Lehrstuhl für Realzeit-Computersysteme

Visual Tracking and Grasping of a Dynamic Object: From the Human Example to an Autonomous Robotic System

Michael Sorg

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr.-Ing. Klaus Diepold
Prüfer der Dissertation:	1. Univ.-Prof. Dr.-Ing. Georg Färber
	2. Hon.-Prof. Dr.-Ing. Gerd Hirzinger

Die Dissertation wurde am 19.02.2003 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 16.07.2003 angenommen.

München, den 11.11.2002

First of all I want to thank my advisor Prof. Georg Färber for having given me the opportunity to work at a real thrilling topic. His manner of not pushing me in a certain direction, but leaving enough room to develop own ideas, trying things where the outcome was unsure and leaving the freedom to decide many things "on my own" were very valuable for me. Thereby I learned a lot that I can need in the future. Many thanks to Prof. Gerd Hirzinger for his spontanous promise to help as a corrector of this thesis.

But this would not have been possible if there hadn't been Alexa Hauck. Having been already my advisor for my master's thesis, she fascinated me for "hand-eye coordination" and finally was developing valuable ideas and a plan for this thesis. Besides these "hard facts" she was the best advisor and colleague that I can think of. It was always fun and very motivating to work together. I often think where I would be now if I hadn't knocked on her door ...

Special thank goes to Thomas Schenk and Andreas Häussler from the "neuro" team. Besides the fact that they brought "light into dark" when we were discussing neuroscientific literature, their interest in robotics and my work was a special motivation and gave me the feeling that I (and my students) were doing something valuable.

The work would never have been possible without all the students working with me during their diploma thesis. They developed really great ideas and were providing all the necessary pieces to let MinERVA *catch. Many thanks to Christian Maier, Hans Oswald, Georg Selzle, Jan Leupold, Thomas Maier, Jean-Charles Beauverger and Sonja Glas. That many of them ended up as my colleagues emphasizes their "good job".*

At the lab I want to thank all the colleagues from the Robot Vision Group for providing a real good atmosphere. Special thanks go to Georg Passig who not only supported me in any problems concerning the robot but had always time to discuss any other problem concerning work and "the world". Thanks to all the people of the Schafkopfrunde. This was (and is!) great fun.

Again special thank to Johanna Rüttinger. Without her debugging thousands lines of other peoples code, integrating new one and finally providing huge amounts of experimental data, the experimental part of this thesis would have been poor. Or to be more precise: without her help I think MinERVA *would have never catched anything!*

Last but not least I want to thank my parents and my brother. Not knowing what I was exactly doing but always trusting that I would do it right was very pleasant.

Michael Sorg

Abstract

In this thesis a robotic hand-eye system capable of visual tracking of a moving object and reaching out to grasp this object using a robotic manipulator is described. A noticeable number of successful methods performing those tasks has been published (also recently) and impressive demonstrations have been shown thereby. Nevertheless, there is still one system that is superior to all the demonstrated ones: the human. Humans perform catching tasks with a high degree of accuracy, robustness and flexibility. Therefore this thesis investigates results of neuroscience and applies them to design a robotic hand-eye system for grasping a moving object. From the experimental data of human catching movements it can be derived that humans are performing different subtasks during catching: tracking of the target object, prediction of the future target trajectory, determination of an interaction point in space and time, and execution of an interceptive arm movement. Thereby the different subtasks are performed in parallel and the coordination between "hand and eye" is reactive: the human can easily adapt and correct its interceptive movement triggered either by (sudden) changes in the targets trajectory or by refinement of the predicted object trajectory and the hand-target interaction point.

Transferring knowledge gained by the neuroscientists to robotics is often difficult since the underlying physical systems are very different. Nevertheless there exist interesting models or experimental data that offer the possibility of transfer. In this thesis for two of the above noticed subtasks biological concepts are deployed: for visual tracking and for the execution and timing of the interceptive catching movement.

For the tracking subtask the used visual sensors are closely related to those found in the human brain: Form, color and motion (optic flow). Through analysis of human visual processing from the eye up to the visual cortex three main concepts could be separated: parallel information flow, pre-attentive processing and reentry of information. These mechanism allow the human the optimal utilization of the presented information before attention is put on a certain stimulus. This can be seen as a form of image pre-processing. Integrating those concepts in a robotic hand-eye system improves image pre-processing for still images as well as in a tracking task noticeable.

For the determination of hand-target interaction points and the timing of the arm movement relative to the target motion a human-like behavior is adopted. Based on experimental data a four phasic model for the determination of interaction points and the generation

of appropriate via-points for a robotic manipulator for reach-to-catch motions is developed. This model satisfies the purpose of flexibility: depending on the current object motion (and prediction) the via-points are adapted and the interceptive movement is corrected during motion execution.

The validity of these concepts is investigated thoroughly in simulations. Together with modules for target object prediction (using autoregressive models), for the determination of grasping points and for robot arm motion control a robotic hand-eye system is demonstrated that proves its practicability in real experiments performed on the experimental hand-eye system MinERVA.

Contents

Chapter 1

Introduction

1.1 Motivation

Over the last decade, using sensor information to control robots has become a very popular field of research, as it promises to lead to the design of *autonomous robots*. In contrast to their preprogrammed industrial counterparts, autonomous robots are to be able to deal with unexpected events, e.g. obstacles, misplaced objects or objects in motion. On the one hand, this is especially important for *personal robots* as they are operating in a world which is not adapted to the needs of machines. On the other hand, dealing with objects in motion, which is the main concern in this thesis, might in the future also be interesting for industrial robots: not stopping a conveyor belt while accurately placing or picking parts on/from it by a robot might have different advantages. First, one could think of energy reduction: starting and stopping a belt costs more energy than letting it continously run. Secondly, less wastage: starting and stopping a belt is more mechanical wearing. Thirdly, less calibration effort: by less mechanical wearing the calibration cycles will drop. And finally, a economic gain: having less mechanical wearing can lead to use cheaper parts (dimensioning of the system).

Nowadays, vision is by far the most commonly used sensor in robotics, due to the fact that cameras are cheap and versatile. An additional advantage is that vision mimics the most important human sense, thus making it possible for the human operator to understand intuitively the information the robot gets.

In the field of visually controlled robot manipulators, two strategies have been proposed: *look-then-move systems* and *visual servoing* systems. The former systems try to determine the object's pose from the visual input as accurately as possible and then move the robot appropriately. Unfortunately, its accuracy heavily depends on the accuracy of the sensor/robot calibration and of the sensor itself.

The later systems have been proposed as an alternative approach to overcome these prob-

lems. Here, visual information about the current position of the manipulator is used in a feedback control loop to guide the robot.

For both approaches, there exists a large number of successfully realized hand-eye systems which cope very well with a specific problem. These systems deal with grasping of *static objects* as well as with hitting or catching of *dynamic objects*.

Yet, in comparison with the human example they all show a considerable lack of performance possibilities, robustness and flexibility. The main difference in the context of *grasping* is that humans can grasp successfully using only little visual information, for example looking at the target once with only one eye. More visual information, for example a view of the hand or stereo vision, results in a more precise and efficient grasp.

In the context of *catching* the situation is quite similar. Humans can catch an object even if the have seen only a small part of the object's trajectory. Additionally, they can react very flexible on sudden changes of the objects motion what implies that the movement is not pre-programmed for the whole motion.

One has to conclude that there might be something to learn for robotics research by taking a closer look at the human example. Instead of refining control methods or speeding up image processing due to massive hardware support, this thesis therefore sets out to explore the results of neuroscience and to apply them in the design of a robotic hand-eye system with special emphasis on *catching*.

1.2 Context

The work presented in this thesis was part of an interdisciplinary project on human and robotic hand-eye coordination, which in its turn was part of a Special Research Program on "*Sensorimotor – Analysis of biological systems, modeling, and medical-technical applications*" (SFB 462) funded by the *Deutsche Forschungsgemeinschaft* (DFG). Our project (TP C_1) was a cooperation of the *Institute for Real-Time Computer Systems* (Technische Universität München) and the *Neurological Clinic* (Ludwig-Maximilians-Universität München).

From the start, the common goal of the project was to develop a model of hand-eye coordination that, on the one hand, fits and predicts experimental data on human reaching, grasping and catching. On the other hand, the model should be used to control a robot. The clinical part mainly concentrated on analyzing human catching movements, to answer the question which visual information is used to control which parameters of motion.

The technical part used this knowledge on the one hand to develop and synthesize the modules necessary for autonomous robotic *grasping* (first project period). Developed models, experiments and results to prove the models were shown very thoroughly in the thesis of my predecessor in this project (see [Hau99] for more details).

On the other hand the experimental data from catching experiments (from the literature as well as from the project partners) served as a basis to develop modules for robotic *catching* (second project period). Developed models, experiments and results are the content of this thesis.

The project definition already limited the extent of the "model" that had to be developed in both cases: Due to the different hardware addressed in the clinical and the technical part, a common model was only possible on higher levels of abstraction. In the terminology of Marr [Mar82], an information processing device should be analyzed at the levels of computational theory, representation/algorithm, and hardware implementation; in the project, the model of hand-eye coordination was restricted to the first two levels.

1.3 Contributions and Limitations

This dissertation contributes to research in the field of robotic hand-eye coordination in three ways: First, it provides a fairly comprehensive survey of the neuroscientific literature related to: (a) human visual processing from the retina up to the visual cortex and (b) human hand-eye coordination for the two cases of interaction with a static target (grasping) and interaction with a dynamic target (catching). Secondly, it provides a model of visual processing for a robot, which is derived from a model of human visual processing in the visual cortex. And thirdly, a model for hand-target interaction for a robotic manipulator taking into account experimental results of human catching experiments is developed.

The obvious limitation of this work is that it addresses "only" reach to catch movements, i.e. purely translational movements, and leaves out the orientation of the hand. A way how to flexible control the orientation of a robotic manipulator is derived (and also implemented), but for the catching experiments orientation was left constant. This was due to the fact that the determination of the 3D orientation of the object to catch was an unsolved problem.

1.4 Organization of the Dissertation

The main problems researchers from the area of robotics encounter when trying to learn from neuroscience are that the two sciences are concerned with very different physical systems and speak about them using very different languages. The former problem rules out a direct copying of results, the second impedes their transfer. Nevertheless the transfer is possible if one is trying to exploit the main principles that lie behind a biological concept. This is what was tried in this thesis.

The dissertation can be separated into four main parts: the first part (Chap. 2) is concerned with a review on neuroscientific literature dealing with human visual processing (Section 2.1) and human hand-eye coordination for grasping (Section 2.2.1) and catching

(Section 2.2.3), respectively. Current models for visual processing (Section 2.1.2), reach-to-grasp (Section 2.2.2) and reach-to-catch movements (Section 2.2.4) are also reviewed and shortly analysed.

The second part (Chap. 3) is concerned with a review of methods for visual tracking of moving objects (Section 3.2.1) using different sensor modalities (Section 3.2.1.1 for form, Section 3.2.1.2 for color and Section 3.2.1.3 for motion), prediction of time series (Section 3.3) and a review on literature and methods for robotic manipulator control for interaction with static (Section 3.4.1) and moving targets (Section 3.4.2).

The third part (see Chap. 4) describes the methods and algorithms that were developed in this thesis. Those methods and algorithmns were either developed from (a) knowledge from the common robotics literature (see Section 4.1 for models) or (b) extensions of methods described in Chap. 3 (see Section 4.2 for tracking, and Section 4.5 for motion prediction) or (c) developments derived from the analysis of results described in Chap. 2 (see Section 4.3 for sensor fusion and integration, Section 4.6 for robot arm motion control and Section 4.7 for hand-target interaction). Finally, implementation details serving as the basis for experiments are shortly described (Section 4.8).

The fourth and last part (Chap. 5) summarizes results obtained by simulations (using *MATLAB* and *Simulink*) and real robot experiments performed on the experimental robotic hand-eye system MinERVA.

The connection and interaction between the different methods described in the third part (Chap. 4) gets obvious by looking at the proposed hand-eye *system architecture* in Figure 1.1.

As a guideline for reading the reader should notice that all relevant chapters (Chap. 2, Chap. 3, Chap. 4 and Chap. 5) are designed to have internally the same or a similar structure reflecting the main topics of the thesis: vision (or visual tracking) and hand-target interaction. Naturally, there are not always one to one correspondences. This is on the one hand because neuroscience research lacks of explanations (e.g. for "how" humans predict), on the other hand contributions to robotic "needs" have to be made when necessary (e.g. the determination of grasping points). Nevertheless the interfaces between the methods (and modules in Figure 1.1) are thin and easy and the transported information is obvious.

This allows the reader to select only the parts for reading that are interesting for her/him. Additionally, to support this kind of "cross reading", summaries and discussions are given at the end of each section which are sufficient to understand the content of the read section as well as the "input/output" relationship between the sections. To keep the "whole" in view at any time one can always refer back to the system architecture.

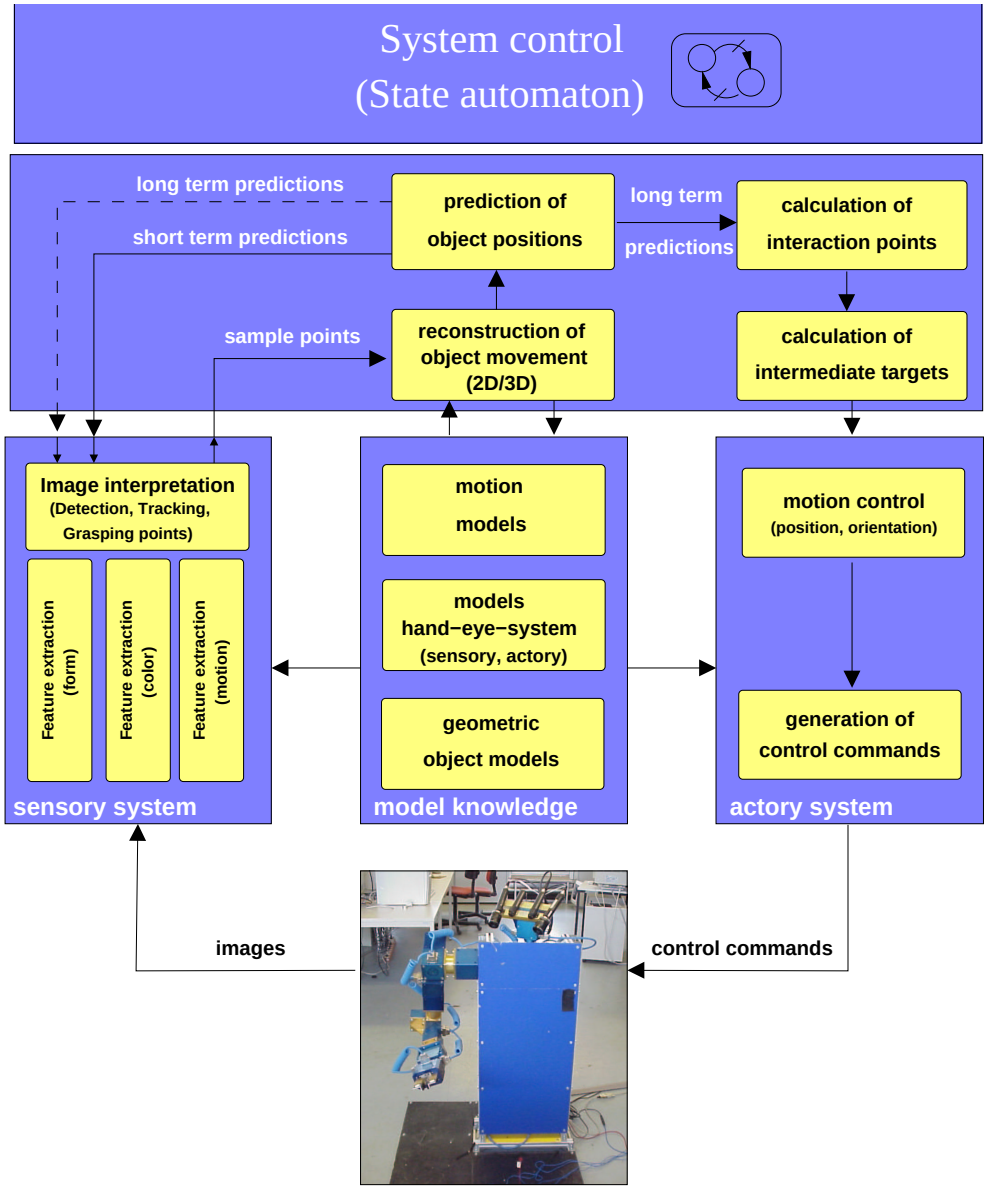


Figure 1.1: System Architecture

Chapter 2

Neuroscience

This chapter is concerned with a review on neuroscientific literature dealing with human visual processing (Section 2.1) and human hand-eye coordination for grasping (Section 2.2.1) and catching (Section 2.2.3), respectively. Current models for visual processing (Section 2.1.2), reach-to-grasp (Section 2.2.2) and reach-to-catch movements (Section 2.2.4) are also reviewed and shortly analysed.

2.1 Vision

Since nature is a source of inspiration for the design of technical systems, taking a closer look at the functionality of e.g. human or animal behavior can be very useful. With this intention information about human vision was collected, i.e. about pathways of visual information from the eye up to higher cortical areas. Interested in how an obviously well working system has been developed by nature, a way was sought of how principles of the human visual processing can be used to improve a robotic vision system.

The rest of the section is organized as follows: The first paragraphs (Section 2.1.1) describe the main components of the human eye, i.e. the *retina*, the *photo-receptors* and the *Ganglion cells*. The succeeding paragraph describes the way of the visual information over the *Lateral Geniculate Nucleus (LGN)* to the *visual cortex*; finally, models (Section 2.1.2) of its processing in the visual cortex, namely the *parallel information flow* and the *reentry hypothesis* are presented. In the last paragraph a model of visual integration in higher cortical areas using the concept of *feature maps* and *master map* is presented.

2.1.1 Anatomy of the Human Visual System

Retina Light enters the eye through the cornea and the lens and is projected onto the retina. The lens can change its size by muscles to refract the light waves for focusing. Figure 2.1 shows a schematic drawing of the main components of the eye.

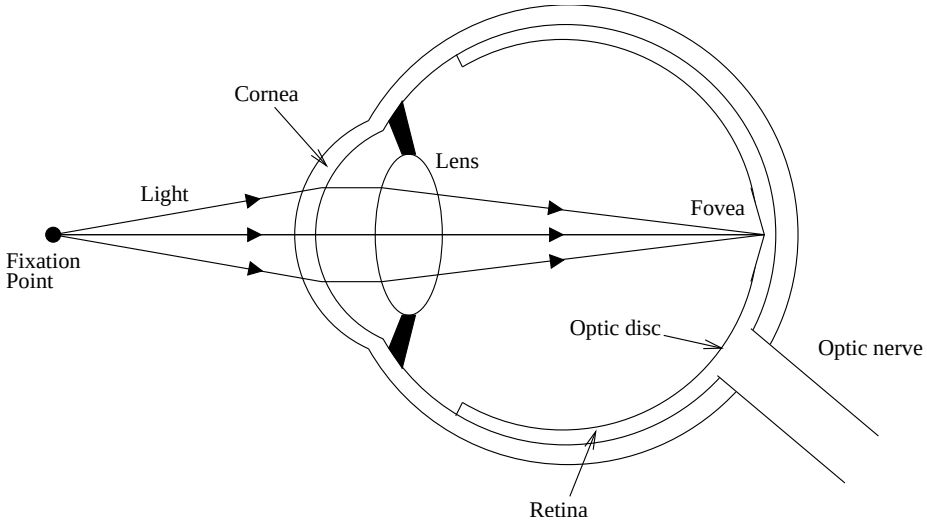


Figure 2.1: The human eye

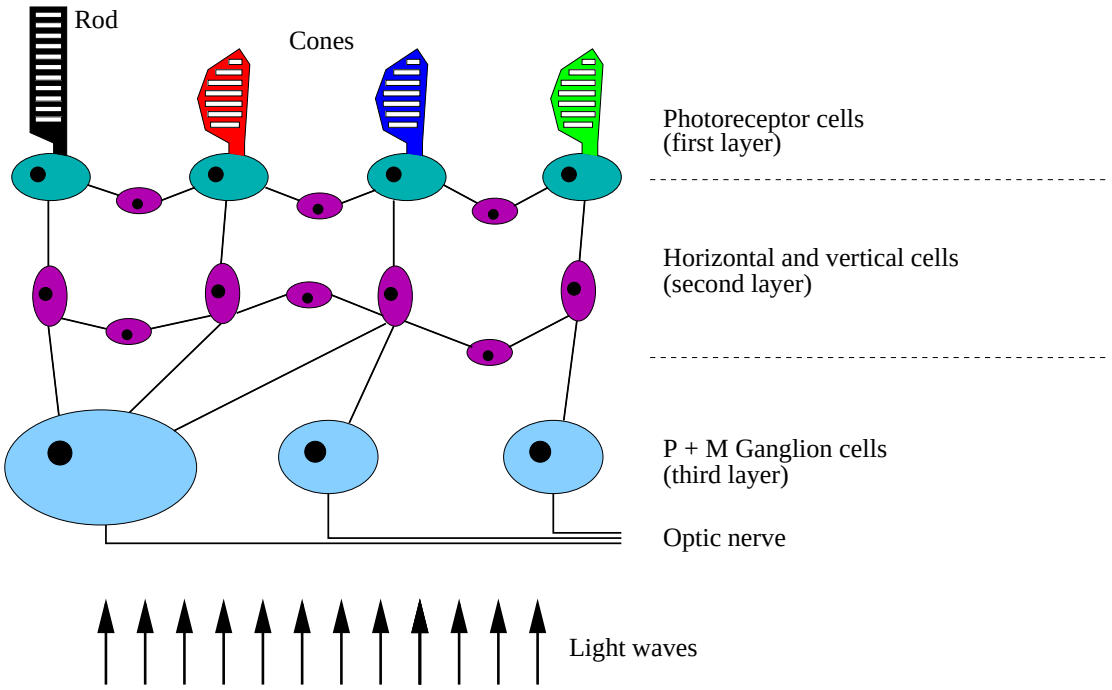


Figure 2.2: Layered retina structure

The retina has a layered structure (see Figure 2.2):

- Photo-receptor cells transform light wave energy into electric signals.
- A network of horizontal and vertical cells connects these signals to the ganglion cells.
- Ganglion cells interpret incoming signals and forward their results to the Lateral Geniculate Nucleus or LGN through the optic nerve.

Interestingly light has to pass through all of the ganglion and network cell layers to reach the photo-receptor cells. Those photo-receptor cells exist in different types, allowing for *daylight vision* (cone vision) and *night vision* (rod vision)[1].

The network layer is responsible for switching between daylight and night vision. Additionally, it controls the size of receptive fields of the Ganglion cells.

There exist two special areas of the retina (see Figure 2.1):

- The *optic disc* is the connection of the eye to the optical nerve. This nerve enters the eye here and leaves no space for photo-receptor cells, making the eye blind at this area.
- The *fovea* is the point of maximum concentration of photo-receptor cells on the retina, leading to a very high spatial resolution of the fovea. Additionally, Ganglion cells are moved aside at this area so that light does not have to pass through any other layer to reach the photo-receptor cells.

Photo-receptors The human retina consists of two types of photo-receptor cells: rods and cones. While cones are responsible for daylight vision, rods serve for night vision. Rods function in dim light that is present at dusk or at night, when most stimuli are too weak to excite the cone system.

Under daylight conditions only the cone system is active, while the rod system is saturated. When illumination conditions change from bright daylight to darkness, first a mixture of both cell systems is utilized. If light intensity gets too low for the cone system it is inactivated, while the rod system is completely active. This mechanism is controlled by the network layer of the retina.

Cones perform better than rods in all visual tasks, except for the detection of dim stimuli. They provide a better temporal and spatial resolution. There exist three types of cones, each of them having a different spectral sensitivity characteristic. Depending on their sensitivity cones are called blue (short wavelength), green (medium wavelength) or red (long wavelength) cones. Information provided by the cones is used in the brain to get the perception of *color vision*.

[1] In daylight vision the color of objects is perceived. During night vision only gray-level contrasts can be perceived.

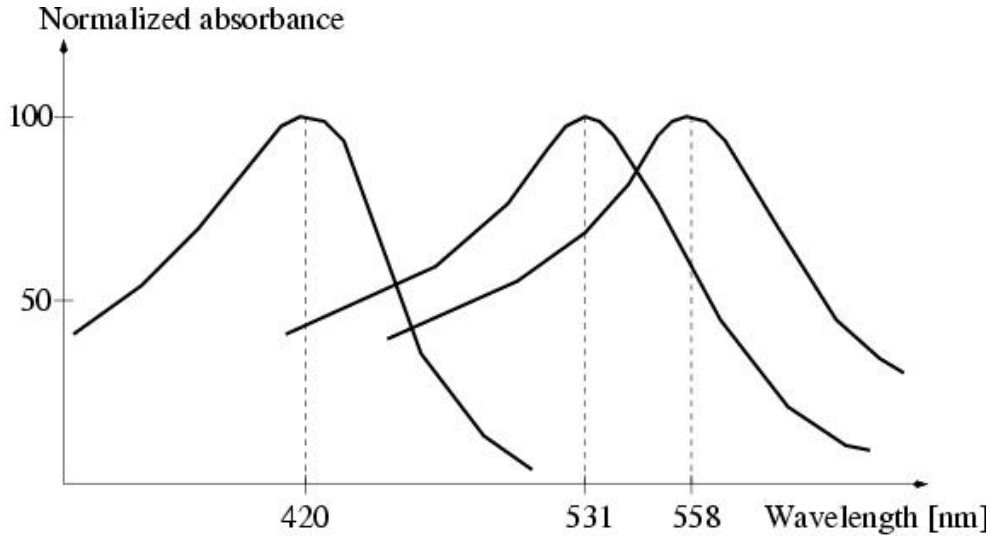


Figure 2.3: Spectral sensibility characteristics of receptor cell types

Rods contain more photosensitive visual pigment than cones, but provide only achromatic visual information. They amplify light signals more strongly than cones and saturate at daylight conditions. Although there are 20 times more rods on the human retina, the spatial resolution of the cones is higher, as most of the cones are in the small area of the fovea, whereas the rods are distributed all over the rest of the retina. Also, in the peripheral areas of the retina ganglion cells receive pooled receptor cell information. The larger these pools are, the smaller is the spatial resolution. Concerning the temporal resolution cones are also superior: The maximum flicker frequency of rods is at about 12Hz, whereas cones can detect flicker up to 55Hz.

Ganglion Cells Ganglion cells are the final output neurons of the vertebrate retina. A ganglion cell collects the electrical messages concerning the visual signal from the two layers of nerve cells preceding it in the retinal wiring scheme. Intensive preprocessing has been accomplished by the neurons of the vertical pathways (photo-receptor → bipolar → ganglion cell chain), and by the lateral pathways (photo-receptor → horizontal cell → bipolar → amacrine → ganglion cell chain) before presentation to the ganglion cell which serves as the ultimate signaler of retinal information to the brain. Ganglion cells are larger on average than most preceding retinal inter-neurons and have large diameter axons capable of passing the electrical signal, in the form of transient spike trains, to the retinal recipient areas of the brain. The optic nerve collects all axons of the ganglion cells. This bundle of more than one million fibers then passes information to the next relay station in the brain (especially the LGN) for sorting and integration into further information processing channels ([KFN02]).

Despite being the same kind of cell, Ganglion cells differ in their

- size of the receptive field and
- ON/OFF area distribution.

Receptive Field As mentioned above, there are many millions of receptors on the retina, but there is only about one million fibers in the optic nerve sending visual signals up to higher brain centers. Consequently, individual receptors do not have private lines up to the visual cortex. Rather, multiple receptors converge on to subsequent neural units on their way to the higher visual centers. This convergence results in a physiological concept known as *receptive fields*[2]. The receptive field of a ganglion cell describes the area of the perceived image that this ganglion cell is processing. This field normally is a *circular* area. The retina holds two different types of ganglion cells with different sizes of receptive fields:

- The M-cells have large receptive fields, whereas
- the P-cells only have small receptive fields.

M-cells are responsible for transporting mainly motion information, as their temporal resolution is better than their spatial resolution. Large receptive fields merge information of many photo-receptor cells and though reduce the spatial resolution.

P-cells are built to transport mainly form and color information, as their spatial resolution is higher than their temporal resolution. In the area of the fovea, some ganglion cells merge only very few receptor cells, resulting in a very high spatial resolution.

ON/OFF Area Distribution Ganglion cells use a differential system to measure light intensity. This means that no absolute signal level is transported to the brain, but the difference signal of the ON and OFF areas. The ON and OFF areas are parts of the receptive fields and have the form of concentric circles. Figure 2.4 shows the two possible constellations. The ON areas are marked with a $+$, the OFF areas with a $-$.

The more light falls on receptor cells connected to an ON area, the higher is the signal level output. The more light falls on receptor cells connected to an OFF area, the lower is the signal level output. The signal level, coded in the firing rate of neuron cells, is then being transported to the brain. A low signal level corresponds to a low fire rate, whereas a high signal level corresponds to a high fire rate. There are always both ganglion cell types with center ON/peripheric OFF and center OFF/peripheric ON present in all receptive fields.

Depending on which type of photo-receptor cell is connected to the ganglion cell, the output provides different information. The most important constellations of receptor cell connections are illustrated in Figure 2.5.

[2]To put it another way, a receptive field is the receptor area which, when stimulated, results in a response of a particular sensory neuron.

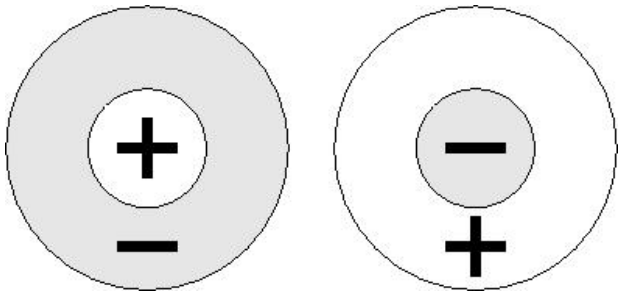


Figure 2.4: ON/OFF receptive areas

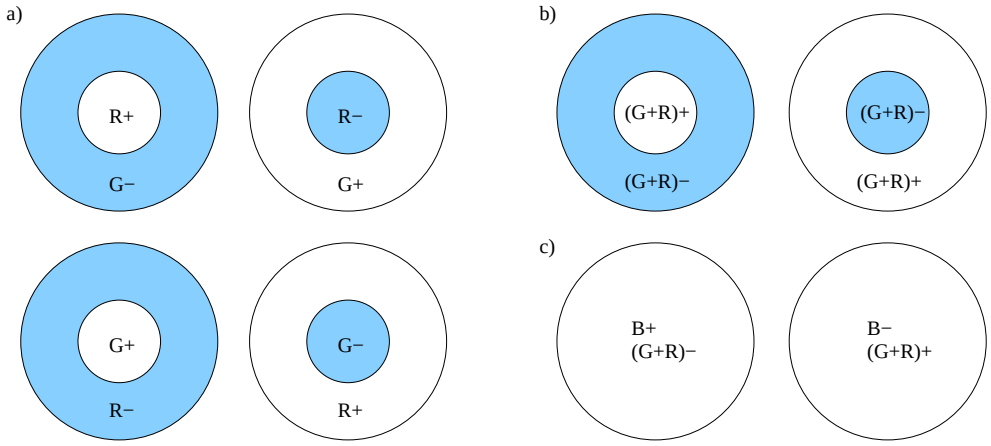


Figure 2.5: Differential ganglion cells

Cells in group (a) are called concentric single-opponent cells. They are the most common cell type and compute information about red minus green color contrast. Cells in group (b), called concentric broad-band cells, respond to illumination differences. Cells in group (c) do not have the above mentioned ON/OFF area distribution, but they compute a difference signal between the output of the blue cone cells and the illumination level for all cells in the receptive field.

Lateral Geniculate Nucleus (LGN) The LGN, located on either side of the rear end of the thalamus, lies midway between the eyes and the visual cortex and has three basic functionalities:

- to separate the information stream coming from the eye into dorsal and ventral pathways,
- to continue differential signal processing in a similar fashion as it is performed in the ganglion cells[3],

[3]Cells found in the LGN have similar receptive field sizes like the retinal ganglion cells, as only very few ganglion cells connect to LGN cells.

- to map retinal areas on visual cortex areas in relation to their importance: the important fovea area is mapped to a large area in the visual cortex whereas peripheral areas of the retina are mapped to smaller areas in the visual cortex.

Figure 2.6 shows the image information flow. After leaving the eye through the optic nerve, information is split for the left and right visual field. Border for this separation is the fovea on the retina. Both left visual fields are transported to the left LGN, whereas both right visual fields are transported to the right LGN. The figure only shows the right LGN, as processing is symmetrical for both fields. In the so called *optic chiasm* the nerve fibers of both nasal halfs cross on their way to their LGNs.

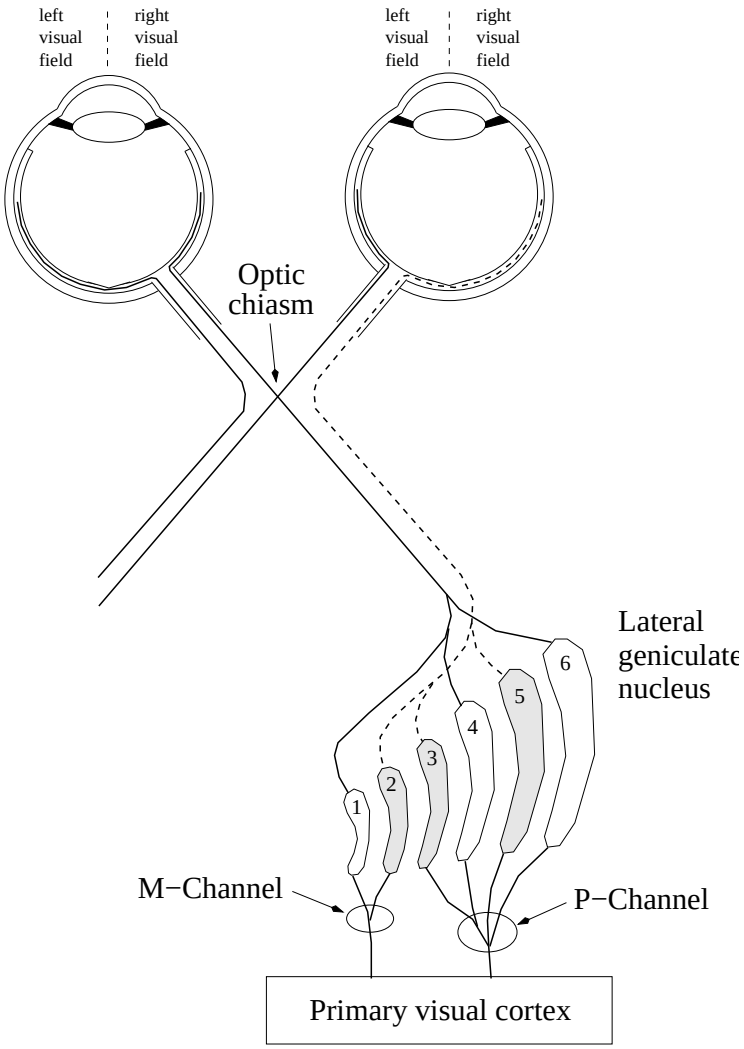


Figure 2.6: The lateral geniculate nucleus

The LGN has also a layered structure. M-cells in the retina propagate their information to the magnocellular layers, numbered 1 and 2 in Figure 2.6. Output of these layers is later

merged to the M-Channel, or *ventral pathway*, which is concerned with the initial analysis of movement of the visual image. P-cells propagate their information to the parvocellular layers, numbered $3-6$ in Figure 2.6. Output of these layers is later merged to the P-Channel, or *dorsal pathway*, which is concerned with the analysis of fine structure and color vision.

Visual Cortex The visual cortex is the input stage of the brain for incoming visual signals from the eyes. Image information is received through the LGN via the M-Channel and the P-Channel as described above.

The visual cortex has been divided into numbered areas named V1, V2, ..., with each area having a designated function. Here, only areas V1 up to V5 will be discussed, as most *known* visual processing occurs there [DE88]. Figure 2.7 shows a schematic drawing of a model describing the assumed information flow in the visual cortex.

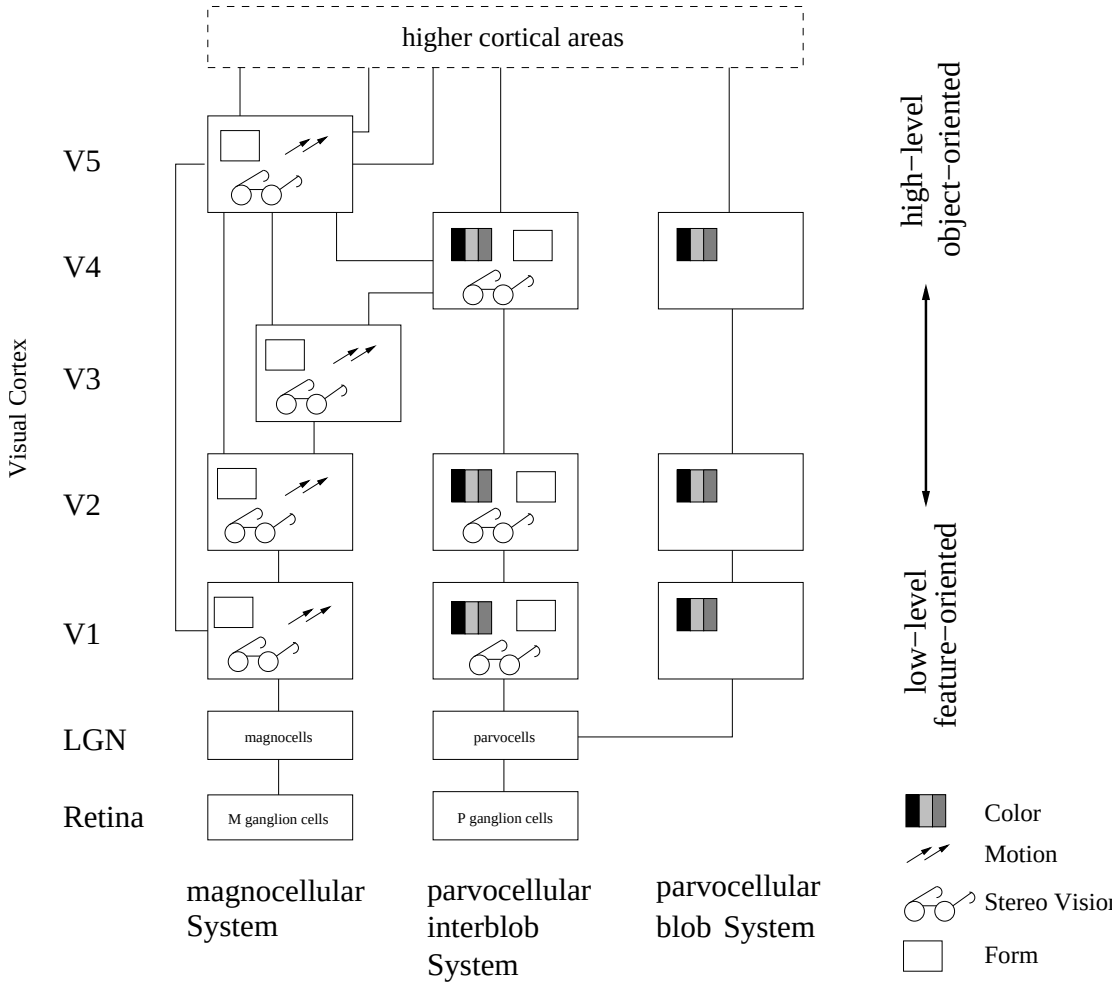


Figure 2.7: Schematic of visual cortex information flow (Adopted from [Kan91])

The mechanism of processing information with different cell types is continued in the visual cortex. V1, to which the LGN pathways connect, has three different cell types, each of them connected either to the parvocellular or to the magnocellular pathway. The three different cell types constitute three *streams of information*, with each stream processing mainly one property of the incoming information. These properties are *form, color and motion*. Stereo vision aspects are also processed in the visual cortex, but are not further treated in this thesis.

In the following we will concentrate mainly on the mechanisms of the visual cortex, since they are interesting for robotic visual processing.

2.1.2 Models of Human Visual Processing

2.1.2.1 Parallel Information Flow and Reentry of Information

Parallel Information Flow All streams process their information in parallel [LH87]. The first stream deals with motion and form information from the magnocellular system as the magnocellular system is achromatic. Form information is reduced here to gray image form processing. The second stream processes form and color-form information from the parvocellular system. The term color-form specifies the form perceived by color gradients in contrast to the afore-mentioned achromatic form information in the first stream. The third stream only treats color information. The advantage of this parallelism is that it allows the optimal utilization of the present information at a time where no attention is put on a certain stimulus ("pre-attentive processing" after Treisman and Jules in [Kan91]).

Each area forwards its processing results up to a higher area in the visual cortex. On its way up the areas, the information type changes from low-level, small sized, feature oriented local image information to clustered, high-level object-oriented information. This can be interpreted as a hierarchical processing of the information. From one stage to the next higher stage in the processing hierarchy there exist growing receptive fields of the retina, that means there is increasing visual abstraction of the retinal image (see Hubbel and Wiesel in [MK91]).

Areas V4 and V5 are the highest known areas with a specific specialization (V4 deals with color, V5 with motion information). Succeeding cortical areas and areas responsible for object recognition are still subject of intense research, as there is not much knowledge about them presently. Notwithstanding, Tononi [TSE92] states that there is no area in the brain, where information of the three streams merges together (*binding problem*). To support this hypothesis, Engel [EKaTBSS92] proposes a system of temporal correlated firing of neurons to integrate stream information in higher cortical areas.

The Reentry Hypothesis It is common to all cell types that they can forward processed output up to higher areas. These intra-stream connections are commonly acknowledged [HW65, NS99, Kan91]. Information is fed forward to higher areas, as well as *fed back* into lower cortical areas at almost all cortical levels.

In experiments to attention control it could be shown that cells in V1 increase their activity, if there is increased attention in a higher cortical region to the stimulus they detected (after Wurtz in [Kan91]). Due to this and other similar experiments it is postulated in [TSE92] that the strong connectivity between the areas of the visual cortex may support a dynamic process of *reentry* with a continuous parallel and recursive signaling between the areas. In [TSE92] a simulation model is proposed that suggests the thesis of *reentry* as a principle in cortical integration.

To support the thesis an image processing experiment is described, wherein cortical areas were simulated by neuronal networks. Results from higher networks feed their information back to lower networks and influence the *parameters* of the lower neuronal networks in the spatial domains of the recursive information. As their results were promising, it was tried to use this effect in our robot.

As information in higher cortical areas already represents clustered object information, the reentry effect is supposed to force more attention and therefore more sensitive segmentation at lower levels towards the indicated object positions through the cluster information.

2.1.2.2 Feature Maps and Integration of Information: Visual Attention

Notwithstanding the statement of Tononi that there is *no area in the brain where information of the three streams merges together*, the question that arises is how the information about color, motion and form is organized into a cohesive perception. Obviously, the information of the groups of cells, each of which processing a distinct property, must be brought together in temporary association i.e. there must be a mechanism whereby the brain associates the processing carried out independently in different cortical areas. This mechanism, as yet unspecified, is called the *binding problem*.

In psycho-physical studies it was shown (Treisman and Julesz) that the formation of these associations requires *attention*. They also found that distinctive boundaries are created from the elementary properties: brightness, color and orientation of line. If these boundaries are made up of elements that are clearly different from the rest of the image they *pop out* almost automatically within few milliseconds. From different experiments and the resulting observations Treisman and Julesz suggested that there are two distinct processes in visual perception. An initial *pre-attentive* process (see above!) acts as a rapid scanning system and is only concerned with the *detection* of objects. This process rapidly scans the objects' overall features and encodes the useful elementary properties of the scene: color, orientation, direction of movement etcetera. At this point, variation in a simple property may be discerned as a border or contour, *but complex differences in combinations*

of properties are not detected. Treisman proposes that different properties are encoded in different *feature maps* in different brain regions. The later *attentive process* directs attention to specific features of an object, *selecting and highlighting features that are initially segregated in the separate feature maps*. This attentive process takes a *winner-takes-all* strategy, whereby the salient features of the object are emphasized and attended to (*focus of attention*) while other features and other objects are inhibited or ignored.

But how does object recognition occur? Recognition requires attention. Stated in neural terms Treisman and Julesz argue that cells in different feature maps (e.g. maps for color, motion, and form) must be scanned, and associated with our *memory* of that object. To solve this binding problem, Treisman postulates that there may be a *master* or *saliency map* that codes only for key aspects of the image (see Figure 2.8). This master map receives input from all feature maps but abstracts only those features in each map that distinguish the object of attention from its surroundings. Once these salient feature have been selected, the information associated with this location in the master map is retrieved back to the individual feature maps. In this way the master map selects the *details* in the feature map that are essential for *attentive recognition*. Recognition finally occurs when these salient locations in different feature maps are associated or bound together.

2.1.3 Summary

In this section the way of visual information from the retina up to the visual cortex and its processing in higher cortical areas was presented. Thereby following results can be summarized: the *retina* consists of two different cell types, cones and rods, responsible for day-light and night vision. Rods provide only achromatic visual information and saturate at daylight conditions. Cones have a higher spatial resolution as most of them are on the fovea, whereas rods are equally distributed all over the retina. Additionally, the temporal resolution of cones is superior. The emphasis of perception is on color and form perception in the area of the fovea having a high spatial resolution, and on motion perception in the peripheral regions having a high temporal resolution. Information coming from the retina is grouped in so called receptive fields.

The *visual cortex* is the input stage of the brain for incoming visual signals from the eyes. It has been divided into numbered areas named V1, V2, ..., with each area having a designated function. The visual cortex is further characterized by a *parallel feed-forward processing* in three *streams of information*, with each stream processing mainly one property of the incoming information. These properties are *form, color and motion*. Furthermore it is commonly acknowledged that there exist many intra-stream connections as well as inter-stream connections. Thereby information is fed forward to higher areas, as well as *fed back* into lower cortical areas at almost all cortical levels. It was postulated that the strong connectivity between the areas of the visual cortex supports a dynamic process of *reentry* with a continuous parallel and recursive signaling between the areas. This

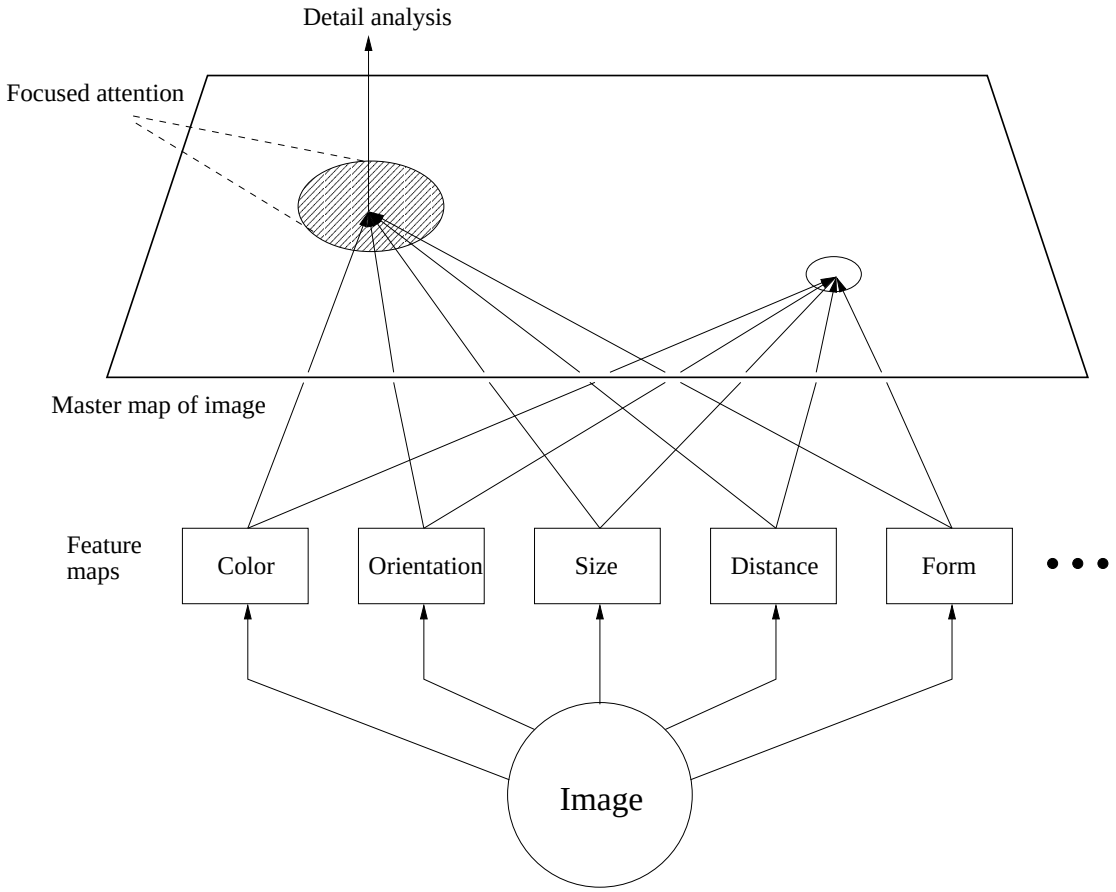


Figure 2.8: A hypothetical model of the stages in visual perception based on experiments by Ann Treisman (Adopted from [Kan91])

mechanism called *reentry* is seen as a principle in cortical integration.

Finally, in *higher brain areas*, the information coming from the visual cortex in categorized into features to build so-called *feature maps*. Later *attentive processes* direct attention to specific features of an object, selecting and highlighting features that are initially segregated in the separate feature maps. This attentive process takes a *winner-takes-all* strategy, whereby the salient features of the object are attended to (*focus of attention*) while other features and other objects are inhibited.

A *master map* codes all key aspects of an image. This master map receives input from all feature maps but abstracts only those features in each map that distinguish the object of attention from its surroundings. For object recognition information from the memory is used and compared with the master map.

2.2 Hand–Target Interaction

This section describes experiments and models performed to analyze the interaction between hand and target for the two case "reach-to-grasp" and "reach-to-catch". First, experiments and results of different researchers leading to qualitative descriptions of the characteristics of human reaching movements are shortly presented (Section 2.2.1), then models developed to explain these characteristics are shown (Section 2.2.2). Thereby the analysis is limited to the main topics that are necessary to understand the findings described for the catching experiments. For a more profound analysis of "reach-to-grasp" movements [Hau99] is recommended.

For interaction with a moving target experiments (Section 2.2.3) and models about catching are shortly described (Section 2.2.4). One very promising is then analyzed in more detail with special emphasis on the aspects "movement initiation" (Section 2.2.4.1) and "online control" (Section 2.2.4.2) of the reaching phase.

2.2.1 Interaction with a Static Target: Reaching

Human prehensile movements can be separated into a *transport* component and a *manipulation* component. While the former is mainly effected by shoulder and elbow the later is effected by wrist and fingers. That these components are in fact separate processes can be seen in experiments in which different task parameters like target position and size are varied ([PMMJ91], [BMMZ94]).

The transport component is used to *reach* the object to grasp (for a review see [Geo86, Ros94]). Those movements are highly automated; the common opinion is that such movements are preprogrammed by the central nervous system in the form of so-called *motor programs* which are scaled according to the task parameters. However, moving the hand from point to point is a heavily under-constrained task as pointed out in [Kaw96]: (1) The human arm has a redundant number of degrees of freedom (three both at the shoulder and at the wrist, one at the elbow), so there exist an infinite number of possible configurations at the end point. (2) There exist infinite ways, i.e. an infinite number of trajectories to get from the starting point to the end point. (3) Those trajectories can be realized using different muscle synergies. To limit this enormous space of possible solutions, researchers have looked for parameters that remain invariant. Morasso [Mor81] analyzed hand trajectories of planar movements[4] in Cartesian and joint space and found that the Cartesian space trajectories are much more invariant than the resulting joint space trajectories. Kaminski et al. [KG89] supported this finding by analyzing single and multi-joint movements leading to similar results. In general it can be stated that the path of the hand

[4]The problem with the analysis of 3D movements is that it is difficult to compensate external effects like gravitation; this is perhaps the main reason why research has focused on movements in the horizontal plane.

is a roughly straight line in Cartesian space and the tangential (Cartesian) hand velocity profile is bell-shaped.

In most experiments like those of Morasso, the hand was hidden from sight, e.g. by moving below a table, thus corresponding to a pure visual feed-forward control. However, such movements are quite inaccurate concerning both direction and extent (see [Geo86] for a review). In opposition accuracy increases significantly if visual feedback is permitted [PEKJ79]. In 1899 already, Woodworth proposed that a reaching movement consists of two components: an *"initial impulse propelling the hand towards the target"* and a *"current control to home in on the final position via successive approximations"* [Woo99]. The former was found to be dependent on visual information only at the beginning, to generate a trajectory (visually *directed* motion); the latter depends on primarily visual feedback during motion (visually *guided* motion). This qualitative description was supported experimentally by Milner [Mil92] who measured the trajectories of human subjects inserting a pin into a hole. For small holes and therefore high precision requirements, the velocity profile showed small oscillations at the end, corresponding to a sequence of sub-movements.

However, there is a price to pay for increased accuracy: movement duration. Woodworth was probably the first to investigate this so-called *speed-accuracy trade-off*[5] (see [MSW82] for a review); he found it to be more pronounced in the presence of visual feedback, but existent as well for pure feed-forward movements. The latter is not surprising; even if visual information may be the primary source of feedback it is not very probable that proprioceptive information is not used at all. The first one to formalize this relation was Fitts [Fit54]. Equation 2.1 states the relation between movement duration T and "task difficulty" D/W (with D being the distance to cover and W the width of the target along the direction of the movement) in the form which is commonly called "Fitts' law":

$$T = C_1 + C_2 \cdot log_2(2D/W) \tag{2.1}$$

This relation holds in a variety of tasks (see [MBGL94] for 2D writing, [BMMZ94] for grasping). However, there are cases where different relations seem to hold, e.g. a linear one which has been explained by Meyer et al. [MSW82]. Plamondon [Pla95] furthermore showed that a power law might fit the data in both cases better than the logarithmic/linear law.

By introducing external disturbances e.g. by changing target size, the principles behind motion control can be examined even further. An experiment suited to the analysis of reaching or aiming movements is the so-called *double-step target*: here, the position of the target changes stepwise, either before or after the beginning of motion. It has been shown that motion is adapted smoothly [FH91]; the velocity profile shows multiple peaks only if the target step appears later during the movement or if its direction is very different to that of the primary movement. If a small step occurs very early during a saccade of the

[5] In fact, the variable measured in experiments on the speed-accuracy trade-off is not speed, but movement duration!

eyes, Pélisson et al. [PPGJ86] reported that the subjects correct their movement without being aware of it. The reaction time to a visually perceived target step lies between $100ms$ and $250ms$ (*visual reaction time*) [Ros94][6], and is therefore much shorter than the reaction time to the first target. This indicates that the movement is not completely re-planned. This finding of the duration of the visual reaction time corresponds well to the findings of the reaction times to continuously moving target objects (see Sec. 2.2.3).

There has not been much research on which visual information is actually used as an input to human motion control. Paillard [Pai96] showed that visual motion information is indeed incorporated into motion control. The principal source, however, seems to be positional information. Vercher et al. [VMPG94] analyzed the accuracy of pointing movements depending on whether the target was foveated (domain of "positional vision") or viewed peripherically (domain of "motion vision") and found the former case to lead to more accuracy (due to the higher resolution in the fovea area). Goodale and Servos showed that the availability of binocular cues before or during a movement increases its accuracy and efficiency [GS96] in contrast to purely monocular cues. Hu et al. [HEG99] show that it is indeed 3D information that is used for motion control, not 2D information from the retinal planes.

Summarizing this section, one can say that, using the robotic terms human hand-eye coordination for reaching movements is based on a flexible combination of (mainly) visual feed-forward and (optionally) visual feedback control, the latter being important to increase accuracy. The system is position-based; binocular information is not a prerequisite but increases the accuracy and efficiency of movements.

2.2.2 Models of Human Reaching Movements

The former section dealt mainly with a qualitative description of the characteristics of human reaching movements, this section describes models developed to explain these characteristics. Conceptually the models have been divided into three "classes": models based on the so-called *equilibrium point hypothesis*, those based on *optimization theory*, and models *reproducing the form of the velocity profile*.

Models Based on the Equilibrium Point Hypothesis The equilibrium point (EP) hypothesis, first proposed by Feldman in 1996 [Fel66], is based on the mechanical properties of muscles which can be described as "tunable damped springs". According to Hooke's law, the tension of an undamped spring is proportional to (1) its stiffness and (2) the distance it is stretched from the resting position. In the human musculo-scelettal system, a joint is often moved by a muscle pair, the agonist and the antagonist. If both are modeled as

[6]The shortest times have been measured by Paillard [Pai96] for the integration of visual motion information; the larger boundary seems to be the time it takes for estimating integrating positional information.

springs, then for given resting lengths and stiffnesses there exists a unique joint state where the two-muscle system is at an equilibrium.

Limb motion might therefore be generated by the central nervous system by changing the spring parameters. In the so-called α-*model*, the stiffness parameters are influenced by a continuous activation of the muscles; in the λ-*model*, the central nervous system once specifies desired resting lengths which are then achieved using proprioceptive feedback. The former is the base for the work of Bizzi et al. (see e.g. [BHMIG92]), who found that monkeys could reach a learned target position (one joint movement) even without any visual or proprioceptive feedback. However, as the α-model completely lacks the notion of feedback, it cannot account for "normal" movements which definitely use proprioceptive feedback, e.g. the stretch reflex. Feldman's group (e.g. [FOF93]) followed the λ-model, which is completely dependent on feedback and therefore fails to explain the performance of the monkeys. A combination of the two models can be found in [MB93].

Lacking the possibility to describe trajectories of multi-joint movements with those models, in [FOF93], the λ-model was extended by postulating a time-variant equilibrium point that moves at constant speed on the straight line between start and goal position. In the case of double-step targets, the equilibrium point is shifted a second time, from the first to the second target position. Note, that the shifting time is considerably smaller than the duration of the movement. The α-model was extended in a similar way by moving the equilibrium point on the minimum jerk trajectory [Fla89] which is discussed in the following section.

The fact that in order to account for trajectories, the EP models both have been combined with high-level, "kinematic" models suggests that the equilibrium point hypothesis should not be treated as a stand-alone model, but as an intermediary, dynamic model.

Models Based on Optimization Theory Applying methods of optimal control theory [BH75] to a criterion function which describes the objective of the movement makes it possible to find a trajectory which minimizes this criterion function (with respect to the dynamic and algebraic constraints imposed by the system). Not all trajectories obtained by this method are confirmed by experimental data. Nelson [Nel83] shows the result of minimizing different physical "costs" like time, force, impulse, energy, and more. Thereby e.g. minimizing impulse results in a trapezoidal velocity profile. Two optimization models that are experimentally confirmed are the **Minimum Torque Change Model** [UKS89] (dynamic objective function) and the **Minimum Jerk Model** [FH85] (kinematic objective function).

In the latter model the objective is to minimize *jerk*[7] or, to put it in other words, movements are planned in a way to assure a maximal smoothness. Equation 2.2 states the objective

[7] Jerk = derivative of acceleration.

function C for a 2D-movement, Eq. 2.3 the corresponding hand path

$$C = \frac{1}{2} \int_{t=0}^{t=T} \{(\frac{d^3x}{dt^3})^2 + (\frac{d^3y}{dt^3})^2\} dt \tag{2.2}$$

$$\mathbf{x}_1(t) = \mathbf{x}_0 + (\mathbf{x}_{T1} - \mathbf{x}_0) \cdot (10\tau_1^3 - 15\tau_1^4 + 6\tau_1^5) \tag{2.3}$$

with $\mathbf{x}_0$ and $\mathbf{x}_{T1}$ being the start and the target position, T being the movement duration, and $\tau_1 = t/T$. The resulting paths are straight Cartesian lines, the tangential velocity profiles are symmetric and bell-shaped.

Reactions to double-step targets, i.e. the case that the target "jumps" during the movement from $\mathbf{x}_{T1}$ to $\mathbf{x}_{T2}$, are explained by the so-called *superposition scheme* [FH91]: At the time $t = t_2$, a second minimum jerk trajectory (Eq. 2.4) is superimposed on the first:

$$\mathbf{x}_2(t \geq t_2) = (\mathbf{x}_{T2} - \mathbf{x}_{T1}) \cdot (10\tau_2^3 - 15\tau_2^4 + 6\tau_2^5) \tag{2.4}$$

with $\tau_2 = (t - t_2)/(T - t_2)$. The parameters of the model were estimated by fitting it to experimental data. It should be mentioned that the model has been successfully implemented on a robot [Hen91].

Burdet [Bur96] extended the minimum jerk superposition model in two ways: First, he combined it with the model of Meyer et al. [MSW82] which proposes that every movement consists of sub-movements with random variable duration and extent, and thereby accounts for the different forms of the speed-accuracy trade-off. In contrast to Flash, the duration and extent of the sub-movements can now be computed, and only the total duration, distance, accuracy, and the number of sub-movements have to be extracted from experimental data. In contrast to Meyer, the sub-movements minimize jerk, not time. Burdet also integrated one of the results of Milner's peg-in-hole experiments [Mil92], namely that sub-movements seem to be triggered periodically with a rate corresponding to the visual reaction time. While Milner's observations stem from visual feedback experiments, Burdet's model does not account for this case. In later work, Burdet and Milner [BM98] proposed another algorithm for the parameterization of sub-movements for the case of high accuracy requirements in the presence of feedback on the current position. Some of the model parameters, as for example the rate of the final sub-movements, were estimated from experimental data; duration and extent of the first two sub-movements ("the plan") can be learned. The form of the sub-movements itself was modeled only qualitatively by using bell-shaped velocity profiles.

Model Reproducing the Velocity Profile Nelson's analysis of the effect of different objective functions [Nel83] also showed that different objective functions, namely minimum energy and minimum jerk, result in a very similar velocity profile, the well-known bell shape. This observation lead **Goodman** et al. [GGC92][8] to propose that the precise objective is not important, but that by achieving the bell-shaped velocity profile more than

[8]The paper was published under his former name, Gutman.

one objective is minimized. Following this line of reasoning, they modeled the velocity during a reaching movement with a differential equation which computes the current velocity from the remaining distance to the target. More detailed information to this model can be found in Section 4.6.1.

2.2.3 Interaction with a Moving Target: Catching

Despite the fact that catching appears to be more complicated than reaching, the ability to reach toward moving targets develops at a very young age ([vH82]). This ability develops rapidly from birth and already by age of 36 weeks infants can accurately *intercept a moving object* ([vH79]). Furthermore, infants seem to use a predictive strategy in which the initial movement is directed toward the interception point rather than tracking the object ([vH80]).

Catching or hitting experiments with adults have been performed by many researchers. Thereby different aspects were in the focus of attention: [MC99] shows which characteristics of target motion are important in the control and coordination of the transport and grasp-preshape components of prehensile movements during a interception task. [PPG96] investigates the performance of human subjects during the interception of real and path-guided apparent motion targets. [vdKSS97] studies the effect of multiple information sources on interceptive timing. Thereby the influence of background structure and monocular vs. binocular vision on the timing of a grasp in a simple one-handed catch is evaluated. [DLC00] investigates target velocity effects on manual interception kinematics. In two experiments the influence of early or late vision on the target was evaluated. As a result it is suggested that subjects use visual information early in the target's trajectory to form a representation of the target motion that is used to facilitate manual interception. [BP92] investigate the performance of human subjects in ball catching. They show that predictive information about *when* an approaching ball will be *where* is available in the transformation of the optic array sampled at the point of observation of the catcher. They state that (a) humans are able to use these information sources, (b) the power of this predictive information lies in the possibility it offers for prospective control. Success in actions like catching and hitting would be almost impossible without such predictive information.

2.2.4 Models for Human Catching Movements

Despite the fact that there have been many experiments accomplished describing the catching behavior of humans or even patients, almost no quantitative models have been developed so far. A model that describes the manual interception of moving targets in a *quantitative* way is the model of Port, Lee and Georgopoulos [PLDG97, LPG97]. In the experiments leading to this model two main questions have been addressed: (1) when does a catching movement start and which trigger initiates it, and (2) how is the interceptive

movement performed regarding position, velocity and acceleration of the intercepting hand.

2.2.4.1 Movement Initiation

Explanations for the initiation of arm motion in intercepting moving targets have been given by two models: the *threshold-distance* and the *threshold-τ model*. The first of these was put forward by Collewijn (1972) [Col72] to explain the optokinetic response time in a rabbit and proved to be also applicable to the interception of moving targets by human subjects. This "threshold-distance model" is so named because it postulates that a stimulus must travel a certain visual angle before a motor response can be elicited.

The second model of target interception incorporates the concept of the *time-to-contact* and τ as proposed by Lee [Lee76]. Under conditions of constant velocity, τ is a variable that could be computed by the brain to determine the time to contact with the object. The variable τ was originally defined under conditions of self motion in an optic flow field [Lee76], in which it was proven that τ is equal to the inverse of the rate of dilation of the retinal image.

For interceptive movements as described in the experiments of Georgopoulos [PLDG97, LPG97] the τ of the target (which is a continuous variable changing with time) is defined as the distance of the target to the interception zone at a time t divided by the velocity of the target at time t.

$$\tau = \frac{d(t)}{\dot{d}(t)} = \frac{\text{target distance}}{\text{target velocity}} \tag{2.5}$$

In the following first an experiment is presented which examined the initiation and performance of "reach-to-catch" movements of human subjects to intercept moving targets [PLDG97], then the relation of the aforementioned models with experimental results is shown.

Experiment of Georgopoulos In [PLDG97] the initiation (i.e. the reaction time) of the reaching movement in response to a moving target has been examined and explanations for the observed behavior have been given. To examine those human arm movements following methods were applied: Subjects sat unrestrained in front of a two-dimensional articulated manipulandum. The manipulandum was on a table which was approximately $10cm$ above the subjects waist. By moving the manipulandum subjects controlled the location of a position feedback cursor on a computer screen. At the beginning of a trial subjects had to move the cursor to a start zone, which was centered at the lower vertical meridian of the screen. After the subjects maintained this position for a random period of $1 - 3s$ a target appeared in the lower left or right corner of the screen. The target then traveled

along a 45° path until it reached the vertical meridian of the screen, where it stopped at a location $12,5cm$ directly above the center of the start zone. The subject was required to move the feedback cursor so as to intercept the target just as it reached its final position at the center of the interception zone. No other requirements, e.g. to move as fast as possible, were put on the subject. The target traveled either with a constant velocity, a constant acceleration or a constant deceleration. The target movement duration took one of 6 values between $0.5s$ and $2.0s$, i.e. altogether 36 combinations were possible. For each trial one possibility was chosen randomly.

With this setup two different reactions could be observed which can be seen in Figure 2.10: In the plots the response time as a function of movement duration (also called Target Motion Time) is drawn. Thereby it can be observed that for subjects 1 and 2 the response time is continuously rising with the target motion time, while for subjects 4 and 5 the response time stays almost constant for all target motion times. For both behaviors models can now be assigned:

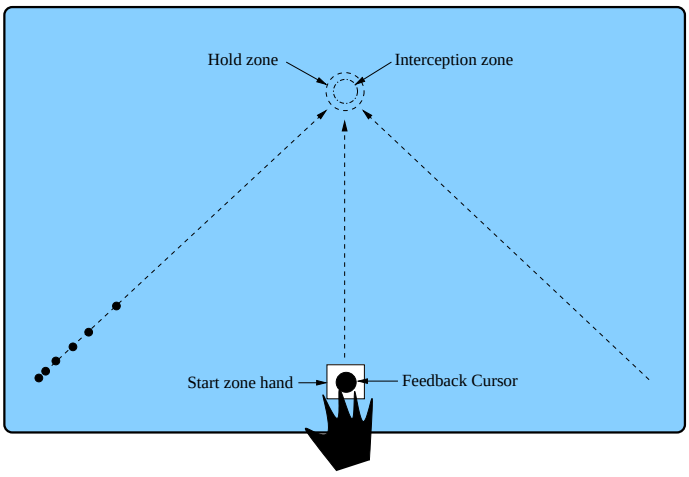


Figure 2.9: Setup for experiments to determine the reaction time and to analyze interceptive movements

- *threshold-distance model*
  The threshold-distance model predicts that a subject will respond after some constant processing time *plus* the time it takes the target to travel a certain distance (threshold distance). This model complies to a *reactive strategy*. For targets traveling at a constant velocity[9] the response time is almost independent of the velocity of the target object (see subjects 4 and 5 in Figure 2.10):

$$\text{Response time} \quad = \quad \text{processing time} + \frac{\text{threshold distance}}{\text{target velocity}} \tag{2.6}$$

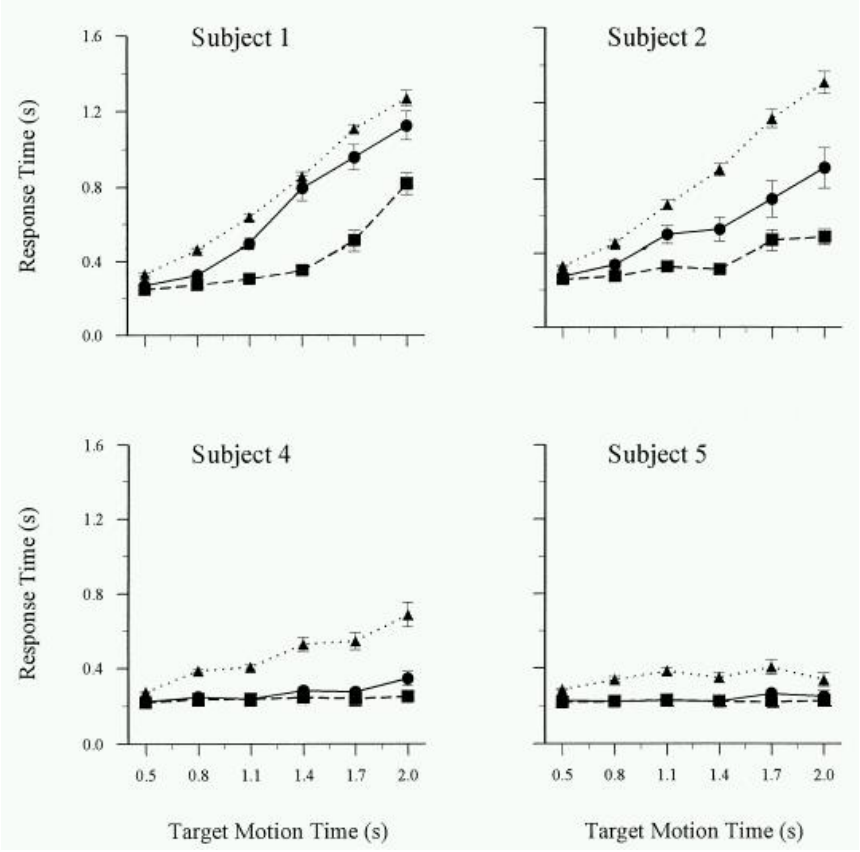


Figure 2.10: Response time as a function of movement duration (from [PLDG97]).

- *threshold-τ model*
  In the threshold-τ model, a movement is initiated after a certain processing time from the moment at which the τ of the target decreases below a certain threshold. This model complies to a *predictive strategy*. For targets traveling at a constant velocity

[9]For acceleration or decelerating targets, respectively, the response time is slightly different to Equation 2.6 (see [PLDG97])

the response time strongly depends on the velocity of the target (see subjects 1 and 2 in Figure 2.10), whereby faster targets elicit shorter response times.

$$\text{Response time} \quad = \quad \text{processing time} + \frac{\text{DSI} - \tau v_0}{v_0} \tag{2.7}$$

with v_0 being the initial target velocity and DSI being the distance between starting position of the target and the interception zone.

Which strategy is used depends on the subject, but is chosen unconsciously. The analysis revealed, if the experiments are performed with more emphasis on the speed of the response rather than its accuracy, it is likely that subjects are biased towards adopting the reactive strategy. In this case using a predictive strategy might be too time consuming since it would require an estimation of target displacement and velocity. On the other hand, if experiments are designed to put more emphasis on the temporal accuracy of target interception, this might bias the subjects towards the predictive strategy [LPG97].

2.2.4.2 On-line Control of Hand Movement

Experiment of Georgopoulos Analyzing the kinematic characteristics of arm movements for the above mentioned experiments it was found out that (1) for fast moving targets, subjects produced single movements with symmetrical, bell-shaped velocity profiles and (2) for slowly moving targets, hand-velocity profiles displayed multiple peaks, which suggested a control mechanism that produces a series of discrete sub-movements according to the characteristics of target motion. Despite the fact that the number of sub-movements, their amplitude and the intermediate time between two consecutive sub-movements change from subject to subject, the chosen strategy, the duration of the target motion as well as from the target's velocity, nevertheless some invariants could be seen by more detailed analysis: (a) the number of sub-movements was roughly proportional to the movement time, resulting in a relatively constant sub-movement frequency ($\approx 4Hz$). (b) the median duration T_s of a sub-movement is approximately $0.5s$ (see Figure 2.12). (c) the onset of a sub-movement has a relatively constant temporal relationship with the offset, instead of the onset, of the preceding sub-movement. On average the onset of a sub-movement precedes the offset of the preceding sub-movement by $0.25s$ ($\approx T_s/2$!). This duration is called *Intersubmovement Interval* or ISMI [10] (see Figure 2.13). (d) the sum of the amplitudes of all sub-movements is approximately constant.

Analyzing sub-movement amplitude and its relation to target motion revealed that the subjects achieved interception mainly by producing a series of sub-movements that would

[10] It should be noted that the constancy of the ISMI found in [LPG97] is probably not a general principle that applies to all types of movements, since it has been shown that, with sudden change in target location, an ongoing movement can be modified at any time during its execution.

keep the displacement of the hand proportional to the first-order estimate of target position at the end of each sub-movement along the axis of hand movement.

This finding lead Georgopoulos to formulate a *Position control hypothesis*: At the beginning of a sub-movement the subject tries to determine the position of the target at the end of the sub-movement and to reach this position with its hand in time with the target. In case of a constant target velocity the situation is as follows:

$$position_{hand} = P_G + V_G D_S \tag{2.8}$$

with D_S being the duration of the sub-movement, P_G and V_G being the position and velocity of the target at the beginning of this sub-movement.

The amplitude of hand motion is therefore:

$$amplitude_{hand} = P_G + V_G D_S - P_H \tag{2.9}$$

with P_H being the position of the hand at the beginning of the sub-movement. Taking a constant time for the duration of a sub-movement, e.g. the median value of all sub-movement durations D_m, the amplitude becomes:

$$amplitude_{hand} = P_G + V_G D_m - P_H \tag{2.10}$$

For the described experiment Equation 2.9 as well as Equation 2.10 gave good results compared to the real behavior of the human subjects.

What could be additionally observed was that depending on the chosen movement initiation strategy the "overall" control of the arm motion differed. Subjects using a reactive strategy tried to keep up with the target object, i.e. the subject tried to keep the error between the cursor position and the current target position to be zero:

$$E = Y_{target} - Y_{feedbackcursor} = 0 \tag{2.11}$$

If this condition is fulfilled until the target enters the interception zone, target and feedback cursor will contact finally.

Resulting velocity profiles of the hand motion can be seen in Figure 2.11 on the right side.

In contrast, for subjects using a predictive strategy no straight relationship between hand motion and target motion can be seen. The motion of the hand is significantly shorter as with the reactive strategy and the subject produces less sub-movements. To explain this behavior Georgopoulos [LPG97] gives to possibilities: (1) The subject is unable to generate sub-movements at a rate fast enough to keep up with the target or (2) the subject can estimate the target's velocity very well and is able to predict the time of contact with the interception zone. Therefore one impulse, maybe slightly corrected, is enough to contact the target accurately.

Finally, the effect of target acceleration was analyzed. It was found out that, if the target is accelerating during the trial, on average subjects move the feedback cursor too late into the interception area. In opposite, if the target is decelerating, subjects move the cursor too early into the interception area. This implies that subjects cannot take accelerations or decelerations into account for the control of their hand movement.

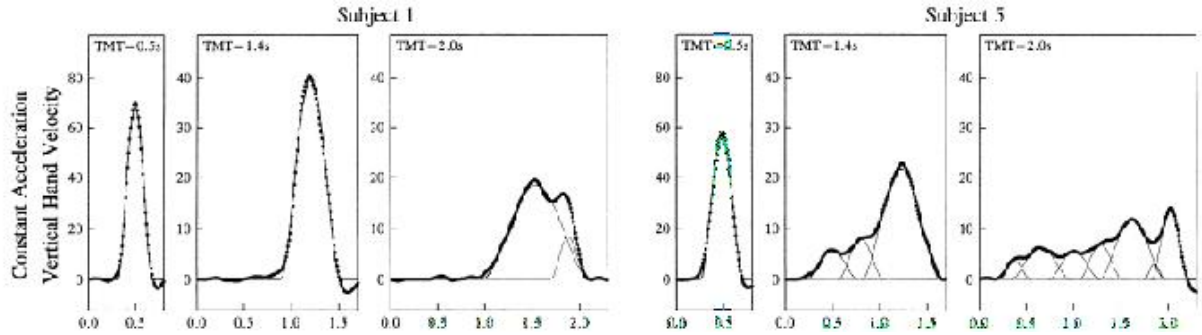


Figure 2.11: Hand motion for different target motion times. Left: predictive strategy. Right: reactive strategy.

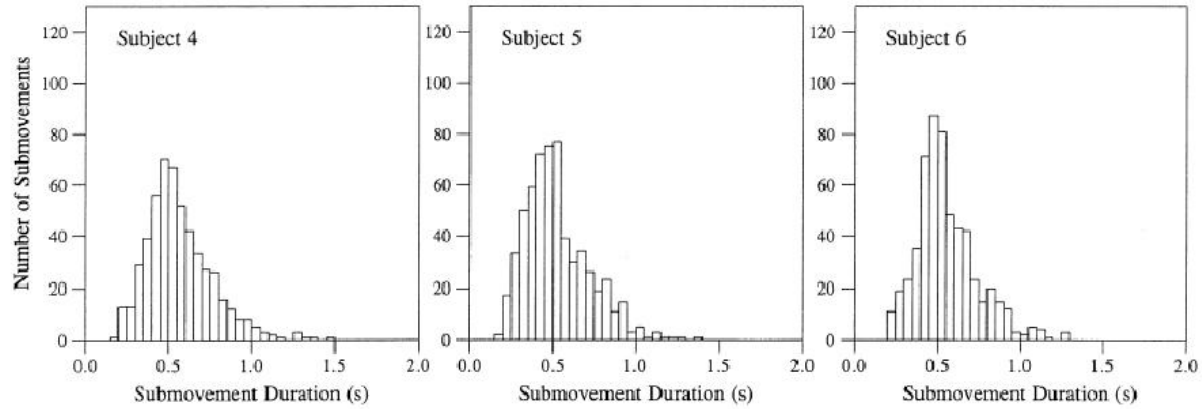


Figure 2.12: Duration of sub-movements. The median duration is approximately 0.5 s.

2.2.5 Summary

In this section "reach-to-grasp" as well as "reach-to-catch" hand motions were analyzed in more detail. Thereby results from different researches were compared. For the case of "reach-to-grasp" it can be stated that there exist some invariant characteristics for those motions, namely that (a) these motions are single movements with a straight hand path and a bell-shaped velocity profile, and (b) the 3D position of the target object serves as input data. Different models explaining and describing these findings have been presented.

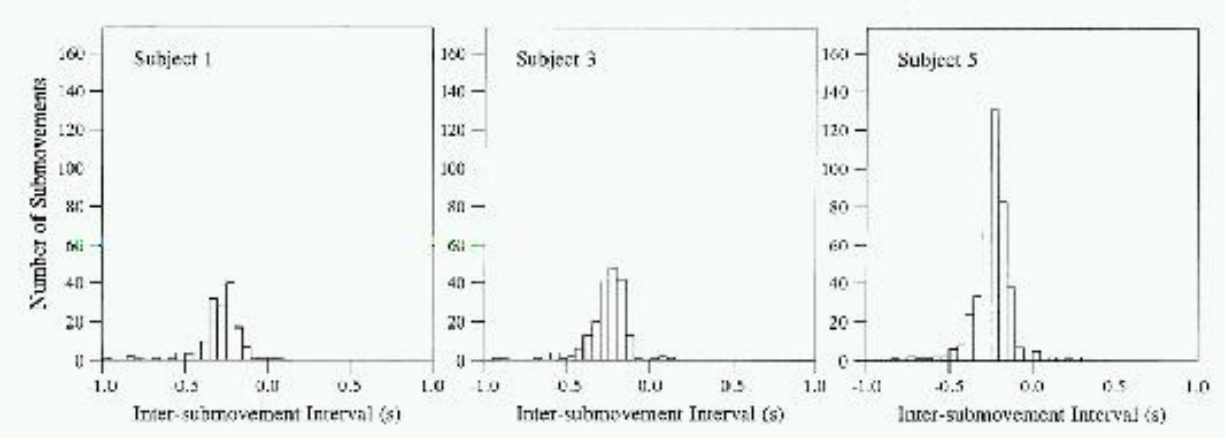


Figure 2.13: Duration between the end of a sub-movement and the beginning of the next. The ISMI is approximately $-0.25s$.

In the case of "reach-to-catch" movements the situation is quite similar. There exist also some invariant characteristics. Namely, (a) for movement initiation either a predictive (threshold-τ model) or a reactive strategy (threshold-distance model) is chosen. (b) the arm movement consists of a single or multiple sub-movements depending on initiation strategy and target motion time. (c) the median duration T_s of a sub-movement is approximately $0.5s$. (d) the number of sub-movements is roughly proportional to the movement time. The rate of sub-movements is approximately constant. Thereby every $250ms$ a new sub-movement is started (ISMI = Intersubmovement interval). (e) the sum of the amplitudes of all sub-movements is approximately constant. (f) it seems that most subjects use a "position control strategy": At the beginning of a sub-movement the subject tries to determine the position of the target at the end of the sub-movement and to reach this position with its hand in time with the target.

When comparing the results of reaching and catching some similarities can be observed: (a) while reaching movements consist of one single peaked bell-shaped velocity profile, "reach-to-catch" movements are mainly produced by an superposition of those profiles. (b) reaching with high accuracy demands and catching using a reactive strategy are very similar: This can be seen from trajectories of human subjects inserting a peg into a hole. For small holes and therefore high precision requirements, the velocity profile showed small oscillations at the end, corresponding to a sequence of sub-movements. Additionally, these sub-movements seem to be triggered periodically with a rate corresponding to the visual reaction time. (c) the reaction time to a visually perceived target step (double step target experiment) lies between $100ms$ and $250ms$ (*visual reaction time*). In case of a reactive strategy sub-movements are generated with a rate that the ISMI is approximately $250ms$. Thereby hand and target are observed which implies that visual processing time is the limiting factor for the generation of movements. (d) if the subject is unsure about its movements and a higher control of the arm movement is necessary (as e.g. for aforementioned high precision movements) subjects use feedback about the current target and hand

position to control the motion which results in sub-movements.

2.3 Discussion

The main question that arises when reviewing this chapter is: What can be learned or derived, respectively, from the knowledge gained by the analysis of human visual processing and the reaching or catching behavior? Since the goal is to control a robot only those results can be transfered that provide quantitative information or results that provide a good principle to transfer. The latter is true for the vision part were the mechanisms of *parallel information flow* and *reentry* are interesting and imply further observation. For the "catching" the notion of super-positioned sub-movements at constant movement rates and movement time implies the transfer.

Both topics build the basis for developed models as they are described in Chap. 4.

Chapter 3

Robotic Hand-Eye Coordination

The following chapter treats topics of the term "hand-eye coordination". In the first part basics for "internal model" needed for every calibrated hand-eye system are presented (Sec. 3.1).

In the second part aspects of image processing are treated with emphasis on methods of visual tracking. Thereby, a review of methods for visual tracking of moving objects (Section 3.2.1) using different sensor modalities (Section 3.2.1.1 for form, Section 3.2.1.2 for color and Section 3.2.1.3 for motion) is given. Additionally methods to determine grasps on the tracked object are introduced (Section 3.2.3).

Since the prediction of object motion is essential for the successful catching, methods for the prediction of time series (Section 3.3) are presented whereby their suitability for our purpose is analyzed.

The chapter closes with a review on literature and methods for robotic manipulator control for interaction with static (Section 3.4.1) and moving targets (Section 3.4.2).

3.1 Internal Models

An internal model is a "simplified description of an object or of the function of an object" that is used by a system. In an hand-eye system, different sub-systems need different internal models. E.g. the sub-system charged with translating planned trajectories into robot commands needs an inverse kinematics model of the arm. To reconstruct spatial information from the images an inverse model of the camera-head system is needed. Furthermore, this reconstructed information has to be translated into arm coordinates.

In a robotic hand-eye system, one can imagine two classes of applications: Grasp or catch a known object, perhaps even interact with it in a special way, or pick up, grasp or catch

an unknown object. In the first class (geometric) object models have to be stored; these models are mainly needed for contour based tracking (see Sec. 3.2.1.1), determination of grasps (see Sec. 3.2.3) as well as for e.g. tracking of precomputed grasps (see Sec. 4.4.1).

Since this thesis mainly deals about interacting with a moving target, in Section 3.3 motion models to describe the target's way are shortly introduced.

3.1.1 Models of the Hand-Eye System

Model of the Manipulator

In this section on the one hand geometric model properties of a manipulator are described. On the other hand, and thereby maybe going above the word "model", also kinematic relationships for non-redundant and redundant manipulators are described.

Coordinate systems and -transformations An object in Cartesian space has 6 degrees of freedom (d.o.f.), whereby 3 are translatory and 3 are rotatory. To be able to describe these facts mathematically, the following definitions are needed:

- a base coordinate system (or world coordinate system) as a reference system. This world coordinate system is called $\mathcal{F}_w$ in the following. It is assumed that $\mathcal{F}_w$ is a Cartesian right hand system in $\mathcal{R}^3$ with the orthonormal base matrix:

$$ {}^{w}\mathbf{W} = \begin{bmatrix} {}_{w}\mathbf{b}_x & {}_{w}\mathbf{b}_y & {}_{w}\mathbf{b}_z \end{bmatrix} \tag{3.1} $$

  In general matrix ${}^{w}\mathbf{W}$ can be any 3×3 matrix, that fulfills above conditions. In the simplest case one can take the identity matrix, so that

$$ {}^{w}\mathbf{W} = \mathbf{I} \tag{3.2} $$

- a three-dimensional Cartesian vector that describes the position of an object relative to $\mathcal{F}_w$. This vector is a vector from the origin in coordinate system $\mathcal{F}_w$ and describes the position of a fixed point of an object. This vector is called ${}^{w}\mathbf{p}$ in the following.

$$ {}^{w}\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.3} $$

- the rotation of an object relative to $\mathcal{F}_w$ can be described by an object coordinate system $\mathcal{F}_o$, whose origin is in the point that is defined by the position vector ${}^{w}\mathbf{p}$. This object coordinate system is also defined as a Cartesian right hand system in $\mathcal{R}^3$

with an orthonormal base matrix, whose base vectors are defined in $\mathcal{F}_w$. This matrix is denoted as

$$ {}^{w}_{o}\mathbf{R} = \left[\begin{array}{ccc} {}^{w}_{o}\mathbf{b}_x & {}^{w}_{o}\mathbf{b}_y & {}^{w}_{o}\mathbf{b}_z \end{array} \right] \tag{3.4} $$

and defines a rotation matrix. For matrix ${}^{w}_{o}\mathbf{R}$ applies:

$$ \det {}^{w}_{o}\mathbf{R} = 1, \quad {}^{w}_{o}\mathbf{R}^T \, {}^{w}_{o}\mathbf{R} = {}^{w}_{o}\mathbf{R} \, {}^{w}_{o}\mathbf{R}^T = \mathbf{I} \tag{3.5} $$

The geometric data of an object (e.g. edges, planes, ...) can now be described relative to $\mathcal{F}_o$ and therefore independently of the world coordinate system $\mathcal{F}_w$. To transform a vector ${}^{o}\mathbf{x}$ in $\mathcal{F}_o$ (e.g. an edge point of a plane) into the world coordinate system, following transformation is necessary:

$$ {}^{w}\mathbf{x} = {}^{w}_{o}\mathbf{R} \, {}^{o}\mathbf{x} + {}^{w}\mathbf{p} \tag{3.6} $$

This equation can be simplified, if one arranges ${}^{w}_{o}\mathbf{R}$ and ${}^{w}\mathbf{p}$ appropriately into a 4×4 matrix ${}^{w}_{o}\mathbf{T}$:

$$ \left[\begin{array}{c} {}^{w}\mathbf{x} \\ 1 \end{array} \right] = {}^{w}_{o}\mathbf{T} \left[\begin{array}{c} {}^{o}\mathbf{x} \\ 1 \end{array} \right] = \left[\begin{array}{cc} {}^{w}_{o}\mathbf{R} & {}^{w}\mathbf{p} \\ \mathbf{0}^T & 1 \end{array} \right] \left[\begin{array}{c} {}^{o}\mathbf{x} \\ 1 \end{array} \right] \tag{3.7} $$

Now objects can be described not only relatively to the world coordinate system, but also relative to other objects. Therefore the object coordinate system of one object is the reference system of another object. If we define $\mathcal{F}_a$ as the object coordinate system of object A, that is defined relatively to $\mathcal{F}_o$

$$ \left[\begin{array}{c} {}^{o}\mathbf{x} \\ 1 \end{array} \right] = {}^{o}_{a}\mathbf{T} \left[\begin{array}{c} {}^{a}\mathbf{x} \\ 1 \end{array} \right] = \left[\begin{array}{cc} {}^{o}_{a}\mathbf{R} & {}^{o}\mathbf{p} \\ \mathbf{0}^T & 1 \end{array} \right] \left[\begin{array}{c} {}^{a}\mathbf{x} \\ 1 \end{array} \right] \tag{3.8} $$

one gets the coordinates of the vector ${}^{a}\mathbf{x}$ in the world coordinate system by following concatenated transformation:

$$ \left[\begin{array}{c} {}^{w}\mathbf{x} \\ 1 \end{array} \right] = {}^{w}_{o}\mathbf{T} \left[\begin{array}{c} {}^{o}\mathbf{x} \\ 1 \end{array} \right] = {}^{w}_{o}\mathbf{T} \, {}^{o}_{a}\mathbf{T} \left[\begin{array}{c} {}^{a}\mathbf{x} \\ 1 \end{array} \right] \tag{3.9} $$

The concatenation is therefore a simple multiplication of the transformation matrices. In general one can write for a transformation from $\mathcal{F}_m$ to $\mathcal{F}_0$:

$$ {}^{0}_{m}\mathbf{T} = {}^{0}_{1}\mathbf{T} \, {}^{1}_{2}\mathbf{T} \ldots {}^{m-1}_{m}\mathbf{T} \quad \text{with} \quad m \in \mathcal{N} \tag{3.10} $$

The general form of a transformation matrix $\mathbf{T}$ can be described as:

$$ {}^{i-1}_{i}\mathbf{T} = \begin{bmatrix} {}^{i-1}_{i}\mathbf{R} & {}^{i-1}\mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{3.11} $$

The form of matrix ${}^{0}_{m}\mathbf{T}$ is exactly the same as the form of the single transformation matrices according to Equation 3.7. This can be verified be inserting the accordingly indexed Equation 3.11 into Equation 3.10 (see also [Hau99]).

Denavit-Hartenberg Transformation In robotics it is common to use Denavit-Hartenberg parameters for the spatial description of a manipulator.

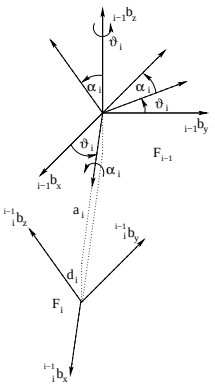


Figure 3.1: Coordinate transformation with Denavit-Hartenberg parameters

To determine those parameters, one assumes that two objects with object coordinate systems $\mathcal{F}_i$ and $\mathcal{F}_{i-1}$ are connected by a joint. The rotation axis of the joint is identical with the z-axis of $\mathcal{F}_{i-1}$, and the joint turns counterclockwise by the angle ϑ_i around this z-axis. By specification of an additional angle α_i, which turns around the x-axis of $\mathcal{F}_i$, and of two additional scalars d_i and a_i for an additional translation, the resulting coordinate system $\mathcal{F}_i$ can be aligned in a way that its z-axis is the rotation axis of the next joint. Where required it is additionally necessary to add an constant offset to the angle ϑ_i, since otherwise it is not possible to model manipulator configurations with 90° offset.

Calculation of End-effector Position and Orientation The position and orientation of the end-effector can be calculated with the concatenated transformation matrix

$$ {}^{w}_{e}\mathbf{T} = {}^{0}_{m}\mathbf{T} = {}^{0}_{1}\mathbf{T}\ {}^{1}_{2}\mathbf{T} \dots {}^{m-1}_{m}\mathbf{T} \tag{3.12} $$

that transforms the coordinates from the end-effector coordinate system $\mathcal{F}_e = \mathcal{F}_m$ to the world coordinate system $\mathcal{F}_w = \mathcal{F}_0$. With Equation 3.7 one gets the position vector ${}^{w}\mathbf{p}$ of the end-effector, if following operation is executed:

$$ \begin{bmatrix} {}^{w}\mathbf{p} \\ 1 \end{bmatrix} = {}^{w}_{e}\mathbf{T} \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \tag{3.13} $$

The three elements in ${}^{w}\mathbf{p}$ are therefore equal to the first three elements of the fourth column vector in ${}^{w}_{e}\mathbf{T}$. Accordingly, one gets the base vectors of $\mathcal{F}_e$ from the upper left 3×3 sub-matrix of ${}^{w}_{e}\mathbf{T}$. This sub-matrix is a rotation matrix ${}^{w}_{e}\mathbf{R}$ according to the definition in paragraph *Coordinate systems and -transformations*. By this rotation matrix the orientation of the end-effector is determined well-defined and can therefore be used for orientation declarations.

For certain tasks it can be sensible to use another possibility for the description of the end-effector orientation. Like stated in [Mai99] the description of the end-effector orientation can be obtained by using angles relative to the world coordinate system. This is possible e.g. by using Euler XYZ, ZYZ or RPY angles. Hereby the calculation of these angles from the rotation matrix ${}^{w}_{e}\mathbf{R}$ is necessary. Unfortunately, this conversion is not trivial, since for a given rotation matrix the solutions for the angles are not unique. Therefore, it is necessary, e.g. for the trajectory generation of an orientation trajectory, to use previously calculated values to obtain a continuous course of the angles.

As an alternative the use of *approach* and *normal vector* is suitable. Hereby the orientation of the end-effector is uniquely described by two normed vectors in a plane. The approach vector ${}^{w}\mathbf{a}$ fixes two out of three rotatory degrees of freedom, it therefore fixes the "direction" of the end-effector. For the fixation of the third rotatory degree of freedom the normal vector ${}^{w}\mathbf{n}$ is used. This vector is normally perpendicular to the approach vector, so ${}^{w}\mathbf{a} \cdot {}^{w}\mathbf{n} = 0$.

But this does not need to be fulfilled always. The orientation of the end-effector is uniquely determined, if following applies for the scalar product:

$$ -1 < {}^{w}\mathbf{a}^{T}\ {}^{w}\mathbf{n} < 1 \quad \text{with} \quad |{}^{w}\mathbf{a}| = |{}^{w}\mathbf{n}| = 1 \tag{3.14} $$

whereby ${}^{w}\mathbf{a}$ and ${}^{w}\mathbf{n}$ lie in a plane and are linearly independent.

7-d.o.f Kinematics Kinematics is the science of motion which treats motion without regard to the forces that cause it. Within the science of kinematics one studies the position, velocity, acceleration and all higher order derivatives of the *position variables* with respect to time. Hence, the study of kinematics of manipulators refers to all the geometrical and time based properties of the motion.

In the following the kinematics of a 7-d.o.f redundant manipulator[1] is described in more detail. Hereby the following is defined:

- A Cartesian position vector in $\mathcal{F}_w$

$$ {}^{w}\mathbf{p} = {}^{w}\mathbf{p}(t) = \begin{bmatrix} p_x(t) \\ p_y(t) \\ p_z(t) \end{bmatrix} \tag{3.15} $$

- A Cartesian angular velocity vector in $\mathcal{F}_w$

$$ {}^{w}\boldsymbol{\omega} = {}^{w}\boldsymbol{\omega}(t) = \begin{bmatrix} \omega_x(t) \\ \omega_y(t) \\ \omega_z(t) \end{bmatrix} \tag{3.16} $$

  The three entries determine a rotation with the angular velocity ω_i around the according coordinate axis in $\mathcal{F}_w$.

- A joint angle vector. This contains all $m = 7$ joints of the manipulator. For $q_i = 0$ the base position of the manipulator is defined.

$$ \mathbf{q} = \mathbf{q}(t) = \begin{bmatrix} q_1(t) \\ \vdots \\ q_m(t) \end{bmatrix} \tag{3.17} $$

Direct Kinematics The term "direct kinematics" denotes the transformation that enables to determine the position and orientation of the end-effector from the joint angle vector $\mathbf{q}$(t). This is a transformation from joint angle space ($\mathcal{R}^m$) to Cartesian space ($\mathcal{R}^3$). Formally, this can be described as follows, whereby $\mathbf{x}(t)$ contains the information for position (translation) and orientation (rotation):

$$ \mathbf{x}(t) = \mathbf{F}(\mathbf{q}(t)) \tag{3.18} $$

For the calculation of the direct kinematics this formal definition is split into:

[1] This is sensitive here since our robot MINERVA is also a 7-d.o.f redundant manipulator.

$$
\begin{aligned}
{}^{w}\mathbf{p}(t) &= \mathbf{F}_p(\mathbf{q}(t)) && (3.19)\\
{}^{w}_{e}\mathbf{R}(t) &= \mathbf{F}_R(\mathbf{q}(t)) && (3.20)
\end{aligned}
$$

In general, $\mathbf{F}_p$ and $\mathbf{F}_R$ are non-linear functions that map the joint angle vector $\mathbf{q}$(t) into the Cartesian position vector ${}^{w}\mathbf{p}(t)$ or the rotation matrix ${}^{w}_{e}\mathbf{R}(t)$, respectively.

The transformation matrix ${}^{w}_{e}\mathbf{T} = {}^{w}_{e}\mathbf{T}(\mathbf{q}(t))$ contains already all information according to paragraph *Coordinate systems and -transformations*, to determine ${}^{w}\mathbf{p}(t)$ and ${}^{w}_{e}\mathbf{R}(t)$ from $\mathbf{q}(t)$. The calculations can be described as follows:

$$
{}^{w}\mathbf{p}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} {}^{w}_{e}\mathbf{T}(\mathbf{q}(t)) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (3.21)
$$

$$
{}^{w}_{e}\mathbf{R}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} {}^{w}_{e}\mathbf{T}(\mathbf{q}(t)) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad (3.22)
$$

Inverse Differential Kinematics The inverse operation to the direct kinmatics of Equation 3.18 is defined by:

$$
\mathbf{q}(t) = \mathbf{F}^{-1}(\mathbf{x}(t)) \quad (3.23)
$$

For a redundant manipulator Equation 3.23 cannot be solved in a closed form, since in general, for a given position and orientation of the end-effector an infinite number of solutions for $\mathbf{q}(t)$ exist. Additional information is therefore needed to get a unique solution.

This can be reached by integrating preceeding joint angles, i.e. the inverse kinematics is calculated from the Cartesian translational velocities and the angular velocity of the end-effector. Resulting angular velocities for the joint angles are integrated over time to get the joint angles $\mathbf{q}(t)$ finally.

The direct kinematics in velocity space can be written as follows:

$$
\dot{\mathbf{x}}(t) = \begin{bmatrix} {}^{w}\dot{\mathbf{p}}(t) \\ {}^{w}\boldsymbol{\omega}(t) \end{bmatrix} = \mathbf{J}(\mathbf{q}(t))\, \dot{\mathbf{q}}(t) \quad (3.24)
$$

whereby $\mathbf{J}$ is called *geometric Jacobian*.

The inversion of Equation 3.24 for the calculation of the inverse kinmatics in velocity space is not possible by a simple inversion of matrix $\mathbf{J}$, since $\mathbf{J}$ is not a square, but a $6 \times m$ matrix (with $m > 6$)! In the same way inversion of matrix $\mathbf{J}$ is not possible if the matrix becomes singular. Therefore, for inverse kinmatics a *Pseudo-Inverse* is used, which is defined by:

$$\mathbf{J}^{\#} = \mathbf{J}^T(\mathbf{J}\,\mathbf{J}^T)^{-1} \tag{3.25}$$

Therewith the differential inverse kinmatics can be written as:

$$\dot{\mathbf{q}}(t) = \mathbf{J}^{\#}(\mathbf{q}(t)) \begin{bmatrix} {}^w\dot{\mathbf{p}}(t) \\ {}^w\boldsymbol{\omega}(t) \end{bmatrix} \tag{3.26}$$

This solution is also called *Minimum Norm Solution (MNS)*, since the solution for the pseudo-inverse $\mathbf{J}^{\#}$ minimizes the cost function

$$G(\dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T\dot{\mathbf{q}} \tag{3.27}$$

i.e. the angular velocities are minimized.

A drawback of this procedure is that it cannot (always) be avoided that the pseudo-inverse $\mathbf{J}^{\#}$ becomes singular. Two different methods can be applied to avoid that $\mathbf{J}^{\#}$ becomes a singular matrix:

- Weighted Least Norm Method (WLNM)
- Gradient Projection Method (GPM)

In the following only the GPM is explained in more detail, since this method is better applicable to describe the inverse kinematics of a redundant manipulator (see also [Mai99] for more explanations). The GPM is defined by:

$$\dot{\mathbf{q}}(t) = \mathbf{J}^{\#} \begin{bmatrix} {}^w\dot{\mathbf{p}}(t) \\ {}^w\boldsymbol{\omega}(t) \end{bmatrix} + (\mathbf{I} - \mathbf{J}^{\#}\mathbf{J})\ \dot{\mathbf{q}}_0 \tag{3.28}$$

whereby $\dot{\mathbf{q}}_0$ is a m-dimensional angular velocity vector whose transformation by a transformation matrix $(\mathbf{I} - \mathbf{J}^{\#}\mathbf{J})$, also called *null space operator*, is added to the *Minimum Norm Solution*. $\dot{\mathbf{q}}_0$ can be any angular velocity vector, that changes the configuration of the manipulator and thereby prevents that $\mathbf{J}^{\#}$ becomes singular. The projection of $\dot{\mathbf{q}}_0$ by the *null space operator* causes, that the configuration changes, i.e. the joint angles change but not the position and orientation of the end-effector!

In Equation 3.28 the two input values $\mathbf{J}$ as well as the $\dot{\mathbf{q}}_0$ are yet unspecified. In the following methods are presented to obtain these values.

Determination of the Geometric Jacobian As has been seen before it is necessary to have a geometric Jacobian matrix of the manipulator. One way to obtain this Jacobian is described in the *Method of Sciavicco and Siciliano* in [SS96].

Thereby the geometric Jacobian $\mathbf{J}$ can be calculated straight from the geometry of the manipulator. For the calculation of $\mathbf{J}$ a well-know law from mechanics is used

$$\mathbf{v} = \boldsymbol{\omega} \times \mathbf{r} \tag{3.29}$$

that describes the velocity of a point mass, that is rotated around an axis $\boldsymbol{\omega}$ with the angular velocity $|\boldsymbol{\omega}|$ and having the distance $|\mathbf{r}|$ from the axis.

For the calculation of the translational portion in $\mathbf{J}$ (upper half of the matrix), Equation 3.29 is applied to each joint, whereby the rotation axis of the joint matches the z-axis of the according coordinate system. For the calculation of the rotational portion the normalized z-base vectors (rotation axis = orientation and direction of the angular velocity vectors) of the single coordinate systems the joints are transformed to the world coordinate system.

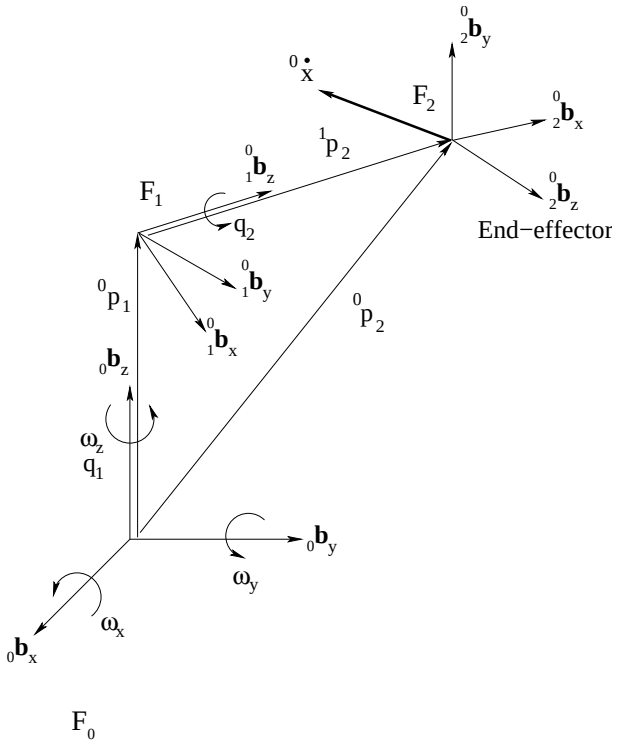


Figure 3.2: Calculation of the geometric Jacobian for a manipulator with two joints

Mathematically this can be described by using $\boldsymbol{\omega}_i = \mathbf{z}_i \cdot q_i$:

$$\mathbf{A}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{z}_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{p}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{p}_i(\mathbf{q}) = \mathbf{A}_0 \cdot {}^0_i\mathbf{T}(\mathbf{q}) \cdot \mathbf{p}_0 \quad \text{with} \quad i = 1 \ldots m \tag{3.30}$$

$$\mathbf{z}_i(\mathbf{q}) = \mathbf{A}_0 \cdot {}^0_i\mathbf{T}(\mathbf{q}) \cdot \mathbf{z}_0 \quad \text{with} \quad i = 1 \ldots (m-1) \quad \Rightarrow |\mathbf{z}_i(\mathbf{q})| = 1 \tag{3.31}$$

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \mathbf{p}_m(\mathbf{q}), & \mathbf{z}_1(\mathbf{q}) \times (\mathbf{p}_m(\mathbf{q}) - \mathbf{p}_1(\mathbf{q})), & \ldots, & \mathbf{z}_{m-1}(\mathbf{q}) \times (\mathbf{p}_m(\mathbf{q}) - \mathbf{p}_{m-1}(\mathbf{q})) \\ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, & \mathbf{z}_1(\mathbf{q}), & \ldots, & \mathbf{z}_{m-1}(\mathbf{q}) \end{bmatrix} \tag{3.32}$$

It can be seen from equation Equation 3.32 that $\mathbf{J}$ only depends from the joint angles $\mathbf{q}$ and the geometric distances, that are described in ${}^0_i\mathbf{T}$.

Performance Criterions For the differential inverse kinematics it is necessary to determine the vector $\dot{\mathbf{q}}_0$. Mainly it shall be avoided thereby that $\mathbf{J}^{\#}$ becomes singular. In the calculation of $\dot{\mathbf{q}}_0$ it is also possible to consider other criteria, that change the configuration of the manipulator, namely criteria like:

- Avoidance of singularities of matrix $\mathbf{J}^{\#}$
- Avoidance of joint limits
- Collision detection between manipulator and surrounding objects
- ...

In the following only the first two points are treated. Since the collision detection implemented as a performance criterion causes a huge calculational effort (see [Mai99] for more details). A simpler way for the collision detection will be described later in Section 4.6.2.3.

For $\dot{\mathbf{q}}_0$ the following is defined:

$$\dot{\mathbf{q}}_0 = \dot{\mathbf{q}}_0(t) = \nabla H(\mathbf{q}(t)) \tag{3.33}$$

where H is a sum of so-called *Performance Criteria* H_i weighted by k_i, that shall influence the configuration of the manipulator.

$$H(\mathbf{q}(t)) = \sum_i k_i H_i(\mathbf{q}(t)) \quad \text{with} \quad k_i = const \in \mathcal{R} \tag{3.34}$$

k_i and H_i have to be chosen in a way to obtain values for $\mathbf{q}_0$ which are suitable for the avoidance of above mentioned states. That means, if it is detected from the timely course of $\mathbf{q}$, that for velocity $\dot{\mathbf{q}}$ an illegal state is reached, $\mathbf{q}_0$ has to counteract the velocity $\dot{\mathbf{q}}$.

$\dot{\mathbf{q}}_0$ is obtained by construction of the gradient of Equation 3.34:

$$\dot{\mathbf{q}}_0 = \nabla H(\mathbf{q}(t)) = \sum_i k_i \, \nabla H_i(\mathbf{q}(t)) \tag{3.35}$$

For the criteria, *joint limit avoidance* and *singularity avoidance* H is defined by:

$$H(\mathbf{q}(t)) = k_l H_l(\mathbf{q}(t)) + k_s H_s(\mathbf{q}(t)) \tag{3.36}$$

Unfortunately, this method has a drawback: By summing up the criteria, single criteria can influence or even nullify each other! Therefore, it is especially necessary for the joint limit avoidance to check joint angles for sensible values after calculating the inverse kinematics and the integration of the angular velocities.

Furthermore, by using performance criteria the differential inverse kinematics tends to instable behavior if the weighting factors are chosen disadvantageously (see [Mai99] for details). To avoid this, appropriate values, in general experimentally determined, have to be found for the weighting factors k_i. By limiting $|k_i \, \nabla H_i(\mathbf{q}(t))|$ to a maximal value the tendency for oscillations can be further suppressed. This maximal value has to be determined for each criterion experimentally.

Joint Limit Avoidance For the joint limit avoidance following function H_l can be used:

$$H_l(\mathbf{q}) = \sum_{i=1}^{m} \frac{q_{i,min} - q_{i,max}}{(q_{i,max} - q_i)(q_i - q_{i,min})} \tag{3.37}$$

The value for k_l has therefore to be fixed to:

$$k_l > 0 \tag{3.38}$$

The gradient of H_l assures that the configuration of the manipulator changes if the angle of one joint reaches the border of its allowed range of values. In Figure 3.3 it can be seen,

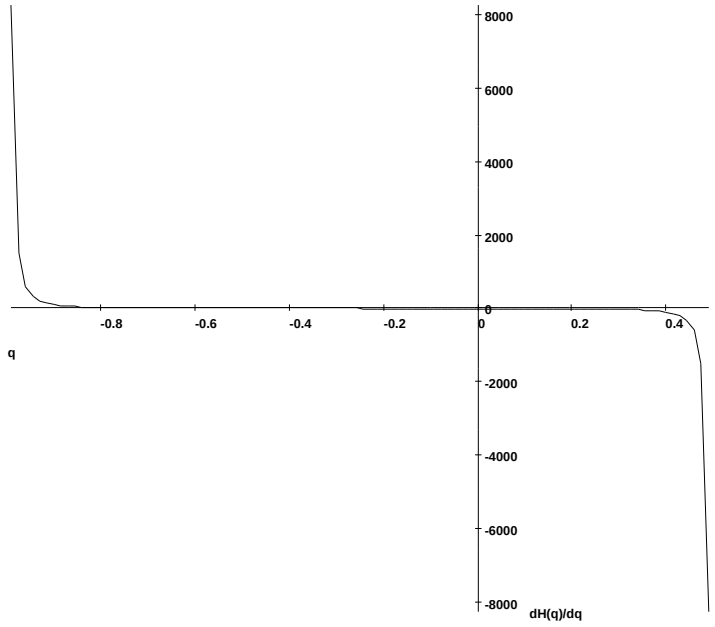


Figure 3.3: Example for $\frac{\delta H_l(q_i)}{\delta q_i}$ with $q_{i,max} = 0.5$ and $q_{i,min} = -1$

that at the borders of the allowed range of values, values for $(\nabla H_l(\mathbf{q}))_i$ are generated that counteract this critical condition.

$$(\nabla H_l(\mathbf{q}))_i = \frac{\delta H_l(q_i)}{\delta q_i} = \frac{(q_{i,max} - q_{i,min})(q_{i,max} + q_{i,min} - 2q_i)}{(q_{i,max} - q_i)^2(q_i - q_{i,min})^2} \quad \text{with} \quad i = 1 \ldots m \quad (3.39)$$

Singularity Avoidance For the avoidance of singularities the following function H_s is used:

$$H_s(\mathbf{q}) = -\frac{1}{\sqrt{\det(\mathbf{J}\,\mathbf{J}^T)}} \tag{3.40}$$

For k_s has to be assured:

$$k_s > 0 \tag{3.41}$$

Since it is not possible to calculate a derivative of the determinate straightly, the gradient is determined with the definition of the differentiation rule:

$$(\nabla H_s(\mathbf{q}))_i = \frac{\delta H_s(\mathbf{q})}{\delta q_i} = \lim_{\Delta q_i \to 0} \frac{H_s(\mathbf{q}+\mathbf{v}\Delta q_i) - H_s(\mathbf{q})}{\Delta q_i} \tag{3.42}$$

$$\text{with} \quad \mathbf{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix}; \quad v_k = \begin{cases} 1, & k = i \\ 0, & k \neq i \end{cases} \quad ; \quad k, i = 1 \ldots m$$

If for a positive Δq_i for an angle $\mathbf{q}$ the criterion $H_s(\mathbf{q}+\mathbf{v}\Delta q_i)$ gets more negative than $H_s(\mathbf{q})$, i.e. the manipulator gets even closer to a singularity, the according element $(\nabla H_s(\mathbf{q}))_i$ is negative and counteracts the approach of the singularity of $\mathbf{J}$.

By the numerical calculation of $\nabla H_s(\mathbf{q})$, Δq_i can be used as a constant (e.g. $0.1°$) or alternatively chosen according to the current movement direction of the joint (e.g. $\pm 0.1°$). The second possibility can possibly generate better results, since the gradient is calculated for the direction of the current joint angle motion.

Models of the Camera and the Head

In the robotics and computer vision communities, video (CCD) cameras are described using a range of models. A good overview of these camera models is given in [MZ92] or [SZB95]. From the point of view of hand-eye coordination, some of the models are described in a visual servoing tutorial by Hutchinson et al. [HHC96].

The most commonly used camera model is the *pinhole camera model*. It is appealing because of its simplicity compared to other used camera models (e.g. the *projective camera model* or *camera models approximating perspective projection*, e.g. the *affine camera model*), but nevertheless the accuracy of measurements obtained with this model is very high.

For the "head", where the cameras are mounted, usually pan-tilt units are used. Thereby the camera(s) are fixed on these units. A pan-tilt head can by modeled using Denavit-Hartenberg parameters as described before for the manipulator. Thereby the pan-tilt head is seen as a two joint manipulator.

How the head and the cameras are located relative to the manipulator depends on the hand-eye configuration. Two possibilies are common: either the camera is located on the manipulator (*eye-in-hand*) or a *stationary camera* is used. In the first case the camera is usually fixed relative to the manipulator, i.e. it moves only if the manipulator moves. In the second case the camera(s) can be moved independently from the manipulator.

Since camera and head models are not in the main focus of this thesis the topic is not treated further here.

3.1.2 Models of the Object to be Grasped

If object models are to be used in the context of hand-eye coordination, they have to meet two requirements: First, they have to contain information by which the objects can be recognized in the images; secondly, they have to provide a way to store grasp positions (e.g. to track a grasp as described in Section 4.4.1).

In the context of the first task, object recognition, two orthogonal approaches for object representation have evolved: *appearance-based* representations are "learned" from a set of images of an object, taken from different poses and different lightening conditions, implicitly taking into account surface properties like texture or reflectance. Most appearance-based systems use *global features*, such as the *area*, *color* or the *compactness* of an object or rather of the region in the image corresponding to the object. Therefore global features summarize information about the entire visible part of an object. The identification process is thus reduced to comparing the detected image features with those from the model data base and using a measure of difference for classification, which makes such methods very fast. Unfortunately, global features are very sensitive to occlusion and require almost perfect segmentation, which is problematic in the case of cluttered scenes. A wide variety of approaches exist, differing in which image information is used and how data is stored. These approaches range from aspect graphs based on geometrical features and their topological relations [Pop95] to an eigenspace representation on pixel value level [MN95]. Appearance–based approaches facilitate the matching process, since the data compared is very similar from the start;

Geometric representations maintain a 2D or 3D model of the entire object with descriptions that vary from simple ones such as B-Spline curves or triangulated surfaces to more complex ones, such as superquadrics [RDR94], algebraic surfaces [KP94] or generalized cylinders [ZM94]. Geometric models permit the construction of large databases, enable part–based descriptions and, therefore, can be used to describe generalized objects and ob-

ject classes. Geometric models also assist the segmentation process in a top–down manner by predicting views of the object. Geometric representations rely mostly on *local features*, such as line segments or junctions which are often associated with geometric systems. Local features permit recognition even in cluttered scenes [BI98], but require additional stages in the identification process, such as perceptual organization [Low85], establishing of correspondences between image and model features and verification of hypotheses [BI98].

3.1.3 Models of Object Motion

Natural motions underly physical principles. As a simple example a free falling stone can be taken, whose way down to the earth surface can be described by solving the equations for energy conservation. Other examples are a swinging pendulum [OPB+98] or a thrown ball, whose trajectory can be described by the equations of a parabola [N+98].

Also for more complex motions like bouncing balls or objects hitting edges (like in billiard or air-hockey [BS98]) governing equations can be found. In general for all motions that occur in nature equations describing their behavior have been found or can be established, respectively.

It is obvious that in order to build a catching system that shall be able to grasp objects moving in different, not a priori known manners, either models (equations) for all possible motions have to be stored in a data base or more general models that can deal with different object motions have to be provided. The first method would require that the system is able to distinguish from only few measurements which model is to choose from the data base, and then fit the parameters to apply the equations for the current motion. The second method would always take the same set of equations and adapt its parameters to the current motion.

Methods to described object motions, and predict their future motion using the second method are described later in Sec. 3.3.

3.2 Vision

3.2.1 Tracking

Research into tracking has diverged into two camps - informally these can be distinguished as *low-level* vs. *high-level* approaches. Low-level approaches are typically fast and robust, but provide little fine-scale information, whereas high-level approaches can track complex deformations in high-dimensional spaces, but must trade speed against robustness [IB98]. Low-level approaches include “blob-trackers” e.g. for color, or methods like “optic flow”. They are fast, typically e.g. due to hardware support, and robust, but convey little infor-

mation other than object size, centroid and/or main axis orientation (with some additional calculational effort).

For high-level approaches additional a priori information about the object is necessary. This information can be gained either by modeling objects with specific gray-level templates [BJ96] which may be allowed to deform [HT96] or by modeling objects with more abstract templates such as curved outlines [BIR95]. Contour trackers have been constructed which are robust to clutter, e.g. in [BI98] but only by sacrificing real-time performance. A contour tracker working at real-time, but only with a special contour shape, an ellipse, is presented by Vincze [VAZ00].

In the following sections low-level tracking methods for form (see Section 3.2.1.1), color (see Section 4.2.2) and motion (see Section 4.2.3) as well as high-level methods (see Section 3.2.1.1) and the combination of low- and high-level methods (see Section 3.2.2) are presented.

3.2.1.1 Contour-based Tracking

In order to locate an object in an image, the aim will be to detect the real object's contour in the image data. There are several ways to extract features from an given image in order to estimate observed outlines. By use of different filtering methods ridges, valleys, edges or uniformly colored regions can be extracted partially via simple but fast *low-level feature detection* procedures e.g. thresholding. The contour of an object can be estimated very well if the object's outline is clearly visible and not disturbed by clutter. In special applications the possibility to back-light the object or manipulate the object's background is used to receive a maximum contrast between the real object's contour and the rest of the image. Unfortunately, this is not always simply applicable. Clutter or the presence of distractor objects makes it difficult to detect target objects without the use of sophisticated *high-level* methods, which are computationally expensive.

In order to track a contour in a sequence of images, the complexity of the problem rises due to possible changes in the object's outline. Given a clutter-free image, the contour can be estimated in each image separately without problem. In reality one would wish to use the estimated contour information provided by the real object's outline detected in the preceding frames. It is clear that this given contour is *prior information* for the search in the new image frame, which should not be neglected in a cluttered environment.

Obviously, the shape of the same moving 3D object will change in the history of the image sequence. One could consider the changes in size of an approaching object. If the estimated initial contour will not be able to follow these changes, the search for a new arbitrary outline on the dense *feature map*[2] cannot be successful.

[2]The feature map *here* is the output of the filtered image. The desired features (e.g. edges) can be found there if the appropriate filter is applied. Note that not only the object's contour will be detected if

Active contour models[3] or *Snakes* can overcome these problems and are able to adapt a given contour to small variations.

A framework to integrate active contour models into a probabilistic tracking algorithm is the *CONDENSATION algorithm* which is presented shortly in Section 3.2.1.1.

Snakes

> "Snakes are a mechanism for bringing a certain degree of prior knowledge to bear on low-level image interpretation. Rather than expecting desirable properties such as continuity and smoothness to emerge from image data, those properties are imposed from the start."(see [BI98] p.27)

Snake contours can be described through the following minimization problem

$$\underbrace{\left[\frac{\delta(w_1\mathbf{r})}{\delta s} - \frac{\delta^2(w_2\mathbf{r})}{\delta s^2}\right]}_{\text{internal force}} + \underbrace{\nabla F}_{\text{external force}} = 0 \tag{3.43}$$

in which the mechanism of elastic adaption of curves becomes clear.

The contour $\mathbf{r} = \mathbf{r}(\mathbf{s})$ in this *static formulation* describes the parameterized outline of the object, supposed to be elastic and smooth. These *internal properties*[4] want to maintain a certain kind of shape, while the external force - described as ∇F - wants to drag the contour to the highest responses of the feature map.

Generally spoken:

internal force: describes the tendency of the snake to stay smooth and continuous as the information provided a priori.

- Increasing the parameter $w_1 = w_1(s)$ makes the snake behave like stretched elastic, but increases the tendency of shortness.
- Increasing $w_2 = w_2(s)$ encourages the snake to be smooth. Thus, setting $w_2(s_0) = 0$ will introduce a kink at this point.

external force: describes the tendency of the snake being dragged to the extracted features of the image. In a cluttered environment, the highest response of the feature map will **not** provide the solution itself.

---

the background is not clutter free.

[3]Active contours encompass more than the described snakes or B-splines. For further information see the now classical paper by Kass, Witkin and Terzopoulos [KWT87].

[4]It is also called internal energy.

The trade-off between the restoring internal and the measured external forces can provide a good solution.

Expanded to the dynamic case of contour tracking in image sequences, Equation 3.43 turns into

$$\underbrace{\rho \mathbf{r}_{tt}}_{\text{inertial force}} = \underbrace{-\left[\gamma \mathbf{r}_t - \frac{\delta(w_1 \mathbf{r})}{\delta s} - \frac{\delta^2(w_2 \mathbf{r})}{\delta s^2}\right]}_{\text{internal force}} + \underbrace{\nabla F}_{\text{external force}} \tag{3.44}$$

Note that $\mathbf{r} = \mathbf{r}(s,t)$ and $\mathbf{F} = \mathbf{F}(t)$ describe the time-variant contour and feature map, respectively. ρ, which is called *mass density* due to the mechanical analog, and γ (named *viscous resistance* of the surrounding environment) provide further possibilities to adjust the snake model to the so-called *Newton's law of motion for snakes with mass, driven by internal and external forces* (see [BI98]).

An arising problem with snake models (besides the non-trivial search for the right parameters) is that the outline may change *dramatically* in the elapse of time. E.g. if the contour looses lock on the real object, the outline may become too tangled to recover again.

This problem can be overcome by the use of B-splines, whereby the practical advantage of using a B-spline shape model is the possible neglect of the smoothness terms in Equation 3.43 or Equation 3.44, respectively. Derivatives are computationally approximated as finite differences between the assembly of discrete points $\mathbf{r}(s_i), i = (0, \ldots, h)$, on the continuous outline of $\mathbf{r}(s)$[5]. Using B-splines contours, a continuous and smooth contour can be constructed out of a few control points without evaluating any finite differences.

Those B-spline snakes are the basis for other continuative algorithms, such as the successively described CONDENSATION algorithm.

CONDENSATION Algorithm

In general the CONDENSATION algorithm (CONDitional DENSitiy propagATION) [BI98] is a stochastic algorithm that is able to propagate an entire probability distribution (p.d.) for object position and shape. This p.d. is represented by a set of *samples* $\{s_{k-1}^{(n)}, \pi_{k-1}^{(n)}\}$, each representing a certain *state* $\mathcal{X}$ of the object, randomly generated (*selection*) through a process called "factored sampling" [BI98]. In image processing applications a state $\mathcal{X}$ is formed by two consecutive *shape vectors* $\mathbf{X}$, $\mathcal{X} = (\mathbf{X}_t, \mathbf{X}_{t-1})$, whereby a shape vector is a multi-dimensional vector of curve parameters. All possible states of the object span up the *state space*[6]. Standard CONDENSATION [BI98] uses off-line learned dynamical models (*prediction*), together with visual observations (*measurement*), to propagate

[5] h is the number of sampled points on the contour. Thus, h is describing the resolution.

[6] Note: If the sample-set dense enough it is a good approximation to the state space

the random sample set over time. At the end of each cycle (selection, prediction, measurement) every sample is given a new weight π_m according to its measured probability to represent the real object, and a new sample-set is obtained $\{s_k^{(n)}, \pi_k^{(n)}\}$. The p.d. at the start of a cycle is called *prior density* and *posterior density* at the end.

In image processing, each sample is represented by one affine transformation of an object model, an active contour represented by a B-spline $\mathbf{r}(s)$:

$$\mathbf{r}(s) = \mathbf{U}(s)\mathbf{W}\mathbf{X} + \mathbf{r_0}(s) \tag{3.45}$$

where $\mathbf{U}(s)$ is a matrix mapping the control point vector $\mathbf{Q}$ to the image curve $\mathbf{r}(s)$, $\mathbf{W}$ is a shape matrix, $\mathbf{X}$ is the shape vector[7] and $\mathbf{r_0}(s)$ is the initial B-spline curve (*object model*).

In Figure 3.4 one cycle in the CONDENSATION algorithm can be seen.

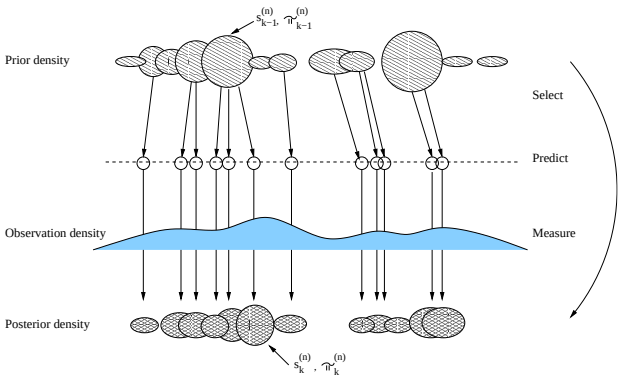


Figure 3.4: One time step in the CONDENSATION algorithm (adopted from [BI98]

[7]To give an example for a simple shape matrix and a shape vector: The shape matrix of a restricted two-dimensional shape space which exclusively allows translational movement looks like

$$\mathbf{W} = \underbrace{\begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{pmatrix}}_{2N_B \times 2} \tag{3.46}$$

An adequate shape vector for this space looks like

$$\mathbf{X} = \underbrace{\begin{pmatrix} t_x \\ t_y \end{pmatrix}}_{2\times 1} \tag{3.47}$$

where t_x and t_y describe the translation in x and y (pixel-)coordinates. A new translated control vector will be calculated out of the old one,

$$\begin{pmatrix} \mathbf{Q^x} \\ \mathbf{Q^y} \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \begin{pmatrix} t_x \\ t_y \end{pmatrix} + \begin{pmatrix} \mathbf{Q_0^x} \\ \mathbf{Q_0^y} \end{pmatrix} \tag{3.48}$$

Since the above explanations can give only a short overview over the CONDENSATION algorithm the interested reader is encourage to read [BI98] or [Osw00], respectively.

3.2.1.2 Color-based Tracking

Color has been widely used in machine vision systems for tasks such as image segmentation, object recognition and tracking. The following sections cover the topics of how to represent color in a computer (Section 3.2.1.2) and how to search for colors (Section 3.2.1.2). Experiments and results with different color spaces and segmentation methods are described in Section 3.2.1.2. Detailed results on tracking with color can be found in Section 5.1.1.

Color spaces The term color is closely related to the wavelength of electro-magnetic waves. The visible spectrum for humans goes from 400nm up to 700nm. Humans learn that objects reflecting electro-magnetic waves at about 558nm have the property "red". As has been mentioned in Section 2.1.1 there are three variations of day-light sensitive photo-receptive cells on the human retina which differ in their spectral sensibility. Those photo-receptive cells are called blue (short wavelength), green (medium wavelength) and red (long wavelength) cones.

In simple CCD color cameras the situation is quite similar to the human retina. Those cameras are equipped with CCD chips were the vertical rows are alternating sensitive for red, green and blue. This is obtained through the use of color filters in front of the CCD pixels, which filter out the complementary colors.

There exist many possibilities of how to describe colors in a machine compatible way. The mathematical term to describe colors in a machine is "color space". A color space should be capable to represent almost all visible colors of our environment.

Common to all color spaces is that they try to describe colors by several parameters. The way these parameters are chosen reflect the intended use of the specific color space. The number of parameters and the dimension of the color space are equal.

A common problem of color spaces is quantization. Normally, integer values are used to represent the parameters to save memory and processing time. Therefore, not all possible colors can be represented with integer based color spaces. In the following the most common color spaces are shortly introduced.

RGB color space. The RBG color space is a hardware-oriented color space. It uses three parameters to describe the red, green and blue components of an electro-magnetic wave. This concept of superposition is justified as the human eye also uses superpositions of three base colors to see the whole spectrum.

The drawback for color segmenting techniques is that similar colors are not always described

by similar parameters. For example to get all different tones of red one has to sweep not only through all variations of the red component, but also through some of the other components. A bluish red and a greenish red are still recognized by humans as red, but the red-green-blue values may differ quite a lot. Another well known, but still not generally solved problem, is the dependency of all three parameters on illumination changes.

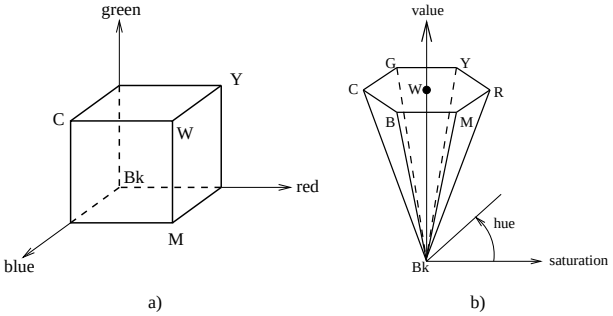


Figure 3.5: (a) the RGB color model and (b) the HSV model (both adopted from [Kra98]). [8]

Figure 3.5(a) shows the color space as a three dimensional cube with the axes representing the parameters red, green and blue and the origin representing black.

YUV color space. The YUV color space is another hardware-oriented color space. The Y parameter describes the luminance value, U and V describe the chrominance value of a pixel. This color space is often used in video applications (a very similar color space is called YIQ that differs only in a rotation of chrominance values [Sol97]). The chrominance values U and V represent the color differences B-Y and R-Y. The third color component green can be calculated out of Y, U and V.

The transformation between RGB and YUV color space can be calculated by first multiplying a matrix on the RGB color vector to get the luminance and color difference values:

$$\begin{pmatrix} Y \\ R-Y \\ B-Y \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.701 & -0.587 & -0.144 \\ -0.299 & -0.587 & 0.886 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

U and V can then be obtained by multiplying the color differences:

$$\begin{aligned} U &= 0.493(B-Y) \\ V &= 0.887(R-Y) \end{aligned}$$

Similar colors can be found in more *compact subspaces* of the YUV color space than in the RGB color space. Changes in illumination intensity almost only affect the lumination value Y, but not the chrominance values U and V. Therefore, this color space is convenient for fast color tracking.

[8] R = red, G = green, B = blue, C = Cyan, M = Magenta, Y = yellow, W = white, Bk = black

HSV color space. The HSV (Hue, Saturation, Value) color space is a more user-oriented color space. Image manipulation programs often offer slider to choose a color in HSV as well as in RGB color space (for example The GIMP [KM]).

The Hue and Saturation parameters determine the color, with Hue representing the wavelength (red, yellow, ...) and Saturation indicating the strength of the color (compared to a gray tone with equal value), while Value represents the intensity (the grade of illumination) of the color.

Figure 3.5(b) gives a three dimensional representation of the HSV color space. The black point is at the apex. If Value is zero, Hue and Saturation do not change the color at all. Hue is the angle between the saturation axes and the color vector projected onto the $S \times H$ plane. Saturation is the length of this projected vector.

The way of selecting first the desired color by choosing hue and saturation and then to choose the value parameter is easier to learn for a human to find a desired color.

The major drawback for using this color space is the increased calculation effort for transforming from and to color spaces used in machine environments. To get Hue, Saturation and Value of every pixel, you have to apply the following three equations [KC99] on every pixel:

$$H = acos \left[\frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\sqrt{(R-G)^2+(R-B)\,(G-B)}} \right]$$

$$S = 1 - \frac{3}{(R+G+B)} min\,(R,G,B)$$

$$V = \frac{1}{3}\,(R+G+B)$$

Other color spaces The number of used color spaces is enormous: the HLS, the HSI, NCC, HDI, CIELUV etcetera. For the interested reader the book [Rus95] or [Haf98] is recommended.

- HLS color space

  The HLS (Hue, Lightness, Saturation) color space is similar to the HSV color space. Hue identical to the HSV definition. See [Kra98] for more information.

- HSI color space

  The HSI (Hue, Saturation, Intensity) color space wants to combine hardware compatibility and user friendliness. Hue is the same as in HSV. Intensity is defined as $I = (R+G+B)/3$. See [Kra98] for more information.

- NCC color space

  The NCC (Normalized Color Components) color space uses only two parameters. Colors are normalized in respect to their intensity. Therefore two parameters are enough to describe a color. See [Kra98] for more information.

- HDI color space

  The HDI (Hue, Distance, Intensity) color space is used in [Kra98] for segmentation purposes. Hue is the same as in HSV. Distance is the distance from the RGB point to the gray diagonal in the RGB cube. Intensity is similar to HSI. See [Kra98] for more information.

- CIELUV and CIELAB

  These two color spaces are designed to be device-independent and perceptually uniform. The involve rather complex transformation operations. See [HC93] for more information about using CIELUV in image color segmentation.

- CIE system for XYZ color space

  This color space is based on the description of color as a luminance component Y, and two additional components X and Z. The spectral weighting curves of X and Z are calculated from statistics of experiments involving human observers. The XYZ tristimulus can describe any color. The magnitudes of the XYZ components are proportional to physical energy, but their spectral composition corresponds to the color matching characteristics of human vision. See [Haf98] for more information.

- I1 I2 I3 color space

  In [PP93] a color space based on experiments is mentioned. It is said that by transforming RGB vectors by the following matrix good color segmentation results can be obtained.

$$\begin{pmatrix} I1 \\ I2 \\ I3 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

  The values of the matrix were found using a Karhunen-Loève transformation. See [PP93, Haf98] for more information.

Segmentation methods Color segmentation tries to segment an image based on color information. Two orthogonal approaches can be mentioned in this context. The more general approach is to segment the whole image into areas holding similar colors and to cluster those pixels (pixel clustering). The alternative possibility is to look only for pixels holding a previously known color and to filter all other pixels (color filtering).

Pixel clustering is a topic of great interest and one can find many publications and works dealing with it. Most of them rely on histogram techniques to find initial cluster regions, and differ mainly in the process of fusing pixels that are not lying within the maxima of this initial histogram. Below a short overview of the methods is given.

In [ZL97] the segmentation process is separated into:

- finding color edges,
- fusing single pixels, and
- fusing regions of similar pixels and color edge information

To find color edges, a color gradient for the whole picture is calculated whose local maxima give color edge pixels. Then, local pixel clusters of similar color are fused using the spatial distance of pixels as a fusion criteria. Finally these local pixel clusters are combined with local edge information from the color gradient function.

Hedley and Yan [HY92] also propose a color gradient function. But in their method first cluster of low-gradient pixels are calculated and then high-gradient pixels are processed. Low-gradient pixels define the number of segmented regions while high-gradient pixels have to be added to the best matching, yet existing region, since they cannot initiate a new region.

In [Sch93] a more complex color space, the CIELUV, is used. After transforming the image into this color space, a one-dimensional histogram (one for each dimension of the color space) classification is computed. Peaks of these histograms define the pixels that are being processed in the next recursion of the histogram algorithm. The recursion is repeated, until the histogram becomes unimodal. Resulting pixels are clustered and subtracted from the original image before the process is restarted on the reduced image. Schettini claims that overlapping regions can be separated with this procedure. If all large pixel clusters have been processed, the found regions are merged based on distance and color criteria.

In [LY94] a multi-resolutional approach is presented. As a first step, coarse resolution histogram operations are used to get a primary segmentation of the image. Upon this segmentation, a quad-tree structure is used to implement the multi-resolution scheme. For each sub image a more fine histogram operation is processed. Neighboring homogeneous regions are merged after the image is split into regions holding only similar pixel color values.

Lim and Lee [LL90] also propose a histogram based, coarse-to-fine algorithm. They use a scale space filter to determine n clusters of a coarse histogram of the image before a fuzzy c-means algorithm computes a measure for each pixel to relate it to one of the n found clusters.

The common problem of all mentioned approaches is their high demand of processing time. Since the computational load for other algorithms used within the hand-eye coordination

task tends to be high an alternative method for segmenting colors in a computationally extensive way is presented next.

Color filters Algorithms looking only for some specific colors are called *color filters*. They are computationally efficient because every pixel of the image has to be processed only once. Therefore, the complexity of these algorithms is directly proportional to the size of the image.

Color Distribution and Color Subspaces As the distribution of color values in a natural image is very wide-spread (see Figure 3.6), there is no single color to search for. As an example in the figure the two lower images display color distributions in the RGB color space. The black point is in the left bottom corner in the back of the white cube and axes are oriented as in Figure 3.5(a). Pixels are drawn with their color at the three dimensional position of their RGB values. The left distribution displays values for all pixels in the upper image, the right distribution only pixels within the rectangular area indicated in the upper image, which marked the color we are looking for. The black pixels in the search area can't be seen in the distributions because they coincide with the black point of the cube.

Supposed, the green color of the box is the color of interest in the image, all pixels with color values within the distribution in the right image had to be included and all pixels outside excluded, i.e. the searched pixels all lie in a subspace of the current color space. In the example of Figure 3.6 this subspace roughly has the form of a cylinder, with its symmetric axis heading from the black point to some green color point.

The form of the subspace depends on the chosen color space. Therefore, the result (matching quality and speed) of the filter depends on both, the color space and on the form of the subspace. Complex subspaces require more processing time (since efficient implementation is most times not possible), while achieving better matching quality, whereas simple subspaces need less processing time, but give worse results.

CLUT versus Windowing Mechanisms A common technique to define a color subspace is to construct a color lookup table (CLUT). This table holds an entry value for every possible color value. The entry value can indicate "anything". Two common purposes are:

- Translate a color into another color. Many applications use this technique to change known features of an image. For example, medical applications enhance visibility of certain areas in images by assigning different colors to almost indistinguishable gray tones. This is very favorable since the human eye can distinguish between thousands of different colors, but only between a couple hundred grey tones [Rus95].

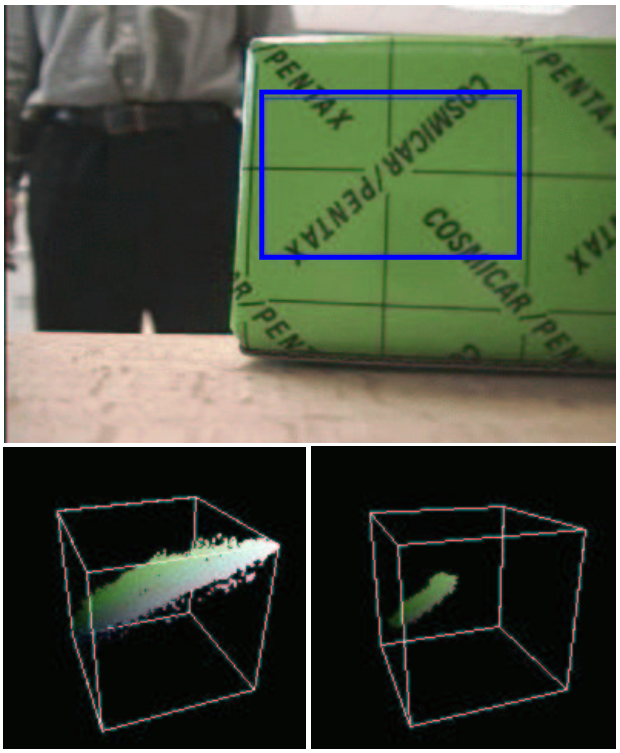


Figure 3.6: Pixel color value distribution of a sample image. On the lower half you can see two color distributions in the RGB color space. The orientation of the RGB cube is the same as in Figure 3.5, i.e. the black point is in the rear, lower, left corner. The left one displays all pixels of the image at the top, the right one displays the distribution of all the pixels within the rectangular area indicated in the upper image.

- Select certain known colors (Filtering). The value in the CLUT defines if the color corresponding to the entry is within or outside the range of colors that the filter is looking for.

Table 3.1 shows an extract of an example CLUT. With a CLUT subspaces of arbitrary form can be constructed. It is also possible to search for several colors, by assigning different values to the entries in the table. To process an image, every pixel has to be compared with the entries of the CLUT to classify if the CLUT contains an entry of the pixel's color value. From this operation it can be decided for the entry whether to filter the pixel or relate it to the result.

As mentioned, a CLUT can implement all kinds of subspaces and its processing time is equal for all of them. Processing speed depends on memory access time of the hardware used. Having rather simple subspaces, such as cubes, spheres or ellipsoids, it is not necessarily needed to read CLUT values from memory. If the subspace can be modeled with few

[9] In the displayed extract the red and green color values are constant, while blue values change. Some of the combinations are marked as inside the subspace (1), while others are interpreted as outside (0).

red	green	blue	CLUT value
...	...	...	...
103	204	3	0
103	204	4	0
103	204	5	1
103	204	6	1
103	204	7	0
...	...	...	...

Table 3.1: Example entries of a CLUT[9]

parameters, an equation can be solved for every pixel's color value that determines if the pixel is within or out of the subspace. A spheric subspace can be modeled by a center point and its radius, a cylinder by its symmetric axis and its radius, etcetera. The easiest equation to solve for the pixels is a *windowing mechanism*, describing a cube as subspace. Figure 3.7 shows a cubic subspace with its minimum and maximum parameters.

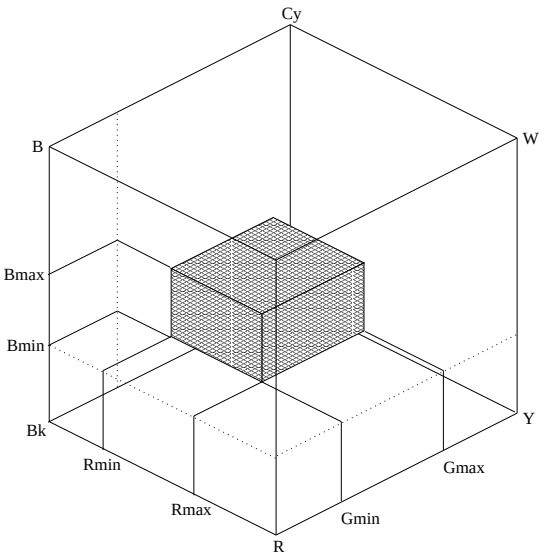


Figure 3.7: Cubic subspace, modeled by a windowing mechanism

In the case of a cubic subspace a windowing mechanism preserves memory and processing time. The windowing algorithm loads the pixel color value and compares each component of the color vector to chosen minimum and maximum values. These upper and lower values

constitute a window in every dimension of the color space. Color vectors lying outside the window are filtered. Prerequisite for the windowing algorithm is that a color space is used that justifies a cubic subspace as the color search subspace (this e.g. not the case for RGB as can be seen from the example in Figure 3.6).

The Dichromatic Reflection Model Klinker, Shafer and Kanade [KSK90] have proposed the Dichromatic Reflection Model (DRM), which tries to form the subspace in a way to incorporate illumination effects.

Scene illumination has an influence on the appearance of object colors. This is particularly true for dichromatic surfaces such as plastic, paper or ceramics. Dichromatic objects have pigments incorporated into their molecular structure which reflect light with their proper color. As an unwished side effect the surface also reflects the illuminating light. As both mechanisms depend on the angle of the incoming light and neither the pigments nor the surface are present in a perfectly aligned structure, the resulting reflected color always is a linear combination of pigment and illumination color. Supposed that the illumination color forms a vector in a color space, as well as the pigment color, the resulting reflected colors lie in the plane spanned by those two vectors. Therefor a CLUT with a subspace forming a plane can now be used to filter all colors the object is reflecting.

3.2.1.3 Motion-based Tracking

Obviously, *motion* is pre-destinated to be used for tracking and is an important part of machine vision. Thereby, the vector field of two-dimensional velocities in the image, also called *optic flow*, is dependent on the three-dimensional motion in the observed scene. The analysis of this flow field can be used for many applications, e.g. for tracking of an object in motion as needed here, but also in e.g. video-based navigation as described in more detail in [Stö01].

The methods proposed in the literature to determine the *optic flow* can be separated roughly into three different ([Stö01]):

- differential or gradient-based methods,
- correlative methods and
- feature tracking methods.

Gradient-based methods suppose time- and position continuous brightness changes on the image plane. The result is a continuously defined velocity flow field. For image sequences the interrelations can be discretised if the sampling intervals in space and time are chosen small enough compared to the velocities of the tracked object.

Correlation and feature based methods suppose constancy of the intensity distribution in small neighborhoods around an image point. Those methods try to find point-to-point correspondences between image pairs and can therefore be used directly for image sequences (and therefore for tracking). In general any pair of images can be used, e.g. images that were captured at the same time with different cameras. But the larger the timely distance between two images gets, the larger is the expense of the correspondence search through the enlargement of search areas and the increase of ambiguities.

Apart from these methods there exist also techniques that cannot be assigned to one of the aforementioned. For the more interested reader a more detailed overview is given in [Stö01].

But also more simple methods to determine image motion exist, and were proven to be used successfully for tracking (and catching). Thereby the use of *image differences* can be mentioned: by subtracting two successive images only parts that have changed are content of the resulting image. As an example for a robotic catching system using this method the "Robotic ball catcher" described in [FHH01] can be stated.

The combination of optic flow methods in junction with active contours has been proposed e.g. in [KHL99]. Thereby the object of interest is being tracked using a snake model. In case that the object is lost in consequence of a jump optic flow is used to determine the jump and reinitialize the snake.

3.2.2 Sensor Fusion and Integration

From the aforementioned the importance of intelligent sensor fusion gets evident: Having different sensor modalities, each delivering interesting information about an image, the remaining question is how to fuse or integrate this information. Naturally, this is a well known and well treated question in the robotics literature.

Sensor fusion is the main form of data fusion in robotic systems and has been deployed in many applications such as mobile robot localization, manipulation and telerobotics. To define the term, following is stated as

Sensor Fusion: [CM99] "evidence from multiple sensors is combined in order to produce information which is more precise than the output from an individual sensor; the fusion may involve raw sensor signals, or features and attributes extracted through preprocessing of the raw signals".

In the same way the integration of sensor signals can be defined as

(Multi-) Sensor Integration: [CM99] "data obtained from multiple sensors is combined for the purpose of tracking (and identification of multiple targets)."

To integrate multiple sensor information to generate an output about the assumed object's

position and shape many methods exist. One method, a probabilistic approach to this problem, is the ICONDENSATION algorithm which is described in the next paragraph.

ICONDENSATION Algorithm The ICONDENSATION algorithm was mainly developed to (1) combine low- and high-level information in a consistent probabilistic framework, using the statistical technique of *importance sampling* combined with the CONDENSATION algorithm [IB98] and (2) to overcome real-time performance problems of the CONDENSATION algorithm.

Importance sampling offers a mathematically principled way of directing search, combining prediction information based on the previous object position and motion with any additional knowledge which may be available from auxiliary sensors (→ multi-sensor integration). On the one hand this combination confers robustness to temporarily sensor failures in one of the measurement processes, on the other hand the tracker can take advantage of different information sources, each having distinct qualities.

The drawback of the CONDENSATION algorithm was that the areas of the image (the portions of the state space, respectively) which are to be examined are determined *before* any measurements are made. This is appropriate when the sample-set approximation to the state density is sufficiently accurate. In principle, as the state density evolves over time, the random nature of the motion model[10] induces some non-zero probability everywhere in the state space that the object is present at this point. With a sufficiently good sample-set approximation this would tend to cause all areas of state space to lie near some samples, so even motions which were extremely unlikely would be detected, and could therefore be tracked. In practice most samples will concentrate near the most likely object positions, building up distinct clusters. The result will be that large areas of the state space (and therefore of the image) will not contain any sample at all. In order to robustly track sudden motions of the tracked object either the process noise of the motion model must be artificially high or, what is the case here, this lack of information must be filled by additional sensor information.

Here the above mentioned method of importance sampling is adopted, to improve the efficiency of factored sampling. It applies when auxiliary sensor information is available in form of an importance function $g(x)$ describing which areas of the state space contain most information about the posterior density. The idea is then to concentrate samples in those areas of the state space, generating sample positions $s^{(n)}$ from $g(x)$ rather than sampling from the prior density. The desired effect is to avoid as far as possible generating any samples with low weights, since they are "wasted" in the factored sampling representation as they provide negligible contribution to the posterior density.

[10] With the later introduced Auto-regressive models this can be ensured.

3.2.3 Grasp Determination

Tracking the object to be catched is necessary to have knowledge about its position in space. But still the question arises where the end-effector has to be placed on the object in order to smoothly grasp and manipulate the object. Therefore *grasp determination* is necessary.

It is obvious that manipulation of objects is a fundamental class of operations in many areas of robotics. The term includes both *fixturing* –restraining an object with the fingers of the robot hand– and *dexterous manipulation* –use of the fingers to change the position of the object within the robot arm– [BK00]. Here, the terms *grasping* and *grasp determination* will be used in the sense of fixturing.

In manipulation, the ability of the robotic system to deal with known and unknown objects is very important. In the latter case, the system must be able to use its sensors to detect and extract the necessary information about the object. This information will be used to decide for the most convenient way, possibly constrained through the task, how to grasp the object.

The stability of a grasp executed on an object depends on the forces exerted by the gripper. The number and types of these forces depends on how the contacts between the object and the robot hand are modeled [BK00]. Works on grasp stability have distinguished between *fingertip grasps* –in which each finger of the robot hand ideally contacts with the object at a point– and *enveloping grasps* –which are formed by wrapping the fingers around the object [BK00]. In these works, the grasp stability is often analyzed in terms of *force* and *form closure* conditions [Ngu88], which ensure grasp stability assuming point contacts with friction. Nevertheless, the search for stable grasps on the object based on force analysis often requires solving complex operations.

For this reason, many works base their search on the analysis of the geometric properties of the object.

Although predefined object models have been used by many [FP91], the object description is often extracted from vision data. In this case, the problem is how to extract such a description. In addition, if the relative position between the object and the camera changes, because one of them -or both- is moving, there is, in order to control the approach of the gripper to the object, a need to track not only the object itself but also the points at which it should be grasped.

Another difficulty is locating the object and extracting a description of it for the grasp search. Usually, images where there is a clear distinction between the object and the rest of the scene have been used, in order to extract a description as accurate as possible [MRSdP01, PSMP00, SdPIR98]. Nevertheless, although there has been a lot of effort over the last years to relax the working conditions (including light and type of background) of this segmentation, its precision is still highly dependent on them.

The use of deform-able contours to represent its shape [KHL99, BI98, IB98] has proved useful in some works, since they can also be used for tracking the object along a sequence of images [PSMP00]. In contrast to that, some works have based their grasp search on the skeleton of the object shape [HRSF99, Jar88]. But nevertheless, the use of the contour is more common [MRSdP01, PSMP00, SdPIR98].

However, many works have considered only *squeezing grasps* –executed by closing the fingers on the object– while, in some cases, *expansion grasps* –performed with an opening of the fingers– are also possible [MRSdP01]. In general, these works do not rely on models, so they do not require an identification of the object [Sta91].

Vision-based works on grasp determination usually perform a geometric analysis based on the shape of the object. Often, a number of measures of grasp stability are defined and a quality value is computed from them [MRSdP01, PSMP00, SdPIR98]. These methods are heuristic in nature, since they estimate generic values of some parameters related to the force closure (friction, force to apply to the object,...) based on either vision or previous experiments.

3.3 Motion Reconstruction and Prediction

Motion Reconstruction In order to catch a moving object in a way as described in the system architecture (see Figure 1.1) the motion has to be reconstructed. Many successful catching systems restrict themselves to known object motions and therefore known motion models. Famous examples of this kind are the "Volleyball" playing robot of Nakai et al. [N+98] as well as the robotic catching system of Hong and Slotine [HHS97]. In both aforementioned systems it is supposed that a ball is thrown towards the robot. The task is here to reconstruct the current parameters of the parabolic model of the trajectory[11] each time the ball is thrown anew. This is done using recursive least square techniques.

Motion models can be found for most motions occurring in nature. In every physics book one will find the governing equations for spring-mass systems, pendulum systems, masses moved by gravitational force etcetera. In general these systems are of linear behavior.

If the kind of object motion is not known beforehand, the motion reconstruction system has to rely on more general equations to describe the current object motion. One way is to use Auto-regressive Models (ARM) as they are described in the following sections. They are suitable for the description of most of the interesting object motions examined in this work (see Section 5.2 for examples). One important factor for the result of the prediction is the quality of the measured values. In general the output of the tracking process has sporadic errors. Thereby two types of errors can be distinguished:

- *Inexact measurements* can occur through the limited resolution of the camera or

[11] i.e. $\mathbf{r}(t) = (x(t), y(t), z(t)) = (v_0 t \cos\alpha, 0, v_0 t \sin\alpha - 1/2 g t^2)$

through mismatches of the assumed object contour with the real object contour. These in-certainties can be modeled through white Gaussian noise and can be compensated e.g. by low-pass filtering.

- *Outliers* occur through errors during the tracking e.g. through loosing track of the object. Outliers can lead to large deviations of the prediction. One way to handle outliers is by smoothing the measured data.

Since any kind of post-processing of the time series of the measured data tends to be time-consuming an efficient way to get stable prediction data was searched for. The chosen method which proved to be working satisfactory for the catching task is presented in Section 4.5.1.

Prediction Every three-dimensional object motion $\mathbf{s}(t)$ can be separated into its single motions along the Cartesian axes of a reference system:

$$\mathbf{s}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}$$

The future trend of the object motion can be calculated for every coordinate axis separately by a so-called *1D-Predictor*. Therefore, we will concentrate only on the prediction of a one-dimensional object motion for the rest of the section. The task of the prediction is to determine future values $\{x_{n+1}, x_{n+2}, \ldots\}$ out of a row of n measured values $\{x_1, \ldots, x_n\}$. For simplification of the prediction methods given below we assume that the sampling interval T_a between two consecutive measured values is equidistant. For predictions which cover more than one step into the future two prediction methods can in general be distinguished:

- *iterative prediction*
  For the iterative prediction the value of x_{n+2} is calculated from the known measured values and the predicted position x_{n+1}:
  $$x_{n+2} = F(x_1, \ldots, x_{n+1})$$
- *direct prediction*
  In contrast the value of x_{n+2} is only calculated from the measured values $x_1, \ldots, x_n$:
  $$x_{n+2} = F(x_1, \ldots, x_n)$$

Unfortunately, there is no general rule to determine which of the methods produces better results. By the *direct* prediction method inaccuracies can occur, since with only one calculation step values for a large time period are covered. Though this is not the case with the iterative method, it might rely on inaccurate former predictions. A combination of both methods could also be considered, but is not covered within this work. In the following only the iterative prediction is used for all mentioned algorithms. Further informations to direct prediction methods can be found in [SJ99].

3.3.1 Prediction with Auto-regressive Models

In Auto-regressive Models the output value of the system is determined by a linear combination of the l preceding values. An AR model of order l can be described by the following equation:

$$x_{n+1} = \sum_{j=0}^{l-1} a_j x_{n-j} + b \tag{3.49}$$

Often the term IIR-Filter (Infinite Impulse Response) is used for this kind of model, since the system produces in infinitely long response at the output as an reaction to single peaked stimulation. In general there exist three distinct possibilities for the time course of the output signal $x(t)$ [WG94]:

- The output signal converges against the value b.
- The output signal oscillates periodically.
- The output signal diverges.

To be able to use this model for the prediction of object movements, the measured object motion has to be describable by Equation 3.49. For this equation the parameters a_j and b have to be estimated from the former measured position values of the object motion.

3.3.1.1 Global AR Model (least square)

One way to calculate the parameters $\{a_0, \ldots, a_{l-1}, b\}$ is the minimization of the quadratic error. Thereby Equation 3.49 is applied to all measured values and the parameters are determined by minimizing the least squares error. For a model of order l, thereby $n-1$ equations can be set up. The error which has to be minimized is the mean quadratic deviation between the measured value x_j and the output value $\hat{x}_j$ of the model:

$$E = \frac{1}{n-l} \sum_{j=l+1}^{n} (x_j - \hat{x}_j)^2 \tag{3.50}$$

The system of equations containing $(n-1)$ equations has the following form:

$$\begin{pmatrix} x_{l+1} \\ x_{l+2} \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_l & x_{l-1} & \cdots & x_1 & 1 \\ x_{l+1} & x_l & \cdots & x_2 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_{n-1} & x_{n-2} & \cdots & x_{n-l} & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_{l-1} \\ b \end{pmatrix}$$

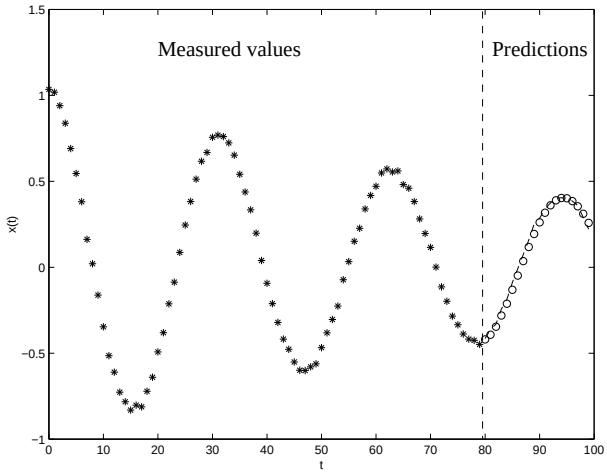


Figure 3.8: Prediction with a global AR model (least square)

$$\mathbf{x} = \mathbf{R}\mathbf{p} \tag{3.51}$$

The solution of the searched parameter vector $\vec{p}$ can be calculated by using the pseudo-inverse matrix $\mathbf{R}^{+}$[12]:

$$\mathbf{p} = \mathbf{R}^{+}\mathbf{x} \tag{3.52}$$

To avoid problems through the occurrence of singularities when calculating matrix $\mathbf{R}^{+}$, the inversion was done by using *Singular Value Decomposition (SVD)*. Having the calculated parameter vector $\mathbf{p}$, the estimated value for x_{n+1} can now be calculated by use of Equation 3.49. For further prediction steps this procedure is continued iteratively.

3.3.1.2 Local AR Model (maximum likelihood)

Contrary to the above described AR model, the maximum likelihood method tries to estimate the model parameters through the use of equations of motion. Therefore, the theory of the in [EG98] described method is shortly presented:

[12] using the pseudo-inverse matrix applies for the general case that matrix R is not square

Determination of the model For the calculation of an object position x_{n+1} out of the last measured values following AR model is used:

$$x_{n+1} = \sum_{j=1}^{p} \alpha_j x_{n-j+1} + e^x_{n+1} \tag{3.53}$$

with e^x_{n+1} being zero-mean white Gaussian noise. If the sampling steps of sensing the environment are small enough, one can assume that the acceleration $\ddot{x}(t)$ of the object is constant or is changing only slowly from one sampled value to the next. Therefore, the following applies for the acceleration:

$$\ddot{x}_{n+1} = \beta \ddot{x}_n + e^{\ddot{x}}_{n+1} \tag{3.54}$$

If one normalizes the sampling time Δt to 1, the velocity $\dot{x}_n$ and the acceleration $\ddot{x}_n$ can be expressed as follows:

$$\dot{x}_n = x_n - x_{n-1} \tag{3.55}$$

$$\ddot{x}_n = x_n - 2x_{n-1} + x_{n-2} \tag{3.56}$$

By substituting Equation 3.56 into Equation 3.54 one gets an AR model of order $l = 3$.

$$x_{n+1} = (2+\beta)x_n + (-1-2\beta)x_{n-1} + \beta x_{n-2} + e^{\ddot{x}}_{n+1} \tag{3.57}$$

In more compact notation:

$$x_{n+1} = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{pmatrix} \begin{pmatrix} x_n \\ x_{n-1} \\ x_{n-2} \end{pmatrix} \tag{3.58}$$

with

$$\begin{aligned} \alpha_1 &= 2+\beta \\ \alpha_2 &= -1-2\beta \\ \alpha_3 &= \beta \end{aligned} \tag{3.59}$$

Estimation of parameters To estimate the coefficients $\{\alpha_1, \ldots, \alpha_3\}$ from a given series of measurements $\{x_1, \ldots, x_n\}$, the Gaussian noise has to be minimized. For this estimation of the coefficients and of the variance of the noise σ_x^2 the maximum likelihood function is used:

$$l_c(\alpha_1, \alpha_2, \alpha_3, \sigma_x^2) = -\frac{n-3}{2}\ln(2\pi) - \frac{n-3}{2}\ln(\sigma_x^2) - \frac{1}{2\sigma_x^2}\sum_{k=4}^{n}\left(x_k - \sum_{j=1}^{3}\alpha_j x_{k-j}\right)^2 \tag{3.60}$$

Since all coefficients $\alpha_1, \ldots, \alpha_3$ only depend on the parameter β, the maximum likelihood function can be simplified:

$$l_c(\beta, \sigma_x^2) = -\frac{n-3}{2}\ln(2\pi) - \frac{n-3}{2}\ln(\sigma_x^2) - \frac{1}{2\sigma_x^2}\sum_{j=4}^{n}(\ddot{x}_j - \beta\ddot{x}_{j-1})^2 \tag{3.61}$$

To maximize the function l_c the partial derivatives are calculated and set equal to zero:

$$\frac{\delta l_c}{\delta\beta} = 0 \qquad \frac{\delta l_c}{\delta\sigma_x^2} = 0$$

As result one obtains:

$$\beta = \frac{\sum_{k=4}^{n}\ddot{x}_k\ddot{x}_{k-1}}{\sum_{k=4}^{n}\ddot{x}_{k-1}^2} \tag{3.62}$$

$$\sigma_x^2 = \frac{1}{n-3}\sum_{k=4}^{n}(\ddot{x}_k - \beta\ddot{x}_{k-1})^2 \tag{3.63}$$

The value σ^2 obtained by the above calculation rule is only an estimation of the true value. It can be shown that a better estimated value can be obtained by slight modifications of Equation 3.63. Thereto the sum of the squared distances has to be divided not by $n-3$ but by $n-4$ (see [Sch84]).

$$\sigma_x^2 = \frac{1}{n-4}\sum_{k=4}^{n}(\ddot{x}_k - \beta\ddot{x}_{k-1})^2 \tag{3.64}$$

By the system of Equation 3.59 now the searched coefficients α_i can be determined. Since for this estimation only the parameter β has to be determined, the calculation time is short compared to other methods.

Contrary to the AR model described in Section 3.3.1.1 this model tries not to find the global parameters of the object motion, but uses only the current (local) acceleration of the object for the prediction. Therefore, it is especially useful for long data sets to have the estimation of the parameters only local. For the calculation of β not all accelerations are summed up, but only the last 10 to 20 measured values are used. As an example for this parameter estimation out of the last 20 values see the following equation:

$$\beta = \frac{\sum_{k=n-19}^{n} \ddot{x}_k \ddot{x}_{k-1}}{\sum_{k=n-19}^{n} \ddot{x}_{k-1}^2} \tag{3.65}$$

3.3.2 Nearest Neighbor Predictions

Throughout scientific research, such as physics, biology or medicine, measured time-series are often a basis for characterizing a observed system and for predicting its future behavior. This is based on the fact that for every deterministic dynamic system it is possible to calculate future states of the system basing only on the knowledge of the current system state [WG94]. To achieve this a method has to be found via that the dynamics of the system can be reconstructed from former measurements of the object motion.

One method to do this is *Delay-Coordinate Embedding*. With this method it is possible to reconstruct linear as well as non-linear object motions and the method promises insights to the observed systems that traditional approaches (as the before described AR models) cannot provide [WG94].

Unfortunately, the method has two main drawbacks: the first is that it requires large data sets containing recurring states of the dynamic system to produce any predictions. This was not the case with most of the time series treated in this thesis. The second drawback is the unreliability of the results compared to traditional approaches [WG94].

Due to these reasons the treatment of NN predictions is reduced on a comparison of results of NN predictions as well as ARM predictions with artificial data series. This is shown in Section 5.2.

Nevertheless for the interested reader following books and articles are recommended: [WG94], [YLH98], [Sel00].

3.4 Hand-Target Interaction

In case of hand-target interaction the same differentiation shall be made as for the human example: *Interaction with a static target* (Sec. 3.4.1) vs. *Interaction with a moving target* (Sec. 3.4.2). Thereby the "state of the art" is presented as it can be found in the well known literature for robotics.

3.4.1 Interaction with a Static Target

For the interaction with a static target two cases can be distinguished: "positioning" (Sec. 3.4.1.1) and "reaching and grasping" (Sec. 3.4.1.2). Thereby positioning is *the* classical task addressed in the literature on visual servoing systems. Grasping objects can be seen as a special positioning task (reaching) with a subsequent interaction with the target object. But how to stably grasp an object is a research field of its own what was already addressed before (see Sec. 3.2.3).

3.4.1.1 Positioning

"The task in visual servoing is to use visual information to control the pose of the robot's end-effector relative to a target object or a set of target features"([HHC96]). Depending on the hand-eye configuration the task is further distinguished: given an *eye-in-hand configuration*, the camera is positioned relative to a target; given *stationary cameras*, the task is to move the end-effector or a tool held by it.

Since positioning is *not* the main interest within this thesis a short repetitive abstract of related work presented in [Hau99] shall be given in the following:

Chaumette et al. has addressed image-based camera positioning in the early nineties. First approaches [CRE91] used geometric features (mostly points) in conjunction with a coarsely calibrated pinhole camera model for 6 d.o.f. positioning. The feature configuration at the goal position was taught. This approach was extended to non-geometric features, using image motion of textured planes [CC97] and its temporal integration [CC98] as a control input. Problems like how to avoid kinematic limits or losing sight of the target were also addressed [MCB99, MCR96, MH98].

Allotta and Colombo [AC99] show that, for the task of camera positioning, the performance and robustness of image-based motion control can be improved by using a combined feed-forward/feedback strategy. Based on a coarsely calibrated para-perspective camera model and affinely deform-able active contours, the control signal is computed from the sum of a preplanned image-based "trajectory" and the current image-based error signal. The goal position is taught in beforehand.

For robotic assembly tasks Nelson et al. [NPK93] show in which way visual servoing can be useful. They discuss and implement strategies for parts mating supervised by a stationary camera and intelligent camera placement, based on a pinhole camera model with known intrinsic parameters.

Gangloff et al. [GdMA99] apply position-based eye-in-hand visual servoing to the task of following a 3D profile. The form of the profile is unknown, but marked with three parallel lines. Another example of position-based eye-in-hand positioning is the work of Martinet et al. [MDGD97]. Yoshimi and Allen [YA95] describe an image-based eye-in-

hand system specialized on a peg-in-hole task. The robot itself is calibrated, the Image Jacobian is determined from a sequence of known robot motions; features are the ellipsoid projections of the holes.

Three-dimensional reaching tasks, for example moving a spoon over a cup, are realized by Grosso et al. [GMOS96] using optical flow features extracted by a stationary stereo camera system. Only qualitative information about the hand-eye configuration is necessary. The task specification, i.e. the determination of the part to be servoed and of the goal configuration, could be called "hard-coded";

Hager [HD97] presents a modular system for realizing up to 6 d.o.f. positioning tasks with a stationary image-based stereo system. He mainly uses geometrical features like point and lines, but also projective in-variances [Hag95]. The perhaps most spectacular task achieved is the insertion of a disk into a disk drive, other examples include positioning a screwdriver on a screw.

Jägersand presents an integrated approach to uncalibrated hand-eye coordination [Jäg97b]. The main feature is that sensory information is used at the same time for motion control, via image-based visual servoing, and for the on-line estimation of the Image Jacobian [JFN96], without the need for special robot movements. The Jacobian is estimated locally, global convergence is assured by using an adaptive step size and by generating "way-points" on the visual trajectory [Jäg96]. Tasks are specified in visual space, by user interaction or via teaching [JN95], and then broken down into motion primitives [Jäg97a]. This allows for example to control only 3 d.o.f. in a longer reaching movement, and then to switch to 6 d.o.f. control for fine manipulation.

3.4.1.2 Reaching and Grasping

Grasping objects can be seen as a special positioning task with a subsequent interaction with the target object. How to stably grasp an object is a research field of its own; more information can be found e.g. in [BBY98]. Following this line of reasoning, Horaud et al. [HDE98] for example propose to apply an image-based visual servoing approach to the task of object grasping. The goal position, i.e. the position in which the robot is ready to close its gripper, is taught, with the corners of the (polyhedral) object and artificial markers on the gripper serving as features. The main argument against the teaching of goal positions, namely that they are tied to a certain camera, is answered very elegantly by introducing a projective framework for "translating" goal positions measured with one camera setup to a different one. For the servoing control law, they extended the approach of Chaumette et al. to the stationary camera configuration. The underlying pinhole camera model is coarsely calibrated based on known robot motions.

However, approaches based on teaching grasping positions definitely do not scale well in the framework of autonomous service robots. Here, the ultimate goal would be to enable the robot to autonomously extract suitable grasp positions. In the case of known

(and recognized) objects, grasp positions could be retrieved from a database. Blessing et al. [BLZ96] for example present a position-based eye-in-hand look-then-move system; grasping positions and collision-free grasping motions are calculated from a CAD model. Hanebeck et al. [HFS97] use a position-based look-then-move strategy on their service robot ROMAN, which is equipped with a monocular camera system on a tilt-axis. Namiki et al. [NNII99] use special image processing hardware in order to obtain visual information with a rate of $1ms$. In their experiments, a (known) object is first tracked and then grasped using repeated pose estimations from a stationary camera.

The work-group of Nagel has the long-term goal of using visual servoing for the disassembly of used cars. Based on CAD-models of the parts to disassemble and on a calibrated[13] hand-eye system with an independent stereo camera system mounted on a second manipulator, they use a simple position-based visual servoing control law [TSSN97]. For pose estimation and tracking, an iterative extended Kalman Filter is employed [TSHN97]; the features (edge elements) are extracted using specialized hardware. In [KTNG98], a method for automatic camera placement is described.

In the case of unknown objects, an additional "degree of difficulty" is introduced, i.e. the determination of suitable grasping points from visual information. Research in this area mostly focuses on the vision problem and regards the motion control part as given. Examples for this approach are the systems of Taylor et al. [TBC94] or Kamon et al. [KFE96]; they will be described in more detail in Sec. 4.4.1.

Only few groups work on both the vision and the motion control problem. Hollinghurst and Cipolla [CH97] search for planar surfaces which they treat as candidate grasping positions for polyhedral objects. Motion control is realized using position-based visual servoing based on an affine model of the (stationary) stereo camera system which is coarsely calibrated observing known robot movements. Gripper and object are tracked using active contours. As an additional feature, the object to grasp can be "selected" by pointing at it.

Sanz et al. [SdPIR98] address the problem of finding grasping positions on (quasi-planar) free-form objects (discussed in more detail in Sec. 4.4.1). Such grasping positions are the input for an image-based eye-in-hand visual servoing system. The approaching phase is divided into two parts. The first serves to bring the manipulator (and the camera) just above the object to grasp; a set of such approach positions is taught off-line and stored in a look-up-table. In the second phase, a typical visual servoing control law is executed, with the Image Jacobian being learned from known robot motions.

[13] How to adapt the kinematic model of the manipulator from visual measurements is described in [RTHN97].

3.4.2 Interaction with a Moving Target

For the interaction with a moving target again two cases can be distinguished: "tracking" (Sec. 3.4.2.1) and "catching and hitting" (Sec. 3.4.2.2).

3.4.2.1 Tracking

In a tracking task, the manipulator starts out in the certain position relative to the target and is to follow target motion in order to keep this relative position constant; it is therefore similar to a positioning task, with the degree of difficulty depending on the complexity and velocity of the target motion.

Papanikolopoulos et al. [PNK95] describe an eye-in-hand system for tracking an object moving in 3D. They use specialized image processing hardware to extract textured patches and to track them via SSD correlation; how to select "good" features is described in [PS95], how to detect the object in [RP95]. The Image Jacobian is derived from the pinhole camera model; the necessary information about depth is estimated using controlled robot motions [SBP97], other parameters are adapted online.

Hashimoto and Kimura [HK93] test their optimal control schemes in an image-based eye-in-hand tracking task. They track simple planar movements. Robustness against sensor latency and delays is assured by using observer for the target and robot motion; in [HK95] a full, nonlinear model of the robot dynamics is used, in [HN99] a linearized one.

Wilson et al. [Wil93] also use an observer, a Kalman Filter, put in a position-based eye-in-hand setup. In [WHB96], they extend this approach to 3D motion.

Bensalah and Chaumette [BC95] also use a Kalman Filter to estimate target motion; additionally, they use the generalized likelihood ratio algorithm to compensate for abrupt motion changes.

Piepmeier et al. [PML98] compare the performance of different observers for the tracking of linear and circular target motions. The experimental setup consists of a two-link manipulator servoed by a stationary stereo camera system. Image processing is held simple by using one point feature on manipulator and target. This approach was extended for more complex planar motions by integrating the motion observation into the estimation of the Image Jacobian [PML99].

Oh and Allen [OA99] apply their approach of partitioned control to monitoring tasks in an assembly work-cell. A camera mounted on a pan-tilt head which itself is mounted on a 3 d.o.f. Cartesian gantry robot tracks for example people moving around the work-cell; the redundant degrees of freedom allow a robust tracking in a huge work space. The region-based SSD tracking is implemented in software, using Hager's XVision library, the targets to track are specified manually. The robot system and the pinhole camera model are coarsely calibrated.

3.4.2.2 Catching and Hitting

Interaction with a moving object, e.g. catching or hitting it, is perhaps the most difficult task for a hand-eye system. Most successful systems presented in literature use precisely calibrated, stationary stereo camera systems and image-processing hardware together with a simplified visual environment: e.g. the target can be distinguished from the background because of its color, its features are the centroid coordinates in the two images. Visual information is used continuously to reconstruct the 3D motion, but still in a feedforward structure concerning motion control.

One of the first works dealing with "dynamic grasping" is [And86]. An approach is described there how a ping-pong ball rolling down a slope can be catched. Thereby a simple motion prediction is performed: the trajectory of the ball is fitted to a straight line. But during the "catch" only the position, not the velocity or acceleration, of the end-effector and the target object is matched.

Some time later Lin et al. [LZP89] propose a two-step heuristic approach for catching. In the first step the end-effector is moved towards the target object in the shortest possible time, in the second step end-effector trajectory and target trajectory are matched.

Houshangi [Hou90] uses Auto-regressive time-discrete models to predict the target trajectory and to compensate for image processing time delays. The used trajectory planning module automatically reacts on changes of the target trajectory. Intermediate targets are calculated from the current end-effector position and the predicted target positions. But the prediction period is limited to one sampling interval. Additionally, the target object has to be inside the workspace of the manipulator at every time.

Gosselin [GCL93] uses state estimators to predict the target trajectory, but limits his considerations to free falling objects. The nearest point to the robot's base is selected as the catching location.

Zhang et al. [ZB94] adapts a geometric controller, originally developed for robot juggling, to determine a desired target pose. But thereby no models of object motion or position predictions are used.

Allen et al. [ATYM93b, ATYM93a] developed a robotic system that could grasp a toy train moving in a plane. The train's position is estimated from (hardware-supported) measurements of optic flow with a stationary, calibrated stereo system. Using a non-linear filtering and prediction, the robot tracks the train and finally grasps it. Thereby the system is dependent of real-time image processing and sensitive to sensor failure.

The train scenario was also addressed by Burdet and Luthiger [BL96]. Instead of grasping the train after tracking, the robot lets an object drop into a wagon. In this approach, only a single overhead camera is used. Due to the lack of image processing hardware they simplify vision by tracking a light bulb mounted on the train. Visuo-motor transformations and reaction times are learned during training motions. For target tracking and position

prediction a Kalman Filter is used, based on a third-order motion model. The movements themselves are optimized in order to reach the target smoothly and fast; they can be smoothly adapted to new information about the target's position.

One of the earliest "ball playing" robots was Andersson's ping-pong player [And89]. Bishop and Spong present a robot playing air hockey [BS98]; to solve this 2D problem, they use a calibrated grey-scale camera mounted over the hockey table and a 3 d.o.f. planar robot arm. Real-time image processing is assured with the help of a DataCube; trajectories are estimated based on a 2D least-squares fit.

Rizzi and Koditschek [RK93] presented a sophisticated system architecture for robotic juggling, including attentional control[14]. Nakai et al. [N+98] developed a robotic volleyball player. The colored balls are continuously fixated by a stereo camera head, the estimated positions are fit into a 3rd order polynomial equation of the trajectory, from which a suitable hitting point in space and time is computed.

Hong and Slotine [HS95, Hon95] present a system with a 4 d.o.f. head and a 7 d.o.f. cable-driven manipulator which can catch (and throw) tossed balls and even paper airplanes. Image processing is facilitated by using colored markers on the objects and supported by specialized hardware. The object is fixated continuously, its path is estimated and a suitable interception trajectory for the robot is calculated based on calibrated kinematic models of the hand-eye system and learned dynamic models.

An exception to the rule of using visual feed-forward control for catching objects resulted from the cooperation of the labs of Hirzinger [HBDH93] and Dickmanns [FDD94] in the framework of the D2 space shuttle mission: One of the experiments saw ROTEX, the first European robot in space, catch a (not too fast) free-flying polyhedron. The success was based on the concept of "shared control": At first, a human operator initialized the (6 d.o.f.) pose of the target by superimposing a CAD model using a space mouse and moved the robot into a suitable start position. Then, the system switched to the autonomous mode. Based on feedback from a miniature camera in the hand, the robot first moved to reduce the "errors" in the directions perpendicular to the line of sight, then moved towards the object to intercept it. The almost unbelievable fact is that image processing was done *on earth*. The approximately 6s delay were compensated by sophisticated prediction schemes based on precisely calibrated models of the robot system and of the motion of the object.

Okhotsimsky [OPB+98] shows that by using exact dynamical models the grasp position and time can be determined even for very complex object motions. But the determination of the models is very complex and shown only exemplarily for two cases.

Nagahama et al. [NHNT00] describes a visual servoing system that is able to pick up a moving object from a table surface. They use a model based predictor to be able to close the visual servoing loop in case that the target object is lost temporarily.

[14] In later work [RK96], they proposed a robust visual state estimator and pointed out it's duality to the task of visual servoing.

Finally, Frese et al. [FHH01] developed a robotic ball catcher using of-the-shelf components for visual tracking. The thrown ball is observed by a large baseline stereo camera system, comparing each image to a slowly adapting reference image. Tracking and prediction of target position is performed by an Extended Kalman Filter (EKF), taking into account the air drag. The system is limited to thrown ball trajectories and needs a motionless background for successful tracking.

3.5 Summary

In this chapter methods and models needed for every calibrated hand-eye system were described, i.e. manipulator model, head model, camera model, object model as well as motion models.

Different methods for visual tracking of an object in motion were presented. Thereby tracking methods using different sensor modalities (as e.g. form, color and motion) were introduced and the according examples from the literature were given. Most of the tracking methods were "low-level" tracking methods, i.e. those methods are fast and robust but provide little information apart from objects position and size. One "high-level" tracking method, the Condensation algorithm, taking into account apriori known object information, in this case the objects shape, was presented.

Fusion and Integration of sensor information was shortly discussed. Thereby one algorithm capable of integrating sensor information from different sensors was presented more thoroughly. This algorithm, called ICondensation, is an extension to the aforementioned Condensation algorithm.

Relevant notations and methods from the literature concerning the problem of grasp determination were presented. It was shown that many works base their determination on the analysis of the geometric properties (e.g. shape) of the object. In the case of moving objects the use of deform-able contours to represent its shape has proved useful, since they can also be used for tracking the object along a sequence of images.

To reconstruct and predict the motion of an object two different methods have been shown: the prediction with auto-regressive models and the prediction with nearest neighbor methods. Thereby autoregressive models are suitable to predict linear motions whereas nearest neighbor methods can also be applied to highly non-linear motions, but with the drawback that large data sets are needed.

The hand-target interaction for the two cases, interaction with a static target and interaction with a dynamic target, was presented thoroughly. Thereby many examples of well working systems as they are described in the literature were reviewed.

3.6 Discussion

Again, like in the preceding chapter, the question: What can be learned from the gained knowledge for the goal to control a robot for a catching task? Certainly there are many interesting aspects that on the one hand imply to be "copied as is" for fulfilling the intended goal, on the other hand some aspects are interesting for further evaluation and extensions. The goal would be to transfer this knowledge and fuse it with the knowledge from the preceding chapter Chap. 2 to achieve better results of the algorithms. This procedure will be applied for different aspects, as e.g. image preprocessing, sensor fusion and integration as well as for object motion prediction in the next chapter (Chap. 4).

Chapter 4

Hand-Eye System and Interaction with a Moving Target

In this chapter all algorithms for the hand-eye system are developed. For the readers orientation the system architecture is given again in Figure 4.1. The single blocks given in the architecture can be put into relation with the content of the successive sections. The architecture is orientated on the assumed human behavior in a catching task. There are five main blocks: the model knowledge (Sec. 4.1), the sensory system (Sec. 4.2 and Sec. 4.3), the motion reconstruction and prediction system (Sec. 4.5), the actory system (Sec. 4.6) and the state automaton (Sec. 4.8). Each block is subdivided into different modules: the sensory system distinguishes between form (Sec. 4.2.1), color (Sec. 4.2.2) and motion processing (Sec. 4.2.3) for tracking and the fusion (Sec. 4.3.1) and integration (Sec. 4.3.2) of this information. Additionally, grasps can be determined and tracked (Sec. 4.4). The motion r+p system covers the prediction of future object positions (Sec. 4.5.1) from the tracked object positions. From these predicted positions appropriate interaction points and intermediate targets (Sec. 4.7) are derived. Finally, the actory system generates trajectories (for position (Sec. 4.6.2.1) and orientation (Sec. 4.6.2.2)) of the robot. Between the different system modules exist connections. Along these connections information is passed from module to module.

In a catching task the different blocks have to be activated dynamically. For this activation the state automaton is responsible which is explained in Sec. 4.8.2.

4.1 Internal Models

As could be seen from the system architecture and descriptions in Sec. 3.1 internal models are necessary to describe the hand-eye system. Within these sections the specific models used for the hand-eye system MinERVA will be described, i.e. the manipulator model,

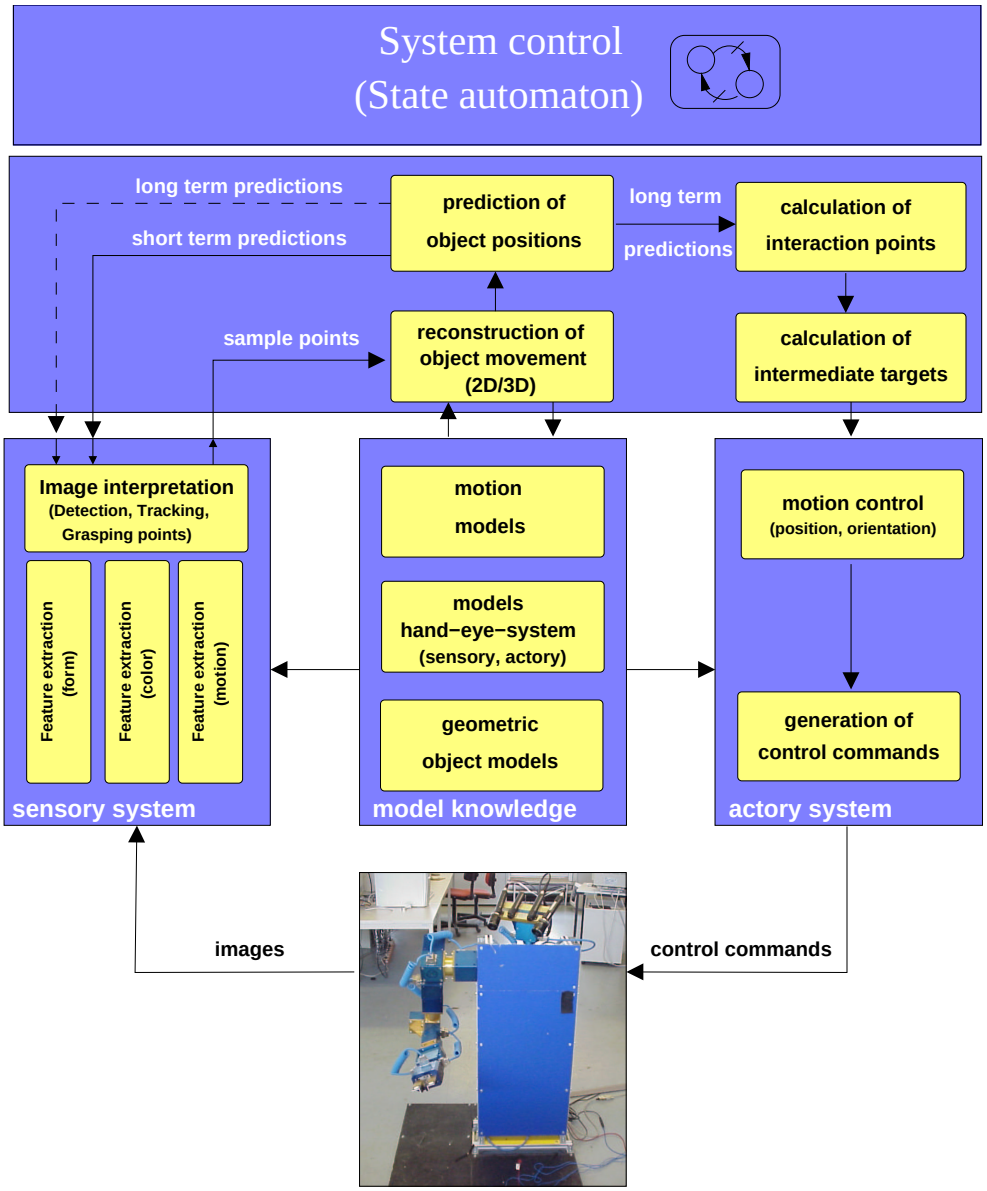


Figure 4.1: System Architecture

the camera and head model, the object models as well as object motion models.

Models of the Hand-Eye System

Model of the Manipulator

In general all relationships of Sec. 3.1.1 count also for the manipulator of our robot MINERVA. Therefore to describe the manipulator, in the following only those adaptations are

described that are necessary to model our specific robot. These adaptions concern (a) the Denavit-Hartenberg parameters, (b) the coordinate systems, and (c) the determination of the end-effector orientation.

Denavit-Hartenberg Transformation As mentioned before it is common in robotics to use Denavit-Hartenberg parameters for the spatial description of a manipulator. How they can be determined was described in Sec. 3.1.1.

Nevertheless, to describe the transformations from a frame $\mathcal{F}_i$ to frame $\mathcal{F}_{i-1}$ with Denavit-Hartenberg parameters ϑ_i, α_i, d_i and a_i different possibilities exist. With MINERVA following combination and order is used:

1. Rotation around the z-axis of $\mathcal{F}_{i-1}$ by angle ϑ_i.
2. Rotation around the x-axis of the resulting coordinate system by angle α_i.
3. The translation vector is thereby the linear combination of the x-base vector of $\mathcal{F}_i$ scaled by a_i and the z-base vector of $\mathcal{F}_{i-1}$ scaled by d_i.

The transformation matrix thereby results to:

$$
{}^{i-1}_{i}\mathbf{T} = \begin{bmatrix} \cos\vartheta_i & -\sin\vartheta_i\cos\alpha_i & \sin\vartheta_i\sin\alpha_i & a_i\cos\vartheta_i \\ \sin\vartheta_i & \cos\vartheta_i\cos\alpha_i & -\cos\vartheta_i\sin\alpha_i & a_i\sin\vartheta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}
$$

Coordinate Systems of MINERVA By use of Equation 4.1 the single transformations for the joints of MINERVA can be specified. Thereby the Denavit-Hartenberg transformation matrices are specified for the case, that the manipulator, i.e. the arm, is fully stretched. This is shown in Figure 4.2. For the description of the manipulator it is assumed that the origin of the frames $\{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2\}$, $\{\mathcal{F}_3, \mathcal{F}_4\}$ and $\{\mathcal{F}_5, \mathcal{F}_6\}$ is the same at each case (i.e. $d_i = 0$).

By additional offsets in the angles ϑ_i the matrices are adapted in a way that the configuration of the manipulator refers to the wished base position (all angles q_i=0). The configuration with regard to q_i is shown in Figure 4.3.

The Denavit-Hartenberg parameters of MINERVA are shown in Table 4.1. s is hereby the length of the "upper arm", t the length of the "lower arm" and u the distance of the "wrist" to the end-effector.

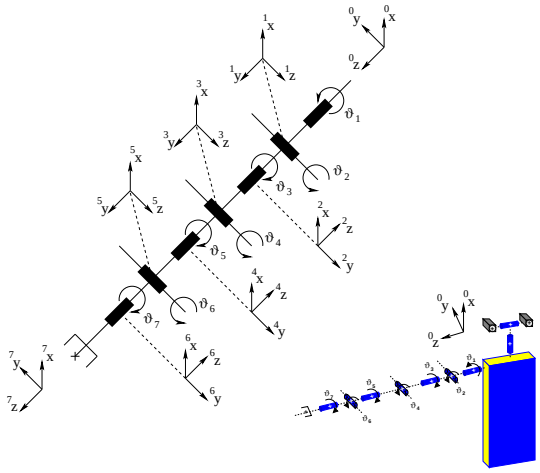


Figure 4.2: Coordinate systems of MINERVA with joint angles ϑ_i

Insertion of these parameters results, after short calculations, in the following transformation matrices for each joint:

$$ {}^{0}_{1}\mathbf{T} = \begin{bmatrix} \cos q_1 & 0 & \sin q_1 & 0 \\ \sin q_1 & 0 & -\cos q_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2} $$

$$ {}^{1}_{2}\mathbf{T} = \begin{bmatrix} -\sin q_2 & 0 & \cos q_2 & 0 \\ \cos q_2 & 0 & \sin q_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3} $$

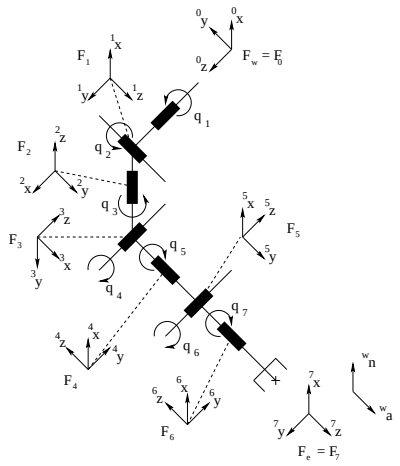


Figure 4.3: Coordinate systems of MINERVA with joint angles q_i in base configuration ($\mathbf{q} = \mathbf{0}$) and definition of approach- and normal-vector

$$
{}^{2}_{3}\mathbf{T} = \begin{bmatrix} -\sin q_3 & 0 & -\cos q_3 & 0 \\ \cos q_3 & 0 & -\sin q_3 & 0 \\ 0 & -1 & 0 & s \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}
$$

$$
{}^{3}_{4}\mathbf{T} = \begin{bmatrix} \sin q_4 & 0 & -\cos q_4 & 0 \\ -\cos q_4 & 0 & -\sin q_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}
$$

$$
{}^{4}_{5}\mathbf{T} = \begin{bmatrix} \cos q_5 & 0 & -\sin q_5 & 0 \\ \sin q_5 & 0 & \cos q_5 & 0 \\ 0 & -1 & 0 & t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}
$$

Table 4.1: Denavit-Hartenberg parameters for MinERVA

i	α_i [rad]	d_i [m]	a_i [m]	ϑ_i [rad]
1	$+\frac{\pi}{2}$	0	0	q_1
2	$+\frac{\pi}{2}$	0	0	$q_2 + \frac{\pi}{2}$
3	$-\frac{\pi}{2}$	$s = -0.37$	0	$q_3 + \frac{\pi}{2}$
4	$+\frac{\pi}{2}$	0	0	$q_4 - \frac{\pi}{2}$
5	$-\frac{\pi}{2}$	$t = -0.31$	0	q_5
6	$+\frac{\pi}{2}$	0	0	q_6
7	$-\pi$	$u = -0.24$	0	q_7

$$ {}^{5}_{6}\mathbf{T} = \begin{bmatrix} \cos q_6 & 0 & \sin q_6 & 0 \\ \sin q_6 & 0 & -\cos q_6 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7} $$

$$ {}^{6}_{7}\mathbf{T} = \begin{bmatrix} \cos q_7 & \sin q_7 & 0 & 0 \\ \sin q_7 & -\cos q_7 & 0 & 0 \\ 0 & 0 & -1 & u \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.8} $$

Calculation of End-effector Orientation For MinERVA the vectors ${}^{w}\mathbf{a}$ and ${}^{w}\mathbf{n}$ are defined as shown in Figure 4.3. Here approach and normal vector can straightly be read from the rotation matrix. ${}^{w}\mathbf{a}$ is equal to the z-base vector ${}^{w}_{e}\mathbf{b}_z$ of $\mathcal{F}_e$, and ${}^{w}\mathbf{n}$ is equal to the x-base vector ${}^{w}_{e}\mathbf{b}_x$ of $\mathcal{F}_e$:

$$ {}^{w}\mathbf{a} = {}^{w}_{e}\mathbf{b}_z, \qquad {}^{w}\mathbf{n} = {}^{w}_{e}\mathbf{b}_x \tag{4.9} $$

Vice versa the rotation matrix ${}^{w}_{e}\mathbf{R}$ can be calculated from this selection of ${}^{w}\mathbf{a}$ and ${}^{w}\mathbf{n}$. ${}^{w}\mathbf{a}$ and ${}^{w}\mathbf{n}$ can be seen as "input vectors", which, according to Equation 3.14, not necessarily need to be perpendicular. As a result one gets the orthonormal rotation matrix

$$ \begin{aligned} {}^{w}_{e}\mathbf{R} &= \left[\; {}^{w}_{e}\mathbf{b}_x \quad {}^{w}_{e}\mathbf{b}_y \quad {}^{w}_{e}\mathbf{b}_z \; \right] \\ &= \left[\; ({}^{w}\mathbf{a} \times {}^{w}\mathbf{n}) \times {}^{w}\mathbf{a}; \quad {}^{w}\mathbf{a} \times {}^{w}\mathbf{n}; \quad {}^{w}\mathbf{a} \; \right] \end{aligned} \tag{4.10} $$

For the calculation of Equation 4.10 also unnormed approach and normal vectors can be used, if they are normed before insertion.

Model of the Head and Camera

The model of the head and the cameras are necessary here to transform any point projected on the camera planes to the head base frame to obtain its 3D position relative to the head base frame. The used models here are the *Pin-hole camera model* and a two joint "manipulator" for the pan-tilt head. The according equations describing both models for MINERVA can be found in the appendix.

Transformation from Head to Arm The transformation matrix from head to arm coordinates can also be found in the appendix. The output obtained from this transformation is a 3D coordinate in the manipulator base coordinate system. Usually target positions, determined via vision, are transformed hereby from head centered coordinates to arm centered coordinates.

Models of the Object to be Grasped

Within this thesis (and within the context of tracking) appearance-based as well as geometric representations of the object are used. The global feature representing the object is simply *color* represented as a subspace of a chosen color space. As a geometric representation "deform-able templates" or "Snakes" [BI98], i.e. parametric shape models with relatively few degrees of freedom are used. Those shape models are represented by parametric spline curves, i.e. B-spline curves, as it is common in computer graphics.

One problem occurring with B-spline curves is how to get the initial B-spline curve describing the outline of an object. Thereby an automatic procedure for this determination is useful. A procedure developed within this thesis is described in the following section.

4.1.0.3 Automatic Initialization of B-spline Contour Models

A complete method for automated 2D reconstruction of object silhouettes by B-splines was developed in [Gla02]. The most important steps for this automated mathematical approximation of the silhouette are: (a) the *initialization* of the B-spline, (b) the *correspondence search* between the B-spline and silhouette for detecting deviation and (c) the *improvement* of the generated initial B-spline to optimize it (fitting).

For the automated initialization of the first B-spline a method was developed that uses the shape information of the object's outline to determine the initial B-spline. Due to this method the first B-spline already provided a good approximation of the object's silhouette. Using this initialization method is essential and subsequent methods for controlling and fitting (improvement) the initial B-spline are represented as relatively straight-forward methods to complete the automated 2D representation.

Initialization. The most important step of the automated B-spline determination is the initial B-spline. A method to transform the shape information of an object to the first B-spline representation is presented in the following. As in [GM99] the shape information of outlines is quantified[1]. The method described here translates the essential shape information into a first spline and is targeted for approximating splines needed for above mentioned *Snakes* or as a initial form template for the CONDENSATION algorithm (see Section 3.2.1.1). In the following it can be seen that the object's shape information already has to be included in the initial B-spline to let the later introduced improvement method be successful.

This initialization method uses the curvature function as a criterion for the shape information of an outline. There exist many methods to detect a discrete curvature. One way to calculate a discrete curvature with low calculational effort is to determine the so called *k-curvature* [Val96, SdPIR98]. Thereby the curvature value at a point p on the object's contour is the angle between the vectors v_1 and v_2 shown in Figure 4.4, where k is the number of pixels covered in each direction on the contour. The object's curvature can be determined in every point of the outline with the k-curvature function:

$$\text{k-curvature} = angle = \frac{v_1\vec{(k)} * v_2\vec{(k)}}{|\vec{v_1}| * |\vec{v_2}|} \tag{4.11}$$

whereby $\vec{v}$ is the vector from pixel (index p) to pixel (index p$\pm k$) and $|\vec{v}|$ is the norm of the vector v.

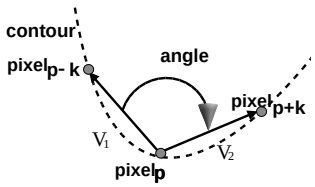


Figure 4.4: Calculation of the curvature

The curvature function parameter *vector length* is directly proportional to the object's size and depends indirectly proportional on the degree of exactness which is desired for the initial B-spline. The parameter k[2] can be used to smooth a noisy curvature function (large

[1] Another initialization method for the automated determination of 'control points' (control point has slightly different properties) is proposed in [PP96].

[2] The size of k is about one tenth of the object size. One tenth of the length of the smaller side of the surrounding rectangle proved to be a suitable value for k.

k) or to improve the *exactness* E of the first set of initialization points (small k).

$$\text{step size} = k = \frac{\text{contour length [in pixels]}}{\text{E [in pixels]}} \tag{4.12}$$

The first and second derivative of the curvature function are calculated to determine the minima (convexities), maxima (concavities) and inflexions (zero curvature). The criteria to decide if a detected extremum is a characteristic one, are:

$$\begin{aligned} &\text{Criteria 1: } [|\text{curvature value}| > \text{threshold} \ \wedge \quad \mathit{first\ derivative} = 0] \\ &\qquad \vee \\ &\text{Criteria 2: } [\text{curvature value} = 0 \quad \wedge \quad \mathit{first\ derivative} = \text{extremum }] \end{aligned}$$

where *threshold* is used to limit the possible values for a solution. This threshold could be set to zero if the contour consists of low noise and the calculated values by the curvature function and its derivatives could be determined without any calculation errors (e.g. rounding, quantization (=discretisation) errors).

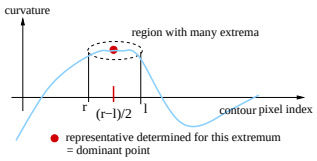


Figure 4.5: Representative of an accumulation of extrema

One representative has to be chosen for all extrema which are determined within a region which represents the same curve of the contour (Figure 4.5). The representative is the middle over all extrema related to the spatial distribution (index of the contour pixels) of the extrema on the curve. This representative is called a *dominant point*.

Figure 4.7, left side, shows an object outline with the determined dominant points and Figure 4.6 shows the graphics of the curvature function and its derivatives.

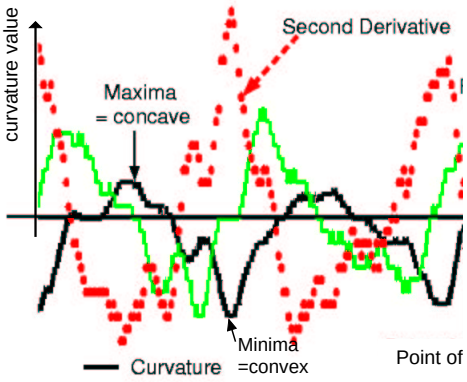


Figure 4.6: The calculated curvature function and its first and second derivatives

The central idea is to transform the shape information of the object provided by the curvature function and its dominant points into the first B-spline representation to obtain a similar-shaped B-spline. Thus the next step is to transform the information of the dominant points into a first set of control points, the *initialization points*, to determine a B-spline. The initialization points are determined using a tangential approximation of the outline. The tangents are placed at the dominant points. The points of intersection of each tangent with the neighboring tangents are the first control points to initialize the B-spline. These control points are the vertices of the so-called control polygon, an approximation to the B-spline.

Additionally dominant points are pre-selected because it is more efficient to use not all dominant points for tangent placement. In general the following combinations of dominant points exist: (1) a point with zero curvature is neighbored by a convexity or concavity. (2) more convexities or concavities are consecutive. This is the case if different curvature radia describe a complex curvature. The conclusion is that a convexity never can follow a concavity without a zero in between, and that two subsequent zeros are impossible. The best approximated B-spline to the outline is computed if the dominant points are selected as follows: if the shape is (I) a convexity all points representing a convexity with different radia as well as the two zero dominant points (points of inflexion) which edge the consecutive convexities are taken. In contrary (II) a concavity is represented in two ways depending on the number of consecutive concavities: (1) if there is only one concavity, it is skipped and the neighboring zeros are sufficient to approximate the shape. (2) if more than one concavity is in row all concavity points are used as well as the edging zeros. In

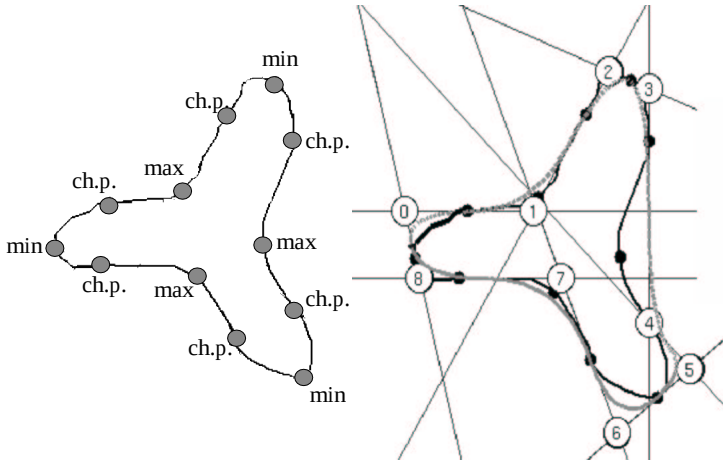


Figure 4.7: The dominant points *convexities, concavities and inflexions* and the resulting control points

Figure 4.7 right side the initialization by a control polygon formed by tangents is illustrated (at simple concavities no tangents are set).

Correspondence and Deviation Detection Method. To measure the accuracy of a B-spline approximation the B-spline is assigned to the real contour. Once a B-spline has been calculated using a given set of control points, the relation between the B-spline and the real contour is determined. This is done by searching for the points on the contour which relate to the points on the span of the B-spline.

According to this relation of the two curves the deviation of the B-spline approximation can be determined and evaluated. The distance between the two curves is the deviation of the approximated B-spline which is called **error**. This error will be reduced to a desired minimum during the improvement procedure (see paragraph below). To determine a relation between the real contour and the B-spline the normals of the B-spline are used.

The intersection of the normal in the point p_n on the B-spline with the contour is the corresponding point p'_n. If a corresponding point is found the B-spline can be assigned to the real contour. Every span of the B-spline is assigned separately to the real contour. Therefore the corresponding point for the initial and the end point of the span is determined. The assignment has to be unique and so on the one hand every point on the B-spline has its correspondent point on the real contour and on the other hand no part of the real contour is assigned twice. This is a good instrument to control the approximation. If it is not possible to assign a certain part of the B-spline a false constellation of control points is detected. The analysis of this false constellation leads to a change of the control points to adapt the B-spline and correct it.

It can happen for example that a false set of control points is determined because of two inaccurate tangents (Figure 4.8). The B-spline cannot be assigned correctly. An heuristic to analyze the fault could be e.g. a decision which control point is deleted. As three control points determine a span the following possibilities exist: (1) the beginning of a span can not be assigned or the two assignment normals intersect with each other before crossing the contour the second control point is deleted (Figure 4.8(b)). Or (2) the end of a span cannot be assigned, the third control point is deleted. Other possibilities to handle a false control point could be considered. To ensure the success of the improvement procedure the B-spline has to be assigned completely and uniquely to the real contour.

The measure of quality of an approximation is the distance between two corresponding points (**error**). The error between two corresponding points is defined as:

Error norm Euclidean distance between point p_n on the spline and the intersection point p'_n on the real contour (where point p'_n is the point where the normal of the spline at point p_n intersects with the real contour).

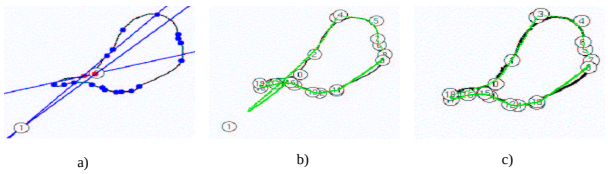


Figure 4.8: Compensation of a wrong control point setting. (a) Construction of a bad control point, (b) resulting B-spline, (c) after elimination of the bad control point

Improvement Method. After initialization of the first B-spline, its assignment to the real contour and the detection of the deviation (error) the B-spline is now improved. The fitting of the B-spline bases on a refinement of the spline. If the error norm of the correspondence procedure (see paragraph above) detects a higher inaccuracy than the desired accuracy threshold according to the error norm, the B-spline approximation has to be improved. This improvement is done by inserting new control points (and therefore new

knots) to refine and fit the spline, which implies a higher number of control points used for the improved B-spline. To improve the B-spline span-wise a knot is inserted into each span which needs to be improved. The knot is inserted at the position called *break location* which is the point of maximum error referring to the absolute value of the error. This is the point of a span where the deviation of the B-spline from the real contour is maximal. This method has been proofed to work more adequate (faster convergence, good accuracy) in combination with an accurate initialization method, than more complex methods as the PERM method introduced by [CC99]. This comparison is evaluated and explained in more detail in [Gla02].

Once new knots have been determined, a new set of control points is constructed and a new spline is calculated (Figure 4.9). The construction of the new control points is similar to the construction of the initialization points. In spans which are judged to be accurate enough, knots and control points are retained. If a span is improved, a new knot is inserted at the *break location*. The span, now consisting of the two old knots and one new knot, has to be adapted to this change. Two new control points are determined instead of one old one (middle control point referring to the three control points influencing a span). Thus all three knots are projected to the contour (assignment) and the tangents at these points provide two intersections. These intersection points are the new control points.

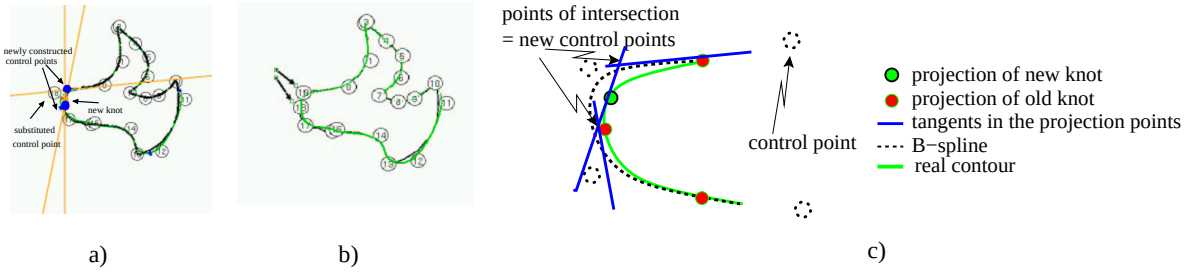


Figure 4.9: Improvement of the B-spline by inserting new control points. (a) construction of the new control points, (b) resulting improved B-spline, (c) details of the new control point construction

If the error between spline and contour is still too large, the process is repeated to improve the B-spline further.

Results. The automatically produced B-spline is compared to an optimal B-spline to show the quality. In this context optimal signifies that the B-spline is determined by its unambiguous mathematical description, with possible control points randomly set, and with the constraint that the final B-spline is a closed curve with no intersections. The special shape of the optimal B-spline is determined using an independent B-spline calculation program where control points are randomly set by the user. As before only closed B-splines of order d = 3 are taken into account. This optimal B-spline is the reference object to fit the generated B-spline.

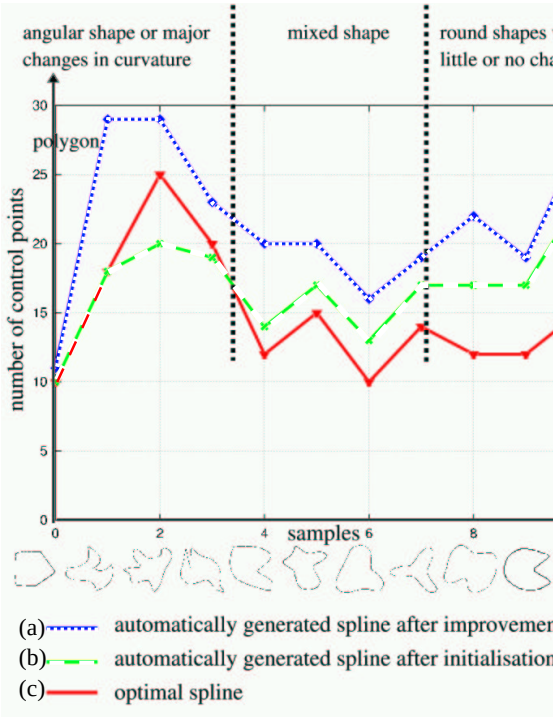


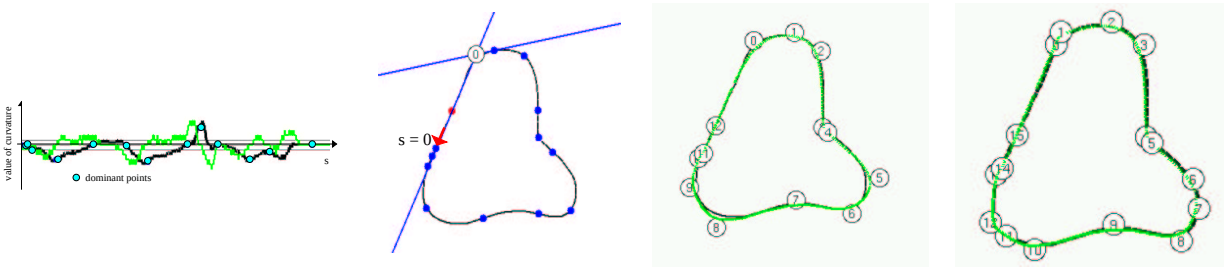
Figure 4.10: Number of control points compared for different samples according to the type of shape for the three cases optimal, initialised and improved B-spline

The automatically generated B-spline is compared to the optimal B-spline with regard to the number of (1) control points, (2) iterations and (3) the final deviation of the B-spline from the optimal sample. The initialization and the improvement methods are examined separately as two parts of the whole automated procedure. It should also be mentioned that the automated B-spline generation procedure is designed to produce a B-spline with a desired error (the final deviation) and *not* to obtain a B-spline with a minimum number of control points. Most times the first automatically generated B-spline determined by the initialization method fits the optimal B-spline with good accuracy.

In Figure 4.10 the different samples are arranged according to the objects' shape. The difference in the number of control points between the optimal spline in Figure 4.10(c) and the initialization spline in Figure 4.10(b) is shown for three different types of shapes: angular (or major changes in curvature), smoothly rounded (with little change in curvature) and types with a mixture of both. For angular splines the difference between the generated and the reference angular spline is small. Corners are very significant shape points and therefore the information transformed from the curvature to the first set of control points matches the shape with a high degree of accuracy. In contrast to this, round shapes are imitated well by the first set of control points but with an unduly high number of control points. The reason for this high number is that if there is little or no change in curvature (the worst example is the circle) almost no shape information can be transmitted by the

curvature function. To determine a spline it is therefore necessary to have information about the shape regardless if the curvature changes or not.

The worst case, a circle, has a theoretical curvature with a constant value. The spline representing a circle can be determined by only 4 control points. More dominant curvature points are detected because the curvature and its derivative contain quantization noise. In Figure 4.11 the object consists of parts with higher and lower curvature. Thus the control point distribution is not ideal in the part with no curvature.



a) b) c) d)

Figure 4.11: B-spline curve with an adequate setting of dominant curvature points in all regions except for a region with a small curvature. (a) Curvature and first derivative with dominant points (b) dominant points (c) first B-spline (d) best B-spline

Furthermore the calculation of the curvature using a discrete angle between two vectors of definite length[3] is inaccurate. At this point a more accurate curvature function would improve this situation which is topic of further investigations.

Curve (a) in Figure 4.10 shows the number of control points needed by the automated representation of the B-spline for the best possible approximated B-spline. The best approximation was limited to 3 pixels deviation per span, as mentioned earlier.

Figure 4.12 illustrates how the number of control points rises when the spline is improved to a desired threshold. It is significant to note that the number of control points rises in proportion to the desired accuracy, but the number of iterations is very limited and does not rise proportionally to the accuracy. In many cases the number of iterations does not rise at all. This means that the number of iterations only bears a very weak relation to the desired accuracy.

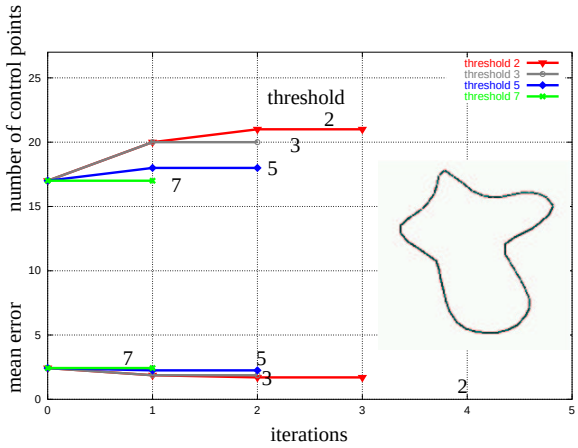


Figure 4.12: An object with different desired accuracies, showing the number of iterations required and the number of control points used

[3]If the curve is gentle and relatively short vectors are used (small k), the difference between an angle of e.g. 180° and 135° degrees is very small. Therefore the curvature value approximates zero.

Models of Object Motion

Models of object motion are determined online. No a priori knowledge about the object motion is needed thereby. The used method is described in Sec. 4.5.1, which is an adaption of the ARM method described in Sec. 3.3.

4.1.1 Discussion

In this section the models were described that are necessary for the used hand-eye system MinERVA. For the manipulator model the obtained model is an adaption of a general manipulator model as described in Sec. 3.1. For the camera the standard pin-hole camera model was used. The head model describes the transformation matrices of a two-link manipulator, since a pan-tilt head is used. Object motion models are derived from standard ARM models.

As a new method a procedure to obtain geometric shape models based on B-splines was developed. The proposed method automatically initializes and refines a B-spline that approximates a given object outline (silhouette). The most important steps for this automated mathematical approximation of the silhouette are: (a) the *initialization* of the B-spline, (b) the *correspondence search* between the B-spline and the silhouette for detecting deviation and (c) the *improvement* of the generated initial B-spline to optimize it (fitting). These B-spline object models are important for the later described tracking algorithm for tracking objects based on shape information.

4.2 Tracking of Moving Objects

First raw image segmentation methods will be analyzed and their suitability for tracking will be evaluated. Thereby visual tracking using different sensor modalities (form, color and motion) will be evaluated and the advantages and drawbacks of each sensor modality will be shortly discussed. Additionally explanations and statements for the chosen methods are given.

4.2.1 Contour-Based Tracking

In this section the methods and algorithms that were proposed for contour based tracking in Sec. 3.2.1.1 are evaluated further. Emphasis is put on the Condensation algorithm. Thereby additional explanations to the algorithm are given before advantages and drawbacks of this algorithm are discussed.

Explanations to the Condensation algorithm. "Practically speaking" for image processing, the CONDENSATION algorithm tries to find a known object outline $\mathbf{r}(\mathbf{s})$ in a set of contour edges. The search is done by "spreading" (using the *factored sampling* algorithm) a specified number of sample outlines (e.g. a sample-set of B-spline curves as described above) over an image and for each sample outline the degree of fit into the present set of contour edges is evaluated.

To obtain the sample-set that shall be evaluated, the algorithm first selects samples with high probability measures of the previous frame (see the large blobs in Figure 3.4). As those already had a high response in the old frame, it is assumed that new samples generated (*selected*) from an "good" old sample also will give high responses in the new frame. Therefore, samples with a high probability of fit are used as a basis for most of the new samples (i.e. multi-modal probability distribution). Since it is the goal to track an object in motion, the parameters of the new samples are propagated using a stochastic *prediction* model (e.g. an ARM overlaid by random noise) for each parameter (e.g. translation, rotation, scale), to have the evaluation of the new sample outlines at the most likely target object positions.

The way sample outlines can vary in their image position, rotation and size, i.e. which transformations are allowed is limited by the chosen *shape space* $\mathbf{W}$ (see [BI98]). E.g. in an *affine shape space* size variations are possible in horizontal and vertical direction independently. With these variations, almost all movements of a flat object can be tracked. Movement in a three dimensional workspace is covered by image position (x and y) and size (z = depth) (size changes equally in horizontal and in vertical direction). Rotation about the axis perpendicular to the camera image is covered by the rotation value. Another rotation about one of the two remaining axes is covered by splitting size into horizontal

and vertical components.

The *measurement* is obtained by searching for correspondences of sample outline pixels with contour edges. The more correspondences an outline has, the higher is its probability measure π_m.

This procedure *selection*, *prediction* and *measurement* is repeated as long as the object is being tracked.

Advantages and drawbacks of contour-based tracking Contour-based trackers are a mixture of a low-level and a high-level tracking method since additional knowledge about the form of the tracked object has to be available. Most contour trackers work only well with simple object forms as e.g. ellipses or lines (see [VAZ00]) as these forms can be calculated very fast and efficiently. Thereby usually real-time performance can be obtained.

To evaluate the contour in the image different "observation models" are used. In the simpliest case the evaluation is only along the contour. More sophisticated is to evaluate the contour along normals perpendicular to the contour, i.e. to raise the exactness of the observation model. The tracker gets less error prone, but demands additional calculational effort. Nevertheless contour trackers are easily attracted by background edges (clutter).

The big advantage using the contour for tracking is that grasping points on the object (contour) can be determined easily.

Advantages and drawbacks of CONDENSATION algorithm Being a statistical algorithm and its feature to maintain a multi-modal probability distribution temporal tracking failures (e.g. loss of the object) can be overcome by the Condensation algorithm. Additionally by the use of shape spaces it is assured that the model outline does not take forms that are impossible (because they are no transformations of the original form). Thereby it is also avoided that the tracked outline becomes too tangled where it can never recover.

The drawbacks are that for obtaining a good tracking performance a large number of samples, i.e. sample set is necessary. But if the sample set becomes too large real-time performance of the algorithm e.g. on a standard PC becomes impossible. Small sample sets result in inaccurate results or, even worse, loose track of the object too often. So it can be stated that the algorithm has a bad speed-accuracy trade-off. Additionally the algorithm is dependent on two exact models: an exact observation model and an exact motion model. Especially, if the motion model which has to be determined beforehand and off-line does not match the real object motion well tracking results get bad or tracking is even impossible.

To set this argumentation on a sound base different experiments have been performed which can be found in Sec. 5.1.2. For further experimental results [Osw00] is recommended.

4.2.2 Color-Based Tracking

Color tracking is reduced to a successive processing of single images. Other than in form tracking with the CONDENSATION algorithm, where the history of the object motion is used to support successful tracking, here only the current information available from the image data is exploited. Three criteria were interesting for our purpose: (1) one interesting object color has to be segmented by the chosen method[4]. (2) The segmentation method has to give positional and *rough* orientation information about the tracked object. And (3) the method has to be fast and computationally inexpensive.

The exactness of the filter regarding *critical* pixels, i.e. pixels that lie on the border of the subspace, was not problematic since it is the goal to compensate for those errors through the use of complementary filters, sensor preprocessing/fusion (see Section 4.3.1) and successive sensor integration (see Section 4.3.2).

The method chosen finally for most tracking experiments was a *YUV color filter with a cubic filter window*. The reasons for this choice were mainly quality reasons compared to computational costs i.e. (a) the YUV cube performed only marginally worse compared to e.g. the Dichromatic Reflection Model implementation, but at enormous lower computational costs (see Section 5.1.1). (b) using a color filter is favorable for image sequences since again computational load is less than for pixel clustering methods. (c) a simple subspace for a windowing method can be efficiently implemented and bears again computational efficiency compared to e.g. a CLUT. For more details on results and implementation details, respectively, see Section 5.1.1 or [Leu00].

Advantages and drawbacks of color-based tracking The advantages of a color-based tracking is simply: its fast, computationally inexpensive (even on a standard PC) and sufficiently robust. Additionally rough positional, size and even orientation information can be obtained by analyzing the found color blobs.

Drawbacks are that, when using a color filter, the filter has to be tuned beforehand. Additionally, depending also strongly on the filter, it is sensitive to illumination changes.

Color segmentation methods work well as blob trackers, but may be easily distracted by objects bearing the same color.

Again to support this argumentation different experiments were performed to evaluate the different methods. These experiments can be found in Sec. 5.1.1 or [Leu00].

[4]This seeming constraint can be justified by the fact that objects to be grasped are quite small and other sensor modalities can be used to obtain fine-scale information.

4.2.3 Motion-Based Tracking

Motion based tracking methods often base on the determination of the *optical flow field*. These methods are pre-destinated for tracking since they produce an interpretable output when there is motion in the scene. Therefore they are optimal for pre-segmenting an image into regions containing motion, and still image regions. In the nomenclature defined in Chap. 2 these regions can be defined as *attentive regions* and can serve to control the *focus of attention*.

One way to determine the optical flow field from a sequence of images in real-time was developed in [Stö01]. This approach combines the correlation of rectangular image regions (a common procedure used in video compression) with an efficient confidence measure. This approach allows on the one hand the efficient implementation of the algorithms either on special multi-media processors or on standard PC processors (e.g. Intel Pentium) in real-time, on the other hand the vector field determined this way relates well to the real image motion.

This sensor which relates to the class of low-level sensors can be described as fast and robust, but provides usually only sparse fine scale information. Therefore it is ideal for the combination with high level tracking methods like the later described Modified ICONDENSATION algorithm (see Section 4.3.2.1).

Advantages and drawbacks of motion-based tracking The main advantages of motion-based trackers are: (a) they are pre-destinated for tracking, since they generate an output if there is motion in a scene. (b) they are fast and the optic flow field can be calculated efficiently (even on standard PCs). (c) they are well suited for pre-segmenting an image, i.e. separate an image into regions where there is motion and regions without motion. (d) they are ideal to generate a "focus of attention", i.e. a region that is interesting to be analyzed further.

The drawback is that motion based tracking methods usually generate only little fine scale information.

4.2.4 Discussion

The reason for the former paragraphs was to review and discuss the different tracking methods that were evaluated and tested in visual tracking experiments. The goal was originally to find a reliable tracking method to be used for hand-eye experiments. To summarize it can be stated that: Up to now raw image segmentation methods were analyzed and their suitability for tracking was evaluated. Contour based trackers promise to work well in conjunction with high level methods, but may have drawbacks mainly in real-time performance, robustness and the need of a priori knowledge, e.g. a motion model

(see also Section 5.1.2). Color segmentation methods work well as blob trackers, provide sufficient information about the object, but may be easily distracted by objects bearing the same color. Finally motion detection methods like optic flow are fast and robust, but provide little fine scale information. In summary it can be said, that the advantage of one method is the drawback of another. Therefore it is self-evident that fusion and integration of the above methods may serve to obtain a robust object tracker suitable for hand-eye experiments.

4.3 Sensor Fusion and Integration

In the following a biologically motivated approach to sensor fusion and integration is proposed which was derived from the analysis of human visual pathways and the processing of visual information in the visual cortex or higher cortical areas (see Section 2.1.2), respectively. The approach can be split into "Sensor Preprocessing and Fusion" which can be set under the generic term *Pre-attentive Processing*, as well as "Sensor Integration" which will be termed *Attentive Processing* in this context.

4.3.1 Sensor Preprocessing and Fusion: Pre-attentive Processing

Image segmentation and preprocessing intends to obtain interesting image regions, e.g. areas where there is motion, from the available data. This operations can be performed *in parallel* for all interesting image features. Fusion or *dynamic linking* of different clues such as form, color and motion is obtained by applying the aforementioned (Sec. 2.1) broadly accepted principles of human visual processing, summarized under the term *Pre-attentive processing*, namely: (1) *parallel information flow*, (2) *pre-attentive processing*[5], (3) *reentry of information*[Kan91] and (4) *feature map* generation.

Parallel information flow applies thereby to the fact that visual information is processed in three parallel pathways (form, color, motion) divided horizontally several times, where an hierarchical processing (see *pipeline concept* below) of the information takes place.

Pre-attentive processing describes that this parallelism allows the optimal utilization of the present information at a time where no attention is put on a certain stimulus.

Reentry of information postulates that there exists a dynamic process of continuous parallel and recursive signaling between the strongly connected areas of the visual system.

Finally, *feature map* generation describes the transfer and entry of information from the three processing sources into an abstracted description. The information entered into the map can be any information (*feature*) about a region's position, size etcetera.

[5]Unfortunately the generic term is named the same way

A implementation model that covers the aforementioned principles is described below. The input images obtained by the camera are processed by an edge filter, a color filter and an optic flow sensor. The information obtained from this processing level is merged in higher levels and finally clustered in the highest level. Information from the highest level can be propagated back to lower levels of the same and other pathways. Thereby image regions which contain a high match of signals from different pathways are *dynamically intensified*. As an output feature maps are obtained which carry (in our case) *estimations about the current object shape and position*.

Pipeline concept Common to all sensor modules is a three stage pipeline design, with *input*, *segmentation* and *output* stages. Images and reentry information are received in the input stage, whereby images are provided by the cameras. The process of segmentation is performed in the second stage (see e.g. Section 3.2.1.2). The output stages remove noise from the segmentation results by morphological operations and propagate their results to both the *feature maps* and the sensor input stages. While moving from input- to output stage, sensor information turns from pixel information into clustered region information.

Figure 4.13 shows a schematic layout of this pipeline concept. The input stage also includes handling of reentry implementation, as described next.

Implementation of the Reentry Hypothesis Reentry information is backpropagated from the output stage of the visual sensors. This information is used to modify raw images in order to change segmentation parameters for spatial image regions. Regions indicated in the reentry information data will be treated with more attention in the reentry step than regions outside. These reentered regions are therefore called *attentive regions* and are always slightly enlarged copies of the original sensor result output regions[6].

Additional to the visual sensors which can incite each others sensor input, in the technical implementation of the reentry hypothesis "prediction" information is also seen as an incitation sensor. When tracking and searching an object, the prediction of the object's future position gives a hint were to search the object in the next frame. This feature is adopted here in a way that the prediction module can also simulate a sensor output stage and send information about the next estimated position, orientation and size of the object to the input layers of sensors.

The process of modifying the camera images can be described with the following formula:

$$P_m(x, y) = f(x, y)P_o(x, y)$$

with $P(x, y)$ being the pixel values at position (x, y) in an image. The indices m and o signify the modified and the original version. $f(x, y)$ is the modification function, defined separately for each visual sensor module. Figure 4.13 shows how the reentry hypothesis is implemented with the pipeline concept.

[6]This helps other filters to find their features within those enlarged regions (see cross influence)

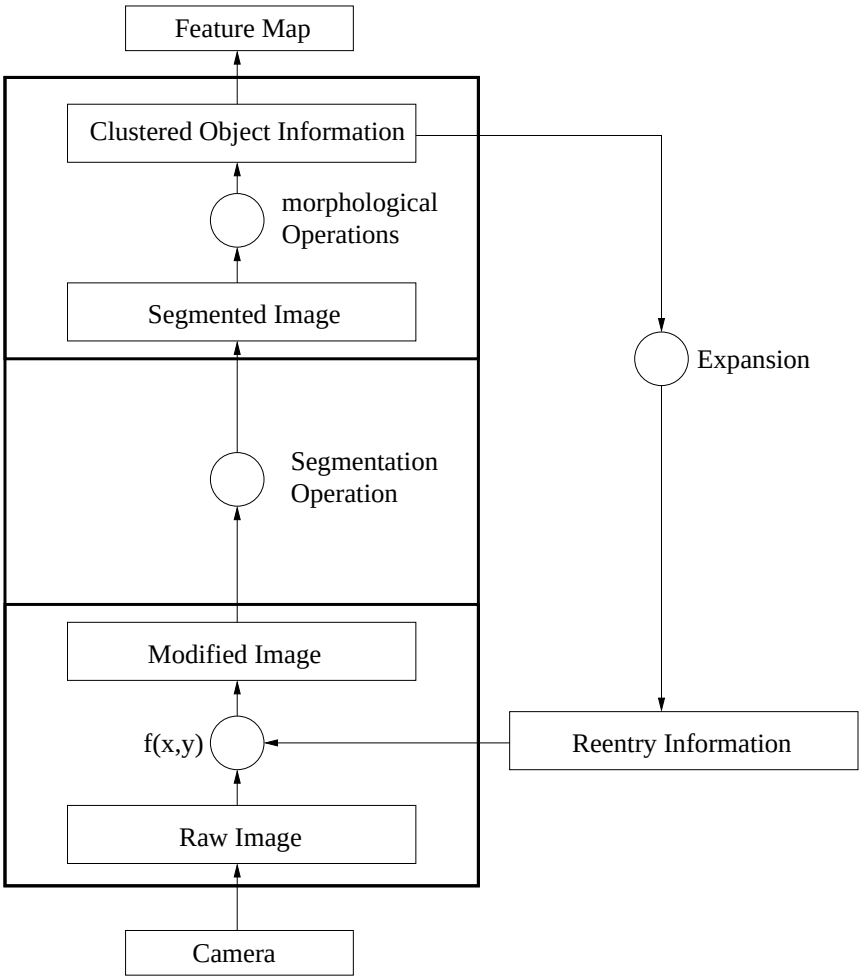


Figure 4.13: The pipeline concept

Form Sensor The form sensor used here is a combination of a Sobel filter with a successive threshold operation[7].

The sobel filter is applied *before* images enter the input stage of the form sensor. In the used implementation (see Section 4.8) a sobel matrix of size 3×3 is applied on the gray value image supplied by the camera. A successive threshold operation segments strong object contour edges. When reentry data is present, the sobel image pixels are multiplied by $m \geq 1$ before applying the threshold operation in regions indicated by the reentry data, and multiplied by $n = 1/m$ in regions outside. After multiplication, the threshold operation is applied on the modified sobel image, but with another threshold value (the reentry threshold value). This multiplication ensures that weak edge pixels in attentive regions get over the reentry threshold value. On the other side, strong edge pixels in non-

[7]With this implementation we leave the human model, as in the visual cortex form information is obtained by oriented cells, each of them responding to oriented line stimuli [MK91].

attentive regions are decreased in their value and therefore may end below the reentry threshold value. $f(x, y)$ of the form sensor is defined as:

$$\begin{aligned} f(x,y) &= \begin{cases} m & (x,y) \in \mathcal{A} \quad \textit{Enhancement} \\ \frac{1}{m} & (x,y) \notin \mathcal{A} \quad \textit{Inhibition} \end{cases} \\ m &\geq 1 \\ \mathcal{A} & \text{ is the union of all attentive regions} \end{aligned}$$

Attentive regions can be supplied to this sensor from all possible sources. If this sensor's output stage is connected to its own input stage, a high threshold in the first run allows only strong edges to pass. In the second filter run a lower reentry threshold value allows weaker edges lying close to strong edges to pass the second threshold operation. Supported by successive morphological operations fragmented edges are closed (line filling). The effect of supplying other attentive regions to this sensor is that more contour edges are found in attentive reentry regions and less, or even no edges are found in non-attentive regions.

Entry into Feature Map: an *enhanced* contour image. Contours that have had a strong response already in the first processing "run" will be intensified further by reentry, weak contours will be diluted.

Color Sensor The color sensor is implemented as a UV window filter (see Section 3.2.1.2 and Sec. 5.1.1 for reason for this choice). Pixels are filtered according to their chrominance values. Object color therefore is known as a range of accepted U and V values. For speed purposes the filter is implemented using Intel MMX technology [Int] and processes eight pixels a time in one cycle. Accepted pixel regions are smoothed by dilation and opening operations.

Reentry regions modify the chrominance pixels by zeroing and thus moving window pixels lying outside the attentive regions out of the filter. A different set of filter window parameters is applied to the modified chrominance values. $f(x, y)$ for the color sensor is defined as:

$$\begin{aligned} f(x,y) &= \begin{cases} 1 & (x,y) \in \mathcal{A} \quad \textit{Enhancement} \\ 0 & (x,y) \notin \mathcal{A} \quad \textit{Inhibition} \end{cases} \\ \mathcal{A} & \text{ is the union of all attentive regions} \end{aligned}$$

There can of course be other functions applied for enhancement and inhibition e.g. a function describing a "inverted Mexican hat" which would enhance the borders of the color region. Here only the most simple one is depicted.

Attentive regions can be supplied to this sensor from all possible sources. If this sensor's output stage is connected to it's own input stage, a small range of accepted chrominance

values allows only pixels matching almost exactly the known color. A broader range of accepted values in the second run allows to fill gaps between regions, as sometimes, due to lighting conditions, more shades of the known color are produced on the object's surface. The effect of supplying other attentive regions to this sensor is that color filtering is only applied to those regions. Non-attentive regions do not influence sensor results as they are moved completely to the border of the color space.

Entry into Feature Map: Information about color "blobs". This information may contain size, area, position or form information (e.g. compactness, roundness etc.)[8].

Motion Sensor The calculation of the optic flow was performed on a specialized hardware board containing a MEP (Motion Estimation Processor) that was developed at our institute [Stö01]. The output that can be obtained are rectangular areas containing objects that are moving against the background.

$$f(x,y) = \begin{cases} 1 & (x,y) \in \mathcal{A} \quad Enhancement \\ 0 & (x,y) \notin \mathcal{A} \quad Inhibition \end{cases}$$
$$\mathcal{A} \quad \text{is the union of all attentive regions}$$

Entry into Feature Map Regions in which motion information could be identified. Analogously to color the same region informations can be obtained and entered into the map.

Cross Influence of Sensors Color, Form and Motion: Sensor fusion As mentioned above, the back propagation of sensor output information is not limited to a single sensor modality. The sensor modalities can influence each other as well, only appropriate modification functions have to be chosen. This is depicted in Figure 4.14.

This kind of recurrent signaling process which is termed *Biologically motivated sensor preprocessing* here can well be seen as an extension of a typical sensor fusion process as defined before in Section 3.2.2: "evidence from multiple sensors is combined in order to produce information which is more precise than the output from an individual sensor; the fusion may involve raw sensor signals, or features and attributes extracted through preprocessing of the raw signals".

The definition fits our adoption of the human visual sensor pre-processing in many ways. On the one hand raw sensor signals such as the optic flow sensor output is combined with the edge sensor signal to obtain enhanced edge features. On the other hand selected edge features (obtained through the preprocessing of the raw signal) can be combined with e.g. color regions to obtain more precise information about the true borders of an color region.

[8]Those region information can easily be obtained using image processing libraries like *HALCON*([HAL])

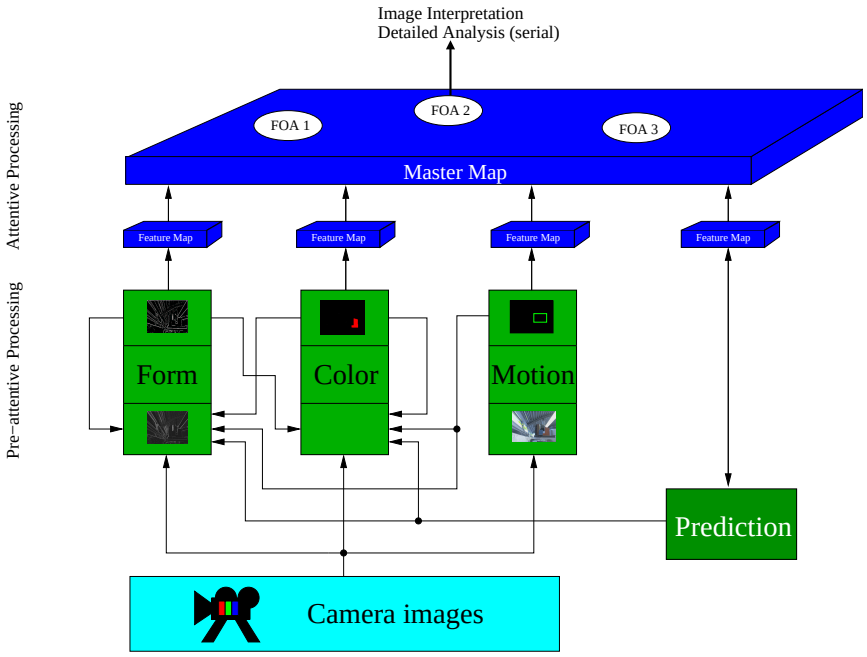


Figure 4.14: Combing the results of different sensor modalities

4.3.2 Probability Based Sensor Integration: Attentive Processing

After the fusion process locally enhanced or intensified, respectively, image regions are obtained and stored into feature maps. But strictly speaking those maps and the information they contain are still separated. What is needed is that the information is integrated in some way. This can be obtained by integrating the information of the different feature maps into a coherent description, a *master map*. This master map contains only those image regions (and all information about them provided by the feature maps), which are interesting for further evaluation and are processed one after the other, i.e. the *focus of attention* shifts from one region to the next. Working on a "focus of attention" the input from the feature maps is tested on coherence in a serial way, e.g. if there is a region of interest where is motion it is tested if this region is also entered in other feature maps. Since all this processing is done in a *serial way* and is "blocking all other processing" it is called *Attentive Processing*. For this kind of processing *model knowledge* is necessary.

In our case, tracking, the goal is to determine the position and orientation of an object in motion through a sequence of images. That means that after processing an image frame a distinct decision has to be made about *where* the tracked object is located in this frame.

Entry into Master Map The master map builds the output of the preprocessing and therefore the input to any further processing algorithms. Here the master map is represented by a locally intensified edge image. The intensified regions are enhanced according to the entries in the different feature maps. Additionally, information about the position,

size and orientation of interesting image regions is stored in the map[9].

This additional information is valuable and necessary for the succeeding algorithms, in this case the Modified ICONDENSATION algorithm as described below (Sec. 4.3.2.1).

In the next section Sec. 4.3.2.1 it is explained how the original ICONDENSATION algorithm was modified to (1) improve the distribution of samples which shall be evaluated, (2) to use the maximum information available from auxiliary sensors (and therefore all entries and information of the master map), and (3) to be able to adapt tracking parameters online.

4.3.2.1 Modified ICONDENSATION Algorithm

The original ICONDENSATION algorithm has been modified in four ways. First, samples are here classified into (a) *difference prediction samples*, (b) *prediction samples*, (c) *importance samples* and (d) *prior samples* depending on the sampled probability distribution. All samples are accompanied with a probability correction factor λ which is multiplied to the measured probability value π_m (see [IB98] and algorithm 1 for more explanation).

Second, the original prediction samples are split into two different types, the *difference prediction* samples and the *prediction* samples. For a *difference prediction* sample $\mathbf{s}_t$ the position of a base sample (acquired through sampling from the posterior density) is moved by the difference of the predicted object positions:

$$\mathbf{s}_t = \mathbf{s}_{t-1} + (\mathbf{p}_t - \mathbf{p}_{t-1}) + \mathbf{w}_k$$

where $\mathbf{w}_k$ is a vector of normal random (Gaussian) variates, $\mathbf{p}_t$ is the current predicted object position and $\mathbf{p}_{t-1}$ is the predicted position from the previous frame. The *difference prediction* samples behave similar as samples sampled from a prediction density described in the original CONDENSATION algorithm in [BI98].

The third modification concerns the *prediction* samples which get their new position, rotation and scale values from a prediction module:

$$\mathbf{s}_t = \mathbf{p}_t + \mathbf{w}_k$$

In standard ICONDENSATION object dynamics are learned off-line from training sequences. With our tracker, object movements are predicted from an *Average Autoregressive Model* (see Section 4.5.1) whose parameters are acquired and refined during tracking.

This is possible through the fourth modification which concerns *importance* samples. Here additional image information is not only used for providing a rough position estimation

[9]This storage is actually a table were the "numerical" region information is entered

but also shape and orientation parameters are acquired to place samples more efficiently. Altogether the modified algorithm is independent from off-line training, but more sensitive to the results of low level tracking (e.g optic flow).

Algorithm 1: Modified ICONDENSATION

Require: $step \in [1, N]$, $q \in [0, 1]$ and $r \in [0, 1]$ with $q + r \leq 1$
  Setup prior equal distribution of N samples.
  Measure π_m^{t-1} for all N samples. Set $\pi_i^{t-1} = \pi_m^{t-1}$
  loop
    Generate N new samples from base samples with high π_i^{t-1} (e.g. use cumulative probabilities [IB98])
    for all generated samples i at time t **do**
      Choose a random number $\alpha \in [0, 1]$
      Sample and predict as follows
      if $(i\%step == 0)$ **then**
        Use *difference prediction sampling*
        Sample from difference prediction p.d. to predict all sample properties. Set $\lambda = 1$
      else if $(\alpha < q)$ **then**
        Use *prior sampling*
        Sample from additional sensor information p.d. to get sample position. Set random values for rotation and size. Set $\lambda = 1$
      else if $(q \leq \alpha < q + r)$ **then**
        Use *importance sampling*
        Sample from additional sensor information p.d. to get sample position, rotation and size
        Set λ proportional to the distance of the predicted object position to the sample position
      else if $(\alpha \geq q + r)$ **then**
        Use *prediction sampling*
        Sample from AARM prediction p.d. for all sample properties (see 4.5.1). Set $\lambda = 1$
      end if
      Measure π_m^t. Calculate $\pi_i^t = \pi_m^t \lambda$. Store sample
    end for
    Produce output (e.g. Mean). Pass output to AARM (see Section 4.5.1) and the Grasp Tracker (see Section 4.4.1).
  end loop

Online adaption of tracking parameters The parameters $step \in [1, N]$, $q \in [0, 1]$ and $r \in [0, 1]$ with $q + r \leq 1$ influence the distribution of samples. Since this distribution does not necessarily be constant during tracking, it can be favorable to *online adapt* the distribution to the current tracking status: In the beginning a large number of samples is necessary to accurately detect the object to track. Additionally, in the beginning of a tracking sequence no knowledge about the object motion is available. Therefore, tracking can only be performed reliable if unique additional sensor knowledge is available. As a

consequence to this it is reasonable to have 100% of the samples spent for importance samples. After some time when tracking was successful and a motion model could be generated online the sample distribution can be changed to allow prediction and difference prediction samples. In the same rate the number of importance samples can drop. Now the tracking gets less sensitive to temporal sensor failures, i.e. it gets more robust. Additionally, when a sufficiently accurate motion model has been built up online, the total number of samples can be reduced to reduce processing time. The optimal distribution of samples to the appropriate type depends on the current tracking state. Ideally, the tracker has to evaluate its performance online to adopt the distribution to the current state. This adaption is equivalent to online *learning* of the system.

Experiments where the tracking parameters were adjusted in a priori fixed manner during tracking are shown in Section 5.

Sample Selection Strategies At this point N samples have been distributed, π_m has been measured and π_i has been calculated. Now we want to select out of all samples the one sample representing the current object position, rotation and size. Four different selection strategies have been tested to find the best sample s_t at a time t:

- Best matching sample

$$s_t : \pi_i(s_t^{(n)}) = \text{Max}(\pi_m(s_t^{(n)})) \qquad \forall\ n \in \{1, \ldots, N\}$$

- Mean of all samples

$$s_t : E(s_t) = \sum_{n=1}^{N} \pi_i^{(n)} s_t^{(n)}$$

- Best matching sample in the best cluster of samples

$$\begin{aligned} s_t : \pi_i(s_t^{(n)}) &= \text{Max}(\pi_m(s_t^{(n)})) \qquad \forall\ n \in \mathcal{B} \\ \mathcal{B} &\quad \text{is the set of samples in the best Cluster} \end{aligned}$$

- Mean of all samples in the best cluster of samples

$$\begin{aligned} s_t : E(s_t) &= \sum_{n} \pi_i^{(n)} s_t^{(n)} \qquad \forall\ n \in \mathcal{B} \\ \mathcal{B} &\quad \text{is the set of samples in the best Cluster} \end{aligned}$$

The first two strategies ignore the spatial correlation of the samples and operate on all samples. The best matching sample is the sample with the highest π value of all existing samples. Mean of all samples calculates mean values for all sample properties, weighted by their π value.

The last two strategies build spatial clusters of samples. The clustering algorithm is implemented as:

1. Take a sample $s^{(n)}, n \in \{1, \ldots, N\}$.
2. Compare its position to all existing clusters and remember the nearest cluster C_b.

$$\begin{aligned} C_b : d(s^{(n)}) &= \min(d(s_k^{(n)})) \qquad \forall\ k \in \{1, \ldots, K\} \\ d(s_k^{(n)}) &\quad \text{is the Euklidian distance of sample } n \text{ to cluster } k \\ K &\quad \text{is the number of clusters in } \mathcal{C} \\ \mathcal{C} &\quad \text{is the set of all clusters} \end{aligned}$$

3. If $d(s^{(n)}) \geq d_{max}$ or if there are no clusters yet, then make a new cluster and add this sample to the new cluster. Else add this sample to the nearest cluster C_b.
4. Go to 1 if there are more samples, end if there are no more samples left.

If a sample is added to a cluster, the cluster remembers the sample's π value and adds this to its own probability sum value $\pi_{cluster} = \sum_{n=1}^{N} \pi_i^{(n)}$. Additionally, the cluster recalculates its center point to which the sample distance is calculated. Then, the best cluster is chosen upon their probability sum value $\pi_{cluster}$. Best matching sample now searches for the sample with the highest π value within the best cluster. Mean of all samples in the cluster calculates position mean values, weighted with the samples π value and copies rotation and size values from the sample with the highest π value in the cluster.

The two clustering strategies were introduced to solve problems arising with mean sample calculation, when additional sensor regions produce groupings of samples on distractor objects that have values of $\pi > 0$. The unclustered mean strategy will shift its result towards the group of distractor samples. The clustering strategies overcome this problem by selecting the best cluster first and therefore calculating the median only for samples lying over the tracking object.

4.3.3 Discussion

Summarizing it can be stated that a method of fusing and integrating sensor information was proposed that is adopted from the visual information processing in the human brain. Thereby two concepts were transfered: the *reentry hypothesis* (fusion) and he concept of *feature maps* and *master map* (integration). For the fusion process the sensor modalities form, color and motion are first processed in parallel in separate pathways before they influence each other in a second processing cycle. Thereby image regions which contain a high match of signals from different pathways are *dynamically intensified*. As an output of this processing feature maps are obtained which contain *estimations about the current object shape and position*. To integrate the information of different feature maps into one coherent description, a *master map* is generated. This master map contains only those image regions which are interesting for further evaluation. Those regions are then processed

one after the other in a serial way, i.e. the *focus of attention* shifts from one region to the next. To evaluate this master map to generate an output, i.e. to determine the position and orientation of an object in motion a distinct decision has to be made about *where* the tracked object is located. This is done by the Modified ICONDENSATION algorithm.

The second proposal was an extension of the standard ICONDENSATION algorithm. Thereby the algorithm was modified in four ways: (1) to improve the distribution of samples which shall be evaluated, (2) to use the maximum information available from auxiliary sensors, (3) to be able to adapt tracking parameters online, (4) to online build up a motion model.

By these extensions one main problem occurring with the standard ICONDENSATION can be counteracted: the need of a priori known motion models for all parameters like position, scale and orientation. Additionally, by the online adaption of the tracking parameters the maximum number of needed samples can be reduced and thereby processing time can be lowered, if the tracking is successful for a certain time period.

4.4 Determination of Grasping Points

4.4.1 Search and Tracking of Grasps

The determination of grasping points was not in the focus of this thesis. The following algorithms were developed by Recatala and Sanz (see [RSL+02]) and are only shortly reviewed within the next section. Thereby two methods can be distinguished: The grasp search and the grasp tracking. The algorithms for grasp search can be applied on any contour description. For the grasp tracking the use of the aforementioned B-spline contours proved to be very favorable: since the tracked contour is always an affine transformation of a base contour also the grasping points are transformed affine and can be uniquely related to the current spline contour via the spline parameter s.

The methods were integrated with the aforementioned tracking methods. Results of the tracking together with the grasp determination can be found in Section 5.1.4 as well as in [RSL+02].

Grasp search Once an active contour has been associated with the object, it can be processed to search for pairs of grasp points at which the object can be stably grasped. The method proposed for searching for grasp points along the object contour is an extension of the one proposed in [MRSdP01]. The only *a priori* information it requires is the width of its fingers ($=2k$ in Figure 4.15(a)). For the evaluation of each grasp, point contacts with friction are assumed, and the Coulomb friction model [Ngu88] is used. The static coefficient between the object and the fingers is assumed unknown beforehand. This method is

described by algorithm 2.

Algorithm 2: Grasp Search

Extract a list of grasp regions
Generate a list of pairs of compatible regions
for each pair of compatible regions **do**
  Select best pair of grasp points
end for
Evaluate each selected grasp using a set of quality parameters
Sort grasps using this eval. and select the best one

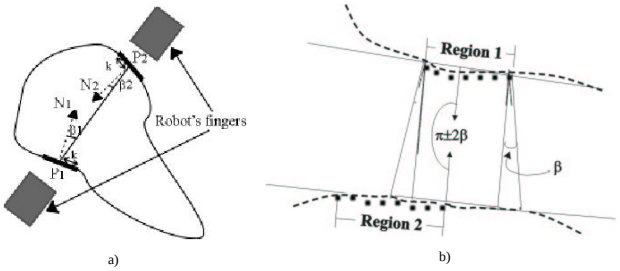


Figure 4.15: (a) Geometric interpretation of the quality criteria for the selection of stable grasps. (b) Compatibility test for grasp regions.

Extraction of grasp regions. In this step, the object contour is explored in search for *grasp regions* –regions of consecutive contour points where the curvature at each point remains under a given curvature threshold α. The curvature is discretely evaluated at each point. For that, a neighborhood of radius k centered at that point is considered. This evaluation is based on the expression of the k-torsion [SdPIR98], which includes a smoothing of the curvature. Ideally, it should be possible to approximate grasp regions as straight lines. Nevertheless, this method could produce regions covering portions of the contour that could not be reduced to straight lines. For this reason, the accumulated curvature along a grasp region is used to limit its size. The minimum size is also limited, since, being too small, they would not allow for tolerances in positioning of the fingers.

Unlike in [MRSdP01], the contours corresponding to internal holes of the object are not considered, since only one active contour has been associated to the external boundaries of the object.

Selection of compatible regions. Once the grasp regions have been found, a list of pairs of grasp regions is built. These regions are defined by including two points –one per region– such that, if the robot fingers are placed at them, the grasp complies with

the force-closure criterion. The regions in each pair will be termed hereinafter *compatible regions*.

The compatibility between two regions is verified through two conditions (see Figure 4.15(b)): (1) the angle between the normal vectors to each region is $\pi \pm \beta$, β being an angular tolerance threshold; and (2) the projection of each region, in the direction of the normal, intersects with the other region, that is, the regions are confronted.

These tests produce pairs of compatible regions such that the normals to each region point towards the space between both regions, and pairs in which they point towards outside this space. The first ones correspond to *squeezing grasps*, which are the only ones considered by many works on grasp determination. The others correspond to *expansion grasps*.

Grasp selection within regions. This step involves the selection of a pair of grasp points within each pair of compatible regions, and can be seen as a refinement of these regions. In this work, instead of an exhaustive checking of these regions to select the pair that best complies with the quality criteria, a faster search is performed that limits the checking to the neighborhood of the center of each region.

Grasp sorting and selection. The selected grasps are evaluated again according to how loosely they comply with each of the two quality criteria. In addition, the distance to the centroid is also considered. This produces a set of three quality values. Since these values correspond to different magnitudes, a normalization is performed. A global evaluation of each grasp is then computed based on this set of values. The grasps are sorted according to their global evaluation, and the grasp with the highest evaluation is selected[10]. The selected grasp is described by a pair of points on the contour, each one being the center of the region where the robot fingers should be placed. It is also specified if it is an expansion or a squeezing one.

Grasp tracking When an object is tracked along a sequence of images, an active contour is associated to it at every image (see Section 4.3.2.1). This contour is the result of applying some transformations (translation, rotation and/or scale) to the same reference contour used in the previous image. As the location of the grasp points selected for a contour is expressed with respect to the contour –or the object– itself, this location is invariant with respect to the above transformations. Therefore, in two contours that are a transformation of the same reference one, the location of the grasp in one can also be applied to the other. This allows tracking the grasp points along a sequence of images.

The precision of this tracking depends on the suitability of the relative location of the grasp points with respect to the object, considering the transformations applied to the contour.

[10]The global evaluation of the grasp is task-dependent, but usually a linear combination of the quality values has been considered, each value being given the same weight.

Algorithm 3: Grasp Tracking

```
if no grasp has been previously computed then
  Search for a grasp on the contour (see alg. 2)
  if a stable grasp has been found then
    Add the grasp to the tracking window
  end if
else
  {Search for a stable grasp in the tracking window}
  Start at the end of the tracking window of grasps
  repeat
    Take previous grasp in the window
    Translate loc. of taken grasp to current contour
    Evaluate the taken grasp
  until a stable grasp is found or all the grasps in the window have been evaluated

  {Search for a new grasp if no stable one is found}
  if a stable grasp has been found then
    Add the grasp to the tracking window
  else
    Search for a grasp on the contour (see alg. 2)
    if a stable grasp has been found then
      Add the grasp to the tracking window
    else
      Use last selected grasp (although unstable)
    end if
  end if
end if
```

For this reason, different location possibilities are allowed, and the tracking algorithm does not rely on any specific reference system. Here, the grasp points are expressed with respect to the contour of the object. Nevertheless, with some straightforward transformations, it is possible to have them located with respect to other reference systems within the object that, depending on the task to perform, may be preferable.

The algorithm for grasp tracking (see algorithm 3) tries to apply a previously computed grasp, or a number of them, to the current contour. These grasps are evaluated for stability. If none of them is stable, then a new grasp search is performed. The sets of previously computed grasps that are checked on the new contour constitute a *tracking window of grasps*. The length of this window can be set to an appropriate value depending on the task to perform. If it is zero, then no tracking at all is performed.

4.4.2 Discussion

A method to determine grasps on a spline contour and the succeeding tracking of these grasps is proposed. This procedure is favorable since tracking of a grasp –and finding a new one when required– uses only a small percentage of video rate period (see [RSL+02]). Therefor it is possible to track a contour and a grasp on it at video rate. In addition, this procedure allows a more efficient control of the robot arm towards the grasping points, since only in the case when a new grasp is determined a *relative* motion of the grasp on the object occurs that adds to the *absolute motion* of the tracked object.

4.5 Object Motion Reconstruction and Prediction

In this section an algorithm is described that is capable to predict future positions of a tracked object. Thereby the smoothing of the input data is inherent with the algorithm. The method is a extension of the ARM method described in Sec. 3.3.

4.5.1 Average ARM Prediction

As we have seen in the preceding Section 3.3, AR models are suitable for predicting future positions of an object in motion. In a real scenario the input data for the prediction comes from the vision processing module. This input data is usually corrupted by measurement noise, or even worse large outliers will occur occasionally. To tackle these problems either the measured values have to be pre-processed e.g. like described in Section 3.3 to smooth the input data and eliminate outliers, or the predictions themselves, used as input data for other processing modules, have to be stabilized somehow. The method described next explains a way how the predictions were stabilized in this work.

To predict more than one step into the future, the prediction is continued recursively to predict M values. To stabilize this prediction, a technique of building median values at different points of time was developed. The first step is to predict at different points of time and store all values in a prediction buffer. This process is defined as:

1. Get the measured value
2. Predict the next M values based on the already measured values and
3. Enter these predicted values in the prediction buffer.

After some time, the prediction buffer looks as shown in Figure 4.16. Usually, after re-

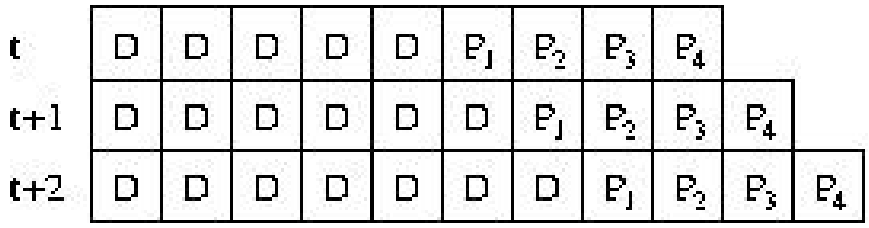


Figure 4.16: The prediction buffer with $M = 4$

ceiving the next measured value, M new prediction values are calculated and a new line is added at the bottom of the prediction buffer, holding the measured and the new predicted positions. Measured values are expected to arrive at every time interval Δt. If this is not the case, the prediction has to get by without a measured position. The corresponding

prediction fields (marked with "P") in the prediction buffer are left empty and the old measured positions (marked with "D") are copied to the new line in the prediction buffer. When the next measured value arrives, the missing measured value is interpolated. In Figure 4.17 the missing measured value is marked with "I".

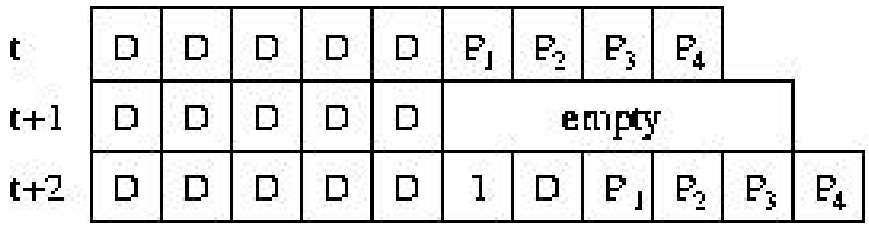


Figure 4.17: Buffer values in case of lost measurements

Starting from a given buffer state at time t, the predicted values $\mathbf{P}_t[j]$ at time t for points of time $t + j\Delta t$ are calculated as:

$$\begin{aligned} \mathbf{P}_t &= [x_t(t+\Delta t), x_t(t+2\Delta t), \ldots, x_t(t+M\Delta t)] \\ \mathbf{P}_t[j] &= x_t(t+j\Delta t) \qquad \forall\ j \in [1,\ldots,M] \end{aligned}$$

The predicted values $\mathbf{P}_{t+i\Delta t}[j]$ at time $t+i\Delta t$ for points of time $t+(i+j)\Delta t$ are calculated as:

$$\begin{aligned} \mathbf{P}_{t+i\Delta t} &= [x_{t+i\Delta t}(t+(i+1)\Delta t),\ x_{t+i\Delta t}(t+(i+2)\Delta t), \ldots,\ x_{t+i\Delta t}(t+(i+M)\Delta t)] \\ \mathbf{P}_{t+i\Delta t}[j] &= x_{t+i\Delta t}(t+(i+j)\Delta t) \qquad \forall\ i \geq 0,\ j \in [1,\ldots,M] \end{aligned}$$

The average ARM prediction values $AP_{t+i\Delta t}(j)$ are calculated as mean average values of k $P_{t+i\Delta t}(j)$ values, whereby k denotes the number of average iterations: $1 \leq k \leq i$.

$$\begin{aligned} AP(j) &= \frac{1}{\sum_{m=0}^{m=k-1} w_m} \sum_{m=0}^{m=k-1} p_{i-m}(j+m) w_m \\ w_m &= \begin{cases} w_0 = 1 \\ w_m = 0.6 w_{m-1} \end{cases} \\ p_{i-m}(j+m) &= \begin{cases} 0 & \text{if } \dim(\mathbf{P}_{t+(i-m)\Delta t}) = 0 \text{ (lost frame)} \\ 0 & \text{if } j+m > i \\ \mathbf{P}_{t+(i-m)\Delta t}[j+m] & \text{else} \end{cases} \end{aligned}$$

Figure 4.18 illustrates this process for $M = k = 4$. The average ARM prediction results are written to the buffer instead of the $\mathbf{P}_{t+i\Delta t}[j]$ results and used in the next prediction at time $t + (i+1)\Delta t$ as predicted values.

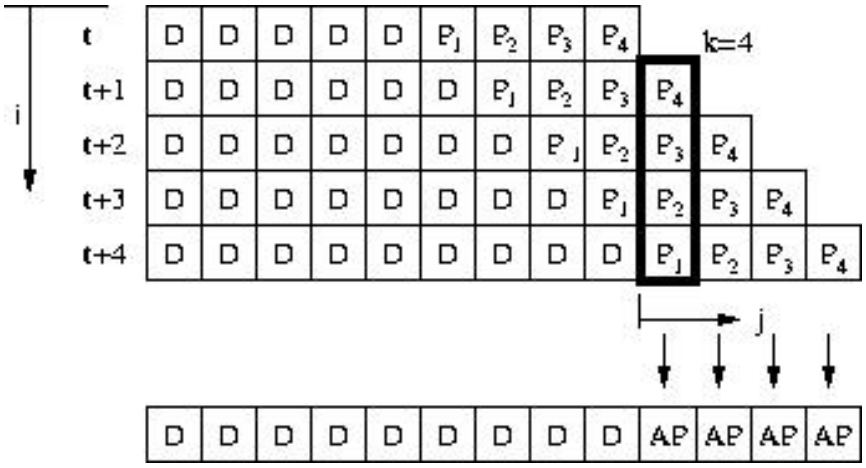


Figure 4.18: Averaging k prediction values.

4.5.2 Discussion

Despite the fact that this algorithm is very simple, since it is mainly generating weighted mean values of predicted positions at different points of time, it is well applicable in a real tracking and prediction scenario (see Sec. 5.2). Its accuracy proved also to be sufficient for hand-eye experiments (see Sec. 5.4). Additionally, it showed to be computationally very efficient compared to methods were the input data was smoothed before the prediction step (see [Sel00] for details).

4.6 Robot Arm Motion Control

4.6.1 Human Trajectory Generation

As has already been mentioned in Section 2.2.1 in prehensile movements, the transport component is used to reach the object to grasp. However moving the hand from point to point is a heavily under-constrained task [Kaw96]. To limit the enormous space of solutions researchers looked for parameters that stayed invariant for different reaching movements. Summarizing, it was found out that there exist indeed two invariant features, namely (1) that the path of the hand is roughly a straight line in Cartesian coordinates and (2) that the tangential velocity profile along this hand-path is bell-shaped.

We have seen that for modeling those invariant features researchers applied different methods. Mainly two could be distinguished. The first method tries to find models applying *optimization theory*, the second tries to *reproduce the tangential velocity profile*. Two representatives for those methods will be discussed more detailed in the following, namely the "Model of Flash" (optimization) and the "Model of Goodman" (reproduction of velocity profile).

Model of Flash

In engineering, obtaining a unique solution to a under constrained task is a typical problem solved by *optimal control theory*. The basic principle is to specify a so-called *objective function* or *performance index* and select a solution that minimizes it.

An optimization criterion that has been shown to match experimental data of human reaching very well is *minimum jerk*[11] [FH85], which was proposed by **Tamar Flash**. In other words, movements are planned in a way to assure maximal smoothness. Equation 4.13 states the objective function C for a 2D-movement, equation 4.14 the corresponding hand path

$$C = \frac{1}{2} \int_{t=0}^{t=T} \{(\frac{d^3x}{dt^3})^2 + (\frac{d^3y}{dt^3})^2\} dt \tag{4.13}$$

$$\mathbf{x}_1(t) = \mathbf{x}_0 + (\mathbf{x}_{T1} - \mathbf{x}_0) \cdot (10\tau_1^3 - 15\tau_1^4 + 6\tau_1^5) \tag{4.14}$$

with $\mathbf{x}_0$ and $\mathbf{x}_{T1}$ being the start and the target position, T being the movement duration, and $\tau_1 = t/T$. The resulting paths are straight Cartesian lines, the tangential velocity profiles are symmetric and bell-shaped.

Reactions to double-step targets, i.e. the case that the target "jumps" during the movement from $\mathbf{x}_{T1}$ to $\mathbf{x}_{T2}$, are explained by the so-called *superposition scheme* [FH91]: At

[11] Jerk = derivative of acceleration.

the time $t = t_2$, a second minimum jerk trajectory (Eq. 4.15) is superimposed on the first:

$$\mathbf{x}_2(t \geq t_2) = (\mathbf{x}_{T2} - \mathbf{x}_{T1}) \cdot (10\tau_2^3 - 15\tau_2^4 + 6\tau_2^5) \tag{4.15}$$

with $\tau_2 = (t - t_2)/(T - t_2)$. The parameters of the model were estimated by fitting it to experimental data. This model has been successfully implemented on a robot [Hen91].

Model of Goodman

It was found out that for many criteria that are minimized (e.g. jerk, energy etc.) the resulting velocity profile is always very similar, the aforementioned bell-shape. As already mentioned in Section 2.2.1, this lead **Goodman** to the proposal that it is not the objective during a human reaching movement to minimize one certain criterion, but that by achieving the bell-shaped profile many criteria are minimized. Therefore it was his objective not to find another criteria, but to find a description modeling the velocity profile. Here he found out, that a human reaching trajectory can be described by the following differential equation computing the current velocity of the hand:

$$\dot{\mathbf{x}}(t) = \frac{1}{\tau_1} \cdot \dot{g}(t) \cdot (\mathbf{x}_{T1} - \mathbf{x}(t)), \qquad g(t) = t^3 \tag{4.16}$$

$g(t)$ assures a bell-shaped velocity profile and a smooth, two-phasic acceleration profile. Following Goodman et al., it corresponds to a nonlinear perception of time found in psychophysical experiments. To transform Cartesian velocities into joint velocities, they presented a general scheme applicable also in the case of redundant degrees of freedom [GG95]. Equation 4.17 shows the solution of Eq. 4.16[12]:

$$\mathbf{x}_1(t) = \mathbf{x}_0 + (\mathbf{x}_{T1} - \mathbf{x}_0) \cdot (1 - e^{-\frac{1}{\tau_1} g(t)}) \tag{4.17}$$

For the case of a double-step target, they proposed that a second differential equation given in Eq. 4.18 is superimposed [GG95]:

$$\dot{\mathbf{x}}_2(t \geq t_2) = \frac{1}{\tau_2} \cdot \dot{g}(t - t_2) \cdot (\mathbf{D}_2 - \mathbf{x}_2(t)) \tag{4.18}$$

with $\mathbf{D}_2 = (\mathbf{x}_{T2} - \mathbf{x}_{T1})$. This leads to the solution

$$\begin{aligned} \mathbf{x}(t \geq t_2) \quad = \quad & \mathbf{x}_0 + (\mathbf{x}_{T1} - \mathbf{x}_0) \cdot (1 - e^{-\frac{1}{\tau_1} g(t)}) + \\ & (\mathbf{x}_{T2} - \mathbf{x}_{T1}) \cdot (1 - e^{-\frac{1}{\tau_2} g(t - t_2)}) \end{aligned} \tag{4.19}$$

Unfortunately, due to the nonlinearity in time Eq. 4.16 and Eq. 4.18 cannot be superimposed to form a single "visual-servoing-like" equation. However, simulations of Eq. 4.19 agreed well with experimental results.

[12]Note: Eq. 4.16 is very similar to the basic visual servoing control law [HHC96] only differing in the time-variant gain. But Goodman et al. however did not address visual feedback.

4.6.1.1 Static, Double-Step and Dynamic Targets

Two cases of trajectory generation have been described above. Both methods were able to describe the human behavior for reaching to a *static* and a *double-step* target.

Is is possible to use the aforementioned equations to describe also the reaching for a dynamic target? A dynamic target changes its position continuously. If this continuous change is described by discretization into *succeeding target steps*, those target positions could be used as an input for a hand (manipulator) to follow or track the target, respectively. To apply the method of e.g. Goodman the equations have to be generalized first:

taking the target point $\mathbf{x}_{T(i-1)}$ of a trajectory as the starting point of a succeeding trajectory with the target point $\mathbf{x}_{Ti}$, one gets by superposition on the velocity level:

$$\dot{\mathbf{x}}(t \geq t_n) = \sum_{i=1}^{n} \frac{3(t-t_i)^2}{\tau_i} (\mathbf{x}_{Ti} - \mathbf{x}_{T(i-1)}) e^{-\frac{(t-t_i)^3}{\tau_i}} \tag{4.20}$$

This is shown exemplarily in Figure 4.19 for the superposition of one element of $\dot{\mathbf{x}}(t)$ of two trajectories.

An application and the results of the continuous overlay of targets can be seen in Section 5.4.2 where the manipulator follows a circular moving target (the theory for that you can find in the next section). It is obvious that by this method the manipulator never will reach the target until the target object stops. Therefore other methods have to be applied to find points on the target's trajectory that lie "in the future" where the interaction of the manipulator with the target object shall take place. The determination of these hand-target interaction points and the resulting intermediate targets for the hand approaching the target object are described in Section 4.7.

4.6.2 Robotic Trajectory Generation

Trajectory generation for a robotic manipulator specifies the generation of position and orientation data, velocity and acceleration profiles to move the end-effector from a starting position to a target position. As input parameters for the trajectory generation serve the starting and the target position, the current end-effector position and orientation[13] as well as special parameters, which are necessary for planning the trajectory (e.g. the time given to reach the target position, the accuracy that shall be reached at the target position, ...). With this input data the trajectory generation calculates necessary intermediate targets, which shall be reached by the end-effector in certain time-steps before the end position is finally reached. By use of the inverse differential kinematics it is possible that the trajectory

[13] feedback is mainly used for error corrections, see also [Hau99]

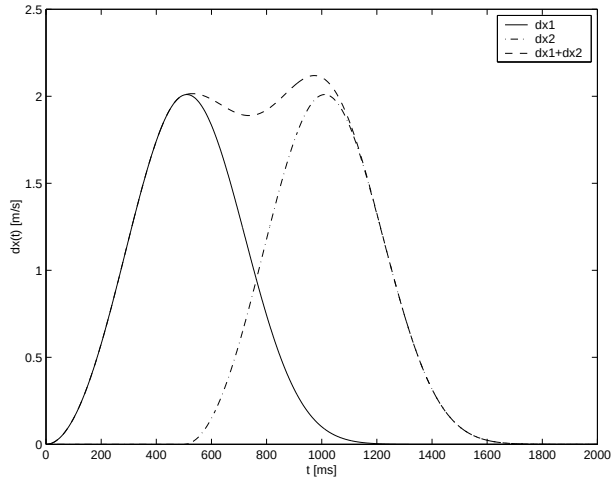


Figure 4.19: Superposition of biologically motivated trajectories

generation produces translatory and rotatory velocities instead of the according position and orientation vectors. All those data is referenced to the world coordinate system.

Biologically Motivated Trajectories As described in Section 2.2.1 and modeled by Equation 4.14 and Equation 4.16 a human grasping trajectory between a starting position and a target position has the invariant features that (1) the hand moves on a straight line and (2) the tangential velocity profile of the hand is bell-shaped.

For the trajectory generation of MINERVA an extended version of the model of Goodman was developed in [Hau99]. This extended model generalizes the model of Goodman in a way that it is able to handle *multiple target jumps* as well as the *integration of visual and proprioceptive feedback* during the movement. Especially the extension that as a reaction on a target jump ongoing arm trajectories can be super-positioned is very useful for the grasping of moving objects, since usually continuous online adaptions of the end-effector

trajectory are necessary to accurately catch an object.

In the following first a short repetition of the main results concerning the trajectory generation on positional level according to [Hau99] is given (Section 4.6.2.1), then its extension to control the robots orientation is presented in Section 4.6.2.2.

4.6.2.1 Determination and Control of Hand's Position

With the trajectory generation for the position, the viapoints for the position ${}^{w}\mathbf{p}(t)$ of the robot's end-effector shall be calculated, i.e. in this case the translatory velocity ${}^{w}\dot{\mathbf{p}}(t)$ of the end-effector.

Therefore following abbreviations which were already defined before are shortly repeated:

$$\mathbf{D}_i = {}^{w}\mathbf{p}_{Ti} - {}^{w}\mathbf{p}_{T(i-1)} \quad \text{with} \quad {}^{w}\mathbf{p}_{T0} = {}^{w}\mathbf{p}(t=0) \tag{4.21}$$

$$e_i(t) = e^{-\frac{1}{\tau_i}(t-t_i)^3} \tag{4.22}$$

where $\mathbf{D}_i$ is the vectorial difference between two target points (for a target jump).

By superposition on velocity level one gets for ${}^{w}\dot{\mathbf{p}}(t)$:

$${}^{w}\dot{\mathbf{p}}(t \geq t_n) = \sum_{i=1}^{n} \frac{3(t-t_i)^2}{\tau_i} \cdot \mathbf{D}_i \cdot e_i(t) \quad \text{with} \quad t_1 = 0 \tag{4.23}$$

Equation 4.23 describes the time course of the translatory velocities after n target jumps.

The solution of Equation 4.23 is therefore:

$${}^{w}\mathbf{p}(t \geq t_n) = {}^{w}\mathbf{p}_{Tn} - \sum_{i=1}^{n} \mathbf{D}_i \cdot e_i(t) \tag{4.24}$$

One drawback of equation Equation 4.23 is, that ${}^{w}\mathbf{p}(t)$ is not used for the calculation of ${}^{w}\dot{\mathbf{p}}(t)$. But for the integration of visual or proprioceptive feedback[14] it is necessary to integrate the current end-effector position in the calculation of the end-effector's velocity.

Use of substitutions and mathematical transformations leads to following equation (see

[14] see [Hau99] for the use of proprioceptive or visual feedback in human or robotic systems

[Hau99] p. 52 et seqq. for more details):

$$ {}^{w}\dot{\mathbf{p}}(t \geq t_n) = \frac{3(t-t_n)^2}{\tau_n}({}^{w}\mathbf{p}_{Tn} - {}^{w}\mathbf{p}(t)) + \sum_{i=1}^{n-1} \mathbf{D}_i \cdot e_i(t) \cdot \left(\frac{3(t-t_i)^2}{\tau_i} - \frac{3(t-t_n)^2}{\tau_n} \right) \tag{4.25} $$

Finally, only the determination of the factor τ_i is left. τ_i can be determined by specifying of the desired end-point exactness ϵ_{p_i} and time T_i for the end-effector to move from the starting position to the target position.

With:

$$ |\, {}^{w}\mathbf{p}_{Ti} - {}^{w}\mathbf{p}(T_i)| \leq \epsilon_{p_i} \tag{4.26} $$

Elimination of the "is lower than" sign and use of Equation 4.24 with $n = 1$ leads to:

$$ \epsilon_{p_i} = |\mathbf{D}_i| \cdot e^{-\frac{T_i^3}{\tau_i}} \tag{4.27} $$

By a simple transformation one gets finally the necessary value for τ_i:

$$ \tau_i = \frac{T_i^3}{\ln\left(\frac{|\mathbf{D}_i|}{\epsilon_{p_i}}\right)} \tag{4.28} $$

With Equation 4.28 only the exactness regarding the position is granted. If it is desired that the velocity at the target position reaches a certain value, this can be integrated into the determination of τ_i in the following manner:

According to [Hau99] the non-equation is prerequisite here:

$$ |\, {}^{w}\dot{\mathbf{p}}(T_i)| \leq \epsilon_{v_i} \tag{4.29} $$

which leads to

$$ \epsilon_{v_i} = \frac{3T_i^2}{\tau_i} \cdot |\, {}^{w}\mathbf{p}_{Ti} - {}^{w}\mathbf{p}(T_i)| = \frac{3T_i^2}{\tau_i} \cdot \epsilon_{p_i} \tag{4.30} $$

and finally τ_i is calculated to

$$ \tau_i = 3T_i^2 \frac{\epsilon_{p_i}}{\epsilon_{v_i}} \tag{4.31} $$

Finally, to map the position values from Cartesian space to joint space the equations described in Section 3.1.1 have to applied.

4.6.2.2 Determination and Control of Hand's Orientation

To determine the "trajectory" for the orientation of the manipulator the same principle as for the position control shall be used, i.e. using a biologically motivated strategy for the control. The main difficulty here is to transform those rules to be suitable for orientation control.

In [Mai99] is suggested to apply biologically motivated trajectories (see Equation 4.16) to calculate trajectories for Euler angles, which describe the current orientation of the end-effector. Because of reasons which are described in more detail in Section 3.1.1, the use of Euler angles is left out here, but the use of approach and normal vector is prefered in the following.

Thereto it is necessary to extend the principle of biologically motivated trajectories to be usable with approach- and normal vectors. In the following the solution to this problem is explained.

Like explained in Section 3.1.1 one can use an approach vector ${}^{w}\mathbf{a}$ and a normal vector ${}^{w}\mathbf{n}$ to describe the orientation of an end-effector. With these vectors a rotation matrix ${}^{w}_{e}\mathbf{R}$ according to Equation 4.10 can be calculated which also describes the orientation of the end-effector in an unique way. In the following a rotation matrix $\mathbf{R}$ is used having the properties:

- being an orthonormal matrix with row vectors forming a right-handed coordinate system
- $$\det \mathbf{R} = 1 \tag{4.32}$$
- $$\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I} \tag{4.33}$$

The same features hold for matrix ${}^{w}_{e}\mathbf{R}$.

Determination of the rotation matrix from an rotation axis and an rotation angle
A rotation matrix in $\mathcal{R}^3$ can be calculated by a normalized vector $\mathbf{u}$ and a rotation angle

φ. $\mathbf{u}$ forms thereby a rotation axis, φ is an angle with a value range $0 \leq \varphi < 2\pi$ by which the counterclockwise rotation around $\mathbf{u}$ takes place.

$$\mathbf{R} = (\mathbf{u}\mathbf{u}^T) + \cos\varphi(\mathbf{I} - \mathbf{u}\mathbf{u}^T) + \sin\varphi \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \quad \text{with} \quad |\mathbf{u}| = 1 \tag{4.34}$$

Solved and summarized gives for matrix $\mathbf{R}$:

$$\begin{aligned} \mathbf{R} &= \begin{bmatrix} u_x^2 + \cos\varphi(1-u_x^2), & u_x u_y(1-\cos\varphi) - u_z\sin\varphi, & u_x u_z(1-\cos\varphi) + u_y\sin\varphi \\ u_x u_y(1-\cos\varphi) + u_z\sin\varphi, & u_y^2 + \cos\varphi(1-u_y^2), & u_y u_z(1-\cos\varphi) - u_x\sin\varphi \\ u_x u_z(1-\cos\varphi) - u_y\sin\varphi, & u_y u_z(1-\cos\varphi) + u_x\sin\varphi, & u_z^2 + \cos\varphi(1-u_z^2) \end{bmatrix} \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \end{aligned} \tag{4.35}$$

Determination of an rotation axis and an rotation angle from a rotation matrix
The other way around it is also possible to calculate a rotation axis $\mathbf{u}$ and the according rotation angle φ from a given rotation matrix $\mathbf{R}$.

By use of the condition $|\mathbf{u}| = 1$, the rotation angle φ having a value range $0 \leq \varphi < 2\pi$ can be calculated from the sum of the diagonal elements of $\mathbf{R}$:

$$u_x^2 + u_y^2 + u_z^2 + \cos\varphi(3 - u_x^2 - u_y^2 - u_z^2) = r_{11} + r_{22} + r_{33} \tag{4.36}$$

$$\Rightarrow \quad \cos\varphi = \frac{1}{2}(Spur\mathbf{R} - 1) = \frac{1}{2}(r_{11} + r_{22} + r_{33} - 1) \tag{4.37}$$

$$\text{with} \quad 0 \leq \varphi < 2\pi, \quad \text{and clockwise rotation around axis } \mathbf{u}$$

The rotation axis can be obtained by determination of the only (!) Eigen vector of $\mathbf{R}$.

$$(\mathbf{R} - \mathbf{I})\,\mathbf{u} = \mathbf{0} \quad \wedge \quad |\mathbf{u}| = 1 \tag{4.38}$$

Thereby the orientation of $\mathbf{u}$ is determined, but not the direction of $\mathbf{u}$. Using φ the correct direction of $\mathbf{u}$ has to be specified finally (one gets the same rotation matrix, if one turns clockwise around the one possible direction of $\mathbf{u}$ with φ, as well as, if one turns counterclockwise by the angle $(2\pi - \varphi)$). This problem can be solved if it is defined that always the minimal rotation angle φ shall be used.

$$0 < \varphi < \pi \tag{4.39}$$

For $\varphi = 0$ the calculation of the rotation axis is not reasonable, since in this case no rotation shall take place. The same holds for $\varphi = \pi$, since in that case a limit had to be set. This problem can be avoided, if $\mathbf{u}$ is determined from Equation 4.38, whereby the direction of $\mathbf{u}$ can be chosen randomly (rotation by 180°).

Instead of using Equation 4.38 a simple comparison of the coefficients of both matrices on the right side of Equation 4.35 is done. Exemplarily one gets from r_{32} and r_{23} by subtraction of (Equation 4.40–Equation 4.41):

$$r_{32} = u_y u_z (1 - \cos\varphi) + u_x \sin\varphi \tag{4.40}$$

$$r_{23} = u_y u_z (1 - \cos\varphi) - u_x \sin\varphi \tag{4.41}$$

$$r_{32} - r_{23} = 2u_x \sin\varphi \quad \Rightarrow \quad u_x = \frac{r_{32} - r_{23}}{2\sin\varphi} \tag{4.42}$$

Applying this operation on all non-diagonal elements of $\mathbf{R}$, $\mathbf{u}$ and the according φ results to:

$$\begin{gathered} \mathbf{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \tfrac{1}{2\sin\varphi} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} = \mathbf{F}_{\mathbf{u}}(\mathbf{R}) \\ \varphi = \arccos(\tfrac{1}{2}(r_{11} + r_{22} + r_{33} - 1)) = F_{\varphi}(\mathbf{R}) \\ \text{with} \quad 0 < \varphi < \pi, \quad |\mathbf{u}| = 1 \end{gathered} \tag{4.43}$$

With these Eq. 4.43 a rotation axis $\mathbf{u}$ with a minimal necessary rotation angle φ can be determined from a rotation matrix $\mathbf{R}$. Inserting $\mathbf{u}$ and φ in Equation 4.34 again, $\mathbf{R}$ can be calculated.

Application on biologically motivated trajectories Basing on the world coordinate system (see Figure 4.20)

$$\mathcal{F}_w : \begin{bmatrix} {}_w\mathbf{b}_x & {}_w\mathbf{b}_y & {}_w\mathbf{b}_z \end{bmatrix} = \mathbf{I} \tag{4.44}$$

the orientation of the end-effector of a starting point

$$\mathcal{F}_{e1} : \quad {}^w_{e1}\mathbf{R} = \begin{bmatrix} {}^w_{e1}\mathbf{b}_x & {}^w_{e1}\mathbf{b}_y & {}^w_{e1}\mathbf{b}_z \end{bmatrix} \tag{4.45}$$

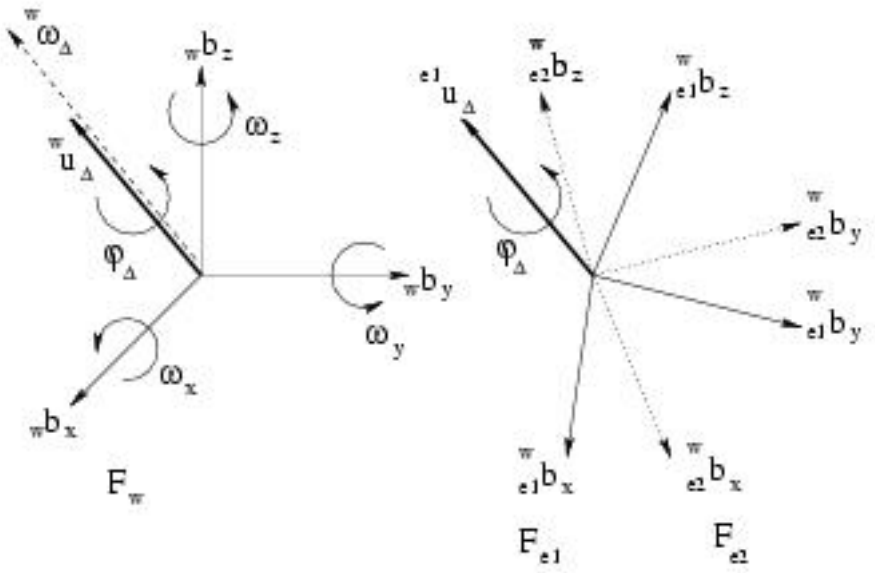


Figure 4.20: Calculation of the angular velocity vector for the orientation trajectory

is given, and the orientation of the target point is given by

$$\mathcal{F}_{e2}: \quad {}^{w}_{e2}\mathbf{R}_T = \left[\begin{array}{ccc} {}^{w}_{e2}\mathbf{b}_x & {}^{w}_{e2}\mathbf{b}_y & {}^{w}_{e2}\mathbf{b}_z \end{array} \right] \tag{4.46}$$

Now a "rotation difference" ${}^{e1}\mathbf{R}_\Delta$ relative to $\mathcal{F}_{e1}$ can be determined:

$${}^{w}_{e2}\mathbf{R}_T = {}^{w}_{e1}\mathbf{R} \cdot {}^{e1}\mathbf{R}_\Delta \tag{4.47}$$

$$\Rightarrow \quad {}^{e1}\mathbf{R}_\Delta = {}^{w}_{e1}\mathbf{R}^T \cdot {}^{w}_{e2}\mathbf{R}_T \tag{4.48}$$

From ${}^{e1}\mathbf{R}_\Delta$ the rotation axis and the rotation angle is calculated using Equation 4.43:

$$\begin{aligned} {}^{e1}\mathbf{u}_\Delta &= \mathbf{F}_\mathbf{u}({}^{e1}\mathbf{R}_\Delta) && (4.49) \\ \varphi_\Delta &= F_\varphi({}^{e1}\mathbf{R}_\Delta) && (4.50) \end{aligned}$$

whereby the vector of the rotation axis is given in coordinates of $\mathcal{F}_{e1}$.

The important point hereby is that ${}^{e1}\mathbf{u}_\Delta$ is exactly the orientation and direction of an angular velocity vector around that the end-effector turns with the angular velocity $\dot{\varphi}_\Delta$ to

get from the starting orientation to the target orientation (see also Figure 4.20):

$$ {}^{w}\boldsymbol{\omega}_{\Delta} = {}^{w}\mathbf{u}_{\Delta} \cdot \dot{\varphi}_{\Delta} = {}^{w}_{e1}\mathbf{R} \cdot {}^{e1}\mathbf{u}_{\Delta} \cdot \dot{\varphi}_{\Delta} \tag{4.51} $$

Now the principle behind the biologically motivated trajectories can be applied to the rotation angle φ_{Δ}. Therefore the differential Equation 4.16 is adapted appropriately:

$$ \begin{aligned} \dot{\varphi}_{\Delta}(t) &= \frac{3t^2}{\tau_r}(\varphi_{\Delta_T} - \varphi_{\Delta}(t)) \\ &= \frac{3t^2}{\tau_r}(\varphi_{\Delta_T} - \varphi_{\Delta_0}) \cdot e^{-\frac{t^3}{\tau_r}} \\ &\quad \text{with} \quad \varphi_{\Delta_0} = 0 \quad \text{and} \quad \varphi_{\Delta_T} = \varphi_{\Delta} \end{aligned} \tag{4.52} $$

Since the angle φ_{Δ_0} is always 0 at $t = 0$ and since the "target angle" is $\varphi_{\Delta_T} = \varphi_{\Delta}$, Equation 4.51) can be written as:

$$ {}^{w}\boldsymbol{\omega}_{\Delta} = {}^{w}_{e}\mathbf{R} \cdot {}^{e}\mathbf{u}_{\Delta} \cdot \varphi_{\Delta} \cdot \frac{3t^2}{\tau_r} \cdot e^{-\frac{t^3}{\tau_r}} \tag{4.53} $$

With the abbreviation

$$ e_{r_i}(t) = e^{-\frac{1}{\tau_{r_i}}(t-t_i)^3} \tag{4.54} $$

the equation of motion for the angular velocity vector in $\mathcal{F}_w$ can be obtained by superposition from Equation 4.53, like it is needed for the inverse differential kinematics. Matrices ${}^{w}_{e}\mathbf{R}_{Ti}$ form the particular target orientations.

$$ \begin{aligned} {}^{w}\boldsymbol{\omega}(t \geq t_n) &= \sum_{i=1}^{n} {}^{w}_{e}\mathbf{R}_{T(i-1)} \cdot {}^{e}\mathbf{u}_{\Delta_i} \cdot \varphi_{\Delta_i} \cdot \frac{3(t-t_i)^2}{\tau_{r_i}} \cdot e_{r_i}(t) \\ \text{with} \quad & t_1 = 0 \\ & \mathbf{R}_{\Delta_i} = {}^{w}_{e}\mathbf{R}^{T}_{T(i-1)} \cdot {}^{w}_{e}\mathbf{R}_{Ti} \\ & {}^{e}\mathbf{u}_{\Delta_i} = \mathbf{F}_{\mathbf{u}}(\mathbf{R}_{\Delta_i}) \\ & \varphi_{\Delta_i} = F_{\varphi}(\mathbf{R}_{\Delta_i}) \\ & {}^{w}_{e}\mathbf{R}_{T0} = {}^{w}_{e}\mathbf{R}(t_1 = 0) \end{aligned} \tag{4.55} $$

Here it is also desired to integrate the current orientation of the end-effector into the calculation of the angular velocity. Since Equation 4.25 is derived from Equation 4.23 the same calculation can be formally applied to Equation 4.55. But it is not necessary to apply this non-trivial conversion to Equation 4.55. It is sufficient to substitute the vector difference $({}^{w}\mathbf{p}_{Tn} - {}^{w}\mathbf{p}(t))$ in Equation 4.23 by the "rotation difference" (same structure of Equation 4.25 and Equation 4.55!) to get:

$$
\begin{aligned}
{}^{w}\boldsymbol{\omega}(t \geq t_n) &= \frac{3(t-t_n)^2}{\tau_{r_n}} \cdot {}^{w}_{e}\mathbf{R}(t) \cdot {}^{e}\mathbf{u}_{\Delta}(t) \cdot \varphi_{\Delta}(t) \\
&+ \sum_{i=1}^{n-1} {}^{w}_{e}\mathbf{R}_{T(i-1)} \cdot {}^{e}\mathbf{u}_{\Delta_i} \cdot \varphi_{\Delta_i} \cdot e_{r_i}(t) \cdot \left(\frac{3(t-t_i)^2}{\tau_{r_i}} - \frac{3(t-t_n)^2}{\tau_{r_n}} \right)
\end{aligned}
\tag{4.56}
$$

with

$$t_1 = 0$$

$$\mathbf{R}_{\Delta_i} = {}^{w}_{e}\mathbf{R}^{T}_{T(i-1)} \cdot {}^{w}_{e}\mathbf{R}_{Ti}$$

$${}^{e}\mathbf{u}_{\Delta_i} = \mathbf{F}_{\mathbf{u}}(\mathbf{R}_{\Delta_i})$$

$$\varphi_{\Delta_i} = F_{\varphi}(\mathbf{R}_{\Delta_i})$$

$${}^{w}_{e}\mathbf{R}_{T0} = {}^{w}_{e}\mathbf{R}(t_1 = 0)$$

$${}^{e}\mathbf{R}_{\Delta}(t) = ({}^{w}_{e}\mathbf{R}(t))^{T} \cdot {}^{w}_{e}\mathbf{R}_{Tn}$$

$${}^{e}\mathbf{u}_{\Delta}(t) = \mathbf{F}_{\mathbf{u}}({}^{e}\mathbf{R}_{\Delta}(t))$$

$$\varphi_{\Delta}(t) = F_{\varphi}({}^{e}\mathbf{R}_{\Delta}(t))$$

${}^{w}_{e}\mathbf{R}(t)$ is the current orientation of the end-effector at point of time t, and ${}^{e}\mathbf{R}_{\Delta}(t)$ is the "rotation difference matrix" between the target orientation and the current orientation.

The calculation of the factor τ_{r_i} is performed analogously to Equation 4.28 and Equation 4.31, but instead of the distance $|\mathbf{D_i}|$ between start- and target position the needed rotation angle φ_{Δ_i} between start- and target orientation is used. ϵ_{rp_i} is the exactness in rad by which the target position shall be reached, ϵ_{rp_i} the remaining velocity (in rad/s) at

the target position.

$$\tau_{r_i} = \frac{T_i^3}{\ln\left(\frac{\varphi_{\Delta_i}}{\epsilon_{rp_i}}\right)} \tag{4.57}$$

$$\tau_{r_i} = 3T_i^2 \frac{\epsilon_{rp_i}}{\epsilon_{rv_i}} \tag{4.58}$$

4.6.2.3 Collision Detection and Workspace

The task of collision detection is the avoidance of collision between parts of the manipulator themselves and other objects, especially the base on which the manipulator is mounted. Like explained in Section 3.1.1 it is possible to integrate performance criteria in the inverse differential kinematics. This procedure comes along with an enormous calculational effort. Nevertheless it is not granted that no collision occurs since the criteria are simply added and can therefore influence or even extinguish each other.

Therefore in this work a different, more simple method is applied to avoid collisions. Collisions of two links which are connected by a joint can be avoided by limiting the value ranges of the joints. Unfortunately collisions of parts of the arm which are not straightly connected by a joint as well as collisions with other parts cannot be avoided by this procedure.

One way dealing with this problem is to choose an appropriate workspace. Doing so collisions of the end-effector can be avoided in most cases.

By defining a workspace it can also be avoided that the manipulator moves towards target points which cannot be reached due to the manipulator's configuration or geometry. In case that a target position is chosen that lies outside the workspace, the target position must be corrected in a way that it comes to lie inside the workspace. The simplest method to achieve this is to move the point on the border of the workspace in a way that the distance between the corrected and the given target point is minimal. If further target points are given that are outside the workspace the manipulator moves on the border of the workspace having a minimal distance to the given targets.

One difficulty thereby is, that the workspace continuously changes through the changing orientation of the manipulator. As a simple solution to this problem the orientation of the manipulator can be limited and the workspace can be adjusted accordingly, so that every point inside the workspace can be reached with the allowed orientation.

In the following a simple geometric form for the workspace is chosen, namely a bowl. This bowl is described by a radius r_s and the position of the center of the bowl in world coordinates ${}^w\mathbf{p}_s$.

If it is necessary to correct a target point ${}^w\mathbf{p}_T$ such that it comes to lie inside the workspace,

this can be achived in the following way (see also Figure 4.21):

$$
\begin{aligned}
{}^{w}\mathbf{p}'_{T} &= \frac{r_s}{|\,{}^{w}\mathbf{p}_T - {}^{w}\mathbf{p}_s|}({}^{w}\mathbf{p}_T - {}^{w}\mathbf{p}_s) + {}^{w}\mathbf{p}_s \\
&\text{if} \quad |\,{}^{w}\mathbf{p}_T - {}^{w}\mathbf{p}_s| > r_s
\end{aligned}
\tag{4.59}
$$

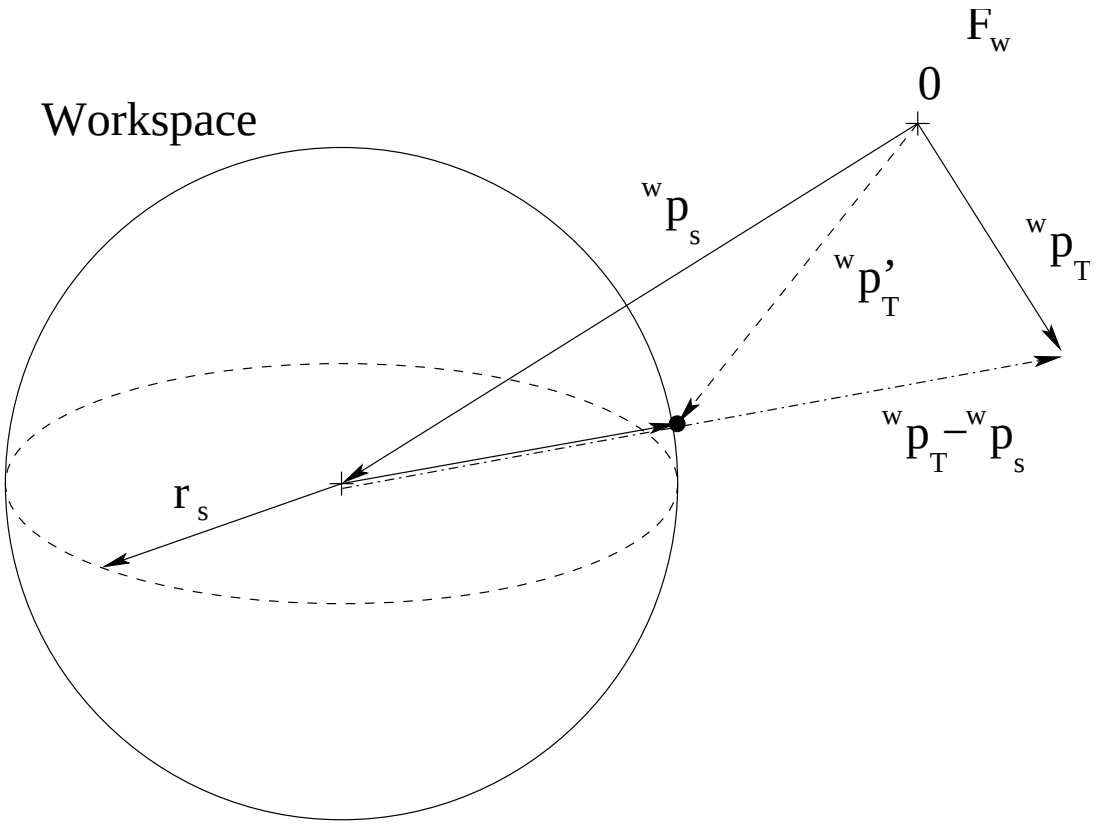


Figure 4.21: Workspace and correction of a target point

4.6.3 Discussion

Finally, the information gathered in the preceding sections is summarized here and presented in a form that allows the control of the end-effector of a redundant manipulator by simple input of target points (see Figure 4.22).

This feedback structure, having a time base Δt, contains the following elements:

1. **Trajectory planning**
   The trajectory planning is a superimposed system (in our case the "intermediate

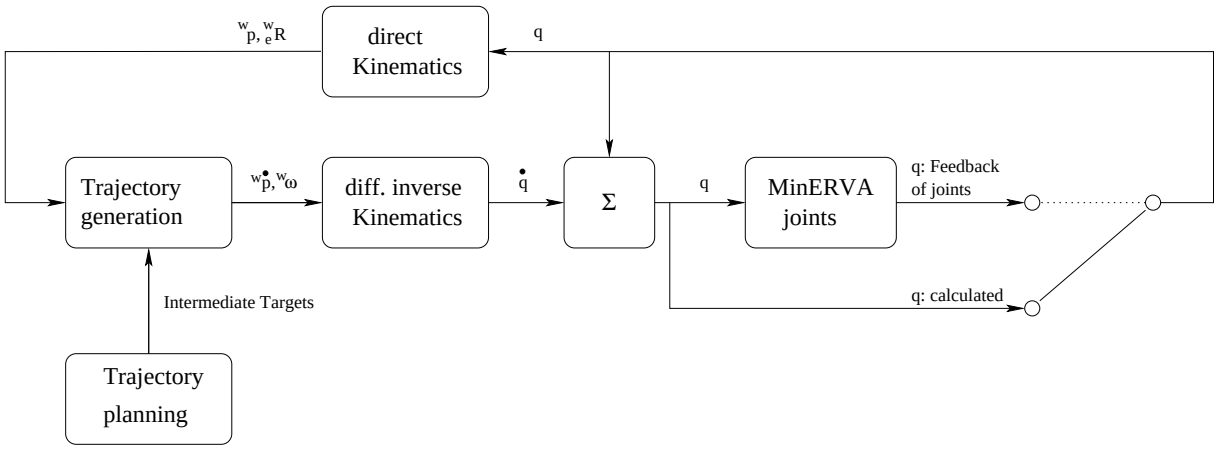


Figure 4.22: Overview for the calculation of joint angles from given target coordinates

target generator", see Section 4.7) which generates intermediate targets which shall be reached by the manipulator. The generated intermediate targets (position and orientation) are forwarded to the trajectory generation and form the input to the system.

2. **Trajectory generation**
   The trajectory generation is responsible for the following:
   - Acceptance of new intermediate targets from the trajectory planning module and correction of intermediate targets if necessary (see also Section 4.6.2.3).
   - Calculation of translatory and rotatory velocities for the inverse differential kinematics from the intermediate targets, the time to reach the targets, the exactness specifications and the feedback values (current position and orientation). Therefore equations Equation 4.25 et seqq. and Equation 4.56 et seqq. are used.
   - Management of all targets. Removal of targets, if
     - the target point is reached with the demanded exactness or
     - the time $T + \Delta T$ for the movement is over or
     - the exponential term in Equation 4.25 resp. Equation 4.56 is below a certain threshold and does not add to the whole movement anymore.

3. **Inverse differential kinematics**
   The inverse differential kinematics receives as input the Cartesian velocities and calculates the necessary angular velocities (see also Section 3.1.1). In the criteria for joint limit avoidance and singularity avoidance appropriate experimentally determined values for the weights k_i are used.

4. **Integration**
   The integration is a summation in a numerical sense and is used here to calculate

new joint angles $\mathbf{q}(t + \Delta t)$ from the current joint angles $\mathbf{q}(t)$ and the joint velocities $\dot{\mathbf{q}}(t)$. It is also checked that the joint velocities stay in predefined limits, i.e. it is checked that the real joints can be driven with these values. Additionally, the new calculated joint angles are checked on their validity.

5. **Transmission of the calculated joint values to** MINERVA
   The calculated joint angles are transmitted to the joint controllers.

6. **Feedback, direct kinematics**
   By the direct kinematics the calculated **or** measured joint angles are transformed into the appropriate position and orientation information and then transmitted to the trajectory generation. The measurement of the real joint angles is not necessary for every time step Δt (see also [Hau99] or [Mai00]).

4.7 Hand-Target Interaction Point Determination and Intermediate Target Calculation for Interaction with a Dynamic Target

Up to now it is still unspecified *when* and *where* the interaction of the end-effector and the target object shall take place. Additionally, the way or trajectory of the moving end-effector towards the target object needs to be determined, or i.e. the via-points need to be specified depending on the target trajectory.

To answer these questions a transfer of the knowledge about human interceptive movements that was gained in Sec. 2.2 shall take place.

4.7.1 Open Questions and Hypotheses

Analysing the results of the experiments about Hand-Target interaction for human catching movements in Sec. 2.2.3 still some questions are unanswered that are necessary to generate a model for a robot to perform reach-to-catch movements.

- Which movement initiation strategy shall be chosen?
- *Which kind of movement online control shall be selected? Reactive or predictive?*
- How long shall a sub-movement duration be chosen?
- How long shall a intersubmovement interval be chosen?
- *What speed shall the end-effector have at the time of contact with the object?*
- At which slope of the tangential velocity profile shall the interaction take place? At the ascending or at the descending slope?
- Where are the intermediate targets of the end-effector movement, i.e. how large is the sub-movement amplitude?
- Where is the final target of the whole end-effector movement?
- *Where is the interaction point in the workspace of the manipulator?*

Naturally, these are not all questions that can be asked. But they build a good starting point.

In a human catching movement the motion is divided into two phases: the reaching phase and the grasping phase. Here these phases are specified in more detail to obtain a model consisting of four phases.

Additionally, as in the human catching experiments, the area where the hand-target interaction shall occur, i.e. the grasping or interaction area is specified beforehand.

Overview In order to grasp a continuously moving object it has to be decided *where* and *when* the contact of the manipulator with the object shall take place. This is called *Hand-Target Interaction Point Determination* within this context. Since the interaction point is a point in space and time which lies in the future, it is determined from the predictions of the object position. In reaction to this determination the robot has to move its end-effector to the interaction point on time. In most cases the interaction point is changing with time, i.e. it is periodically updated integrating current knowledge of object tracking and prediction.

The choice of the position of the possible interaction point depends on the current motion phase of the robot arm, but the interaction point has always to lie inside a pre-defined *catching area*. Four different motion phases are distinguished:

1. an *approach phase* (Section 4.7.2.1) where the end-effector shall get close to the object to catch,
2. an *adaption phase* (Section 4.7.2.2), where the trajectories of end-effector and the target object shall get close to "match",
3. a *contact phase* (Section 4.7.2.3) where the target object shall be contacted, and
4. a *follow phase* (Section 4.7.2.4) where the target object shall be smoothly grasped.

For targets moving at a moderate speed all motion phases will be activated sequentially. As a result the arm motion will not be a single movement but an overlay of separate sub-movements. Therefore appropriate *intermediate targets* for the manipulator have to be calculated.

Predictions Predictions are generated in regular time intervals from the prediction module (see Figure 4.1). These predictions are points in space and time that will most probably be reached by the moving object. For every prediction the coordinates (x_p, y_p, z_p)[15] of the future object position are calculated as well as the duration T_p that the object needs, to travel from its current position to the predicted position. Two distinct predictions are essential for the described model: the prediction with $T_p = 2T_s$, which triggers the transition from the approach phase to the adaption phase, when the prediction appears in the catching area, and the prediction with $T_p = 3/2T_s$, which triggers the transition from the adaption phase to the contact phase, when the prediction is chosen as the final interaction

[15]We restrict the consideration on the position here. Certainly the same applies for the orientation coordinates, approach vector $\mathbf{a}$ and normal vector $\mathbf{n}$

point. Thereby T_s is the duration of a sub-movement as described in Section 4.6.1 and Section 4.6.1, respectively.

Every time when a new intermediate target shall be sent to the motion control unit, one prediction out of the current prediction-set is chosen as a possible *Interaction Point* with coordinates (XIP, YIP, ZIP). The time which is left for the arm to reach this target (interaction point) is called "Time-to-contact"or TtC. The *interaction point* is used in all phases of arm motion in a different way to calculate the intermediate targets.

Determination of optimal catching position The determination of the target interaction point on the future trajectory of the moving object must be restricted to a point that lies inside the workspace of the manipulator. But even this does not guarantee that a chosen point is suitable for catching. To keep the situation simple a user-defined area is selected as the *catching area*, where it is possible to grasp (or catch, respectively) the object with the manipulator. The catching area can have any three dimensional geometric form (e.g. a sphere), but must lie completely inside the workspace of the manipulator. One point within this area, where it is *optimal* to grasp the object e.g. the point with maximum manipulability of the manipulator, is defined as the "Optimal Point" with the coordinates (XOP, YOP, ZOP). The closer a prediction lies relative to the "Optimal Point" the better it is to grasp (catch) the object there.

4.7.2 Four Phase Model of Hand Motion towards a Moving Target

4.7.2.1 Approach Phase

The approach phase starts when the first prediction appears in the catching area. The goal of this phase is to reduce continuously the distance between the manipulator and the first possible hand-target interaction point.

Determination of possible Hand Target Interaction Points In the approach phase the manipulator targets an possible[16] interaction point that is located at the border of the catching area. Here the question may arise, why not the prediction that lies closest to the "Optimal Point" at this time is chosen as the possible interaction point? The reason for this is that at the end of this phase the manipulator shall be *close* to the target object when the object enters the catching area, but still have enough time and space to grasp it smoothly.

To illustrate the procedure of *Target Interaction Point Determination* during the approach phase see Figure 4.23. At the beginning there is no prediction inside the catch area (Fig-

[16]The point is termed as possible here, since in succeeding phases the interaction point will be changed to a point that is more optimal to grasp

ure 4.23a), that means that there is no Interaction Point to select and therefore no arm motion. When the first predictions are inside the catch area (Figure 4.23b), the one closest to the border is chosen (the black colored asterics surrounded by the small dashed circle in Figure 4.23b) as the possible Interaction Point. Each time when new predictions arrive or are queried from the prediction module a new calculation of the Interaction Point takes place, which usually changes the Interaction Point (Figure 4.23c and Figure 4.23d), but only inside a limited area (indicated by the small dashed circle)[17].

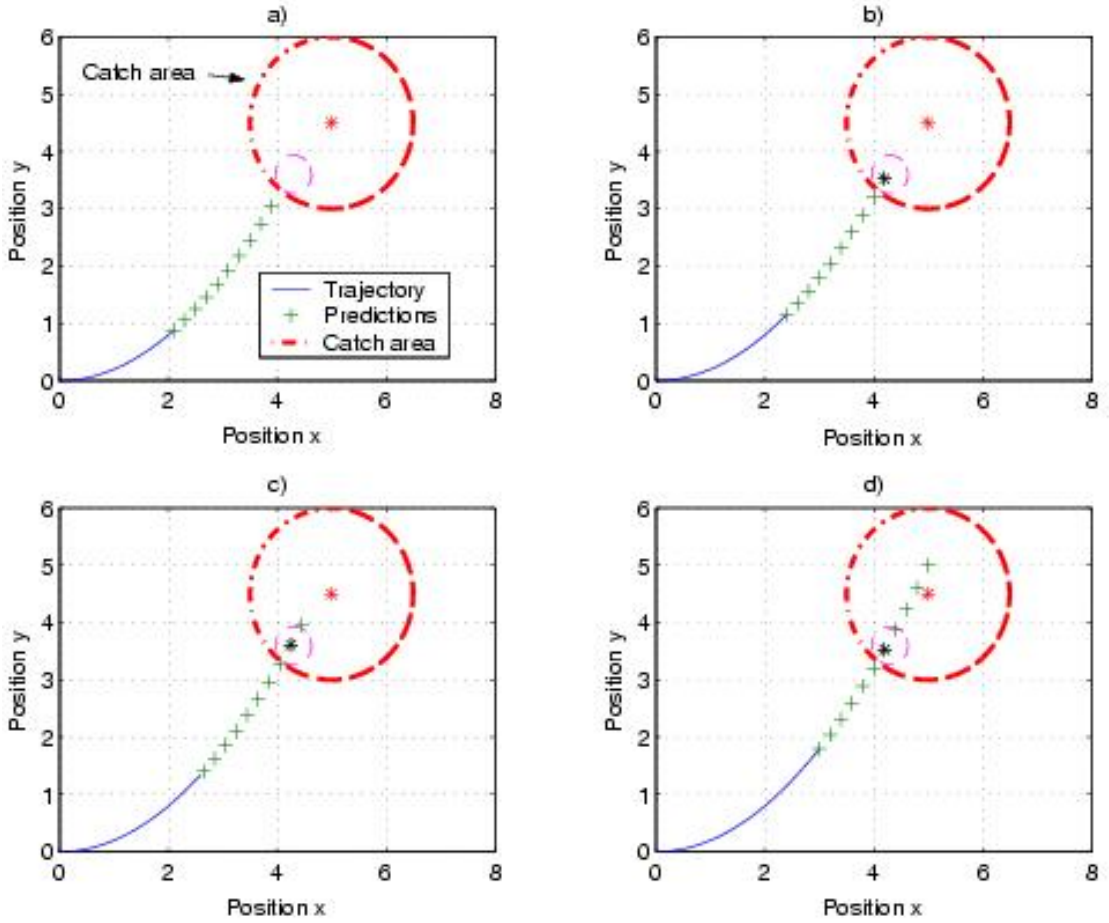


Figure 4.23: In (a) no possible interaction point exists, because there is no prediction in the catching area. In (b), (c) and (d) the Interaction Point is indicated by the black asterics, which is the prediction that is inside the catching area and closest to the border of the area.

Calculation of appropriate intermediate targets When a target Interaction Point has been chosen, the robot arm has to move. Similar to the human behavior

[17]This assumption is reasonable if one assumes that the target object makes no sudden changes in direction or acceleration that cannot be foreseen. But still the predictions underlie some jitter.

regularly every $T_s/2$ a new sub-movement shall be started for which a intermediate target has to be calculated. Given the current position of the end-effector $(xeepos, yeepos, zeepos)$, the duration of a sub-movement T_s and the data of the possible target interaction point (XIP, YIP, ZIP, TtC) the intermediate target for the manipulator $(xeedes, yeedes, zeedes)$ can be calculated. Thereby the idea, illustrated in Figure 4.24, is that if the end-effector shall cover the distance $XIP - xeepos$ in TtC it has to cover the distance $xeedes - xeepos$ with one sub-movement or within T_s:

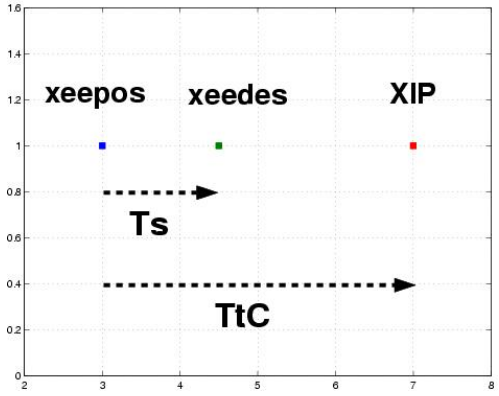


Figure 4.24: The manipulator shall reach XIP in TtC. Therefore it shall reach $xeedes$ in T_s

$$\frac{XIP - xeepos}{TtC} = \frac{xeedes - xeepos}{T_s} \tag{4.60}$$

Solved for the desired unknown:

$$xeedes = (XIP - xeepos)\frac{T_s}{TtC} + xeepos \tag{4.61}$$

4.7.2.2 Adaption Phase

The adaption phase starts when the prediction with a $TtC = 2T_s$ appears in the catching area. The goal of this phase is to approximate the trajectory of the target object by the manipulator trajectory. This phase can be devided in two separate phases:

Determination of Possible Hand Target Interaction Points

- Sub-phase a
  As mentioned the adaption phase starts when the prediction with a $T_p = TtC = 2T_s$ appears in the catching area. $2T_s$ was chosen, to leave the manipulator enough time

to approach the target object without contacting it. After each new prediction query, the predicted position with a $T_p = TtC = 2T_s$ is chosen as the Interaction Point. During this sub-phase the TtC stays constant on $TtC = 2T_s$. Figure 4.25a and Figure 4.25b show the continuous displacement of the Interaction Point, whereby in this example the black asterics, indicating the Interaction Point, is always the fifth prediction.

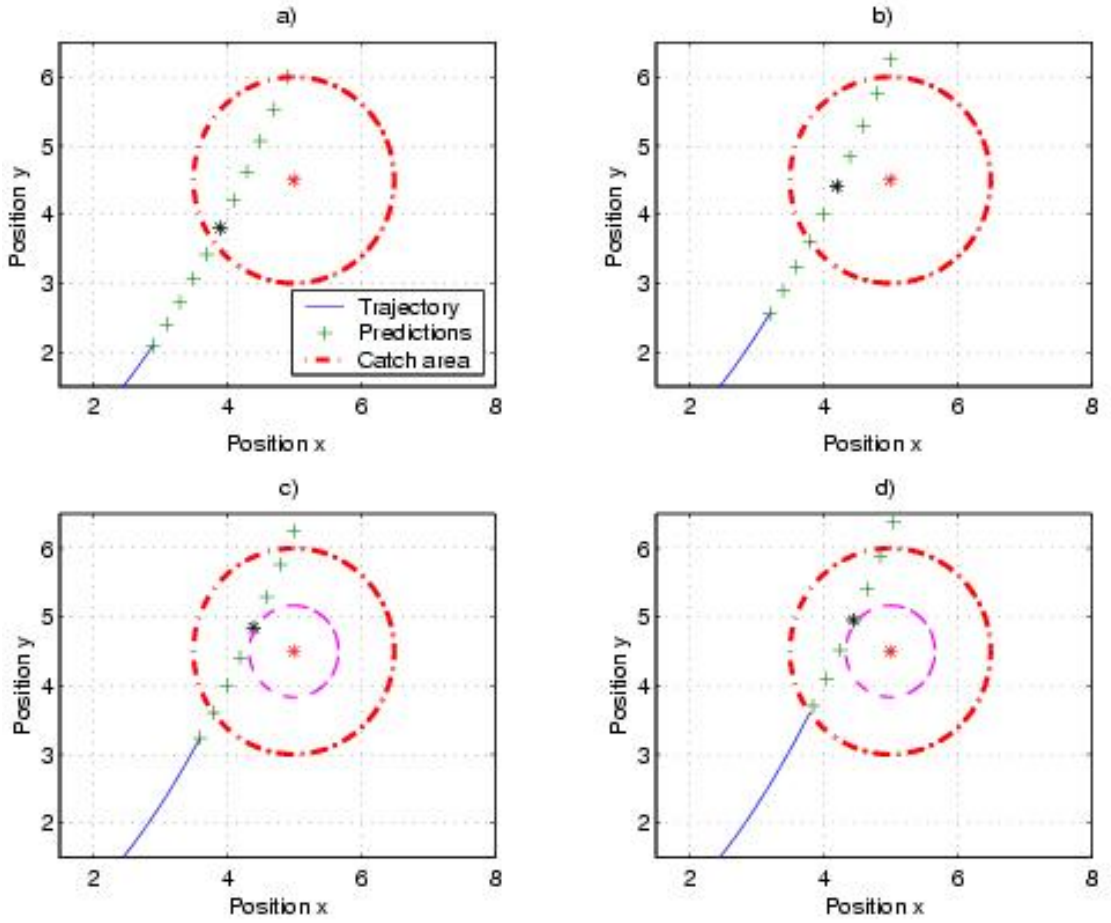


Figure 4.25: a and b: the Interaction Point is the predicted position with a $TtC = 2T_s$. c and d: the Interaction Point is the prediction that lies closest to the "Optimal Point".

- Sub-phase *b*
  After each new prediction query the prediction with a $T_p = 2T_s$ gets closer to the "Optimal Point" and finally gets the prediction which is closest to the "Optimal Point". In this region the target object should ideally be grasped. If the above mentioned condition is fulfilled the displacement of the Interaction Point is stopped. Now the prediction which is *closest* to the "Optimal Point" is chosen as the Interaction Point. In Figure 4.25c and Figure 4.25d the fourth and the third prediction, respectively, are the possible Interaction Points. Similar to the situation in the approach

phase these Interaction Points vary after each new prediction query, but normally stay within a limited region.

Calculation of appropriate intermediate targets

- Sub-phase a
  The intermediate targets $(xeedes, yeedes, zeedes)$ are calculated the same way as in the approach phase (see Equation 4.61). Through the *continuous displacement* of the possible interaction point the trajectory of the manipulator approximates the target object's trajectory asymptotically.

- Sub-phase b
  The intermediate targets $(xeedes, yeedes, zeedes)$ are again calculated the same way as in the approach phase (see Equation 4.61). Through the *positional stability* of the Interaction Point the manipulator moves in a similar way as in the approach phase.

4.7.2.3 Contact Phase

The contact phase starts when the prediction that is closest to the "Optimal Point" has a $T_p = TtC \approx 3/2T_s$.

Determination of possible Hand Target Interaction Points At this point of time the manipulator is expected to be very close to the target object and the "Optimal Point" and the trajectories of target and manipulator almost coincide. These conditions are convenient preliminaries to finally catch the target object. As already mentioned above, the contact phase starts when the prediction that is closest to the "Optimal Point" has a $T_p = TtC \approx 3/2T_s$. In Figure 4.26 this point in time is defined as T_1. At this time the final Interaction Point is fixed and the final velocity $(XVIP, YVIP, ZVIP)$ of the target object at the contact point is predicted. It is determined that the end-effector shall contact (catch) the object in $TtC = 3/2T_s$ in (XIP, YIP, ZIP) having the velocity $(XVIP, YVIP, ZVIP)$. Fulfilling this condition it is guaranteed that the grasping of the target is smooth.

Calculation of appropriate intermediate targets At time t_{n-1} the last predictions are stored and no more predictions are queried. Before the contact three sub-movements are started, namely the sub-movements $n-1$, n and $n+1$, of which only the first one $(n-1)$ will be finished before the contact. Each beginning of a sub-movement is called t_i and the corresponding intermediate target (X_i, Y_i, Z_i). The last three sub-movements must be executed to fulfill the final conditions, namely to *match the target position and velocity* in the final Interaction Point.

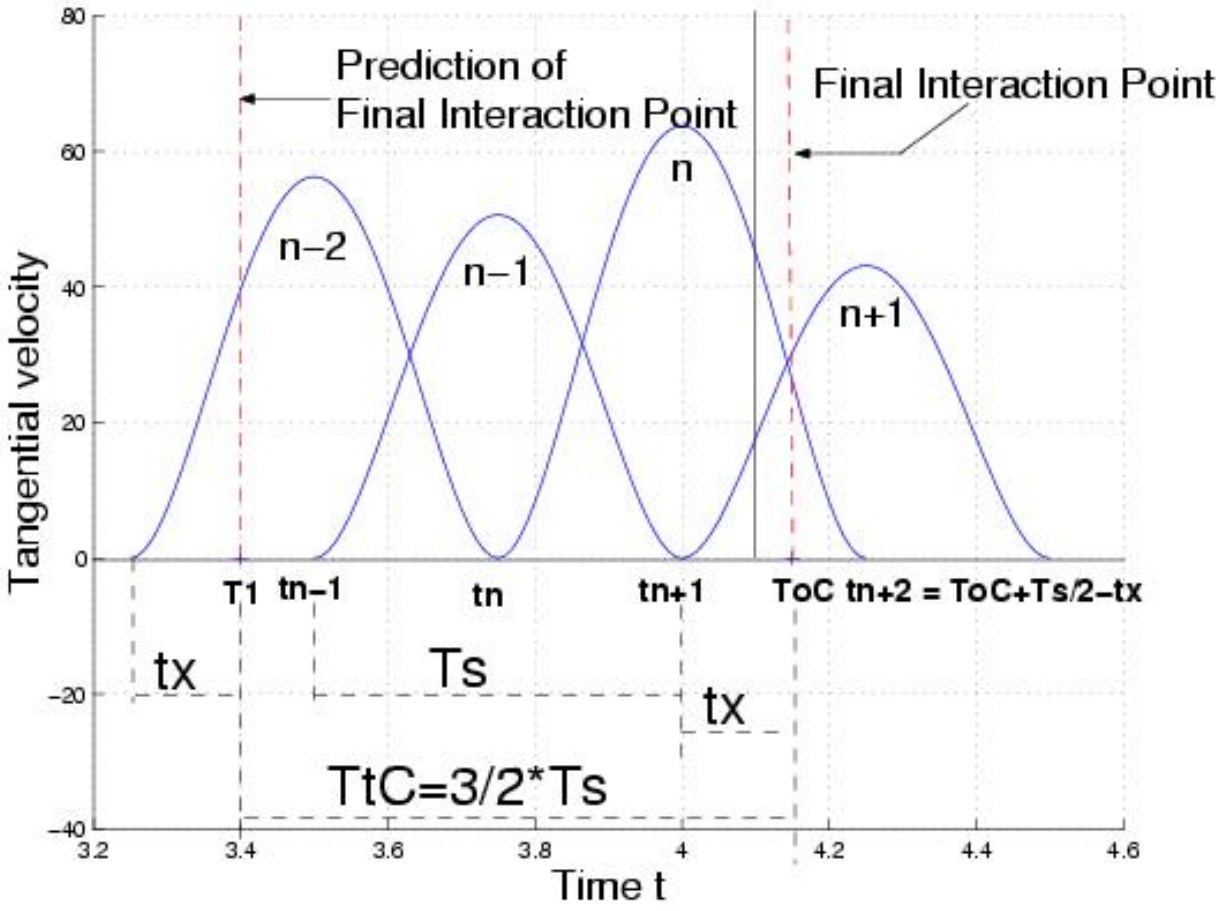


Figure 4.26: At T_1 the prediction with $T = 3/2T_s$ is chosen as the final interaction point. At this point in time it is decided to contact the target object at $ToC = t_{n+1} + tx$, whereby t_i are the starting times of the sub-movements i.

The difference of the generated trajectories of the "Model of Flash" (Section 4.6.1) and the "Model of Goodman" (Section 4.6.1) are negligible. One can assume that the Model of Flash approximates the trajectory of the robot arm sufficiently accurate. Due to the simplicity of Flash's equations they are used to calculate the intermediate targets for the contact phase.

In the equations below only the x-dimension is shown. The calculations for y and z are analog.

In general for the position of the end-effector according to Flash's model is true:

$$\begin{aligned}
\text{Sub-movement } n-1 &: \forall t \in [t_{n-1}; t_{n+1}] \\
x_{n-1}(t) &= (X_{n-1} - X_{n-2})(10\tau_{n-1}^3 - 15\tau_{n-1}^4 + 6\tau_{n-1}^5) \qquad (4.62) \\
\text{Sub-movement } n &: \forall t \in [t_n; t_{n+2}] \\
x_n(t) &= (X_n - X_{n-1})(10\tau_n^3 - 15\tau_n^4 + 6\tau_n^5) \qquad (4.63) \\
\text{Sub-movement } n+1 &: \forall t \in [t_{n+1}; t_{n+3}] \\
x_{n+1}(t) &= (X_{n+1} - X_n)(10\tau_{n+1}^3 - 15\tau_{n+1}^4 + 6\tau_{n+1}^5) \qquad (4.64)
\end{aligned}$$

with $\tau_i = \frac{t-t_i}{t_{f_i}-t_i}$, t_i being the beginning and t_{f_i} the end of sub-movement i. Here $t_{f_i} = ti+T_s$.
Respectively, for the velocity of the manipulator:

$$\begin{aligned}
\text{Sub-movement } n-1 &: \forall t \in [t_{n-1}; t_{n+1}] \\
\dot{x}_{n-1}(t) &= (X_{n-1} - X_{n-2})(\frac{30}{T_s}\tau_{n-1}^2 - \frac{60}{T_s}\tau_{n-1}^3 + \frac{30}{T_s}\tau_{n-1}^4) \qquad (4.65) \\
\text{Sub-movement } n &: \forall t \in [t_n; t_{n+2}] \\
\dot{x}_n(t) &= (X_n - X_{n-1})(\frac{30}{T_s}\tau_n^2 - \frac{60}{T_s}\tau_n^3 + \frac{30}{T_s}\tau_n^4) \qquad (4.66) \\
\text{Sub-movement } n+1 &: \forall t \in [t_{n+1}; t_{n+3}] \\
\dot{x}_{n+1}(t) &= (X_{n+1} - X_n)(\frac{30}{T_s}\tau_{n+1}^2 - \frac{60}{T_s}\tau_{n+1}^3 + \frac{30}{T_s}\tau_{n+1}^4) \qquad (4.67)
\end{aligned}$$

In general the prediction of the final Interaction Point (contact point) occurs *asynchronously* to the manipulator sub-movement starting times. The time between the start of the $(n-2)$nd sub-movement and T_1, the point in time at which $T_p = TtC = 3/2Ts$ for the prediction that is closest to the "Optimal Point", is called tx (see Figure 4.27). Depending on t_x it is decided which succeeding sub-movements belong to the contact phase or the follow phase, respectively.

Two different cases are distinguished:
either $0 \leq tx < T_s/4$ (Case 1) or $T_s/4 \leq tx < T_s/2$ (Case 2).

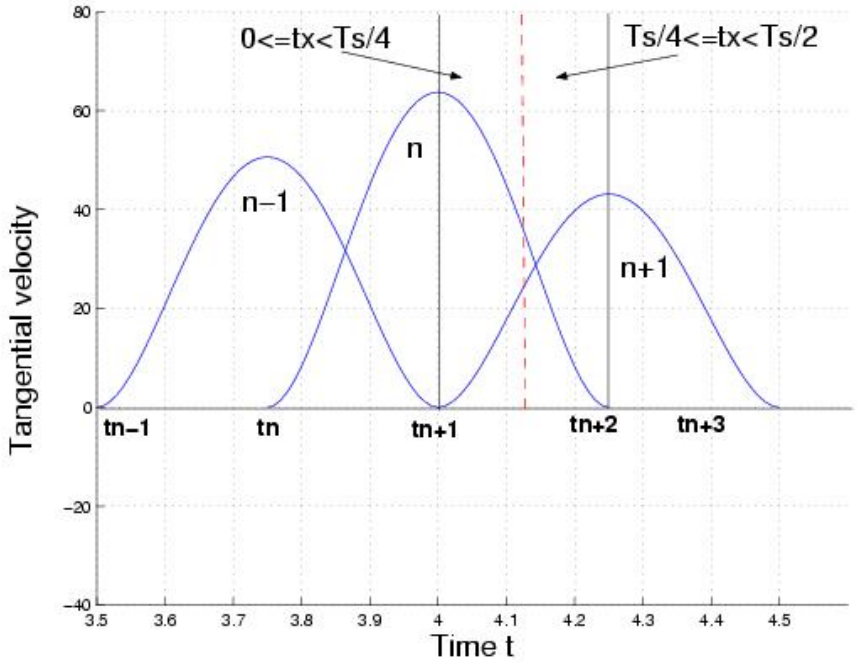


Figure 4.27: If $0 \leq tx < T_s/4$ (left side of the dotted line), case 1 is chosen and the sub-movements $n-1$ and n define the contact phase. If $T_s/4 \leq tx < T_s/2$ (right side of the dotted line), case 2 is chosen and the sub-movements $n-1$, n and $n+1$ define the contact phase.

In both cases the following equations are true:

$$\begin{aligned} \forall t \in [t_{n+1}; t_{n+2}] & \\ x(t) &= X_{n-1} + x_n(t) + x_{n+1}(t) \\ &= X_{n-1} + \\ &\quad (X_n - X_{n-1})(10\tau_n^3 - 15\tau_n^4 + 6\tau_n^5) + \\ &\quad (X_{n+1} - X_n)(10\tau_{n+1}^3 - 15\tau_{n+1}^4 + 6\tau_{n+1}^5) \end{aligned} \tag{4.68}$$

$$\begin{aligned} \dot{x}(t) &= \dot{x}_n(t) + \dot{x}_{n+1}(t) \\ &= (X_n - X_{n-1})(\frac{30}{T_s}\tau_n^2 - \frac{60}{T_s}\tau_n^3 + \frac{30}{T_s}\tau_n^4) + \\ &\quad (X_{n+1} - X_n)(\frac{30}{T_s}\tau_{n+1}^2 - \frac{60}{T_s}\tau_{n+1}^3 + \frac{30}{T_s}\tau_{n+1}^4) \end{aligned} \tag{4.69}$$

- Case 1: $0 \leq tx < T_s/4$
  In the first case $x_{n+1}(t \geq t_{n+1})$ and $\dot{x}_{n+1}(t \geq t_{n+1})$ are small at the calculated contact point[18] and therefore negligible compared to $x_n(t \geq t_n)$ and $\dot{x}_n(t \geq t_n)$ (see

[18]For the case that t_x reaches close to $T_s/4$ this condition does not hold and results in a velocity mismatch at the contact point

Figure 4.26). Therefore, it is sufficient to calculate X_{n-1} and X_n to achieve the contact conditions. The equations for position and velocity of the end-effector are:

$$\begin{aligned} \forall t \in [t_{n+1}; t_{n+2}] & \\ x(t) &= X_{n-1} + x_n(t) \\ &= X_{n-1} + (X_n - X_{n-1})(10\tau_n^3 - 15\tau_n^4 + 6\tau_n^5) \end{aligned} \tag{4.70}$$

$$\begin{aligned} \dot{x}(t) &= \dot{x}_n(t) \\ &= (X_n - X_{n-1})(\frac{30}{T_s}\tau_n^2 - \frac{60}{T_s}\tau_n^3 + \frac{30}{T_s}\tau_n^4) \end{aligned} \tag{4.71}$$

At t_{n-1}, X_{n-1} and X_n shall be calculated:

$$x(t_{n+1} + tx) = XIP \tag{4.72}$$

$$\dot{x}(t_{n+1} + tx) = XVIP \tag{4.73}$$

Defining:

$$A = 10\tau_n^3 - 15\tau_n^4 + 6\tau_n^5 \tag{4.74}$$

$$\dot{A} = \frac{30}{T_s}\tau_n^2 - \frac{60}{T_s}\tau_n^3 + \frac{30}{T_s}\tau_n^4 \tag{4.75}$$

results in:

$$X_{n-1} + (X_n - X_{n-1})A = XIP \tag{4.76}$$

$$(X_n - X_{n-1})\dot{A} = XVIP \tag{4.77}$$

summarized:

$$\begin{pmatrix} A & 1-A \\ \dot{A} & -\dot{A} \end{pmatrix} \begin{pmatrix} X_n \\ X_{n-1} \end{pmatrix} = \begin{pmatrix} XIP \\ XVIP \end{pmatrix} \tag{4.78}$$

$$\begin{pmatrix} X_n \\ X_{n-1} \end{pmatrix} = \frac{1}{-\dot{A}} \begin{pmatrix} -\dot{A} & -1+A \\ -\dot{A} & A \end{pmatrix} \begin{pmatrix} XIP \\ XVIP \end{pmatrix} \tag{4.79}$$

$$X_{n-1} = XIP - \frac{AXVIP}{\dot{A}} \tag{4.80}$$

$$X_n = XIP + \frac{(1-A)XVIP}{\dot{A}} \tag{4.81}$$

These calculations are executed at time t_{n-1}. The *contact phase* consists of the intermediate targets X_{n-1} and X_n.

- Case 2: $T_s/4 \leq tx < T_s/2$
  In the second case the influence of the second sub-movement $(n+1)$ is not negligible any more. Therefore the contact phase consists of the intermediate targets X_{n-1}, X_n and X_{n+1} which have to be calculated in a way to fulfill the catching conditions, i.e matching position and velocity in the contact point.

  To achieve this calculation, one *assumes* that the manipulator will grasp the target object at time $t = t_{n+1}$ (see Figure 4.26). The Interaction Point (XIP, YIP, ZIP) is represented at time t_{n-1} by the prediction with $T_p = TtC = 3/2T_s$. Now intermediate targets X_{n-1} and X_n are calculated to fulfill the conditions of Equation 4.70 and Equation 4.71 for point of time t_{n+1}.

  One obtains Equation 4.81 and Equation 4.80 with $tx = 0$ and $\tau_n = \frac{t_{n+1}-t_n}{t_{n+2}-t_n} = 1/2$. But only the intermediate target X_{n-1} is sent to the motion control unit. X_n and X_{n+1} are calculated according to equations Equation 4.93 and Equation 4.94, which will be derived in the following:

$$X_{n-1} = XIP - \frac{AXVIP}{\dot{A}} \tag{4.82}$$

  Now intermediate targets X_n and X_{n+1} are calculated to obtain the correct final contact conditions at point of time ToC:

$$\begin{aligned} \forall t \in [t_{n+1}; t_{n+2}] & \\ x(t) &= X_{n-1} + x_n(t) + x_{n+1}(t) \\ &= X_{n-1} + (X_n - X_{n-1})(10\tau_n^3 - 15\tau_n^4 + 6\tau_n^5) + \\ &\quad (X_{n+1} - X_n)(10\tau_{n+1}^3 - 15\tau_{n+1}^4 + 6\tau_{n+1}^5) \end{aligned} \tag{4.83}$$

$$\begin{aligned} \dot{x}(t) &= \dot{x}_n(t) + \dot{x}_{n+1}(t) \\ &= (X_n - X_{n-1})(\frac{30}{T_s}\tau_n^2 - \frac{60}{T_s}\tau_n^3 + \frac{30}{T_s}\tau_n^4) + \\ &\quad (X_{n+1} - X_n)(\frac{30}{T_s}\tau_{n+1}^2 - \frac{60}{T_s}\tau_{n+1}^3 + \frac{30}{T_s}\tau_{n+1}^4) \end{aligned} \tag{4.84}$$

  Defining:

$$\begin{aligned} A &= 10\tau_{n+1}^3 - 15\tau_{n+1}^4 + 6\tau_{n+1}^5 \\ \dot{A} &= \frac{30}{T_s}\tau_{n+1}^2 - \frac{60}{T_s}\tau_{n+1}^3 + \frac{30}{T_s}\tau_{n+1}^4 \\ B &= (10\tau_n^3 - 15\tau_n^4 + 6\tau_n^5) - (10\tau_{n+1}^3 - 15\tau_{n+1}^4 + 6\tau_{n+1}^5) \\ \dot{B} &= (\frac{30}{T_s}\tau_n^2 - \frac{60}{T_s}\tau_n^3 + \frac{30}{T_s}\tau_n^4) - (\frac{30}{T_s}\tau_{n+1}^2 - \frac{60}{T_s}\tau_{n+1}^3 + \frac{30}{T_s}\tau_{n+1}^4) \qquad (4.85) \\ C &= 10\tau_n^3 - 15\tau_n^4 + 6\tau_n^5 - 1 \qquad (4.86) \\ \dot{C} &= \frac{30}{T_s}\tau_n^2 - \frac{60}{T_s}\tau_n^3 + \frac{30}{T_s}\tau_n^4 \qquad (4.87) \end{aligned}$$

results in:

$$AX_{n+1} + BX_n = XIP + CX_{n-1} \tag{4.88}$$
$$\dot{A}X_{n+1} + \dot{B}X_n = XVIP + \dot{C}X_{n-1} \tag{4.89}$$

summarized:

$$\begin{pmatrix} A & B \\ \dot{A} & \dot{B} \end{pmatrix} \begin{pmatrix} X_{n+1} \\ X_n \end{pmatrix} = \begin{pmatrix} XIP + CX_{n-1} \\ XVIP + \dot{C}X_{n-1} \end{pmatrix} \tag{4.90}$$

$$\begin{pmatrix} X_{n+1} \\ X_n \end{pmatrix} = \frac{1}{A\dot{B} - \dot{A}B} \begin{pmatrix} \dot{B} & -B \\ -\dot{A} & A \end{pmatrix} \begin{pmatrix} XIP + CX_{n-1} \\ XVIP + \dot{C}X_{n-1} \end{pmatrix} \tag{4.91}$$

$$\tag{4.92}$$

Position of the end-effector:

$$X_n = \frac{-\dot{A}(XIP + CX_{n-1}) + A(XVIP + \dot{C}X_{n-1})}{A\dot{B} - \dot{A}B} \tag{4.93}$$
$$X_{n+1} = \frac{\dot{B}(XIP + CX_{n-1}) - B(XVIP + \dot{C}X_{n-1})}{A\dot{B} - \dot{A}B} \tag{4.94}$$

4.7.2.4 Follow Phase

At point of time ToC the end-effector has contacted the target object. But to smoothly grasp the object it is necessary to follow the target for the amount of time it takes to close the gripper.

Determination of possible Hand Target Interaction Points In the ideal case end-effector and the target object still have no physical contact at this time. The goal is now that the end-effector follows the object on its trajectory long enough to close the gripper and to grasp it smoothly. It is chosen (arbitrarily) that the *position* of end-effector and target object shall coincide for e.g. all $T_s/2$, i.e. every time when sub-movements reach their peak velocity. Again depending on t_x two cases have to be distinguished for which following conditions have to be fulfilled (see Figure 4.28):

- Case 1: $x(t_{n+2}) = x_p(ToC + T_s/2 - t_x)$ for $0 \leq tx < T_s/4$
- Case 2: $x(t_{n+3}) = x_p(ToC + T_s - t_x)$ for $T_s/4 \leq tx < T_s/2$

Calculation of Appropriate Intermediate Targets

- Case 1: $0 \leq tx < T_s/4$
  In this case the sub-movements $(n-1)$ and (n) are part of the contact phase. The sub-movement $(n+1)$ is therefore the first in the follow phase. This sub-movement is calculated at point of time t_{n+1}, in a way that $xeepos - xobjpos = 0$ in t_{n+2}. (see Figure 4.28). The interaction point and time is then covered by the prediction with $T_p = ToC + T_s/2 - t_x$.

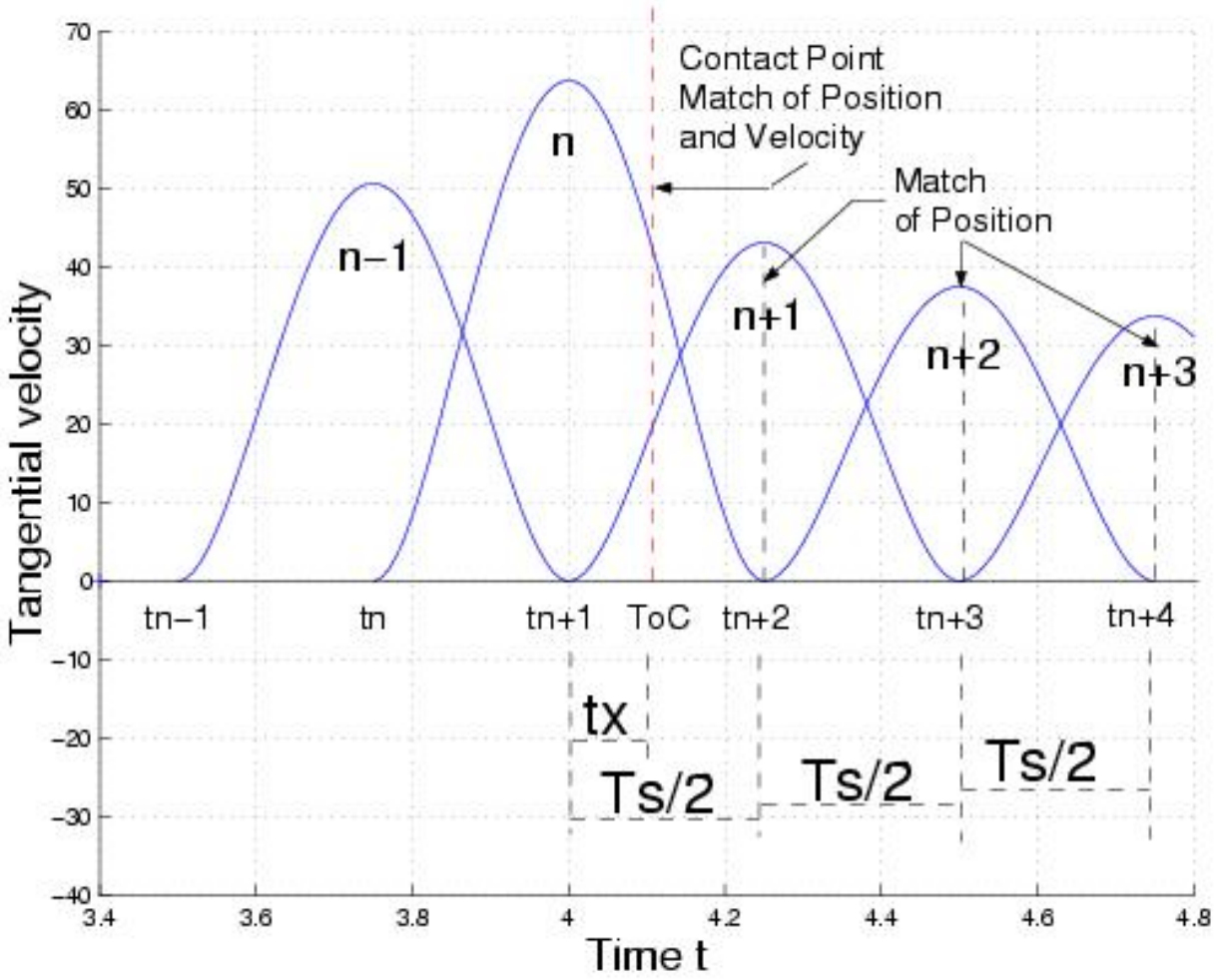


Figure 4.28: Contact point for ToC. If $0 \leq tx < T_s/4$ the follow phase starts with sub-movement $n+1$. Further contacts between end-effector and target object occur at point in time t_{n+2} and t_{n+3}. If $T_s/4 \leq tx < T_s/2$, the follow phase starts with sub-movement $n+2$. Further contact between end-effector and target object occur at point in time t_{n+3}.

Since only the position shall be matched, only the equation for the position is solved. The resulting velocity at the ("follow") contact point is not matching the object velocity, but this is negligible, since the catching conditions have been fulfilled for

ToC. With the sub-movement $(n+1)$ end-effector and target object are in contact. The duration between these events is short enough to assume that the object velocity changes only marginally.

Solving:

$$\begin{aligned} x(t_{n+2}) &= X_n + (X_{n+1} - X_n)(10\tau_{n+1}^3 - 15\tau_{n+1}^4 + 6\tau_{n+1}^5) \\ &= x_p(ToC + T_s/2 - t_x) \end{aligned} \tag{4.95}$$

results in:

$$X_{n+1} = \frac{x_p(ToC + T_s/2 - t_x) - X_n}{10\tau_{n+1}^3 - 15\tau_{n+1}^4 + 6\tau_{n+1}^5} + X_n \tag{4.96}$$

with $\tau_{n+1} = \frac{t_{n+2} - t_{n+1}}{t_{n+3} - t_{n+1}}$

These operations can be repeated every $T_s/2$ and as often as necessary. Usually, one iteration is enough.

- Case 2: $T_s/4 \leq tx < T_s/2$
  Now sub-movement $(n+2)$ is the first of the follow phase and is calculated at point in time t_{n+2}, such that $xeepos - xobjpos = 0$ in t_{n+3} (see Figure 4.28). The prediction with a $T_p = ToC + T_s - t_x$ is used for calculating the intermediate target:

$$X_{n+2} = \frac{x_p(ToC + T_s - t_x) - X_{n+1}}{10\tau_{n+2}^3 - 15\tau_{n+2}^4 + 6\tau_{n+2}^5} + X_{n+1} \tag{4.97}$$

  Afterwards there is no difference between the two cases. The target object is tracked by the manipulator as long as necessary. For that every $T_s/2$ a new intermediate target is calculated.

4.7.3 Discussion

To summarize this section it can be stated that an algorithm was developed that is capable to determine interaction points and intermediate targets for a robotic manipulator to catch a moving target in a human like way. The motivation for the algorithm was given by the analysis of experiments of Georgopulos described in Sec. 2.2.3. In the algorithm interaction points determine the points in space and time where the contact with the target object shall occur. Those interaction points are derived from the predicted target trajectory. Intermediate targets determine the points in space and time that the manipulator shall pass on its way to the interaction points. The algorithm distinguishes between four phases of manipulator motion: an *approach phase*, an *adaption phase*, an *contact phase* and an

follow phase. In each phase appropriate interaction points and intermediate targets are calculated.

In Sec. 4.7.1 open questions and hypotheses where formulated that were derived from the catching experiments with humans. Some answers can be given here: (1) As a movement initiation strategy a predictive strategy was chosen. (2) The online control of the movement is an adaption of a reactive strategy. (3) The sub-movement duration and (4) the intersubmovement interval are free chosable parameters of the algorithm. (5) The speed of the end-effector at the point of contact is the same as the target object speed. (6) The interaction with the target occurs at the descending slope of the hand velocity profile. (7) The sub-movements amplitudes are derived from the determined interaction points (points in space and time!), and therefor from the predicted targets trajectory. (8) The final interaction point is determined within a pre-defined catch area and fixated at the end of the adaption phase.

In comparison to the findings of Georgopulos experiments the algorithm obeys "robotic needs" for smoothly grasping a moving object, i.e. the adaption of the right speed at the contact with the object and the following of the object with the manipulator to give the end-effector enough time to close the gripper. Despite the fact that the algorithm consists of different motion phases the passing from phase to phase is smoothly. Additionally, no assumption was made about the targets trajectory, i.e. any kind of a predictable target trajectory can be handled by the algorithm. The only restriction is that the "time-to-contact" the target is at least 1,5 times the sub-movement duration, since at least 3 sub-movements are needed to adapt position and velocity in the contact point.

The extension of the algorithm to obeys also orientation trajectories is straightforward. The formulas can also be used for the components of the normal and approach vector. The only problem thereby is the determination of the 3D target objects orientation from the image processing.

4.8 Implementation

4.8.1 System Preliminaries

To realize the designed algorithms on the robot system, most of them were implemented in C++, running under Linux on a standard desktop PC equipped with a Intel Pentium III 500 MHz processor. Robot control is realized by an ISA bus card containing a microcontroller for communication and a DSP for kinematics calculations. Positioning of the robot's joints is controlled by micro-controllers (one per joint). Image processing including the interaction with the frame grabbers (one Matrox Meteor for each camera) was realized using the image analysis system *HALCON*, an extensive domain-independent software library providing low-level and medium-level image processing operators [ES97, HAL]. For acceleration of specific operators (e.g. sobel filter [Bla00], color filters [Leu00]) implementations using MDSI commands (multiple data, single instructions) were used. Graphical user interfaces were constructed using *Qt* [Qt].

Task that are running on the system can be divided into hard real-time tasks and soft real-time tasks. According to their required reaction time they were allocated on the given hardware appropriately (for an overview of system architectures for time-critical technical processes see [Bur02]).

Hard Real-Time Requirements The manipulator and the pan-tilt head is commanded by sending "messages" to the ISA bus card provided by the manufacturer; for the realization of the communication between PC and ISA card, a Linux device driver has been developed as well as a C++ class library where all robot commands are encapsulated. This class library allows to control the robot locally as well as over the network using CORBA. On the card an Analog Devices ADSP 21060 is used to calculate (inverse) kinematics and the resulting joint angles for the manipulator drives in real-time (see [Mai00]).

Soft Real-Time Requirements Since Linux is no hard real-time operating system, drawbacks had to be taken into account. Image processing tasks (*form, color, motion*), sensor fusion and integration, grasping point determination, prediction algorithms, interaction point and intermediate target calculations could not all be performed at frame rate (25Hz). Therefore, the algorithms were designed to cope with this drawback.

To realize a periodic timer the Linux Kernel signal SIGALRM was used. This timer has low jitter under normal system load, and was used to send intermediate targets to the robot control unit in time. Naturally any jitter is problematic in a robot control, especially if right timing is demanded as in a catching task. To tackle this problem following procedure could be used: the jitter is subtracted or added, respectively, from/to the movement duration of the next sub-movement. The problem occurs then only at the contact of the end-effector

with the target: by this procedure the contact velocity will not be the pre-calculated one. But this "workaround" was not necessary as the experimental validation showed (see e.g. Sec. 5.4)

4.8.2 State Automaton and Timing Charts

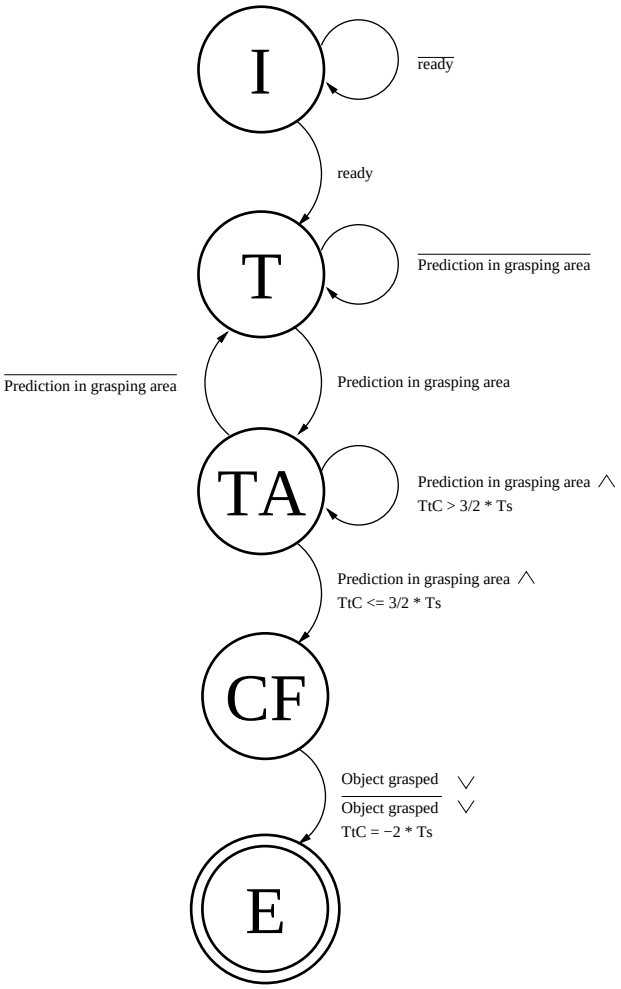


Figure 4.29: State Automaton for Catching. I: Initialization, T: Tracking, TA: Track and Approach, CF: Catch and Follow, E: Exit

To test the algorithms as "a working whole" a simple state automaton was designed and implemented in C++ according to Figure 4.29.

In the *Initialization* state the human operator mainly prepares the system for one catching trial. Thereby e.g. the tracker module has to be adjusted or the robot arm has to be moved to the starting position. To leave the state the user enters a command that all

preparations for the trial have been performed successfully (see timing chart in Figure 4.30 for more details).

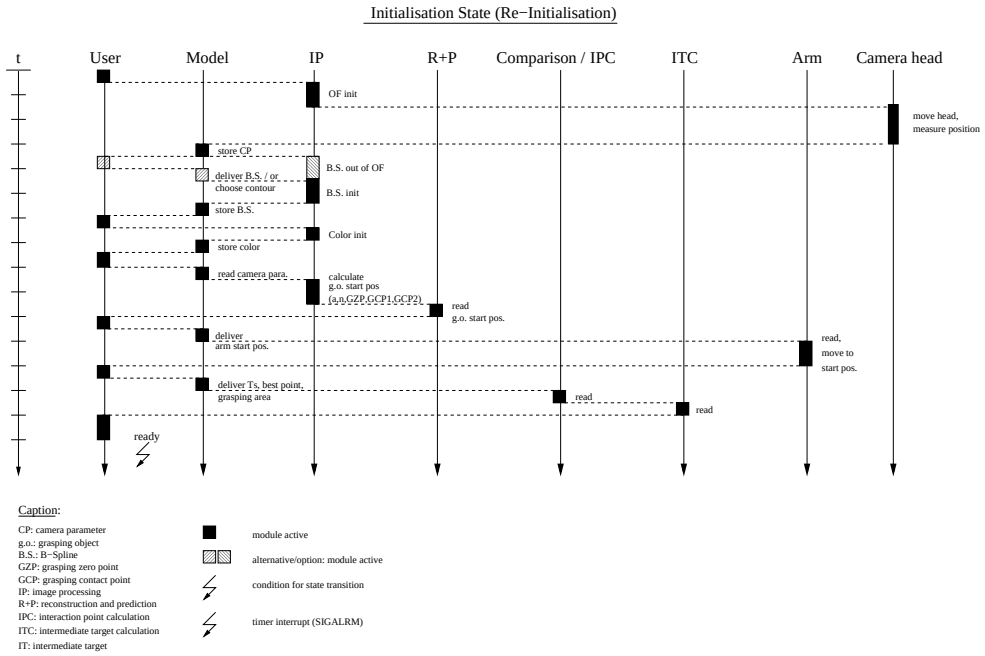


Figure 4.30: Timing chart for state 1

In the *Tracking* state the system functions autonomously. The input image is first processed for form, color and motion[19] to extract the object to be tracked. Thereafter, features of the object, e.g. its center of area are calculated. With this information predictions about the future objects positions in 2 and 3D are calculated. After each 3D prediction it is checked if the object will enter the catching area in the near future. If this is not the case the next image is processed, otherwise the next state is entered (see timing chart in Figure 4.31 for more details).

In the *Track and Approach* state tracking and arm motion occur in parallel. A system timer is started to be able to send intermediate targets to the motion control unit in regular intervals. From the predicted object position that are in the catching area interaction points are derived. From those appropriate intermediate targets are calculated. At each timer interrupt the current intermediate target is transmitted to the motion control unit until the critical time limit is reached. This time limit, $3/2T_s$, is derived from the duration of a sub-movement (see Section 4.7 and timing chart in Figure 4.32 for more details).

In the *Catch and Follow* state the system is in a "simple" control mode. No more feedback about the object's position is processed anymore. Depending on the time elapsed since the last sub-movement started, the final interaction point and all following intermediate

[19]In many experiments only color tracking was performed for simplicity and real-time performance reasons

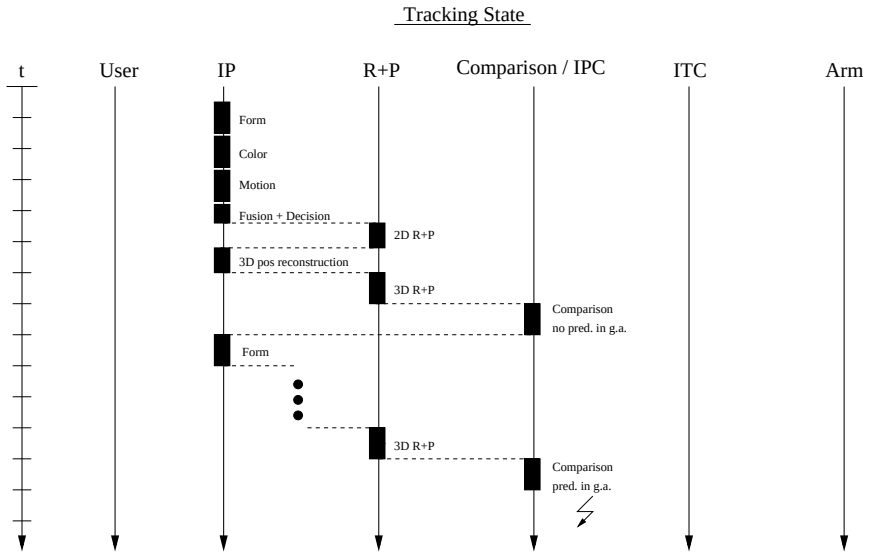


Figure 4.31: Timing chart for state 2

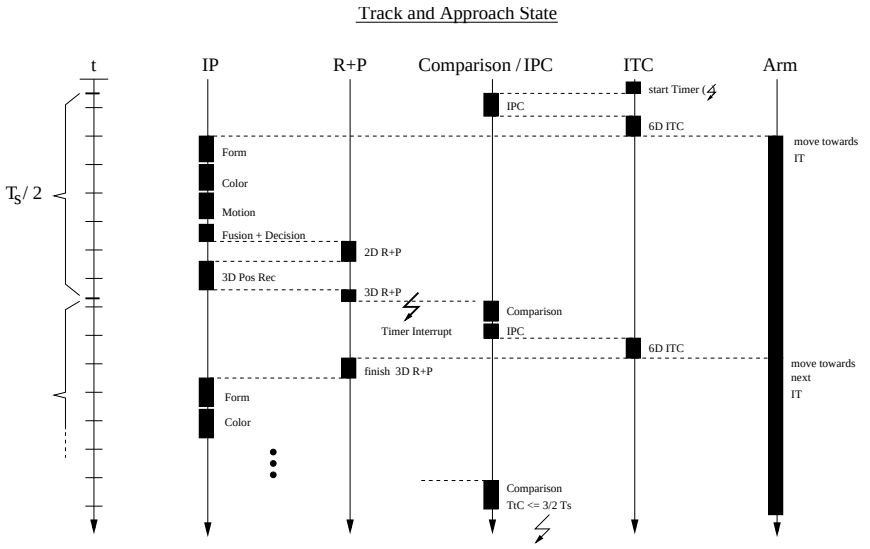


Figure 4.32: Timing chart for state 3

targets are calculated (see Section 4.7) from the last predictions made, and are appropriately transmitted to the robot control unit at the timer interrupts (see timing charts in Figure 4.33 and Figure 4.34 for more details).

Finally the *Exit* state is reached if the trial is over. This can either be if the trial was successful, i.e. the object was grasp, or a specified amount of time after the calculated grasping point in time is expired.

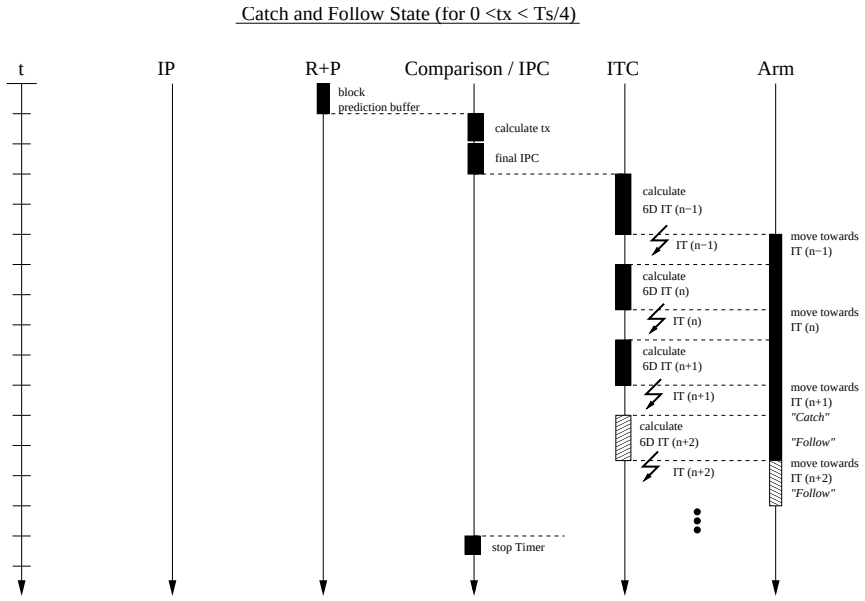


Figure 4.33: Timing chart for state 4a

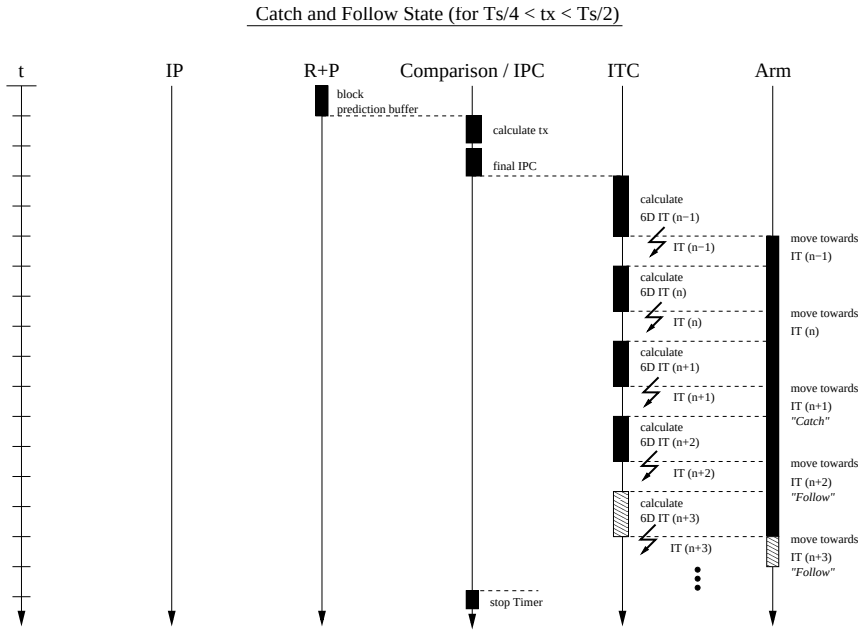


Figure 4.34: Timing chart for state 4b

Chapter 5

Simulations, Experimental Validation and Results

Overview: This chapter summarizes the results of simulations and experimental validations to proof the algorithms proposed in this thesis. To start in Sec. 5.1 tracking experiments and results with color (Sec. 5.1.1), form (Sec. 5.1.2) and motion (Sec. 5.1.3) are presented. Sec. 5.1.4 describes the results obtained by the *Modified ICondensation* algorithm as well as by the *Grasp Determination* algorithm. Also an example showing the effect of the online adaption of tracking parameters is given.

Sec. 5.1.5 summarizes the results of *biologically motivated image processing*. Hereby the *Reentry mechanism* is tested for the case that color information is entered into the form path.

In Sec. 5.2 results of the different *prediction algorithms* are presented. Thereby in Sec. 5.2.1 the ARM prediction methods as well as the nearest neighbor methods are compared in simulation. In Sec. 5.2.2 results of the Average ARM prediction algorithm used on real image sequences is presented.

In Sec. 5.3 simulation results validating the algorithms of the *Hand-Target Interaction* are shown.

Real robot experiments in Sec. 5.4 start with examples showing the control of the robots end-effector position and orientation during tracking of a moving target (Sec. 5.4.2).

Finally, Sec. 5.4.3 shows the results of real *catching experiments* for a linearly escaping and approaching target as well as for a target moving tangential to the robot. Sec. 5.4.4 shows the case of a circular approaching target.

Color Space	Subspace Representation	Time [ms][1]	subjective rating
RGB	CLUT (Cone)	7.4	good
RGB	Window (Cube)	5.3	satisfactory
YUV[2]	CLUT (Cone)	7.0	satisfactory
YUV[2]	Window (Cube) [YUV][3]	2.7	good
YUV[2]	Window (Rectangle) [UV][4]	1.2	good
RGB	DRM	7.4	very good[5]

Table 5.1: Results of implemented filters

5.1 Tracking with Color, Form and Motion

5.1.1 Color Tracking

Test conditions: Several of the previously described (see Section 3.2.1.2) CLUT and windowing techniques were tested on their performance in [Leu00]. The clustering techniques, performing well subjectively rated, always tended to be too slow to serve the needs, keeping in mind that the process of color segmentation is only a small part of the whole system. Therefore they are not considered in the following.

Color Filter Performance

Results: Table 5.1 presents an excerpt of performance measurements. The input image is either grabbed as a RGB or YUV full color image having a resolution of 384x288 pixel (half PAL). The measured times refer only to the process of performing a filtering operation on the grabbed image and writing results into main memory.

Comments: As can be seen windowing techniques have speed advantages, but tend to give worse segmentation results, as a cubic subspace is not optimal in every color space. Usually chrominance values in YUV color space are sub-sampled by a factor of two. This halves calculation cost, but also reduces the spatial resolution of the filter, especially if only windowing in U and V coordinates.

[1] Measured on a Intel Pentium III 500MHz, 256 MB RAM.

[2] As the chrominance values in YUV color space are under-sampled by factor two, the processing times of these filters also decrease by this factor.

[3] Windowing all three parameters: Y, U and V.

[4] Windowing only U and V parameters. Problems arise with black and white, as they can be coded in many different ways in YUV color space.

[5] But adjusting all the parameters tends to be difficult.

Output Examples of Color Filters

For comparison of different color filters a test image was used that can be seen in Figure 5.1

Figure 5.1: Original example image

Test conditions: In the image several objects of different colors are displayed. All color filters were parameterized to segment for red objects. The red objects present in the image are: (a) two circles on white paper background, (b) the visible part of a Chinese lantern between these two circles, (c) the polygonal surface at the bottom and (d) the rectangle on black background at the left side of the image. These objects do not have the same color tone, but differ slightly to provide a broad range of possible object colors. Filter output is optimized and judged to segment all red objects, but none of the other objects. This way the best filter is defined as the one segmenting all kinds of red, but eliminating all other colors.

Results: In Figure 5.2 outputs of filters operating in the RGB color space are presented. On the left side bitmaps delivered by each color filter are shown. White pixels in the

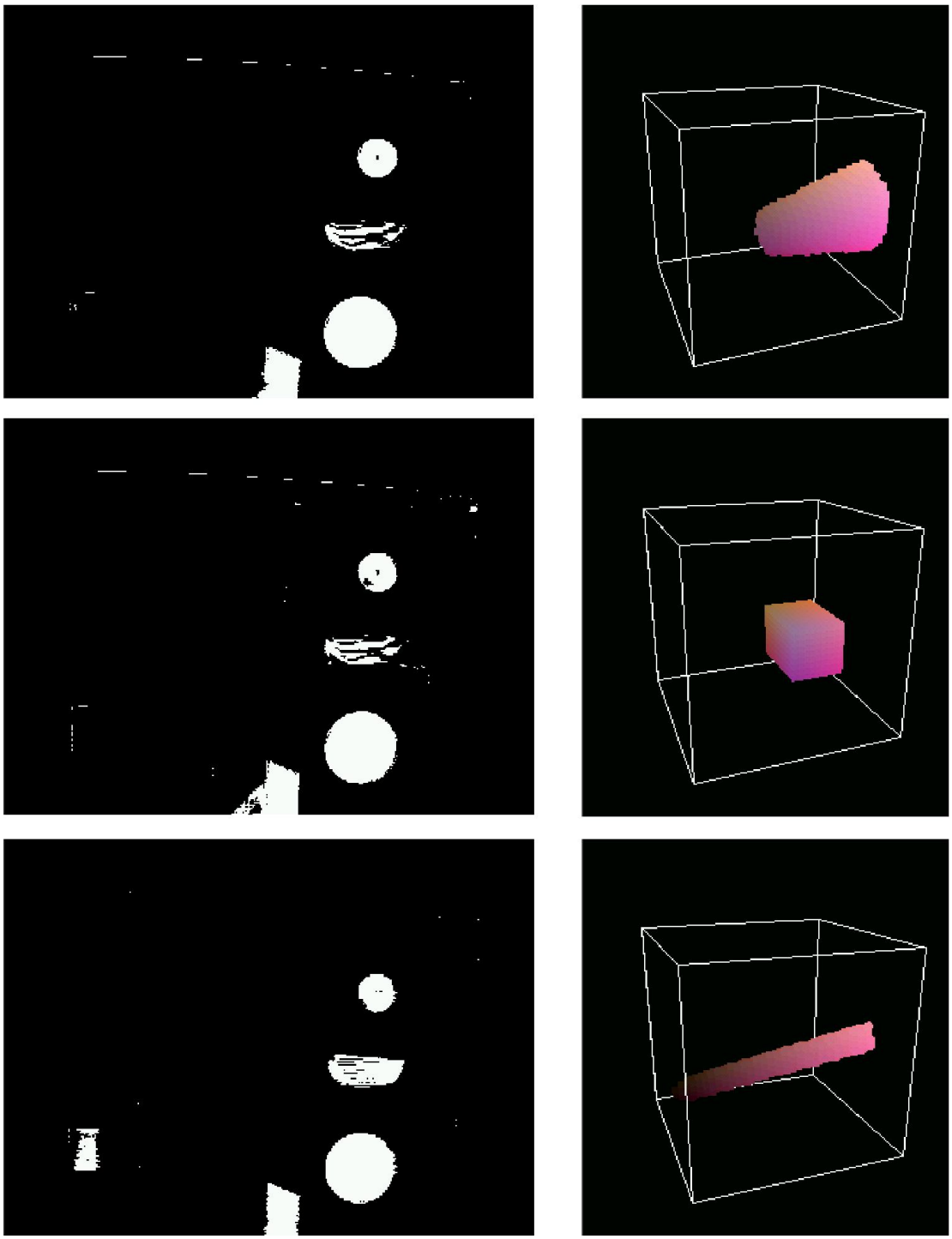


Figure 5.2: Color filter output examples

bitmap represent pixels that have passed the color, a black pixel represents a pixel that was filtered out. On the right side three dimensional representations of the used subspaces

are plotted. In this figure the colors of the pixels in the representations correspond to the colors of the subspace pixels.

In the first row of Figure 5.2 the output of a ConeRGB filter can be seen. Its representation shows a frustum with the truncated peak in the origin of the color space. The output bitmap lacks white pixels for the dark rectangle at the left of the image, but all other red objects are segmented well.

In the second row of Figure 5.2 the output of the CubeRGB filter can be seen. Its representation shows a cube as subspace within the color space. Here the output also lacks white pixels for the dark rectangular object at the left, and reveals a false segmentation for the polygonal object at the bottom of the image.

In the third row of Figure 5.2 the output of the Dichromatic Reflection Model (DRM) filter can be seen. Its representation is a thin plane as a subspace within the color space. The output shows all objects segmented very well, and thereby almost no noise. The drawback of this filter is however, that parameterization is hard to find as there are eight parameters to be adjusted, including the illumination color that changes noticeable if sunlight (with its own spectral distribution) suddenly enters the scene. For comparison filters working on the YUV color space are presented in Figure 5.3. Hereby the colors in the representation do *not* correspond to the color values of the subspace pixels.

In the top row of Figure 5.3 the output of the ConeYUV filter is shown. The form of its representation is the same as for the ConeRGB filter. The output shows many noise pixels, a small segmentation error of the polygonal surface at the bottom of the image and lacks white pixels for the dark object at the left, apart from some outline pixels that would be eliminated in consecutive noise reduction steps. Obviously, a subspace in form of a frustum cannot be applied for the YUV color space.

The middle row of Figure 5.3 shows the output of the CubeYUV filter. Its representation is a cube within the color space. Segmentation results are obviously better than for the ConeYUV filter. All objects have been segmented correctly and only some noise pixels in the upper half of the image disturb the result.

The CubeUV filter shown at the bottom row of Figure 5.3 is very similar to the CubeYUV filter. As the luminance value is not restricted with this filter, the representation is a cube reaching from one side of the color space until the opposite side, allowing all luminance values. As expected, the output of this filter does not differ markedly from the CubeYUV filter. The important advantage of this filter is its speed performance compared to the other filters.

Comments: For the purpose of tracking speed against robustness and accuracy has to be evaluated. In general tracking differs not from the evaluation of still images if motion blur is removed. Hereby the decision was in favor of the *CubeUV* filter, since it gave the optimal speed-accuracy trade-off of all tested filters. Tracking output examples can be found in Section 5.2 as well as in Section 5.4 where the position of the target object was

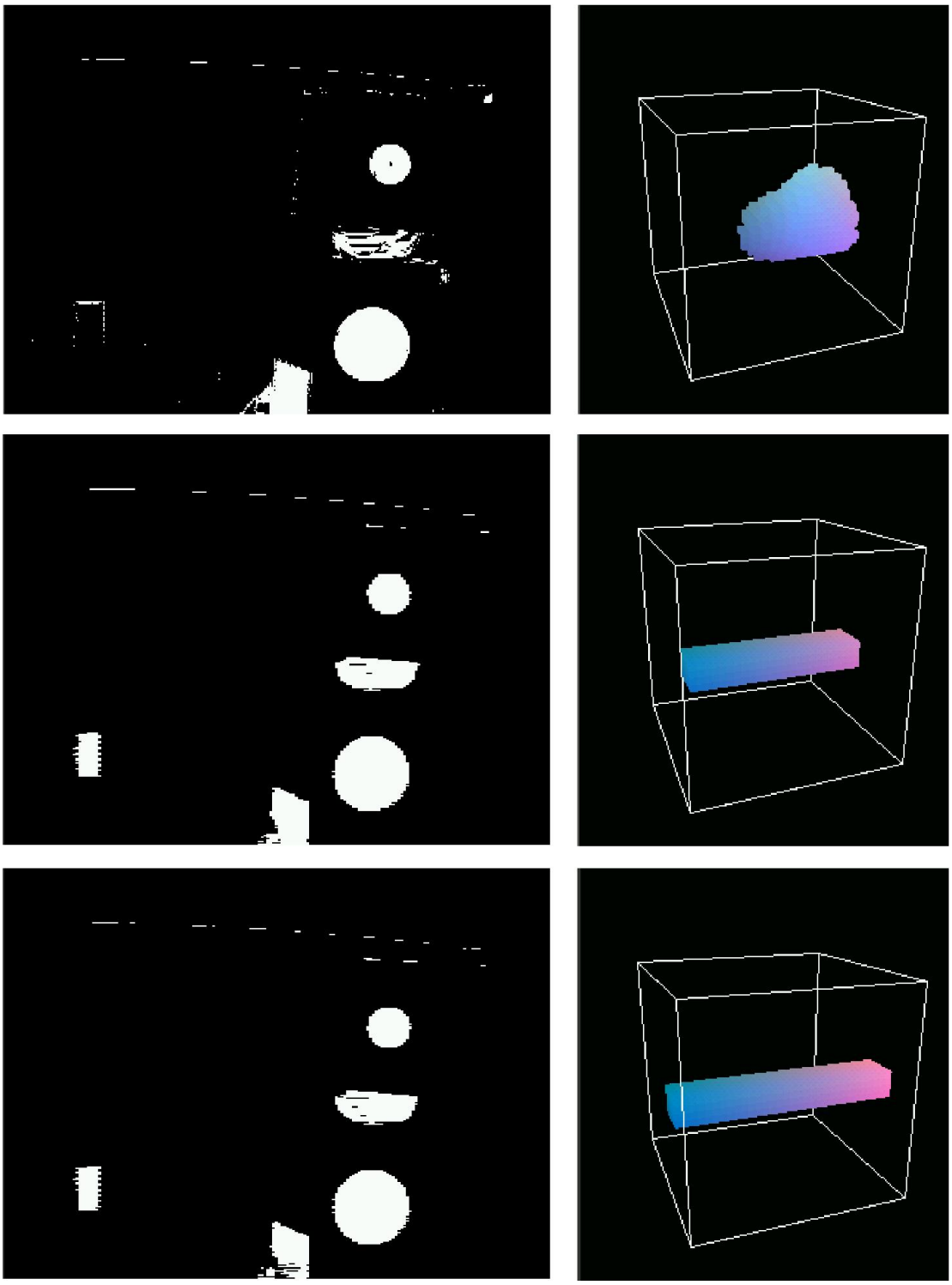


Figure 5.3: Color filter output examples

determined using only color tracking with the *CubeUV* filter.

5.1.2 Form Tracking (CONDENSATION Algorithm)

Test conditions: For the purpose of form tracking the CONDENSATION algorithm was evaluated (see also Section 3.2.1.1). Contour models described as B-Splines were generated using the automated procedure (see Section 4.1.0.3) and used for observation. Simple motion models were set up by hand and in the image sequence of Figure 5.4 a constant velocity model was used for the position coordinates x and y whose parameters were obtained experimentally. For the parameters *scalex*, *scaley* and *teta* (rotation) random motion models with low jitter were used. This is contrary to the procedure proposed by Blake and Isard who determined *exact* motion models through off-line training. Evaluation of the sample contours was performed along the contour, i.e. the match of a sample contour with the image edges was evaluated on discrete points along the contour, or along the contour and few contour normals.

Figure 5.4: Tracking of a commuting bottle

Results: In Figure 5.4 the output of the algorithm during tracking a commuting bottle can be seen. As input to the algorithm *image differences* served in the first two frames (for initalisation purpose), after that the whole image was used. The following tables review the results obtained in this example. Note that the outline of the commuting bottle is quite complex, the bottle shape consists of 8 spans, and 32 contour points were evaluated if matching on contour only was applied. 8 supplementary normals are used if observation along the contour and normals was chosen. Time consumption is reviewed in Table 5.3, while Table 5.2 describes intuitively the obtained exactness of results. Hereby results

Matching on	200 samples	400 samples	600 samples
the curve points only	–	–	–
curve points and a few normals	–	–	0

Table 5.2: Intuitive evaluation: + good, 0 satisfactory, – bad

were not satisfactory as the commuting bottle changes speed, size and rotation quickly. After the initial phase, where the sample set could concentrate on the bottle due to using *image differences* as input for the observation model, too many edges distract tracking. Mainly after the bottle turns around the bend on the left side (see Figure 5.4) and changes speed quickly, which is not modeled in the constant velocity model, tracking is lost. Pre-segmenting the regions of interest, e.g. by maintaining the image differences as input for the observation model or using other modalities (e.g. color or optic flow) can overcome this problem.

Matching on	200 samples	400 samples	600 samples
the curve points only	26	52	78
curve points and few normals	45	100	140

Table 5.3: The time consumption per step in the Condensation algorithm in ms [1].

Comments: Taking a closer look at the above described experiment the *drawbacks* of the method get obvious:

- Clutter attracts the spline contours of the samples leading to misinterpretations of the current object position.
- Exactness of the observation model is important. Matching on the contour only leads to unsatisfactory results.
- Exactness of the used motion models is important, otherwise samples are not placed at the right positions, leading again to misinterpretations.

[1]Note that this time measurement is applied to a careful, but non optimal implementation

- Good results need a large number of samples as well as an exact observation model, both leading to real-time performance problems.

As already stated in the theory chapters the conjunction of different sensor modalities can overcome some of these problems which will be described in the following paragraphs.

5.1.3 Motion Tracking

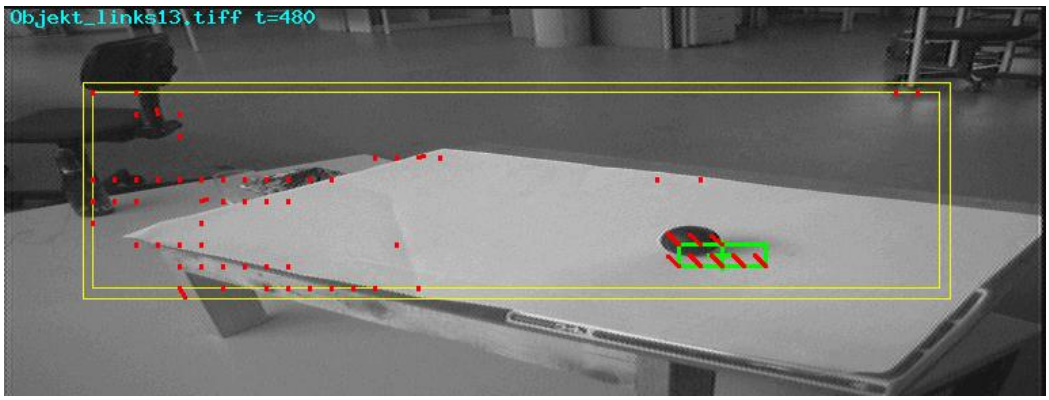


Figure 5.5: Output of the Optic flow sensor

Test conditions: The calculation of the optic flow was performed on a specialized hardware board containing a MEP (Motion Estimation Processor) that was developed at our institute. Further information about the used methods can be found in [Stö01].

Results: In Figure 5.5 the output of the Optic flow sensor is shown. Thereby red lines indicate regions where the flow is non-zero. It can be seen that there are regions detected where is obviously no motion, e.g. on the left part of the table. This comes mainly from illumination changes. The object of interest, in this case a rolling ball, is detected well, what is indicated by the surrounding green rectangle. Here the red lines are definitely longer indicating a clear image flow.

Comments: Unfortunately, experiments with optic flow where limited to the above mentioned example due to hardware problems.

5.1.4 Modified ICONDENSATION

In the following the results using the Modified ICONDENSATION algorithm are evaluated. As described in Section 4.3.2.1 the algorithm is capable of integrating sensor information from different modalities. In the successive experiments the information from the color sensor was integrated with the information from the form sensor.

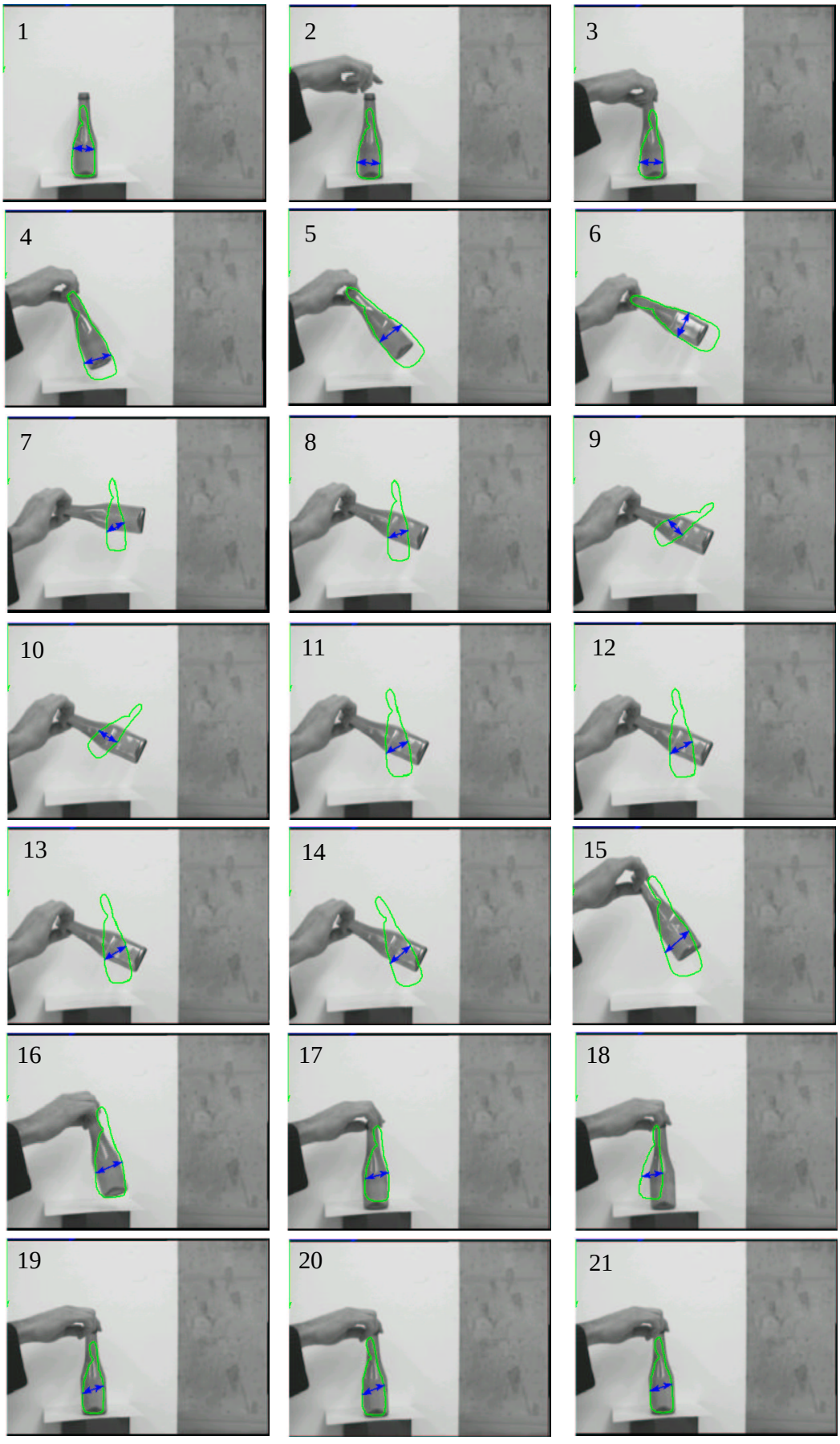


Figure 5.6: Tracking a green bottle moved by hand. The output of the Modified ICONDENSATION algorithm is shown (estimated outline representing the *Mean* of all samples)

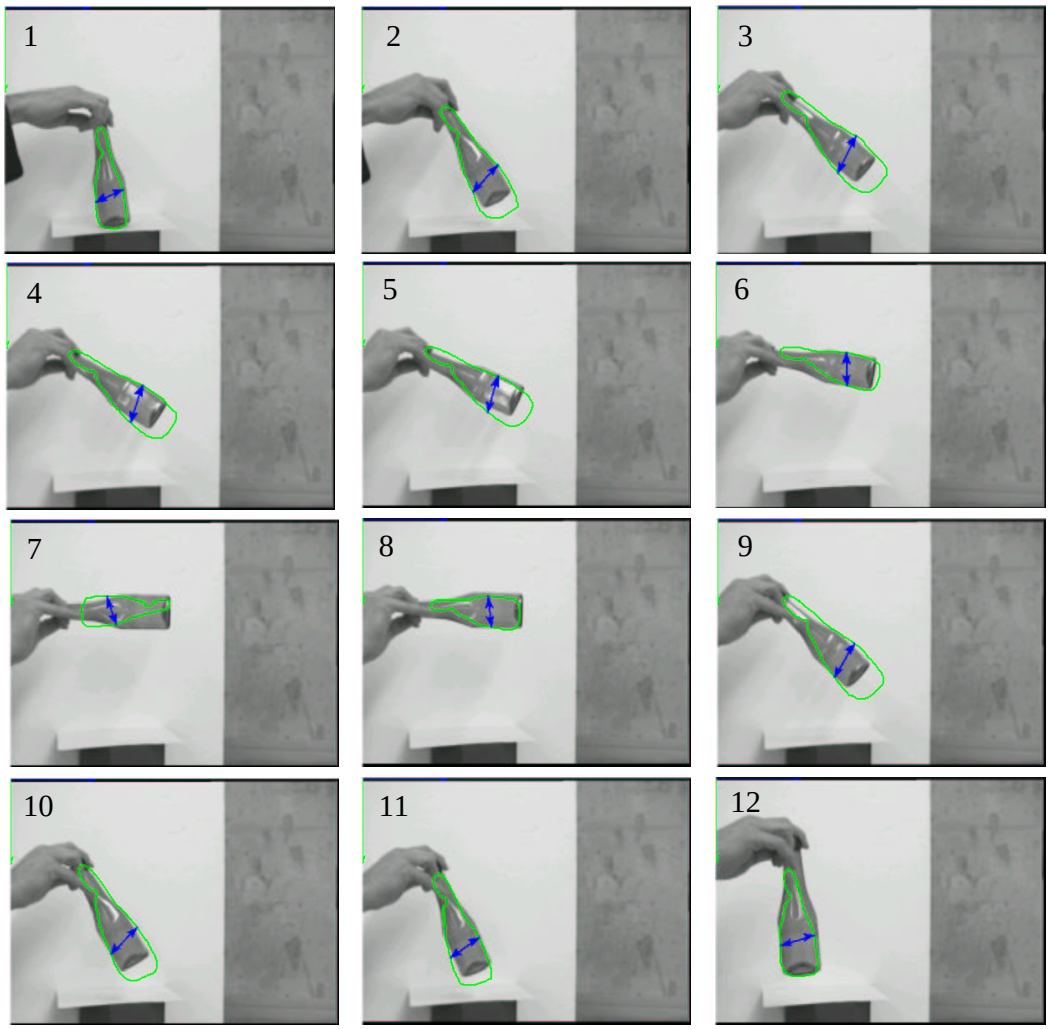


Figure 5.7: Tracking a green bottle moved by hand. The output of the Modified ICONDENSATION algorithm is shown (estimated outline representing the *Mean* of all samples)

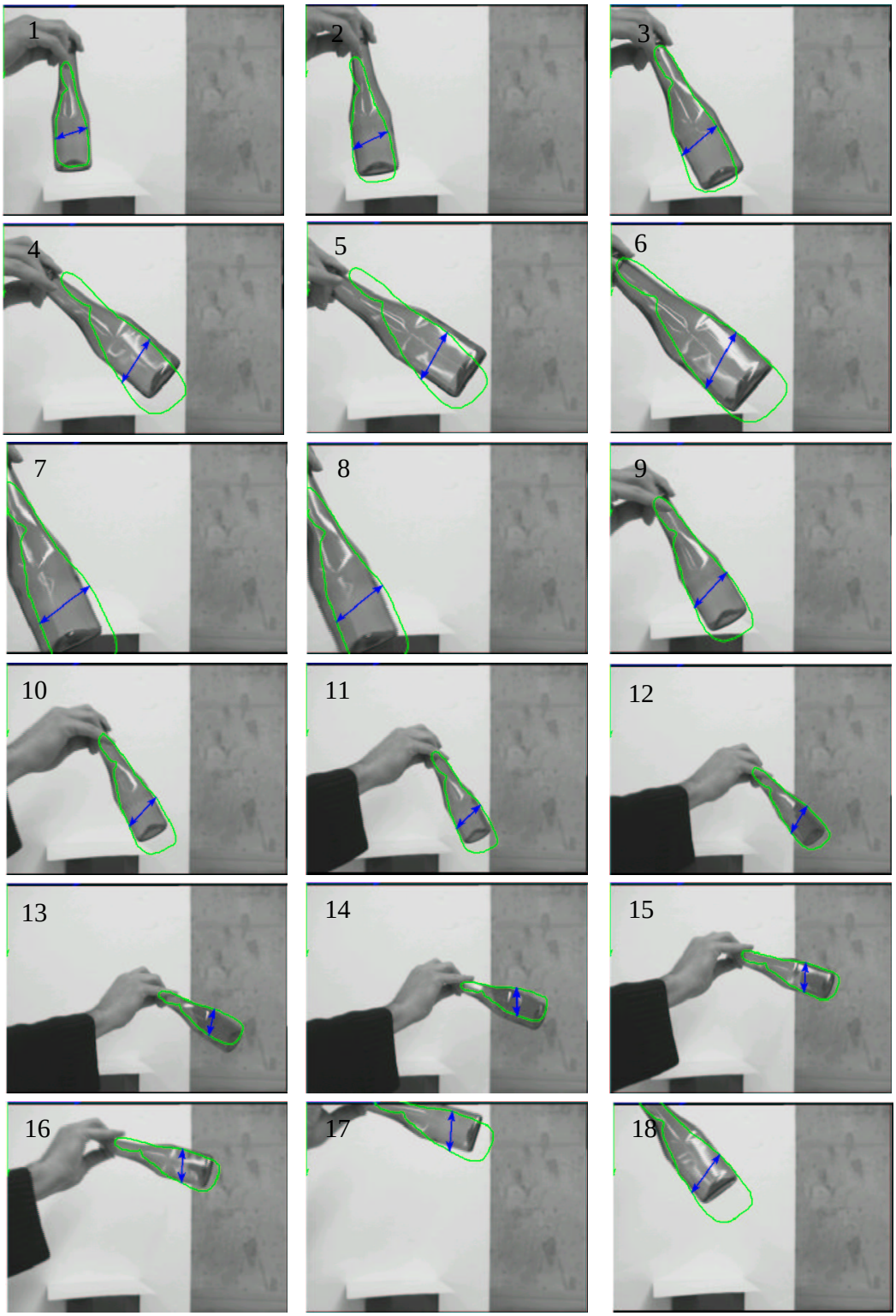


Figure 5.8: Tracking a green bottle moved by hand. The output of the Modified ICONDENSATION algorithm is shown (estimated outline representing the *Mean* of all samples)

Test conditions: In all sequences a green bottle is moved by hand against a plain white background. The Modified ICONDENSATION algorithm described in Section 4.3.2.1 is applied. Thereby as additional sensor input only color is analyzed and the current size and orientation of the bottle is estimated from the found color regions[2]. In the examples given here the required input parameters $step \in [1, N]$, $q \in [0, 1]$ and $r \in [0, 1]$ with $q+r \leq 1$ (see algorithm 1 in Section 4.3.2.1) were chosen to have only importance samples ($\approx 95\%$) and prior samples ($\approx 5\%$), since the tracking sequence was quite short and the tracked motion quite irregular. Sample contours were evaluated only along the contour. The number of evaluated samples was constantly 100.

Results: In the first images of the sequence in Figure 5.6 the bottle is detected well and the tracking error is low (except of some overestimation of the bottle's size). In the 7th frame the bottle is lost, i.e. the orientation is estimated wrong and the tracker does not recover until the 15th image.

In the second sequence (Figure 5.7) the color region is evaluated well throughout the sequence. The required parameters *step*, q and r were the same as in the sequence of Figure 5.6. Misinterpretation occures only once when the orientation flips by 180° (frame 7). This is due to the large emphasis on importance samples, i.e. if the orientation determination goes wrong in one frame this is not compensated by prediction samples (taking into account the "tracking history") in this case.

In the third sequence (Figure 5.8) the determination of the objects size by analyzing the color region is shown. One can see that the B-spline contour grows in size when the bottle is moved closer to the camera. Additionally tracking against dark background and a strong edge (white to dark transition from the paper background to the wall) is shown in frames $10 - 16$. Throughout the whole sequence tracking is not lost.

Comments: Through the choice of a fixed sample distribution without any prediction samples the drawback of a cut out of a motion model gets obvious: tracking can only rely on (additional) sensor information that massively influences the quality of the output and thereby leads to wrong results if the sensor information is misinterpreted.

Nevertheless, it can be stated, that use of additional sensor information (as e.g. in this case color) can overcome problems occurring with the Standard Condensation algorithm, namely,

- The spline contours tend to stay away from clutter background edges.
- Observation along the contour is sufficient.
- Missing motion model can partly be compensated through additional sensor information.

[2]To estimate size and orientation standard operators provided by the image processing library *HALCON* were used, i.e. *area_center* or *elliptic_axis* (see the *HALCON* manuals available at *www.mvtec.de*)

- The accuracy is satisfactory even given only a small number of sample outlines, but temporarily misinterpretations occur.

These results are a good starting point for further evaluation of the Modified ICONDENSATION algorithm, i.e the online adaption of tracking parameters to change the sample distribution. The experiments thereby are given below.

Grasp Determination and Grasp Tracking

Test conditions: see above. The grasp was calculated and tracked on the output spline.

Results: In the image sequences shown in Figure 5.6, Figure 5.7 and Figure 5.8 the output of the *Grasp Determination and Grasp Tracking* algorithm (see Section 4.4.1) is indicated by arrows drawn on the bottles. In the first image of each sequence an initial grasp is determined as the *optimal grasp* for the given shape. Thereafter this grasp is being tracked in successive frames. There the advantage of using B-spline contours for tracking becomes obvious. Independently of the transformation that is applied to the initial spline (e.g. the scaling operation for the approaching bottle) the grasp stays stable on the same position of the bottle.

Comments: The constancy of the grasp over many frames is a very important feature for catching experiments, since the pair of grasping points, or the grasping center point, respectively, is used as an input to the motion reconstruction and prediction algorithms, and finally to the motion control of the robot arm. It is obvious, that a relative motion of the "grasp" on the moving bottle would lead to a disturbance for the motion reconstruction and prediction algorithms, what would make it even more difficult to catch that moving object.

Online adaption of tracking parameters

Test conditions: To test the proposed online adaption of tracking parameters (see Section 4.3.2.1) a simple tracking scenario was chosen where a model train moved on a circular railroad with constant speed, except for the acceleration phase in the beginning (see Figure 5.9). To obtain comparable results **this setup served as the tracking scenario for all experiments on tracking described in the rest of this section!** As additional sensor input again color was used. The maximum translatory jitter for prediction and difference prediction samples was set to 30 pixel. The number of used samples was constantly set to 100. As output of the algorithm the *mean* of all samples was evaluated. The distribution of the samples during the trial is shown in Table 5.4. Figure 5.9 middle and right shows exemplarily how the samples were placed during a trial. On the left of the figure the color region and the *mean*, a B-spline curve, can be seen.

Results: The results of the tracking can be seen in Figure 5.10. In the first row $x(t)$ and $y(t)$ are plotted, in the second row the according errors, and in the third row the error

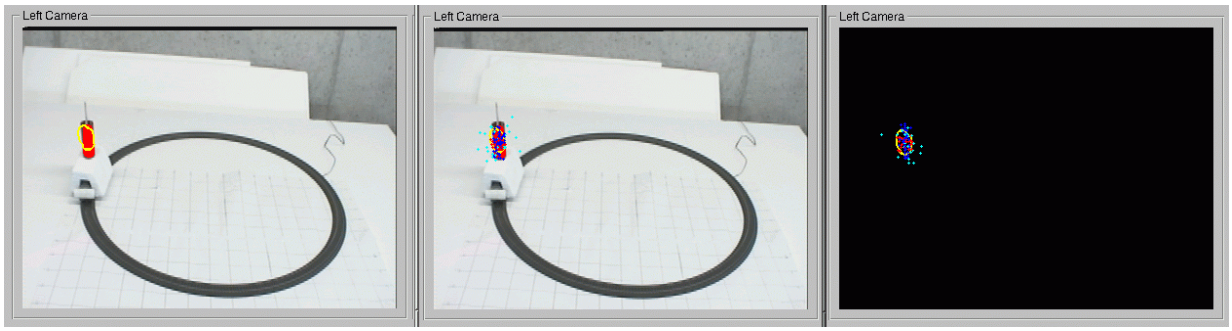


Figure 5.9: The tracked object with sample distributions: 25% difference prediction samples (light blue dots), 24.75% prediction samples (red dots), 50.25% importance samples (dark blue dots). In the left image the segmented color region and the mean sample is shown, in the middle and right image the sample distribution with the mean sample.

Frame	importance	prediction	difference prediction
1 - 29	100.0%	0.00%	0.00%
30 - 59	90.25%	4.75%	5.00%
60 - 89	80.10%	9.90%	10.00%
90 - 119	69.43%	16.28%	14.29%
120 - 149	60.00%	20.00%	20.00%
150 - end	50.25%	24.75%	25.00%

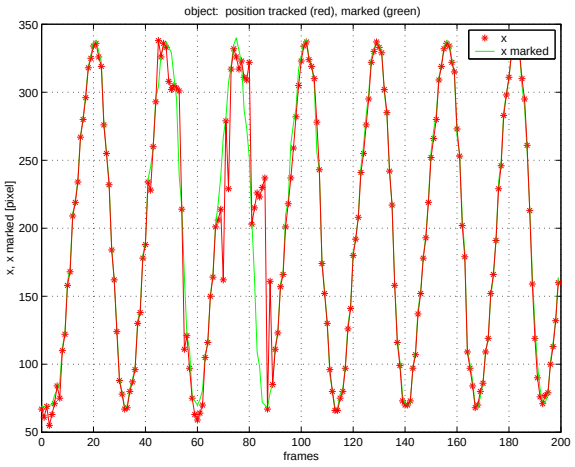
Table 5.4: Temporal course of sample distribution during tracking

norm. It can be seen that the object is temporarily lost in the part of the sequence where the error norm is *above* $\approx$ 40 pixel. Nevertheless the tracker recovers again and stable tracking is performed for the last two rounds.

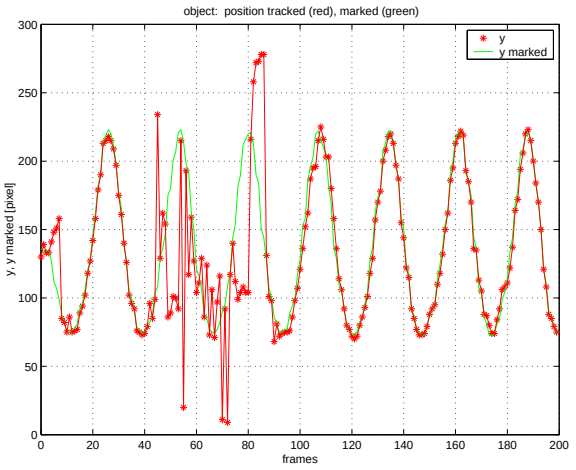
Comments: First, it can be stated that the Modified ICONDENSATION and the online adaption is applicable for tracking the sequence. Despite the fact that the object is lost early during tracking, the AARM motion model can be generated successfully and subserves successful tracking. This can be seen from the last part of the plots (from $\approx$ frame 100 to frame 200) where the number of importance samples drops from $\approx$ 70% to $\approx$ 50% and therefore tracking gets dependent on reliable predictions. Also it has to be noted that the absolute number of samples, 100, is low compared to they sample-set usually used in e.g. [BI98].

5.1.5 Reentry of Color in Form Path

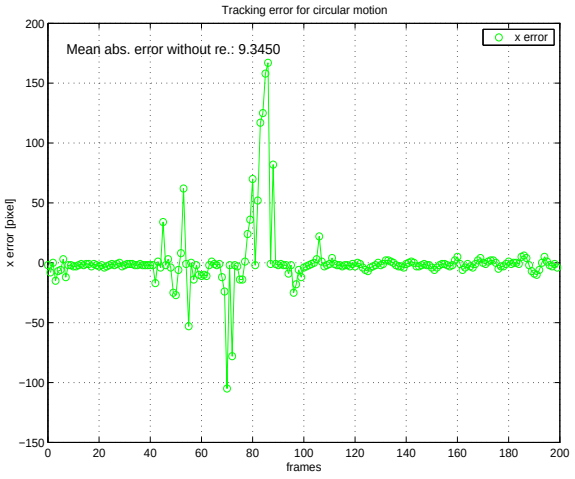
Two biological concepts of human visual processing have been presented in Section 2.1.2: the concept of *reentry* as well as the concept of *feature maps*. The later applied well to the



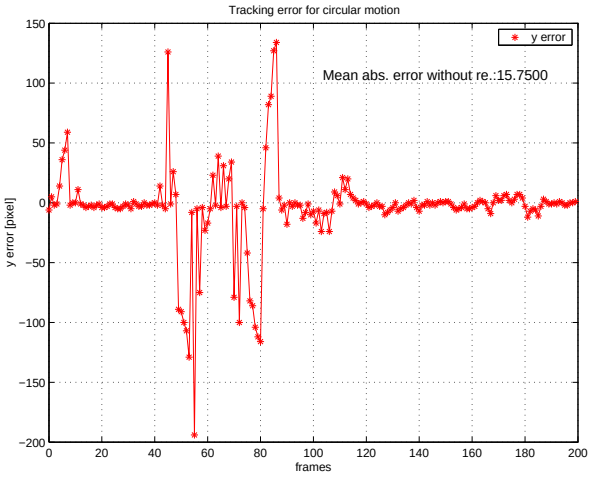
a)



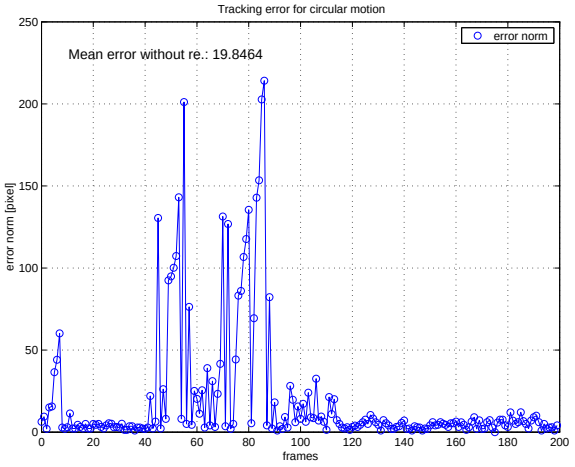
b)



c)



d)



e)

Figure 5.10: Tracking with Modified ICONDENSATION and online adaption of sample distribution according to Table 5.4. Thereby displays a) $x(t)$, b) $y(t)$ c) $x_{error}(t)$, d) $y_{error}(t)$, e) error norm $\sqrt{x^2 + y^2}$

ideas of sensor integration as performed within the ICONDENSATION framework. The former is analyzed more detailed in the following, thereby distinguishing between the effect on still images as well as on image sequences. Concerning image sequences the interesting question was, if the effect provided by reentry improves tracking, i.e. if the output of the Modified ICONDENSATION algorithm can be improved.

Still Images The images in Figure 5.11 show the effect of parallel information flow processing combined with the reentry concept. On the image in the upper left the original camera image is given, showing a red circle printed on white paper as the object of interest. Several distractor objects and a random cluttered background fill the rest of the image. The color segmention is configured to filter for red objects. On the image right to the camera image you can see the output of the color segmention, indicated by the red area.

The second and third row of Figure 5.11 show examples of both, the Sobel image and the edge areas. On the left side the results are displayed *without* reentry of color filter information into the form sensor. On the right side the results *with* color reentry are shown. In the Sobel images you can see the effect of multiplying

$$f(x,y) = \begin{cases} 2 & (x,y) \in \mathcal{A} \\ 0.5 & (x,y) \notin \mathcal{A} \end{cases}$$

on the Sobel image pixels. Pixels near or within the color region are amplified, thus giving stronger edge pixels, whereas pixels far away from the attentive color region are weakened. Therefore the number of edges decreases when applying a threshold on the modified Sobel image, if the threshold value is configured to accept only strong edge pixels. This decrease also accelerates the whole application in the following (see Table 5.6 for the dynamic case).

Image Sequences (with Modified ICOND.), constant tracking parameters

Figure	importance	prior	prediction	difference prediction
Figure 5.12a	40.00%	10.00%	0.00%	50.00%
Figure 5.12b	40.00%	10.00%	0.00%	50.00%
Figure 5.12c	40.00%	10.00%	0.00%	50.00%
Figure 5.12d	50.25%	0.00%	24.75%	25.00%
Figure 5.12e	50.25%	0.00%	24.75%	25.00%
Figure 5.12f	50.25%	0.00%	24.75%	25.00%

Table 5.5: Sample distribution during tracking

Test conditions: Tracking of the model train, moving at constant speed, was performed again. Thereby constant tracking parameters according to Table 5.5 were adjusted for the

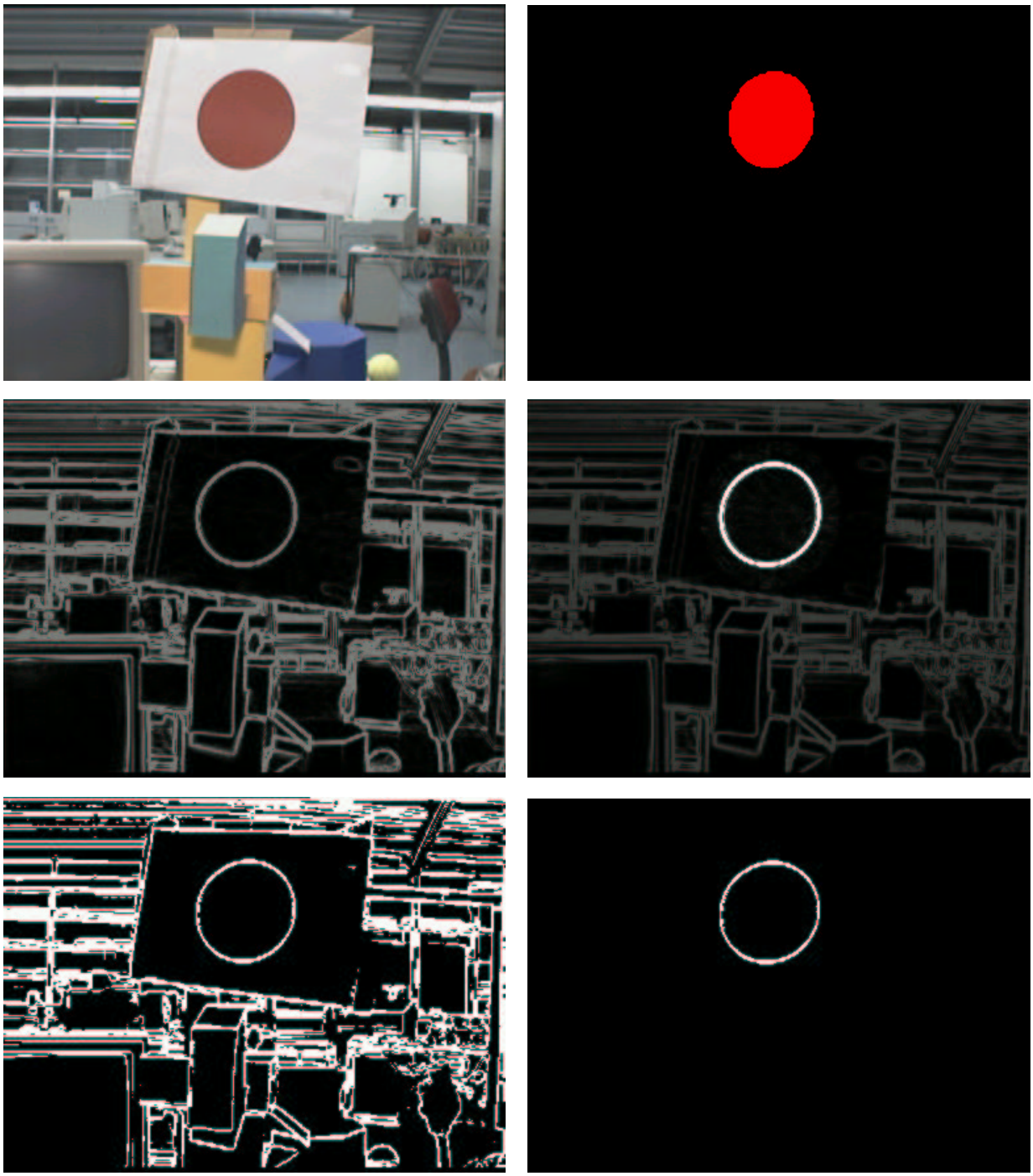


Figure 5.11: Examples images showing the effect of the implemented biological concepts, i.e. the reentry mechanism. Upper left: original image. Upper right: output of color segmentation. Middle left: Sobel image. Middle right: color information *reentered* into Sobel image. Lower left: edge image. Lower right: color information *reentered* into edge image.

different trials. Compared are the two cases: "tracking *with* color reentry" and "tracking *without* color reentry".

Results: Plots is Figure 5.12 show x-y plots of the tracked motion. Plot a) shows that with and without reentry tracking is succesful for $\approx 80\%$ of the trial. Large errors occur only in the upper segment of the circle. This error (that can also be seen b) and c)) is caused by a background edge. Plot b) shows for both cases tracking errors, which are lower for the "with reentry" case. Plot c) shows a similar behavior as plot a). Plot d) shows perfect tracking for both cases with almost no errors. In plot e) a trial is displayed where the train is completely lost in the case "without reentry". Tracking is nevertheless good in the "with reentry" case. Plot f) finally shows similar tracking behavior for both cases as is shown in plot a).

To quantify these observations error plots for the: *x error*, *y error* and the *error norm* for all six trials are summarized in Figure 5.13. Thereby plot a) shows the x error for all data obtain from plots a), b) and c) of Figure 5.12. Plot b) shows the same for the y error. Plot c) shows the x error for all data obtain from plots d), e) and f) of Figure 5.12. Plot d) shows the same for the y error. Plot e) shows the error norm, defined as $\sqrt{x^2 + y^2}$, of a) and b). And finally, plot f) the error norm of c) and d). Thereby it can be stated that a error norm of more than ≈ 40 pixels indicates a loss of the object.

Plots in Figure 5.14 show details of the tracked circular movement of plot b) in Figure 5.12 for the case "without reentry", whereas plots in Figure 5.15 show details for the case "with reentry". In both cases can be seen that the object can be successfully tracked throughout the sequence, i.e. the object is never lost. Comparing the *mean errors* reveals that in case of reentry the error is marginally lower, but not significantly. That means that reentry is not improving the exactness of a *successful* tracking trial.

The contrary case to the above mentioned successful trials show the plots in Figure 5.16 and in Figure 5.17, respectively. Thereby Figure 5.16 corresponds again to the "without reentry" case, whereas Figure 5.17 corresponds to the "with reentry" case. Here the real fortitude of the reentry mechanism gets obvious: it assures that the object is not lost during tracking, respectively recovered quickly in the case of temporal failure. This is in particular favorable during early tracking when no motion model could be acquired yet. In this case without reentry there is no recovery from failure.

Comments: It can be stated that the effect of reentry on tracking is very positive in three ways:

- The mean tracking error is considerably lower than without reentry[3].
- Loss of the tracking object is considerably more seldom.
- Recovery from a loss (re-initialization) is faster and always possible. Thus the online

[3]It should be noted that the mean here takes into account all errors, i.e. also the losses.

build motion model is not “destroyed”.

Otherwise there no significant difference can be seen for the different sample distributions, i.e. the use of prediction samples shows no real improvement.

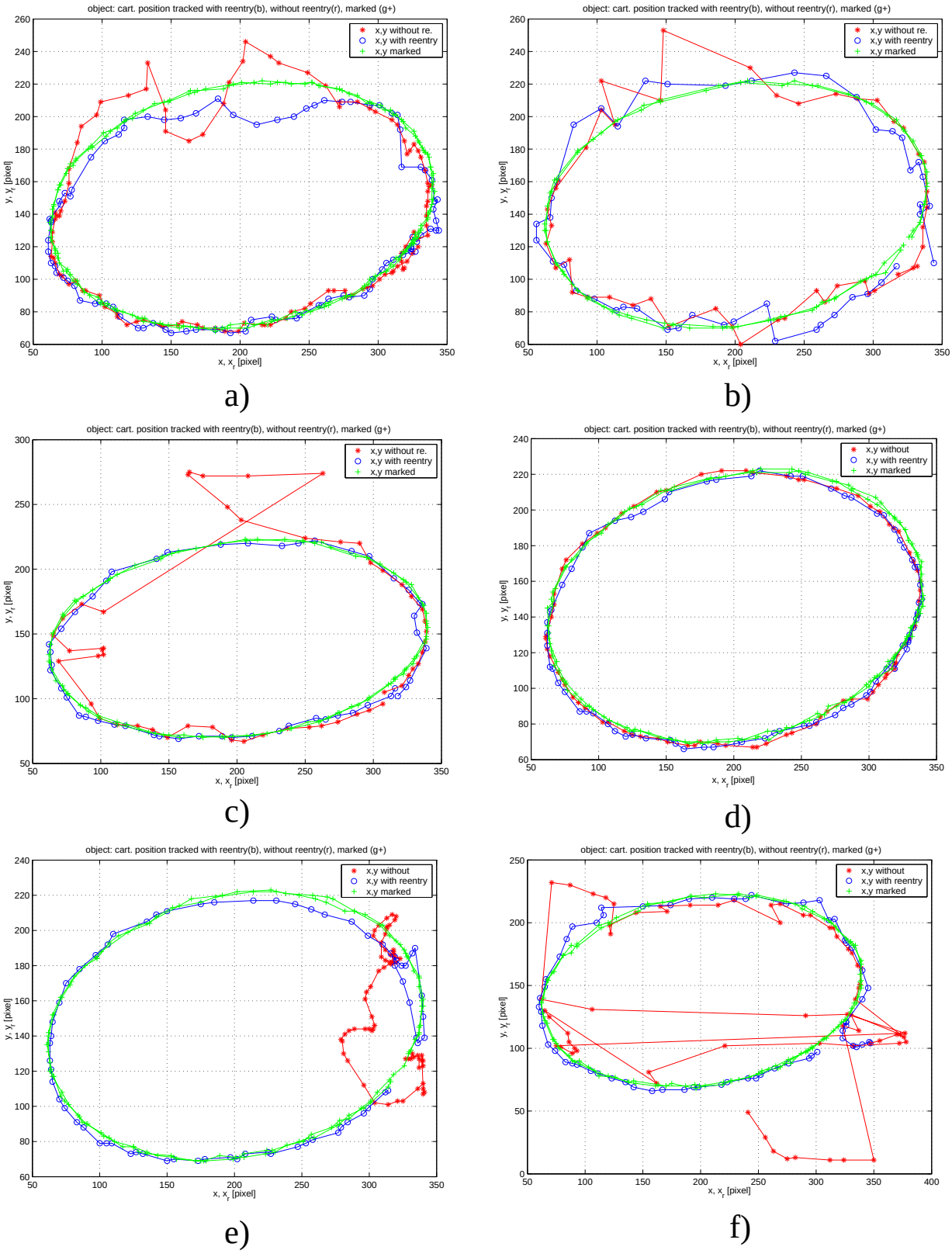
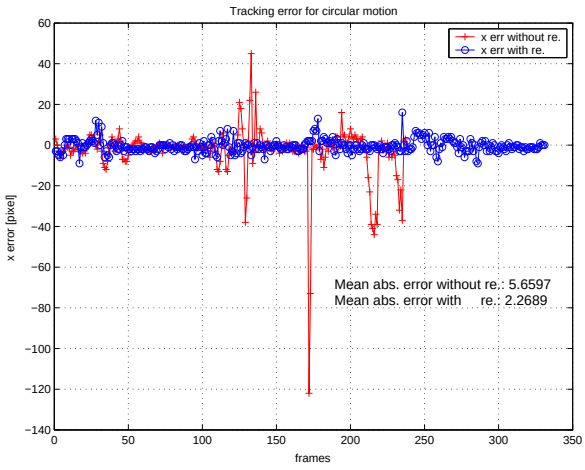
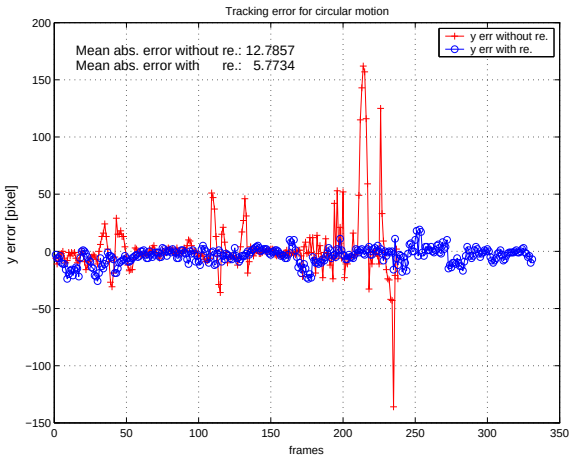


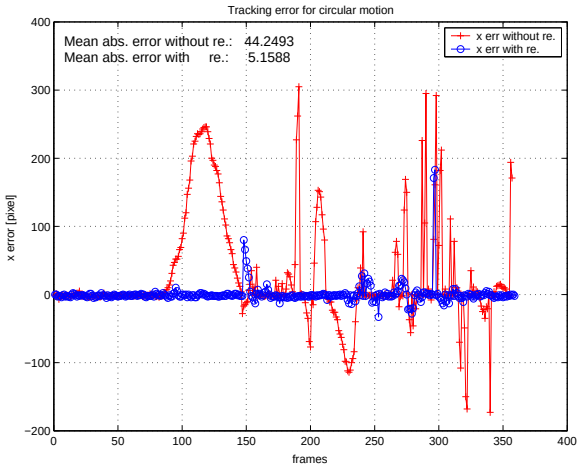
Figure 5.12: Plots a), b), c), d), e), f) all show x-y plots of a circular motion tracked with the modified ICONDENSATION and a sample distribution according to Table 5.5



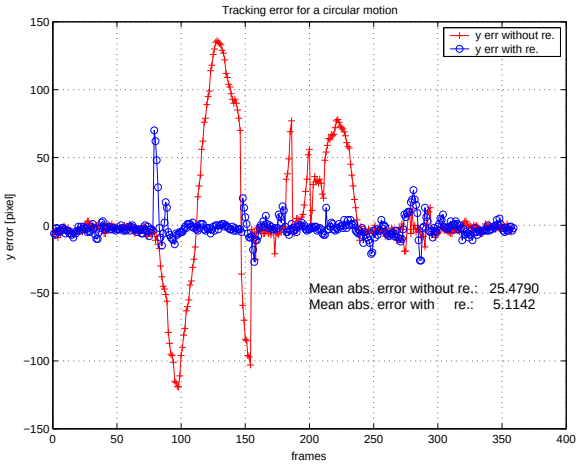
a)



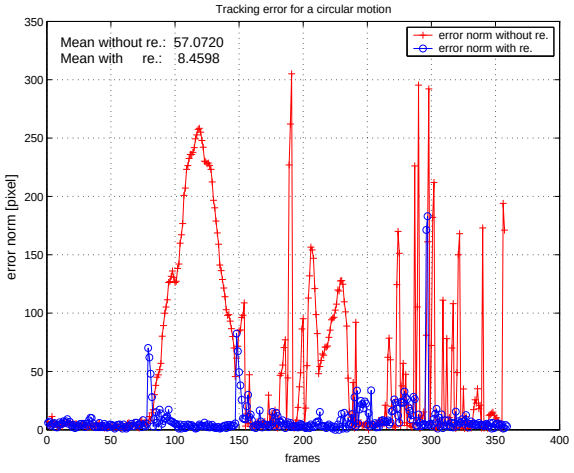
b)



c)



d)



e)

Figure 5.13: Plot a) shows the x error for all data obtained from tracking plots a), b) and c) of Figure 5.12. Plot b) shows the same for the y error. Plot c) shows the x error for all data obtained from tracking plots d), e) and f) of Figure 5.12. Plot d) shows the same for the y error. Plot e) shows the error norm of a) and b). Plot f) the error norm of c) and d). Definitions: Mean error $x = 1/frames \sum_{i=1}^{frames} x_i$, Mean abs error $x = 1/frames \sum_{i=1}^{frames} \|x_i\|$, error norm $\sqrt{x^2 + y^2}$

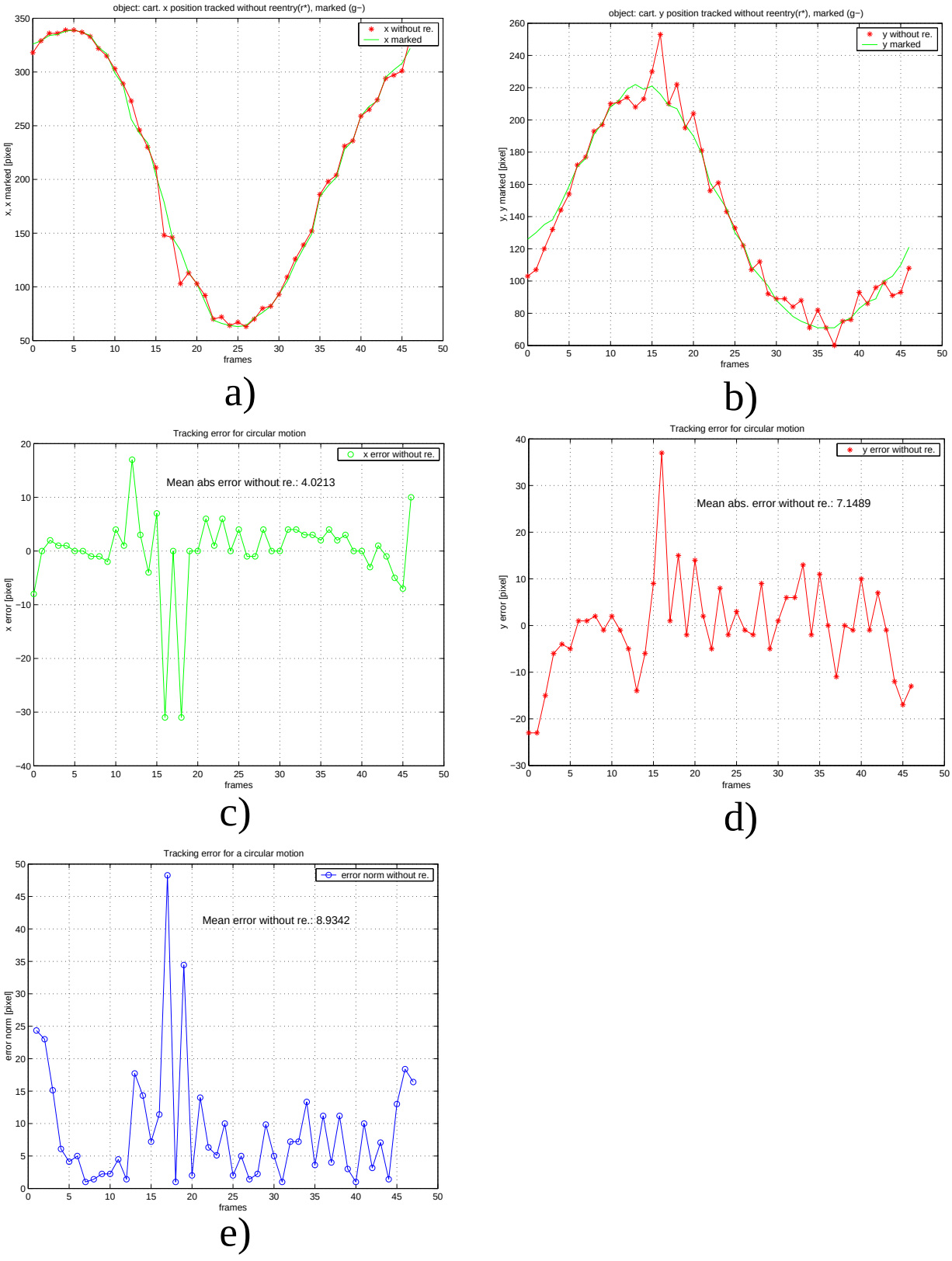
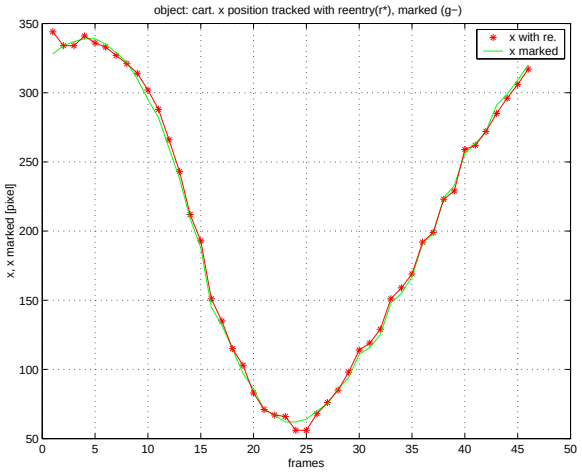
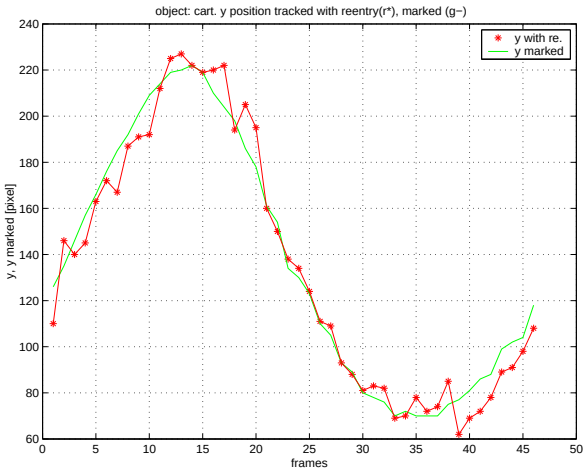


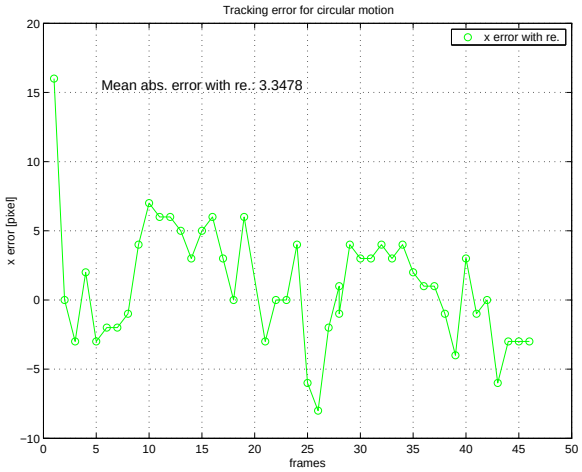
Figure 5.14: Plots show details of trial “without reentry” in Figure 5.12b



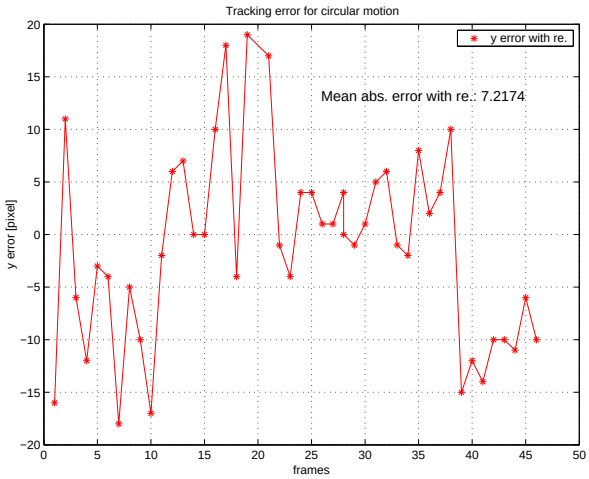
a)



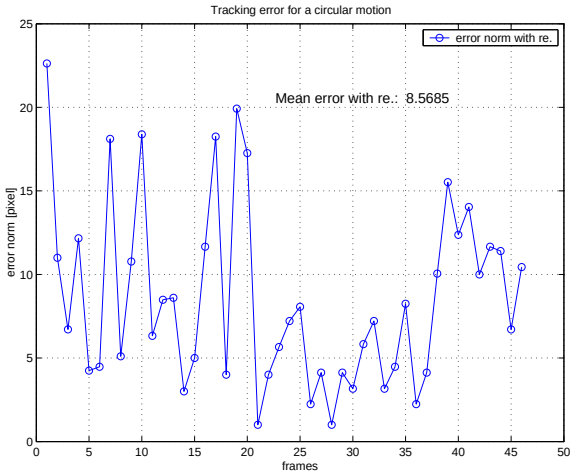
b)



c)

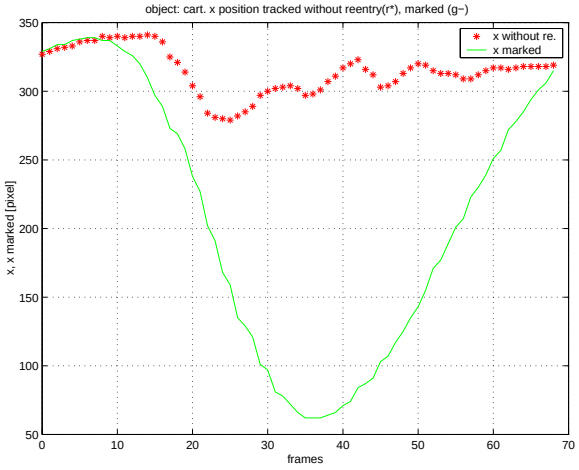


d)

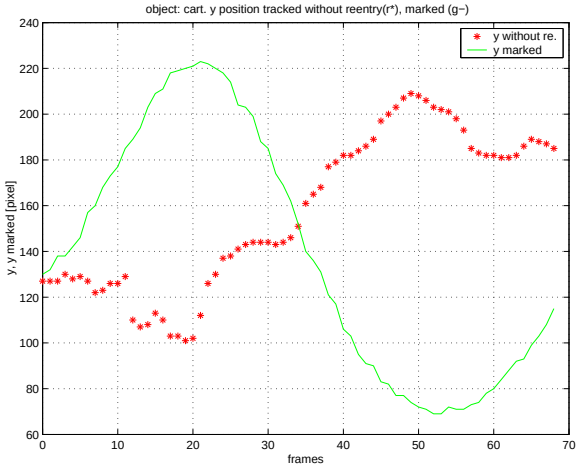


e)

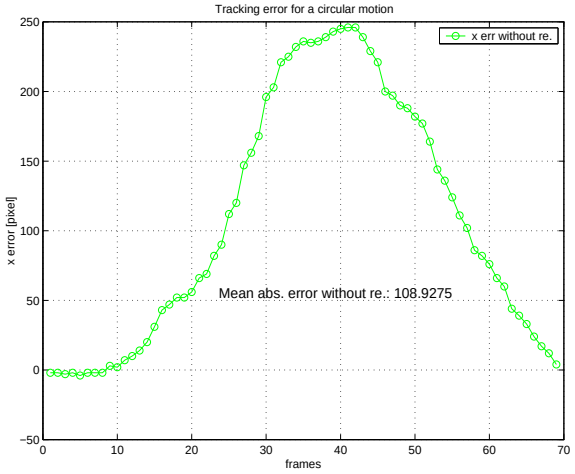
Figure 5.15: Plots show details of trial “with reentry” in Figure 5.12b



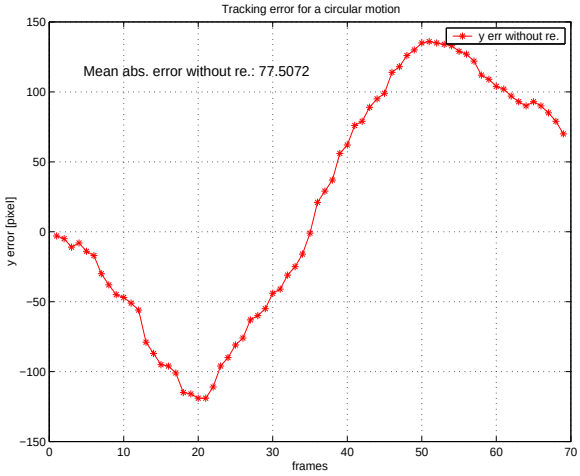
a)



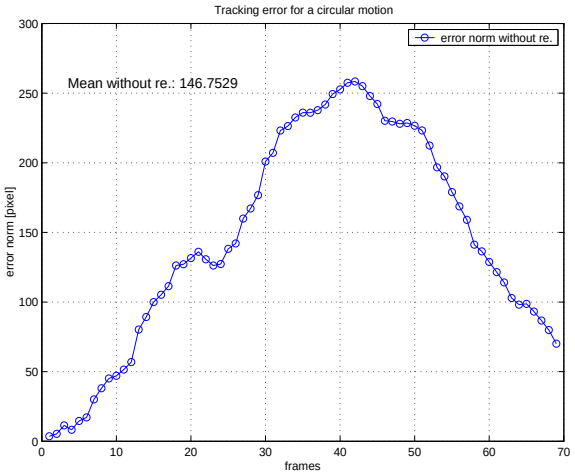
b)



c)

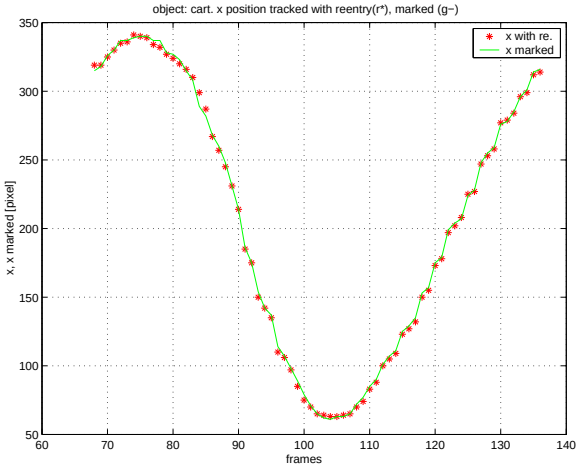


d)

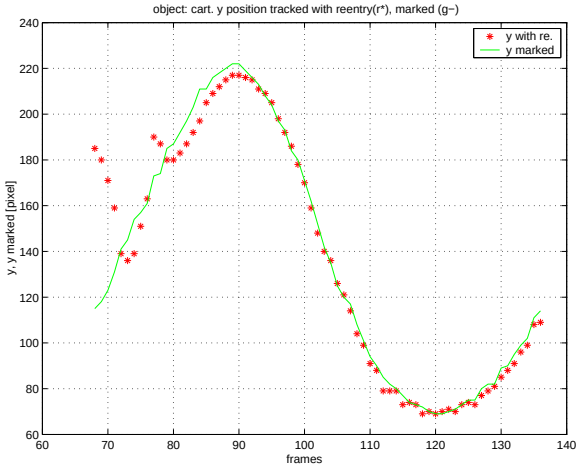


e)

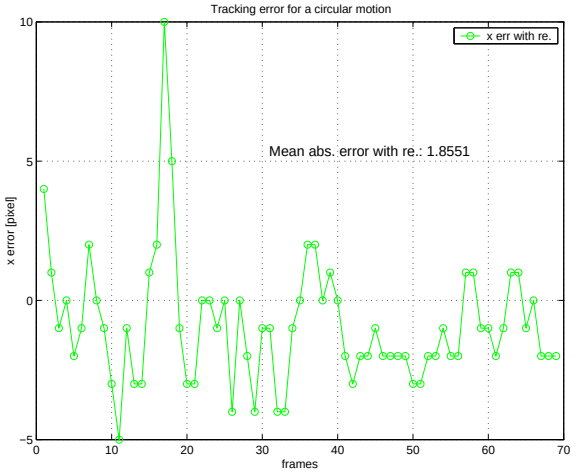
Figure 5.16: Plots show details of trial “without reentry” Figure 5.12e



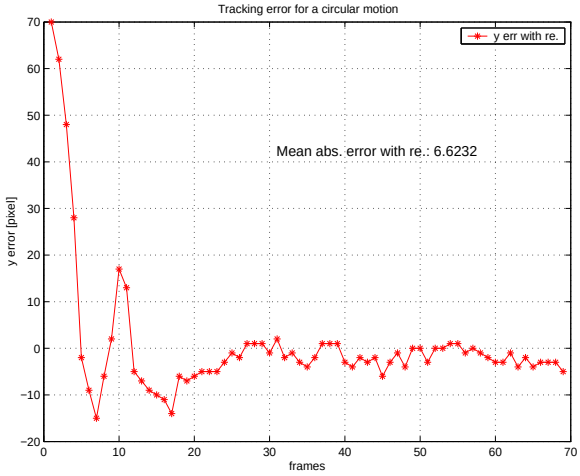
a)



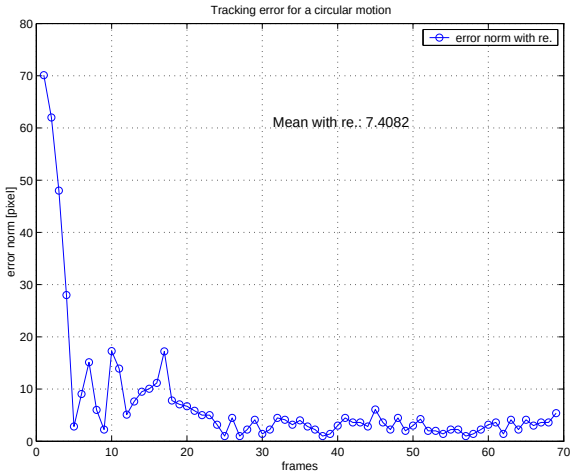
b)



c)



d)



e)

Figure 5.17: Plots show details of trial “with reentry” Figure 5.12e

Image Sequences (with Modified ICOND.), variable tracking parameters

In Figure 5.10 the case of online adaption of the tracking parameters was shown for the Modified ICONDENSATION algorithm. Here the same experiment was repeated with the only difference that color reentry was allowed. Results of this trial can be seen in Figure 5.18. Here the comparison reveals again the advantage of the reentry mechanism: The Modified ICONDENSATION was capable to track an object knowing its color and contour and to generate a motion model during tracking. Failures, i.e. loss of the object, can occur if by small deviations wrong assumptions about the current object motion occur. In the "with reentry" case this problem is avoided throughout the complete sequence[4].

Samples	time with re.	time without re.	Difference
500	91.105	99.079	7.974
400	77.725	85.939	8.214
300	68.000	72.792	4.792
200	58.030	62.024	3.994
100	46.711	51.171	4.460
70	42.137	47.351	5.214
40	39.350	43.456	4.106
10	35.784	40.368	4.584
5	35.577	39.219	3.642

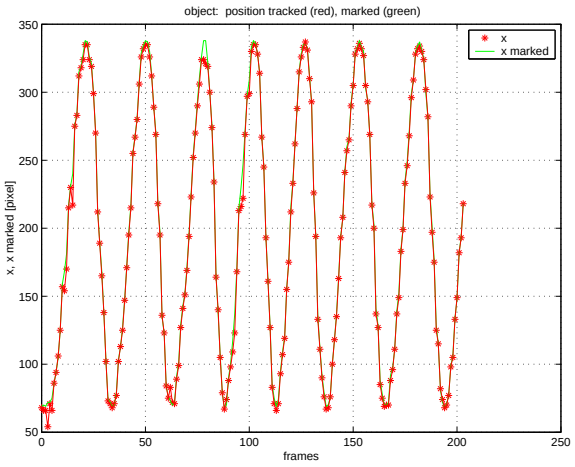
Table 5.6: Measured time consumption in ms of Modified ICONDENSATION algorithm with and without color reentry. Used fixed sample distribution: 50.25% importance samples, 24.75% prediction samples, 25.00% difference prediction samples.

Comparison of time consumption The last analysis regarding reentry compared the time consumption of the Modified ICONDENSATION algorithm again for the cases "with reentry" and "without". Results are shown in Table 5.6. It can be seen that reentry has no negative effect on image processing times, in contrary processing times are always marginally lower than in the "without reentry" case. This shows that the reentry mechanism does not only improve tracking performance, but also improves processing speed.

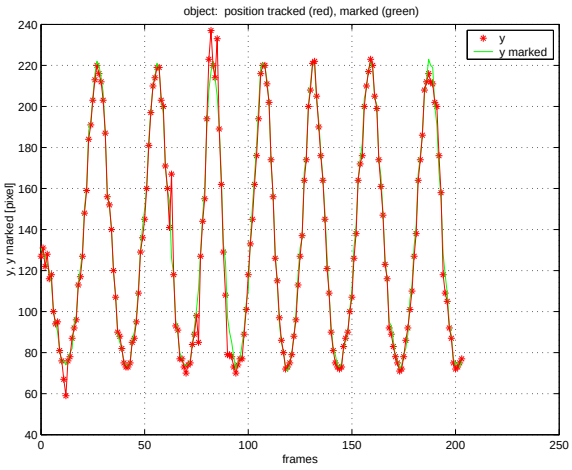
5.2 Prediction of Target Motion

To test the prediction algorithms that were described in Sec. 3.3 simulations using *MATLAB* were performed. To test the Average ARM algorithm (see Sec. 4.5.1) real image sequences were used.

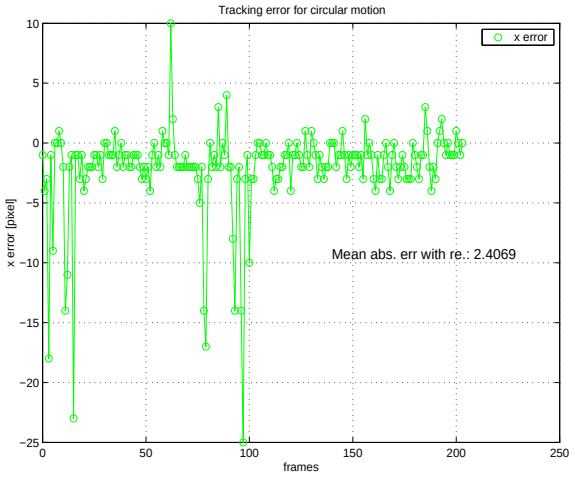
[4]It can also be shown that disturbances through objects of the same color as the tracked object have no negative influence if they are not moving.



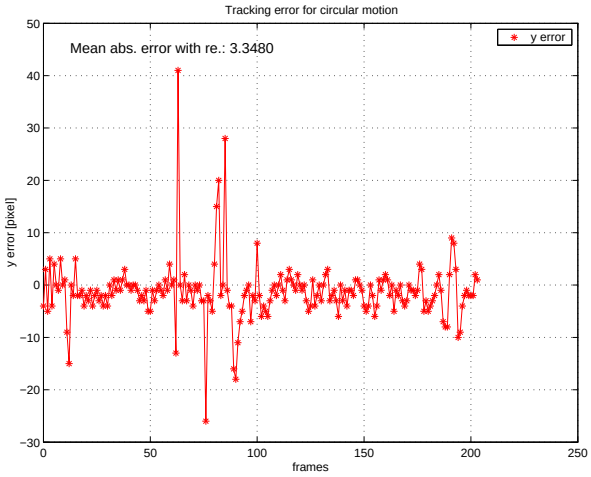
a)



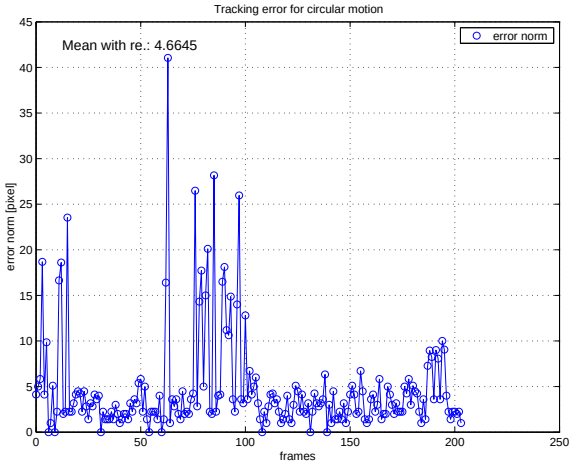
b)



c)



d)



e)

Figure 5.18: Tracking with Modified ICONDENSATION, color reentry into the form path and online adaption of sample distribution according to Table 5.4. Thereby displays a) $x(t)$, b) $y(t)$ c) $x_{error}(t)$, d) $y_{error}(t)$, e) error norm $\sqrt{x^2 + y^2}$

5.2.1 Simulation: Comparison NN, Global ARM, Local ARM

Test conditions: To validate different prediction methods simulation models were developed using *MATLAB*. Hereby the algorithms were tested with "artificial data" e.g. data obtained using mathematical functions e.g. $sin(t)$ or numerical rows set up by hand.

Results: In Figure 5.19a-l, Figure 5.20a-l, Figure 5.22a-l and Figure 5.21a-l different object motion forms were used to evaluate the performance of the prediction algorithms. In all plots input data is indicated by a $+$, and prediction data by a $\circ$.

In plots a) and c) the input data and the prediction data using the *Global AR Model (least square)* with order $l = 3$ is shown. Plots b) and d) show the detail plots of the prediction and the data for the last 5% of the motion.

In plots e) and g) the input data and the prediction data using the *Local AR Model (maximum likelihood)* with order $l = 3$ is shown. Plots f) and g) show again the details.

Finally, in plots i) and k) input data and prediction data of the *Nearest Neighbor* prediction algorithm are shown, whereby the parameters m, k, l, M were set up accordingly (e.g. m $=$ 4, k $= 5$, l $= 3$, M $= 20$ for plots in Figure 5.21. For more explanations on these parameters see [Sel00]).

It can be seen that for the input data provided in plots of Figure 5.19 both ARM methods provide very good prediction results whereas the Nearest Neighbor method has problems predicting the parabola accurately. For data of plots in Figure 5.20 and Figure 5.21 the *Global AR Model (least square)* and the *Nearest Neighbor* provide very good results wereas the *Local AR Model (maximum likelihood)* fails predicting the periodic input data (see plots g) and h)). Finally, for the more irregular and non-periodic motion in Figure 5.22 the prediction more or less fails for all prediction methods.

Comments: Summarizing it can be stated that linear curves, curves having low order and highly periodic curves or periodic curves multiplied with linear curves (actually all linear combinations of those curves) provide acceptable to very good results.

Due to the obtained results it was decided to use a "derivative" of the *Global AR Model* method applied for real image data. This method, named AARM was described in Section 4.5.1, and some prediction results are shown exemplarily in the following. This method was also used for the performed catching experiments!

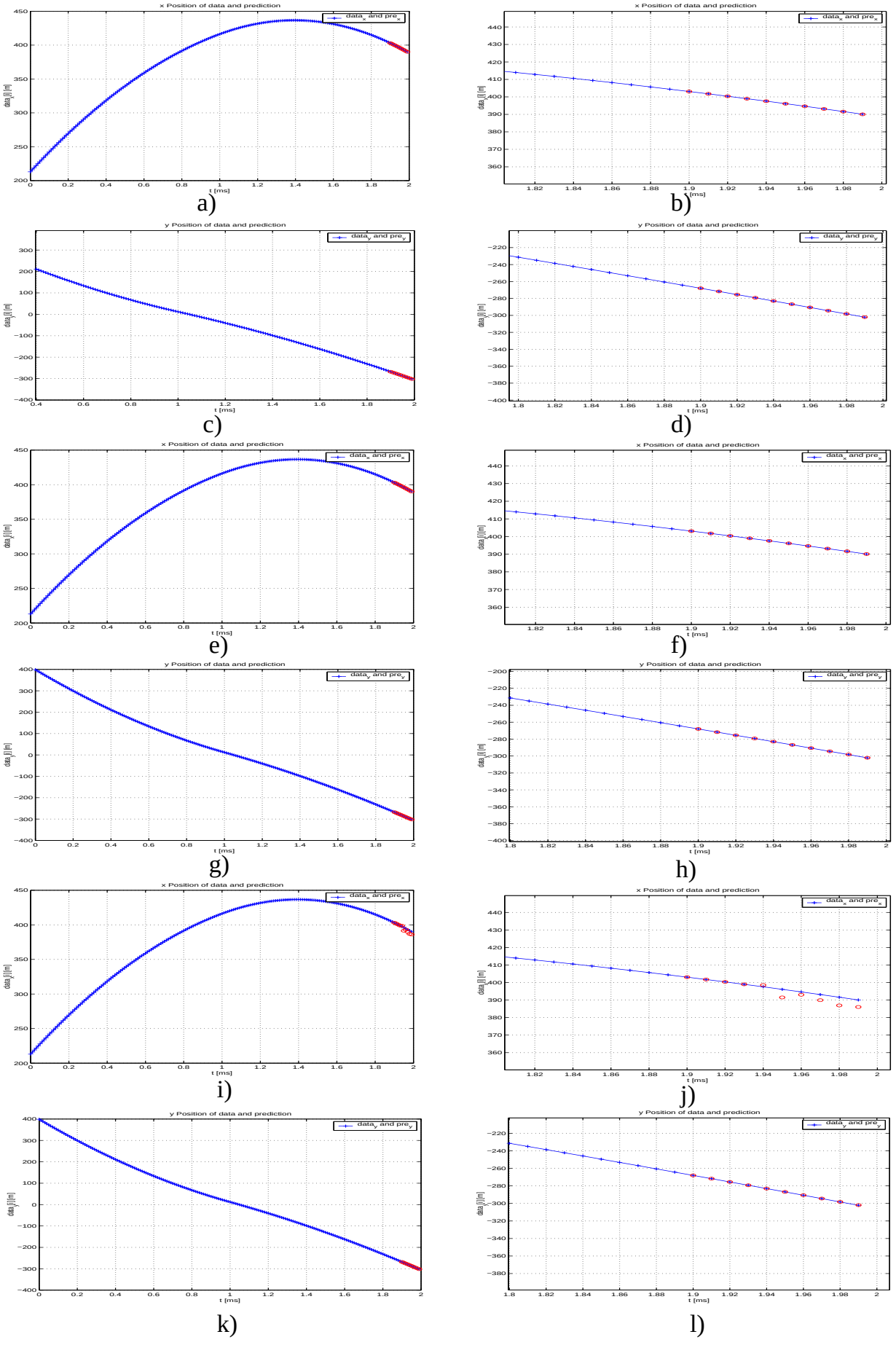


Figure 5.19: Prediction of an artificial movements. + indicate data, ○ indicates a predicted point. For further information see text!

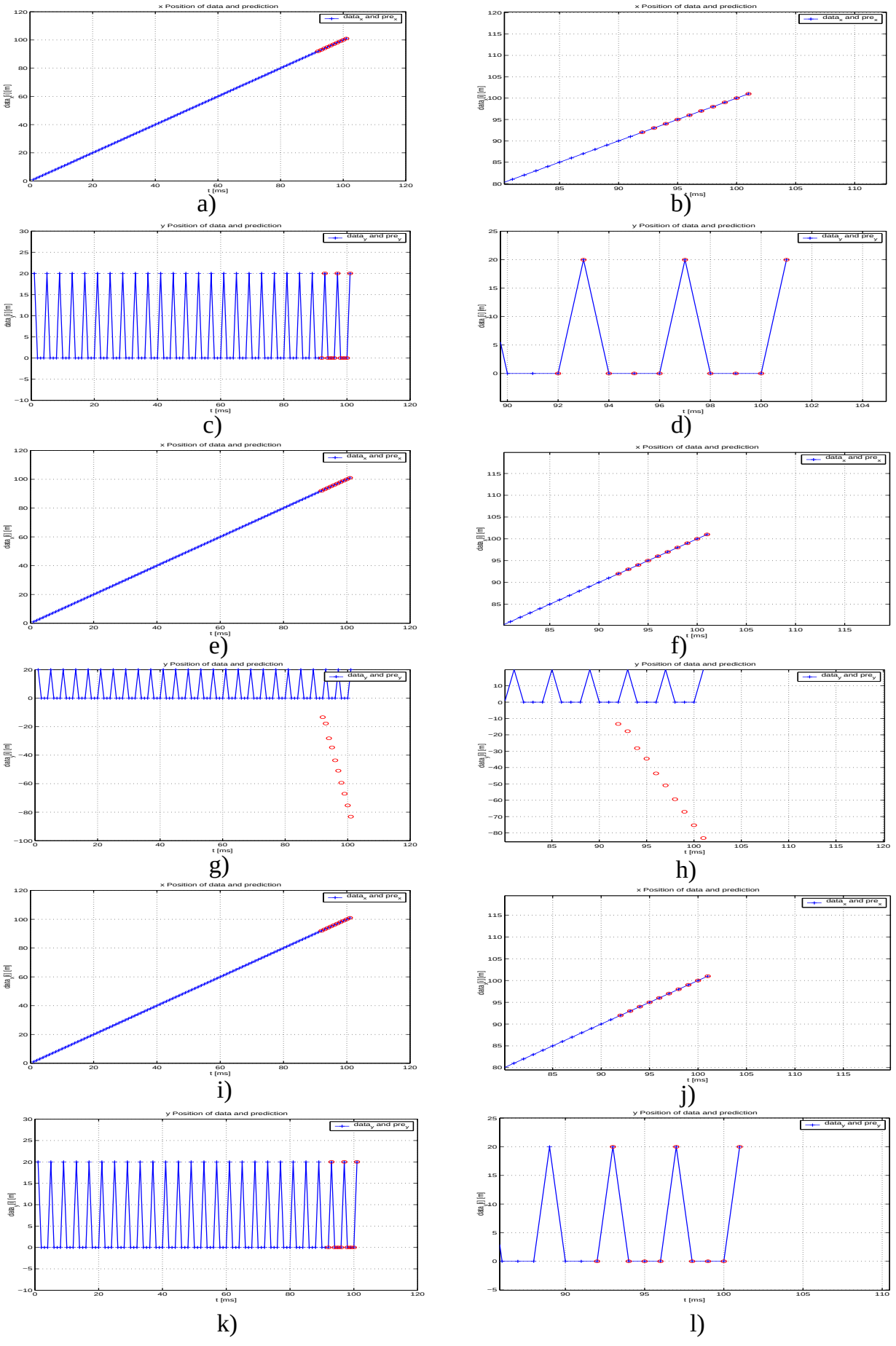


Figure 5.20: Prediction of artificial movements. + indicate data, ○ indicates a predicted point. For further information see text!

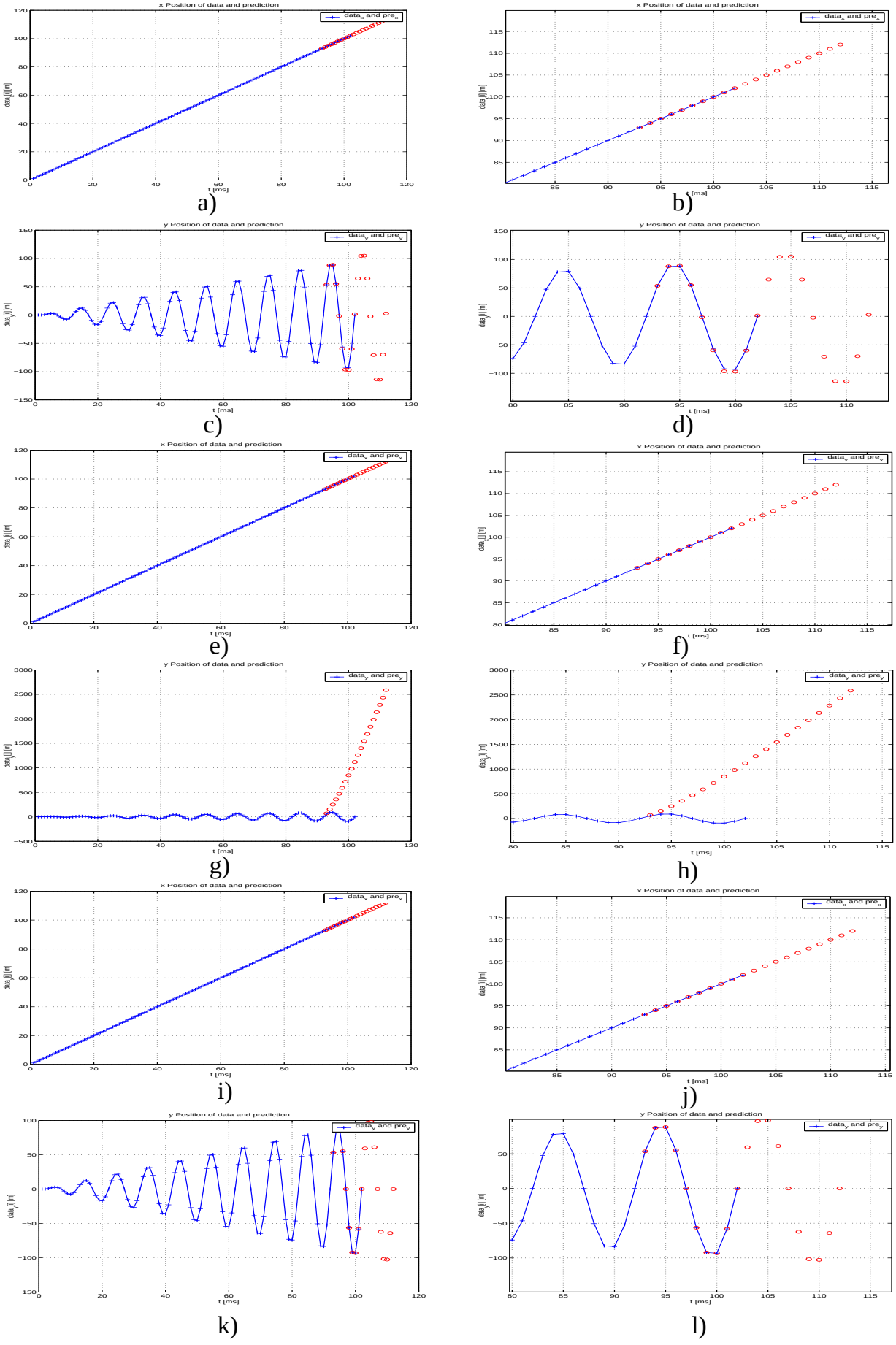


Figure 5.21: Prediction of artificial movements. + indicate data, ○ indicates a predicted point. For further information see text!

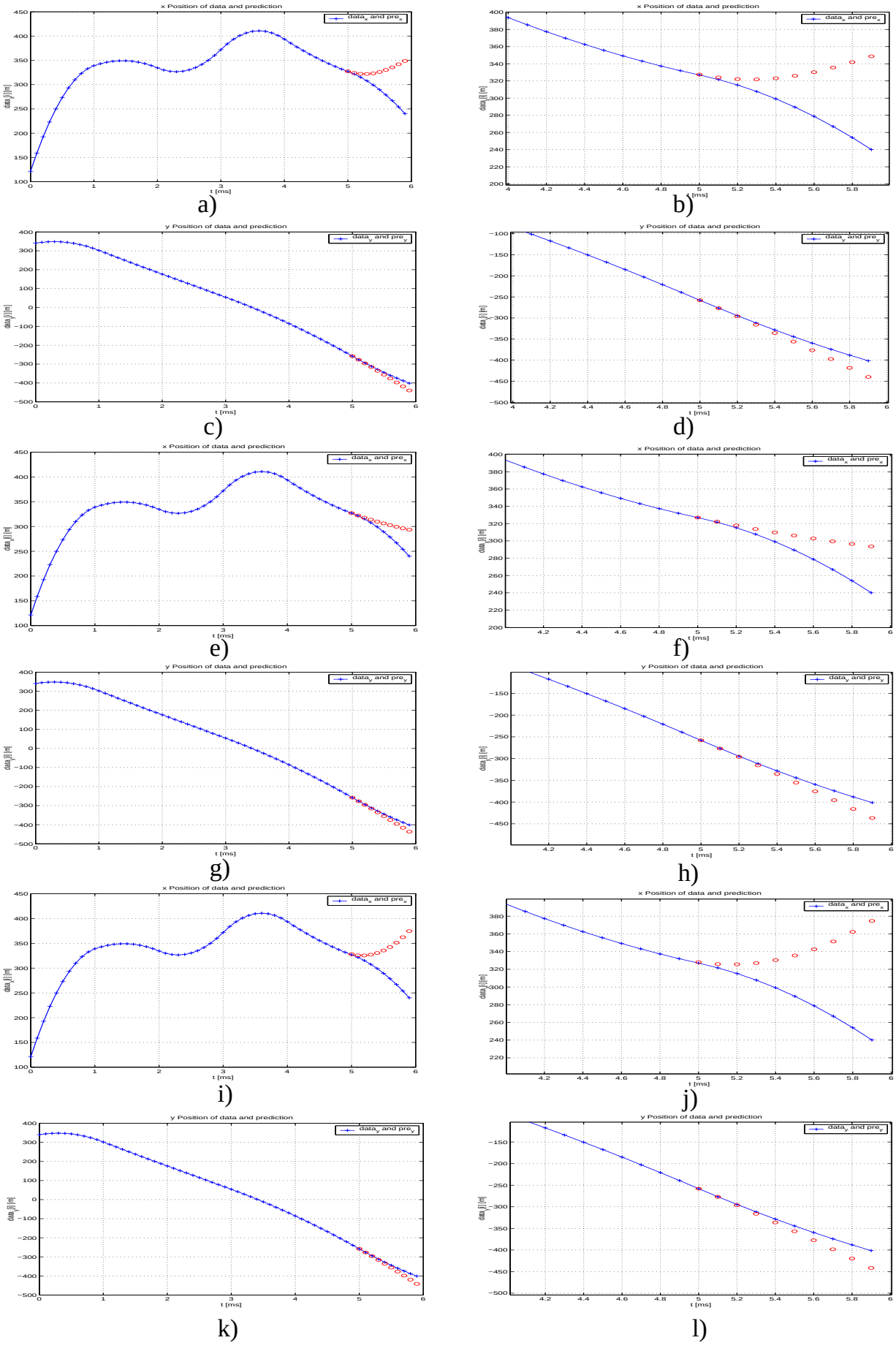


Figure 5.22: Prediction of artificial movements. + indicate data, ○ indicates a predicted point. For further information see text!

5.2.2 Real Tracking: Average ARM

Test conditions: To evaluate the Average ARM algorithm two experimental setups were chosen. In the first setup the model train moved on a table in front of the robot on a *linear* railroad. Predictions were made regularly while tracking continued (plots a) - d) in Figure 5.23). The second setup was tracking and prediction of a tennis ball moved "by hand" in front of the camera (plots e) -h) in Figure 5.23).

Results: In plot a) three prediction series are plotted. It can be observed from the plot that for the first prediction run only few tracked positions were available leading the prediction algorithm to assume a linear object motion with constant velocity. In the second prediction run of plot a) the acceleration of the tracked object is overestimated, leading the predicted position to "overshoot" the real position. Details to that second prediction run can be seen in plot b). In plots c) and d) another trial of the same experiment is shown.

Plots e) and g) show the output of tracking and prediction of the tennis ball (plot f) and h) show details again). Thereby circular motions are performed whose radii are continuously changing while the frequency of the sinus waves and therefore the speed of the object is approximately constant.

Mean errors were not evaluated for these trials, since only the absolute positional error for the predicted point of contact is of interest in the robotic catching experiments.

Comments: This algorithm was used later for prediction of the target position for all "real robot" catching experiments described later (see Section 5.4). Thereby further examples can be seen. It can be stated already at this point, that for many experiments the accuracy of the prediction was sufficient to finally catch the target object.

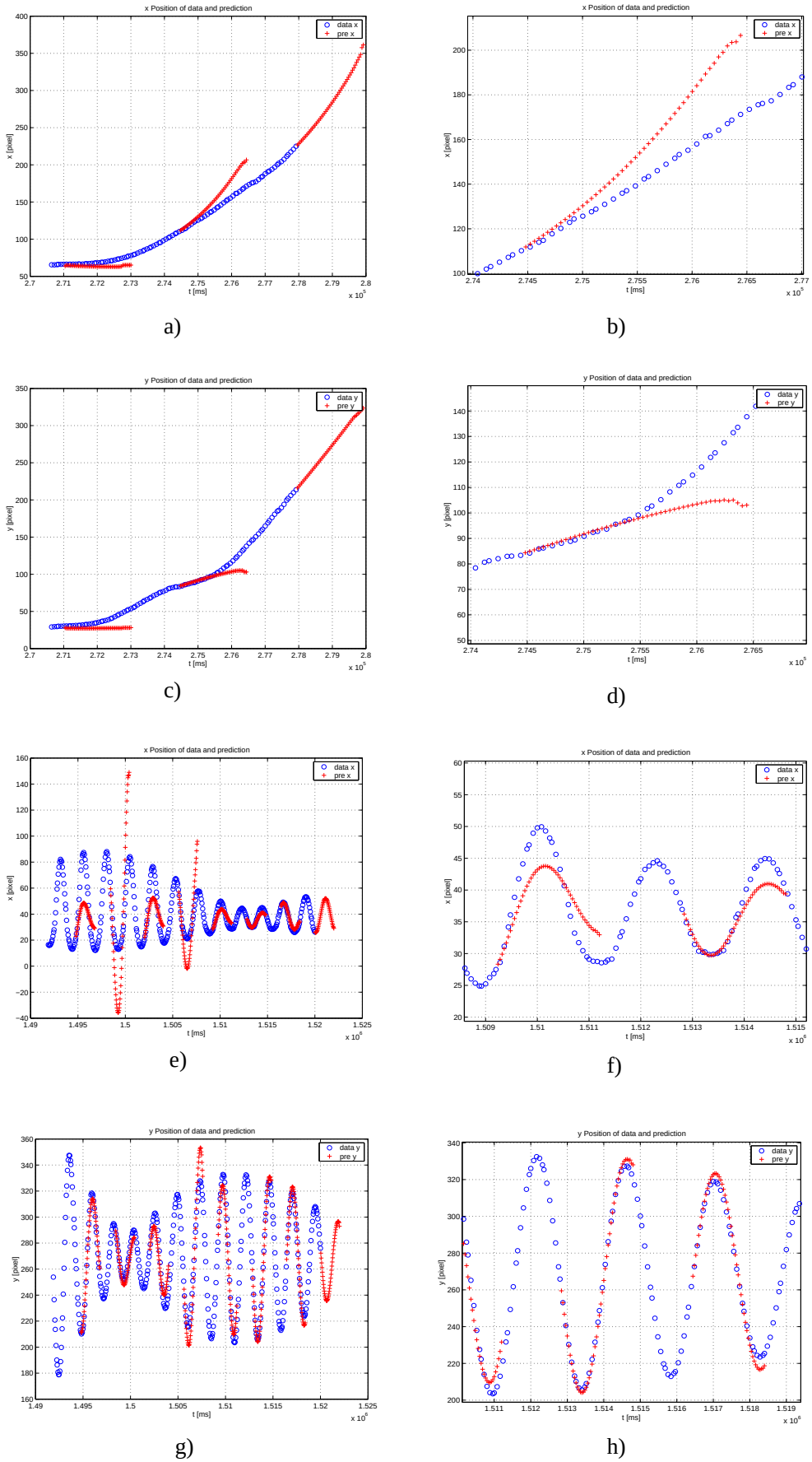


Figure 5.23: Prediction of different motions. All plots show "position over time". a) and c): x, y position of a linearly moving model train. b) and d) detail plots of a) and c), respectively. e) and g): x, y position of a tennis ball moved in front of the camera. f) and h) details of e) and g), respectively.

5.3 Simulation of Hand-Target Interaction

Test conditions: In these experiments only the control of the end-effector position was evaluated using *MATLAB* and *Simulink*. Control of orientation is analogous and can be derived from the preceding chapter. Different kind of target trajectories were tested (see [Bea01]). Here exemplarily the following trajectories are shown: *an approaching target and an escaping target.*

5.3.1 Control of Position

Results: Figure 5.24 shows the 3D target and end-effector trajectories for the trial with the *escaping* target, plots in Figure 5.25 show details of the trial. Thereby in plot a) the x, y and z position over time for end-effector and target trajectory can be seen. The dashed horizontal lines show the beginnings of the different motion phases (specified in plot b). It can be observed that the end-effector first approaches the target object quickly, then adapts its trajectory to the target trajectory, gets into contact with the target and finally follows it for a while. This is in synchrony with the theory described in Section 4.7.

Plot b) shows the timely course of the "Interaction Points" together with the starting times of the sub-phases indicated by the dashed horizontal lines.

Plot c) shows the *desired* end-effector positions or intermediate targets over time, which actually are the input to the motion control unit. It can be observed that for this trial every $250ms$ a new intermediate target is generated (having a duration of $500ms$, see plots d)-f)).

Plots d), e) and f) show the timely course of the Cartesian velocities of the target object and the end-effector. It can clearly be seen that the end-effector velocity profiles are generated by a continuous superposition of bell-shaped sub-movement velocity profiles. Thereby the resulting velocity profiles correspond well to the observed velocity profiles of human subjects using a *reactive strategy* for similar trials (see Section 2.2.3). Additionally, the end point conditions are almost fulfilled in every coordinate, i.e. the position and velocity of target and end-effector match in the contact point (at $t \approx 4s$).

Figure 5.26 shows the 3D target and end-effector trajectories for the trial with the *approaching* target, plots in Figure 5.27 show again details of the trial. Detail plots show no significant differences to the escaping case.

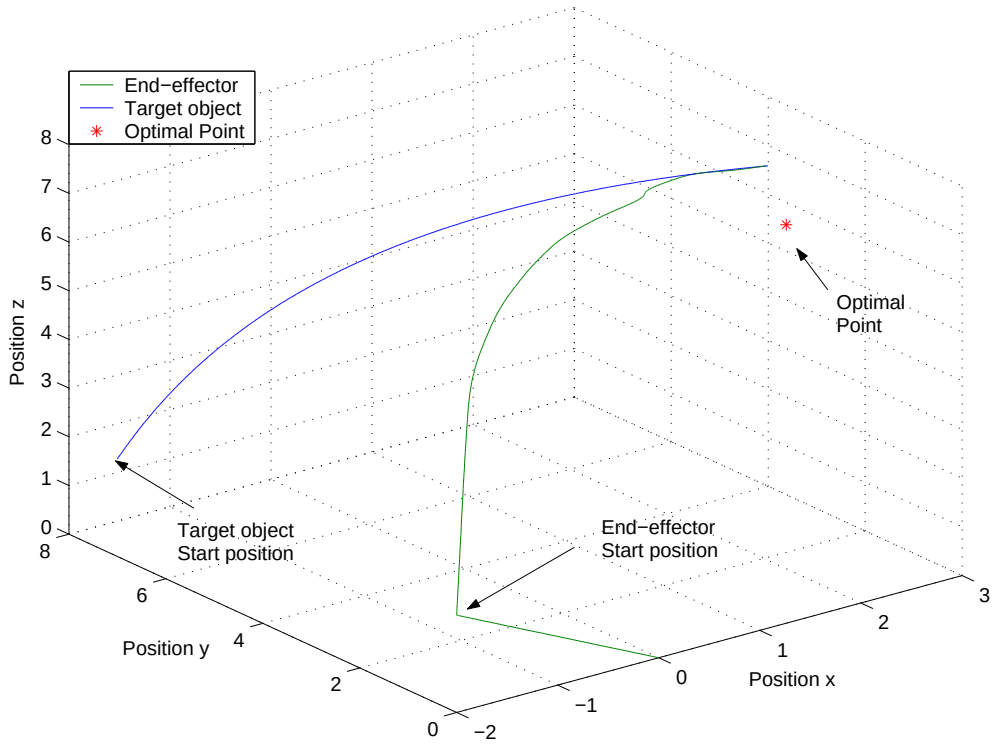


Figure 5.24: 3D Plot of End-effector Position and Target trajectory for a escaping target

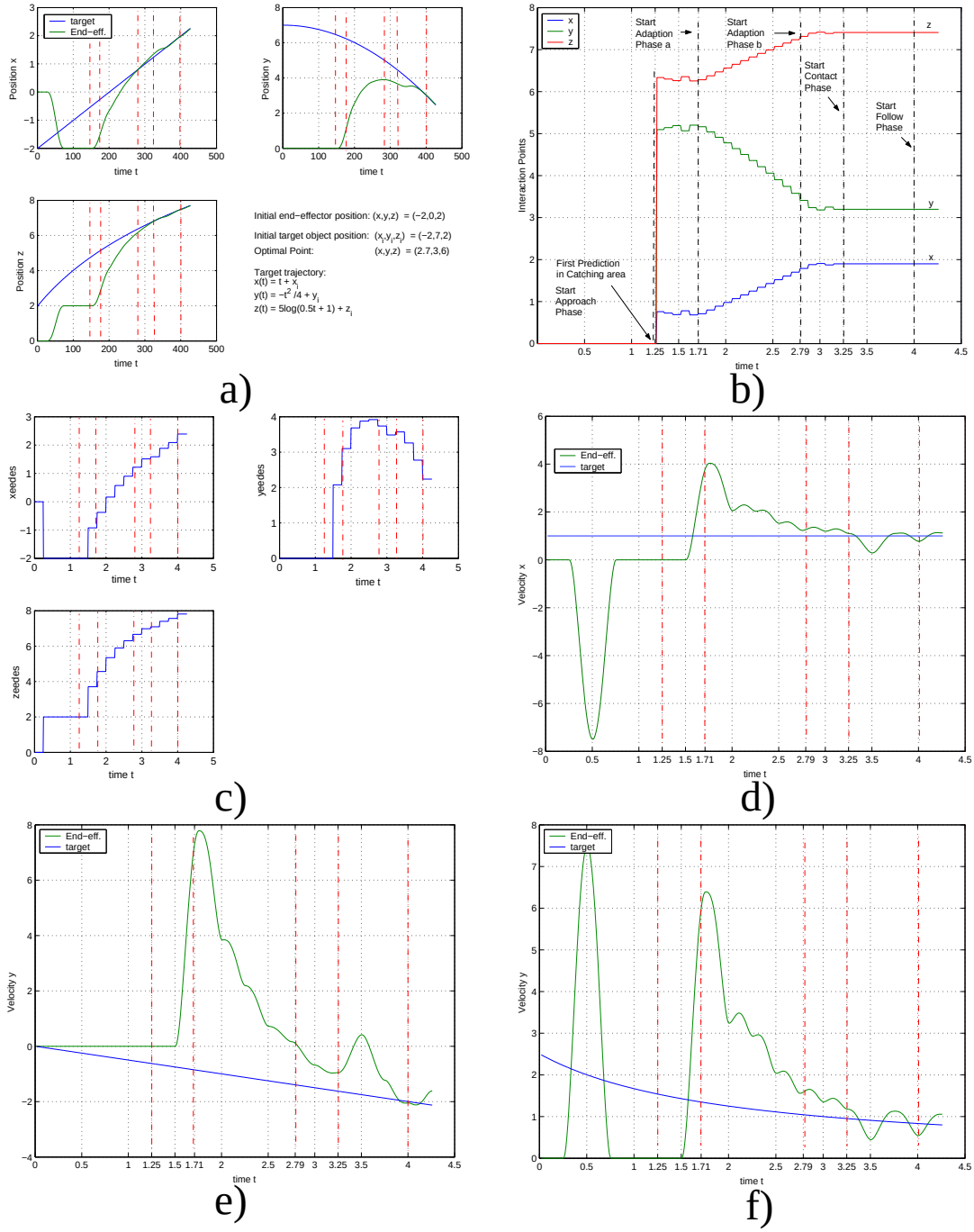


Figure 5.25: Data plots: a) x y z position of target and end-effector over time. b) Position of the possible x, y, z Interaction Points over time. Additionally the starting times of the different hand motion phases are drawn. c) The desired end-effector positions for the end-effector (= input for robot motion control). d), e), f) the x, y, z end-effector and target velocities. **Note:** The first velocity peaks in d) and f) correspond to the movement to the start position!

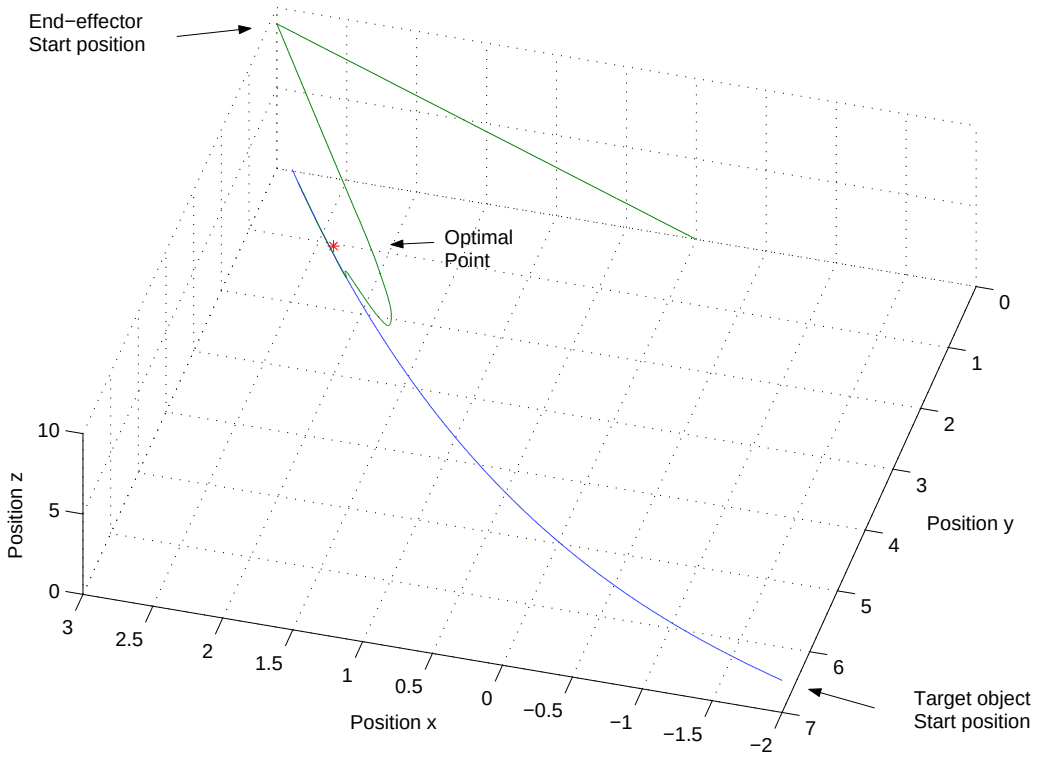


Figure 5.26: 3D Plot of End-effector Position and Target trajectory for a approaching target

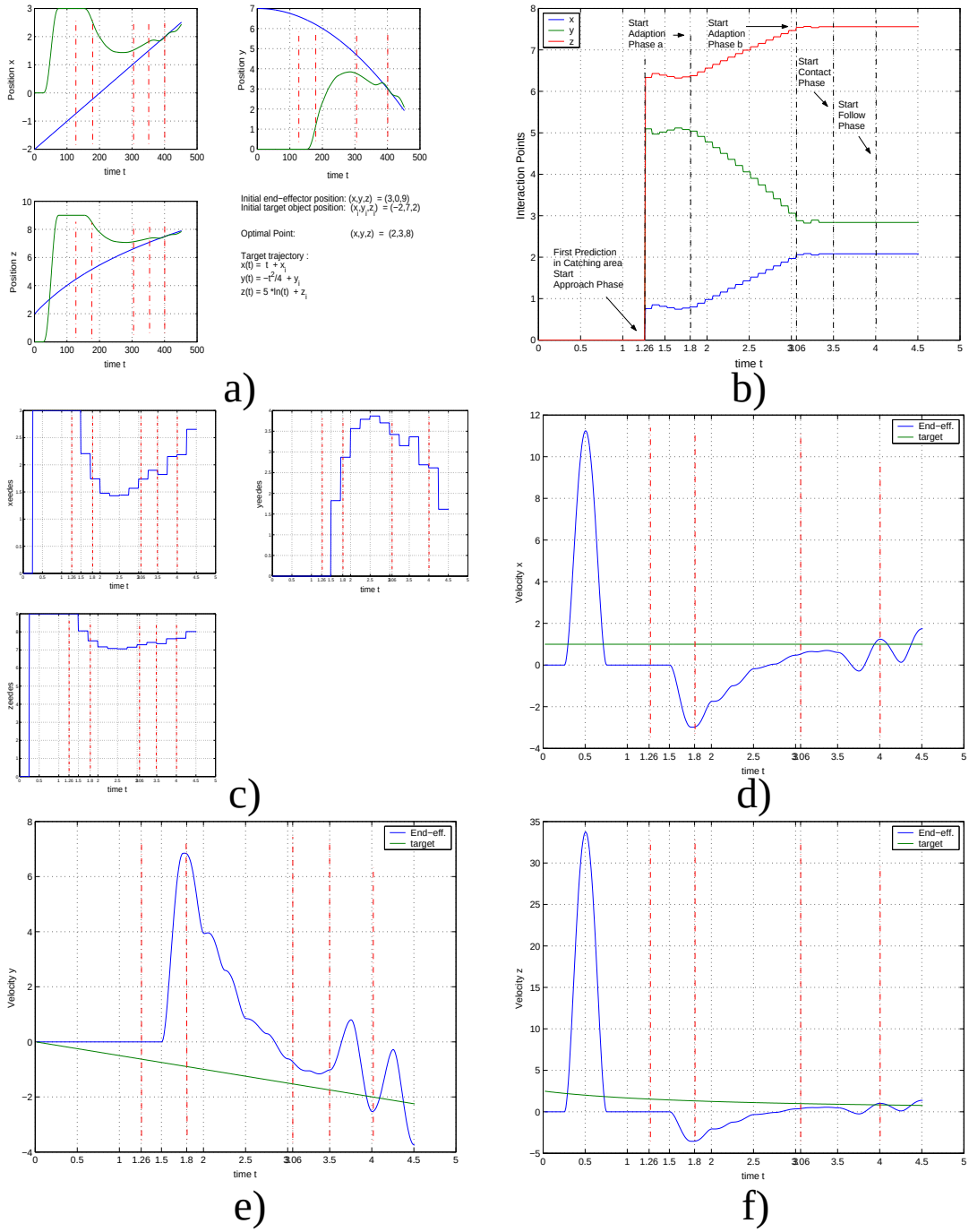


Figure 5.27: Data plots: a) x y z position of target and end-effector over time. b) Position of the possible x, y, z Interaction Points over time. Additionally the starting times of the different hand motion phases are drawn. c) The desired end-effector positions for the end-effector (= input for robot motion control). d), e), f) the x, y, z end-effector and target velocities. **Note:** The first velocity peaks in d) and f) correspond to the movement to the start position!

Comments: The motion of the end-effector behaves according to the expectations. It is shown that it is possible to interact with a moving target in a "human like" way. The resulting end-effector motions and velocity profiles are smooth. This observation allowed the implementation of the strategy on the robot. The corresponding experiments are described in the following.

5.4 Real Robot Experiments

5.4.1 Experimental Setup

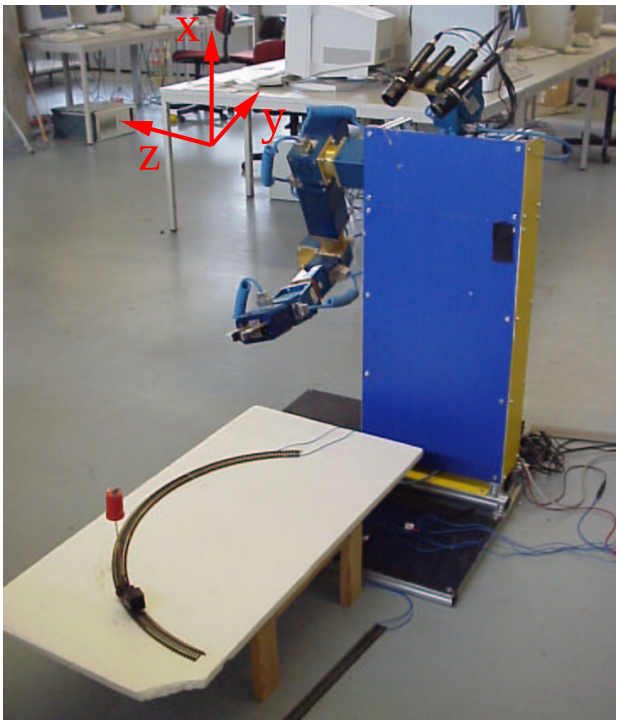


Figure 5.28: Experimental setup

Figure 5.28 shows the robot MINERVA together with the target object which is placed into the chimney of the model train[5]. With the railroads different target trajectories could be modeled, e.g. linear, circular, wave-forms etcetera. The table top is made out of Styrofoam to provide a soft obstacle for the robot arm in case of collisions. The train is controlled by a human operator. Since the workspace of the robot arm is very limited, the grasping area is limited to regions quite close to the robots body (see Section 5.4.1).

[5]The train itself was to chicken-hearted to serve as a target object ...

Experimental Validation of the workspace To assure that the grasping area is placed within the reachable area of the robot different experiments were performed. Thereby the orientation of the robot's end-effector was kept constant to a value that was used in the later catching experiments (this starting orientation can be seen in Figure 5.28)[6]. Plots in Figure 5.29 show the results of the evaluation. Different target positions laying on half circles were given as input to the robot. It can be observed that there are clear limits for the experiments: If the target is escaping from the robot it may not move far in z direction towards the robots base, since the robot would have problems to reach it there.

5.4.2 Control of Robots Position and Orientation

As final experiments before real catching experiments were started the tracking performance of the manipulator was tested. Thereby it was important to know if the manipulator can accurately follow a target trajectory. The target trajectory tested was simply a circle in the y-z plane (x constant), and at a constant time interval of $500ms$ successive points of the circle were transmitted to the manipulator as intermediate targets. Results of the test can be seen in Figure 5.30. It can be seen that the end-effector could follow the trajectory very accurately.

The reason for this experiment was to assure that the robot moves to a transmitted target accurately in *time and space*. Thereby it was assured that in case of a failed catching trial (see e.g. Section 5.4.3) the responsibility for failure does not lie in inaccurate robot movements.

In the same sense the control of the end-effector's orientation trajectories was evaluated. Table 5.7 shows the configuration parameters for the trial. Table 5.8 shows the target coordinates for the trial. Thereby the position and orientation of the end-effector is changed during the trial. Figure 5.31 shows the according trajectories. It can be observed that all trajectories are smooth and have the expected form.

5.4.3 Hand-Target Interaction: Grasping a Linear Moving Target

5.4.3.1 Escaping Target

Test conditions: To test all algorithms as a "working whole" the setup with the moving model train was chosen. In the first experiments the model train moved on a straight railroad and was driving away from the robot as can be observed in Figure 5.34 and the plots in Figure 5.32. The train was controlled by a human operator who chose the appropriate speed. To simplify image processing (and reduce processing time) simple color tracking

[6] As mentioned, the orientation of the target object is not determined by image processing. A fixed end-effector orientation is therefore reasonable.

Table 5.7: Configuration parameter

$\Delta t\ [ms]$	10
Length $s\ [m]$	-0.37
Length $t\ [m]$	-0.31
Length $u\ [m]$	-0.24
k_l	0.1
k_s	0.01
$\epsilon_p\ [m]$	0.002
$\epsilon_v\ [\frac{m}{s}]$	unused
$\epsilon_{rp}\ [rad]$	0.0035
$\epsilon_{rv}\ [\frac{rad}{s}]$	unused
Feedback $\mathbf{q}_{measured}$	unused
additional movement duration $\Delta T\ [s]$	0

Table 5.8: Example: targets

	${}^w\mathbf{p}$	${}^w\mathbf{a}$	${}^w\mathbf{n}$	$T[s]$
Start position	$\begin{bmatrix} -0.357 \\ -0.557 \\ 0.02 \end{bmatrix}$	$\begin{bmatrix} 0.05 \\ -0.998 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0.95 \\ 0.049 \\ 0.31 \end{bmatrix}$	–
Target 1	$\begin{bmatrix} -0.37 \\ -0.70 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0.707 \\ 0 \\ 0.707 \end{bmatrix}$	4
Target 2	$\begin{bmatrix} -0.37 \\ -0.60 \\ -0.20 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.707 \\ -0.707 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	4

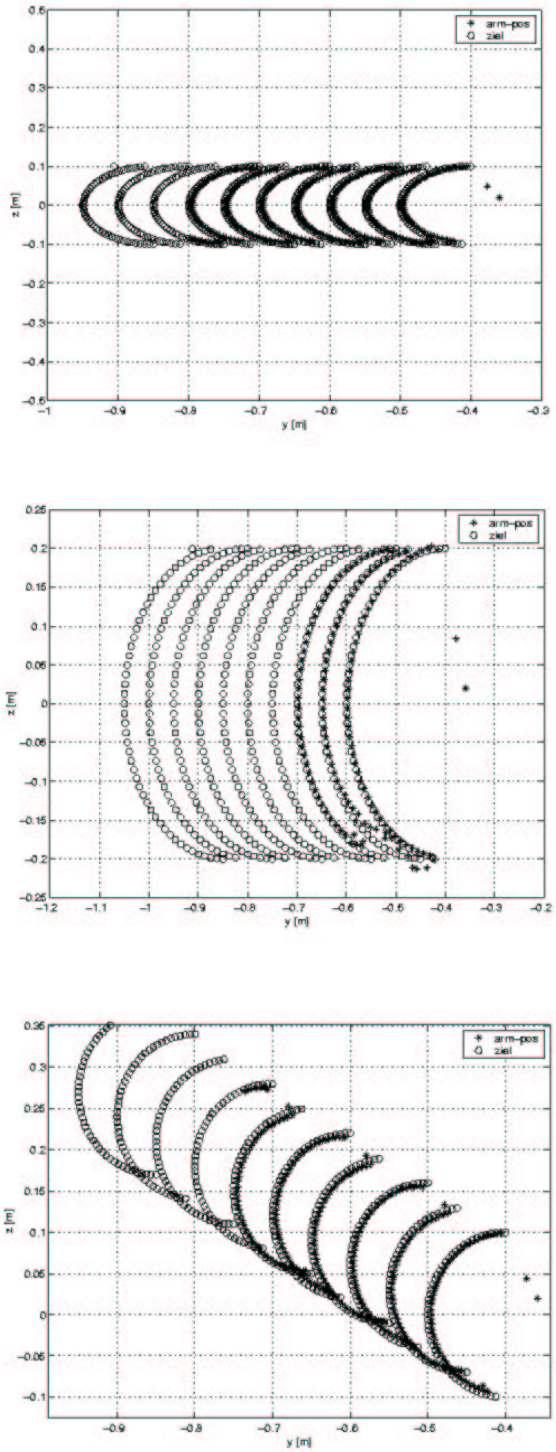


Figure 5.29: Workspace evaluation

using the CubeUV filter was performed. From the tracked color blob the area center was determined and its 3D position calculated. This 3D position was transmitted to the AARM prediction algorithm which predicted the 3D position of th model train $2s$ in the future, every time when a new 3D position arrived. From these predictions "Interaction Points" and intermediate targets were derived using the algorithms as described in Section 4.7. The intermediate targets were then transmitted to the motion control unit. More details on the concrete timing can be found in Section 4.8.

Results: Results can be seen in Figure 5.32. There the x, y, and z position of end-effector and target object are plotted. In the y-z plot the catching area is indicated by the large circle (in 3D the catching area is a sphere, respectively). The predicted time of contact is indicated by the vertical line crossing the trajectories. It can be observed that the end-effector approaches the target trajectory successively. In the trial no real adaption phase occurred. The according condition was only valid for a short time[7]. In the contact phase the trajectory of the end-effector and the predicted trajectory of the target object match at the contact point in all three coordinates within the desired accuracy. In the follow phase the end-effector tracks the target long enough to finally close the gripper.

The verification that the prediction of the target trajectory and the calculation of the intermediate targets is accurate enough for catching can be seen from the movie sequence in Figure 5.34.

[7]This was determined from analysis of a LOGFILE that was generated during the trial.

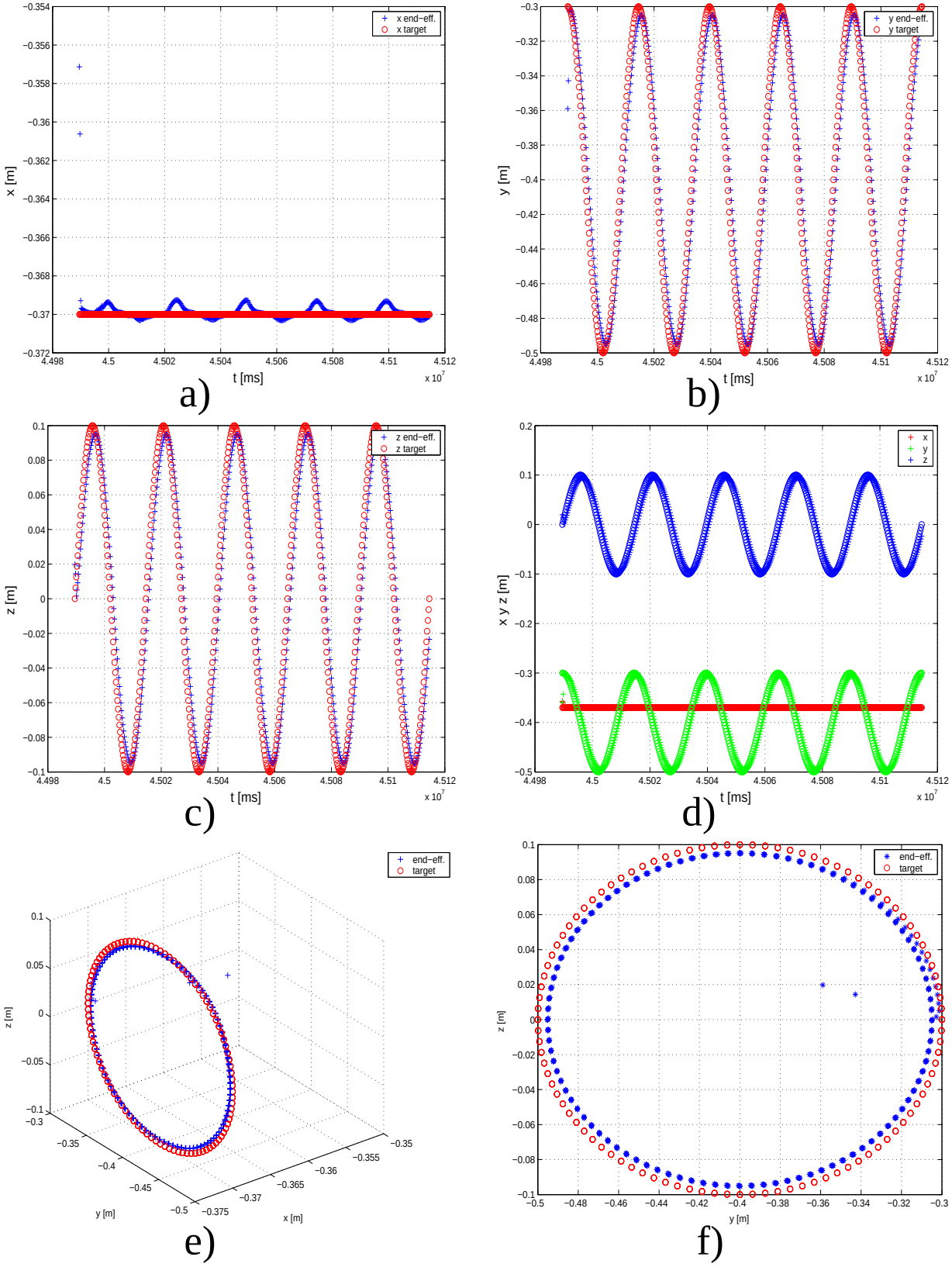


Figure 5.30: Tracking a moving target with the robots end-effector

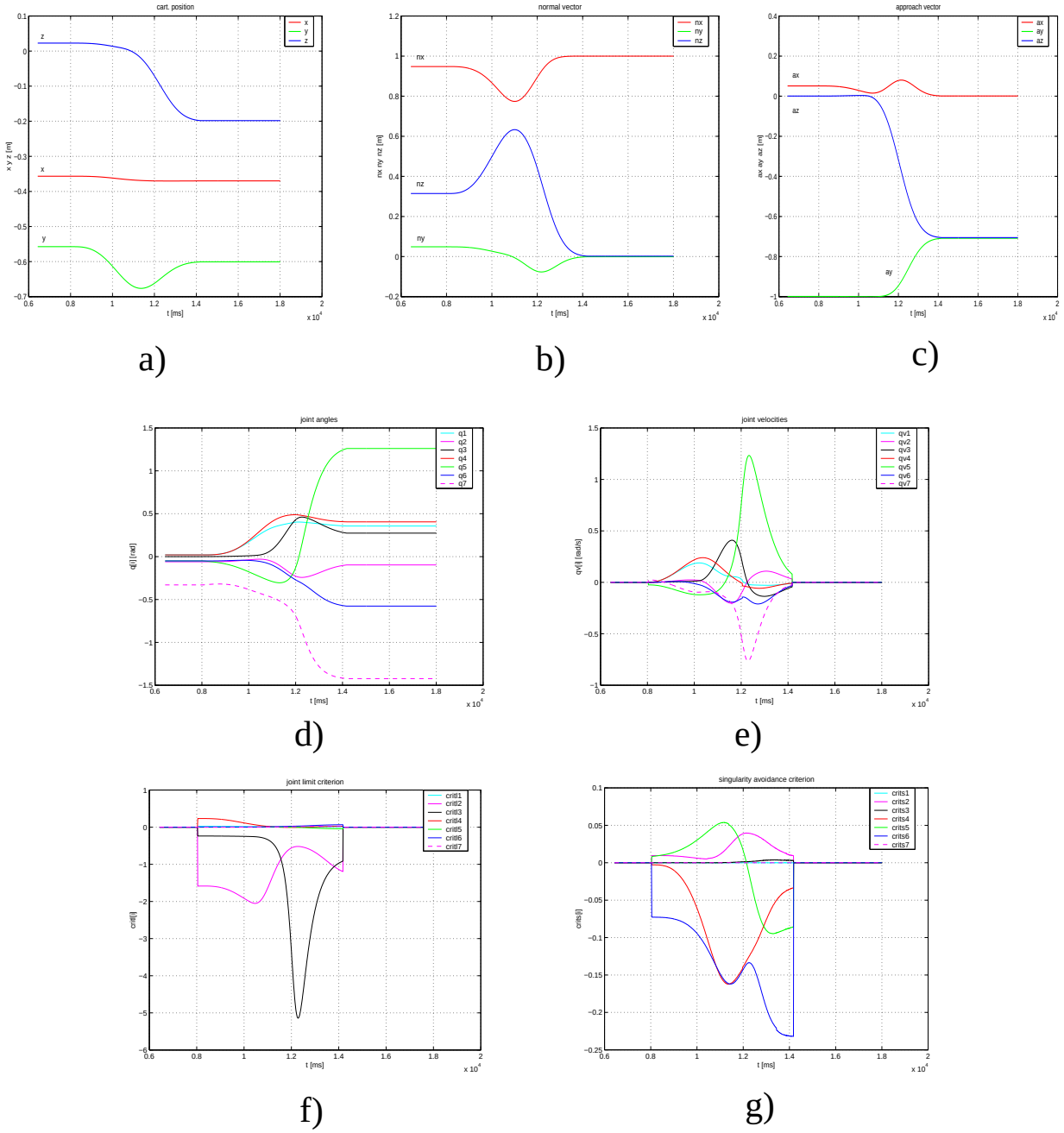


Figure 5.31: Example of controlling the position and the orientation of the robot's end-effector

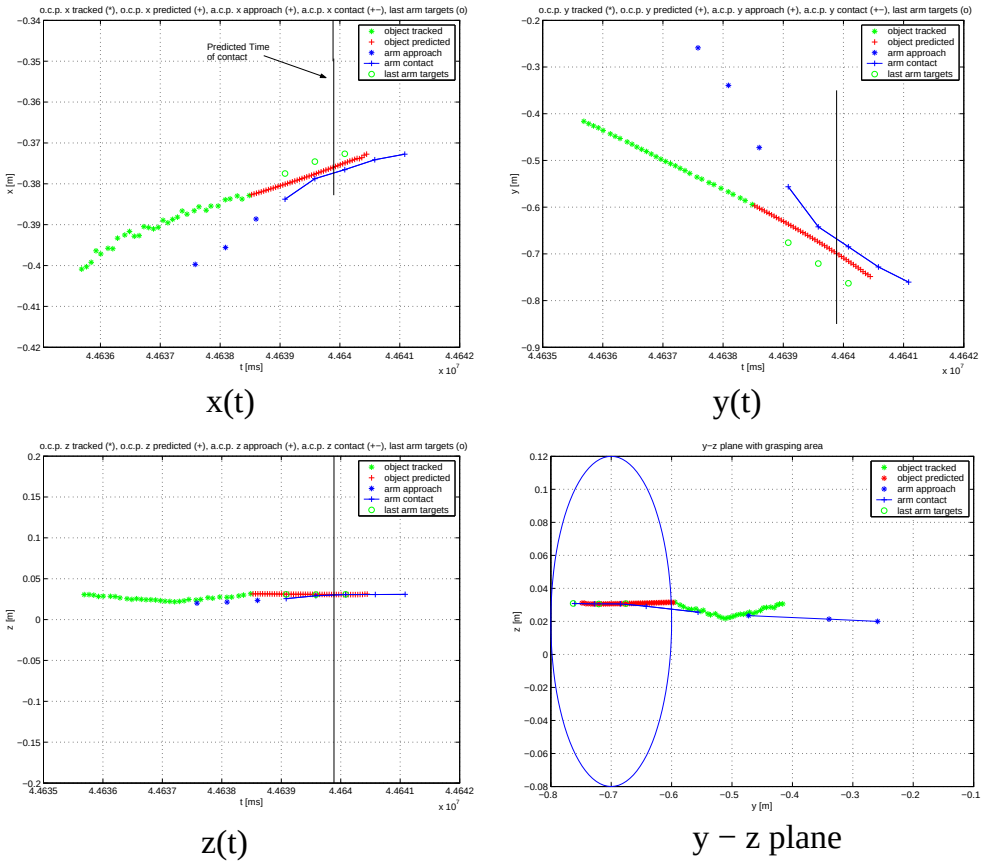


Figure 5.32: Plots of the trial of Figure 5.34. The time of a sub-movement was set to $Ts = 1000ms$

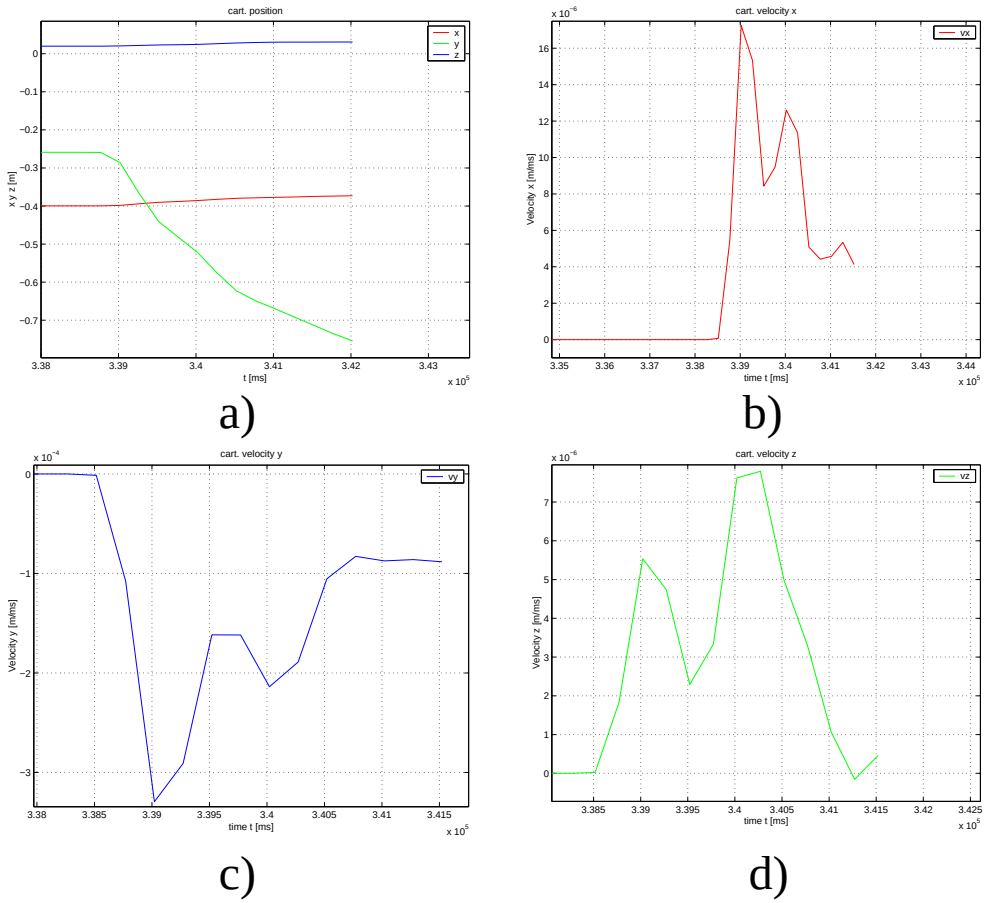


Figure 5.33: Position and velocity plots of end-effector as calculated by the dsp for the trial of Fig. 5.34: a) position x, y, z; b) velocity x; c) velocity y; d) velocity z. Time of a sub-movement $Ts = 1000ms$

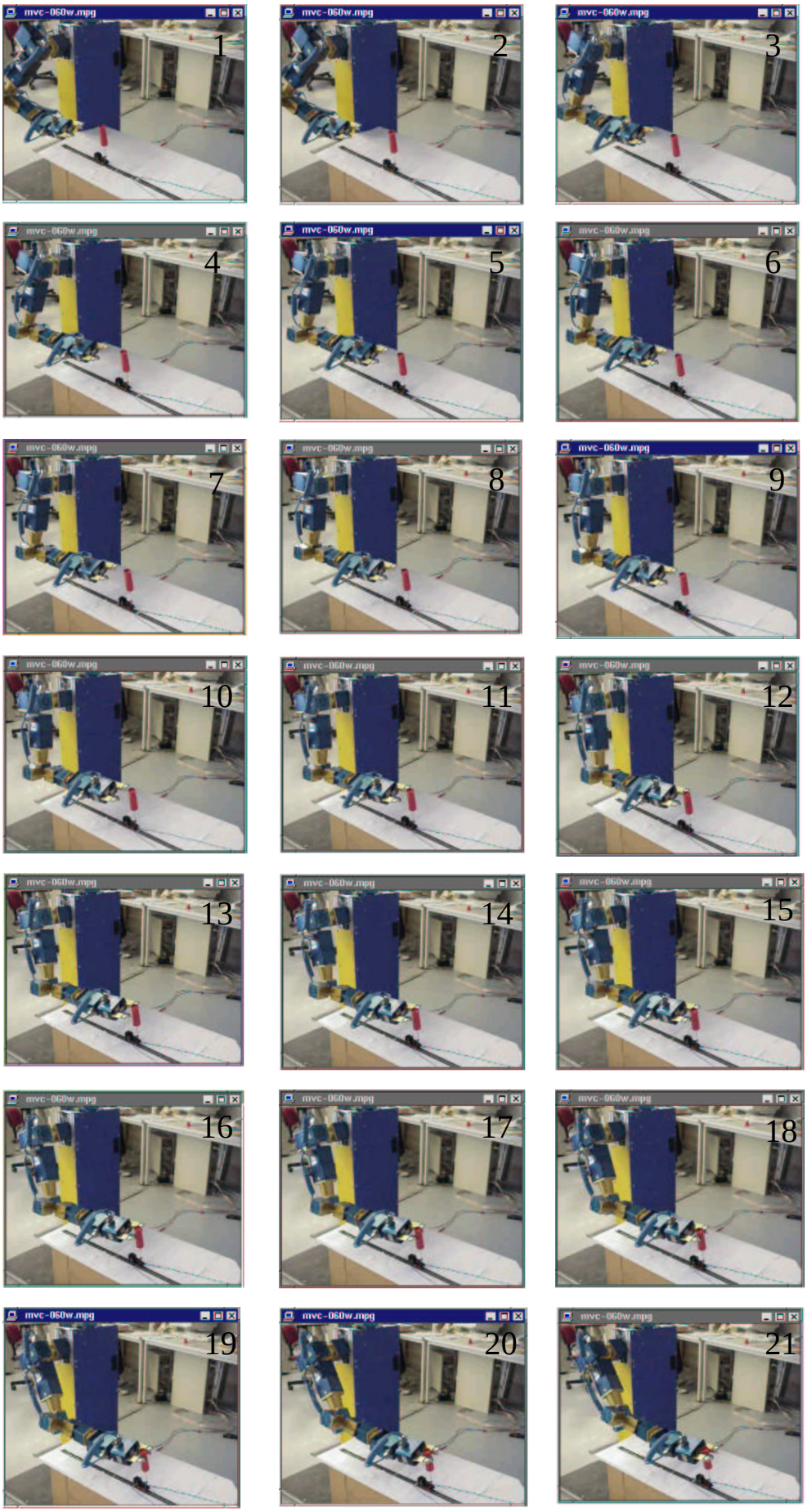


Figure 5.34: "Movie" of a linearly escaping target. Time of a sub-movement $Ts = 1000ms$

5.4.3.2 Approaching Target

Test conditions: The test conditions were the same as for the escaping target. Note: in this trial the catching area is placed at a position more suitable for an approaching target.

Results: Results can be seen in the plots of Figure 5.35 and Figure 5.36. Again x, y, and z position of the end-effector and tracked or predicted target position or plotted. It can be observed that end-effector and target approach each other well until the contact occurs. After that end-effector trajectory and the (predicted) target trajectory diverge. Taking a closer look at the "last arm targets", indicated by the small circles, reveals the cause: the third circle (with respect to time) lies "far away" from the predicted target trajectory. The reason for that was not an algorithmic error, but a security check: the *calculated* arm target was outside the catching area that was determined in Sec. 5.4.1 and therefore (by default) replaced by the values of the first arm target.

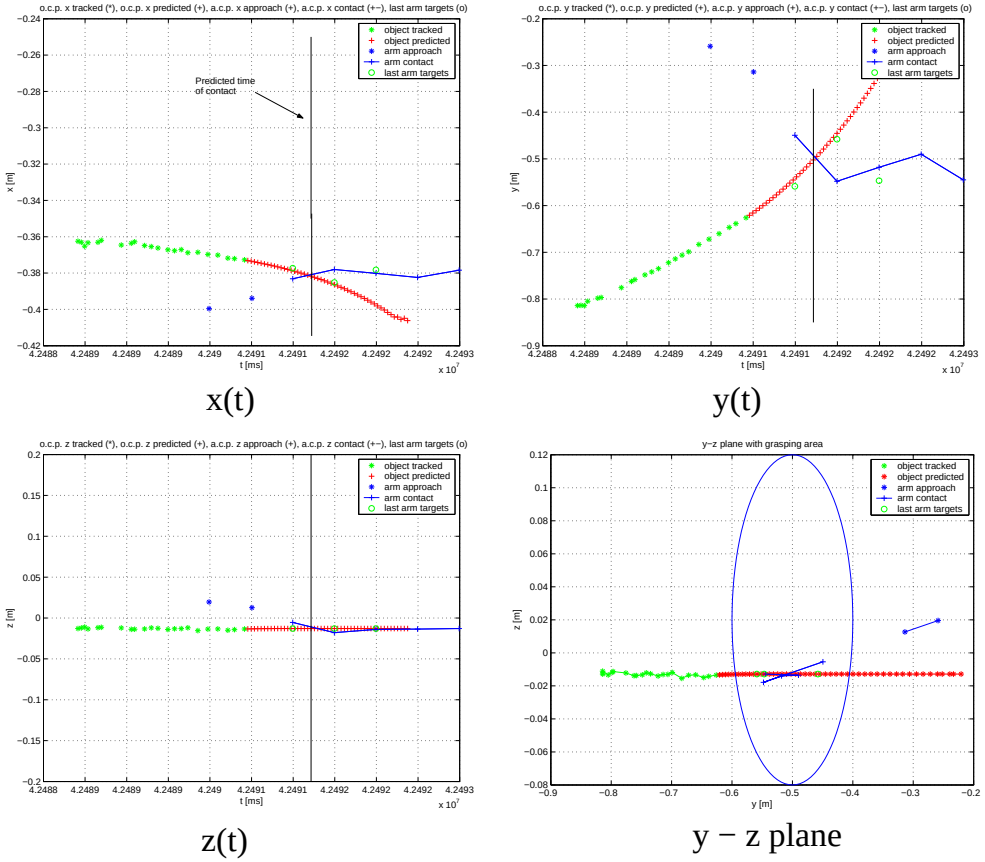


Figure 5.35: Plots of the trial shown in Figure 5.36. Time of a sub-movement $Ts = 1000ms$

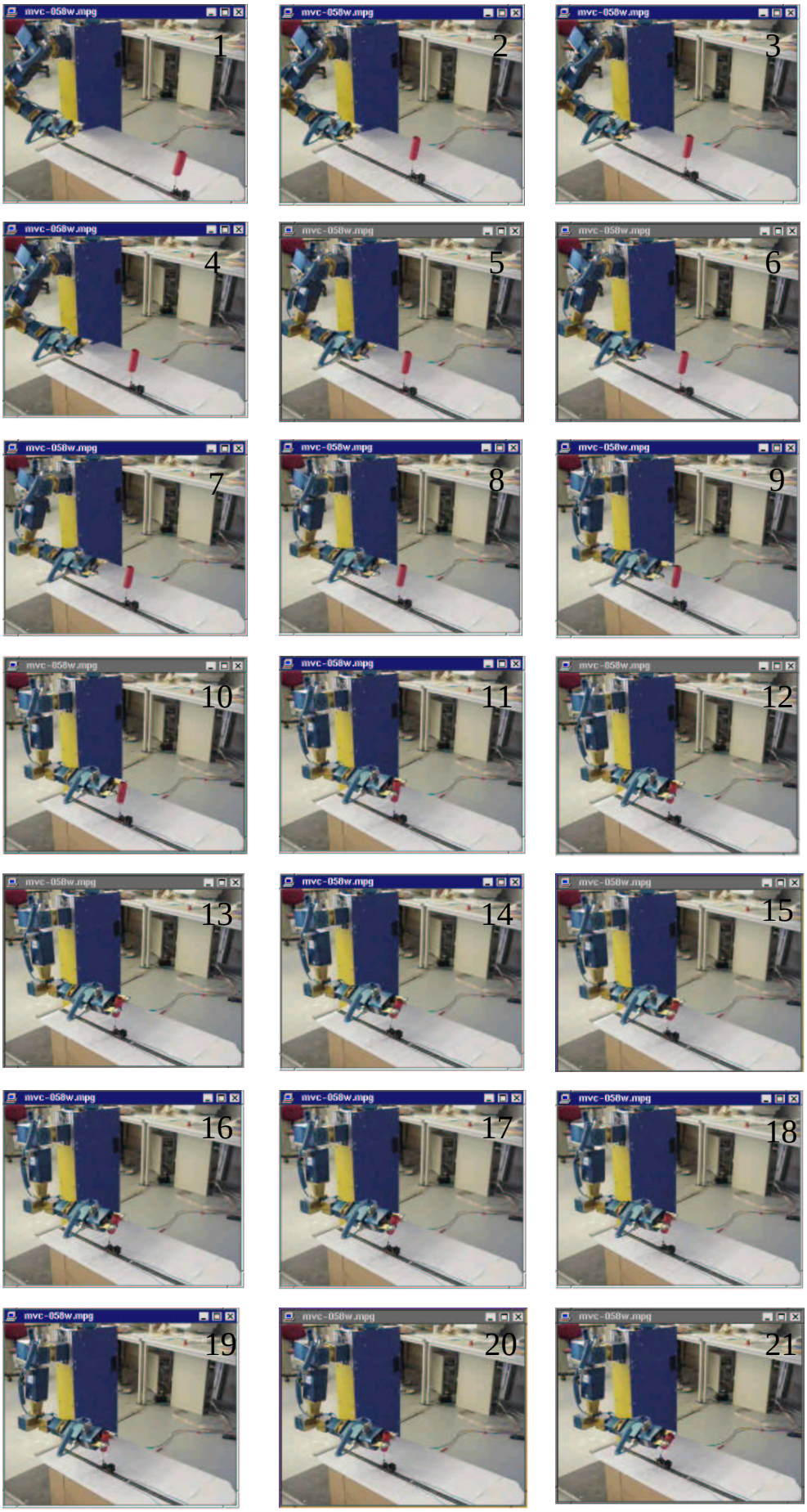


Figure 5.36: Linearly Approaching target. $Ts = 1000ms$

5.4.3.3 Tangential Target

Test conditions: In this trial the railroad was placed in a way that the train passed the manipulator tangential as can be seen in Figure 5.38. Note again the "suitable" placement of the catching area.

Results: Results can be seen in plots Figure 5.37 and Figure 5.38.

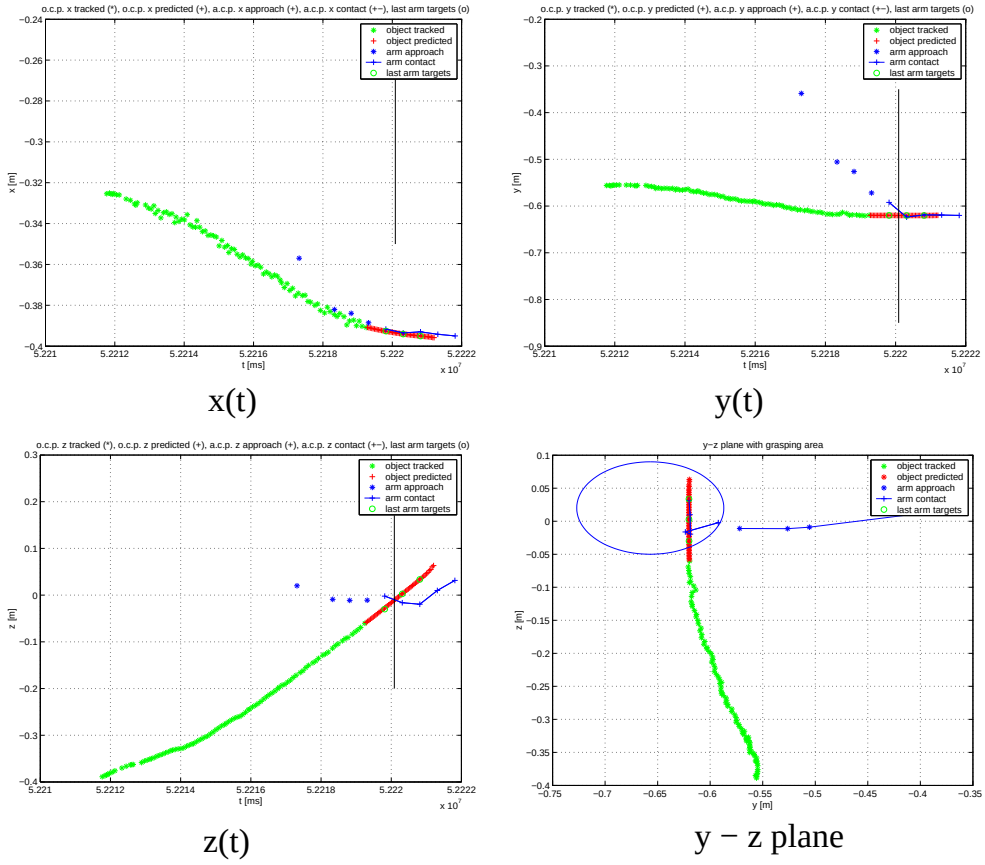


Figure 5.37: Plots of the trial shown in Figure 5.38. Time of a sub-movement $Ts = 1000ms$.

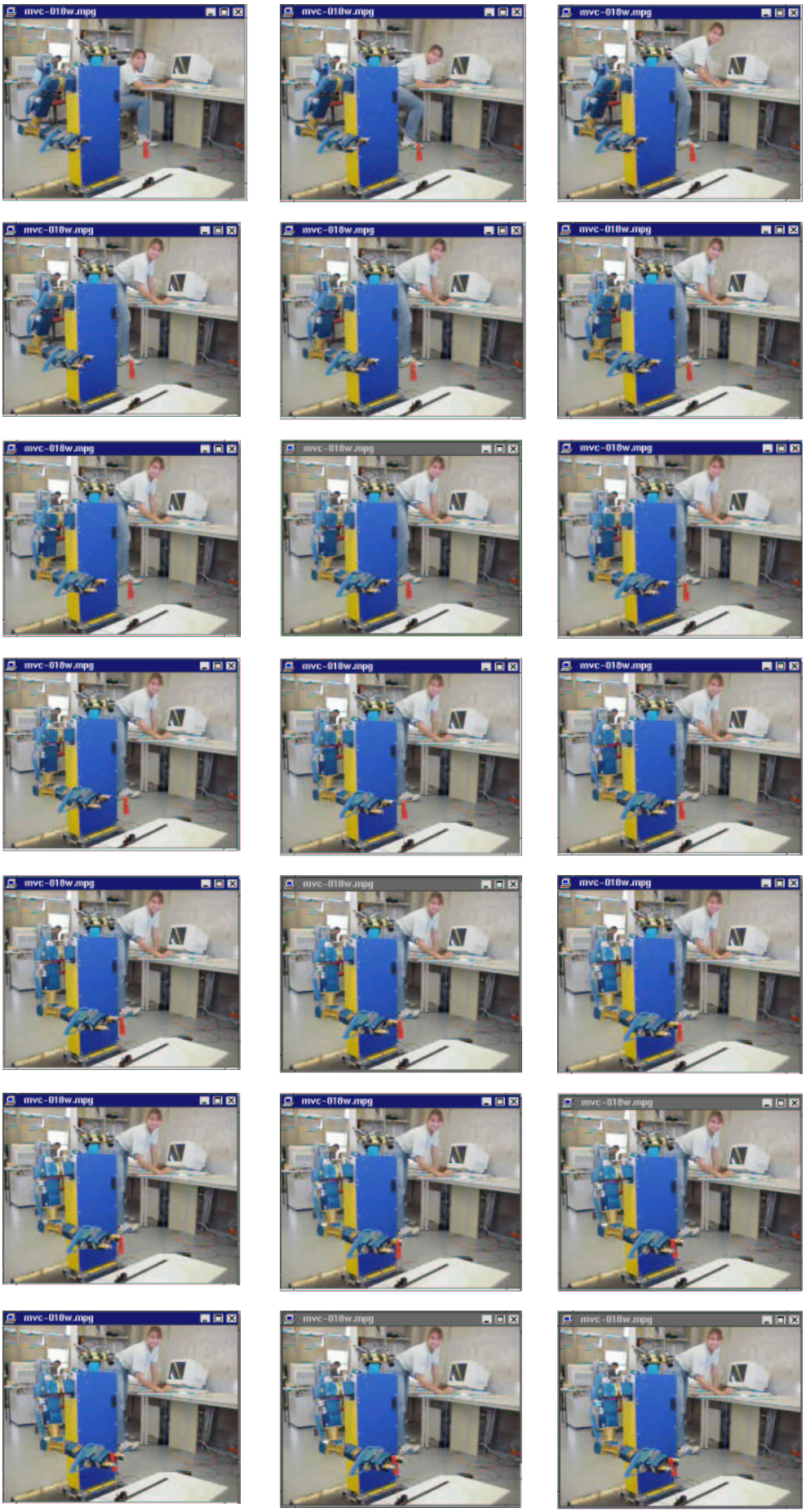


Figure 5.38: Tangential moving target.

5.4.4 Hand-Target Interaction: Grasping a Circular Moving Target

5.4.4.1 Approaching Target

Test conditions: In this trial the model train was placed on a small turning table driven by a belt. The table rotated several times to give the AARM prediction algorithm enough time and data to perform reliable long term predictions. The turning direction of the table was set such that the object approached the manipulator.

Results: Results are plotted in Figure 5.40 and Figure 5.39. It can be observed that in case of noisy tracking data (x coordinate!) the predicted position is qualitatively worse than for well tracked position data (y and z position). Actually there was no real "up and down" motion of the object like implied by the $x(t)$ plot. This "up and down" was mainly caused by vibrations of the turning table and image processing inaccuracies. Nevertheless the prediction of the circle was well and again accurate enough for the manipulator to finally catch the object.

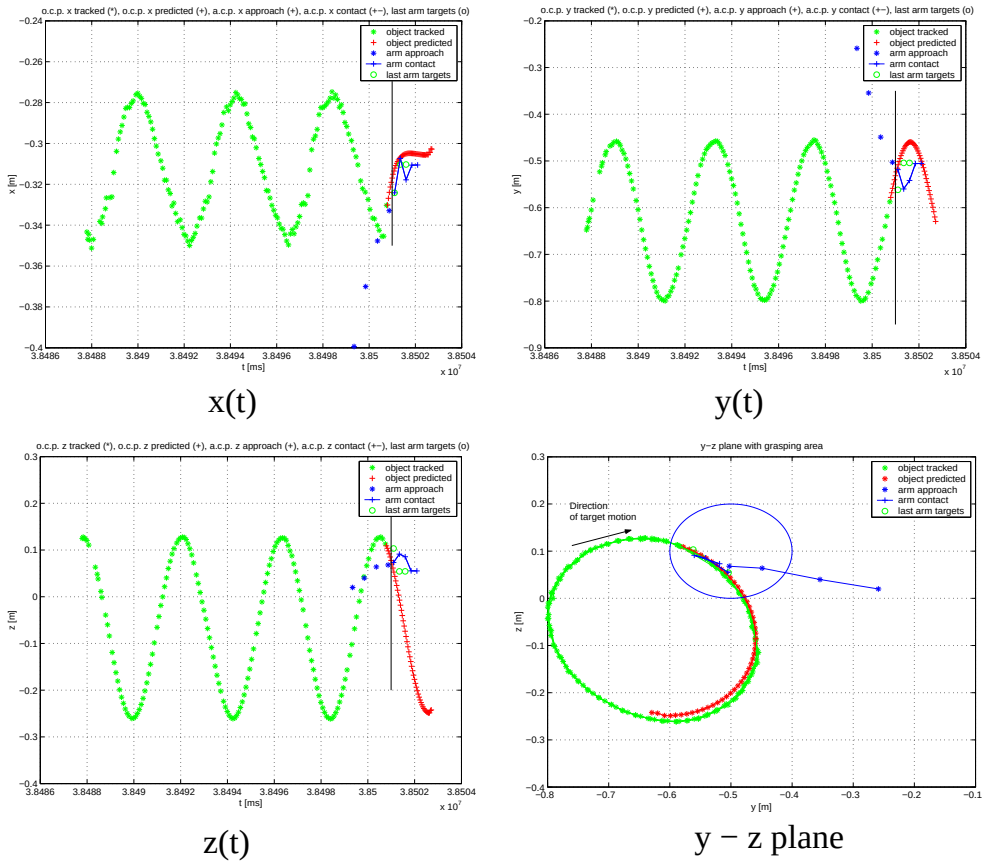


Figure 5.39: Plots of the trial shown in Figure 5.40. Time of a sub-movement $Ts = 1000$.

Figure 5.40: A target approaching on a circular path.

5.5 Discussion

Three main topics have been evaluated in this chapter: tracking of objects in motion, prediction of object motions and hand-target interaction for catching a moving target. For tracking different sensor modalities were tested for their usefulness for tracking. Each of them showed certain advantages and drawbacks, all according with the theory. For intelligent sensor preprocessing and fusion the biological concept of *reentry* and a modified version of the ICONDENSATION were tested. It could be clearly showed that the reentry mechanism strongly improved the tracking results for all test conditions. In detail it can be stated that (a) the mean tracking error was considerably lower for tracking with color reentry into the form path than without reentry. (b) the loss of the tracking object was considerably more seldom and (c) the recovery from a loss (re-initialization) was faster and always possible. Additionally, processing time for tracking with reentry was lower than without.

The prediction of target motion was tested in simulation and for real image sequences. In the simulation was shown that prediction performance of the different methods (ARM, NN) is strongly depended on the form of the input data. For real image sequence the proposed Average ARM prediction algorithm, despite the fact that it is simple and therefore fast, proved to be working accurately enough to be used for hand-target experiments. Test cases were real situations, i.e. objects moved by hand or a moving model train.

In the simulation of the hand-target interaction the motion of the end-effector behaved according to the theory described in th preceding chapter. It was shown that it is possible to interact with a moving target in a "human like" way. Two examples were shown, one for a approaching target and one for a escaping target. The resulting end-effector motions and velocity profiles were smooth. Additionally, in the experiments the timely course of the interaction points and the intermediate targets was demonstrated. Their relation to the predicted object motion and the timing of the different end-effector motion phases was shown. It could be seen that the approach is general enough to control an robotic end-effector for any kind of a predictable target motion.

These results allowed the implementation of the strategy on the robot. The corresponding real hand-eye experiments were fulfilled on the experimental robot MinERVA. Thereby the case of an approaching, escaping and a tangential moving target were shown. The results demonstrated that it is possible to *grasp an moving object with the proposed methods*, i.e. that the combination of tracking, prediction, and manipulator control worked for different target motions.

Chapter 6

Conclusion

Naturally when looking back through this work some questions arise: What has been learned from looking at the results of neuroscience? What would be a consequent continuation of the topic? What could have been improved? But also: where are the limitations of the work, of the approach taken and of the results? And finally looking back on the "work done": what are the personal experiences working on a robotic hand-eye system?

First, neuroscience was (and is) a very good source of inspiration to develop new ideas for robotics. On the one hand there are well known principles as e.g. principles underlying the grasping behavior of humans. On the other hand there are thoroughly recorded experimental data sets that can be well interpreted as e.g. in this case for catching. An thirdly, there are interesting models combined with experimental data about more "fuzzy" topics as e.g. the functioning of the brain, or i.e. the functioning of the visual cortex. Taking into account the knowledge of "standard" robotics and engineering one can get fast to new, adapted models of human (or in general biological) behavior, that can be implemented on a robot and (promise to) lead to better results as standard robotics methods. That this is valid has been shown in this thesis for the cases of controlling a robotic manipulator for a catching task, and to get to more efficient visual tracking. Last but not least the implementation of biological models is interesting for neuroscientists to get feedback about the validity and integrity of their models and inspiration for new experiments. Maybe the results presented here serve to support this feedback process.

A work like this is of course never finished. Obviously, there were things left out or left behind that would need more thorough evaluation. Especially, in this work many different topics have been addressed and discussed, but always keeping in mind that the "hand-eye *system* for catching" was the central point. Thereby the single parts were developed up to an exactness sufficient to fit into the system and provide the needed information to the system. Nevertheless, some topics that were interesting were explored more deeply as e.g. the visual sensor fusion and integration, and its applicability for visual tracking. This evaluation may thereby serve as a good starting point for anyone who is interested in the

transfer of biological concepts to improve robotic vision.

That this procedure was justified can be seen from the experiments. The experimental hand-eye system MinERVA can now actually reach to catch a moving object in a human like manner. The only limitation is that orientation is left constant during the catching experiments.

Looking back "on the work" performed it was very valuable to have a clear understanding of the complete system and the dependencies between the different modules. In the same way as it hopefully helped the reader to read this thesis, it helped me and the students working on that topic not to loose sight for the essential. Additionally, it leaves space for new developments, improvements of the existing methods or integration of new ideas.

Bibliography

[AC99] Benedetto Allotta und Carlo Colombo. On the Use of Linear Camera-Object Interaction Models in Visual Servoing. *IEEE Trans. on Robotics and Automation*, 15(2):350–357, April 1999. 71

[And86] R.L. Anderson. Real time intelligent visual control of a robot. *IEEE Workhop on Intelligent Control*, 1986. 75

[And89] R. L. Andersson. Dynamic sensing in a ping-pong playing robot. *IEEE Trans. on Robotics and Automation*, 5(6):723–739, 1989. 76

[ATYM93a] Peter K. Allen, Aleksandar Timcenko, Billibon Yoshimi, und Paul Michelman. Automated Tracking and Grasping of a Moving Object with a Robotic Hand-Eye System. *IEEE Trans. on Robotics and Automation*, 9(2):152–165, April 1993. 75

[ATYM93b] Peter K. Allen, Aleksandar Timcenko, Billibon Yoshimi, und Paul Michelman. Hand-Eye Coordination for Robotic Tracking and Grasping. In Koichi Hashimoto, Herausgeber, *Visual Servoing*, Seiten 33–69. World Scientific Publishing Company, 1993. 75

[BBY98] Antonio Bicchi, Joel Burdick, und Tsuneo Yoshikawa, Herausgeber. *Workshop on Grasping, Fixturing, and Manipulation: Towards a Common Language*. In association with ICRA'98, Mai 1998. 72

[BC95] Farabi Bensalah und François Chaumette. Compensation of Abrupt Motion Changes in Target Tracking by Visual Servoing. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'95)*, Seiten 181–187, 1995. 74

[Bea01] Jean-Charles Beauverger. Biologisch Motivierte Wegepunktbestimmung für eine Fangaufgabe mit einem robotischen Manipulator. Diplomarbeit, TU München, Lehrstuhl für Realzeit-Computersysteme, Oktober 2001. 191

[BH75] A. E. Bryson und Y. C. Ho. *Applied Optimal Control*. Hampshire Publishing Co., New York, 1975. 22

[BHMIG92] Emilio Bizzi, Neville Hogan, Ferdinando A. Mussa-Ivaldi, und Simon Giszter. Does the nervous system use equilibrium-point control to guide single and multiple joint movements? *Behavioral and Brain Sciences*, 15:603–613, 1992. 22

[BHS94] Bradley E. Bishop, Seth Hutchinson, und Mark Spong. On the Performance of State Estimation for Visual Servo Systems. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'94)*, Seiten 168–173, 1994. A-13

[BI98] A. Blake und M. Isard. *Active Contours*. Springer-Verlag, 1998. 47, 48, 49, 50, 51, 52, 64, 85, 96, 106, 170, A-6

[BIR95] A. Blake, M.A. Isard, und D. Reynard. Learning to track the visual motion of contours. *Journal of Artificial Intelligence*, 78:101–134, 1995. 48

[BJ96] M.J. Black und A.D. Jepson. Eigentracking: robust matching and tracking of articulated objects using a view-based representation. *Proc. 4th European Conf. on Computer Vision (ECCV'96)*, Seiten 329–342, 1996. 48

[BK00] A. Bicchi und V. Kumar. Robotic Grasping and Contact: A Review. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, Seiten 348–353, April 2000. 63

[BL96] Etienne Burdet und J. Luthiger. Adaptation of the Visuo-Motor Coordination. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'96)*, Seiten 2656–2661, April 1996. 75

[Bla00] Martin Blasczyk. Bildverarbeitung mit Pentium-III SIMD Befehlen. TU München, Lehrstuhl für Realzeit-Computersysteme, Studienarbeit, Dezember 2000. 150

[BLZ96] S. Blessing, S. Lanser, und C. Zierl. Vision-based Handling with a Mobile Robot. In M. Jamshidi, F. Pin, und P. Dauchez, Herausgeber, *International Symposium on Robotics and Manufacturing (ISRAM)*, Vol. 6, Seiten 49–59. ASME Press, 1996. 73

[BM98] E. Burdet und T. E. Milner. Quantization of human motions and learning of accurate movements. *Biological Cybernetics*, 78:307–318, 1998. 23

[BMMZ94] Reinoud J. Bootsma, Ronald G. Marteniuk, Christine L. MacKenzie, und Frank T. J. M. Zaal. The speed-accuracy trade-off in manual prehension: effects of movement amplitude, object size and object width on kinematic characteristics. *Experimental Brain Research*, 98:535–541, 1994. 19, 20

[BP92] Reinoud Bootsma und C. Peper. Predictive visual information sources for the regulation of action with special emphasis on catching and hitting. In

L. Proteau und D. Elliott, Herausgeber, *Vision and motor control*, Kapitel 12, Seiten 285–314. Elsevier Science Publishers B.V., 1992. 24

[BS98] Bradley E. Bishop und Mark W. Spong. Vision-Based Objective Selection for Robust Ballistic Manipulation. In *Robust Vision for Vision-Based Control of Motion, Workshop at the ICRA '98*, Mai 1998. 47, 76

[Bur96] Etienne Burdet. *Algorithms of Human Motor Control and their Implementation in Robotics*. Dissertation, ETH Zürich, Switzerland, 1996. 23

[Bur02] Gregor Burmberger. *PC-basierte Systemarchitekturen für zeitkritische technische Prozesse*. Dissertation, TU München, 2002. 150

[CC97] Armel Crétual und François Chaumette. Positioning a Camera Parallel to a Plane Using Dynamic Visual Servoing. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'97)*, Seiten 43–48, September 1997. 71

[CC98] Armel Crétual und François Chaumette. Image-based visual servoing by integration of dynamic measurements. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'98)*, Seiten 1994–2001, Mai 1998. 71

[CC99] T. Cham und R. Cipolla. Automated B-spline Curve Representation Incorporating MDL and Error-Minimizing Control Point Insertion Strategies. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(1):49–53, 1999. 91

[CH97] R. Cipolla und N. J. Hollinghurst. Visually guided grasping in unstructured enviroments. *Robotics and Autonomous Systems*, 19:337–346, 1997. 73

[CM99] Tomasz Celinski und Brenan McCarragher. Achieving Efficient Data Fusion Through Integration of Sensory Perception Control and Sensor Fusion. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'99)*, Seiten 1960–1965, Mai 1999. 61

[Col72] H. Collewijn. Latency and gain of the rabbit´s optokinetic reactions to small movements. *Brain Research*, 36:59–70, 1972. 25

[CRE91] François Chaumette, Patrick Rives, und Bernard Espiau. Positioning of a Robot with Respect to an Object, Tracking it and Estimating its Velocity by Visual Servoing. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'91)*, Seiten 2248–2253. IEEE, 1991. 71

[DE88] Edgar DeYoe und David Van Essen. Concurrent processing streams in monkey visual cortex. *Trends in Neuroscience*, 11:219–226, 1988. 14

[DLC00] Adam Dubrowski, Jerry Lam, und Heather Carnahan. Target velocity effects on manual interception kinematics. *Academic Press, Inc.*, 104:103–118, 2000. 24

[EG98] Ashraf Elnagar und Kamal Gupta. Motion Prediction of Moving Object Based on Autoregressive Model. *IEEE Trans. on Systems, Man and Cybernetics Part A*, 28(6):803–810, November 1998. 67

[EKaTBSS92] Andreas K. Engel, Peter König, Andreas K. Kreiter and Thomas B. Schillen, und Wolf Singer. Temporal coding in the visual cortex: new vistas on integration in the nervous system. *Trends in Neuroscience*, 15:218–226, 1992. 15

[ES97] Wolfgang Eckstein und Carsten Steger. Architecture for Computer Vision Application Development within the HORUS System. *Journal of Electronic Imaging*, 6(2):244–261, April 1997. 150

[FDD94] Christian Fagerer, Dirk Dickmanns, und Ernst D. Dickmanns. Visual Grasping with Long Delay Time of a Free Floating Object in Orbit. *Autonomous Robots*, 1:53–68, 1994. 76

[Fel66] A. G. Feldman. Functional tuning of nervous system with control of movement or maintenance of a steady posture. II. Controllable parameters of the muscles. *Biophysics*, 11:766–775, 1966. 21

[FH85] T. Flash und N. Hogan. The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model. *J. Neuroscience*, 5(7):1688–1703, 1985. 22, 118

[FH91] Tamar Flash und Ealan Henis. Arm trajectory modification during reaching towards visual targets. *J. Cognitive Neuroscience*, 3:220–230, 1991. 20, 23, 118

[FHH01] Udo Frese, Steffen Haidacher, und Gerd Hirzinger. Off-the-shelf Vision for a Robotic Ball Catcher. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'01)*, Seiten 1623–1629, 2001. 61, 77

[Fit54] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *J. Experimental Psychology*, 47:381–391, 1954. 20

[Fla89] Tamar Flash. Generation of Reaching Movements: Plausibility and Implications of the Equilibrium Trajectory Hypothesis. *Brain Behavioural Evolution*, 33:63–68, 1989. 22

[FOF93] J. Randall Flanagan, David J. Ostry, und Anatol G. Feldman. Control of Trajectory Modifications in Target-Directed Reaching. *J. Motor Behavior*, 25(3):140–152, 1993. 22

[FP91] B. Faverjon und J. Ponce. On Computing Two-Finger Force-Closure Grasps of Curved 2D Objects. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'91)*, Seiten 424–429, 1991. 63

[GCL93] C. Gosselin, J. Cote, und D. Laurendeau. Inverse kinematic functions for approach and catching operations. *IEEE Trans. on Systems, Man, and Cybernetics*, 23(3):783–791, Mai 1993. 75

[GdMA99] Jacques A. Gangloff, Michel de Mathelin, und Gabriel Abba. Visual servoing of a 6 DOF manipulator for unknown 3D profile following. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'99)*, Seiten 3236–3242, Mai 1999. 71

[Geo86] Apostolos P. Georgopoulos. On Reaching. *Annual Reviews Neuroscience*, 9:147–170, 1986. 19, 20

[GG95] Simon R. Goodman und Gerald G. Gottlieb. Analysis of kinematic invariances of multijoint reaching movement. *Biological Cybernetics*, 73:311–322, 1995. 119

[GGC92] Simon R. Gutman, Gerald G. Gottlieb, und D. Corcos. Exponential model of a reaching movement trajectory with nonlinear time. *Comments Theoretical Biology*, 2(5):357–383, 1992. 23

[Gla02] Sonja Glas. Automatic Initialisation of B-Splines and Fitting Them to 2D Object Silhouettes. Diplomarbeit, TU München, Lehrstuhl für Realzeit-Computersysteme, Februar 2002. 85, 91

[GM99] Antony Galton und Richard Meathrel. Qualitative Outline Theory. In *Proc. 16th Int. Joint Conf. on Artificial Intelligence*, Seiten 1061–66, Juli 1999. 86

[GMOS96] Enrico Grosso, Giorgio Metta, Andrea Oddera, und Giulio Sandini. Robust Visual Servoing in 3-D Reaching Tasks. *IEEE Trans. on Robotics and Automation*, 12(5):732–742, Oktober 1996. 72

[GS96] Melvyn A. Goodale und Philip Servos. Visual Control of Prehension. In Howard N. Zelaznik, Herausgeber, *Advances in Motor Learning and Control*, Kapitel 5. Human Kinetics, 1996. 21

[Haf98] W. Hafner. *Segmentierung von Video-Bildfolgen durch Adaptive Farbklassifikatoren*. Dissertation, Technische Universität München, Institut für Informatik IX, 1998. 54, 55

[Hag95] Gregory D. Hager. Calibration-Free Visual Control Using Projective Invariance. In *Proc. Int. Conf. Computer Vision*, Seiten 1009–1015, 1995. 72

[HAL] MVTec Software GmbH. *HALCON – The Software Solution for Machine Vision Applications*. http://www.mvtec.com/halcon/. 104, 150, A-14, A-15

[Hau99] Alexa Hauck. *Vision-Based Reach-To-Grasp Movements: From the Human Example to an Autonomous Robotic System*. Dissertation, TU München, 1999. 2, 19, 36, 71, 120, 121, 122, 123, 133

[HBDH93] Gerd Hirzinger, Bernhard Brunner, Johannes Dietrich, und Johann Heindl. Sensor-Based Space Robotics – ROTEX and Its Telerobotic Features. *IEEE Trans. on Robotics and Automation*, 9(5):649–663, Oktober 1993. 76

[HC93] Thomas J. Hebert und Seenwei Chen. Object classification using half-contour features. In *Pattern Recognition Letters*, Vol. 14. Elsevier Science Publishers B.V., North-Holland, Juni 1993. 55

[HD97] Greg Hager und Zachary Dodds. A Projective Framework for Constructing Accurate Hand-Eye Systems. In *Proc. Workshop on New Trends in Image-Based Robot Servoing, In assoc. with IROS'97*, Seiten 71–82, September 1997. 72

[HDE98] Radu Horaud, Fadi Dornaika, und Bernard Espiau. Visually Guided Object Grasping. *IEEE Trans. on Robotics and Automation*, 14(4):525–532, August 1998. 72

[HEG99] Y. Hu, R. Eagleson, und M. A. Goodale. Human Visual Servoing for Reaching and Grasping: The Role of 3-D Geometric Features. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'99)*, Seiten 3209–3216, Mai 1999. 21

[Hen91] Ealan Henis. *Strategies Underlying Arm Trajectory Modification During Reaching Toward Visual Targets*. Dissertation, Weizmann Institute of Science, Israel, 1991. 23, 119

[HFS97] U. D. Hanebeck, C. Fischer, und G. Schmidt. ROMAN: A Mobile Robotic Assistant for Indoor Service Applications. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'97)*, Seiten 518–525, September 1997. 73

[HHC96] Seth Hutchinson, Gregory D. Hager, und Peter I. Corke. A Tutorial on Visual Servo Control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, Oktober 1996. 45, 71, 119

[HHS97] W. Hong, T. Hornung, und J. J. E. Slotine. Robotic Catching of Free Flying Objects. Artificial Intelligence Laboratory Research Abstract, MIT, 1997. 64

[HK93] Koichi Hashimoto und Hidenori Kimura. LQ Optimal Control and Nonlinear Approaches to Visual Servoing. In Koichi Hashimoto, Herausgeber, *Visual Servoing*, Seiten 165–198. World Scientific Publishing Company, 1993. 74

[HK95] Koichi Hashimoto und Hidenori Kimura. Visual Servoing with Nonlinear Observer. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'95)*, Seiten 484–489, 1995. 74

[HN99] Koichi Hashimoto und Toshiro Noritsugu. Visual Servoing with Linearized Observer. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'99)*, Seiten 263–268, 1999. 74

[Hon95] W. Hong. Robotic Catching and Manipulation Using Active Vision. Diplomarbeit, Massachusetts Institute of Technology, USA, September 1995. 76

[Hou90] N. Houshangi. Control of a robotic manipulator to grasp a moving target using vision. *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'90)*, 1:604–609, 1990. 75

[HRSF99] Alexa Hauck, Johanna Rüttinger, Michael Sorg, und Georg Färber. Visual Determination of 3D Grasping Points on Unknown Objects with a Binocular Camera System. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'99)*, Oktober 1999. 64

[HS95] W. Hong und J. J. E. Slotine. Experiments in Hand-Eye Coordination Using Active Vision. In *Proc. 4th Int. Symp. Experimental Robotics (ISER'95)*, 1995. 76

[HT96] G. Hager und K. Toyama. Xvision: Combining image warping and geometric constraints for fast visual tracking. *Proc. 4th European Conf. on Computer Vision (ECCV'96)*, 2:507–517, 1996. 48

[HW65] David Hunter Hubel und Thorsten Niels Wiesel. Receptive fields and functional architecture in two non-striate visual areas (18 and19) of the cat. *J. Neurophysiology*, 28:229–289, 1965. 16

[HY92] Mark Hedley und Hong Yan. Segmentation of color images using spatial and color space information. In *Journal of Electronic Imaging*, Vol. 1, Department of Electrical Engineering, Sydney, New South Wales 2006, Australia, Oktober 1992. 56

[IB98] Michael Isard und Andrew Blake. ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework. In *Proc. 5th European Conf. on Computer Vision (ECCV'98)*, Seiten 893–908, 1998. 47, 62, 64, 106, 107

[Int] Intel. Intel Architecture MMX Technology, Programmer's Reference Manual. 103

[Jar88] R. A. Jarvis. Automatic Grip Site Detection for Robotics Manipulators. *Australian Computer Science Communications*, 10(1):346–356, 1988. 64

[JFN96] Martin Jägersand, Olac Fuentes, und Randal Nelson. Acquiring Visual-Motor Models for Precision Manipulation with Robot Hands. In *Proc. 4th European Conf. on Computer Vision (ECCV'96)*, Seiten 603–612, 1996. 72

[Jäg96] Martin Jägersand. Visual Servoing using Trust Region Methods and Estimation of the Full Coupled Visual-Motor Jacobian. In *Proc. IASTED Appl. Control and Robotics*, Seiten 105–108, 1996. 72

[Jäg97a] Martin Jägersand. Image Based Visual Simulation and Tele-Assisted Control. In *Proc. Workshop on New Trends in Image-Based Robot Servoing, In assoc. with IROS'97*, Seiten 33–44, September 1997. 72

[Jäg97b] Martin Jägersand. *On-line Estimation of Visual-Motor Models for Robot Control and Visual Simulation*. Dissertation, University of Rochester, USA, 1997. 72

[JN95] Martin Jägersand und Randal Nelson. Visual Task Specification, Planning and Control. In *Proc. IEEE Int. Symp. Computer Vision*, Seiten 521–526, 1995. 72

[Kan91] E.R. Kandel. Perception of Motion, Depth and Form. In E.R. Kandel, J.H. Schwartz, und T.M. Jessel, Herausgeber, *Principles of Neural Science*, Seiten 421–438. Elsevier Science Publishers B.V., 1991. 14, 15, 16, 18, 100

[Kaw96] Mitsuo Kawato. Trajectory Formation in Arm Movements: Minimization Principles and Procedures. In Howard N. Zelaznik, Herausgeber, *Advances in Motor Learning and Control*, Kapitel 9. Human Kinetics, 1996. 19, 118

[KC99] D. Kragić und H. I. Christensen. Integration of visual cues for active tracking of an end-effector. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'99)*, Oktober 1999. 54

[KFE96] Ishay Kamon, Tamar Flash, und Shimon Edelman. Learning to Grasp Using Visual Information. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'96)*, Seiten 2470–2476, 1996. 73

[KFN02] H. Kolb, E. Fernandez, und R. Nelson. WEBVISION: The Organization of the Retina and the Visual System. Technical report, Interdepartmental Program in Neuroscience, University of Utah, University of Utah, 2002. 10

[KG89] T. R. Kaminski und A. M. Gentile. A kinematic comparison of single and multijoint pointing movements. *Experimental Brain Research*, 78:547–556, 1989. 19

[KHL99] Won Kim, Sun-Gi Hon, und Ju-Jang Lee. An Active Contour Model using Image Flow for Tracking a Moving Object. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'99)*, Vol. 1, Seiten 216–221, Kyongju, Korea, 1999. 61, 64

[KM] Spencer Kimball und Peter Mattis. The GNU Image Manipulating Program. http://www.gimp.org/. 54

[KP94] D. J. Kriegman und J. Ponce. Representation for Recognizing Complex Curved 3D Objects. In *Proc. NFS–ARPA Workshop on Object Representation in Computer Vision*, Seiten 125–138. Springer-Verlag, 1994. 46

[Kra98] Vladimir Kravtchenko. Using MMX Technology in Digital Image Processing. Technischer Bericht TR-98-13, University of British Columbia, Departement of Computer Science, 1998. 53, 54, 55

[KSK90] G. J. Klinker, S. A. Shafer, und T. Kanade. A physical approach to color image understanding. In *International Journal of Computer Vision*, Vol. 4, 1990. 60

[KTNG98] F. Keçeci, M. Tonko, H.-H. Nagel, und V. Gengenbach. Improving visually servoed disassembly operations by automatic camera placement. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'98)*, Seiten 2947–2952, Mai 1998. 73

[KWT87] M. Kass, A. Witkin, und D. Terzopoulos. Snakes: Active contour models. In *Int. Conf. on Computer Vision*, Seiten 259–258, 1987. 49

[Lee76] D.N. Lee. A theory of visual control of braking based on the information about time-to-contact. *Perception*, 5:437–459, 1976. 25

[Leu00] Jan Leupold. Biologically motivated visual sensor integration for scene segmentation and object tracking. Diplomarbeit, TU München, Lehrstuhl für Realzeit-Computersysteme, Dezember 2000. 98, 150, 156

[LH87] MS Livingstone und DH Hubel. Psychophysical evidence for separate channels for the perception of form, color movement and depth. *J. Neuroscience*, 7:3416–3468, 1987. 15

[LL90] Young Won Lim und Sang Uk Lee. On the color image segmentation algorithm based on the thresholding and the fuzzy c-MEANS techniques. In *Pattern Recognition*, Vol. 23, 1990. 56

[Low85] David G. Lowe. *Perceptual organization and visual recognition*. The Kluwer International Series In Engineering and Computer Science. Robotics and vision. Kluwer Academic Publishers, Dordrecht, 1985. 47

[LPG97] Daeyol Lee, Nicholas Lindman Port, und Apostolos Georgopoulos. Manual interception of moving targets: On-line control of overlapping submovements. *Experimental Brain Research*, 116:421–433, 1997. 24, 25, 28, 29

[LY94] Jianqing Liu und Yee-Hong Yang. Multiresolution Color Image Segmentation. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 16, 1994. 56

[LZ95] S. Lanser und Ch. Zierl. Robuste Kalibrierung von CCD-Sensoren für autonome, mobile Systeme. In R. Dillmann, U. Rembold, und T. Lüth, Herausgeber, *Autonome Mobile Systeme*, Informatik aktuell, Seiten 172–181. Springer-Verlag, 1995. A-15

[LZB95] S. Lanser, Ch. Zierl, und R. Beutlhauser. Multibildkalibrierung einer CCD-Kamera. In G. Sagerer, S. Posch, und F. Kummert, Herausgeber, *Mustererkennung*, Informatik aktuell, Seiten 481–491. Deutsche Arbeitsgemeinschaft für Mustererkennung, Springer-Verlag, 1995. A-14, A-15

[LZP89] Z. Lin, V. Zeman, und R. Patel. On-line robot trajectory planning for catching a moving object. *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'89)*, 1:1726–1731, 1989. 75

[Mai99] Christian Maier. Biologically Motivated Kinematic Control of a 7-DOF Redundant Manipulator. Diplomarbeit, TU München, Lehrstuhl für Realzeit-Computersysteme, Dezember 1999. 37, 40, 42, 43, 124

[Mai00] Thomas Maier. Ansteuerung eines redundanten Manipulators in Echtzeit mit Hilfe eines Digitalen Signalprozessors. Diplomarbeit, TU München, Lehrstuhl für Realzeit-Computersysteme, September 2000. 133, 150

[Mar82] D. Marr. *Vision*. Freeman, New York, 1982. 3

[MB93] Joseph McIntyre und Emilio Bizzi. Servo Hypotheses for the Biological Control of Movement. *J. Motor Behavior*, 25(3):193–202, 1993. 22

[MBGL94] Denis Mottet, Reinould J. Bootsma, Yves Guiard, und Michel Laurent. Fitts' law in two-dimensional tasks. *Experimental Brain Research*, 100:144–148, 1994. 20

[MC99] Andrea Mason und Heather Carnahan. Target viewing time and velocity effects on prehension. *Experimental Brain Research*, 127:83–94, 1999. 24

[MCB99] Ezio Malis, François Chaumette, und Sylvie Boudet. 2-1/2-D Visual Servoing. *IEEE Trans. on Robotics and Automation*, 15(2):238–250, April 1999. 71

[MCR96] Éric Marchand, François Chaumette, und Alessandro Rizzi. Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'96)*, Seiten 1083–1090, Oktober 1996. 71

[MDGD97] P. Martinet, N. Daucher, J. Gallice, und M. Dhome. Robot Control using Monocular Pose Estimation. In *Proc. Workshop on New Trends in Image-Based Robot Servoing, In assoc. with IROS'97*, Seiten 1–12, September 1997. 71

[MH98] Éric Marchand und Greg Hager. Dynamic Sensor Planning in Visual Servoing. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'98)*, Seiten 1988–1993, Mai 1998. 71

[Mil92] T. E. Milner. A model for the generation of submovements requiring endpoint precision. *J. Neuroscience*, 49(2):487–496, 1992. 20, 23

[MK91] C. Mason und E.R. Kandel. Central Visual Pathways. In E.R. Kandel, J.H. Schwartz, und T.M. Jessel, Herausgeber, *Principles of Neural Science*, Seiten 421–438. Elsevier Science Publishers B.V., 1991. 15, 102

[MN95] H. Murase und S. K. Nayar. Visual Learning and Recognition of 3D Objects from Appearance. *Int. J. Computer Vision*, 14(1):5–24, Januar 1995. 46

[Mor81] P. Morasso. Spatial Control of Arm Movements. *Experimental Brain Research*, 42:223–227, 1981. 19

[MRSdP01] A. Morales, G. Recatalá, P.J. Sanz, und A.P. del Pobil. Heuristic Vision-Based Computation of Planar Antipodal Grasps on Unknown Objects. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'01)*, Seiten 583–588, Seoul, Korea, Mai 2001. 63, 64, 110, 111

[MSW82] David E. Meyer, J. E. Keith Smith, und Charles E. Wright. Models for the Speed and Accuracy of Aimed Movements. *Psychological Review*, 89(5):449–482, September 1982. 20, 23

[MZ92] Joseph L. Mundy und Andrew Zisserman. Projective Geometry for Machine Vision. In Joseph L. Mundy und Andrew Zisserman, Herausgeber, *Geometric Invariance in Computer Vision*, Kapitel 23, Seiten 463–519. MIT Press, 1992. 45, A-12

[N+98] Hiroaki Nakai et al. A Volleyball Playing Robot. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'98)*, Seiten 1083–1089, Mai 1998. 47, 64, 76

[Nel83] W. L. Nelson. Physical Principles for Economies of Skilled Movements. *Biological Cybernetics*, 46:135–147, 1983. 22, 23

[Ngu88] V.-D. Nguyen. Constructing Force-Closure Grasps. *The Intl. J. of Robotics Research*, 7(3), 1988. 63, 110

[NHNT00] K. Nagahama, K. Hashimoto, T. Noritsugu, und M. Takaiawa. Visual Servoing based on Object Motion Estimation. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'00)*, Vol. 1, Seiten 245–250, 2000. 76

[NNII99] Akio Namiki, Yoshihiro Nakabo, Isaku Ishii, und Masatoshi Ishikawa. High Speed Grasping Using Visual and Force Feedback. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'99)*, Seiten 3195–3200, Mai 1999. 73

[NPK93] Bradley J. Nelson, Nikolaos P. Papanikolopoulos, und Pradeep K. Khosla. Visual Servoing for Robotic Assembly. In Koichi Hashimoto, Herausgeber, *Visual Servoing*, Seiten 139–164. World Scientific Publishing Company, 1993. 71

[NS99] Heiko Neumann und Wolfgang Sepp. Recurrent V1-V2 interaction in early visual boundary processing. *Biological Cybernetics*, 81:425–444, 1999. 16

[OA99] Paul Y. Oh und Peter K. Allen. Performance of a Partitioned Visual Feedback Controller. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'98)*, Seiten 275–280, Mai 1999. 74

[OPB+98] D. E. Okhotsimsky, A. K. Platonov, I. R. Belousov, A. A. Bogulavsky, S. N. Emelianov, V. V. Sazonov, und S. M. Sokolov. Real-Time Hand-Eye System: Interaction with Moving Objects. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'98)*, Seiten 1683–1688, Mai 1998. 47, 76

[Osw00] Hans Oswald. Probabilistic Tracking of Moving Objects in Image Sequences using Shape Information. Diplomarbeit, TU München, Lehrstuhl für Realzeit-Computersysteme, Mai 2000. 52, 97

[Pai96] J. Paillard. Fast and Slow Feedback Loops for the Visual Correction of Spatial Errors in a Pointing Task: A Reappraisal. *Can. Journal of Physiological Pharmacology*, 74:401–417, 1996. 21

[PEKJ79] C. Prablanc, J. F. Echallier, E. Komilis, und M. Jeannerod. Optimal Response of Eye and Hand Motor Systems in Pointing at a Visual Target. *Biological Cybernetics*, 35:113–124, 1979. 20

[Pla95] Réjean Plamondon. A kinematic theory of rapid human movements. II. Movement time and control. *Biological Cybernetics*, 72:309–320, 1995. 20

[PLDG97] Nicholas Lindman Port, Daeyeol Lee, Paul Dassonville, und Apostolos Georgopoulos. Manual interception of moving targets: Performance and movement initiation. *Experimental Brain Research*, 116:406–420, 1997. 24, 25, 27

[PML98] Jenelle Armstrong Piepmeier, Gary V. McMurray, und Harvey Lipkin. Tracking a Moving Target with Model Independent Visual Servoing: A Predictive Estimation Approach. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'98)*, Seiten 2652–2657, Mai 1998. 74

[PML99] Jenelle Armstrong Piepmeier, Gary V. McMurray, und Harvey Lipkin. A Dynamic Quasi-Newton Method for Uncalibrated Visual Servoing. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'99)*, Seiten 1595–1600, Mai 1999. 74

[PMMJ91] Y. Paulignan, C. MacKenzie, R. Marteniuk, und M. Jeannerod. Selective perturbation of visual input during prehension movements. *Experimental Brain Research*, 83:502–512, 1991. 19

[PNK95] Nikolaos P. Papanikolopoulos, Bradley J. Nelson, und Pradeep K. Khosla. Six Degree-of-Freedom Hand/Eye Visual Tracking with Uncertain Parameters. *IEEE Trans. on Robotics and Automation*, 11(5):725–732, Oktober 1995. 74

[Pop95] A. R. Pope. *Learning to Recognize Objects in Images: Acquiring and Using Probabilistic Models of Appearance*. Dissertation, University of British Columbia, Canada, 1995. 46

[PP93] Nikhil R. Pal und Sankar K. Pal. A Review on image segmentation techniques. Technischer Bericht, Machine Intelligence Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700035 India, 1993. 55

[PP96] I. Pavlidis und N. P. Papanikolopoulos. Automatic Selection of Control Points for Deformable-Model-Based Target Tracking. *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'96)*, Seiten 2915–2920, 1996. 86

[PPG96] Nicholas Lindman Port, Guiseppe Pellizer, und Apostolos Georgopoulos. Intercepting real and path-guided apparent motion targets. *Experimental Brain Research*, 110:298–307, 1996. 24

[PPGJ86] D. Pélisson, C. Prablanc, M. A. Goodale, und M. Jeannerod. Visual control of reaching movements without vision of the limb. *Experimental Brain Research*, 62:303–311, 1986. 21

[PS95] Nikolaos P. Papanikolopoulos und Christopher E. Smith. Computer Vision Issues During Eye-In-Hand Robotic Tasks. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'95)*, Seiten 2989–2994, 1995. 74

[PSMP00] D. Perrin, C.E. Smith, O. Masoud, und N.P. Papanikolopoulos. Unknown object grasping using statistical pressure models. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'00)*, Seiten 1054–1059, San Francisco, CA, April 2000. 63, 64

[Qt] Troll Tech. *The Qt GUI Toolkit*. http://troll.no/products/qt.html. 150

[RDR94] E. Rivlin, S. J. Dickinson, und A. Rosenfeld. Recognition by Functional Parts. In *Proc. Computer Vision and Pattern Recognition (CVPR'94)*, Seiten 267–272. IEEE Computer Society Press, 1994. 46

[RK93] Alfred A. Rizzi und Daniel E. Koditschek. A Dynamic Sensor for Robot Juggling. In Koichi Hashimoto, Herausgeber, *Visual Servoing*, Seiten 229–256. World Scientific Publishing Company, 1993. 76

[RK96] Alfred A. Rizzi und Daniel E. Koditschek. An Active Visual Estimator for Dexterous Manipulation. *IEEE Trans. on Robotics and Automation*, 12(5):697–713, 1996. 76

[Ros94] D. A. Rosenbaum. *Human Motor Control*, Kapitel 6. Harcourt, Brace, Jovanovich, 1994. 19, 21

[RP95] Charles A. Richards und Nikolaos P. Papanikolopoulos. The Automatic Detection and Visual Tracking of Moving Objects by Eye-in-Hand Robotic Systems. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'95)*, Seiten 228–233, 1995. 74

[RSL+02] G. Recatalá, M. Sorg, J. Leupold, P.J. Sanz, und A.P. del Pobil. Visual grasp determination and tracking in 2D dynamic scenarios. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'02)*, 2002. to appear. 110, 114

[RTHN97] A. Ruf, M. Tonko, R. Horaud, und H.-H. Nagel. Visual Tracking of an End-Effector by Adaptive Kinematic Prediction. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'97)*, Seiten 893–898, September 1997. 73

[Rus95] John C. Russ. *The Image Processing Handbook*. CRC Press, Inc., 1995. 54, 57

[SBP97] Christopher E. Smith, Scott A. Brandt, und Nikolaos P. Papanikolopoulos. Eye-In-Hand Robotic Tasks in Uncalibrated Environments. *IEEE Trans. on Robotics and Automation*, 13(6):903–914, Dezember 1997. 74

[Sch84] Elmar Schrüfer. *Zuverlässigkeit von Mess- und Automatisierungseinrichtungen*. Carl Hanser Verlag München Wien, 1984. 69

[Sch93] Raimondo Schettini. A segmentation algorithm for color images. In *Pattern Recognition Letters*, Vol. 14, 1993. 56

[SdPIR98] P. J. Sanz, A. P. del Pobil, J. M. Inesta, und G. Recatalá. Vision-Guided Grasping of Unknown Objects for Service Robots. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'98)*, Seiten 3018–3025, Mai 1998. 63, 64, 73, 86, 111

[Sel00] Georg Selzle. Rekonstruktion und Prädiktion von 3D-Objektbewegungen. Diplomarbeit, TU München, Lehrstuhl für Realzeit-Computersysteme, August 2000. 70, 117, 184

[SJ99] Michael Small und Kevin Judd. Variable prediction steps and longer term prediction. Technischer Bericht, Centre for Applied Dynamics und Optimization, Department of Mathematics and Statistics, University of Western Australia, 1999. 65

[Sol97] Stephen J. Solari. *Digital Video and Audio Compression*. McGraw-Hill Book Company, 1997. 53

[SS96] L Sciavicco und B. Siciliano. *Modeling and Control of Robot Manipulators*. McG, University of Naples, Naples, 1996. 41

[Stö01] Norbert O. Stöffler. *Realzeitfähige Bestimmung und Interpretation des optischen Flusses zur Navigation mit einem mobilen Roboter*. Dissertation, TU München, 2001. 60, 61, 99, 104, 164

[Sta91] S.A. Stansfield. Robotic Grasping of Unknown Objects: A Knowledge-Based Approach. *The Intl. J. of Robotics Research*, 10(4):314–326, August 1991. 64

[SZB95] Larry S. Shapiro, Andrew Zisserman, und Michael Brady. 3D Motion Recovery via Affine Epipolar Geometry. *Int. J. Computer Vision*, 16(2):147–182, Oktober 1995. 45, A-12

[TBC94] Michael Taylor, Andrew Blake, und Adrian Cox. Visually guided grasping in 3D. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'94)*, Seiten 761–766, 1994. 73

[Tsa87] Roger Y. Tsai. A Versatile Camera Calibration Technique for High Accuracy 3D Machine Vision Metrology Using Off-The-Shelf TV Cameras and Lenses. *IEEE Trans. on Robotics and Automation*, 3(4):323–344, August 1987. A-14

[TSE92] G. Tononi, O. Sporns, und G.M. Edelman. Reentry and the Problem of Integrating Multiple Cortical Areas: Simulation of Dynamic Integration in the Visual System. *Cerebral Cortex*, 2:310–335, Juli 1992. 15, 16

[TSHN97] M. Tonko, K. Schäfer, F. Heimes, und H.-H. Nagel. Towards Visually Servoed Manipulation of Car Engine Parts. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'97)*, Seiten 3166–3171, April 1997. 73

[TSSN97] M. Tonko, J. Schurmann, K. Schäfer, und H.-H. Nagel. Visually Servoed Gripping of a Used Car Battery. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'97)*, Seiten 49–54, September 1997. 73

[UKS89] Y. Uno, M. Kawato, und R. Suzuki. Formation and Control of Optimal Trajectory in Human Multijoint Arm Movement: Minimum Torque Change Model. *Biological Cybernetics*, 61:89–101, 1989. 22

[Val96] Pedro J. Sanz Valero. *Rozonamiento Geométrico Basado en Visión para la Determinación y Ejecución del Agarre en Robots Manipuladores.* Dissertation, Department of Computer Science at the Jaume-I University of Castellón, 1996. 86

[VAZ00] Markus Vincze, Minu Ayromlou, und Michael Zillich. Fast Tracking of Ellipses using Edge-Projected Integration of Cues. In *Proc. 15th Int. Conf. on Pattern Recognition*, Vol. 4, Seiten 72–75, Barcelona, E, September 2000. IEEE Computer Society Press. 48, 97

[vdKSS97] John van der Kamp, Geert Savelsbergh, und Jeroen Smeets. Multiple information sources in interceptive timing. *J. of Human Movement Science*, 16:787–821, 1997. 24

[vH79] C. von Hofsten. Development of visually guided reaching: the approach phase. *J. of Human Movement Studies*, 5:160–178, 1979. 24

[vH80] C. von Hofsten. Predictive reaching for moving objects by human infants. *J. of Experimental Child Psychology*, 30:369–382, 1980. 24

[vH82] C. von Hofsten. Eye-hand coordination in newborns. *Developmental Psychology*, 18:450–461, 1982. 24

[VMPG94] J. L. Vercher, G. Magenes, C. Prablanc, und G. M. Gauthier. Eye-head-hand coordination in pointing at visual targets: spatial and temporal analysis. *Experimental Brain Research*, 99:507–523, 1994. 21

[WG94] Andreas S. Weigend und Neil A. Gershenfeld. *Time Series Prediction : Forcasting the Future and Understanding the Past.* Perseus Books Publishing,L.L.C., 1994. 66, 70

[WHB96] William J. Wilson, Carol C. Williams Hulls, und Graham S. Bell. Relative End-Effector Control Using Cartesian Position Based Visual Servoing. *IEEE Trans. on Robotics and Automation*, 12(5):684–696, Oktober 1996. 74

[Wil93] William J. Wilson. Visual Servo Control of Robots Using Kalman Filter Estimates of Robot Pose Relative to Work-Pieces. In Koichi Hashimoto, Herausgeber, *Visual Servoing*, Seiten 71–104. World Scientific Publishing Company, 1993. 74

[Woo99] R. S. Woodworth. The accuracy of voluntary movement. *Psychological Review*, 3:1–114, 1899. 20

[YA95] Billibon H. Yoshimi und Peter K. Allen. Alignment Using an Uncalibrated Camera System. *IEEE Trans. on Robotics and Automation*, 11(4):516–521, August 1995. 71

[YLH98] Dejin Yu, Weiping Lu, und Robert G. Harrison. Phase-space prediction of chaotic time series. *Proceedings in Dynamics and Stability of Systems, Vol. 13, No. 3, 1998*, 1998. 70

[ZB94] M. Zhang und M. Buehler. Sensor-Based Online Trajectory Generation for Smoothly Grasping Moving Objects. *Int. Symp. Intelligent Control*, 1:141–146, 1994. 75

[ZL97] Didier Zugaj und Vincent Lattuati. A new approach of color images segmentation based on fusing region and edge segmentations outputs. In *Pattern Recognition*, Vol. 31, Februar 1997. 56

[ZM94] M. Zerroug und G. Medioni. The Challenge of Generic Object Recognition. In *Proc. NFS–ARPA Workshop on Object Representation in Computer Vision*, Seiten 217–232. Springer-Verlag, 1994. 46

Glossary of Symbols

a, b, c ...	scalar variables
$\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ ...	vector variables
$\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ ...	matrix variables
$\mathbf{I}$	identity matrix
$\mathbf{0}$	vector of zeroes
$\mathcal{F}_f$	Base (Frame) f with $\mathbf{B} = \left[\begin{array}{ccc} {}_f\mathbf{b}_x & {}_f\mathbf{b}_y & {}_f\mathbf{b}_z \end{array} \right]$
${}^f_g\mathbf{b}_i$	base vector i of $\mathcal{F}_g$ in $\mathcal{F}_f$
${}^f\mathbf{a}_i$	vector in $\mathcal{F}_f$ and index i
${}^f\mathbf{A}_i$	matrix in $\mathcal{F}_f$ and index i
${}^f_g\mathbf{T}_i$	transformation matrix of $\mathcal{F}_g$ to $\mathcal{F}_f$ with index i
${}^f\mathbf{R}_i$	rotation matrix in $\mathcal{F}_f$ with index i
${}^f\mathbf{p}_i$	translation vector in $\mathcal{F}_f$ with index i
${}^f\mathbf{x}_i$	Cartesian position vector in $\mathcal{F}_f$ and index i
${}^f\boldsymbol{\omega}_i$	angular velocity vector in $\mathcal{F}_f$ and index i
$\mathbf{q}$	joint angle vector
m	number of joint angles
$\mathcal{F}_w$	world coordinate system
$\mathcal{F}_e$	end-effector coordinate system
$\mathbf{J}^\sharp$	pseudo-inverse of matrix $\mathbf{J}$
$r(s)$	parameterized, two dimensional b-spline curve
$\mathbf{W}$	shape matrix
$\mathbf{X}$	shape vector
XOP	x position of optimal catch point
XIP	x position of interaction point
$XVIP$	x velocity of target object in the interaction point
T_s	time of a sub-movement
T_p	time for the target object to travel from the current position to target position
TtC	time to contact
ToC	time of contact with the target
$xeepos$	current x position of the end-effector

$xeedes$	desired x position of the end-effector
t_i	time of beginning of a sub-movement
t_{fi}	time of end of a sub-movement
f	(Gaussian) focal length of a camera, camera constant
S_x, S_y	scaling factors determining pixel size
C_x, C_y	pixel coordinates of the principal point
κ	radial distortion coefficient
B	baseline of the stereo system
β_L, β_R	vergence angles of left and right camera

Appendix A

B-Splines

Splines are nowadays a well exploited and commonly used mathematical method to describe smooth curves. The following paragraphs constitute an introduction to the basic principles of this method.

Basic mathematical background Originally a spline was a draughtsman's aid. It was a thin elastic wood or metal strip that was used to draw a curve through certain fixed points. Mathematically speaking, a spline is in general a piecewise polynomial function, where the pieces are smoothly connected.

An example of a spline is given Figure A.1. The connection points are called either **knots** $\mathbf{k}_i$, **hinges** or **breakpoints** and the pieces of the spline curve are known as **spans**. The individual spans are connected smoothly in the knots. The **order d** of a spline depends on the degree of the polynomials of its polynomial pieces. The order d is one degree higher than the degree of its polynomial (basis) functions.

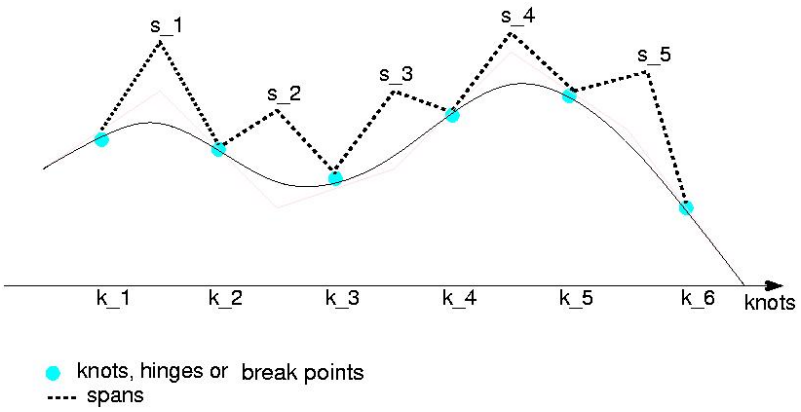


Figure A.1: Spline curve with knots and spans

The **B** before the spline stands for Basis-spline. A B-spline is is a convenient computational way of representing a spline function. The spline **r**(s) is defined as a weighted sum of basis functions (polynomial functions):

$$\mathbf{r}(s) = \sum_{i=0}^{N_B-1} w_i * B_{i,d}(s) \tag{A.1}$$

where:

N_B:	the number of basis functions
w_i:	weight i
B(s) :	basis function which is a polynomial function
$B_{i,d}(s)$:	translation of the basis function B(s) of order d to $B(s+i)$

The basis function $B_{i,d}(s)$ can be determined in different ways; a very comprehensive possibility is the convolution of a rectangular impulse *b(s)*:

$$b(s) = \begin{cases} 1 & \text{if } 0 \leq s < 1, \\ 0 & \text{otherwise} \end{cases} \tag{A.2}$$

The following Figure A.2 shows the rectangular impulse b(s) and its convolutions as well as the notation $B_{i,d}$ used below:

The basis functions can also be calculated with the recursive formula (A.3):

$$B_{i,d}\,(s) = \frac{(s-k_i) * B_{i,d-1}(s)}{k_{i+d-1} - k_i} + \frac{(k_i - s) * B_{(i+1),(d-1)}(s)}{k_{i+d} - k_{i+1}} \tag{A.3}$$

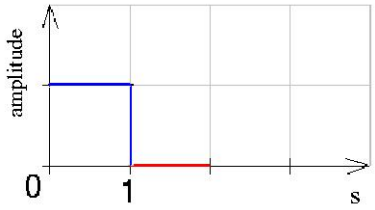
where:

d :	order of the spline
k_i:	knot i
$B_{i,d}\,(s)$:	basis function of the order d and translated by i

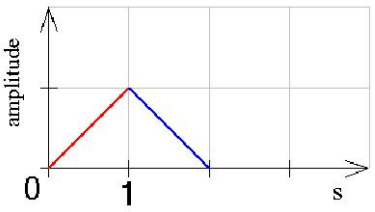
$B_{0,1}\,(s)$ is equivalent to the impulse b(s) above:

$$B_{0,1}(s) = \begin{cases} 1 & \text{if } k_0 \leq s < k_1, \\ 0 & \text{otherwise} \end{cases} \tag{A.4}$$

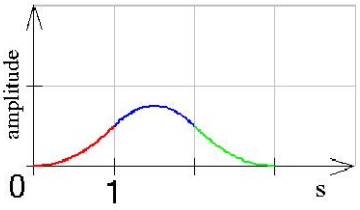
An important characteristic of the basis functions is the **local support property** of the B-spline, which indicates that the basis function vanishes outside the interval $[k_i, k_{d-i}]$. The result is that changes of a weight index w_i only affect the B-spline locally in the interval $k_i \leq s < k_{d-i}$.



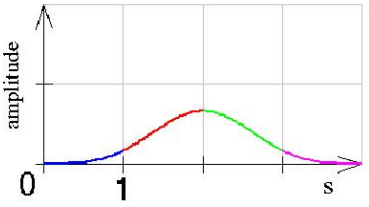
(a) basis function $B_{0,1}$ of order d = 1 equal to b(s)



(b) basis function $B_{0,2}$ of order d = 2 equal to b(s) convoluted with b(s)



(c) basis function $B_{0,3}$ of order d = 3 equal to $B_{0,2}$ convoluted with b(s)



(d) basis function $B_{0,4}$ of order d = 4 equal to $B_{0,3}$ convoluted with b(s)

Figure A.2: Basis functions of a B-spline of orders d = [1,...,4]

For example the basis function $B_{0,3}$ has the formula:

$$B_{0,3}(s) = \begin{cases} \frac{s^2}{2} & \text{if } k_0 \leq s < k_1, \\ \frac{3}{4} - (s - \frac{3}{2})^2 & \text{if } k_1 \leq s < k_2, \\ \frac{(s-3)^2}{2} & \text{if } k_2 \leq s < k_3, \\ 0 & \text{otherwise} \end{cases} \tag{A.5}$$

where k_i is a knot at the position s = i.

The order d of $B_{0,3}$ is 3 and the degree of its polynomials is two, one less than the order of the spline. In the knots the basis functions are smoothly connected in the same way as a spline is always smoothly connected in the knots.

A constraint for each point is that the sum of all basis functions B(s) equals one:

$$\sum_i B_i(s) = 1 \quad \text{for all s} \tag{A.6}$$

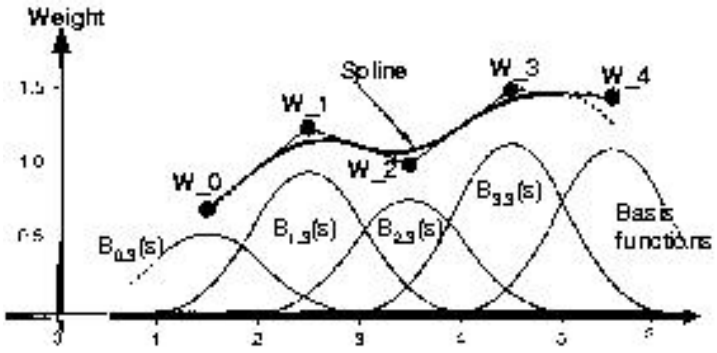


Figure A.3: A B-spline is the sum of weighted basis functions

A B-spline of order d is now calculated as the sum of the weighted basis functions of order d according to equation (A.1).

Figure A.3 shows the basis functions $B_{i,3}(s)$ of order d = 3 and the spline which is the sum of the basis functions weighted by w_i.

2-Dimensional B-Splines Calculation of the B-spline for e.g. object contour outlines, requires two-dimensional B-splines and 2-dimensional weights, also called control points c_i. In analogy to equation Equation A.1 a 2-dimensional B-spline is defined as:

$$\mathbf{r}(s) = \sum_{i=0}^{N_B-1} B_{i,d}(s) * c_i \tag{A.7}$$

where:

$$\mathbf{r}(s) = \begin{pmatrix} x(s) \\ y(s) \end{pmatrix}$$

$$c_i = \begin{pmatrix} w_i(x) \\ w_i(y) \end{pmatrix}$$

$B(s)$ basis function

Figure A.4 shows an example of a 2-dimensional closed B-spline with its pieces, the spans, connected by knots k. The control polygon is determined by the control points c. As can be seen in Figure A.3 the B-spline is an approximation to the control polygon which is formed by the weights. An example of the 2D case is given in Figure A.4.

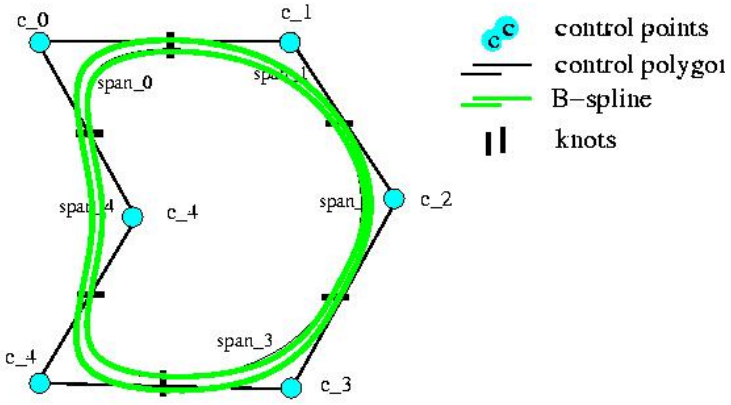


Figure A.4: 2-dimensional B-spline

A more convenient mathematical expression for this B-spline can be obtained using matrices. The vector sum of Equation A.7) can be expressed by matrices as follows:

$$\mathbf{r}(s) = \underbrace{\begin{pmatrix} \mathbf{B(s)^T} & \mathbf{0} \\ \mathbf{0} & \mathbf{B(s)^T} \end{pmatrix}}_{\mathbf{U}(s)} * \mathbf{Q} =$$

$$= \begin{pmatrix} \begin{pmatrix} B_{0,d} & B_{1,d} & ... & B_{N_B-1,d} \end{pmatrix} & \mathbf{0} \\ \mathbf{0} & \begin{pmatrix} B_{0,d} & B_{1,d} & ... & B_{N_B-1,d} \end{pmatrix} \end{pmatrix} * \begin{pmatrix} c_0^x \\ c_1^x \\ ... \\ c_{N_B-1}^x \\ c_0^y \\ ... \\ c_{N_B-1}^y \end{pmatrix} \tag{A.8}$$

Multiplicity The B-splines discussed up to now have always been curves smoothly connected in the knots k_i. Nevertheless it is often necessary to introduce a vertex or hinge at a certain point along such a parametric curve, to fit around sharp corners of an object outline. A way to do so is to introduce multiple knots[1]. At a regular breakpoint, the degree of smoothness is at its maximal value (continuity of all derivatives up to the (d-2)th). If a double knot is introduced at a breakpoint the degree of smoothness is reduced by one. To explain this the following shall serve:

[1]To use multiple control points would also be possible, but is not advantageous due to linearity constraints and bad behavior in the vicinity of the hinge (see [BI98] for details)

The derivative of the B-spline is continuous, as long as the order of the derivative is smaller than the degree of the polynomial pieces. For a B-spline of order d = 3 the first derivative is continuous, forming a hinge, and the second derivative is discontinuous, forming a break. Therefore, the **multiplicity** m_i of a knot can vary from 1 (smooth curve) to d-1 (break in the curve). This is shown in Figure A.5. The same result can be imagined if two knots move closer together until they coincide. The curvature of the B-spline becomes more pronounced until it results in a hinge, then it is a multiple knot.

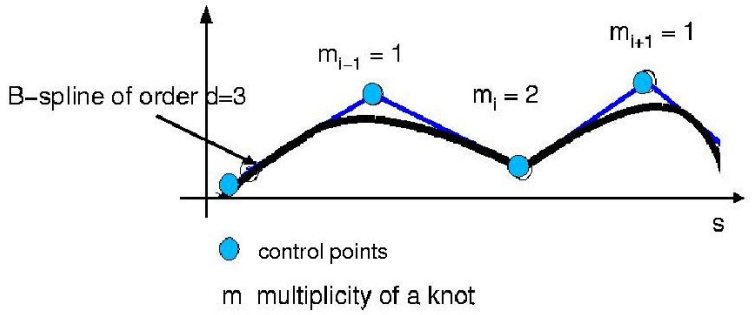


Figure A.5: B-spline with visible hinge at knot 2

If, however, the multiplicity **m** of the knots is not always 1 the situation is slightly different. As was mentioned before, a higher multiplicity m_i equals a coincidence of m_i knots, this is an increase in the number of basis functions B(s) used for combining the B-spline. Now the number of basis functions N_B for a closed B-spline is:

$$N_B = L + \sum_{i=0}^{L} (m_i - 1) \tag{A.9}$$

where: L is the number of spans

Appendix B

Model of the Head and the Camera

Model of the Head

The head frames in Fig. B.1 also follow the Denavit-Hartenberg conventions. θ_1 corresponds to the pan angle, θ_2 to the tilt angle. Table B.1 lists the corresponding link parameters. The resulting forward model

$$ {}^{0}_{2}\mathbf{T} = {}^{0}_{1}\mathbf{T} \cdot {}^{1}_{2}\mathbf{T} = \begin{bmatrix} \cos\theta_1 \cdot \cos\theta_2 & -\sin\theta_1 & \cos\theta_1 \cdot \sin\theta_2 & 0 \\ \sin\theta_1 \cdot \cos\theta_2 & \cos\theta_1 & -\cos\theta_1 \cdot \sin\theta_2 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.1} $$

is very simple and can be inverted directly.

In the case of the head, the equivalent of the end-effector are the cameras. Under the assumption that the cameras are mounted allowing only a 3D translation and a (fixed) vergence angle $\psi_{\{L,R\}}$, Equation B.3 gives the corresponding transforms, with ${}_{cL}\mathbf{Rot}$ ex-

Link	d	ϑ	a	α	transform
1	0	θ_1	0	$-90°$	${}^{0}_{1}\mathbf{T} = \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
2	0	θ_2	0	$90°$	${}^{1}_{2}\mathbf{T} = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 & 0 \\ \sin\theta_2 & 0 & -\cos\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Table B.1: Link parameters of MINERVA's head.

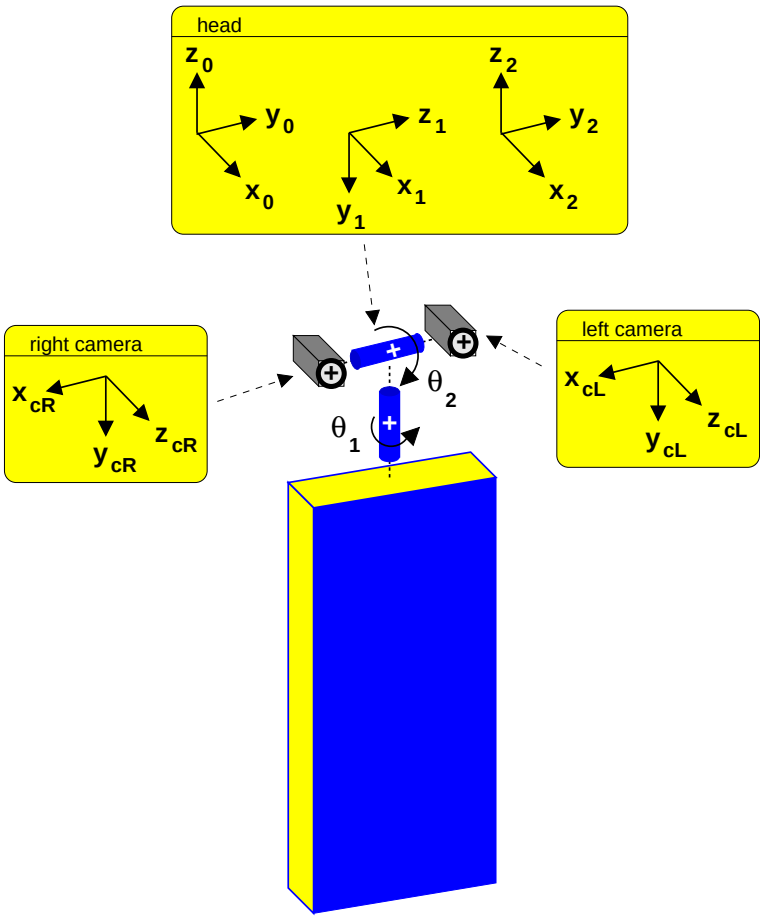


Figure B.1: Model of the Robots Head

changing the axes so that the z-axis points along the optical axis:

$$
\begin{aligned}
{}^{2}_{cL}\mathbf{T} &= {}^{2}_{cL}\mathbf{Trans} \cdot \mathbf{Rot}_z(\psi_L) \cdot {}_{cL}\mathbf{Rot} \\
&= \begin{bmatrix} 1 & 0 & 0 & t_{cL,x} \\ 0 & 1 & 0 & t_{cL,y} \\ 0 & 0 & 1 & t_{cL,z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c\psi_L & -s\psi_L & 0 & 0 \\ s\psi_L & c\psi_L & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} s\psi_L & 0 & c\psi_L & t_{cL,x} \\ -c\psi_L & 0 & s\psi_L & t_{cL,y} \\ 0 & -1 & 0 & t_{cL,z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(B.2)} \\
{}^{2}_{cR}\mathbf{T} &= \begin{bmatrix} s\psi_R & 0 & c\psi_R & t_{cR,x} \\ -c\psi_R & 0 & s\psi_R & t_{cR,y} \\ 0 & -1 & 0 & t_{cR,z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(B.3)}
\end{aligned}
$$

With the additional assumption that the x- and z-axes of the cameras are co-planar, i.e. equal translational offsets in z-direction, $t_{cR,z} = t_{cL,z}$, we obtain a simplified, planar model of the stereo camera system (Figure B.2). This model is of course an approximation of the real system, but errors occuring thereby can be tolerated as the experiments showed (see Section 5).

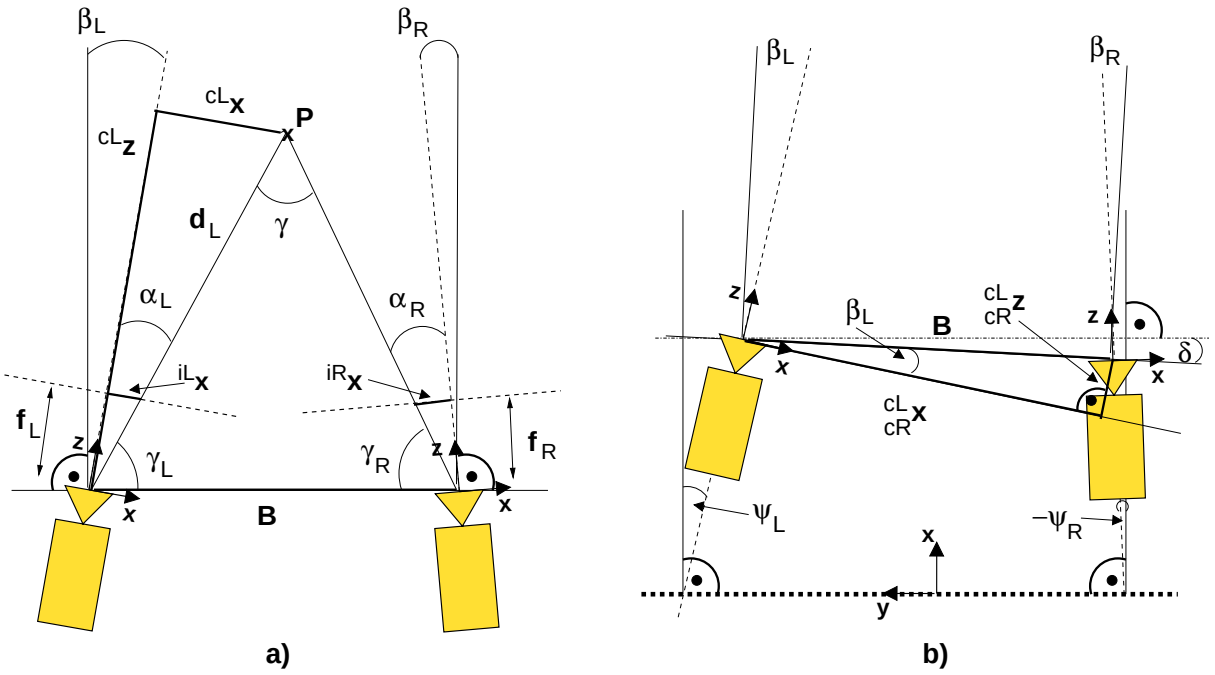


Figure B.2: Simplified model of the stereo camera system: (a) triangulation, (b) relation between stereo and head parameters.

The inverse model of the stereo camera system, i.e. how to reconstruct 3D information from pixel coordinates in the two images, ${}^{c}\mathbf{x} = f({}^{pL}\mathbf{x}, {}^{pR}\mathbf{x})$, is derived based on the pinhole camera model (see later paragraph *Pinhole Camera Model*). As could be seen in Equation B.16, the coordinates of a point ${}^{c}\mathbf{x}$ can be computed if the z-component is known. In our planar stereo model, ${}^{cL}z$ can be calculated using the trigonometric relations of Figure B.2a:

$$ {}^{cL}z = d_L \cdot \cos\alpha_L \tag{B.4} $$

$$ \alpha_L = \arctan\frac{{}^{iL}x}{f_L} \qquad \alpha_R = \arctan\frac{-{}^{iR}x}{f_R} \tag{B.5} $$

with $f_{\{L,R\}}$ being the respective focal lengths of the cameras. Using the tangential formula one can derive:

$$ \tan\gamma = \frac{B \cdot \sin\gamma_L}{d_L - B \cdot \cos\gamma_R} \tag{B.6} $$

with the baseline B, $\gamma_{\{L,R\}} = 90° - \beta_{\{L,R\}} - \alpha_{\{L,R\}}$, $\gamma = 180° - \gamma_L - \gamma_R$, and solve it for d_L:

$$d_L = \frac{B \cdot \cos(\alpha_L + \beta_L)}{\tan(\alpha_L + \beta_L + \alpha_R + \beta_R)} + B \cdot \sin(\alpha_L + \beta_L) \tag{B.7}$$

The computed position can be transformed from the frame of the left camera to head coordinates using Equation B.3. The parameters of the stereo model, B, β_L, and β_R, can be derived from the parameters of the model of the camera-to-head relation (Equation B.3) using the geometric relations illustrated in Figure B.2b: In the general case, the baseline is rotated by an angle δ in relation to the y-axis of the top head frame. Therefore, the vergence angles of the stereo model, β_L and β_R, depend on the vergence angles of the head model, ψ_L and ψ_R, as follows[1]:

$$\beta_L \;=\; \psi_L - \delta \qquad \beta_R \;=\; -\psi_R + \delta \tag{B.8}$$

Using the transformation between the two camera frames, ${}^{cL}_{cR}\mathbf{T}$

$${}^{cL}_{cR}\mathbf{T} = {}^{cL}_{2}\mathbf{T} \cdot {}^{2}_{cR}\mathbf{T} = \begin{bmatrix} \cos\psi_\Sigma & 0 & \sin\psi_\Sigma & \sin\psi_l \cdot \Delta t_x - \cos\psi_l \cdot \Delta t_y \\ 0 & 1 & 0 & 0 \\ -\sin\psi_\Sigma & 0 & \cos\psi_\Sigma & \cos\psi_l \cdot \Delta t_x + \sin\psi_l \cdot \Delta t_y \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.9}$$

with $\Delta t_x = t_{cR,x} - t_{cL,x}$, $\Delta t_y = t_{cR,y} - t_{cL,y}$, $\psi_\Sigma = \psi_L - \psi_R$, the stereo parameters can be computed as:

$$\beta_L = \arctan\left(\frac{{}^{cL}z_{cR}}{{}^{cL}x_{cR}}\right), \quad \beta_R = -\psi_R + (\psi_L - \beta_L), \quad B \;=\; \sqrt{({}^{cL}x_{cR})^2 + ({}^{cL}z_{cR})^2} \tag{B.10}$$

with ${}^{cL}x_{cR}$ and ${}^{cL}z_{cR}$ being translational components of ${}^{cL}_{cR}\mathbf{T}$.

Head-centered coordinates are transformed into arm-centered ones using ${}^{a}_{h}\mathbf{T}$, with ${}^{a}_{h}\mathbf{Rot}$ only "exchanging" the axes while neglecting any other rotations:

$${}^{a}_{h}\mathbf{T} = {}^{a}_{h}\mathbf{Trans} \cdot {}^{a}_{h}\mathbf{Rot} = \begin{bmatrix} 1 & 0 & 0 & t_{h,x} \\ 0 & 1 & 0 & t_{h,y} \\ 0 & 0 & 1 & t_{h,z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & t_{h,x} \\ -1 & 0 & 0 & t_{h,y} \\ 0 & -1 & 0 & t_{h,z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.11}$$

Model of the Camera

In the robotics and computer vision communities, video (CCD) cameras are described using a range of models. A good overview of these camera models is given in [MZ92] or [SZB95].

[1] Note, that the angles of the head model are both measured in the mathematically positive direction, while the angles of the stereo model are measured in different directions.

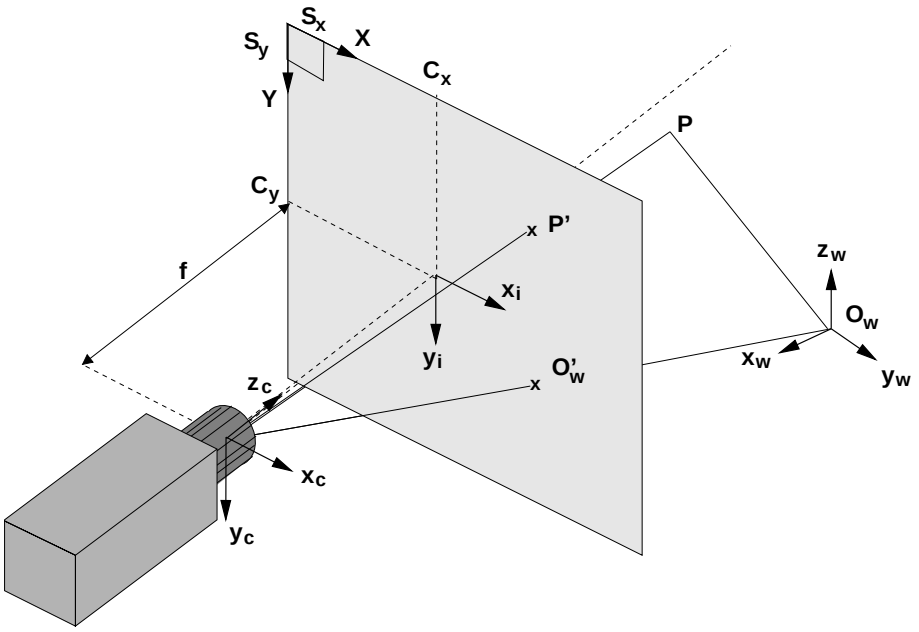


Figure B.3: Frames and parameters in the camera model.

When working with camera models, one has to address the question of how to estimate (calibrate) their parameters. The number of published calibration methods is in fact large. In developing a robotic hand-eye system, however, the estimation of model parameters is not a central problem. Therefore this issue is not discussed more deeply in the following and is restricted to the description of the calibration method we applied to our hand-eye system (see later paragraph *Calibration*).

The most commonly used camera model is the *pinhole camera model*. All measurements refered in this thesis are obtained by using this camera model. It is appealing because of its simplicity compared to other used camera models (e.g. the *projective camera model* or *camera models approximating perspective projection*, e.g. the *affine camera model*), but nevertheless the accuracy of measurements obtained with this model is far above the needed accuracy for the performed hand-eye experiments.

Pinhole Camera Model

A first approximation to the real optics of a camera is the so-called *pinhole camera model*[2]. The projection of a point onto the image plane, $P \mapsto P'$ is modeled as a central projection through the optical center, i.e. the center of the lens. This is illustrated in Figure B.3.

[2]The main approximation lies in not addressing the question of focus. In a real pinhole camera, the projection is always focussed, independent of the location of the image plane. This is not the case for the optics typically mounted on a CCD camera. See [BHS94] for a discussion of various other disturbances of the model on position estimation.

Note, that for a more intuitive visualization the image plane is depicted in front of the optical center instead of behind; therefore, the image does not appear inverted as it would be in reality. The distance between image plane and optical center, f, corresponds to the *focal length* of the camera.

Similar to Tsai's model [Tsa87], three frames in the camera model are used. The *camera frame* $\mathcal{F}_c$, is positioned in the center of the lens; its z-axis points along the optical axis. The (two-dimensional) *image frame* $\mathcal{F}_i$ is oriented as $\mathcal{F}_c$, but translated along the optical axis into the image plane. Its origin is called the *principal point*. The *pixel frame* $\mathcal{F}_p$, is oriented as $\mathcal{F}_i$, but translated into the left upper corner of the image. Coordinates in $\mathcal{F}_p$ are specified in *Pixel* instead of m. In the case of more than one camera, the frame identifiers are extended, as for example in $\mathcal{F}_{cL}$, which denotes the camera frame of the left camera in a stereo system.

Points ${}^c\mathbf{x}$ given in camera coordinates are projected onto points in the image plane, ${}^i\mathbf{x}$:

$$ {}^i\mathbf{x} = \begin{bmatrix} {}^ix \\ {}^iy \end{bmatrix} = \frac{f}{{}^cz} \cdot \begin{bmatrix} {}^cx \\ {}^cy \end{bmatrix} \tag{B.12} $$

Equation B.13 describes the transformation from image to the pixel coordinates, ${}^i\mathbf{x} \mapsto {}^p\mathbf{x}$:

$$ {}^p\mathbf{x} = \begin{bmatrix} {}^px \\ {}^py \end{bmatrix} = \begin{bmatrix} \frac{1}{S_x} & 0 & C_x \\ 0 & \frac{1}{S_y} & C_y \end{bmatrix} \cdot \begin{bmatrix} {}^i\mathbf{x} \\ 1 \end{bmatrix} \quad \rightarrow \quad {}^px = \frac{{}^ix}{S_x} + C_x, \quad {}^py = \frac{{}^iy}{S_y} + C_y \tag{B.13} $$

with S_x and S_y being scaling factors specifying the metric dimensions of a pixel ($[S_{x,y}] = \frac{m}{pix}$), and (C_x, C_y) being the pixel coordinates of the principal point.

Especially when using wide-angle lenses, radial distortions due to imperfect lenses or imprecise optical configuration become noticeable; distortions of radial distances in the image, ${}^ir$, can be modeled for example with the following geometrical sequence, with κ_i being weighting coefficients[3]:

$$ {}^ir_{dist} = {}^ir + \kappa_3 \cdot \left({}^ir\right)^3 + \kappa_5 \cdot \left({}^ir\right)^5 + \ldots, \qquad {}^ir = \sqrt{\left({}^ix\right)^2 + \left({}^iy\right)^2} \tag{B.14} $$

Distorted image coordinates, ${}^i\mathbf{x}_{dist}$, are computed in a first approximation by neglecting all terms with an order higher than three:

$$ {}^ix_{dist} = \frac{2 \cdot {}^ix}{1 + \sqrt{1 - 4\kappa \cdot {}^ir^2}} \qquad {}^iy_{dist} = \frac{2 \cdot {}^iy}{1 + \sqrt{1 - 4\kappa \cdot {}^ir^2}} \tag{B.15} $$

with $\kappa = \kappa_3$ being called the *radial distortion coefficient*[4].

[3] This is but one possible model. It is the base of the calibration method we applied ([LZB95, HAL]

[4] Distortions with $\kappa < 0$ are called *pillow-shaped*, as the sides of a projected rectangle will be curved inward like a pillow. Distortions with $\kappa > 0$ are called *barrel-shaped*, as the sides of a projected rectangle will be curved outward.

f, S_x, S_y, C_x, C_y, and κ constitute the so-called *internal* or *intrinsic parameters* of the (forward) camera model. The inverse model, i.e. the mapping from pixel coordinates to 3D points, ${}^{p}\mathbf{x} \mapsto {}^{c}\mathbf{x}$, can be computed in an unambiguous way up to the image coordinates ${}^{i}\mathbf{x}$; then, however, the only 3D information one can derive is that the point projected lies on the ray through ${}^{i}\mathbf{x}$ and the optical center, with λf being equivalent to the unknown distance, ${}^{c}z$:

$$ {}^{c}\mathbf{x} \in g : \mathbf{x} = \lambda \cdot \begin{bmatrix} {}^{i}x \\ {}^{i}y \\ f \end{bmatrix} \qquad \lambda \geq 0 \tag{B.16} $$

If the points to project are given relative to the world frame, they first have to be transformed into camera coordinates. The six pose parameters corresponding to the transformation matrix, ${}^{c}_{w}\mathbf{T}$, are called the *external* or *extrinsic parameters* of the camera model. The complete projection model without radial distortion can be denoted as follows:

$$ {}^{c}z \cdot \begin{bmatrix} {}^{p}\mathbf{x} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{S_x} & 0 & 0 & C_x \\ 0 & \frac{f}{S_y} & 0 & C_y \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot {}^{c}_{w}\mathbf{T} \cdot \begin{bmatrix} {}^{w}\mathbf{x} \\ 1 \end{bmatrix} \tag{B.17} $$

Calibration

The internal models described in the former section each possess a set of parameters that have to be estimated. In the case of the arm model, these are the lengths of upper arm s, of the lower arm t, and the distance from the wrist to the gripper, u. The first two parameters were extracted from the manufacturer's specifications. The distance from the wrist to the end of the gripper was measured using a measuring tape.

Intrinsic Camera Parameters. The intrinsic parameters of the pinhole camera model were determined using the camera calibration method provided by the image analysis software *HALCON* [HAL]. This method determines the intrinsic parameters including the radial distortion coefficient from multiple views of a 2D calibration table [LZB95]. The calibration setup is depicted in Figure B.4a.

Camera pose relative to head. In fact, the Calibration algorithm does not only estimate the intrinsic parameters, it also estimates the pose of the calibration table relative to the camera, respectively the transformation matrix between the frames of the calibration table and the camera, ${}^{c\{L,R\}}_{cal}\mathbf{T}$ (see Figure B.4b). This can be used to calibrate the pose of the cameras relative to the top head frame, ${}^{2}_{c\{L,R\}}\mathbf{T}$ by keeping the pose of the calibration table constant and obtaining multiple views by moving the head [LZ95]. The principle idea is to write an equation of transformations

$$ {}^{cal}_{cR}\mathbf{T} = {}^{cal}_{0}\mathbf{T} \cdot {}^{0}_{2}\mathbf{T} \cdot {}^{2}_{cR}\mathbf{T} \tag{B.18} $$

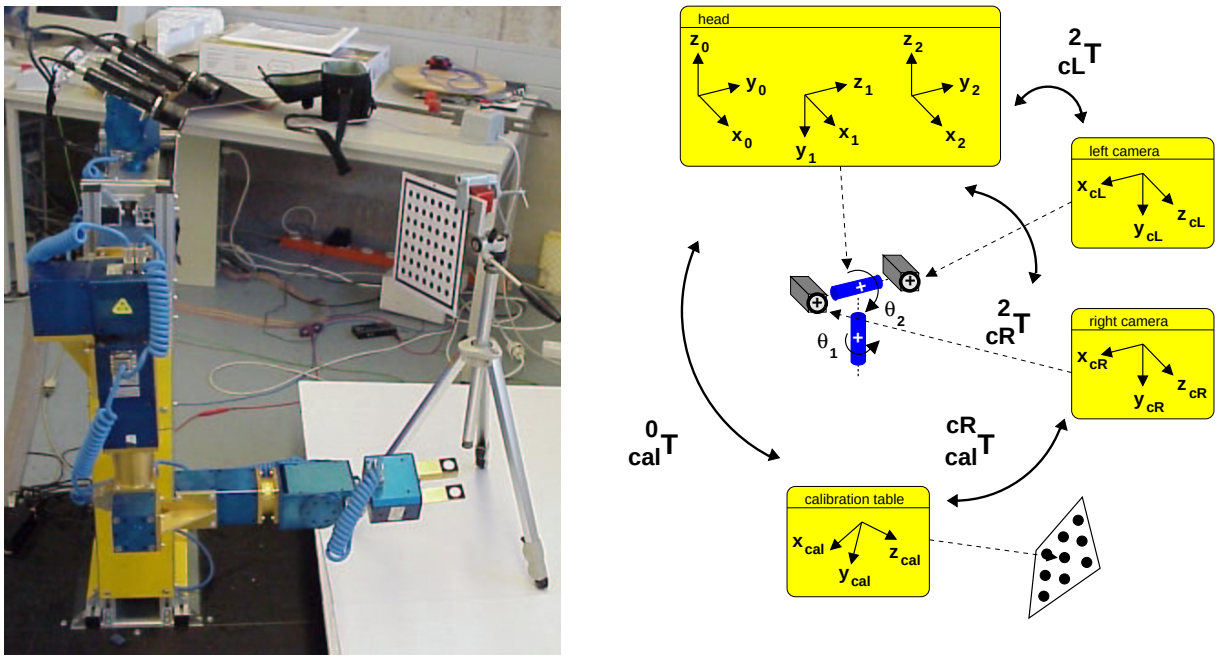


Figure B.4: (a) Calibration setup, (b) chain of transformations

in which the left side is estimated from visual information, and the right side consists of one unknown but constant transformation, ${}^{0}_{cal}\mathbf{T}$, one changing but known transformation, ${}^{2}_{0}\mathbf{T}(\theta_1, \theta_2)$, and the also unknown and constant transformation of interest, ${}^{2}_{cR}\mathbf{T}$.

Transformation from Head into Arm Coordinates. The method described in the previous paragraph could also have been applied to calibrate the transformation between head and arm frame by putting the calibration table into MINERVA's hand and moving it while keeping the head fixed. However, this would have been a rather time-consuming process. Instead, we assumed the simplified relation given in Eq. B.11 and measured the three translational offsets with a measuring tape.