the ray emanating from the entrance pupil of the camera. Therefore, the current pan and tilt angles have a small influence on the change of the camera image when the PTU is moved by $\Delta\varphi, \Delta\theta$. See the appendix in Schenck (2008) for a detailed analysis of the kinematics of the PTU used for our experiments.

2.6. Conclusions & Outlook

The problem of visual prediction approached in this chapter is to anticipate the effects of a camera movement on the current camera image. If the camera movement can be parameterized, e.g. by two relative angles, $\Delta\varphi$, $\Delta\tau$, we can perceive the problem as a functional mapping that maps the input image (i.e. the current camera image) onto the new camera image after the movement using the relative angles as additional parameters. We employ an adaptive class of functions, the so-called radial basis function (RBF) networks, to "learn" this mapping. RBF networks are a linear combination of non-linear kernel functions.

We presented a comparative study on the influence of the used kernel function in an RBF network on the quality of the approximation in the context of a visual prediction task. Furthermore, a more efficient evaluation scheme—the so-called two-staged mapping model—was presented. The experimental results show that the choice of kernel function has a certain impact on the quality of the prediction. We showed that there exist alternatives to the well-established Gaussian kernel that stem from the so-called semi-definite class of kernels. These kernels have the main advantage that they lack a width parameter that depends on the scaling of the data and thus would have to be estimated for the problem at hand. Secondly, the Euclidean kernel which belongs to this class is computationally very simple since it only depends on the Euclidean distance between data points and input weights of the corresponding RBF network. For the visual prediction task considered in this chapter, the Euclidean kernel even beats the Gaussian one in terms of prediction quality.

The costly "full" mapping model, which is based on passing each pixel from the output image through a very large RBF network, shows only a moderate improvement over the "classic" method proposed by Schenck and Möller (2007). A two-staged mapping model, in which prediction of only a few anchor points and image interpolation are decoupled, shows a better performance in terms of the prediction error and in terms of computation speed. The "full" mapping model has a complexity of MN operations, were M denotes the number of pixels in the output image and N denotes the number of RBFs. Note that both quantities are, for the application considered in this chapter, in the magnitude of 10 000. The "classic" approach still has a complexity of Mk operations, where $k = 1500$ is the number of anisotropic basis functions. In contrast to that, the two-staged mapping model has a complexity of $Mn + 2Nn$ operations, where $n \ll N$ (in our case $n = 169$) is the number of grid points which means a considerable reduction of the computational cost.

One remaining issue is the classification of the validity of grid points in the two-staged approach. By now, we use a separate classifier—currently an MLP—although it would be more desirable to use the information already stored in the kernel matrix (used for the prediction stage) to perform this classification. This would yield a kind of novelty detector which assigns a confidence value to each grid point (based on the current $(\Delta\varphi, \Delta\theta)$). In the context of Gaussian processes this is known as the predictive variance (Rasmussen and Williams, 2005) or kriging error (Dubrule, 1984) which is defined as

$$\mathrm{Var}[\hat{f}(\vec{x})] = \kappa(\vec{x}, \vec{x}) - \vec{k}(\vec{x})^\top K^{-1} \vec{k}(\vec{x}), \tag{2.18}$$

where $\vec{x}$ is the test input and $\vec{k}(\vec{x})$ is the cross-kernel vector between $\vec{x}$ and all input weights. A data point at which the predictive variance exceeds a certain threshold would be classified as invalid.

In contrast to the technical definition of RBF networks we use in this chapter, Gaussian processes and kriging both have a sound probabilistic interpretation, viewing the kernel function as covariance function. Thus, (2.18) can be interpreted as variance in the strict mathematical sense. However, it is unclear if this interpretation still holds if the kernel function is only conditionally positive definite. Preliminary experiments show that $\mathrm{Var}[\hat{f}(\vec{x})]$ can be computed using these kernels (if K is augmented in the same fashion as in (2.5)) such that it is strictly positive. However, this statement still needs to be proven. From a computational point of view, the complexity of computing (2.18) grows quadratically with the number of RBF units, which amounted to $\sim 11,000$ in our experiments. In order to reduce this effort one could use the best rank-p approximation of K which is given by $\hat{K} = U_p \Lambda_p U_p^\top$, where Λ_p is a diagonal matrix with the p largest magnitude eigenvalues and U_p the corresponding eigenvectors (as columns). Thus (2.18) becomes

$$\widetilde{\mathrm{Var}}[\hat{f}(\vec{x})] = \kappa(\vec{x}, \vec{x}) - \vec{k}(\vec{x})^\top U_p \Lambda_p^{-1} U_p^\top \vec{k}(\vec{x}). \tag{2.19}$$

Numerical experiments (not shown) suggest that this approximation imposes a regularization on $\widetilde{\mathrm{Var}}[\hat{f}(\vec{x})]$ that is controlled by p, the number of ranks used for the approximation. It should be noted that this approximation scheme is closely related to the reconstruction error of kernel PCA which has already been successfully applied to the problem of novelty detection (Hoffmann, 2007b).

For the "full" mapping model, a heuristic validator model which is based on the extrapolation behavior of certain kernels, i.e. the Euclidean and the log-kernel, can be easily incorporated. Thus, no external classifier is needed. Furthermore, in order to practically apply the full RBF-based mapping to images of even moderate sizes, we proposed a highly parallel GPU implementation of the RBF network. Comparisons show that this implementation considerably reduces the execution time of the image warping in comparison to serial (CPU) implementations. By now, the big overhead still lies in the acquisition of training data and the "learning" phase of the RBF network which is performed by means of solving a linear system. Improving these two aspects will be the subject of further investigations.

There is a multitude of potential applications for a visual forward model. Visual prediction can be used, for example, for shifting an agent's attention without executing overt eye movements (Schenck, 2013). The "classic" method (Schenck and Möller, 2007) has already been used in the context of extra-foveal grasping (Schenck et al., 2011; Schenck, 2013), i.e. grasping of objects that are not fixated. The experimental setup consists of a robot manipulator with attached gripper and a stereo vision system with four degrees of freedom (DOF). The vision system uses retinal images (see Section 2.2.2). In the case of foveal grasping, the cameras fixate the object prior to grasping. The viewing direction of the cameras along with some visual information (encoding the object's orientation) are used as input to a motor controller. The motor controller then generates a motor command that controls the manipulator in such a way that the object can be grasped. In the case of extra-foveal grasping, the fixating movement of the cameras is omitted and replaced by pure visual prediction. Using visual prediction instead of the raw unfixated, extra-foveal camera views drastically improves the success rate of the graping task (Schenck et al., 2011). So the same motor controller that has been trained on fixated views can be used for solving the extra-foveal grasping task.

Another application is in the context of view synthesis. Here, an agent generates views of its own body (especially its manipulators) based on a set of pose parameters. These pose parameters might relate to the body configuration and the viewing direction. The task is to establish a mapping between these pose parameters and the view of the agent. To reduce the complexity of the learning task, the association problem, i.e. the association between the body pose and its appearance, can be decoupled from the visual prediction task (see Kaiser et al. (2011); Schenck (2013), and Chapter 4). This means that during the learning phase, the agent (or its parts) always appears in a specific way, e.g. as if fixated by the camera. In the application phase, visual prediction can be used to shift the gaze such that the agent's body parts appears from an arbitrary viewing direction, passing the output of the association as input to the visual forward model.

In this chapter we reviewed efficient methods for visual prediction in a fully adaptive framework. The adaptability has the great benefit that (besides the optic flow induced by camera movements) the model also learns the image distortion caused by the optical components of the camera. We showed, that the adaptive model can cope even with strongly (retinally) warped images. In the next chapter, we will review the problem of visual prediction from a purely geometric perspective. We will employ an approximate geometric model of the camera–PTU assembly which will serve as a starting point for the derivation of a geometric visual forward model. The geometric model is computationally simpler than the fully adaptive approach but less accurate—due to the fact that it relies on a set of pre-defined parameters.

3. A Geometric Model for Visual Prediction and Saccade Control

3.1. Introduction

In this chapter, we will put the problem of visual prediction under close scrutiny by viewing it from a purely geometric angle. From this perspective, we will also derive a solution to its inverse problem, i.e. saccade control. The resulting visual forward model and its corresponding inverse, the saccade controller, represent computationally efficient alternatives to their fully adaptive counterparts (see Chapter 2) and other models in the literature. The internal models that will be derived in the following are still approximations (i.e. not perfectly capturing every aspect of a real camera–PTU assembly), but rely on a significantly smaller set of parameters.

While the problem of visual prediction has received only little attention by others—which may be due to its great specificity—there exists a large body of literature on saccade control, both in the context of robotics and cognitive science since saccade control is an important prerequisite for attention and visual perception. Thus, giving an exhausting review of existing models is difficult.

Dean et al. (1994) propose a neurally-inspired adaptive model for saccade control based on feedback-error learning (Gomi and Kawato, 1993). In feedback-error learning, a simple feedback controller is gradually improved by an adaptive controller which is trained on-line. Convergence to a perfect controller has been proved for this principle (Gomi and Kawato, 1993) which is biologically plausible as well. The adaptive model for saccade control has been tested successfully in a series of simulated robot experiments (Dean et al., 1994). Based on similar principles, Bruske et al. (1997) devised an architecture for the control of a binocular camera head. Their architecture has been successfully tested on a real robotic setup. Shibata and Schaal (2001) employed feedback-error learning for a variety of oculomotor behaviors. They also introduce the notion of retinal vision, i.e. the non-uniform distribution of cones on the retina. They tested their controller on a stereo camera head equipped with two cameras per eye: one for peripheral and one for foveal vision, respectively. Milighetti et al. (2011) suggest an extended gaze control scheme that encompasses the control of the upper body as well. Considering the whole upper body results in additional redundant degrees of freedom which have to be dealt with. The model which they devise, (besides a feedback controller and an adaptive controller) also comprises an adaptive Kalman filter for the prediction of future object locations. They successfully tested their architecture on a humanoid robot.

Besides the above studies (which are mostly based on feedback-error learning), Schenck and Möller (2004) proposed an alternative learning strategy for saccade control (or motor control in general), termed *learning by averaging*, which does not rely on a feedback controller, but is based on a "smart" choice of training examples and the averaging behavior of feed-forward networks (i.e. the error in the training examples is nearly averaged out). For retinally distorted images, Schenck (2013) successfully applied direct inverse modelling (Kuperstein, 1988) for the training of a fully adaptive saccade controller. Furthermore, a visual forward model was employed in order to re-identify the targets under the retinal distortion. Both approaches were successfully tested on a real robot camera head.

In the following, we will first introduce a geometric model of a single 2-DOF robotic eye or pan-tilt-unit (PTU). Based on this model, we will devise corresponding internal models for visual prediction (forward model) and saccade control (inverse model). The internal models will be tested on a real robotic setup. Furthermore, we will devise an adaptive control scheme for the saccade controller that drastically improves its performance.

3.2. Geometric model

The starting point for the derivation of our geometric model is the assumption that the effect on the visual content of the camera image during a PTU movement (i.e. the optic flow) is only a function of the relative movement parameters and independent of the initial viewing direction. This assumption has been already made in the last chapter. Schenck has shown (Schenck, 2008) that this assumption holds partly for the real PTU–camera assembly used for the experiments: while the current pan angle has no influence on the optic flow resulting from a saccade, the current tilt angle still has an effect (see Schenck (2008) for a detailed analysis).

However, in the following, we will approximate the real kinematics of the PTU by a *gimbal* or spherical manipulator. The camera is modeled by a pin-hole camera (Jaehne, 2005) whose optic center lies in the intersection of the pan and tilt axes. Under these circumstances, the above-mentioned assumption holds.

The workspace of the gimbal (PTU) describes a sphere of radius f, the focal length of the camera–lens system. The image plane is given by a plane tangential to the gimbal sphere; the plane's normal vector $\vec{n}$ is given by a 3D vector in spherical coordinates, $\Delta\varphi, \Delta\theta, f$. Furthermore, we will restrict the image plane to a $w_s \times h_s$ square region around the intersection of the image plane and its normal, where w_s and h_s correspond to the physical extents of the image sensor. The coordinate system is chosen such that the z-axis is initially (i.e. for $\Delta\varphi = \Delta\theta = 0$) congruent with the normal vector and the x- and y-axes span the image plane.

Figure 3.1 shows the main components of the model: the gimbal sphere is shown in gray, the image plane is shown in green, and the normal vector $\vec{n}$ in red. The coordinate system is located in the center of the sphere. The figure depicts the situation in which the camera is panned by $\Delta\varphi = \pi/6$ and tilted by $\Delta\theta = \pi/8$; the

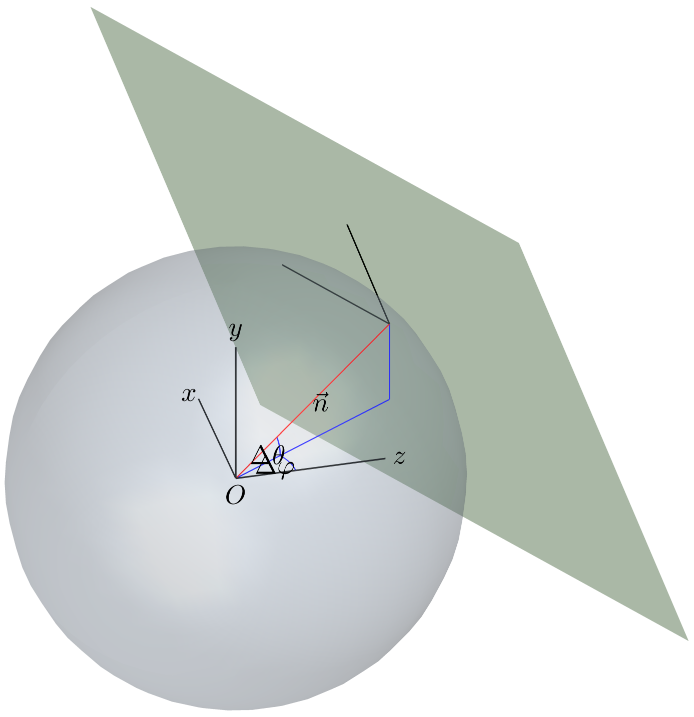


Figure 3.1.: Geometric model of the PTU–camera assembly. The gimbal sphere is shown in gray, the camera plane is shown in green. The plane normal $\vec{n}$ is panned by angle $\Delta\varphi$ and tilted by angle $\Delta\theta$.

focal length is $f = 0.75$ in this example.

The problem of visual prediction can thus be formulated as a rotation of the tangential image plane around the gimbal sphere, while the problem of saccade control can be solved by computing the inverse kinematics of the model. The resulting internal models only depend on a few parameters, i.e. f, w_s and h_s, which can be either obtained from the technical specifications of the camera or alternatively learned by a similar approach as described in Chapter 2, albeit the choice of initial values of the parameters might be critical.

We furthermore extended our model to encompass image distortion introduced by the lens of the optic system or artificial distortions like the retinal mapping (see Section 3.3.3). This extension introduces further parameters whose number depends on the chosen distortion model.

In the following, we describe the necessary steps taken for the derivation of the geometric visual forward model and its inverse, and present experimental results in a similar fashion to those in Chapter 2.

3.3. Visual forward model

The problem of visual prediction—i.e. modelling the effect of a camera movement on the content of the camera image—has been treated from a learning perspective in the previous chapter. In this section we will treat visual prediction from a geometric point of view to gather further insights into the nature of the problem. Furthermore, we will overcome two major shortcomings of the learning-based approach: (i) the problem of data collection which was tackled by employing a statistical learning approach (using cumulator units), and (ii) the problem of finding a reasonably computationally efficient neural network that interpolates the training data during the application phase at a good quality.

The training data for the adaptive visual forward model of Chapter 2 was collected defining a grid of cumulator units in a 4 dimensional space. The aim was to capture the forward characteristics of the camera–PTU assembly, i.e. the sparse optic flow within the camera image for a given number of camera images. Such a learning approach requires a large number of steps (depending on the size of the grid) to converge. The quality of the resulting training set cannot be determined directly. Such an assessment would require the solution of the correspondence problem (i.e. finding homologous points in a pair of images) which, if solved, would supersede the statistical learning procedure itself. Therefore, the training data may already contain inaccuracies.

Once the training data is collected, a neural network model has to be fitted in order to apply the forward model. The gridded nature of the training data lead to the choice of using full RBF networks in this context. Full RBF networks use the whole training set as input weights which make them computationally expensive if the training set is large. Using only a restricted number of input weights on the other hand leads to over-smoothing of the data and eventually to even larger inaccuracies of the forward model (in addition to those already present in the training set).

For complex cognitive architectures (like the model for simulated object interaction, see Chapter 1)—in which visual forward models are crucial building blocks—it is important that all sub-models are computationally tractable so that the internal simulation threads terminate in a reasonable time. These requirements justify the development of computationally simple, yet sufficiently accurate model which are non-adaptive but still retain the main characteristics of their adaptive counterparts—they should operate on real sensory data (i.e. images) and only use as little knowledge about the world as possible.

In the following, we will develop a visual forward model based on the geometric model of the camera–PTU assembly. We will derive a predictive transform for points on the image plane, given a specific camera movement, address the problem of lens distortion and discuss the problem of image interpolation. The final model has the same capabilities as the adaptive forward models and can also be used on retinal images (Schenck and Möller, 2007; Schenck, 2008). We compare its performance to the adaptive model from Chapter 2 by performing the same experiment.

3.3.1. Transformation

Before we start can start to derive a visual forward model that transforms (warps) the entire image, we will focus on the transformation of a single point $\vec{p}$ on the image plane. We will consider relative changes of the viewing direction, referred to as (abstract) motor commands in the following. Such a motor command is defined by a vector $\vec{m} = (\Delta\varphi, \Delta\theta)^\top$, where $\Delta\varphi$ and $\Delta\theta$ denote the relative changes of the pan and tilt angles, respectively. Our aim is now to (i) find a transformation that transforms $\vec{p}$ according to $\vec{m}$, and (ii) find the projection of the transformed point back onto the image plane.

Based on the gimbal assumption, we can safely assume that our geometric model is independent of the current viewing direction, and always define the world coordinate system such that the initial (unrotated) image plane is orthogonal to the z-axis.

Let the point on the image plane be given by

$$\vec{p} = \begin{pmatrix} u \\ v \\ f \end{pmatrix},$$

where u and v denote the position of the corresponding pixel on the image sensor (in millimeters) and f denotes the focal length (also in millimeters. The pixel positions can be calculate by a simple transformation (see Section 3.3.2). Alternatively one could transform the focal length into pixel units—but this would complicate the following calculations because the pixels on the sensor are not necessarily quadratic but rectangular which would result in separate values of the focal length in x and y direction.

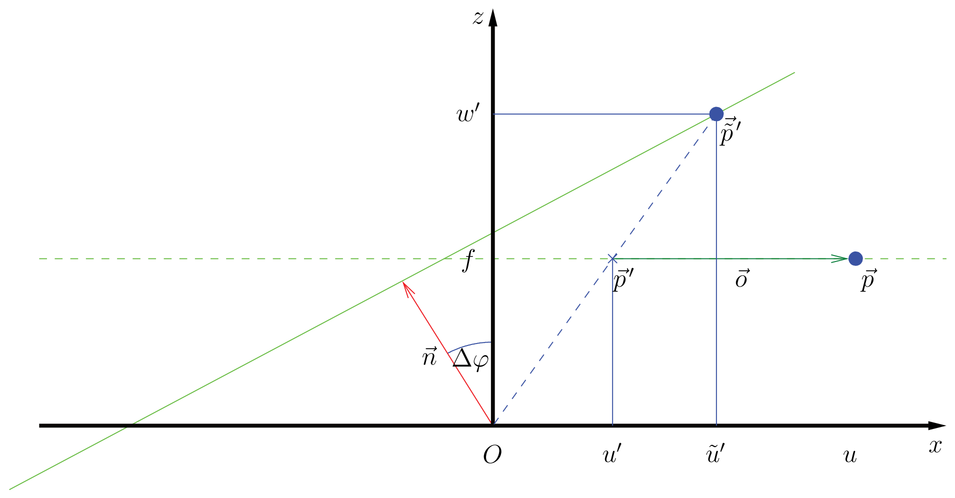


Figure 3.2.: Perspective transform for the visual forward model (2D). A point $\vec{\tilde{p}}'$ on the rotated image plane (green) is projected onto the unrotated image plane (stroked). The coordinates of resulting point $\vec{p}'$ are determined using the intercept theorem.

The PTU has two degrees of freedom, it can rotate around the y-axis (pan) and around the x-axis (tilt) (see Figure 3.1). Therefore, the position of $\vec{p}$ on the image plane after the "execution" of a motor command is given by

$$\begin{aligned}\vec{\tilde{p}}' = \begin{pmatrix} \tilde{u}' \\ \tilde{v}' \\ w' \end{pmatrix} &= R_x(\Delta\theta) R_y(\Delta\varphi)\,\vec{p} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\Delta\theta) & -\sin(\Delta\theta) \\ 0 & \sin(\Delta\theta) & \cos(\Delta\theta) \end{pmatrix} \begin{pmatrix} \cos(\Delta\varphi) & 0 & -\sin(\Delta\varphi) \\ 0 & 1 & 0 \\ \sin(\Delta\varphi) & 0 & \cos(\Delta\varphi) \end{pmatrix} \begin{pmatrix} u \\ v \\ f \end{pmatrix}, \end{aligned} \tag{3.1}$$

where $\Delta\varphi$ and $\Delta\theta$ denote the relative changes in of the pan and tilt angles.

The above equation gives the new position of $\vec{p}$ after a pan an tilt movement in world coordinates. In order to calculate the optic flow (i.e. the displacement of $\vec{p}$ on the image plane), we need to project $\vec{\tilde{p}}'$ back onto the initial (i.e. non-rotated) image plane.

Figure 3.2 shows the situation in 2D: the initial image plane is shown as a dashed green line, at distance f parallel to the x-axis. The rotated image plane is shown as solid green line, rotated by $\Delta\phi$ around the optic center (O). Point $\vec{\tilde{p}}'$ is shown on the rotated image. The task is now to determine the unknown position u', i.e. the projection of $\vec{\tilde{p}}'$ onto the initial image plane. Obviously, this so-called perspective

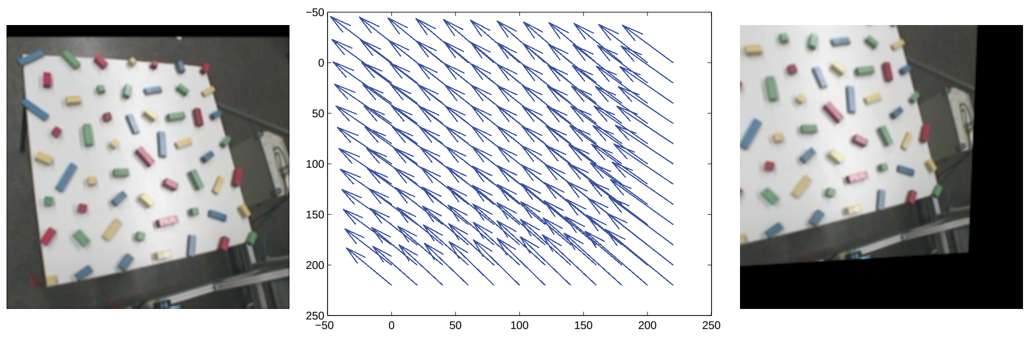


Figure 3.3.: Example for the geometric visual forward model: Input image (left), sparse optic flow field (middle), interpolated output image (right).

transform can directly be derived from the intercept theorem by observing that

$$\frac{w'}{\tilde{u}'} = \frac{f}{u'}$$
$$\Leftrightarrow u' = \frac{f\,\tilde{u}'}{w'}.$$

We can now extend the 2D case to 3D by observing that $v' = \frac{f\,\tilde{v}'}{w'}$, and formulate the perspective transformation for the geometric visual forward model which is given by

$$\vec{p}\,' = \begin{pmatrix} u' \\ v' \\ f \end{pmatrix} = \begin{pmatrix} \frac{f\,\tilde{u}'}{w'} \\ \frac{f\,\tilde{v}'}{w'} \\ f \end{pmatrix}, \tag{3.2}$$

where u' and v' denote the position of pixels u and v on the image plane after a camera movement. The optic flow vector (from source to destination point) for a given motor command $\vec{m}$ can now be computed by $\vec{o} = (u, v)^\top - (u', v')^\top$ (see Figure 3.2). Furthermore, we see that the optic flow is independent of the depth of the scene—i.e. the world coordinates of $\vec{p}\,'$—which is an inherent property of the underlying geometric model (gimbal: no translation).

Figure 3.3 (middle) shows an example of an optic flow field for a combined pan and tilt movement. The field was generated by transforming every pixel of the image sensor by the visual forward model. However, only a 12×12 grid of the whole (dense) flow field that describes the translation of each pixel in the image is depicted.

3.3.2. Interpolation

In the last section we have derived a visual forward model based on the geometric model of the camera–PTU assembly. The forward model allows us to compute the desired optic flow for points on the image plane (i.e. the image sensor) for a given camera movement. In order to apply this forward model to a digital image, we need to transform the pixel coordinates of the image into position on the image plane (the sensor frame) and employ an interpolation scheme to warp the whole image.

The first step is necessary to ensure that the pixel coordinates are given in the same unit as the focal length (in this case millimeters)—which is a crucial parameter of the geometric model. If we assume that the image is sufficiently principal point corrected (i.e. the principal point lies in the image center), we can apply the following transformations:

$$\begin{aligned} u &= \frac{w_S\, u_I}{w_I - 1} - \frac{w_S}{2} \\ v &= \frac{h_S\, v_I}{h_I - 1} - \frac{h_S}{2}, \end{aligned} \tag{3.3}$$

where w_S / h_S denote the sensor width / height (in millimeters), w_I / h_I denote width / height of the image (in pixels) and u_I / v_I denote the discrete pixel coordinates. The inverse transformation is trivial.

Now we are able to compute the mapping for a given motor command, i.e. computing $\vec{p}'$ for every pixel in the output image. The resulting mapping can be applied by using a suitable interpolation scheme. For our experiments, we used the function `remap` from the OpenCV library (Bradski, 2000), and chose the bicubic interpolation method (Hou and Andrews, 1978). Note that the choice of the interpolation method is uncritical here, since the optic flow is nearly linear and does not result in drastic magnifications which would require more advanced interpolation techniques.

Figure 3.3 shows an example of the visual forward model: the source image (left) is warped according to the optic flow field (middle) which results in the destination image (right). Note that the shown flow field only depicts a 12×12 subset of all flow vectors.

The adaptive visual forward model described in Chapter 2 consists of two submodels: the mapping model (for image warping) and the validator model. The task of the validator model is to mask out pixel that do not have a corresponding partner in the source image (and are thus not predictable). This was a necessity, because the RBF networks (especially the Gaussian RBFN) tend to behave erratically in regions without valid training data which leads to erroneous extrapolations. This problem is not an issue for the geometric forward model; the invalid pixels are always mapped beyond the extents of the source image which can be easily detected as in the heuristic validator model (see Chapter 2). Therefore, there is no need for a dedicated validator model.

3.3.3. Image Distortion

So far, we assumed that the camera images are free of geometric distortions and neglected the fact that most real cameras incorporate lenses to focus the light. Lenses always lead to geometric distortions within the images (Jaehne, 2005). The adaptive visual forward model described in Chapter 2 is immune to such distortions, because it not only learns the optic flow of camera movements but also captures

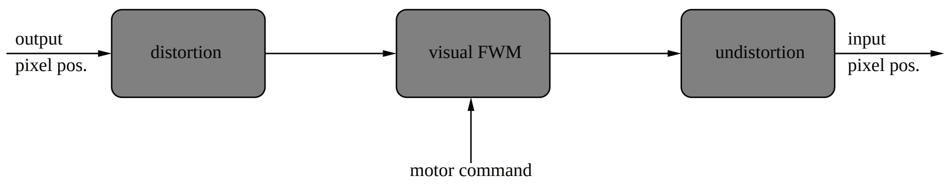


Figure 3.4.: Extended visual forward model. Output pixels are distorted before transformed by the geometric forward model. The resulting input pixels are undistorted by the inverse distortion model.

the distortion characteristics of lens. We used retinally distorted images to demonstrate the feasibility of the adaptive approach even in the context of highly distorted images. A drawback is the fact that the training data (and thus the model) inseparably contains information on the optic flow and the distortion. This leads to the disadvantage that the whole data set needs to be collected anew when the distortion model of the lenses changes.

In the following we will extend the geometric forward model to account for image distortions. Instead of changing the model itself, we will augment it by suitable auxiliary modules that transform the pixels according to a given distortion model. Figure 3.4 shows the structure of the extended model: the output pixel is distorted (reverse mapping) so that the output image has the same distortion as the input image, then the geometric visual forward model is applied, finally the input pixel is undistorted.

There exist a plethora of distortion models for different lens types (e.g. Fryer and Brown (1986); Barreto et al. (2009)) and corresponding toolboxes (e.g. Bouguet (2008); Barreto et al. (2009)) that allow the user to estimate the distortion coefficients for a specific camera. Basically all of those models provide transforms for distortion and (more importantly) undistortion. Therefore, they can be conveniently used in combination with the geometric forward model. However, we will only consider the retinal mapping which is used by the cognitive architecture described in this thesis. This enables us to compare the performance of the geometric forward model to the results obtained in Chapter 2.

The retinal mapping is given by an equation that maps the radius of the polar coordinate of any output pixel (leaving the polar angle unchanged, $\phi_{\text{in}} = \phi_{\text{out}}$) (Schenck and Möller, 2007; Schenck, 2008):

$$r_{\text{in}} = (1-\lambda)r_{\text{out}} + \lambda r_{\text{out}}^{\gamma}, \tag{3.4}$$

where $\lambda = 0.8$ and $\gamma = 2.5$ throughout this thesis, and the output coordinates are normalized such that $r_{\text{out}} \in [0, \sqrt{2}]$. In our current setting, this equation would be the content of the distortion module in Figure 3.4. Defining the undistortion module as the identity, we would yield a visual forward model that takes undistorted camera images as inputs and produces retinal images as output. Such an asymmetric model would not be of great use. Our goal is a model that also works on retinal images as inputs.

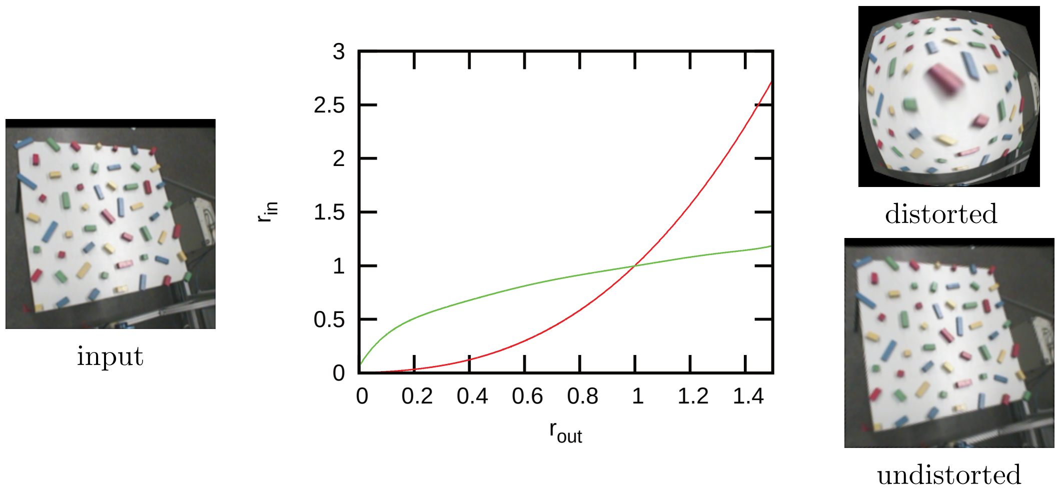


Figure 3.5.: Input image (left), the distortion (red curve, middle) and approximation of the undistortion function (green curve, middle), and two examples of the distortion and undistortion, respectivly (right).

Unfortunately, Equation (3.4) cannot be inverted analytically to obtain the undistortion model. Therefore, we have to find a sufficiently exact approximation to the inverse. We chose a polynomial of degree 7 as an interpolant. The training points for determining the polynomial coefficients were calculated by inverting (3.4) numerically on the interval $[0, 1.5]$ (1501 training points, step size 0.001). We used the Nelder-Mead downhill-simplex method, implemented in the Matlab function `fminsearch`. The polynomial coefficients were determined by the function `polyfit`. Figure 3.5 (middle) shows the distortion function (in red) alongside the polynomial approximation of its inverse (in green) The resulting undistortion is very accurate (Figure 3.5, right bottom).

We now have everything in place to implement a symmetric visual forward model (as depicted in Figure 3.4) that takes retinal images as inputs and produces retinal images as output. Thus, the geometric visual forward model mimics the characteristics of the adaptive model presented in the previous chapter, i.e. it uses retinal images as inputs and generates retinal images as outputs.

3.3.4. Results

In what follows, we will quantitatively asses the performance of the geometric visual forward model. We repeat the same experiment as in Chapter 2. This allows us to directly compare the results those obtained for the adaptive visual forward model in the previous chapter.

We will briefly recapitulate the main steps of the experiment. The goal of the experiment is to compare the predicted images generated by the visual forward model to real images of a scene under different viewing directions. The camera images are

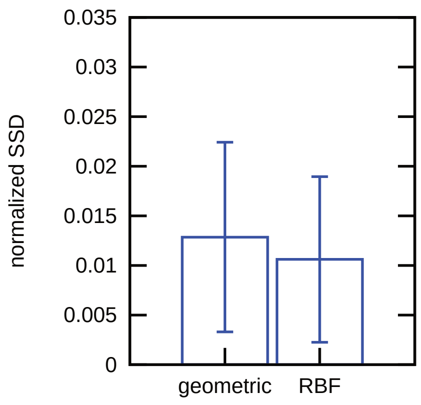


Figure 3.6.: Mean normalized SSD of 3717 post-saccadic image comparisons and corresponding standard deviations. The plot shows the results for the geometric and an RBF-based forward model.

recorded by the real camera–PTU setup. During the course of the experiment, the PTU is moved along a grid of initial viewing direction. For each viewing direction, the camera takes an image, the so-called pre-saccadic image, which serves as input for the visual forward model. Based on the initial viewing direction, the PTU executes a number of saccades (see Chapter 2). The corresponding motor commands are used to generate the predicted post-saccadic images. These predicted images (which correspond to the output of the visual forward model) are compared against the real post-saccadic images taken by the camera.

As a similarity measure, we use the normalized sum of squared differences (SSD) error. In order to be consistent with the adaptive visual forward model, the output images of the geometric forward model were cropped to a 159×159 region around the center prior to the computation of the SSD. During the experiment, a total number of 3717 comparisons were made. Figure 3.6 (left) shows the mean normalized SSD over all image comparisons and the standard deviation as error bar for the geometric forward model.

Figure 3.6 (right) also shows the results for an RBF-based adaptive forward model, in this case the two-staged approach using the Euclidean kernel in both stages (see Chapter 2). In comparison to the geometric model, the RBF-based model performs better. This may be due to the fact that the adaptive model is still an approximation and contains parameters that are not exact (i.e. the size of the image sensor and the focal length. Furthermore, we assume that the camera images are perfectly undistorted which does not hold. We only considered the retinal distortion which is dominant. The adaptive forward model on the other hand captures the image distortion in its entirety (i.e. lens distortion and retinal mapping). Still, the results suggest that for most applications (e.g. image matching) the geometric visual forward model performs sufficiently well.

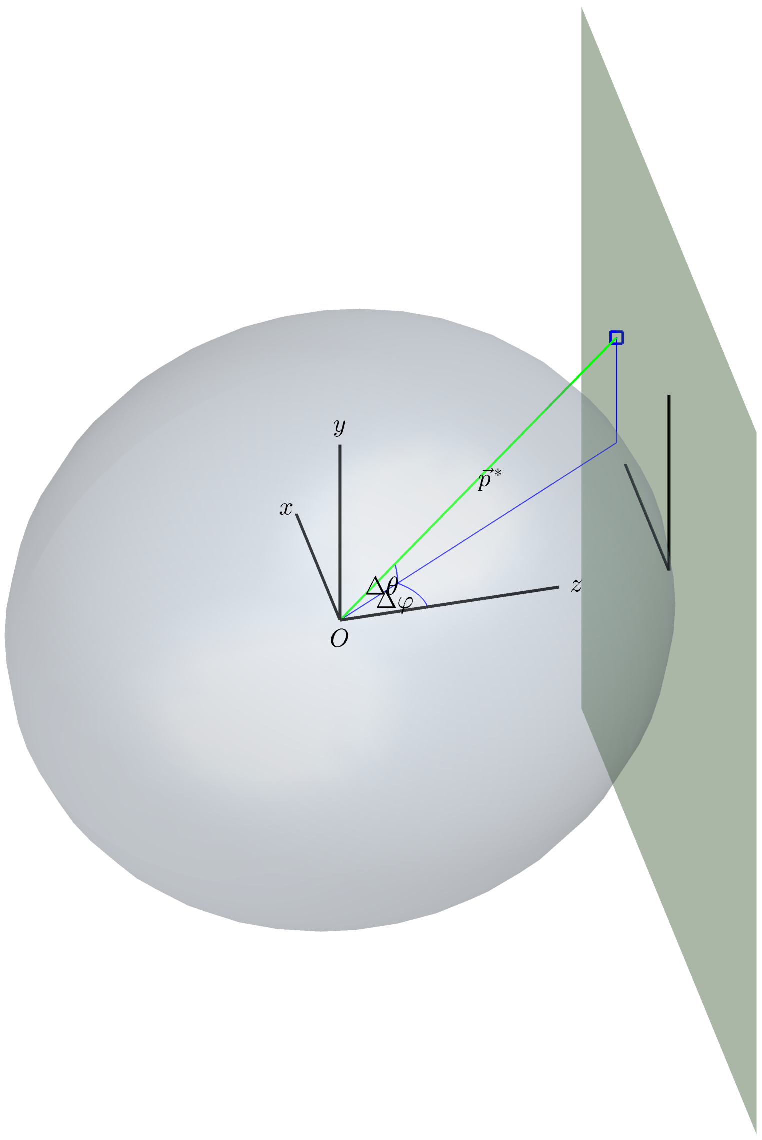


Figure 3.7.: Geometric model for saccade control. The gimbal sphere is shown in gray, the camera plane is shown in green. The position vector $\vec{p}^*$ of the target pixel (u^*, v^*) is also shown in green.

3.4. Saccade Control

The problem of saccade control is conceptionally simpler than visual prediction, but can be derived from the same geometric model. The reason for this is that we have only the target location as input and the corresponding fixation motor command as output. The problem of saccade control can be seen as an inverse to visual prediction. In terms of internal models, the saccade controller is thus an inverse model or motor controller.

In this section we will derive a geometric saccade controller for the gimbal approximation of the PTU. This controller is the exact inverse of the visual forward model. That means that we can use both models in combination to generate perfect simulated fixated views. However, when the saccade controller is used on the real PTU, its performance is unsatisfactory, because the real PTU–camera assembly deviates from an ideal gimbal. Therefore, we will augment the basic geometric saccade controller with an adaptive correction network which yields a higher saccadic accuracy.

3.4.1. Geometric Saccade Controller

The starting point for the derivation of the geometric saccade controller is depicted in Figure 3.7: the image plane is orthogonal to the z-axis. The task is now to rotate the image plane such that the target point $\vec{p}^*$ (depicted as a blue square in Figure 3.7) lies in the image center. This means that we have to determine the pan and tilt angles such that the normal vector $\vec{n}$ becomes collinear to the position vector of the target point.

The position vector to the target point is given by $\vec{p}^* = (u^*, v^*, f)^\top$. It follows from straight trigonometry that the two angles $\Delta\varphi, \Delta\theta$ can be obtained by:

$$\begin{aligned} \Delta\varphi &= \operatorname{atan2}(u^*, f), \\ \Delta\theta &= \operatorname{asin}\left(\frac{v^*}{\sqrt{u^{*2} + v^{*2} + f^2}}\right), \end{aligned} \tag{3.5}$$

where u^* and v^* denote the position of the target pixel on the sensor, and f denotes the focal length (all values in millimeters).

The resulting angles define an abstract motor command $\vec{m} = (\Delta\varphi, \Delta\theta)^\top$ that can be either used to control the real PTU, or used in combination with the visual forward model.

In order to use the saccade controller on digital images, we have to transform the target coordinates according to Equation (3.3). Furthermore, a distortion model can be easily incorporated: the target position has to be distorted to yield the corresponding undistorted coordinates which can then be used as inputs to the controller. The reason why we have to apply the distortion rather that the undistortion function is that the distortion is considered to be implemented as a reverse mapping.

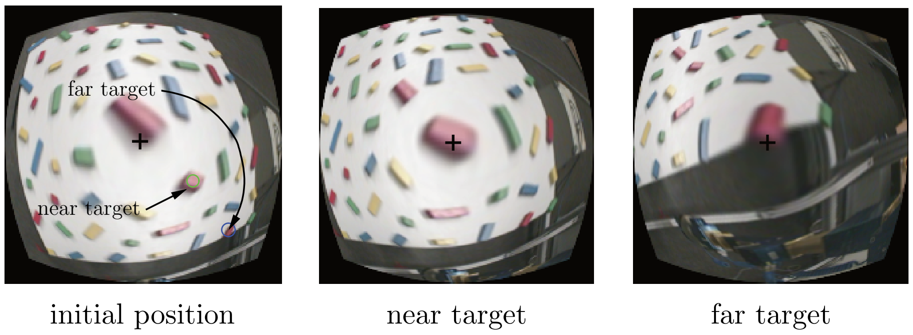


Figure 3.8.: Example for the geometric saccade controller. The two targets marked by a green and blue circles in the leftmost image are fixated: The near target can be fixated accurately (middle image); the far target is not fixated accurately (right image).

Figure 3.8 shows two example fixations using the geometric saccade controller with the real PTU. The leftmost image depicts the scene as seen from the initial viewing position. The target positions for the fixation are marked by green and blue circles. The saccadic motor command for the fixation of the near target is quite accurate (Figure 3.8, middle). However, the saccadic accuracy is low for the far target (Figure 3.8, right), i.e. the target object is not fixated accurately. Note that both targets were fixated from the initial viewing direction. This example suggests that the geometric saccade controller is too crude to generate accurate motor commands for far targets while it works rather accurately towards the image center. The reason for this behavior is that the geometric model is only an approximation of the real PTU since it neglects the influence of the current viewing direction.

In the following, we will investigate an extension of the geometric controller that is homogeneously accurate over the whole extents of the image. Instead of modifying the underlying geometric model, we augment the controller by an adaptive corrector module. The resulting model is closely related to neurophysiological findings in primates.

3.4.2. Adaptive Controller

The basis for the adaptive controller is a neurophysiological model of saccadic accuracy in primates (Dean et al., 1994). The model incorporates circuits located in various areas of the brain (Scudder et al., 2002). The most important role plays the *superior colliculs* which directly receives visual input from the retinae (Dean et al., 1994; Scudder et al., 2002) and encodes the target position in retinotopic coordinates. Based on these information, the superior colliculus generates appropriate saccadic motor commands. These motor commands (which are "place" encoded) are sent to the brain stem which is in charge of controlling the eye muscles (via motor commands that are "rate" encoded) (Dean et al., 1994; Fuchs et al., 1985; Scudder et al., 2002).

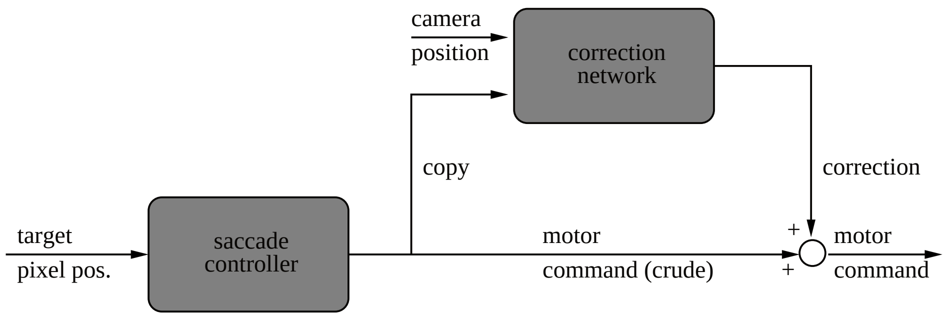


Figure 3.9.: Extended saccadic control scheme. The geometric saccade controller generates a (crude) motor command for a given target position. A copy of this motor command is sent to the adaptive corrector network. The resulting correction motor command is added to the crude command to yield the final (accurate) motor command.

Experiments show that these motor commands issued by the superior colliculus are rather crude (Dean et al., 1994). Therefore various various authors suggest that circuits located in the *cerebellum* are correcting these crude motor commands in order to achieve a higher degree of saccadic accuracy (Dean et al., 1994; Noda, 1991). The cerebellum does not directly receive visual input from the retinae. It rather recieves the output of the superior colliculs (as an effernce copy) and kinesthetic information on the viewing direction (Dean et al., 1994; Noda, 1991). The output of the cerebellar circuit is sent to the brain stem where it modulates the burst generator that controls the eye muscles (Dean et al., 1994; Scudder et al., 2002). The cerebellar circuit is adaptive in the sense that saccadic accuracy is learned during the infancy and maintained throughout the entire life (i.e. it adapts to changes due to the weakening of the eye muscles) (Dean et al., 1994).

Dean et al. (1994) suggest a computational model for the learning of saccadic accuracy based on these neurophysiological findings. This model is based on a P-type controller that generates crude saccadic motor commands that are corrected by an adaptive cerebellar mechanism. They successfully tested their model on a simulated robot. In the following, we will extend our geometric saccade controller based on a modified variant of Dean et al.'s (1994) model: instead of a P-type controller we employ the geometric saccade controller for the generation of crude motor commands that are subsequently corrected by an adaptive network.

Figure 3.9 depicts the components of the adaptive saccade controller. The geometric controller plays the role of the superior colliculus in converting retinal (target) positions to motor commands. In our model, we do not make a distinction between "place" and "rate" encoded motor commands. Here, we stick to the notion of abstract motor commands that are given by $\vec{m} = (\Delta\varphi, \Delta\theta)^\top$ and encode the relative change in viewing direction. The adaptive cerebellar circuit is represented by an artificial neural network that receives the crude motor command $\vec{m}$ and the current viewing direction $\vec{v}_0 = (\varphi_0, \theta_0)^\top$. The corrective motor command

$\vec{m}_c = (\Delta\varphi_c, \Delta\theta_c)^\top$ is already in the same representation as $\vec{m}$ and may thus be simply added. There is no need for a brain stem because of the high level of abstraction at which we consider the problem.

Furthermore, our model is not learning on-line. The learning and application phases are strictly decoupled. Therefore, we do not need to incorporate an error signal into the scheme. As a consequence of this simplification our model is able to learn, but not to maintain saccadic accuracy (which is only important if the oculomotor system undergoes morphological changes). This constraint has merely practical reasons, because the main aim of this chapter is to investigate models that can be readily used as building blocks for complex cognitive architectures and need a minimum effort in their implementation. Also note that on-line algorithms tend to converge at lower rates compared to their off-line counterparts. Furthermore, maintaining saccadic accuracy is not an issue because we can assume that the PTU does not undergo any morphological changes.

In the following, we will devise a proper learning strategy for the corrector network. The underlying assumption for the collection of the training data is that the geometric saccade controller is sufficiently accurate towards the image center. We will present a detailed description of the corrector network and its implementation. Finally, the collected training data will be used for a thorough error analysis of the geometric controller and to compare the results to those obtained by the adaptive saccade controller.

Data Collection

For the collection of training data we use the same database of images that we have already used for the training and evaluation of the adaptive visual forward model (see Chapter 2). The database was captured using the real PTU–camera assembly at a fine resolution. The scene that the robot is facing contains colored wooden bricks located on the surface of a table. These bricks make up excellent targets whose position within the image (center of gravity) can be determined with minimum effort.

We exclusively work on retinal images. This decision does not pose a restriction, because the corrector network does not directly receive sensory information and is thus invariant under the used distortion model.

The data collection process for the corrector network can be summarized as follows:

1. the PTU is moved to a random initial viewing direction $\varphi_{0\,i}, \theta_{0\,i}$ (Figure 3.10, far left)

2. the retinal camera image is color-segmented; the color is chosen at random from the set $\{\mathrm{red}, \mathrm{green}, \mathrm{blue}, \mathrm{yellow}\}$ (Figure 3.10, second from left, color blue)

3. a random segment is chosen as fixation target (purple frame)

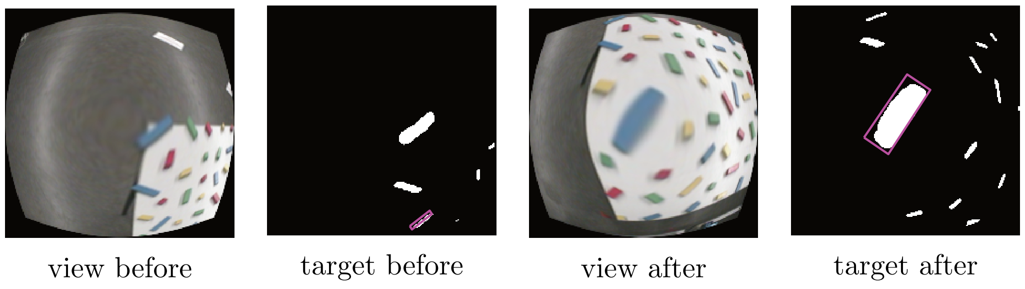


Figure 3.10.: Example of the data collection process. The panel shows the views of the scene before and after the saccade along with the corresponding segmented (binary) images. The target is highlighted by purple frames.

4. the (crude) fixation motor command $\Delta\varphi_i, \Delta\theta_i$ is computed using (3.5)
5. the fixation command is executed (Figure 3.10, second from right)
6. the post-saccadic view is segmented and the distance of the segment closest to the image center $e_i = \sqrt{(u_{Ii} - c_u)^2 + (v_{Ii} - c_v)^2}$, where (u_{Ii}, v_{Ii}) denotes the post-saccadic target position and (c_u, c_v) denotes the image center (both in retinal image coordinates), is calculated (Figure 3.10, far right)
7. the correction saccade $\Delta\varphi_{c\,i}, \Delta\theta_{c\,i}$ is calculated applying (3.5) to the post-saccadic target position

Note that we use the geometric saccade controller to calculate the correction saccade, thereby transforming the (off-centered) post-saccadic target position from sensory into motor space. This approach is based on the assumption that the geometric controller is sufficiently accurate towards the image center.

The above procedure is repeated $N = 10\,000$ times; the resulting training set is given by $\mathcal{T} = \{(\varphi_{0\,i}, \theta_{0\,i}, \Delta\varphi_i, \Delta\theta_i, e_i, \Delta\varphi_{c\,i}, \Delta\theta_{c\,i})\}_{i=1}^{N}$. The post-saccadic error e_i has been included for evaluation purposes only and will not be used in the application phase of the corrector network.

The training process is fairly simple and does neither require sophisticated image processing nor advanced learning strategies like direct inverse modeling (DIM) (Kuperstein, 1988). This is the great benefit of the crude geometric saccade controller that already provides a usable fixation of the target. Former attempts of learning saccade controllers from scratch (i.e. without using a crude controller) relied on target re-identification based on correlation-based measures (e.g. Schenck and Möller (2004)). Those correlation-based measures most likely fail on retinal images (Schenck, 2013). Therefore, Schenck (2013) proposed a learning strategy based on DIM in combination with a visual forward model for this task. Their approach is computationally demanding, albeit fully adaptive (i.e. it does not require prior knowledge on the geometry of the PTU or the image distortion model).

Network Structure and Training

In the following, we will give a detailed description on the structure and training of the corrector network. We chose a network of radial basis functions (RBF) for the interpolation of the training points $\mathcal{T}$. In the context of visual prediction we already examined that key features of RBF networks (see Chapter 2).

The input part of the network is given by vector $\vec{x} = (\varphi_0, \theta_0, \Delta\varphi, \Delta\theta)^\top$, where φ_0, θ_0 denote the angles of the initial viewing direction and $\Delta\varphi, \Delta\theta$ denote the crude saccadic motor command generated by the geometric controller. The network generates outputs, given by vector $\vec{y} = (e, \Delta\varphi_c, \Delta\theta_c)^\top$, where e denotes the post-saccadic error, and $\Delta\varphi_c, \Delta\theta_c$ denote the correction motor command. Note, that the post-saccadic error is only used for evaluation purposes and not necessary for the application of the corrector model within the saccade controller.

The complete network function is given by (Beatson et al., 2001):

$$\vec{f}_{\text{corr}} = \sum_{i=1}^{k} \vec{w}_i^{\text{out}} \left\| \begin{pmatrix} \varphi_0 \\ \theta_0 \\ \Delta\varphi \\ \Delta\theta \end{pmatrix} - \vec{w}_i^{\text{in}} \right\|_2 + \vec{a}_0 + \vec{a}_1\varphi_0 + \vec{a}_2\theta_0 + \vec{a}_3\Delta\varphi + \vec{a}_4\Delta\theta, \qquad (3.6)$$

where $\|\cdot\|_2$ denotes the (Euclidean) 2-norm, $\vec{w}_i^{\text{in}} \in \mathbb{R}^4$ denote the input weights, and $\vec{w}_i^{\text{out}}, \vec{a}_j \in \mathbb{R}^3$ denote the output weights and the polynomial coefficients, respectively. We chose $k = 100$ for all experiments.

The input weights were determined by running an on-line variant of the k-means algorithm (MacQueen, 1967) on the 4D input portion of $\mathcal{T}$, with $T_{\max} = 30\,000$ training steps. The output weights and polynomial coefficients were determined by solving the linear equation system (Beatson et al., 2001)

$$A^\top A\, W = A^\top Y, \qquad (3.7)$$

where

$$A = \begin{pmatrix} K & U^\top \\ V & 0_{5\times 5} \end{pmatrix}, (K)_{ij} = \|\vec{x}_i - \vec{w}_j^{\text{in}}\|_2,$$
$$U = \begin{bmatrix} 1 & \cdots & 1 \\ \vec{x}_1 & \cdots & \vec{x}_N \end{bmatrix}, V = \begin{bmatrix} 1 & \cdots & 1 \\ \vec{w}_1^{\text{in}} & \cdots & \vec{w}_k^{\text{in}} \end{bmatrix},$$
$$W = \left[\vec{w}_1^{\text{out}} \cdots \vec{w}_k^{\text{out}}\, \vec{a}_0 \cdots \vec{a}_4\right]^\top, Y = \left[\vec{y}_1 \cdots \vec{y}_N\, 0_{3\times 5}\right]^\top.$$

We used the QR-decomposition for solving (3.7) for W. Note that from the definition $A = QR$ it directly follows (from the orthonormality of Q) that $A^\top A = R^\top Q^\top QR = R^\top R$ (Golub and Van Loan, 1996), so (3.7) reduces to $RW = Q^\top Y$—which can be efficiently solved since R is triangular.

After the input and output weights have been determined, the corrector network (3.6) can be used as sketched in Figure 3.9. In the next section, we will analyze the spatial error distribution of the crude controller using the additional network output f_e. Furthermore, we will compare several results for the crude controller to corresponding results obtained using the adaptive controller.

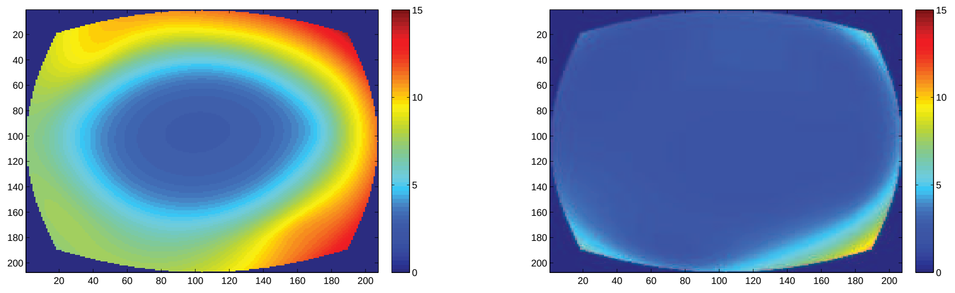


Figure 3.11.: Spatial error distribution over the extent of the retinal image for the geometric (left), and adaptive (right) saccade controller. The x- and y-axes are given in pixels; the blue borders correspond to the invalid regions of the retinal images.

3.4.3. Results

In the following, we will focus on the accuracy of both the geometric and the adaptive saccade controller. We will analyse two important aspects of the controllers, namely the spatial error distribution, and the overall saccadic accuracy. We will furthermore give a visual impression of the saccadic accuracy by reconstructing the spatial positions of the training targets from their corresponding viewing directions.

Spatial Error Distribution

The example saccades (Figure 3.8) executed by the crude controller suggest that it becomes less accurate towards the peripheral part of the image. This suggestion lead to the assumption that, in order to generate more accurate saccades, the crude geometric controller may be applied twice—to perform the initial crude saccade, followed by a small correction saccade (based on the post-saccadic target position). This application scheme has been used in the in the data collection process for the corrector network.

Now that we have a large amount of data, namely in form of the training set $\mathcal{T}$, we can perform a more thorough analysis of the accuracy of the crude controller. We will start by analyzing the spatial distribution of the post-saccadic error, i.e. the distance between the target's centroid and the image center after the initial (crude) saccade. In the previous section, we already mentioned that the corrector network has an extra output f_e for that purpose.

The starting point of the spatial image analysis is to define an arbitrary initial viewing direction φ_0, θ_0. We chose a viewing direction from which all targets are visible. In the next step, we generate the spatial error maps. For this, we iterate the whole retinal image and use the geometric model to generate a fixating motor command for each pixel position within the image. This motor command, along with the initial viewing direction, is used as input for f_e, the error part of the

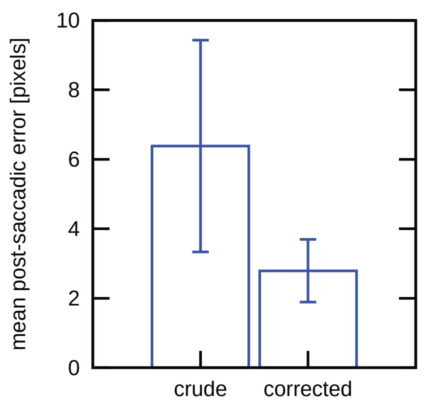


Figure 3.12.: Mean post-saccadic target distance and corresponding standard deviation (as error bars) for the geometric (left) and the adaptive (right) saccade controller.

corrector network, resulting in an interpolated error value for each pixel position. As a result, we get a spatial map of errors in which each pixel position corresponds to the post-saccadic error that would occur if this position is fixated.

Figure 3.11 (left) shows the result for the geometric controller: as expected, the error becomes gradually lower towards the image center. This supports our claim that the geometric controller performs accurately if the target is already located close to the image center, but the accuracy declines gradually towards the image periphery. The reason for this effect is that the geometric model is only an approximation of the true PTU–camera assembly that deviates from a perfect gimbal.

We repeated the data collection process (see Section 3.4.2) once again for the adaptive controller. This time, we collected $N = 1000$ training examples. Again, we fitted an RBF network (using the same parameters as before) to these data. The newly trained RBF network predicts the post-saccadic error for the corrected saccadic motor command. Figure 3.11 (right) shows the corresponding spatial error distribution. The overall error is low and distributed homogeneously throughout the whole image. The high peaks at the corners may be either due to a lack of training examples in these areas, or may be caused by noise in the segmented images.

Saccadic Accuracy

Based on the training set $\mathcal{T}$, we can easily determine the overall post-saccadic target error which is a good overall performance measure in the context of saccade control. In order to be consistent with the data collected using the adaptive controller, we have randomly drawn $M = 1000$ samples from the training set $\mathcal{T}$. Thus both samples have the same size.

We computed the mean post-saccadic error and the corresponding standard de-

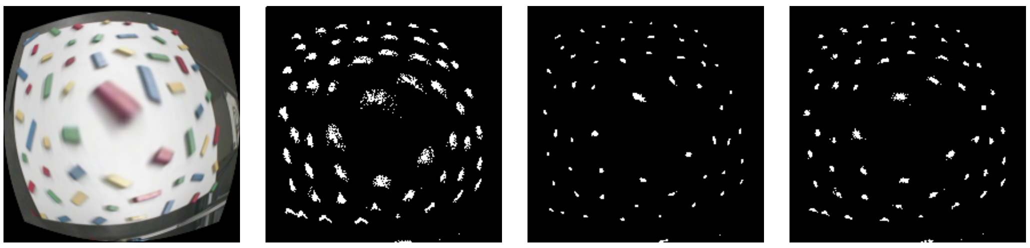

goal view
crude saccades
corrected saccades (training set)
corrected saccades (RBF network)

Figure 3.13.: Reconstruction of target positions from training set $\mathcal{T}$. The goal viewing direction depicting the real targets and various reconstructions are shown. See text for details.

viation. Figure 3.12 shows the results. As expected, the crude geometric controller (left) performs worse than its adaptively corrected variant (right). Both, mean post-saccadic error and the corresponding standard deviation is higher for the purely geometric controller.

Spatial Reconstruction of Target Positions

In the following, we will once more analyze the effect of correcting the saccadic motor commands generated by the geometric controller. In this experiment, we will employ the geometric visual forward model to reconstruct the spatial location of each target in the training set $\mathcal{T}$. The result shall serve as a visual impression of the increase in quality gained by the adaptive saccade controller.

The underlying assumption is that each target is characterized by a unique viewing direction φ_i^*, θ_i^*. This viewing direction can be easily computed by adding the initial viewing direction and the motor command for the fixation saccade: $\varphi_i^* = \varphi_{0\,i} + \Delta\varphi_i, \theta_i^* = \theta_{0\,i} + \Delta\theta_i$. Moving the PTU to this viewing direction would result in the corresponding target object appearing roughly in the image center.

Our goal is now to visualize as many object positions as possible. Therefore, we chose another viewing direction φ_1, θ_1, called the goal direction, that corresponds to a view showing as many targets as possible. Figure 3.13 (far left) shows the scene from the goal direction we have chosen for the purpose of reconstructing the target positions.

We now have the target viewing direction φ_i^*, θ_i^* and the goal viewing direction φ_1, θ_1, as well as the information that the target is roughly located in the image center c_u, c_v when gazed upon. This information is sufficient to compute an estimation of the pre-saccadic target position. We compute first compute the saccadic motor command from target to goal direction, given by

$$\begin{aligned}\Delta\varphi_i^* &= \varphi_i^* - \varphi_1 = \varphi_{0\,i} + \Delta\varphi_i - \varphi_1 \\ \Delta\theta_i^* &= \theta_i^* - \theta_1 = \theta_{0\,i} + \Delta\theta_i - \theta_1.\end{aligned} \tag{3.8}$$

Using $\Delta\varphi_i^*, \Delta\theta_i^*$ and c_u, c_v as inputs for the geometric visual forward model, we can now compute the pre-saccadic target position u_i^*, v_i^*. We expect that, if the saccade controller works perfectly, each target is represented by a single point. However, because of the low accuracy of the geometric visual forward model, each target is represented by a cloud of points (see Figure 3.13, second left).

If we employ use the correction motor commands from the training set, we get the following motor commands for the visual forward model:

$$\begin{aligned} \Delta\varphi_{c\,i}^* &= \varphi_{0\,i} + \Delta\varphi_i + \Delta\varphi_{c\,i} - \varphi_1 \\ \Delta\theta_{c\,i}^* &= \theta_{0\,i} + \Delta\theta_i + \Delta\theta_{c\,i} - \theta_1. \end{aligned} \tag{3.9}$$

As expected, the corrections tighten the point clouds drastically (Figure 3.13, second right). Instead of using the corrections from the training set, we can also apply the corrector network for this purpose: the results are shown in Figure 3.13 (far right). The variance of the point clouds is only slightly larger than for the corrected data from the training set. This effect may be due to the fact that the RBF network uses a layer of $k = 100$ internal units (in contrast to the $N = 10\,000$ training examples). Therefore, the network is regulated and is not able to exactly reproduce the training data. But regularization is also important for a network to generalize (i.e. make correct predictions for data points not included in the training set).

3.5. Conclusions

A geometric model of the camera–PTU assembly has been developed. The PTU is modelled as a perfectly spherical manipulator. The model resets on the assumption that the current viewing direction has no influence on a relative movement of the PTU (i.e. a change in viewing direction).

Based on this model we devised a visual forward model. The visual forward model predicts the optic flow within the camera image for a given motor command. Furthermore, we incorporated a distortion model to account for retinal distortions. The geometric forward model is therefore equivalent to the adaptive visual forward model described in Chapter 2.

However, experiments have shown that the geometric visual forward model performs worse than its adaptive variant. The reason for this may be due to the fact that the parameters of the geometric model are only approximations and the real PTU–camera assembly deviates from a perfect gimbal. Furthermore, the distortion model does account for the retinal mapping, but neglects the distortion underlying the camera images which are due to the lens system.

Although we did not compare the execution times of the geometric and adaptive models, we are certain that the geometric model is computationally much simpler. This assumption is based on the fact that the involved equations are much simpler. Therefore, the geometric forward model represents a computationally efficient alternative to the adaptive model.

We furthermore investigated a restricted inverse problem of visual prediction, namely saccade control. The problem is restricted in the sense that the saccade controller generates motor commands that always lead to the fixation of a specific target location. (The full inverse would require an arbitrary goal position as an additional input to the controller.)

The resulting geometric saccade controller is computationally simple and follows from straight trigonometry. Preliminary results on the PTU showed that the saccadic motor commands are rather crude—i.e. not resulting in accurate saccades. We therefore devised an extended controller, based on neurophysiogical findings, that results in accurate saccades over the whole extent of the image, i.e. for every possible target position.

The two internal models developed in this chapter represent computationally efficient alternatives to their more advanced adaptive counterparts. Yet both models perform in a satisfactory manner and can be used within complex cognitive architectures.

3.6. Outlook

The presented geometric model represents a coarse approximation of a real PTU neglecting several aspects of the kinematics. As a result of this, the model relies on a considerably lower number of parameters. Incorporating more parameters (like the offset between the pan and tilt axes) would theoretically increase the precision of the derived forward and inverse models.

However, putting too much prior knowledge into the model would strip it off of any biological plausibility. Therefore, most authors fall back to fully adaptive models that are based on universal approximators (like neural networks). These models rely on training data; the higher the input dimension and the more parameters the model is based on, the higher the number of training examples needed. Furthermore, depending on the problem, advanced learning schemes need to be used. In some cases, the training process is cut short by falling back to computational methods like collecting training points on a grid or by using the inverse kinematics. These short-cuts simplify the collection of training data, but the prior knowledge enters "through the back door".

The approach we followed in the context of saccade control is almost perfectly in line with findings from neurophysiology. An imperfect controller is supported by an adaptive corrector to generate near-perfect saccades. This scheme is easily applicable to the problem of saccade control, but would most like fail on the problem of visual prediction. In that case the corrector would have to learn the error for every pixel in an image which cannot be determined unless the correspondence problem can be solved in that case. Therefore, the learning would turn out computationally demanding.

The correction scheme for saccade control has also two main drawbacks. The main issue is that the corrector has been trained off-line. This can be easily overcome by training the corrector after every saccade by determining the post-saccadic target position and generating a purely covert (internal) fixation command (using the crude controller) that serves as target input to the corrector. The initialization of the corrector is crucial in this scenario since it should initially not decrease the performance of the controller any further. Another issue is the combination of the crude and the corrective motor commands. We used a simple summation because in our model, both commands are given in the same units (i.e. angles). In the biological model (Scudder et al., 2002; Dean et al., 1994), the two signals are coded differently and are integrated inside the brain stem. We can circumvent this problem by claiming that our model resides on a higher level of abstraction and justify its existence in the need for precise models for robot control.

4. An Associative Model for Mental Imagery[1]

4.1. Introduction

Mental imagery is the process of generating internal sensory experiences and impressions of motor activity without actual sensory inflow and without motor activity. Humans are obviously capable of recalling sensory experiences even in a willful manner. These sensory experiences may relate to changes in visual scenes caused by the execution of covert motor commands (e.g. walking through an environment) or may just be a recall of previously perceived sensations. Neuroimaging studies suggest that cortical areas which are involved in the processing of perceived stimuli are also active during mental imaging (Kosslyn et al., 1993). A similar finding suggests that motor areas are used for executed as well as covert actions (Jeannerod, 1995).

In this chapter we suggest an artificial neural architecture which enables an agent to generate mental views of parts of itself based on the values of a set of postural variables and motor commands. The agent might use these mental images to identify itself (or parts of itself) in its current view. This form of "*self-awareness*" becomes important when the agent starts interacting with objects or with other agents: The self-aware agent is able to discriminate portions of the visual input (e.g. pixels) that belong to the objects it interacts with from portions that belong to its own body.

In the following, we will consider a stationary agent which consists of an arm with an attached two-finger gripper and a stereo camera head. The mental imagery is restricted to views of the gripper based on (a) the arm posture and (b) the current gaze direction. Thus, we can model the generation of a mental image as a mapping from parameter space into image space—a process commonly termed *view synthesis* in the literature (Jägersand, 1997).

The field of view synthesis can be divided into two main branches; *image synthesis* methods are based on models of the scene whereas *morphing* approaches use one or more images of the scene which are morphed into a new view. A special case of the latter is *image warping* which transforms one input image based on a set of parameters (Glasbey and Mardia, 1998).

[1]This chapter is an extended version of the two conference papers *Mental Imagery in Artificial Agents* (Kaiser et al., 2010a) and *A Model Architecture for Mental Imagery* (Kaiser et al., 2011) which represent snapshots of the presented work at different stages.

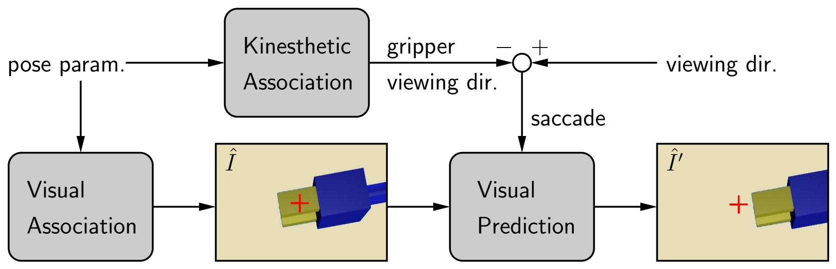


Figure 4.1.: Model architecture for mental imagery: views of the gripper are generated by *visual association* given an arm posture; these views appear as seen from a common gaze direction, the *gripper viewing direction* (image $\hat{I}$). After the *visual prediction* step, the view is in accordance with the desired viewing direction (image $\hat{I}'$).

Typically, image synthesis requires a precise geometrical model of the scene, e.g. a CAD model. While such a model can be acquired for artificial applications, it seems very unlikely that this form of representation is used in biological brains. Appearance-based view synthesis (Jägersand, 1997) does not require an explicit model; the model is rather "learned" directly from provided image data. The view (i.e. an image) is parameterized by an *appearance vector* whose dimensionality is much lower than the number of pixels in the original image. For our application, the objective is now to establish a mapping from postural parameters onto the appearance vector.

Image warping (Glasbey and Mardia, 1998) is a general term referring to a class of methods that involve the mapping of pixel positions in one image plane onto another. Schenck and Möller (2007) describe a *visual forward model* (see also Chapters 2 and 3) which takes as input the current view and predicts an image as it would appear after a given saccade, i.e. a change in gaze direction. Thus, the synthesized view only contains information already present in the input image, but "warped" according to the new gaze direction.

In this chapter we present a method that combines image synthesis with image warping. A robotic agent equipped with a stereo camera head and a serial manipulator with an attached two-finger gripper learns to associate a certain arm posture and gaze direction with the corresponding view of its gripper. Here, the image synthesis part—termed *visual associative model*—takes as inputs the joint angles of the manipulator and returns the corresponding image of the gripper. These images always appear as if the gaze is directed towards the gripper. To generate views of the gripper from arbitrary gaze directions, we employ a *visual forward model* which is used to warp the image of the current view according to a given saccade. The saccade is generated from the desired gaze direction and the gaze direction that would fixate the gripper. Thus we furthermore require a model—the *kinesthetic associative model*—which associates an arm posture with a gaze direction such that the gripper is fixated.

Figure 4.1 depicts a sketch of the overall model architecture. The kinesthetic pose parameters (defined by the joint angles of the manipulator) and a desired viewing direction constitute the input of the model. The pose parameters are used to drive the two sub-models for visual and kinesthetic association, generating a synthesized view of the gripper (fixated) and the corresponding viewing direction. The kinesthetic association can be conceived as a transformation from arm coordinates to head-centered coordinates. The gripper-centered and the desired viewing direction are combined to form a saccadic motor command. Finally, a visual forward model (see Chapters 2 and 3) is used to warp the synthesized gripper view according to the saccadic motor command. This approach enables the model to generate views of the gripper for a given joint angle configuration, viewed from an arbitrary direction. Recently, Schenck (2013) gave a brief description of the overall model (see Figure 4.1).

In the following, we will briefly describe the robotic agent and the necessary constraints of the arm postures. Furthermore, we will derive a learning scheme for the visual and kinesthetic association models. The visual forward model has already been described thoroughly in the last two chapters.

4.2. Robotic Agent

The robotic agent used throughout this study resembles roughly the upper torso of a human: it consists of a camera head with two cameras, each mounted on a pan-tilt unit (PTU), and a 6-degrees-of-freedom serial manipulator with an attached two-finger gripper. The cameras are mounted above the arm. The whole assembly faces a table which, in our experiments, only serves as a background.

4.2.1. Vergence Model

In Chapter 3 we have already studied the control of a single camera mounted on a pant-tilt unit (PTU). In this chapter, we will focus on the control of a binocular camera–PTU assembly which consists of two cameras, each mounted on an individual PTU. Instead of regarding both PTUs individually, we will employ a model in which the individual gaze directions are coupled.

We denote the individual gaze directions by $\vec{v}_{l/r} = (\varphi_{l/r}, \theta_{l/r})^\top$, where $\varphi_{l/r}, \theta_{l/r}$ denote the left/right pan and tilt angles, respectively. The so-called *vergence model* (Schenck, 2008) is a convenient way to describe the binocular viewing direction using the coupled parameters $\varphi, \theta, v_h, v_v$, where φ, θ denote the coupled pan and tilt angles, and v_h, v_v denote the horizontal and vertical vergence values, respectively. The vergence values are abstract quantities that are related to the horizontal / vertical vergence angles, i.e. the horizontal / vertical angles between the cameras. For the control of the PTUs, the transformation from coupled to individual parameters is given by (Schenck, 2008):

$$\begin{aligned} \varphi_{l/r} &= \frac{\varphi \pm 0.25\,(v_h + 1)}{1.5}, \\ \theta_{l/r} &= \frac{\theta \pm 0.2 v_v}{1.2}, \end{aligned} \tag{4.1}$$

Note, that Schenck (2008) requires the angles not to be given in radians but in normalized values in $[-1, 1]$. Following this notion, -1 corresponds to the minimum and 1 the maximum pan/tilt angles of the PTU. Thus, pan and tilt are scaled differently (as most PTUs have different ranges for pan and tilt).

The vergence model is biologically more plausible than coding the gaze directions of both PTUs independently (Mays, 1984). Furthermore, additional depth information is implicitly encoded in the horizontal vergence value: fixations in the near result in smaller horizontal vergence values (i.e. the cameras are rotated towards each other), while fixations in the distance result in greater values (i.e. the angle between the camera axes approaches $0°$).

4.2.2. Arm Postures

The agent is equipped with a robot manipulator with six rotatory degrees of freedom. The joint angles will be denoted by $\psi_1, \ldots, \psi_6$. The manipulator can be decoupled into arm (joints 1–3) and wrist (joints 4–6). Thus, its inverse kinematics (IK) can be calculated in closed form by kinematic decoupling (Spong and Vidyasagar, 2008). Note that this computation is just a short-cut which is required to speed up the collection of a large sample of training data. We will only use IK solutions that correspond to elbow-down configurations of the manipulator.

The manipulator is equipped with a two-finger gripper. The longitudinal axis of the gripper is always kept parallel to the ground plane while its orientation about the vertical axis is constrained to three different angles, i.e. $\alpha \in \{0°, 15°, 30°\}$.[2]

4.3. Kinesthetic Association

The kinesthetic associative model directs the agent's gaze towards the center of its gripper by associating an arm posture (i.e. a tuple of joint angles) with a gaze direction (i.e. pan, tilt, and horizontal/vertical vergence values). Thus, the model performs a transformation of arm-related to head-related coordinates.

The purpose of this model is two-fold: during the training of the visual associative model it is used to collect training images of the gripper for different arm postures. In the application phase, the corresponding arm posture is used to generate saccades which are then used to drive the visual prediction via the visual forward model.

The kinesthetic associative model is implemented by an adaptive neural network. We chose a multilayer perceptron (Rumelhart et al., 1986) with 6 inputs

[2]At an orientation of $0°$, the gripper is pointing straight away from the cameras.

(corresponding to the 6 joint angles), a hidden layer with 40 units, and 4 outputs (corresponding to the pan, tilt and the horizontal/vertical vergence values). Linear activation functions were used for the output layer and a sigmoid function ($\tanh(\cdot)$) for the hidden layer. The training data was collected by approaching points within a regular grid of end effector coordinates. The cameras were controlled by a *vergence controller* (Schenck, 2008) such that the gripper was fixated at every grid point.

We chose an adaptive solution to this problem, although we are aware that there exists a simple engineering solution based on camera calibration and the inverse kinematics of the camera head. However, we think that using an adaptive approach here is more appropriate for a biologically oriented model like ours.

In the following, we will first describe the vergence controller that plays an important role during the collection of training data. We will then outline the procedure for training data collection and describe the training of the neural network.

4.3.1. Vergence Control

The vergence controller is similar to the monocular saccade controller described in Chapter 3: its role is to direct the gaze of the cameras towards a salient object such that it appears in the center of the two camera images. We employ a feed-back controller for this task.[3]

In terms of control theory, the reference input corresponds to the coordinates of both image centers, the system inputs are the pan, tilt and vergence values, and the measured output is the centroid of the salient object in image coordinates (separately for both cameras). Thus, the measured error is the deviation between the coordinates of the image centers and the centroid in the left and right camera image.

We chose to use a simple P-type (proportional) feed-back controller, whose gain matrix has been determined heuristically. The time-discrete controller equation is given by (Schenck, 2008):

$$\underbrace{\begin{pmatrix} \Delta\varphi_t \\ \Delta\theta_t \\ \Delta v_{h\,t} \\ \Delta v_{v\,t} \end{pmatrix}}_{\Delta\vec{v}_t} = \underbrace{\begin{pmatrix} -\frac{1}{2} & 0 & -\frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & 0 & -\frac{1}{2} \\ \frac{1}{2} & 0 & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} \end{pmatrix}}_{G} \underbrace{\begin{pmatrix} \Delta x_{l\,t} \\ \Delta y_{l\,t} \\ \Delta x_{r\,t} \\ \Delta y_{r\,t} \end{pmatrix}}_{\vec{e}_t}, \qquad (4.2)$$
$$\vec{v}_{t+1} = \vec{v}_t + \Delta\vec{v}_t$$

where $\Delta\vec{v}_t$ denotes the change in gaze direction (i.e. the vergence motor command), G denotes the gain matrix, and $\vec{e}_t$ is the current error vector containing the deviations between the image centers and the centroids of the salient object for the left and right camera image, respectively. Note that the image coordinates are scaled

[3]Note that the adaptive saccade controller described in Chapter 3 had not yet not developed at the time this study was conducted.

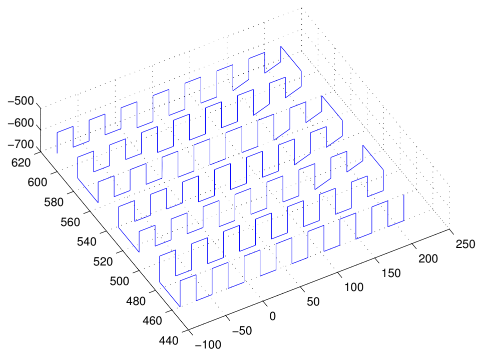


Figure 4.2.: 3D grid of gripper positions and trajectory of the gripper during the collection of training points. Figure only depicts the data for the gripper orientation of 0°. Axes are given in millimeters.

to be in the range $[-1, 1]$, therefore point $(0, 0)$ corresponds to the image center. Furthermore, $\vec{v}_t$ denotes the current viewing direction and $\vec{v}_{t+1}$ denotes the viewing direction after one control step.

The PTU is controlled iteratively (starting from an initial viewing direction $\vec{v}_0$) by converting the coupled viewing direction $\vec{v}_{t+1}$ into individual pan and tilt angles using Equation (4.1). The iteration is terminated if the errors dropped to at least 1 pixel in each direction. The terminating criterion was necessary in order to prevent oscillations of the controller.

Note that the iterative control scheme requires an image processing step at each time step t in order to determine the error vector $\vec{e}_t$. This is in contradiction to physiological findings; fixation movements are usually ballistic open-loop movements, i.e. they do not rely on visual feed-back during their execution. However, generating accurate fixation movements using a real camera–PTU assembly requires precise knowledge of the underlying inverse kinematics and the camera calibration. The vergence controller is an auxiliary module that is only needed for the collection of training data. Therefore, we decided to use the simple feed-back controller for this task.

4.3.2. Collection of Training Data

For the training we defined a rectangular workspace of size 150 mm × 120 mm × 300 mm that was sampled by a $7 \times 6 \times 15$ regular grid. Thus, the distance between adjacent grid positions is approximately 20 mm in each direction. The whole grid was sampled for each gripper orientation $\alpha \in \{0°, 15°, 30°\}$ separately. The six joint angles were calculated from the Cartesian coordinates using inverse kinematics (IK). From the 8 theoretically possible IK solutions of the given arm only those belonging to a specific family (i.e. elbow down) were selected. Furthermore, a collision detector was used in order to avoid collisions of the arm with itself or its environment; if a collision was detected the corresponding position was skipped.

During the training the gripper held a salient tracking target. We used a small styrofoam sphere attached to a fiberglass rod. The sphere was dyed red so that it could be easily detected in the camera images. The gripper held the target such that it appear above the fingers. The rod was placed approximately at the end of the gripper tips. During the training, the target was fixated, using the vergence controller, at every position within the spatial grid.

The procedure for data collection can be summarized as follows:

1. The inverse kinematics is evaluated for the grid position (x_i, y_i, z_i).
2. From the 8 possible joint angle configurations, the elbow-down configuration is chosen.
3. The collision detection is queried for the selected configuration.
4. If the configuration is collision-free, the arm is moved; else $i \leftarrow i + 1$; goto 1.
5. The gripper tip is fixated using the vergence controller.
6. The training example is added to the training set $\mathcal{T}_{\text{kin}} \leftarrow \mathcal{T}_{\text{kin}} \cup \{(\vec{\psi}_i, \vec{v}_i)\}$.
7. $i \leftarrow i + 1$; goto 1.

The procedure is repeated for all values of α separately. Due to the collision detection, the actual number of approached grid points deviates from the theoretical number (630) for the different orientations: 0° (583), 15° (483), 30° (335). The resulting training set $\mathcal{T}_{\text{kin}} = \{(\vec{\psi}_i, \vec{v}_i)\}_{i=1}^{N}$ contains pairs of joint angle configuration and corresponding viewing directions.

Figure 4.2 depicts the spatial training grid and the trajectory of the gripper for the orientation $\alpha = 0°$. Some of the grid points that are close to the table are not reachable without causing collisions and are thus excluded. This is mainly due to the constructional characteristics of the robotic manipulator. See Appendix B for the trajectories that correspond to all three gripper orientations.

4.3.3. Neural Network and Training

The task now is to fit an interpolating function to the training set $\mathcal{T}_{\text{kin}}$. We chose a multi-layer perceptron (MLP) for this task. The network function is given by

$$\begin{aligned} \vec{f}_{\text{kin}} &: \mathbb{R}^6 \to \mathbb{R}^4 \\ \vec{f}_{\text{kin}}\left(\vec{\psi}\right) &= \sum_{i=1}^{k} \vec{w}_i^{\text{out}} \tanh\left(\vec{\psi}^\top \vec{w}_i^{\text{in}} + b_i^{\text{in}}\right) + \vec{b}^{\text{out}}, \end{aligned} \tag{4.3}$$

where $\vec{w}_i^{\text{out}}, \vec{b}^{\text{out}} \in \mathbb{R}^4$ denote the output weights and biases, and $\vec{w}_i^{\text{in}} \in \mathbb{R}^6, b_i^{\text{in}}$ denote the input weights and biases. We chose a hidden layer of size $k = 40$.

The network was trained using the 1381 training samples from $\mathcal{T}_{\text{kin}}$. We used the off-line training method resilient propagation (RProp) (Riedmiller and Braun,

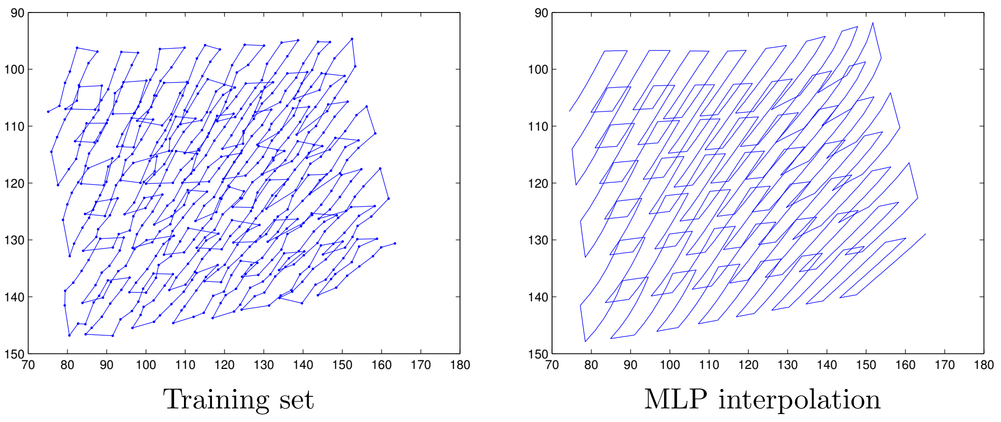


Figure 4.3.: Reconstruction of gripper positions for all arm poses in the training trajectory on the image plane based on the training set (left), and the trained MLP (right); x- and y-axes are given in pixels.

1993). In contrast to standard back-propagation (Rumelhart et al., 1986), RProp uses an heuristic to adjust the weight vectors (instead of using the gradient directly) which leads to faster convergence. Furthermore, the total number of patterns was divided into a training set (70%), and disjoint test and validation sets (both 15%). If the error on the validation set did not decrease for 200 epochs, the training was terminated (early stopping). The early stopping criterion should prevent the network from over fitting the data and yield a smooth regularization. Before the training, the weight vectors have been initialized with small values such that the sigmoid functions are not driven into saturation (LeCun et al., 1998).

In order to give a visual impression of the training set and the MLP smoothing, we project the viewing directions onto the image plane. A similar technique has been used in the context of the adaptive saccade controller in Chapter 3. The output part of the training set $\mathcal{T}_{\text{kin}}$ contains viewing directions that fixate the gripper at every grid point. Thus, the gripper is located in the image center. Theses viewing directions are now used to generate saccadic motor commands

$$\begin{aligned} \Delta\varphi^*_{l/r\,i} &= \varphi_{l/r\,i} - \varphi_1 \\ \Delta\theta^*_{l/r\,i} &= \theta_{l/r\,i} - \theta_1, \end{aligned} \tag{4.4}$$

where $\varphi_{l/r\,i}, \theta_{l/r\,i}$ denotes the individual viewing direction (calculated by the vergence model) of each PTU, and φ_1, θ_1 denotes the destination viewing direction. The destination viewing direction was chosen roughly to point towards the center of the workspace.

These saccadic motor commands, along with the coordinates of the image center were used as inputs for the geometric visual forward model (see Chapter 3) to predict the gripper position within the image as seen from the desired viewing direction φ_1, θ_1. Thus, we can project the gripper trajectory onto the image plane.

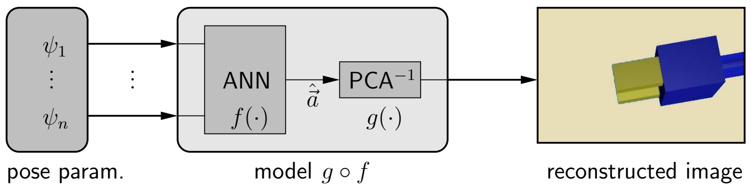


Figure 4.4.: Model architecture for visual association.

Figure 4.3 (left) shows the trajectory for $\alpha = 0$ as seen from the left camera. The plot was generated by transforming the 583 training points and connecting consecutive points by lines for better display. The training points appear unevenly distributed and the trajectory is rough. The jitter in the training set is most likely caused by inaccuracies of the vergence controller; the controller tolerates deviations of 1 pixel in each direction. Furthermore, the target is tracked by computing the centroid (or center of gravity) of the tracking target inside the images. This tracking is inaccurate due to noise within the camera images.

In order to compare the training set to the output of the trained model, we used 5830 test points that were generated by linear interpolation between consecutive training points (joint angle configurations) from $\mathcal{T}_{\text{kin}}$. The test points were transformed into corresponding viewing directions by applying the MLP $\vec{f}_{\text{kin}}$. The resulting viewing directions were then transformed into the image plane as mentioned above. Figure 4.3 (right) shows the result of the MLP interpolation; individual test points are omitted from the plot. The resulting trajectory is much smoother than the training set which is due to the regularization by the MLP. Furthermore, the MLP seems to generalize well, i.e. it produces a smooth trajectory based on the interpolated inputs.

By using this technique, we get a full visualization of the (high dimensional) training set: Figure 4.2 corresponds to the input and Figure 4.3 corresponds to the output part. A thorough visualization of the all gripper orientations and both PTUs (left and right) can be found in Appendix B.

4.4. Visual Association

We now turn to the problem of visual association. Generally, the task of visual association is to generate views of the agent (or specific parts thereof) based on a tuple of postural variables. In our case, the so-called visual associative model takes the posture variables—i.e. the joint angles $\vec{\psi}$—as input and returns the corresponding (fixated) view of the gripper. We will employ an adaptive approach for this problem in which the model is learned by presenting a set of example images.

Image data is usually high-dimensional (depending on the resolution of the images) which would require a large associative network. For this reason, we propose a two-stage architecture for this task (see Figure 4.4): the posture variables are first associated with low-dimensional appearance vectors which are then decoded to form the corresponding images. Thus, the problem boils down to the sub-problems of finding a low-dimensional representation of the images (in terms of appearance vectors), and that of learning an association between postural variables and the low-dimensional appearance vectors.

Seeking a low dimensional representation for the gripper images seems reasonable, because it is most likely that they reside inside a small sub-space of relatively low dimensionality within the image space. We use the well-known principal component analysis (PCA) (Jolliffe, 1986) to compute the low dimensional gripper space and for transforming the images into appearance vectors and vice versa. PCA is a linear—i.e. it is assumed that the data reside on a linear low-dimensional subspace—yet convenient method for dimensionality reduction. It has the great benefit that it provides a forward (encoding) as well as an inverse (decoding) function. Most recent techniques for dimensionality reduction only provide a forward function—i.e. from high-dimensional into low-dimensional space—which is extremely helpful for visualization purposes but only of little use for visual association. Here, we are mainly interested in reconstructing images from their low-dimensional representation.

The association between the joint angles and the appearance vectors is performed by a three-layer feed-forward network with linear output units and sigmoid hidden units akin to the network topology for kinesthetic association. After the data collection, the images are transformed into appearance vectors which served as targets for the network training. During the application phase, the images are reconstructed from the network output by applying the inverse PCA transformation to the network output, i.e. the estimated appearance vector for a given arm posture, see Figure 4.4.

In the following, we will describe the necessary steps for the derivation of the visual associative model. We will start by giving a brief description of an efficient method for the computation of the PCA of image data, the so-called *eigenfaces* (Turk and Pentland, 1991) or generally eigen-images approach. We will furthermore describe the procedure for data collection and the image processing steps, and finally we will take a look at the structure of the associative network and its training.

4.4.1. Eigen-Images

PCA is a procedure for computing a linear mapping from a high dimensional data space into space of lower dimension. In our case, the high dimensional space is the space of images. The main characteristic of the PCA mapping is that it retains the maximum possible variance for every projection direction. Furthermore, the projections are mutually uncorrelated.

In our case, the high-dimensional input space corresponds to the images we seek to associate with their corresponding postural variables, and the low-dimensional output space corresponds to the space of the appearance vectors. Let an image be denoted by vector $\vec{\imath} \in \mathbb{R}^M$, where $M = w \cdot h$ denotes its overall number of pixels, then its appearance vector is given by $\vec{a} = V_p^\top(\vec{\imath} - \bar{\vec{\imath}})$. Here $V_p \in \mathbb{R}^{M \times p}$ is a projection matrix with orthonormal columns and with $p \ll M$, and $\bar{\vec{\imath}}$ is the mean image. If V_p

is chosen such that the elements of $\vec{a}$ are mutually uncorrelated and the variance $\mathrm{Var}[a_i]$ is maximized, then the elements of $\vec{a}$ are the principal components of $\vec{\imath}$.

The classical approach to determine V_p is to compute the eigenvectors of the data covariance matrix, ordered by the magnitudes of their corresponding eigenvalues (Jolliffe, 1986). The p dominant eigenvectors, i.e. those that correspond to the p largest eigenvalues, are then arranged as columns into matrix V_p.

In the case of images, computing the $M \times M$ covariance matrix is prohibitive. Furthermore, we observe that the covariance matrix has only a maximum rank of N. Therefore, we can apply a computational trick that enables us to determine all eigenvectors V without computing the explicit covariance matrix. Note that there is a plethora of efficient approaches which are based on the adaptive learning of the principal eigenvectors. These approaches also circumvent the explicit calculation of the covariance matrix, and are mostly derived from Hebbian learning rules (e.g. Möller and Könies (2004); Oja (1982)). The adaptive methods rely on critical parameters, e.g. the learning rate and the number of time steps, which need to be carefully chosen. Therefore, we chose to use a classical numerical approach that is easier to handle.

Let $X = [\tilde{\vec{\imath}}_1 \dots \tilde{\vec{\imath}}_N]^\top \in \mathbb{R}^{N\times M}$ denote the data matrix of all N images, where $\tilde{\vec{\imath}}_k = \vec{\imath}_k - \bar{\vec{\imath}}$ denotes the k-th mean-centered image. Then the projection can be calculated by performing an eigen-decomposition of the $N \times N$ implicit covariance matrix $XX^\top = U\Lambda U^\top$, and computing $\tilde{V} = X^\top U$. Normalizing the columns of $\tilde{V}$ yields the desired V (Turk and Pentland, 1991).

Proof

We will prove this relation in terms of the singular value decomposition (SVD) of the data matrix X[4]. Let $X = USV^\top$ denote the "compact" SVD (Golub and Van Loan, 1996), where $U \in \mathbb{R}^{N\times N}$ is the matrix of left singular vectors, $V \in \mathbb{R}^{M\times N}$ is the matrix of right singular vectors, and $S = \mathrm{diag}(\sigma_1, \dots, \sigma_m) \in \mathbb{R}^{N\times N}$ contains the N non-zero singular values. The singular values are all positive ($\sigma_i > 0$) and ordered ($\sigma_i \geq \sigma_j \forall j > i$). Furthermore, the matrices U and V have orthonormal columns, such that $U^\top U = V^\top V = I$.

From the orthogonality of the singular vectors, it directly follows that

$$\begin{aligned} \check{C} &= XX^\top = US^2U^\top \\ (N-1)C &= X^\top X = VS^2V^\top, \end{aligned}$$

where $C \in \mathbb{R}^{M\times M}$ denotes the large rank-N explicit covariance matrix and $\check{C}$ denotes the significantly smaller $N \times N$ implicit covariance matrix. The above equations are directly related to the eigen-decompositions of C and $\check{C}$: U are the eigenvectors of $\check{C}$ and V the eigenvectors of C. Furthermore $S^2 = \Lambda$ are the corresponding eigenvalues.

[4]Another proof is given by Turk and Pentland (1991).

Using these relations, we can easily prove the eigenfaces equation:

$$X^\top U = VSU^\top U = VS = \tilde{V}.$$

We see that the resulting matrix $\tilde{V}$ corresponds to the right singular vectors of X (i.e. the eigenvectors of C), scaled by their corresponding singular values. It furthermore follows that the desired matrix V can thus be calculated by normalizing the columns of $\tilde{V}$ as claimed above.

Compression and Reconstruction

The eigen-image approach represents a method for the lossy compression of images. We assume that the images lie on a low dimensional manifold within the high dimension image space. This assumption only holds if the images share common characteristics like images of faces, handwritten digits—or in our case grippers. In this case, we only need to calculate a small number of the eigenvectors of $\check{C}$, namely those that correspond to the p largest eigenvalues.

The resulting appearance vectors are of dimension p and can be determined by multiplying the vectorized mean-centered image $\vec{\imath}$ by $V_p^\top$, the matrix containing the first p eigenvectors:

$$\vec{a} = V_p^\top \left(\vec{\imath} - \bar{\vec{\imath}}\right). \tag{4.5}$$

The approximate inverse transformation is given by a linear combination of the eigen-images:

$$\begin{aligned} \hat{\vec{\imath}} &= \sum_{i=1}^{p} a_i \, \vec{v}_i + \bar{\vec{\imath}} \\ &= V_p \vec{a} + \bar{\vec{\imath}}. \end{aligned} \tag{4.6}$$

Equation (4.6) is basically a two-layer feed-forward network with linear activations. The weights are determined numerically, based on the eigen-decomposition of the covariance matrix, rather than a learning rule. Furthermore, the reconstruction error is a quantitative measure for the compression quality. A low reconstruction error indicates a good choice of p.

The reconstruction error of a single image $\vec{\imath}$ is defined by

$$\begin{aligned} E_{\text{reco}} &= \left\| \hat{\vec{\imath}} - \vec{\imath} \right\|^2 \\ &= \left\| V_p V_p^\top \left(\vec{\imath} - \bar{\vec{\imath}}\right) + \bar{\vec{\imath}} - \vec{\imath} \right\|^2 . \end{aligned} \tag{4.7}$$

The compression formula (4.5) is only of significance for the training of the visual associative model. It will be used during the training to obtain the appearance vectors from the actual views of the gripper. The reconstruction formula (4.6) however is a crucial building block of the final model: In Figure 4.4 it is depicted as PCA^{-1} and will be used to synthesize the fixated views of the gripper.

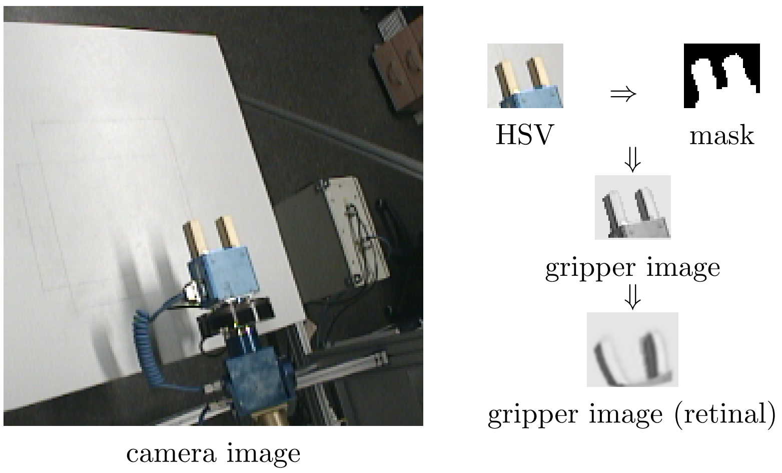


Figure 4.5.: Example for the collection of training examples. The gripper is located in the center of the camera image (left). A small region around the center is cut out and separated from the background (right). The resulting gripper image is warped by the retinal mapping to yield the final training image (bottom right).

4.4.2. Data Collection and Image Processing

In the following, we will outline the data collection procedure of the visual associative model and the image processing steps that were carried out prior to the computation of the eigen-images. The image processing is very crucial, because in order to compute the eigen-images, we need to segregate the gripper precisely from the background in the recorded camera images.

The data collection is conducted in an analogous fashion to Section 4.3.2. This time, the gripper fingers are in an open position as it would be before grasping. We use the same spatial grid of positions as for the kinesthetic associative model. For each grid point, the inverse kinematics is used to calculate the corresponding joint angles and the arm is moved to this position. The previously trained kinesthetic associative model then generates a gaze direction such that both cameras fixate the gripper. The PTUs are moved and the camera images are stored.[5]

The original 240×240 camera images were cropped to a small 45×35 region around the image center where the gripper is located. The gripper is then extracted from the background based on color information. We convert the images into the HSV color space for this purpose: the gripper moves above a white table which has low saturation values in HSV space. Therefore, we can generate "gripper masks" by applying a threshold operation to the images. The resulting masks were used to extract the gripper. The resulting gripper images are converted to gray-scale images and warped by the retinal mapping (see Chapters 2 and 3). Thus, the resulting images have a resolution that is higher in the central part of the image

[5]For the results presented in this chapter, only the left camera is used.

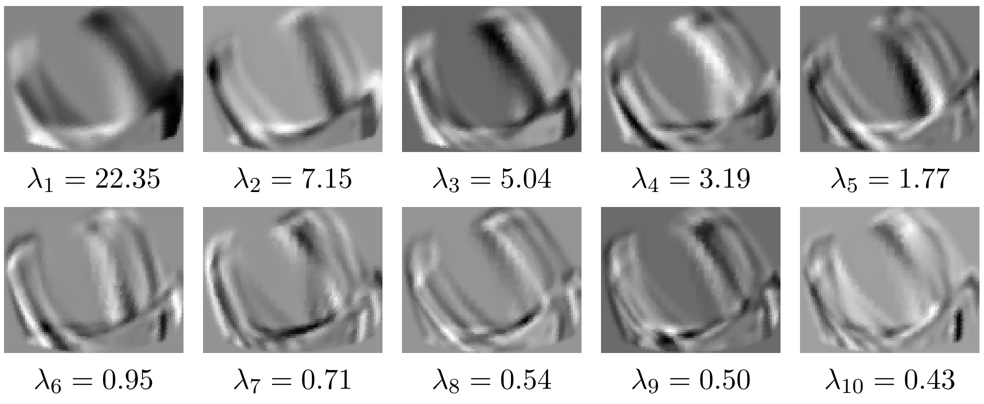


Figure 4.6.: The 10 principal eigen-images and the corresponding eigenvalues. Note that the pixel values were scaled to be in the range $[0, 1]$.

and lower towards the periphery (fovea effect). Figure 4.5 depicts the essential image processing steps.

The resulting retinal images are of size 85×69 pixels; their dimensionality is thus $M = 5865$. We repeated the above procedure for all three gripper orientations resulting in a total number of $N = 1401$ training images. These training images are used to compute the corresponding appearance vectors which will be used as training examples for the visual associative network.

4.4.3. Appearance Vectors and Network Training

The appearance vectors are computed as described in Section 4.4.1. Here we use a number of $p = 10$ eigen-images. The average reconstruction error, i.e. the deviation between the original images and their reconstructions, amounts to $<E_{\text{reco}}>_M = 3.47$. Figure 4.6 shows the 10 principal eigen-gripper images alongside their corresponding eigenvalues. The eigenvalues decay nearly exponentially; the 10th eigenvalue is already around 98% smaller than the first one. This indicates, that the first few eigenvectors capture most of the variance which in turn implies that only a small number of them is needed to reconstruct the gripper images reasonably well.

The visual associative network has a similar structure as the kinesthetic associative model (i.e. a 3-layer topology). We chose a relatively large hidden layer, consisting of $l = 100$ sigmoid units. A training pattern is a pair $(\vec{\psi}_i, \vec{a}_i)$, consisting of an arm posture $\vec{\psi}_i \in \mathbb{R}^6$ and the appearance vector of the corresponding gripper view $\vec{a}_i \in \mathbb{R}^{10}$; the total number of such patterns is $N = 1401$. For the training we use the RProp algorithm with early stopping.

The network function for the visual association looks similar to Equation (4.3), albeit the output weights are of the dimension of the appearance vectors. We can furthermore incorporate the PCA reconstruction into the network. As we already noted, Equation (4.6) resembles a two-layer feed-forward network with linear activations. Appending the linear layer of (4.6) to the associative network is equal to

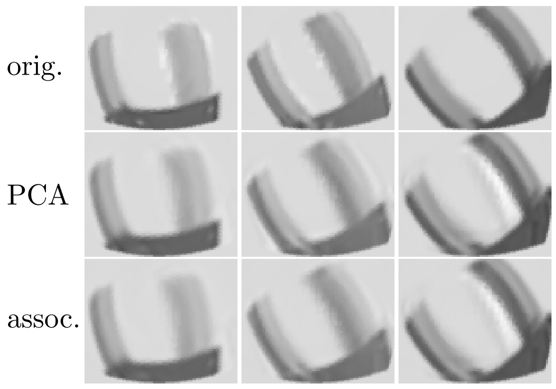


Figure 4.7.: Example images from three different grid positions. Original training images (top row), PCA reconstruction (middle row), and output of the associative model (bottom row).

a linear transformation of the network equation, yielding:

$$\begin{aligned}
&\vec{f}_{\text{vis}} : \mathbb{R}^6 \to \mathbb{R}^M \\
\vec{f}_{\text{vis}}(\vec{\psi}) &= V_p \left(\sum_{i=1}^{l} \vec{q}_i^{\,\text{out}} \tanh\left(\vec{\psi}^\top \vec{q}_i^{\,\text{in}} + c_i^{\text{in}}\right) + \vec{c}^{\,\text{out}} \right) + \bar{\vec{\imath}} \\
&= \sum_{i=1}^{l} \underbrace{V_p \vec{q}_i^{\,\text{out}}}_{\vec{r}_i^{\,\text{out}}} \tanh\left(\vec{\psi}^\top \vec{q}_i^{\,\text{in}} + c_i^{\text{in}}\right) + \underbrace{V_p \vec{c}^{\,\text{out}} + \bar{\vec{\imath}}}_{\vec{d}^{\,\text{out}}} \qquad (4.8) \\
&= \sum_{i=1}^{l} \vec{r}_i^{\,\text{out}} \tanh\left(\vec{\psi}^\top \vec{q}_i^{\,\text{in}} + c_i^{\text{in}}\right) + \vec{d}^{\,\text{out}},
\end{aligned}$$

where $\vec{q}_i^{\,\text{out}}, \vec{c}^{\,\text{out}} \in \mathbb{R}^p$ denote the output weights and biases, $\vec{q}_i^{\,\text{in}} \in \mathbb{R}^6, c_i^{\text{in}}$ denote the input weights and biases, V_p and $\bar{\vec{\imath}}$ are defined as in Section 4.4.1, and $\vec{r}_i^{\,\text{out}}, \vec{d}^{\,\text{out}} \in \mathbb{R}^M$ denote the combined output weights and biases.

Equation (4.8) encompasses the whole visual association (light gray box in Figure 4.4) in one function that maps from arm posture-space into the space of gripper images. This representation is especially useful for the application phase of the visual associative model. However, during the training phase, the two networks (i.e. appearance association and PCA reconstruction) need to be separated according to Section 4.4.1 and the training method mentioned above.

We will now turn to the reconstruction capabilities of the visual associative model (4.8). We only provide a qualitative comparison based on a few example reconstructions. Figure 4.7 shows the original views of the gripper (upper row) for 3 different arm postures and orientations. The reconstruction results in certain artifacts (e.g. most notably the white “shadow” underneath the right finger, rightmost column) which are present in the PCA reconstructions (middle row) as well as the outputs

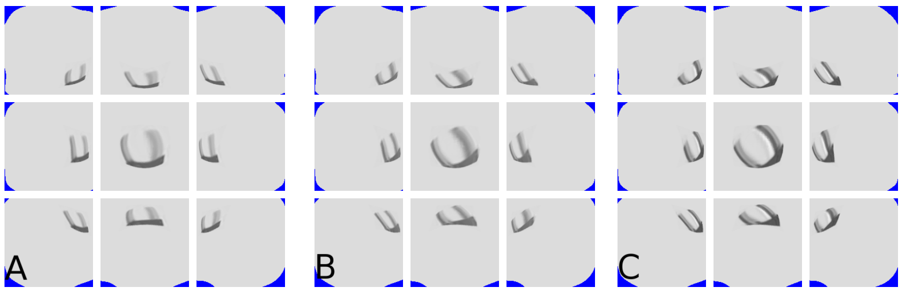


Figure 4.8.: Example outputs from the overall model for three different arm postures (A, B, C) and various viewing directions (see text for details).

of the associative model (bottom row). In general, the reconstructions from the associative model (bottom row) do not differ noticeably from the PCA reconstruction (middle row).

4.5. Results

In the following, we will demonstrate the capabilities of the overall model (see Figure 4.1). A quantitative measure of its performance is difficult to define and most probably meaningless. Therefore, we will focus on a few exemplary synthesized views in order to give a visual impression of the overall visual associative model.

We use the associated images of three arm postures (see Figure 4.7, bottom row) to demonstrate the overall model in the following. The arm postures were selected such that each of the three angles $(0°, 15°, 30°)$ is represented once. Note that the angle space is sampled too coarsely to allow for generalization, e.g. the model would not be able to generate an image corresponding to an angle of 22.5°.

The three different images were fed into the visual forward model (see A, B, and C in Figure 4.8). For each image, we used the visual forward model to predict the result of 9 different saccades (i.e. roughly 10° in each direction); the images in each center of Figures 4.8 A, B, and C correspond to the output of the visual association before a saccade.

It can be seen from Figure 4.8 that the resulting images are smooth and consistent, i.e. the images are not noticeably distorted. The borders in most of the outer images is marked as not predictable by the validator model (see Chapter 2). This is due to the fact that the input image does not contain any pixel information for these regions.

We furthermore moved the real robot arm along a trajectory and recorded a camera image at each via-point. The trajectory comprises 13 via-points. In contrast to the training process, the arm is not fixated in this example. We applied the associative model to generate corresponding simulated views for each posture (at each via-point). Figure C.1 in Appendix C shows the real views of the gripper (right column) alongside the corresponding simulated views (middle column). In order to give a better impression of the quality of the prediction we overlaid both images (right column): the image of the original gripper is used as red channel, and the predicted gripper image as green channel. The predicted gripper images match the real views of the gripper rather good. (In some images, however, the gripper fingers are further apart than in the real images.) The positions of the predicted gripper match the positions of the real gripper accurately. Therefore, we can conclude that, by visual inspection, the predicted gripper images are a good approximation of the real gripper images.

4.6. Conclusions & Outlook

We approached the problem of generating internal visual sensory states by decomposing it into the visual association of images of a gripper based on a set of pose parameters and the prediction of views according to a specific gaze direction. The visual association is performed by a model-free approach to view synthesis. In this approach, an image is reconstructed from its appearance vector (i.e. a dimensionality-reduced representation of the image) which is associated with a set of corresponding pose parameters by a neural network. The views generated by the visual association always appear as if the gripper is fixated. Therefore, we employ a visual forward model to warp these images according to an arbitrary gaze direction.

The architecture is fully adaptive since it is based on artificial neural networks and subspace methods from pattern recognition. We presented some examples which suggest that the association capabilities result in consistent images (i.e. resembling real views of the gripper). Nevertheless, this has still to be analyzed quantitatively. Furthermore, we expect that, in its present form, the model will not be able to generalize over the angle space. For the present study we used a very coarse grid of 3 different angles. Using more angles leads to a higher variability in the training images and thus more eigen-images would be required, and consequently a larger associative network. One possible solution would be to replace the (linear) PCA by a non-linear technique for extracting the appearance vectors such as local PCA (Möller and Hoffmann, 2004) or deep autoencoders (Hinton and Salakhutdinov, 2006).

Several technical shortcuts were used for the experiments in this study, e.g. the number of exploration trials was reduced by defining a regular grid of spatial positions, and the neural networks were trained off-line. These shortcuts are problematic from a modeling perspective; in a more realistic setting, the agent would learn its sensory-motor associations on-line while performing exploration movements. Furthermore, we assumed a priori knowledge about the characteristics of the gripper and performed a color-based segmentation of the camera images (i.e. sensory input) in order to extract the gripper images.

In contrast, Philipona et al. (2003) propose a scenario in the context of sensorimotor contingencies in which simple agents learn internal representations of “physical space” with as little a priori knowledge as possible. This is done by sending random motor commands to the agent's actuators and measuring the correlation between (the unlabeled) sensory and proprioceptive inputs. Following a similar route, we could improve our approach regarding gripper identification by performing small gripper movements around a certain (possibly random) position which is fixated by the camera head. If we assume that the environment is static, all changes in the sensory input would be due to gripper movements. Thus, portions that belong to the agent (i.e. the gripper) could be clearly separated from irrelevant content (i.e. background) without a priori knowledge.

The presented model is intended to be used as a building-block of more complex architectures which rely on the processing of sensorimotor (e.g. visuomotor) data. Consider a scenario in which the agent interacts with different objects. The objective is now to predict a possible displacement of the object (in the visual domain) based on the motor commands carried out by the agent. This can be achieved by detecting changes in the visual input. The movement of the agent's manipulators during the interaction also leads to changes in the visual scene. These changes can be predicted by the proposed architecture and subsequently canceled out. In the context of simulation theory (see Chapter 1), the proposed model could be used to generate a multi-step prediction in which the gripper and the objects can be distinguished from each other and interactions between them predicted in the visual domain.

5. Stereo Matching by Internal Simulation[1]

5.1. Introduction

In this chapter, we will take a glimpse on how to employ internal models for solving the correspondence problem in stereoscopic vision. Solving the correspondence problem—i.e. establishing matches between homologous points in a pair of images—is inevitable for visual depth perception. Judging the depth of a scene on the other hand is an important prerequisite for object manipulation. In the following, we are going to review neurally inspired approaches towards the problem of stereopsis. The first class of approaches we are going to review rely on sensory information alone; the second class makes use of active vision, i.e. by employing movable camera heads.

Most computational approaches to stereo matching address this problem from a purely sensory perspective (Marr and Poggio, 1979; Yokono and Poggio, 2004). The neural approach by Marr and Poggio (1979) is an early attempt a formulating a computational model for stereopsis. Their approach is based on the identification of zero-crossings in difference of Gaussians (DoG) type filtered images. The method yields impressive results on random-dot stereograms (Julesz, 1971). Yokono and Poggio (2004) present a local descriptor-based approach for object detection. They employ local responses of steerable filters, i.e. Gaussian derivatives with spatial orientation selectivity, to construct the descriptor. Their experiments suggest that their method is able to cope with cluttered scenes and partial occlusions.

Methods based on active vision use movable cameras for the fixation and depth estimation (Theimer and Mallot, 1994; Chumerin et al., 2010; Bernardino and Santos-Victor, 1996). The approach by Theimer and Mallot (1994) uses a filter bank of spatially distributed oriented filters. The correspondences are calculated without an explicit searching phase which is inevitable in feature-based methods. Their method uses multiple steps to calculate a dense disparity map of a scene using steropsis and active vision: In a first step, a coarse disparity map is calculated; based on this initial estimate, the vergence angle of the cameras is controlled such that the global disparity is minimized (e.g. the average depth is fixated); finally, the dense disparity map on a finer scale is computed. Chumerin et al. (2010) present a distributed neural model for vergence control. In their model, the disparity is computed implicitly using a convolution network modeling complex cells. Bernardino

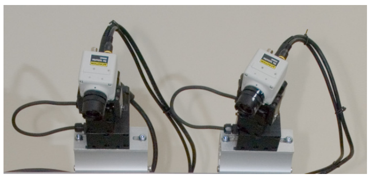
[1]This chapter is derived from *Solving the Correspondence Problem in Stereo Vision by Internal Simulation* (Kaiser et al., 2013) which in turn is based on the extended abstract *Stereo Matching and Depth Perception by Visual Prediction* (Kaiser et al., 2012).

and Santos-Victor (1996) propose a vergence control scheme using log-polar images. Log-polar images are characterized by a non-uniform sampling grid which is more dense towards the center of the image and sparse in the periphery, akin to the photoreceptor distribution on the human retina. The use of log-polar instead of Cartesian has certain computational benefits e.g. shape invariance towards scaling and rotation as being the most prominent. The control architecture by Bernardino and Santos-Victor (1996) is based on the normalized cross-correlation (NCC) measure for image matching and a PID controller for motor control.

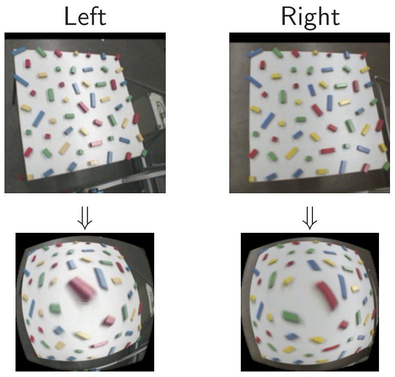
We present a novel approach based on simulated active vision that is entirely based on sensorimotor simulation. Our approach is based on two internal models: a visual forward model (Schenck and Möller, 2007; Schenck, 2008), see also Chapters 2 and 3, that predicts the influence of camera movements onto the current visual input, and a saccade controller (Schenck, 2008), see also Chapter 3, that generates fixation movements, given the location of an object within the camera image. These models, which are learned during a preceding exploration phase, are used to generate "canonical" (i.e. foveated) views of a specific object in one camera image and its potential partners in the other camera image. The canonical views are then compared using a difference-based measure. Thus, the internal models only generate internal (covert) states that do not result in any actions of the robot. Therefore we claim that an instance of internal simulation is taking place. Furthermore, the internal models allow for a image representation that is invariant under the current viewing direction. This is a crucial feature which allows our matching approach to cope even with severely distorted images.

In most robotic studies, internal models serve the preparation of actions or the prediction of expected perception for motor control (Datteri et al., 2003; Jamone et al., 2012; Saegusa et al., 2008). Datteri et al. (2003) describe a control scheme for a robot manipulator based on "expected perception". Their "expected perception generator" is basically what we would term a visual forward model: It predicts the visual consequences of linear arm movements along the x-axis of the images. As the robot arm moves, the expected perception is compared with the current sensory state; if a disassociation occurs, the robot arm is stopped. The visual forward model is restricted to linear movements along one image axis and is thus computationally very simple. Jamone et al. (2012) present an online learning scheme for reaching behavior in a 22-DOF humanoid robot. Their architecture uses a gaze controller for the fixation of target objects and, more importantly, a visuomotor inverse internal model for grasping. The former is implemented as a proportional controller while the latter is learned using fast local on-line techniques. The visual information (target location in both cameras) is at first used to generate a fixation movement which is then converted (by the inverse model) into a grasping command for a robotic arm. Thus, the 3D position of the object is implicitly reconstructed from the gaze direction which. Furthermore, the whole model is bootstrapped without any a priori knowledge.

In our architecture, the central part is a visual forward model which is able to predict the effect of changes in gaze direction on the image content. For the



(a)



(b)

Figure 5.1.: (a) Stereo camera–PTU setup. (b) Left and right camera image for a given gaze direction (top), left and right retinal images (bottom). Note the non-linear image warping due to the radial foveal mapping.

present study, we use "retinal" images rather than ordinary camera images. These retinal images are distorted such that the resolution is high in the center and low in the periphery. This distortion closely resembles the cone distribution on the human retina. It should be noted that the visual forward model is learned and thus able to cope with any type of lens distortion. "Classic" matching approaches like SIFT (Lowe, 2004) fail on these retinal images because the slightly different viewing direction of the two cameras has a strong effect on the visual representation of the scene. We show that the simulation-based matching approach copes well with this fact which is due to its inherent invariance under the current viewing direction. However, we included tests on the SIFT algorithm using undistorted (undistorted) images as to give an idea of its performance under comparable conditions. We are aware that there exist variants of SIFT that are able to deal with certain types of lens distortion, see e.g. Lourenço et al. (2012). These methods are often parametric and thus only applicable to a specific type of distortion, e.g. radial distortion (Barreto et al., 2009). The retinal mapping we use for our experiments is different from the distortion model described in Barreto et al. (2009), therefore the sRD-SIFT method described in Lourenço et al. (2012) is not applicable to this type of image.

5.2. Setup

5.2.1. Robotic Agent

The robotic agent used for our study consists of a pair of stereo cameras mounted on individual pan-tilt units (PTUs). The PTUs allow the cameras' gaze direction to be arbitrarily varied. The camera–PTU setup is looking downward onto a table.

The gaze directions of the cameras are given by the angle pairs (φ_l, θ_l) and

(φ_r, θ_r), respectively. Camera movements, which will be called saccades[2] in the following, can thus be expressed as relative changes $\Delta\varphi, \Delta\theta$. The PTU is constructed such that the ray going through the camera's entrance pupil approximately lies in the intersection of the pan and tilt axes. This property is important because it ensures that the change of the visual content in the camera images (induced by camera movements) only depends on the relative changes but not on the current angles. This is an important prerequisite for visual prediction, because in this case the depth information is irrelevant (Schenck and Möller, 2007; Schenck, 2008). For a detailed description of the kinematics of the PTU see Schenck (2008).

5.2.2. Retinal Images

Our model operates on retinal images rather than the planar images recorded by the cameras. Retinal images coarsely resemble the effect of the human retina, i.e. the image resolution is higher towards the image center and low in the periphery. The images acquired by the stereo camera system are processed by a so-called radial foveal mapping, which yields these retinal images. Using retinal images makes it virtually impossible for non-parametric descriptor-based approaches (i.e. without knowledge of the distortion model) to find correspondences in a pair of stereo images due to the strongly non-linear warping.

The radial foveal mapping is modeled by the following equations (in normalized polar coordinates, i.e. the radial coordinate is normalized to the range $[0, 1]$):

$$\begin{aligned} \phi_{\text{in}} &= \phi_{\text{out}} \\ r_{\text{in}} &= (1 - \lambda) \cdot r_{\text{out}} + \lambda \cdot r_{\text{out}}^{\gamma}, \end{aligned} \tag{5.1}$$

where $(\phi_{\text{in}}, r_{\text{in}})$, $(\phi_{\text{out}}, r_{\text{out}})$ denote the polar coordinates in the input and output image, respectively, and $\gamma = 2.5$ and $\lambda = 0.8$.

Figure 5.1b (top) shows an example pair of stereo images. The recorded images are of size 240×240. The retinal images (Figure 5.1b (bottom)), which are the result of warping the left and right camera images using (5.1), exhibit a strong fish eye effect. These images are of size 207×207.

5.3. Visual forward model

The visual forward model predicts the visual effect of camera movements onto the actual camera image without executing these movements. Therefore, the visual forward model can be seen as warping function that is parameterized by the camera movement $(\Delta\varphi, \Delta\theta)$. By employing a warping function for visual prediction, only those portions can be faithfully predicted that are already present in the input image. The other portion—i.e. those for which no content is present—are marked as invalid.

[2]Saccades are ballistic open-loop eye movements which can reach an angular speed of up to $900°/s$ in humans.

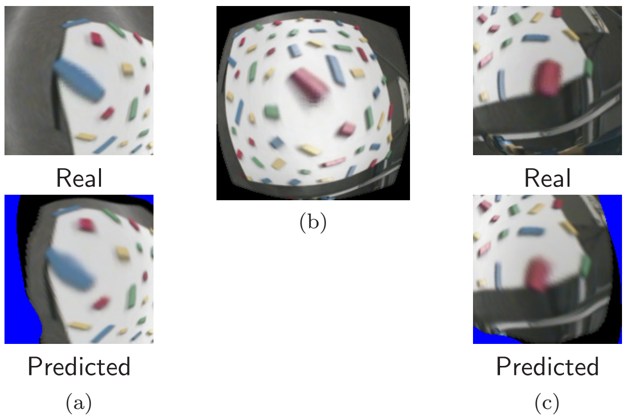


Figure 5.2.: Example for the visual forward model. The input for the visual forward model is shown in (b). A saccade to the left side is shown in (a); (c) shows a saccade to the lower right. The real post-saccadic images are shown for comparison to the output of the VFM.

The proposed visual forward model has thus a two-fold structure (see Schenck and Möller (2007), and Chapter 2): It (i) consists of a mapping model that, given a certain movement, warps the actual input image using reverse mapping, and (ii) a validator model that classifies pixels in the output according to their validity, given the movement. In mathematical terms, these models can be described by two functions $\vec{f_m}$ and f_v that depend on (u, v), i.e. the current pixel in the output image and $(\Delta\varphi, \Delta\theta)$. Note that $\vec{f_m}$ is vector-valued, since it estimates $(\hat{x}', \hat{y}')$, i.e. the position of (u, v) in the input image.

Calculating a mapping from the output image to the input image (reverse mapping) has the advantage that there exists a value for each position, i.e. the output image does not contain any “holes”. These holes would occur due to the quantization, since $\vec{f_m}$ is continuous while the image-coordinates are discrete.

For a detailed explanation of the visual forward model, see Chapters 2 and 3. Furthermore, note that the geometric visual forward model as described in Chapter 3 had not been developed yet at the point when the study described in this chapter has been conducted. For a better understanding, we will recapitulate the main steps taken for the learning and implementation of an adaptive visual forward model and its quasi-inverse, the saccade controller, and ultimately develop a combined inverse–forward model by fusing the former two.

5.3.1. Data Acquisition

The visual forward model is trained by an initial exploration phase. During this exploration phase, the camera is moved systematically while facing a static scene that should contain a large variety of visual stimuli. Formally, we define a $N_u \times N_v \times N_{\Delta\varphi} \times N_{\Delta\theta}$ regular grid $\mathcal{G}$ in the joint motor-pixel position space. Here, N_u, N_v

denote the number of horizontal / vertical pixels in the output image and $N_{\Delta\varphi}, N_{\Delta\theta}$ denotes the number of discrete pan / tilt movements. Each grid point, given by the tuple $(u_i,\ v_j,\ \Delta\varphi_k,\ \Delta\theta_l)$, is associated with a so-called cumulator unit C_{ijkl} which corresponds to the input image and is of size $w_{\text{in}} \times h_{\text{in}}$. For every grid position $(i, j, k, l) \in \mathcal{G}$, each (x, y)-position within the corresponding cumulator unit C_{ijkl} is incremented by one if the absolute difference between the pixel intensity at position (u_i, v_j) in the output image and the intensity at position (x, y) in the input image falls beneath a certain threshold ε_c. This procedure is repeated for all grid positions and for different initial pan-tilt pairs. After the training, all cumulator units that correspond to valid pixels should contain clear maxima and can be used for the training of the mapping model. A cumulator unit is invalid if there is no clear maximum or if the strength of the maximum is lower than a pre-defined threshold and thus not included in the training set.

The above algorithm results in a training set containing samples for the mapping model $\vec{f_m}$, given by

$$\mathcal{T}_m = \{(u_i, v_j, \Delta\varphi_k, \Delta\theta_l, x_{ijkl}, y_{ijkl}) |\, (i, j, k, l) \in \mathcal{G}, \nu_{ijkl} = 1\}, \tag{5.2}$$

where ν_{ijkl} is either 1 or 0, depending on the validity of the corresponding cumulator unit C_{ijkl} (i.e. whether a clear maximum exists), and x_{ijkl}, y_{ijkl} correspond to the location of the maximum in C_{ijkl}.

For our experiments, we used a $13 \times 13 \times 11 \times 11$ training grid, resulting in a total of $20,449$ cumulator units. The training set for the mapping model consisted of $|\mathcal{T}_m| = 10,708$ samples which corresponds to the number of valid cumulator units.

5.3.2. Implementation

For the present study, we use a mapping model that is implemented by a "full" network of radial basis functions (RBF) (Girosi et al., 1995). Here, "full" means that the network comprises as many units as there are points in the training set. This is a reasonable choice, because the training data are gridded and assumed to be sufficiently exact (i.e. the noise level is low).

The RBF network for the mapping model writes out as

$$\begin{aligned}\vec{f_m}(u, v, \Delta\varphi, \Delta\theta) = &\sum_{i=0}^{|\mathcal{T}_m|} \vec{w}_i \kappa((u, v, \Delta\varphi, \Delta\theta)^\top, \vec{x}_i) \\ &+ \vec{a}_0 + \vec{a}_1 u + \vec{a}_2 v + \vec{a}_3 \Delta\varphi + \vec{a}_4 \Delta\theta,\end{aligned} \tag{5.3}$$

where $\vec{w}_i \in \mathbb{R}^2$ are the output weights of the non-linear part, $\vec{x}_i \in \mathbb{R}^4$, $i = 1, \ldots, |\mathcal{T}_m|$ are the training points (i.e. the tuples corresponding to the input part $(u_i, v_i, \Delta\varphi_i, \Delta\theta_i)$ from $\mathcal{T}_m$), and $\vec{a}_0, \ldots \vec{a}_4 \in \mathbb{R}^2$ are the output weights of the linear (affine) part. Furthermore $\kappa(\cdot, \cdot)$ denotes the kernel or radial basis function. In our case, $\kappa(\vec{x}_1, \vec{x}_2) = \|\vec{x}_1 - \vec{x}_2\|$, the so-called Euclidean kernel. The linear part in (5.3) is necessary in order to assure that the linear system, which needs to be solved

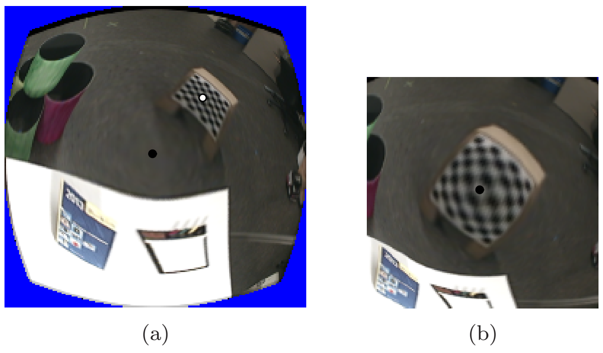


Figure 5.3.: Saccade controller virtually fixating an object. The black dot marks the image center. (a) The target object is marked by a white dot in the initial view. (b) The input image is warped using the combined inverse–forward model. Note that the target point is exactly located in the image center.

for the output weights, has a unique solution (Micchelli, 1986; Girosi et al., 1995). The Euclidean kernel has the benefit that it is computed more efficiently than the widely-used Gaussian kernel and that it does not include a width parameter that would depend on the scaling of the data.

The validator model has been implemented by a heuristic that checks the validity of an image coordinate (u, v) based on two criteria: (i) the mapped coordinate $(\hat{x}', \hat{y}')$ has to be within the boundaries of the input image, and (ii) the pixel value $I_{\text{in}}(\hat{x}', \hat{y}')$ at that position must lie within the interval $[0, 1]$; pixels on the margin around the retinal images have a value of -1 and are thus invalid per default. The function $f_v(u, v, \Delta\varphi, \Delta\theta)$ returns 1 for a valid pixel and 0 otherwise.

Figure 5.2 shows an example for the VFM: from an initial retinal image (Figure 5.2b) two simulated saccades are performed (Figures 5.2a and 5.2b (bottom)). The blue area marks portion of the images which are classified as invalid by the validator model. Note that the predictions by the VFM closely resemble the real views (Figures 5.2a and 5.2b (top)). For a thorough analysis of the performance of the RBF-based VFM, see Chapter 2.

5.4. Saccade Controller

The saccade controller (SC) is an inverse model that, given a target location in the input image $\vec{s} = (x, y)$ and a desired goal location (u^*, v^*) (which will be fixed at the image center in the following), generates a motor command $\hat{\vec{m}} = (\widehat{\Delta\varphi}, \widehat{\Delta\theta}) = \vec{f}_{\text{SC}}(u^*, v^*, x, y)$ that controls the PTU such that the current location moves onto the goal location. In other words, if an object is located at position (x, y) in the image before the movement, it would be located at position (u^*, v^*) after the execution of $(\widehat{\Delta\varphi}, \widehat{\Delta\theta})$.

A training set for the saccade controller can be derived from the VFM's training set (5.2) by changing the roles of (x, y) and $(\Delta\varphi, \Delta\theta)$. The pixel location in the input image is treated as input whereas the motor command is treated as an output; the role of (u, v), the location in the output image, is unchanged.

The function $\vec{f}_{\mathrm{SC}}$ can be approximated, just like the visual forward model, e.g. by a network of radial basis functions. For the present study, we use an RBF network with Euclidean kernels (see (5.3)).

Furthermore, the training sets for VFM and SC can be combined such that a sequential application of the two internal models is avoided. This is done by replacing every occurrences of the tuple $(\Delta\varphi, \Delta\theta)$ in the training set $\mathcal{T}_m$ by the associated value of (x, y)—the target location—for which $u^* = \bar{u}$ and $v^* = \bar{v}$, where $(\bar{u}, \bar{v})$ denotes the image center. This was done using a nearest neighbor search; points for which no corresponding tuple $(\Delta\varphi, \Delta\theta, \bar{u}, \bar{v})$ could be found were excluded from the training set.

The function for the inverse–forward model shall be denoted by

$$(\hat{x}', \hat{y}') = \vec{f}_{\mathrm{SC\text{–}VFM}}(u, v, x, y).$$

Again, we can use an RBF network to approximate this function from the training data. The function $\vec{f}_{\mathrm{SC\text{–}VFM}}$ is used for image warping by applying it to each pixel in the output image just like $\vec{f}_m$.

Figure 5.3 shows an example for the application of the combined inverse–forward model $\vec{f}_{\mathrm{SC\text{–}VFM}}$. The location marked by a white dot in Figure 5.3a corresponds to the target location (x, y). The image center is marked by a black dot. Figure 5.3b shows the input image after the application of $\vec{f}_{\mathrm{SC\text{–}VFM}}$: the former target location is now congruent with the image center; the rest of the image has been warped accordingly.

5.5. Stereo Matching

In the present study, we compare two different approaches for stereo matching: a novel approach, based on sensorimotor simulation and visual prediction, and a purely sensory approach based on the scale-invariant feature transform (SIFT) (Lowe, 2004).

5.5.1. Predictive Matching

A prerequisite for the predictive matching are segmented images, i.e. the potential partner locations to be searched are known beforehand. The predictive matching then performs an exhaustive search on these potential partners, matching each partner against the template object. The search process corresponds to an internal sensorimotor simulation: for each potential partner a motor command that would fixate this partner is generated. In the following, the left camera image is the reference (i.e. in which the target position is known) while the right camera image is subordinate (i.e. which is searched for the target).

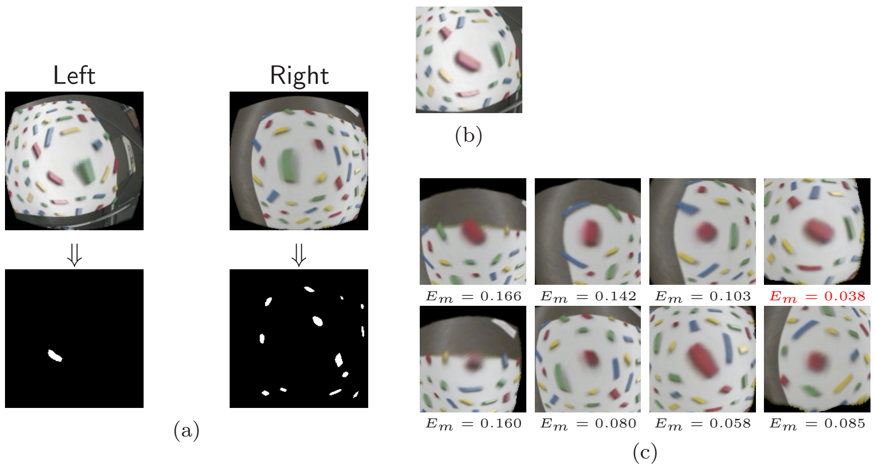


Figure 5.4.: Example of the predictive matching. (a) A target object is selected from the left retinal image, the right retinal image is segmented. (b) The target object is virtually fixated using the SC and the VFM. (c) All potential partners in the right retinal image are virtually fixated and matched against the template; the values of E_m are printed below the images.

Formally, for a given target position $\vec{s}_L = (x^L, y^L)$, we have a set of possible partners $\mathcal{S}_R = \{(x_1^R, y_1^R), \ldots, (x_P^R, y_P^R)\}$ which needs to be iterated accordingly. Because of the radial foveal warping of the images, we need to generate "canonical views" of the objects before the actual matching algorithm is applied. These canonical views are generated internally, using the left and right images and the position $\vec{s}_L$ and all $\vec{s}_{Ri} \in \mathcal{S}_R$. For this purpose, the saccade controller and the visual forward model are used in combination. While the former generates a fixation movement (for each position), the latter generates the corresponding view. Alternatively, the combined inverse–forward model may be used.

The canonical views of the target at position $\vec{s}_L$ is compared to each partner at position $\vec{s}_R \in \mathcal{S}_R$ using a difference-based matching approach. Let the canonical view of the target, the so-called template, be denoted by $\hat{I}_L$ and let the canonical view of the ith partner be denoted by $\hat{I}_{R,i}$ for $i = 1, \ldots, |\mathcal{S}_R|$; this shall simply be called the "partner". The template and the partner are compared using the normalized sum of squared difference (SSD) error, given by

$$E_m(\hat{I}_L, \hat{I}_{R,i}) = \frac{1}{w \cdot h} \sum_{u=0}^{w-1} \sum_{v=0}^{h-1} (\hat{I}_L(u,v) - \hat{I}_{R,i}(u,v))^2. \tag{5.4}$$

If we assume that the pixel values are normalized to the unit interval, then $E_m(\cdot,\cdot) \in [0,1]$. There may be the case that the potential partner is not visible at all in the right image. Therefore, it is desirable to introduce a maximum threshold ϵ_m on E_m to cancel out incorrect matches.

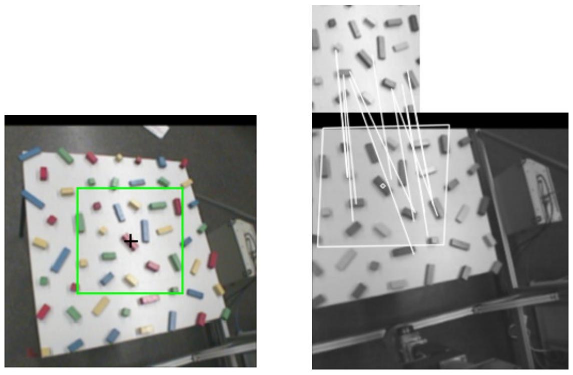


Figure 5.5.: Example for the SIFT-based stereo matching. (a) The left camera image; the selected target is framed by a green rectangle. (b) The template image (top) and the right camera image (bottom). Matching key points are connected by lines. The position of the object as well as the rectangular region haven been projected using the homography.

Figure 5.4 shows the matching process for an example pair of stereo images. First a salient object is selected in the left retinal image, its centroid determines its position $\vec{s}_L$ (Figure 5.4a (left)). The right retinal image is processed in an analogous fashion; all salient objects are retained and their centroids determine their positions $\mathcal{S}_R$ (Figure 5.4a (right)). The target object is fixated by the SC and VFM, yielding a canonical view (Figure 5.4b). This canonical view is matched against the canonical views of all potential partners (Figure 5.4c). The partner yielding the lowest SSD is considered as the correct match (top row, second column in Figure 5.4c).

5.5.2. SIFT-Based Matching

SIFT is a popular algorithm for feature matching that is rather invariant under scale and rotation. It comprises two phases: key point detection and descriptor computation. The first phase identifies "good" key points within the image; in the second phase a 128-dimensional feature descriptor is calculated for each key point (Lowe, 2004).

It is important to note that a target object, which is represented by its position and template in the predictive approach, is now represented by a set of 128-dimensional feature descriptors. The descriptors are calculated at key points found within the template. The number of descriptors highly depends on the structure of the image content.

In contrast to the prediction-based approach, the potential partner positions in the right image are not taken into account. The whole image is searched for matching descriptors. For this, we employ a k-nearest neighbor (k-NN) search with $k = 2$. Let the set of key points for the template be denoted by $\mathcal{K}_L = \{\vec{k}^L_1, \dots, \vec{k}^L_{N_L}\}$ and for the right image $\mathcal{K}_R = \{\vec{k}^R_1, \dots, \vec{k}^R_{N_R}\}$, respectively. Each key point has an associated descriptor $\vec{d}(\vec{k}_i^{L/R}) \in \mathbb{R}^{128}$. We are looking for those key point pairs which are homologous, i.e. correspond to the same 3D world coordinate. Therefore, we determine for each $\vec{d}(\vec{k}_i^L)$ the nearest and second-nearest neighbors, denoted by $\vec{d}(\vec{k}_{1,i}^R)$ and $\vec{d}(\vec{k}_{2,i}^R)$, respectively. Pairs $(\vec{k}_i^L, \vec{k}_{1,i}^R)$ of key points for which $\|\vec{d}(\vec{k}_i^L) - \vec{d}(\vec{k}_{1,i}^R)\| < \epsilon_s \|\vec{d}(\vec{k}_i^L) - \vec{d}(\vec{k}_{2,i}^R)\|$ holds are considered as corresponding. Because correct matches are characterized by a significantly closer nearest neighbor, the smaller ϵ_s is chosen, the more selective the criterion becomes. Moreover, using a relative threshold is more robust than a global one (Lowe, 2004); this is because some descriptors are more discriminative than others.

In order to recover the position $\vec{s}^R$ of the object in the right camera image, we need to calculate the homography (Hartley and Zisserman, 2003), using the pairs of corresponding points, determined by the SIFT algorithm. We use the robust random sample consensus (RANSAC) algorithm (Hartley and Zisserman, 2003) for this purpose. Once the homography is calculated, the object location $\vec{s}^L$ can be transformed according to

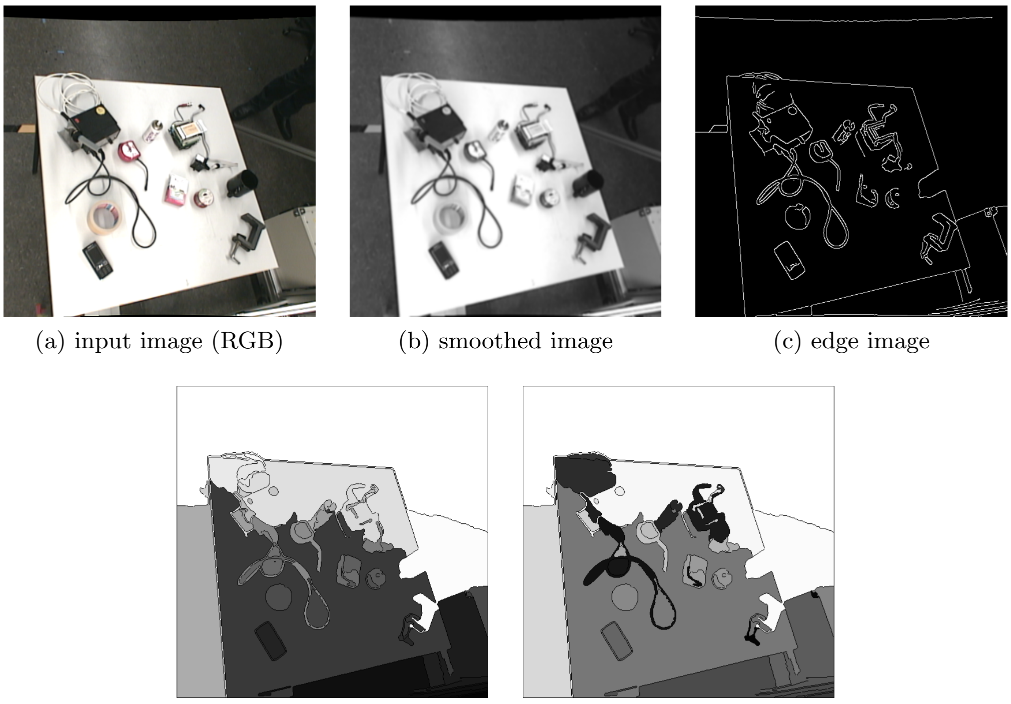
$$\begin{aligned} \begin{bmatrix} \tilde{\vec{s}}^R \\ w \end{bmatrix} &= H \begin{bmatrix} \vec{s}^L \\ 1 \end{bmatrix} \\ \vec{s}^R &= w^{-1}\, \tilde{\vec{s}}^R, \end{aligned} \tag{5.5}$$

where H denotes the 3×3 homography matrix. Note that, in order to estimate the homography matrix, the matching algorithm needs to find 4 corresponding point pairs at least. For our experiments, we used the SIFT and RANSAC implementations which are freely available through the OpenCV computer vision library (Bradski, 2000).

Figure 5.5 shows an example for the SIFT-based matching algorithm. The selected object is framed by a green square (Figure 5.5a) whose size corresponds to the actual template size of 101×101 pixels. The template is converted to grey scale (Figure 5.5b (top)) from which the SIFT descriptors are computed. The same procedure is repeated with the whole right camera image (Figure 5.5b (bottom)). For the pair of example images, the SIFT algorithm has found 107 key points within the template and 264 key points in the scene image. These key points where matched according to a value of $\epsilon_s = 0.4$. The remaining corresponding key points are marked by lines from the template to the scene in Figure 5.5b. The estimated homography was used to transform the target position and the outline of the template (Figure 5.5b (bottom)). Note that the target position has been accurately recovered.

5.6. Saliency Detection

An important prerequisite for the predictive matching approach is the selection of interesting (salient) points within the pair of images. These interest points serve as candidate points for the matching process. In order to establish reliable matches between candidate points in the left and the right image, these points should be


(a) input image (RGB) (b) smoothed image (c) edge image

(d) watershed segmentation (e) merged segments

Figure 5.6.: Steps of the image processing for saliency detection: input image (a), smoothed gray-scale image (b), detected edges (c), watershed segmentation based on edge image (d), merged segments (final saliency map) (e).

chosen consistently. That means that the points should roughly be homologous, i.e. belong to the same 3D world coordinate.

We will determine these candidate points by employing techniques for image segmentation based on color and edge information. These resulting segments serve as "salient" regions during the matching process. For the two experiments we will present in the following section, we use two different approaches to image segmentation. The first approach is based on a simple color segmentation using a pre-defined color, the second approach, which will be explained in depth in the following, is more elaborate and can also be applied to images of real-world scenes. Furthermore, we are only interested in segments of a certain size—i.e. medium-sized segments that most likely belong to prominent or interesting objects in the scene. Therefore, we term the segmentation process *saliency detection*. We are well aware that usually salient detectors suppress other less interesting regions in favor of one salient stimulus (e.g. Walther and Koch (2006); Orabona et al. (2007)). However, for our matching approach, we need a list of quasi-salient object locations readily available. Therefore we chose to define saliency as mentioned above.

The result of the segmentation is—in both cases—a list of coordinates (in image space) comprising the positions of the centroid of each segment. We assume that the centroid is a robust choice in the sense that it leads to fixated views of the corresponding segment (in the left and right image) that can be reliably matched.

The saliency detector is based on the following image processing steps: application of Gaussian blur, application of Canny's edge detector (Canny, 1986), segmentation by a non-parametric watershed transformation (Beucher and Meyer, 1992), and finally reduction of over-segmentation by clustering the segments through a variant of the DBScan algorithm (Ester et al., 1996). This procedure is inspired by recent computational methods for saliency detection (Walther and Koch, 2006; Orabona et al., 2007; Wischnewski et al., 2010); see those references for further details. Ideally the saliency detector should identify segments of homogeneous color and texture, and segment the left and right images in a consistent manner. Large and tiny segments, which most probably belong to background and noise, are removed from the list. The result of the segmentation is a list of target positions that correspond to the centroids of the segments.

5.6.1. Pre-Processing

The pre-processing comprises three major steps: (i) the RGB camera image (Figure 5.6a) is converted into gray-scale (by calculating the mean over each channel for every pixel), (b) the resulting camera image is blurred (Figure 5.6b) to remove noise, and finally Canny's edge detector is applied (Figure 5.6c). The resulting image is binary, with pixel values of 1 indicating edges.

The preprocessing step comprises several parameters. One of them, the size of the Gaussian blur mask, needs to be adjusted according to the image size. For the camera images of 540 × 540 pixel, we chose a blur mask of 7 × 7, for the retinal images of 207 pixels, we chose a smaller mask of 3 × 3.

For edge detection, we use the implementation of the Canny edge detector from the OpenCV library (Bradski, 2000). The implementation requires to set two hysteresis parameters which define which gradient magnitudes are accepted as edge pixels and which pixels should be rejected. The lower threshold was set to 0.2 and the upper threshold to 0.6 (assuming normalized pixel values in $[0, 1]$).

5.6.2. Segmentation and Clustering

The edge image is used to calculate a rough segmentation of the image by applying a parameter-free watershed algorithm (Beucher and Meyer, 1992; Bradski, 2000). The output of the watershed algorithm, applied to the example image, is shown in Figure 5.6d. It turned out in preliminary experiments that the watershed algorithm leads to an over-segmentation of the image. This means that regions that appear homogeneous to a human observer collapse to smaller segments in the watershed segmentation.

Therefore, we implemented a clustering method that merges regions based on a homogeneity criterion and the adjacency structure of the regions. The adjacency graph is computed based on the topology of the watershed segmentation (see Figure 5.6d). Each segment is represented as a vertex; neighboring segments are connected by edges. Furthermore, each segment is associated with a region descriptor $\vec{r_i} = (m^i_{10}, m^i_{01}, m^i_{00}, \bar{L}^i, \bar{a}^i, \bar{b}^i)^\top$ that contains the position of its centroid (m^i_{10}, m^i_{01}), its “mass” m^i_{00}, and its mean Lab color triplet $(\bar{L}^i, \bar{a}^i, \bar{b}^i)$. The moments were normalized with respect to the image size. The Lab values were normalized to lie within $[0, 1]$.

We use a modified version of the DBScan algorithm (Ester et al., 1996) to perform the clustering. The algorithm has been modified in two ways: (i) it respects the adjacency graph, i.e. it only clusters adjacent regions, and (ii) it allows clusters of size one. Segments which are topographically connected and for which $\|\vec{r_i} - \vec{r_j}\| < 0.1$ holds, are merged. Figure 5.6e shows the results of the clustering. In this example, the clustering reduced the total number of segments from 80 to 58. The final segmentation is used to compute the target positions by determining the centroids of each segment. Segments containing more than 10000 pixels were considered as background, segments with less than 10 pixel as noise. Those segments were not considered in the matching process.

5.7. Results

5.7.1. Experiment 1

We tested both matching approaches on a database of image pairs recorded by the stereo camera setup. An arrangement of colored wooden blocks was placed on the table in front of the camera setup. For all experiments, we used 100 random initial viewing directions, i.e. the cameras were pointing in randomly selected directions.

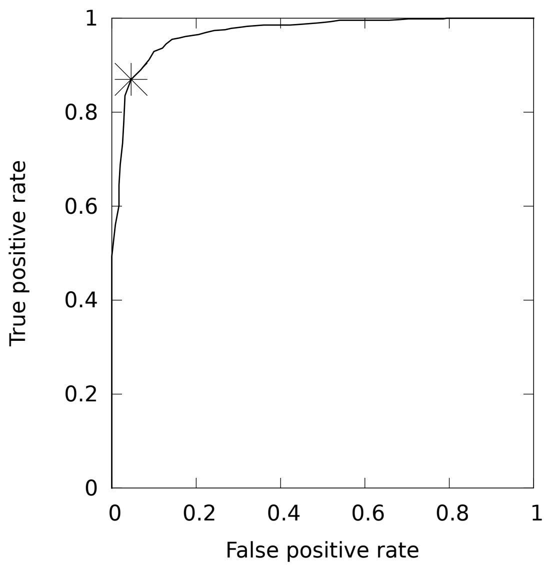


Figure 5.7.: ROC curve for the simulation-based matching approach. The parameter ϵ_m has been varied in the interval $[0, 1]$ with a step size of 0.001. The asterisk marks the optimum ROC at $\epsilon_m^* = 0.039$.

The images were segmented according to the color red. All red objects that were visible in the left image were matched against the candidates in the right image.

Preliminary tests showed that the SIFT-based matching is useless in combination with the retinal images. The number of key points found was too low, so that the homography could not be estimated in most cases. Therefore, we discarded the retinal images and used undistorted camera images of 240×240 pixels for the SIFT-based matching instead. The simulation-based approach was tested exclusively using 207×207 retinal images.

In order to evaluate the quality of the matching algorithms, we use a set of ground truth data which we obtained by manually correcting the results from a preliminary matching process. As the viewing direction of the cameras is variable, we cannot use the objects' positions within the images. Instead, we use the PTU angles (φ, θ) when fixating an individual object as a global reference. Thus, we obtain two lists of PTU angles for all objects in the left and right camera image, respectively. These lists allow us to perform a nearest neighbor search to determine the object's index within either of the two lists. A match is considered positive if the index for the object in the left camera image equals the index for the object in the right camera image.

The simulation-based matching approach involves the virtual fixation of objects. Therefore, calculating the PTU angles is trivial. The SIFT-based approach however relies on sensory information alone; the PTU angles have to be calculated separately. Moreover, the SIFT approach works on plain camera images. Thus, we have to

	#matches	%correct
pred. ($\epsilon_m = 0.039$)	640	97.34
SIFT ($\epsilon_s = 0.2$)	287	89.20
SIFT ($\epsilon_s = 0.3$)	724	85.77
SIFT ($\epsilon_s = 0.4$)	916	85.26
SIFT ($\epsilon_s = 0.8$)	1264	78.88

Table 5.1.: Overall results of the matching test. The predictive matching approach (pred.) reaches a high percentage of correct matches. The SIFT-based matching approach clearly performs worse. The theoretical maximum number of matches is 1400.

convert the object's coordinates into the retinal domain first and then apply saccade controller before we can validate the matching result. This rather complicated method is not part of the approach, but a necessary detour to evaluate the quality of the matching approaches.

Both matching approaches include parameters which control the matching quality. The size of the template is a parameter which is shared by both methods. This parameter also influences their performance. Therefore, it needs to be selected carefully. Based on preliminary experiments, we chose a template size of 101×101 pixels for the SIFT-based matching. For the matching-based approach, we used the whole 159×159 image resulting from the virtual fixation to compute the SSD.

Furthermore, the matching-based approach includes a parameter ϵ_m that controls the rejection of mismatches. This parameter has been optimized by plotting the receiver operating characteristic (ROC) curve (see Figure 5.7). The optimum for this parameter was found to be $\epsilon_m^* = 0.039$. Table 5.1 (top row) shows the results of the simulation-based approach for the optimal parameter value. The matching was very accurate and matched approximately 97% of the targets correctly.

The SIFT-based approach includes a parameter ϵ_s that controls the matching of corresponding (homologous) key points. This parameter has to be chosen such that the number of correspondences found is still high enough to calculate the homography (for most cases) while keeping the number of mismatches low. Table 5.1 shows the number of found correspondences and the percentage of correct matches for different values of ϵ_s. It becomes clear that larger values for ϵ_s increase the number of correspondences, but also drastically increase the number of mismatches. The optimal parameter that yields the most comparable results (i.e. the best trade-off between the total number of matches and the percentage of correct matches) lies between $\epsilon_s = 0.2$ and $\epsilon_s = 0.3$.

5.7.2. Experiment 2

In the second experiment, we compared the performance of both matching approaches on a more commonplace scene: various tools and office utilities were placed on the table in front of the camera setup (see Figure 5.8). In contrast to the first

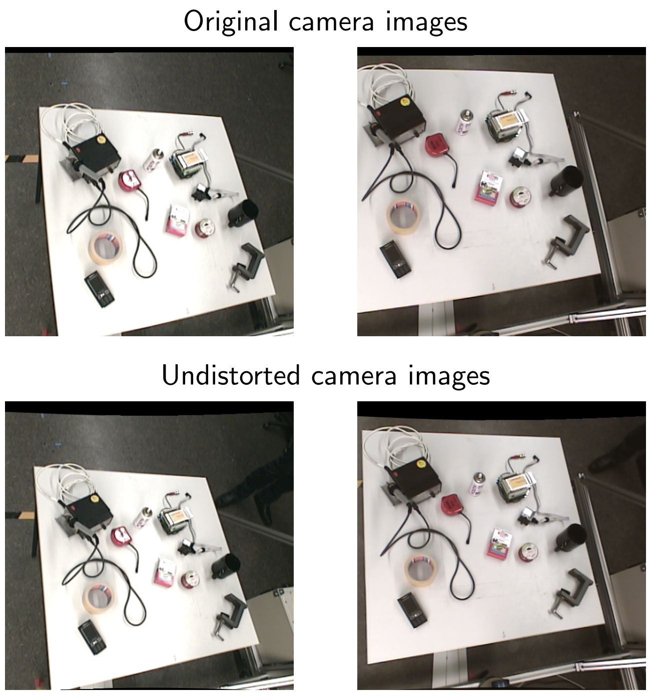


Figure 5.8.: Upper row: left and right camera image showing the scene for the second experiment. Lower row: undistorted version of the original camera images.

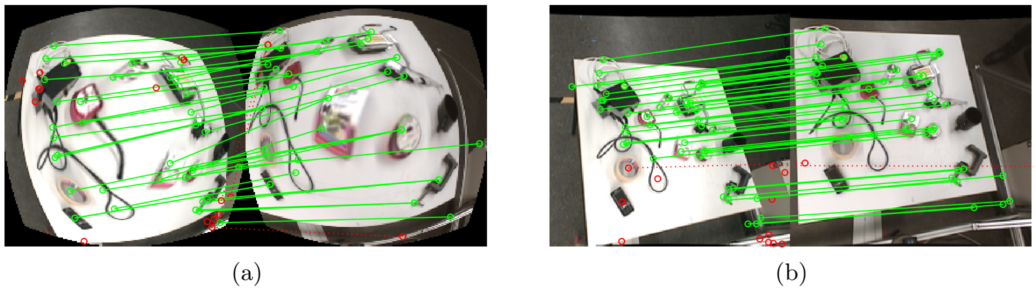


Figure 5.9.: Matches between objects in the left and right images. Correct matches are marked with a green solid line, incorrect ones with a dotted red line. (a) Results for predictive matching. (b) Results for SIFT-based matching. Saliency was computed on the undistorted camera images.

experiment, the initial viewing direction was not varied. Furthermore, the original camera images were undistorted for the SIFT-based matching to remove image distortions which might deteriorate the performance of SIFT (Lourenço et al., 2012) (second row in Figure 5.8). We employed the camera calibration toolbox for Matlab (Bouguet, 2008) for this purpose. The images have a resolution of 540 $\times$ 540 pixels and are thus substantially larger than the ones used in experiment 1.

Instead of detecting target objects for matching by a specific color (as in the first experiment) we chose a more general approach based on saliency detector, see Section 5.6.

The predictive approach works directly on the detected target positions and identifies matches between objects in the left and candidates in the right retinal images (see experiment 1). We used retinal images with a resolution of 207 pixels. For the virtual fixation, we employed the inverse–forward model $\vec{f}_{\mathrm{SC-VFM}}$. The virtual views had a resolution of 159 $\times$ 159 pixels. The matching was performed using the normalized SSD error (5.4) under consideration of the validator model: the conjunction of the validator masks of both the virtual target and partner images were formed, and the SSD was computed only for valid pixels. If the number of valid pixel fell below 5000, the match was discarded. Furthermore, a match was discarded if the SSD was higher than $\epsilon_m = 0.08$.

In contrast, SIFT-based matching relies only on the target positions in the left image. Around each target position, a quadratic region with a size of 159 $\times$ 159 pixels is cut out which serves as a template. SIFT descriptors are determined within the template, and this set of descriptors is matched with all descriptors in the right image as described in Section 5.5.2. The homography is calculated from the matching key points and finally used to determine the corresponding position of the target in the right image. If the number of matching descriptors was too low to compute the homography, a match could not be established.

For a better comparison there are two test cases: In the first one, saliency is computed on the undistorted camera images, in the second one saliency is computed

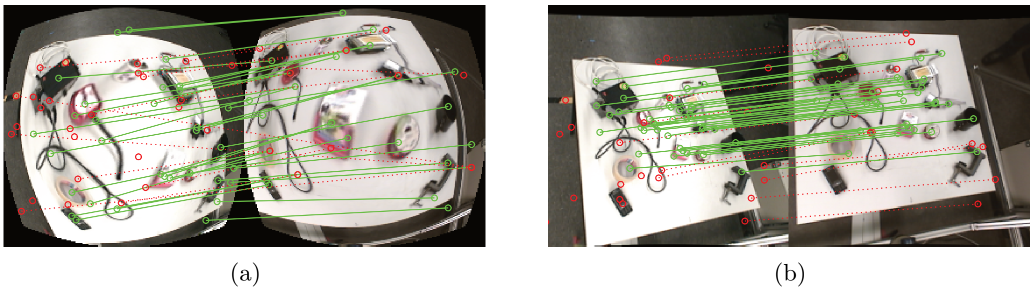


Figure 5.10.: Matches between objects in the left and right images. Correct matches are marked with a green solid line, incorrect ones with a dotted red line. (a) Results for predictive matching. (b) Results for SIFT-based matching. Saliency was computed on the retinal images.

on the retinal images. For the first case, the target positions are mapped onto their corresponding positions in the retinal images for the predictive matching. For the second case, the target positions are mapped from the retinal image onto their corresponding positions in the undistorted images. In this way, it is guaranteed that both matching procedures work on the same set of targets in the left image.

For the undistorted case, there are 55 target locations in the left image. By predictive matching, 35 correct and 3 incorrect matches were established; the SIFT-based matching yielded 42 correct and 2 incorrect ones. Rating was based on visual inspection; a match was rated as correct if the selected position in the right image was in close vicinity to the correct position. The correctly and wrongly established matches are visualized in Figure 5.9. For the retinal case (see Figure 5.10), there are 58 target locations in the left image. The predictive matching yielded 32 correct matches and 13 incorrect ones; the SIFT-based matching established 36 correct matches and 11 incorrect ones. Note that the number of targets deviates due to a different parameterization of the saliency detector.

SIFT worked reliably with overall very good precision. In comparison, predictive matching performed slightly worse. However, one has to consider that with the current test methodology it is not guaranteed that each target in the left images has a correct partner in the right image, resulting in mismatches. This is mainly due to the limitations of the saliency detector.

5.8. Conclusions & Outlook

We presented a novel method for solving the correspondence problem in a pair of stereo images based on an internal sensorimotor simulation. In contrast to most purely sensory, descriptor-based methods, our approach is able to cope with severe lens distortion (retinal images) and is non-parametric—the internal models are learned in a preceding sensorimotor exploration phase. During the simulation, objects are virtually fixated, yielding "canonical" views of the objects that can be compared by simple difference-based methods. Therefore, the images are represented in a way which is invariant with regard to the current viewing direction. Furthermore, the virtual (or covert) fixation movements can be used as a means to determine the global position of an object. This could be used to guide a robotic manipulator to e.g. perform a grasping movement.

The model can be easily extended to directly extract depth information from the covert fixation movements generated during the internal simulation. As one can easily see, the vergence angle, defined by $\alpha = \varphi^L - \varphi^R$, where φ^L, φ^R denote the pan angles of the left and right PTU, is anti-proportional to the depth of the fixated object. This relationship could be used to rank-order the target objects and assign relative depth values which would lead to a sparse density map of the scene.

Our approach makes extensive use of virtual fixations. These need to be carried out for all candidate points. In our experiments the number of candidates was relatively low (14 (experiment 1), ~ 55 (experiment 2)). Therefore, the run time of the algorithm was moderate. With a high number of candidates (as e.g. in the SIFT-based matching, where we had ~ 200 key points) the run time would increase drastically. To overcome this problem, the simulation would have to be parallelized. A parallel version of the simulation-based matching could be easily derived, because most parts of the simulation do not share resources or depend on each other.

Instead of purely sensory data, our approach also relies on motor information. This permits the use of very simple matching measures. For the study presented in this chapter, we used a difference-based measure. Furthermore, the approach is able to cope with retinal images that exhibit an uneven resolution. Extensive experiments on a image database have shown that the predictive matching approach can outperform SIFT-based matching if the location of the potential partners is known (in this case we used a red-segmentation to determine the positions). On images of a realistic scene, the predictive matching appeared to be inferior to SIFT. This is mainly due to the fact that the two images have to be segmented likewise (i.e. segments should be detected consistently in both images). Therefore, the performance significantly relies on the performance of the image segmentation. For our experiments, we used a simple saliency-based image segmentation which only resulted in suboptimal (i.e. slightly inconsistent) segmentations. An alternative approach in favor of predictive matching could be to compute a simple region descriptor at each target in the left image and to pre-select regions in the right image with similar descriptor values for subsequent matching.

In conclusion, predictive matching is competitive to classical approaches from computer vision, and it has moreover the considerable advantage that it is fully adaptive and can cope with highly distorted images. This is accomplished by applying internal sensorimotor simulation and (subconscious) mental imagery to the process of stereo matching.

6. Overall Conclusions & Outlook

We have suggested a computational model for the perception of the functional role of objects based on internal sensorimotor simulations. The model is based on the perception through anticipation paradigm which states that the perception of possible object affordances is based on internally simulated object interactions. The model comprises several sub-models which have been investigated in depth. The main focus of this thesis lies in the efficient implementation of visuomotor models that predict future sensory states based on motor commands and current sensory states. Most of these models were learned based on information gathered during active exploration of the corresponding sensorimotor space. We employed feed-forward neural networks and geometric approximations for the implementation of these models.

6.1. Visual Prediction

We extensively studied the problem of visual prediction—i.e. predicting the optical flow within a camera image given a movement of the camera. Two models were presented: a fully adaptive model based on a statistical learning approach, and a geometric approximation. The adaptive model was implemented by a feed-forward neural network using radial basis functions (RBFs). We employed a statistical learning approach to acquire a training set for this model. Because of the high number of training examples, the process of data acquisition is very costly. Furthermore, each collected data point is represented by a unit within the RBF network, leading to a high computational complexity. Therefore, we had to investigate efficient methods for implementing the model.

We proposed a efficient method for the evaluation of RBF networks—the two-staged approach—in the context of image warping. The two-staged approach relies on the sequential application of two RBF networks: The first stage transforms a small regular grid of pixel positions inside the output image according to a given saccadic motor command; the second stage transforms each pixel position inside the output image by interpolating between the warped grid points. Both RBF networks are "full", i.e. they comprise as many units as there are training points. The training set for the first stage is rather large, consisting of $\sim 10,000$ training examples. The second stage is trained on the grid points only (where the points on the regular grid serve as inputs, and the corresponding warped grid points, generated by the first stage, serve as target outputs), and therefore consists of a much smaller number of ~ 100 units. Hence, the large first stage is only evaluated at a few grid-points, while the significantly smaller second stage is evaluated at all pixel positions. Furthermore, we employed a computationally simple RBF, the Euclidean kernel, for both stages. The results show that the performance of the novel approach outperforms the previous approach by Schenck and Möller (2007) in terms of the prediction quality. An analysis of the number of involved operations shows that the two-staged approach is also computationally more efficient.

As an alternative to the adaptive visual forward model, we proposed a geometric approach for visual prediction. The underlying geometric model is based on the assumption that the camera is mounted on a pan-tilt unit (PTU) in such a way that the pan and tilt axes intersect in the optical center. The optic flow could be predicted by using a perspective transformation and 3D rotations. The geometric model requires explicit knowledge about the actual distortion function underlying the camera optics (which is implicitly learned by the adaptive approach). Both models predict the optical flow within the visual field as the eye moves. Therefore, they can be seen as implementations of predictive remapping (Duhamel et al., 1992).

Both, the adaptive and the geometric model, have their individual advantages. While the adaptive approach is able to cope with arbitrary image distortions, the geometric model requires exact knowledge about the distortion function underlying the camera images. The geometric model, however, constitutes relatively simple equations, and is thus computationally more efficient.

Besides visual prediction, we also studied the inverse problem—the problem of kinematic eye control. From the geometric model, we derived equations that relate a position within the visual field to its corresponding relative pan and tilt angles that could be used to fixate the position (i.e. move it into the center of the visual field). Tested on a real PTU–camera assembly, the geometric controller proved to be only accurate towards the central region of the visual field. In order to circumvent these inadequacies, we proposed an adaptive error-correction scheme that resembles neural structures in the brain. By using this correction scheme we could significantly improved the saccadic accuracy of the controller.

The visual prediction mechanism was successfully applied in Chapter 4 to generate peripheral views of gripper images, and in Chapter 5 to generate (canonical) fixated views of objects within a pair of stereo images. Generally, the visual prediction mechanism is useful to create visual representations of objects that are invariant under the current viewing direction.

6.2. Visuomotor Associations

We proposed an adaptive model for visuomotor associations that was able to associate high-dimensional data, in this case images, to low dimensional kinesthetic states, in this case the postural variables of a robot arm. The model was implemented by a feed-forward neural network for the image synthesis, and the subsequent application of the visual forward model. We employed a linear technique for dimensionality reduction, PCA, which is well suited if the images reside in a linear subspace.

The associative model could be used to transform kinesthetic states into the visual sensory space. The object interaction model presented in Chapter 1 uses these sensory representations to predict possible object interactions. We assume that representing kinesthetic states in this way has several benefits: (i) the redundancies of the kinematics are not present (i.e. the representation abstracts from the concrete arm configuration), (ii) the object interaction-model receives all inputs in the same format (i.e. visual representations), (iii) the geometry of the gripper is implicitly encoded in this representation.

6.3. Future Research Directions

6.3.1. Alternative Learning Methods

For the adaptive models presented in this thesis, we employed feed-forward neural networks of MLP and RBF type. These methods are suitable for learning functional relationships. The training data were collected mainly in a structured systematic manner, i.e. by defining grids in sensorimotor space (Chapter 2) or in motor space alone (Chapter 4). The only exception is the error correction network in Chapter 3 for which we employed a random strategy. The aim of this section is to explore alternatives for the neural network implementations and to discuss different learning strategies.

Neural Network Types

Feed-forward networks capture the static input–output relation between motor commands and their sensory consequence. These networks are well suited for scenarios which can be described by a function. In some cases, however, the input–output relationship is not a function, but a one-to-many mapping. These situations arise when a motor command can have multiple possible sensory outcomes.

Associative neural networks have proved useful in these contexts. One example of such a network is neural gas principal component analysis (NGPCA) (Möller and Könies, 2004; Kaiser et al., 2010b)—an unsupervised neural network that represents data distributions by a set of hyperellipsoids. Furthermore, NGPCA includes a dimensionality reduction method which reduces the number of parameters that need to be learned. In the context of sensorimotor modeling, the NGPCA algorithm is applied to the joint sensorimotor space.

In order to predict sensory states based on motor information, a partial sensorimotor pattern is presented to the network (only including the motor state), and completed based on the learned sensorimotor manifold. Thus allowing the association between motor and sensory states and vice versa (without performing a retraining of the network). Therefore, this kind of network is extremely flexible in practical applications.

Recurrent neural networks (RNNs) (e.g. Lukoševičius and Jaeger (2009)) represent another alternative to feed-forward neural networks. RNNs incorporate a feedback loop from the output to an additional *context layer*. The overall RNN is therefore a dynamical system with an internal state, and can be described by a differential equation. RNNs have already been applied successfully by several authors to capture the dynamic properties of sensorimotor processes (see Chapter 1). Especially in the context of the prediction of object interactions, where complex dynamics occur due to various physical factors, these networks are good candidates for modeling.

Learning Strategies

The learning strategies that were applied in this thesis were mostly systematic and thus biologically implausible. Biological organisms learn through interactions with their environment and not by systematic explorations of the sensorimotor space. In the context of motor learning, these natural strategies are often referred to as motor babbling (Der and Martius, 2006) in analogy to toddlers that babble seemingly random words and phrases during the process of language acquisition.

Babbling strategies may be applied to the problem of learning object interactions, providing a more natural approach. Furthermore, the random movements that occur during the early stages of the learning process might result in a higher sampling of the sensorimotor space than a gridded approach would.

6.3.2. Alternative Approaches of "Perception through Anticipation"

In the following, we will outline how the perception through anticipation approach in general could be implemented in alternative ways. Basically, perception through anticipation relies on an internal sensorimotor simulation that results in a tree of possible action–response sequences, starting at an initial sensory situation S_0. Once the tree is built, the paths within the tree are evaluated based on a pre-defined criterion, resulting in a final percept. The approach may lead to an intractably large amount of simulation threads that need to be evaluated. Furthermore, the generation of the individual threads may allocate a large amount of resources and therefore result in high runtime.

Learning Shortcuts in the Simulation Tree

One possibility to reduce the runtime of the simulation threads could consist in the learning of shortcuts in the simulation tree. Such a shortcut could assign the initial sensory state and a given action to the final state of the simulation by direct association. In such a scenario, the intermediate simulation steps emanating from the initial sensory state would be skipped.

It must be noted that the intermediate steps may carry important information for the subsequent evaluation (e.g. the number of motor commands could be of interest, or the direction of certain motor commands). Therefore, the short-cut must also contain information about the intermediate steps as well. If we employ an associative model for relating initial sensory states to end-states and additional meta-information (i.e. the number of steps skipped), we may encounter one-to-many mappings. A one-to-many mapping, where an input corresponds to a set of possible outputs cannot be modeled by feed-forward neural networks. Therefore, we must investigate alternatives.

Probabilistic Formulation

Another possible reformulation of the perception by anticipation approach might involve probabilistic representations. The big advantage of probabilistic approaches is that they can deal with uncertainty. Therefore, these approaches are well suited for real world applications.

We suggest a particle filter framework (Van Der Merwe et al., 2000) for this purpose. Particle filters are typically applied in situations when the state of a system can only be observed through noisy measurements. The state is represented by a population of particles where each particle contains as many parameters as the system's state. Therefore, the particle population can be regarded as a cloud (or multiple clouds) of vectors that reside in the (possibly high dimensional) state space. As soon as a measurement arrives, the most probable particles (i.e. those that are most compatible with the measurement) are retained for the next step.

In the context of perception through anticipation, each motor command and its resulting sensory consequence could be regarded as a particle in joint sensorimotor space. A prediction step would correspond to a selection of the most probable particles according to a designated probability model. The probability model needs to be formulated (or learned) based on the sensorimotor relations that the agent experiences during overt actions. Eventually, the particle cloud converges towards a set of valid sensorimotor end-states that could be evaluated.

A. Down-Dating the Inverse of a Matrix

Let A be a real $n \times n$ matrix which is regular such that its inverse A^{-1} exists. Now let B denote an $m \times m$ sub-matrix of A. The aim is now to compute B^{-1} by only using elements from A^{-1}.

Let $\tilde{A}$ denote the rearranged version of A such that the columns / rows we seek to remove are the last $n - m$ columns / rows. We can establish this in terms of applying permutation matrices P_r and P_c to A; thus $\tilde{A} = P_r A P_c$. Note that, if A is symmetric, and if we want to preserve its symmetry in B, we have $P = P_r = P_c^\top$.

It can be easily shown that inverting $\tilde{A}$ is equivalent to applying P_r, P_c to A^{-1}, which directly follows from the orthonormality of permutation matrices.

We will derive a formula for down-dating the inverse of $n \times n$ matrix by m ranks by first noting that the inverse of a 4×4 block matrix is given by (provided B and C are invertible) (Horn and Johnson, 1990):

$$\begin{aligned}
\tilde{A}^{-1} &= \begin{pmatrix} B & U \\ V & C \end{pmatrix}^{-1} \\
&= \begin{pmatrix} (B - UC^{-1}V)^{-1} & -B^{-1}U(C - VB^{-1}U)^{-1} \\ -(C - VB^{-1}U)^{-1}VB^{-1} & (C - VB^{-1}U)^{-1} \end{pmatrix} \qquad \text{(A.1)} \\
&= \begin{pmatrix} \check{B} & \check{U} \\ \check{V} & \check{C} \end{pmatrix},
\end{aligned}$$

where B, $\check{B}$ are $(n-m) \times (n-m)$, V, $\check{V}$ are $m \times (n-m)$, U, $\check{U}$ are $(n-m) \times m$, and C, $\check{C}$ are $m \times m$, respectively.

Now, our aim is to express B^{-1} in terms of $\check{B}$, $\check{V}$, $\check{U}$ and $\check{C}$. First, we recall the Woodbury identity, given by (Horn and Johnson, 1990):

$$(B - UC^{-1}V)^{-1} = B^{-1} + B^{-1}U(C - VB^{-1}U)^{-1}VB^{-1} \qquad \text{(A.2)}$$

We can now rearrange (A.2) and identify the appropriate terms from A.1 in order to yield the desired result:

$$\begin{aligned}
B^{-1} &= (B - UC^{-1}V)^{-1} - B^{-1}U(C - VB^{-1}U)^{-1}VB^{-1} \\
&= \check{B} - \check{U}\check{C}^{-1}\check{V} \qquad \text{(A.3)}
\end{aligned}$$

We see that equation (A.3) is a Schur complement which can be conveniently calculated, e.g. by using LU factorization. Furthermore, we note that for a rank-1 down-date (A.3) reduces to

$$B^{-1} = \check{B} - \frac{\check{\vec{u}}\check{\vec{v}}^\top}{\check{c}}, \qquad \text{(A.4)}$$

where $\check{\vec{u}}$ is a column vector, $\check{\vec{v}}^\top$ is a row vector, and $\check{c}$ is a scalar.

In summary, the algorithm for down-dating the inverse of a matrix works as follows:

1. Permute columns / rows of A^{-1} to yield $\tilde{A}^{-1}$
2. Create matrices $\check{B}$, $\check{V}$, $\check{U}$ and $\check{C}$
3. Calculate B^{-1} according to (A.3) or (A.4)

B. Reconstruction of Gripper Positions

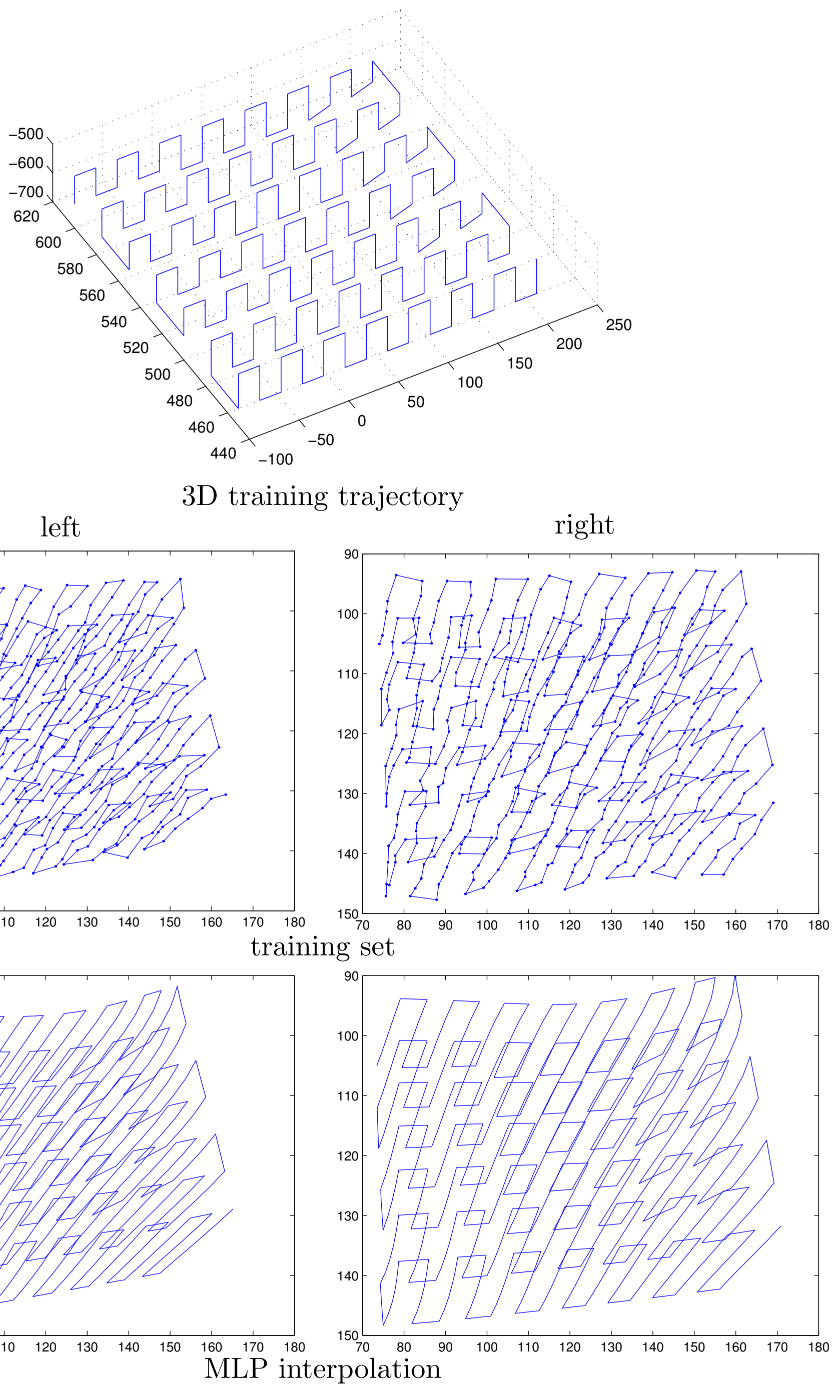


Figure B.1.: 3D trajectory, training set, and interpolated gripper trajectory (as seen in the left and right camera images) for a gripper orientation of 0°; axes are given in millimeters (top) and pixels (bottom 4), respectively.

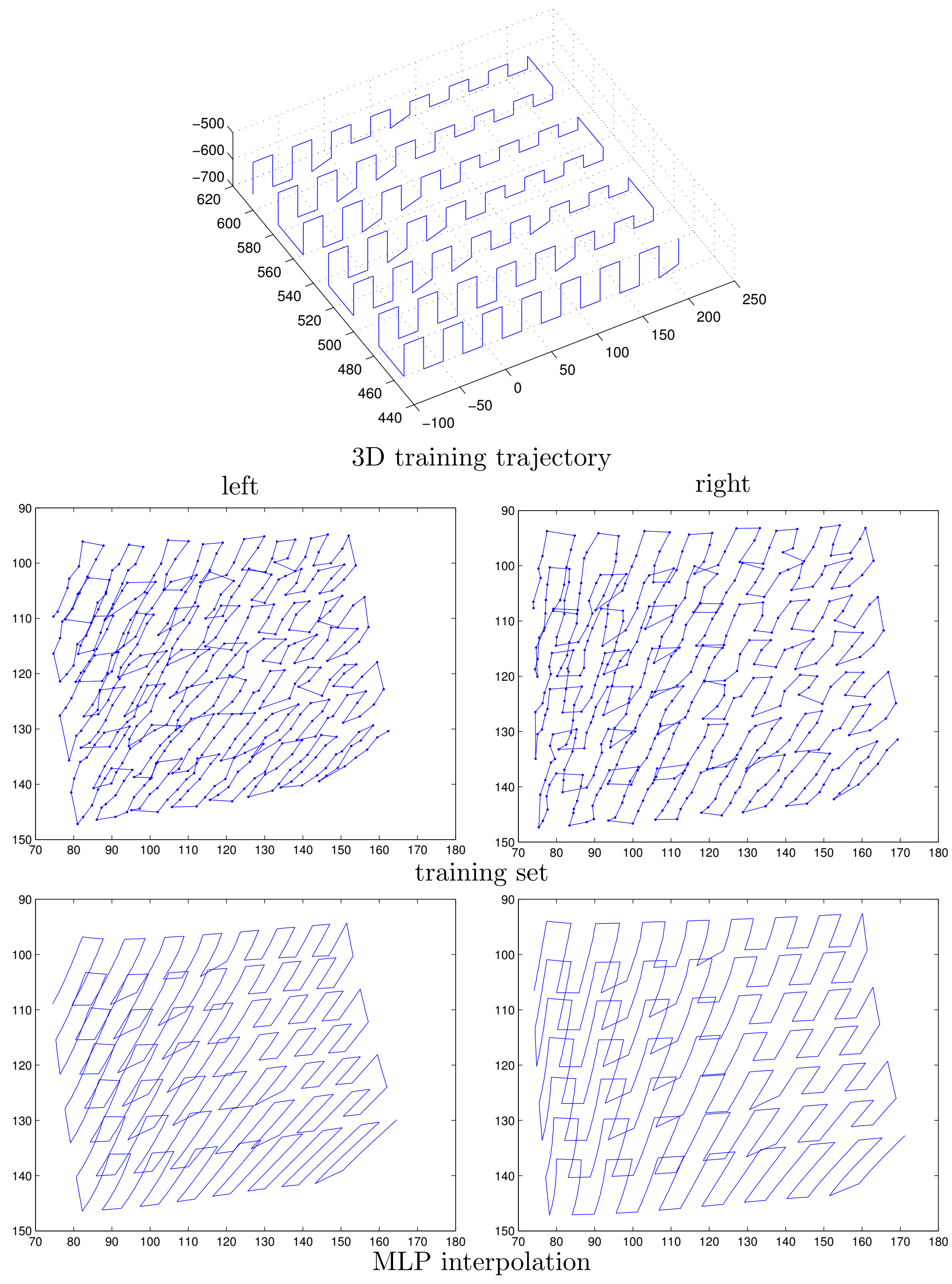


Figure B.2.: 3D trajectory, training set, and interpolated gripper trajectory (as seen in the left and right camera images) for a gripper orientation of 15°; axes are given in millimeters (top) and pixels (bottom 4), respectively.

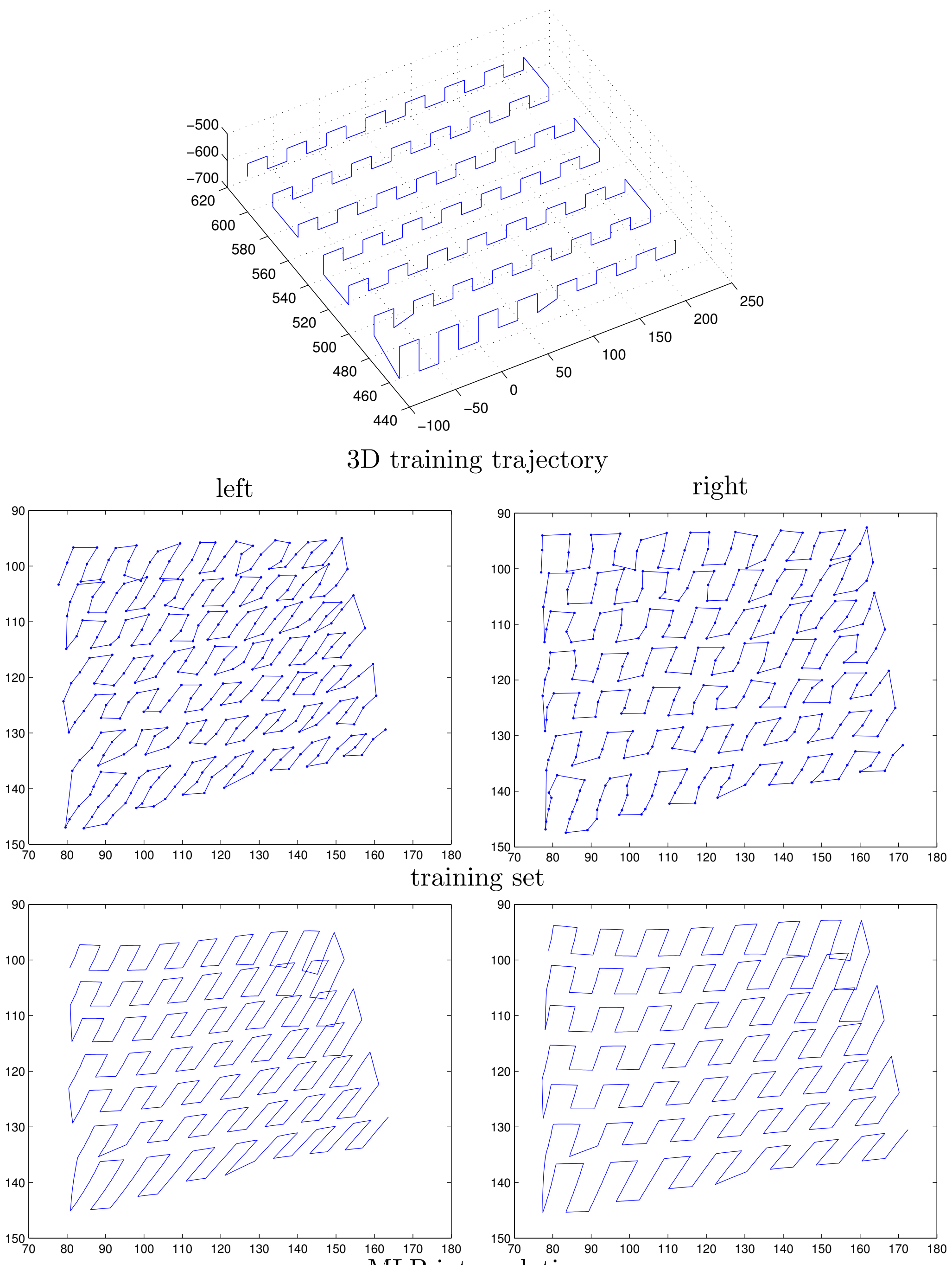


Figure B.3.: 3D trajectory, training set, and interpolated gripper trajectory (as seen in the left and right camera images) for a gripper orientation of 30°; axes are given in millimeters (top) and pixels (bottom 4), respectively.

C. Simulated Gripper Appearance

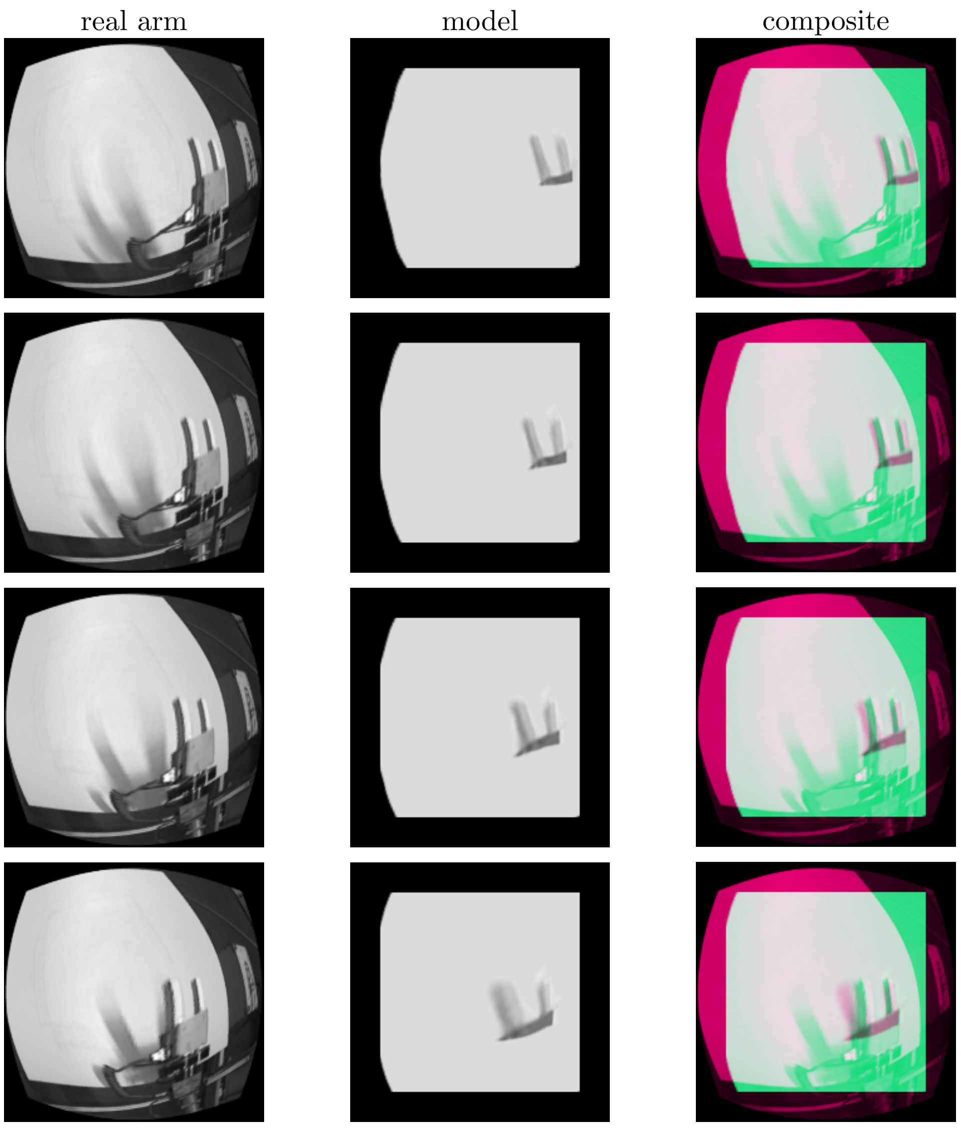


Figure C.1.: Simulated gripper appearance along an example trajectory. The figure depicts the real robot arm (left column), the output of the associative model (middle), and a composite view (right).

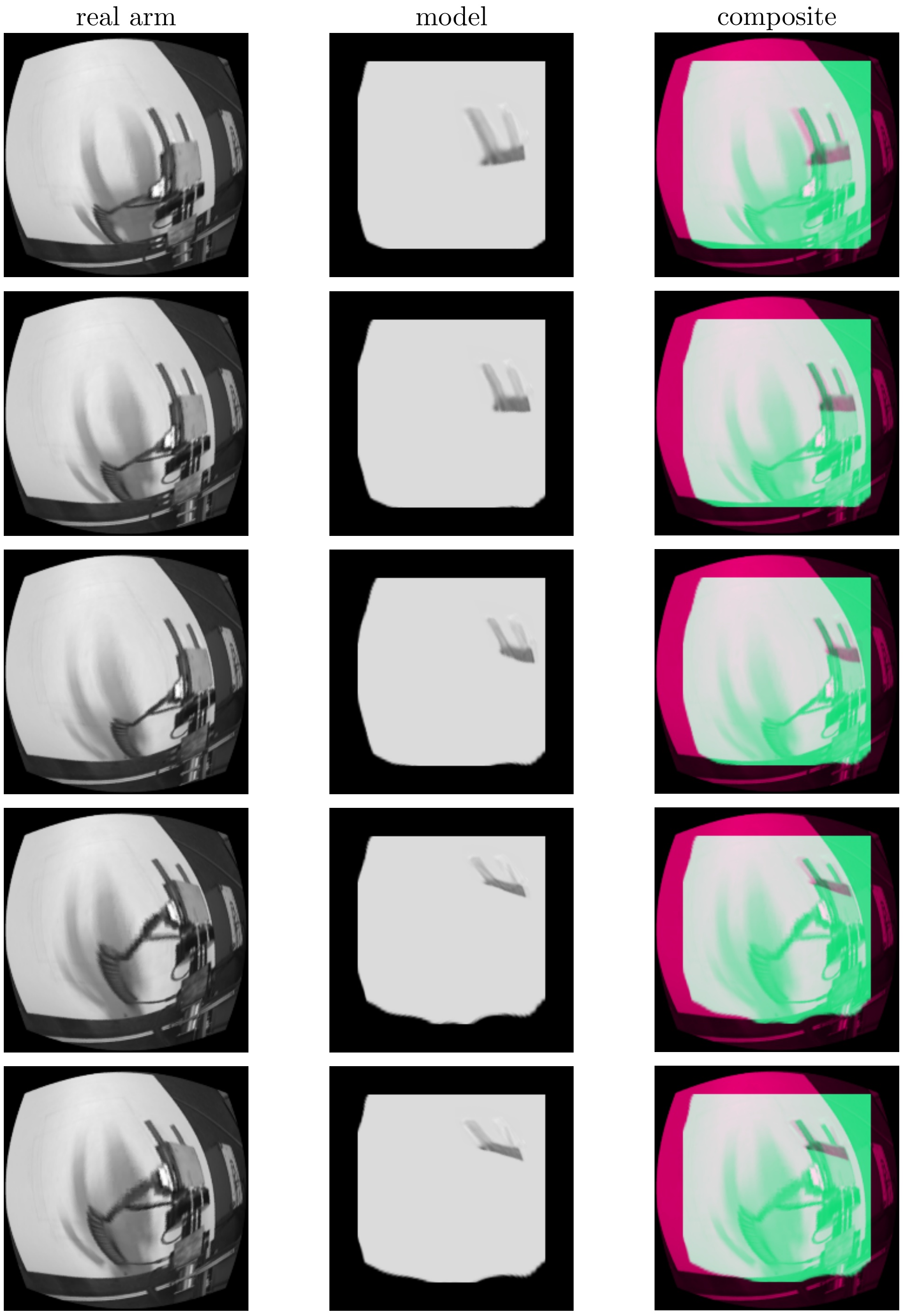


Figure C.2.: Continuation of Figure C.1

.

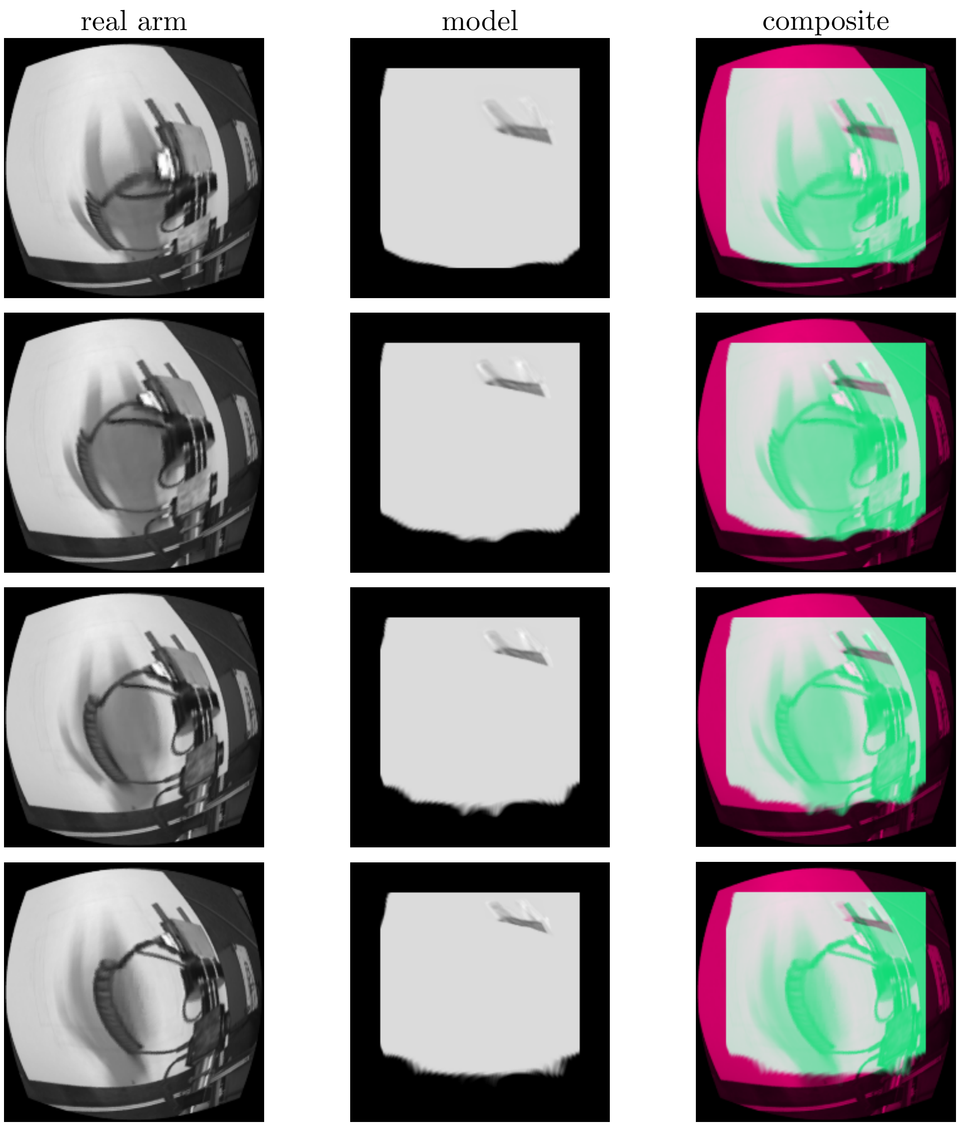


Figure C.3.: Continuation of Figure C.1

.

Bibliography

Nur Arad, Nira Dyn, Daniel Reisfeld, and Yehezkel Yeshurun. Image warping by radial basis functions: Application to facial expressions. *CVGIP: Graphical Models and Image Processing*, 56(2):161–172, 1994.

João Barreto, José Roquette, Peter Sturm, and Fernando Fonseca. Automatic Camera Calibration Applied to Medical Endoscopy. In *20th British Machine Vision Conference (BMVC '09)*, London, Royaume-Uni, 2009. The British Machine Vision Association (BMVA).

Ian Barrodale, D. Skea, M. Berkley, R. Kuwahara, and R. Poeckert. Warping digital images using thin plate splines. *Pattern Recognition*, 26(2):375–376, 1993.

Lawrence W. Barsalou. Perceptual symbol systems. *Behavioral and brain sciences*, 22(04):577–660, 1999.

Lawrence W. Barsalou. Grounded cognition. *Annu. Rev. Psychol.*, 59:617–645, 2008.

Richard K. Beatson and William A. Light. Fast evaluation of radial basis functions: methods for two-dimensional polyharmonic splines. *IMA Journal of Numerical Analysis*, 17(3):343–372, 1997.

Richard K. Beatson and Garry N. Newsam. Fast evaluation of radial basis functions: I. *Computers & Mathematics with Applications*, 24(12):7–19, 1992.

Richard K. Beatson, Jon B. Cherrie, and Cameron T. Mouat. Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration. *Advances in Computational Mathematics*, 11(2-3):253–270, 1999.

Richard K. Beatson, W.A. Light, and S. Billings. Fast solution of the radial basis function interpolation equations: Domain decomposition methods. *SIAM Journal on Scientific Computing*, 22(5):1717–1740, 2001.

Alexandre Bernardino and José Santos-Victor. Vergence control for robotic heads using log-polar images. In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '96)*, volume 3, pages 1264–1271, 1996.

Serge Beucher and Fernand Meyer. The morphological approach to segmentation: the watershed transformation. In E. Dougherty, editor, *Mathematical Morphology in Image Processing*, chapter 12, pages 433–481. Marcel Dekker, 1992.

Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

Sarah-Jayne Blakemore, Daniel M. Wolpert, and Chris Frith. Why can't you tickle yourself? *Neuroreport*, 11(11):R11–R16, 2000.

Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:567–585, 1989.

Sabri Boughorbel, J.-P. Tarel, and Nozha Boujemaa. Conditionally positive definite kernels for SVM based image recognition. In *IEEE International Conference on Multimedia and Expo (ICME '05)*, pages 113–116, 2005.

Jean-Yves Bouguet. Camera calibration toolbox for Matlab, 2008. URL `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

Gary Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 25 (11):120–126, 2000.

Jörg Bruske, Michael Hansen, Lars Riehn, and Gerald Sommer. Biologically inspired calibration-free adaptive saccade control of a binocular camera-head. *Biological Cybernetics*, 77(6):433–446, 1997.

James R. Bunch and Linda Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Mathematics of Computation*, 31(137): 163–179, 1977.

Martin V. Butz. How and why the brain lays the foundations for a conscious self. *Constructivist Foundations*, 4(1):1–37, 2008.

John Canny. A computational approach to edge detection. *IEEETransactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, Nov. 1986.

Nikolay Chumerin, Agostino Gibaldi, Silvio Sabatini, and Marc M. van Hulle. Learning eye vergence control from a distributed disparity representation. *International Journal of Neural Systems*, 20(4):267–278, 2010.

Andy Clark and Rick Grush. Towards a cognitive robotics. *Adaptive Behavior*, 7 (1):5–16, 1999.

Edoardo Datteri, Giancarlo Teti, Cecilia Laschi, Guglielmo Tamburrini, Paolo Dario, and Eugenio Guglielmelli. Expected perception: an anticipation-based perception-action scheme in robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems 2003 (IROS '03)*, volume 1, pages 934–939, 2003.

Paul Dean, John E. W. Mayhew, and Pat Langdon. Learning and maintaining saccadic accuracy: A model of brainstem–cerebellar interactions. *Journal of Cognitive Neuroscience*, 6(2):117–138, 1994.

Ralf Der and Georg Martius. From motor babbling to purposive actions: Emerging self-exploration in a dynamical systems approach to early robot development. In *From Animals to Animats 9*, pages 406–421. Springer, 2006.

Tobin A. Driscoll and Bengt Fornberg. Interpolation in the limit of increasingly flat radial basis functions. *Comput. Math. Appl*, 43:413–422, 2002.

Olivier Dubrule. Comparing splines and kriging. *Computers & Geosciences*, 10(2): 327–338, 1984.

Jean Duchon. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *R.A.I.R.O. Analyse numérique*, 10:5–12, 1976.

Jean-René Duhamel, Carol L. Colby, and Michael E. Goldberg. The updating of the representation of visual space in parietal cortex by intended eye movements. *Science*, 255(5040):90–92, 1992.

Nira Dyn, David Levin, and Samuel Rippa. Numerical procedures for surface fitting of scattered data by radial functions. *SIAM Journal on Scientific and Statistical Computing*, 7(2):639–659, 1986.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of KDD*, pages 226–231. AAAI Press, 1996.

Ronald A. Finke and Stephen M. Kosslyn. Mental imagery acuity in the peripheral visual field. *Journal of Experimental Psychology: Human Perception and Performance*, 6(1):126, 1980.

Jan Flusser. An adaptive method for image registration. *Pattern Recognition*, 25 (1):45 – 54, 1992.

Jerry A. Fodor. *The language of thought*, volume 5. Harvard University Press, 1975.

Bengt Fornberg, Tobin A. Driscoll, G. Wright, and R. Charles. Observations on the behavior of radial basis function approximations near boundaries. *Computers Math. Appl.*, 43(3-5):473–490, 2002.

John G. Fryer and Duane C. Brown. Lens distortion for close-range photogrammetry. *Photogrammetric engineering and remote sensing*, 52(1):51–58, 1986.

Albert F. Fuchs, Chris R. Kaneko, and Charles A. Scudder. Brainstem control of saccadic eye movements. *Annual Review of Neuroscience*, 8(1):307–337, 1985.

James J. Gibson. *The ecological approach to visual perception.* Routledge, 1979/1986.

Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

Chris A. Glasbey and Kantilal V. Mardia. A review of image-warping methods. *Journal of Applied Statistics*, 25(2):155–171, 1998.

Gene H. Golub and Charles F. Van Loan. *Matrix computations*. John Hopkins University Press, 1996.

Hiroaki Gomi and Mitsuo Kawato. Neural network control for a closed-loop system using feedback-error-learning. *Neural Networks*, 6(7):933–946, 1993.

Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2001.

Rick Grush. The emulation theory of representation: motor control, imagery, and perception. *Behavioral and brain sciences*, 27(3):377–396, 2004.

Rolland L. Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research*, 76:1905–1915, 1971.

Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

Germund Hesslow. Conscious thought as simulation of behaviour and perception. *Trends in cognitive sciences*, 6(6):242–247, 2002.

Germund Hesslow. The current status of the simulation theory of cognition. *Brain research*, 1428:71–79, 2012.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Heiko Hoffmann. Perception through visuomotor anticipation in a mobile robot. *Neural Networks*, 20(1):22–33, 2007a.

Heiko Hoffmann. Kernel PCA for novelty detection. *Pattern Recognition*, 40(3): 863–874, 2007b.

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, repr. with corr. edition, 1990.

Hsieh S. Hou and H. Andrews. Cubic splines for image interpolation and digital filtering. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(6): 508–517, 1978.

Masao Ito. Control of mental activities by internal models in the cerebellum. *Nature Reviews Neuroscience*, 9(4):304–313, 2008.

Bernd Jaehne. *Digitale Bildverarbeitung*. Springer, 2005.

Martin Jägersand. Image based view synthesis of articulated agents. In *IEEE Confonference on Computer Visision and Pattern Recognition 1997 (CVPR '97)*, pages 1047–1053, 1997.

Lorenzo Jamone, Lorenzo Natale, Francesco Nori, Giorio Metta, and Giulio Sandini. Autonomous online learning of reaching behavior in a humanoid robot. *International Journal of Humanoid Robotics*, 9(3):1–26, 2012.

Marc Jeannerod. The representing brain: Neural correlates of motor intention and imagery. *Behavioral and Brain sciences*, 17(2):187–201, 1994.

Marc Jeannerod. Mental imagery in the motor context. *Neuropsychologia*, 33(11): 1419–1432, 1995.

Marc Jeannerod. Neural simulation of action: a unifying mechanism for motor cognition. *Neuroimage*, 14(1):S103–S109, 2001.

Ian Jolliffe. *Principal Component Analysis.* Springer, 1986.

Béla Julesz. *Foundations of Cyclopean Perception.* MIT Press, 1971.

Alexander Kaiser, Wolfram Schenck, and Ralf Möller. Mental imagery in artificial agents. In Thomas Villmann and Frank-Michael Schleif, editors, *Machine Learning Reports 04/2010*, number MLR-04-2010, pages 25–32, 2010a.

Alexander Kaiser, Wolfram Schenck, and Ralf Möller. Distance functions for local PCA methods. In *ESANN 2010 proceedings—European Symposium on Artificial Neural Networks*, pages 469–474, Bruges (Belgium), 2010b. d-side publications.

Alexander Kaiser, Wolfram Schenck, and Ralf Möller. A model architecture for mental imagery. In B. Kokinov, A. Karmiloff-Smith, and N. J. Nersessian, editors, *European Perspectives on Cognitive Science*, Sofia, 2011. New Bulgarian University Press.

Alexander Kaiser, Wolfram Schenck, and Ralf Möller. Stereo matching and depth perception by visual prediction. In Alessandro G. Di Nuovo, Vivian M. de la Cruz, and Davide Marocco, editors, *Proceedings of the SAB Workshop on "Artificial Mental Imagery"*, pages 7–10, Odense (Danmark), 2012.

Alexander Kaiser, Wolfram Schenck, and Ralf Möller. Solving the correspondence problem in stereo vision by internal simulation. *Adaptive Behavior*, 21(4):239–250, 2013.

Amir Karniel. Three creatures named forward model. *Neural Networks*, 15(3): 305–307, 2002.

Mitsuo Kawato and Daniel M. Wolpert. Internal models for motor control. *Sensory Guidance of Movement*, 218:291–307, 1998.

Stephen M. Kosslyn, Nathaniel M. Alpert, William L. Thompson, Vera Maljkovic, Steven B. Weise, Christopher F. Chabris, Sania E. Hamilton, Scott L. Rauch, and Ferdinando S. Buonanno. Visual mental imagery activates topographically organized visual cortex: PET investigations. *Journal of Cognitive Neuroscience*, 5(3):263–287, 1993.

Stephen M. Kosslyn, William L. Thompson, and Nathaniel M. Alpert. Neural systems shared by visual imagery and visual perception: A positron emission tomography study. *Neuroimage*, 6(4):320–334, 1997.

Michael Kuperstein. Neural model of adaptive hand–eye coordination for single postures. *Science*, 239(4845):1308–1311, 1988.

George Lakoff and Mark Johnson. *Metaphors we live by.* Chicago: The University of Chicago Press, 1980.

Claus Lamm, Christian Windischberger, Ulrich Leodolter, Ewald Moser, and Herbert Bauer. Evidence for premotor cortex activity during dynamic visuospatial imagery from single-trial functional magnetic resonance imaging and event-related slow cortical potentials. *Neuroimage*, 14(2):268–283, 2001.

Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 1998.

Miguel Lourenço, João Pedro Barreto, and Francisco Vasconcelos. sRD-SIFT: Keypoint detection and matching in images with radial distortion. *IEEE Transactions on Robotics*, 28(3):752–760, June 2012.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, pages 281–297, 1967.

Hugo G. Marques and Owen Holland. Architectures for functional imagination. *Neurocomputing*, 72(4):743–759, 2009.

David Marr and Tomaso Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 204 (1156):301–328, 1979.

Lawrence E. Mays. Neural control of vergence eye movements: convergence and divergence neurons in midbrain. *Journal of Neurophysiology*, 51(5):1091–1108, 1984.

Bartlett W. Mel. MURPHY: A robot that learns by doing. In *Neural information processing systems*, pages 544–553. American Institute of Physics, 1988.

Bartlett W. Mel. A connectionist model may shed light on neural mechanisms for visually guided reaching. *Journal of cognitive neuroscience*, 3(3):273–292, 1991.

Charles A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2(1):11–22, 1986.

Giulio Milighetti, Luca Vallone, and Alessandro De Luca. Adaptive predictive gaze control of a redundant humanoid robot head. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '11)*, pages 3192–3198. IEEE, 2011.

Ralf Möller. Perception through anticipation. a behaviour-based approach to visual perception. In *Understanding Representation in the Cognitive Sciences*, pages 169–176. Springer, 2000.

Ralf Möller and Heiko Hoffmann. An extension of neural gas to local PCA. *Neurocomputing*, 62:305–326, 2004.

Ralf Möller and Axel Könies. Coupled principal component analysis. *IEEE Transactions on Neural Networks*, 15(1):214–222, 2004.

Ralf Möller and Wolfram Schenck. Bootstrapping cognition from behaviora computerized thought experiment. *Cognitive Science*, 32(3):504–542, 2008.

Luis Montesano, Manuel Lopes, Alexandre Bernardino, and José Santos-Victor. Learning object affordances: From sensory–motor coordination to imitation. *Robotics, IEEE Transactions on*, 24(1):15–26, 2008.

John Moody and Christian J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

Allen Newell and Herbert A. Simon. *GPS, a program that simulates human thought.* Defense Technical Information Center, 1961.

Shun Nishide, Tetsuya Ogata, Jun Tani, Kazunori Komatani, and Hiroshi G. Okuno. Predicting object dynamics from visual images through active sensing experiences. *Advanced Robotics*, 22(5):527–546, 2008.

Hiroharu Noda. Cerebellar control of saccadic eye movements: Its neural mechanisms and pathways. *The Japanese Journal of Physiology*, 41(3):351–368, 1991.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

Francesco Orabona, Giorgio Metta, and Giulio Sandini. A proto-object based visual attention model. In Lucas Paletta and Erich Rome, editors, *Attention in Cognitive Systems. Theories and Systems from an Interdisciplinary Viewpoint*, volume 4840 of *Lecture Notes in Computer Science*, pages 198–215. Springer Berlin Heidelberg, 2007.

J. Kevin O'Regan and Alva Noë. A sensorimotor account of vision and visual consciousness. *Behavioral and brain sciences*, 24(5):939–972, 2001.

Giovanni Pezzulo, Lawrence W. Barsalou, Angelo Cangelosi, Martin H. Fischer, Ken McRae, and Michael J. Spivey. Computational grounded cognition: a new alliance between grounded cognition and computational modeling. *Frontiers in psychology*, 3:612–612, 2011.

Giovanni Pezzulo, Matteo Candidi, Haris Dindo, and Laura Barca. Action simulation in the human brain: twelve questions. *New Ideas in Psychology*, 31(3): 270–290, 2013.

Rolf Pfeifer and Christian Scheier. *Understanding intelligence*. MIT press, 2001.

David Philipona, J. Kevin O'Regan, and Jean-Pierre Nadal. Is there something out there? Inferring space from sensorimotor dependencies. *Neural Computation*, 15 (9), 2003.

Friedemann Pulvermüller and Luciano Fadiga. Active perception: sensorimotor circuits as a cortical basis for language. *Nature Reviews Neuroscience*, 11(5): 351–360, 2010.

Zenon W. Pylyshyn. The imagery debate: Analogue media versus tacit knowledge. *Psychological review*, 88(1):16, 1981.

Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning series. The MIT Press, 2005.

Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In *IEEE International Conference on Neural Networks*, volume 1, pages 586–591, 1993.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1: Foundations, pages 318–362. The MIT Press, Cambridge, MA, USA, 1986.

Ryo Saegusa, Sophie Sakka, Giorgio Metta, and Giulio Sandini. Sensory Prediction Learning –How to Model the Self and the Environment–. In *12th IMEKO TC1 & TC7 Joint Symposium on "Man Science & Measurement" (IMEKO '08)*, pages 269–275, 2008.

Wolfram Schenck. *Adaptive Internal Models for Motor Control and Visual Prediction.* MPI Series in Biological Cybernetics. Logos Verlag, Berlin, 2008.

Wolfram Schenck. Robot studies on saccade-triggered visual prediction. *New Ideas in Psychology*, 31(3):221–238, 2013.

Wolfram Schenck and Ralf Möller. Staged learning of saccadic eye movements with a robot camera head. In H. Bowman and C. Labiouse, editors, *Connectionist Models of Cognition and Perception II*, pages 82–91. World Scientific, New Jersey, London, 2004.

Wolfram Schenck and Ralf Möller. Training and application of a visual forward model for a robot camera head. In M. V. Butz, O. Sigaud, G. Pezzulo, and G. Baldassarre, editors, *Anticipatory Behavior in Adaptive Learning Systems: From Brains to Individual and Social Behavior*, number 4520 in Lecture Notes in Artificial Intelligence, pages 153–169. Springer, Berlin, Heidelberg, New York, 2007.

Wolfram Schenck, Heiko Hoffmann, and Ralf Möller. Grasping to extrafoveal targets: A robotic model. *New Ideas in Psychology*, 29(3):235–259, 2011.

Wolfram Schenck, Hendrik Hasenbein, and Ralf Möller. Detecting affordances by mental imagery. In Alessandro G. Di Nuovo, Vivian M. de la Cruz, and Davide Marocco, editors, *Proceedings of the SAB Workshop on "Artificial Mental Imagery"*, pages 15–18, Odense (Danmark), 2012.

Charles A. Scudder, Chris R. Kaneko, and Albert F. Fuchs. The brainstem burst generator for saccadic eye movements. *Experimental Brain Research*, 142(4): 439–462, 2002.

Roger N. Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. *Science*, 171(3972):701–703, 1971.

Tomohiro Shibata and Stefan Schaal. Biomimetic gaze stabilization based on feedback-error-learning with nonparametric regression networks. *Neural Networks*, 14(2):201–216, 2001.

Mark W. Spong and Mathukumalli Vidyasagar. *Robot dynamics and control.* John Wiley & Sons, 2008.

Henrik Svensson, Anthony F. Morse, and Tom Ziemke. Neural pathways of embodied simulation. In *Anticipatory Behavior in Adaptive Learning Systems*, pages 95–114. Springer, 2009.

Jun Tani and Stefano Nolfi. Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. *Neural Networks*, 12(7):1131–1141, 1999.

Wolfgang M. Theimer and Hanspeter A. Mallot. Phase-based binocular vergence control and depth reconstruction using active vision. *CVGIP: Image Understanding*, 60(3):343–358, 1994.

Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

Rudolph Van Der Merwe, Arnaud Doucet, Nando De Freitas, and Eric Wan. The unscented particle filter. In *NIPS*, pages 584–590, 2000.

Erich von Holst and Horst Mittelstaedt. Das Reafferenzprinzip. *Naturwissenschaften*, 37(20):464–476, 1950.

Dirk Walther and Christof Koch. Modeling attention to salient proto-objects. *Neural Networks*, 19(9):1395–1407, 2006.

Yonatan Wexler, Andrew W. Fitzgibbon, and Andrew Zisserman. Learning epipolar geometry from image sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 209–216, 2003.

Margaret Wilson. Six views of embodied cognition. *Psychonomic bulletin & review*, 9(4):625–636, 2002.

Marco Wischnewski, Anna Belardinelli, Werner X. Schneider, and Jochen J. Steil. Where to look next? combining static and dynamic proto-objects in a TVA-based model of visual attention. *Cognitive Computation*, 2(4):326–343, 2010.

George Wolberg. *Digital Image Warping.* IEEE Computer Society Press, Los Alamitos, CA, USA, 1st edition, 1994.

Daniel M. Wolpert and Mitsuo Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7):1317–1329, 1998.

Daniel M. Wolpert, Zoubin Ghahramani, and Michael I. Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

Daniel M. Wolpert, R. Chris Miall, and Mitsuo Kawato. Internal models in the cerebellum. *Trends in cognitive sciences*, 2(9):338–347, 1998.

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, 2003.

Jerry J. Yokono and Tomaso Poggio. Oriented filters for object recognition: an empirical study. In *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 755–760, 2004.

Tom Ziemke, Dan-Anders Jirenhed, and Germund Hesslow. Internal simulation of perception: a minimal neuro-robotic model. *Neurocomputing*, 68:85–104, 2005.