Direct Methods for Vision-Based Robot Control

Application and Implementation

Fast Focus On Structures

Embedded Vision Architecture

This research was supported by Agentschap NL - IOP Precision Technology - Fast Focus On Structures (FFOS) and the Dutch Ministry of Economic Affairs (Pieken in de Delta) - Embedded Vision Architecture (EVA).

Direct Methods for Vision-Based Robot Control: Application and Implementation,
by R.S. Pieters, PhD thesis, Eindhoven University of Technology, The Netherlands, 2013.

A catalogue record is available from the Eindhoven University of Technology Library.
ISBN: 978-94-6191-648-8

This thesis was prepared with the pdfLaTeX documentation system.
Reproduction: Ipskamp Drukkers B.V., Enschede, The Netherlands.
Cover Design: R.S. Pieters

Author details:
pietersroel@gmail.com
http://rspieters.wordpress.com/

Direct Methods for Vision-Based Robot Control

Application and Implementation

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Eindhoven,
op gezag van de rector magnificus, prof.dr.ir. C.J. van Duijn,
voor een commissie aangewezen door het College voor Promoties
in het openbaar te verdedigen
op maandag 25 maart 2013 om 16.00 uur

door

Roel Stephan Pieters

geboren te Meerssen

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr. H. Nijmeijer
en
prof.dr.ir. P.P. Jonker

Summary

With the growing interest of integrating robotics into everyday life and industry, the requirements towards the quality and quantity of applications grows equally hard. This trend is profoundly recognized in applications involving visual perception.

Whereas visual sensing in home environments tend to be mainly used for recognition and localization, safety becomes the driving factor for developing intelligent visual control algorithms. More specifically, a robot operating in a human environment should not collide with obstacles and executed motion should be as smooth as possible. Furthermore, as the environment is not known on beforehand, a high demand on the robustness of visual processing is a necessity.

On the other hand, in an industrial setting, the environment is known on beforehand and safety is mainly guaranteed by excluding a human operator. Despite these reasons, and the fact that visual servoing has gained much attention from industry to become a standard solution for robotic automation tasks, applications are highly simplified. For example, methods such as visual fault detection are already a mature technique in industrial manufacturing, where a fixed camera observes a product (e.g., on a conveyor belt) and checks whether it meets certain requirements. These operations can be executed at a fairly high rate due to the simplicity of the system (e.g., static camera) and the simplification of the processing task (e.g., binary images).

For both areas the identified difficulties are similar. Foremost, this is the slow nature of (robust) visual processing, in respect to the ever growing demand of increasing speed and reducing delay. These two application areas with analogous limitations motivate the design of more direct approaches of vision in visual control systems. Therefore, in order to meet the requirements for next generation visual control systems, this thesis presents approaches which employ visual measurements as a direct feedback to design constrained motion.

First, for industrial robotics, in order to obtain the required positioning accuracy, the measurement and fixation system have to be highly rigid and well-designed, implying high cost and long design time. By measuring the position of objects directly with a camera, instead of indirectly by motor encoders, the requirements of the measurement and fixation system are less demanding. Moreover, this motivates the miniaturization of the complete control system. The approach is validated in experiments on a simplified 2D planar stage (i.e., considerable friction, poor fixation), which attains similar performance compared to encoder-based positioning systems.

Secondly, in a human-centred environment, this direct sensing can improve traditional visual control systems, when subject to certain disturbances. More specifically, a method is proposed that uses an image-based feedforward controller on top of traditional position-based visual servo control to overcome disturbances such as friction or poorly designed local motor controllers. This

visual feedforward control action is only active when an image-based error is present and vanishes when that error goes to zero. The method is validated on an anthropomorphic robotic manipulator with 7 degrees of freedom, intended for operation in the human care environment.

Third, sensing the product directly gives rise to designing motion directly. Whereas in traditional approaches the motion trajectory is designed offline and can not be changed at runtime, direct trajectory generation computes the motion of the next step based on current state and events. This means that at any instance in time, the trajectory of a motion system can be altered with respect to certain desired kinematic or dynamic constraints. For industrial applications this makes manufacturing on near-repetitive or non-rigid structures (e.g. flexible displays) possible. When applied to a robotic manipulator, this enables obstacle avoidance to no longer be on path planning level, but on trajectory planning level, where kinematic or dynamic constraints can be taken into account. This results in a motion that is smoother than when obstacle avoidance with path planning is employed. For both application areas this direct trajectory generation method is implemented and shows high flexibility in constrained motion trajectory design.

Contents

CHAPTER 1

Introduction

Abstract. This chapter gives an introduction to the topic of this thesis. The difficulties of vision-based robot control are addressed and motivations are given for the relevance of this work. In addition, clear research objectives are stated and an outline of the remainder of this thesis is given.

1.1 Introduction

In recent years, robot technology has matured in a way that safe integration in industry is a commodity. This development is motivated by the advantages that a robot offers, compared to a human operator. Properties such as high accuracy and repeatability, continuous operation and therefore considerable savings cannot be matched by human labour. One of the earliest applications for industrial robots can be found in assembly lines[1], where repetitive tasks have to be executed at a conveyor belt (e.g., pick and place tasks, spray painting, spot welding). Even though the environment does not change and events that occur within the robot's reach (e.g., moving parts) are deterministic, safe operation still has top priority. This is mainly due to the fact that a robot is unaware of its environment and sensing is usually limited to the task at hand. This is therefore the main reason for excluding human operators in the working range of the robot.

When a robot needs to operate in an unknown and indeterministic environment, the intelligence for sensing and motion can not be provided in a straightforward manner and entails a complexity several orders of magnitude higher than traditional robot control. This holds particularly for visual sensing which provides a very rich and unordered bulk of information and leaves the segmentation of this data into useful form a non-trivial task. Although the research field of computer vision receives considerable interest and tasks that require visual sensing attain a certain maturity, still the foremost limitation that prevents vision from being integrated directly as a safe sensing technique in robotics is the level of complexity. Due to this complexity, a fundamental difference can be distinguished between traditional robot control and vision-based robot control. In particular, traditional robot control lacks global sensing and essentially executes tasks blind. On the other hand, vision-based robot control separates constrained motion from path planning by focussing on the design of a path (or direction) for positioning. Motion constraints are thus handled by the local motor controller. Moreover, the simple nature of traditional control allows for fast update rates, while vision-based control is restricted to a slow update rate. This observation is also of importance when regarding real-time performance. In general it is implied that the computation time of measurements should not

[1] http://www.prsrobots.com/1961.html

compromise the performance or stability of the motion system. For vision-based control, typical solutions which avoid such difficulties include a local control loop for stability (i.e., ensuring a motion command is quickly reached) and a visual loop which determines motion commands.

In comparison to the human visual system, as much as forty percent of the human brain is devoted to visual processing. The visual cortex, which is the largest system in the human brain, is functionally divided in several areas for processing, where different areas account for different stimuli. Moreover, the slow nature of human vision (i.e., the update rate of the human visual system does typically not exceed 20-30 [Hz].) suggests that a higher level interpretation is responsible for a large part of extracting relevant information. Indeed, combined with the fact that visual processing can be guided (top-down or bottom-up) this proves that the level of parallelism in processing exceeds the current standards in computation power greatly. On top of this, even though Moore's Law states that the overall processing power for computers doubles every two years, greater processing power does not necessarily imply that current solutions would be adequate.

A solution to this mismatch of incorporating global information with constrained motion can therefore be guided in two directions. On the one hand processing can be accelerated to account for the poor performance of vision-based control by employing different processing platforms and limiting the complexity of the process. On the other hand, a combination of traditional robot control and vision-based control would complement each other such that constrained vision-based motion becomes possible.

In this respect, in order to meet the requirements for next generation visual control systems, this work presents insight and solutions to this typical visual control problem.

1.2 Motivation

The work presented in this thesis is motivated by regarding two robotic case-studies: industrial inkjet printing and vision-based service robotics. For both applications, a similar approach towards motion planning can be executed. That is, for industrial inkjet printing, the introduction of a camera in the control loop allows for more complex (i.e., constrained) motion design. For vision-based service robotics a camera is already present, however, motion is not kinematically constrained. Following, this novel approach towards motion planning (i.e., online constrained vision-based motion planning) is highlighted, the difficulties are addressed and an introduction to solutions is proposed.

1.2.1 Vision-Based Automation

Robotics in an industrial setting execute a wide variation of tasks. Examples are for instance pick-and-place tasks, welding, fault detection, and spray painting. The overall similarity of these tasks is that motion should ensure a certain accuracy and repeatability with respect to a certain position or object. Common practise is the use of a specially designed fixation system (e.g., end-stop, product carrier) such that the robot executes the same motion for every product or task sequentially. Variability of the position of the product is as such minimized

and machine vision techniques can be used to achieve a higher robustness for product localization. The desired motion can then be programmed by either online or offline programming techniques. A typical online programming technique is known as programming-by-demonstration where a user manually moves the end-effector of the robotic manipulator to a desired position and orientation and records the relevant robot configurations (i.e., in joint space). These poses are then sequentially set as goal configurations for the manipulator. Offline programming techniques are typically based on models (kinematic or dynamic) of the robotic manipulator where simulations are executed to obtain a desired motion. For both techniques motion control is executed with feedback directly obtained from local motor encoders. Together with the high performance of positioning that industrial robotics can achieve, this is the main reason for the limited use of machine vision in industry.

In particular, consider the industrial application of inkjet printing. In next-generation display technologies this manufacturing process is used to fabricate OLED (Organic Light Emitting Diode) displays. On either rolls of plastic or large sheets of glass, a nozzle has to be positioned over a repetitive structure (i.e., a pixel on the display) where the print-head shoots a droplet of polymer onto the substrate (see Fig. 1.1, left). In order to obtain the required positioning accuracy, the measurement and fixation system have to be highly rigid and well-designed, implying high cost and long design time. In this case, vision is not used for positioning and only encoder-based feedback controls the motion stage.

However, current developments in the display industry demand manufacturing solutions that guarantee similar specifications while increasing product quality (i.e., bigger screens and higher resolutions at lower cost). Furthermore, radical designs in display technology (i.e., flexible displays, see Fig. 1.1, right) even demands manufacturing that cannot be carried out by current state-of-the-art fabrication solutions. Manufacturing products of this nature needs additional sensing to detect exactly where an operation has to be executed. Whereas current solutions assume an undeformable substrate with a constant pitch between pixels, in the case of flexible displays these assumptions no longer hold.

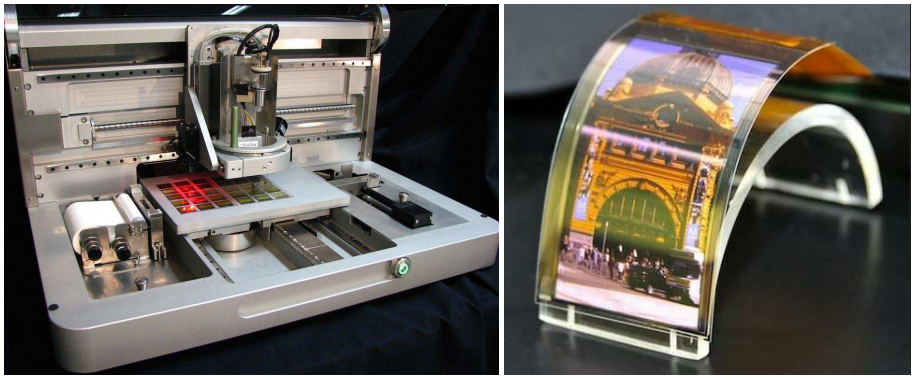


Figure 1.1: Left: Industrial inkjet printer (Image courtesy of OTB Solar - Roth & Rau). Right: flexible display (Image courtesy of Sharp).

This motivates visual sensing with direct feedback to be incorporated into the existing motion stage and enables the design of motion and subsequent manufacturing to be directly dependent on these measurements. Moreover, by measuring the position of objects directly with a camera, instead of indirectly by motor encoders, the requirements of the measurement and fixation system can be less demanding.

1.2.2 Vision-Based Service Robots

Service robots are being developed to assist humans in a wide range of tasks that are considered dull, dirty or dangerous (DDD). Examples range from single purpose robots (e.g., vacuum or lawn mower robots) to robots that are designed as a general autonomous assistant. For such autonomous service robot tasks can be highly complex. Examples are for instance grasping and manipulation in 3D where a path has to be found in a cluttered environment and dynamic objects (as well as the robot itself) can obstruct the manipulator at any moment. This has motivated the development of service robots with anthropomorphic properties as can be seen in Fig. 1.2. This human-like similarity is not only limited to the configuration of a manipulator, but is also often considered for the visual system. Common examples include for instance dual 7-DOF manipulators and a stereo vision system on a pan-tilt stage. Such design offers the robot a freedom and redundancy in sensing and manipulation as is proven by human visual-motor coordination. Considering this redundancy, the extra (i.e., redundant) degrees of freedom of a manipulator can be used for secondary objectives which are not directly related to the main task. Examples of these include the avoidance of obstacles, joint limits and even singularities of the manipulator.

Despite these complex designs, the actual tasks that a service robot can effectively execute are fairly limited and commonly motion has a fairly simple nature (e.g., detect object, pick up object, move object with low velocity motion). Moreover, each individual process (i.e., vision, trajectory planning and control) is separately or independently computed, resulting in slow movements and a slow response.

By studying these issues in more detail, we consider a robot in the human care environment. Robots intended to operate in a human-centred environment have as main priority to maintain a safe operation. This suggests that the executed motion is as smooth as possible and collisions should be avoided at all cost. In most situations the robot has to navigate in an unknown and cluttered environment, where moving obstacles have to be avoided. This means that a path is not known beforehand and has to be planned at runtime. Current solutions solve this problem at a path planning level by determining free points in space for which motion is safe. Subsequently, motion is executed that positions the manipulator from point to point, where, whenever possible, smooth short-cuts are incorporated.

The consequence of this is that kinematic or dynamic constraints are not taken into account and the motion of the robot can not be guaranteed as smooth as possible. Moreover, motion is slow as visual measurements are updated and incorporated (i.e., vision-based control) at a much lower rate than the rate of local motor control. Furthermore, when regarding a robotic manipulator

with multiple degrees of freedom, obstacle avoidance does not only entail the motion of the end-effector but also collisions that may occur with the remaining links of the robot.

These typical problems in vision-based motion control are a clear motivation for the combination of traditional motion control with vision-based control into one direct control solution. A direct trajectory generation method will integrate sensing directly into the trajectory design to form constrained motion.

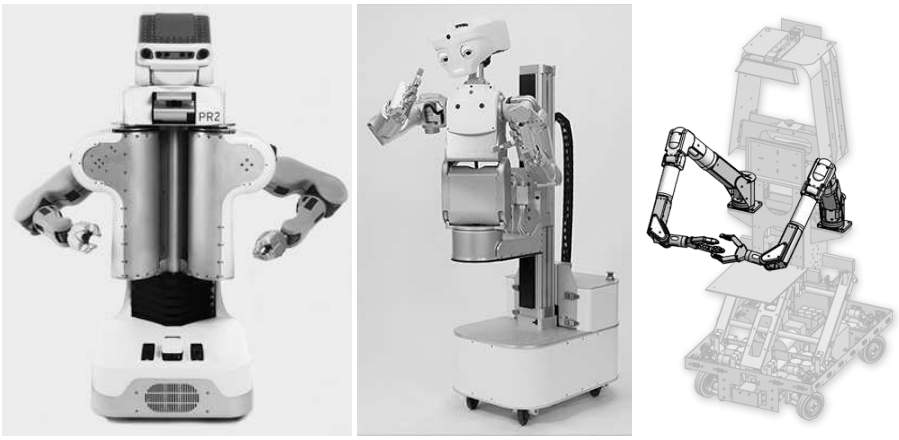


Figure 1.2: Left: Personal Robot (PR2) developed by Willow Garage. Middle: Meka M1 mobile manipulator developed by Meka Robotics. Right: Remotely Operated Service Robot (ROSE) developed by Eindhoven University of Technology.

1.3 Research Objectives and Contributions

Based on the challenges in state-of-the-art visually controlled robotics, as addressed in previous sections, the main objective of this thesis is formulated as:

Design methodologies that provide robots with the ability to use visual measurements in their activities in a direct and constrained way.

By putting more intelligence in visual perception (sensing) and the subsequent design of motion (action), typical problems in vision-based motion systems can be avoided. In more detail, the research objectives are stated as follows.

Research Objectives

1. *Develop and experimentally validate vision-based control methodologies that can incorporate direct visual measurements into robotic motion design.*

2. *Develop and experimentally validate a trajectory planning methodology that incorporates constraints and changes into online trajectory design.*

Contributions

The contributions of this work can be described as follows:

- Feedforward Visual Servoing
  For traditional vision-based control, an additional feedforward control action can improve motion performance, when subject to certain disturbances. More specifically, a method is proposed that uses an image-based feedforward controller on top of traditional position-based visual servo control to overcome disturbances such as high friction or poorly designed local motor controllers. This visual feedforward control action is only active when an image-based error is present and vanishes when that error goes to zero.

  The method is validated on an anthropomorphic robotic manipulator with 7 degrees of freedom, intended for operation in human care environments.

- Direct Visual Servoing
  Visual servoing encompasses that positioning is executed with respect to an object. Direct feedback is preferable as direct measurements bypass the uncertainties in system modelling and the non-rigidity of the measurement and fixation system. When the processing time is short enough this enables a direct feedback to local joint controllers. Moreover, this dismisses the need of local motor encoders and motivates the miniaturization of the complete control system.

  The approach is validated in experiments on a simplified 2D planar stage (i.e., considerable friction, poor fixation), which attains similar performance compared with encoder-based positioning systems. The application at hand is an industrial inkjet printing task where a near-repetitive pattern serves as visual encoder.

- Direct Trajectory Planning
  Traditional motion control designs a motion trajectory offline, which cannot be changed at runtime. The concept of sensing the product directly now gives rise to designing motion directly. In particular, direct trajectory generation enables the design of constrained motion for the next time increment based on the current state and events and a predefined trajectory outline. This means that at any instance in time and at an arbitrary state, the trajectory of a motion system can be altered with respect to certain desired kinematic or dynamic constraints.

  When considered for industrial applications such as industrial inkjet printing, it enables the manufacturing of near-repetitive or non-rigid structures (e.g., flexible displays).

  When considered for robotic manipulators, it enables a direct motion response which is not possible with current state-of-the-art solutions. Obstacle avoidance is then no longer designed on a path planning level, but on a trajectory planning level, where kinematic constraints can be taken into account. This results in a motion that is more smooth than when obstacle avoidance with path planning is employed.

Direct trajectory generation can be either event-based or rate-based. Event-based trajectory generation implies a motion that can be altered online whenever an event occurs. Rate-based trajectory generation implies that motion updates are incorporated at every iteration and as such takes (changing) kinematic constraints into account at every time increment. Moreover, as is possible with traditional trajectory generation, motion can be designed with point-to-point motion or multi-point motion.

1.4 Outline

This thesis is divided in three main parts: I. Modelling and Planning of Robotic Manipulators, II. Visual Control of Robotic Manipulators, III. Application and Implementation (see Fig. 1.3). Each part is written as an independent section, however, wherever necessary, reference to other sections is given.

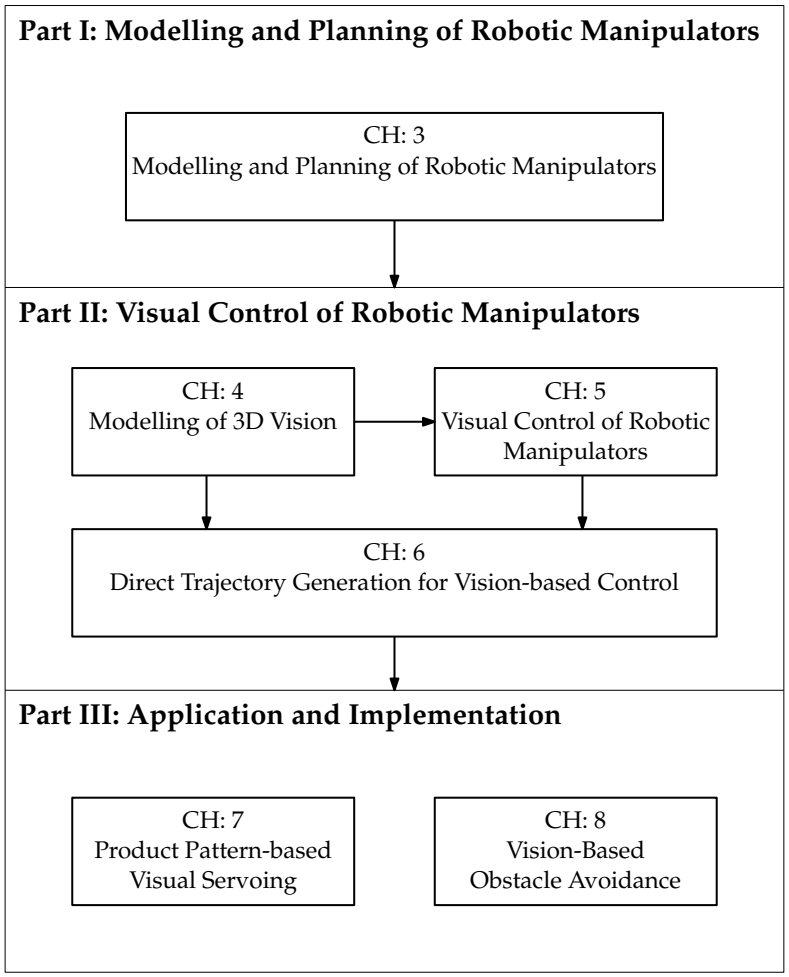


Figure 1.3: Block scheme of the contents of this thesis.

As a prerequisite for the remainder of this work, Chapter 2 is devoted to a literature study. A global overview is presented which lists research and developments in visual servoing. This includes traditional and novel methods in visual control as well as developments in path and trajectory planning.

Outline of Part I: Modelling and Planning of Robotic Manipulators

In the first part, the general concept of modelling and planning of robotic manipulators is presented.

Chapter 3 presents the basics in dynamic and kinematic modelling of robotic manipulators. The focus lies on kinematic modelling as this allows for separating the dynamic behaviour of a manipulator from its kinematic relations. Control can then be executed on a velocity (first-order) level which assumes a separate controller that guarantees velocity tracking with appropriate feedback. This also allows for the exploitation of the redundancy of the robot for secondary tasks. Furthermore, an introduction is given for the planning of motion of robotic manipulators. This is divided in the planning of a path and the planning of a trajectory for both Cartesian and joint space.

Outline of Part II: Visual Control of Robotic Manipulators

The main part of this thesis, presents basic and original work on the visual control of robotic manipulators.

Chapter 4 discusses the modelling of 3D vision for retrieving 3D measurements from single view cameras. The geometry between two views enables the estimation of a planar homography, which can be decomposed into a rotational and translational part. The estimation of this planar homography requires a set of point correspondences as input. Two keypoint detection and matching methods are discussed and their properties are evaluated in experimental setting.

Chapter 5 presents several traditional approaches of visual control and discusses their pros and cons. A novel hybrid visual servoing method is proposed which combines two traditional methods and as such exploits their advantages. The new approach is verified in simulation and experimental setting with a 7-DOF robotic manipulator.

Parts of this chapter are presented in the following publication: [111].

Chapter 6 proposes a novel trajectory generation approach for robotic manipulation. The new method, denoted Direct Trajectory Generation (DTG), is inspired from both traditional motion planning and vision-based motion planning. As such, it can incorporate direct changes of the trajectory and its constraints by updating the trajectory generation every iteration. The approach is analysed and its properties are discussed and simulation and experimental results are carried out for a single degree of freedom system.

Parts of this chapter are presented in the following publications: [115] and [110].

Outline of Part III: Application and Implementation

The final part of this thesis presents two different applications and their implementation using the methods presented in the former part of this work.

Chapter 7 presents the application and implementation of industrial inkjet printing. A near-repetitive pattern serves as visual input for motion generation, where the method of direct trajectory generation incorporates constraints on individual pattern structures. The camera then serves as single feedback for motion control. Experimental results are presented for a 2D planar motion stage with positioning on micrometer scale.

Parts of this chapter are presented in the following publications: [112], [113], [114] and [115].

Chapter 8 presents the application and implementation of obstacle avoidance for a robotic manipulator. 3D visual measurements and the direct trajectory generation approach are designed such that a constrained avoidance motion becomes possible. This effectively means that obstacle avoidance is no longer executed on a path planning level, but on a trajectory planning level. In addition, this method is combined with obstacle avoidance for the self-motion of the manipulator. Experimental results are presented for a 7-DOF redundant manipulator.

Parts of this chapter are presented in the following publication: [110].

1.5 Research Projects

The research presented in this thesis has been performed within the project *Fast Focus On Structures* (FFOS) and was financed by IOP Precision Technology. The objective of the FFOS project is the development of a flexible, low-cost, miniaturized measurement system for the accurate positioning of a production head with respect to a product. The project is carried out by a joint consortium of industrial and academic partners. The industrial partners are OTB Engineering B.V. (now Roth & Rau B.V.), Agilent Technologies Netherlands B.V., M2control, NTS Group and Mitutoyo. In the FFOS project two separate Ph.D. projects have been initiated. The first one has led to the thesis of Jeroen de Best, [34], which focusses on motion control for near-repetitive structures. The second project has led to the present thesis.

The research presented in this thesis has also been performed within the project *Embedded Vision Architecture* (EVA) and was financed by the Dutch Ministry of Economic Affairs (Pieken in de Delta). The objective of the EVA project is the design of architectures and algorithms for vision systems that are embedded in electromechanical equipment for industrial inspection and production. The project is carried out by a joint consortium of industrial and academic

partners. The industrial partners are Chess, Assembléon, Philips and OTB Engineering B.V. (now Roth & Rau B.V.). As such, the EVA project has adopted the FFOS case (i.e., industrial inkjet printing in particular) as practical application. This research involves the implementation of the complete vision pipeline on a FPGA (Field-Programmable Gate Array) processor and a SIMD (Single Instruction Multiple Data) processor.

Moreover, research has also been performed in collaboration with the project *Remotely Operated Service Robot* (ROSE). The objective of the ROSE project is the development of a remotely operated service robot for home care applications. This service robot assists humans in a home environment while being operated from a remote location.

CHAPTER 2

Literature Review

Abstract. This chapter presents a global overview of research and developments in visual servoing. Individual sections cover traditional and novel topics in visual control that are relevant for this thesis. A division is made between historical, traditional and modern work in visual servoing, as well as a review on developments in path and trajectory planning. Finally, recent work on visual control is discussed that most resembles the developments as presented in this thesis. A few similar studies are highlighted and a comparison is made to emphasize the differences between both.

To limit this review to the field of visual control, a general review for the topics of modelling and control of robotics as well as visual processing is not considered. Furthermore, this review discusses a global overview of existing methods. A detailed explanation of visual control and its mathematical details can be found in following chapters.

2.1 Historical Origins

One of the earliest works in visual servoing can be traced back to the early 1970's. In 1973, Shirai and Inoue [133] developed a system that computes the difference between the desired position of a block and the actual position by visual processing and corrects the motion of a robotic manipulator accordingly. The task is to put the block into a box and shows that, when incorporating vision in such control systems, a higher accuracy can be achieved. As it takes about 10 seconds to recognize the box, the visual loop is executed at 0.1 [Hz].

Hill and Park [61] formally introduced the term *visual servoing* in 1979. The presented work describes visual servoing with a Unimate[1] robot for both a planar case and a 3D positioning case. An initial introduction to the different distinctions of visual servoing is made in 1980 by Sanderson and Weiss [124]. The described taxonomy is between the direct and indirect approach towards visual feedback. In particular, direct visual servoing generates a control signal from visual data directly to the robot's joints, while for indirect visual servoing visual data only generates a reference for motion control and a separate, local joint controller executes motion control.

Since these pioneering contributions several surveys and reviews were published that describe the many different methodologies and applications in the field over the years [124], [31], [64], [81]. A most useful introduction for anyone unfamiliar with visual control are the tutorials by Chaumette and Hutchinson [23], [24]. These present the basics as well the advanced approaches and even shed light on some unresolved issues. A recent collection of state-of-the-art research is a published work by Chesi and Hashimoto [26]. Focussing on ad-

[1] http://www.prsrobots.com/unimate.html

vanced numerical methods, it is divided in the main sub-fields of visual control, i.e., vision, control and planning.

2.2 Traditional Visual Servoing

As visual servoing is a well studied research topic, many different sub-fields have emerged over the years. The basic techniques, however, rely on the same general control scheme which is defined to minimize an error as obtained from visual data (see Fig. 2.1). In this, a double control loop structure can be identified, where the visual loop designs the motion to be executed by the robot and a local loop controls the joints of the robot.

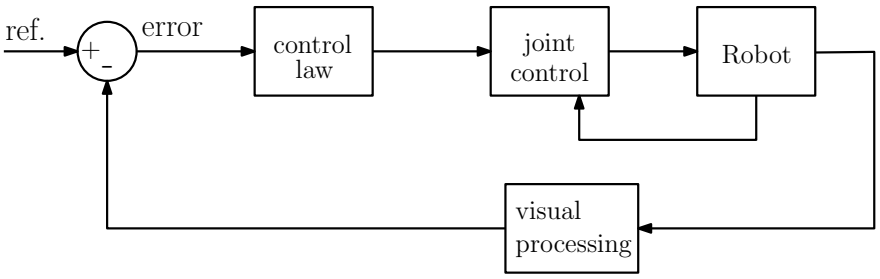


Figure 2.1: General visual servo control structure. In this, a local loop controls the joints of the robot and a visual loop designs the motion to be executed by the robot.

Different visual servo schemes mainly differ from how this error is obtained. A survey and review on the traditional approaches can be found in [64] and [31], a performance review in [47]. The two classical visual servoing methods, image-based and position-based visual servoing (IBVS, PBVS), are extensively used in practice and recent research on them focuses on their performance, stability and behaviour in the face of uncertainties [23], [24].

Position-Based Visual Servoing

Position-based visual servoing (PBVS) employs an estimated object pose with respect to the camera as main control objective [152]. As this visual control method operates in 3D Cartesian space, when using only a single camera, full knowledge of the intrinsic parameters of the camera (i.e., focal length, image format and principal point) is essential. The error is obtained from visual processing, most commonly with either an object model [39], sophisticated processing or simplification of the object model [92].

PBVS allows for two different solutions. One solution decouples the translational and rotational motions, resulting in a straight line trajectory for the camera in Cartesian space [23]. This advantageous property lets the camera follow a deterministic trajectory which is the shortest path in Cartesian space.

However, the main disadvantage is the inability to control image features directly. It may occur that image features (or the object) leave the field of view (FOV), which may compromise the stability of the system. Much research has been targeted at solving this issue. Wilson et al. [154] use an extended Kalman

filter that estimates the state of the system (i.e., position and orientation) which can cope with the loss of features. In this, features are represented in the covariance matrix, which is altered according to the presence and location of a feature in the field of view. In essence, the effect of a missing feature is removed from the pose estimation.

Thuilot et al. [143] proposed a PBVS method to keep the object in the field of view by tracking an iteratively computed trajectory. In this the rate of convergence for rotational and translational motion is separated. Rotational errors can decrease without constraints, while translational errors are adjusted to the field of view constraint.

A known issue in visual servoing is the difficulty of studying the stability properties of the complete system. For PBVS this is due to the sensitivity to pose estimation errors. More specifically, PBVS can be proven to be globally asymptotically stable, under the assumption that the pose estimation is perfect [23].

Image-Based Visual Servoing

Image-based visual servoing (IBVS) takes image measurements directly as control objective. The task function is expressed as an error-function which should be minimized, by using a suitable control law. This function is thus defined from the displacement between corresponding 2D image feature points expressed in planar Cartesian coordinates. As one image feature point is detected in a 2D image space, a minimum of 3 feature points are necessary to control the 6 Cartesian degrees of freedom (DOF). Moreover, due to this feedback in image space, the control executes straight lines in image-space, whereas motion trajectories in Cartesian-space are not considered.

Consequently, this property is the main disadvantage, as it may cause excessive camera motion. This can be easily shown when a pure rotation around the camera's optical axis occurs [22]. This so-called *camera retreat* problem commands a translational motion backward and forward along the camera's optical axis, instead of commanding only a rotational motion. A worst case scenario is encountered when the rotational error is exactly 180 degrees around the camera's optical axis, which then commands the camera to retreat to infinity.

One advantage, is the fact that a 3D model is not necessary and therefore, usually, basic image processing suffices. Traditional IBVS uses the error between corresponding image features that lie on a Cartesian plane.

Besides 2D image features, other types of features are studied and applied as visual measurements for control feedback. IBVS with 3D features [19, 126] takes the depth of 2D image points into account in the control error. Andreff et al. [5] use 3D lines for servoing. The control law is derived for one or multiple lines and requires the depth to be observed.

Moments as visual features is studied in [139] which derives the analytical form of the interaction matrix (i.e., the matrix that relates camera and feature velocities) related to any feature moment. Six combinations of moments are selected to control the six degrees of freedom of the system. The method is designed for configurations such that the object and camera planes are parallel. An extension to this configuration being non-parallel is proposed in [140].

Regarding the stability properties, IBVS is more robust towards calibration errors than PBVS [23]. However, for IBVS, only local asymptotic stability can be proven due to the existence of local minima [23]. These local minima exist due to the design of the task function. Commonly, the number of image features for control exceeds three, creating a nontrivial nullspace of the interaction matrix.

Distinction of Traditional Approaches

For the traditional visual servoing methods, the overall architecture of the robotic visual control system is not regarded. Numerous designs with respect to a different control system, measurement system and their implementation are therefore briefly highlighted [64, 23, 24].

Regarding the placement of the camera, two approaches are common-place; eye-in-hand and eye-to-hand. The former has the camera located on the end-effector, creating a dynamic measurement system. In this case, the kinematic relation between object and camera is measured directly, resulting in more robust and accurate measurements. In the latter case, the camera is static and observes an object as well as the robot from a distance. This creates the extra difficulty that the error to be minimized is not necessarily in the center of the image, and the positioning accuracy depends highly on the accuracy of calibration and visual processing.

A second separation for visual control is the usage of a mono- or stereo camera system. The obvious advantage of a stereo imaging system is the possibility of accurate depth measurements, which for a mono-vision system is significantly more difficult. On the other hand, stereo imaging requires a high accuracy with respect to timing synchronization and knowledge of both camera's position, as well as a duplication of the processing effort. Another camera system worth mentioning is the omni-directional camera, in which a camera covers a 360° field of view. For all mentioned visual control architectures, similar kinematic relations can be derived that describe the motion of objects in the field of view with respect to the motion of the camera. As such, visual control of these systems is, to a certain extend, similar.

Finally, one last important differentiation in visual control systems, is the implementation of the visual control algorithm. The classical method is known as indirect visual servoing, which means that the (slow) visual loop acts as reference for motion control and merely designs the path to be tracked. A local (fast) joint control loop is necessary to achieve appropriate performance and guarantees stability of the complete system. For direct visual servoing, the local joint loop is directly controlled by visual measurements. This implies that the visual control loop should be executed at a sufficiently high rate in order to reduce delay and achieve the required control performance.

2.3 Hybrid, Partitioned and Switching Approaches

The combination of both image-based and position-based visual servoing into one visual control method offers the ability to use the advantages of both approaches. Existing methods can be classified in either hybrid, partitioned or switched visual servoing.

As one of the earliest works in hybrid visual servoing, Deguchi [38] proposes a partitioned approach for IBVS that separates translation and rotation to keep the object in the field of view. A method that is perhaps most well-known as hybrid visual control structure is the work presented by Malis et al. [99] and proposes a method called 2-1/2D visual servoing. The method partitions translational and rotational control by letting IBVS control translational motion and PBVS control rotational motion. The stability of such 'model-free' approaches is discussed in [98]. Another approach is the method developed in [33]. This partitioned approach decouples the rotation around and translation along the z-axis from all other DOFs. This is specifically developed to avoid problems related to a pure rotation around the optical axis [22]. This problem is also met by Kyrki et al. and presented in [86], where a task function is defined in such a way that all translations and the rotation of the optical axis are defined from PBVS and the remaining rotations from IBVS. Although a very suitable method for keeping the object in the field of view, one drawback is that the two IBVS rotational degrees of freedom are controlled in image space and thus do not take Cartesian space into account.

A method that exploits the homography between two camera frames in order to improve visual servoing and its stability is presented in [9]. The homography (a transformation that maps points from one 3D plane to another 3D plane) can be decomposed into a rotational and translational part without requiring the model of the target object. This decomposition then allows for a simple control law, based on direct measurements. Since this is, in essense, an IBVS approach, only local asymptotic stability can be proven. A recent result by Ha et al. [54] guarantees robust global stability under the field of view constraint by introducing a 3D visible set for PBVS which plays the role of the 2D visible set for IBVS. Analysis claims convergence to any desired pose under the field of view constraint, regardless of large camera displacement and large uncertainties in intrinsic and extrinsic parameters.

Aforementioned work executes visual control by combining or partitioning the visual control structure. Closely related is the methodology of switching control, which encompasses the idea that at each iteration it should be decided which controller should be used, based on some performance criteria. Initial work can be found by Hashimoto et al. [58] which proposes a method that enlarges the stable region of visual servoing by using switching control and relay images that interpolate initial and reference image features. Other methods allow switching of the controller based on the state of the system [47, 48] or based on measurements in image space [27] and both employ switching between IBVS and PBVS.

In general, all aforementioned approaches come with their advantages and disadvantages. One common consideration that has to be made is if the control should be executed in either image space or Cartesian space. This choice thus decides the response of the system, where properties such as stability and bounds on error can be analysed beforehand. Furthermore, another property that has to be taken into account is the type of motion planning. Due to the application of a task function, usually motion is planned on a path planning level. Taking kinematic constraints into account by designing a trajectory online is not considered in the traditional visual servoing approaches.

2.4 High-Speed Visual Servoing

Initial developments in using parallel processors for vision dates back to the 1980's. Earliest research on these integrated vision sensors are e.g. the '1ms visual feedback system'-project at the University of Tokyo [67], the 'silicon retina'-project at Caltech [101], the 'vision chip'-project at MIT [155] and the 'high-speed range finder'-project at CMU [52]. The reasoning behind these developments is that visual processing on conventional systems is too slow to achieve sufficient performance. By connecting the vision sensor directly to a highly parallel processing chip the delay between image acquisition and processing will decrease significantly and the processing will speed up significantly.

Following these developments, an initial controlled application was proposed in 1996 by Ishii et al. [66], which presents a tracking algorithm for a 2-DOF pan-tilt platform, controlled at 1 [kHz]. The massively parallel processing vision chip (SPE-256) has 256 processing elements and thus represents a 16×16 [px] image. Developed at the Ishikawa-Komuro Laboratory [105], this vision platform has been improved over the years (e.g., 128×128 [px] [105] and 320×240 [px] [80] with processing at 1 [kHz]) and is still a state-of-the-art. A historical overview of these developments and work by other research groups is documented by Shingo Kagami in [69].

Other applications that employ this sensor can be found in the work of Senoo et al. [129] in which two 2-DOF pan-tilt units forward visual information to a larger robotic manipulator that performs a batting task, or the work of Imai et al. [65] that employs the vision system for dynamic active catching.

Visual sensing in medical applications requires a high robustness (i.e., no feature loss, fast feedback) and thus a high frame rate as is shown by Ginhoux et al. in [50]. This work presents first experimental results in tracking of the human beating heart in robotic assisted surgery. The control objective is to achieve tracking of the robotic instrument with two visual measurements, which is achieved by model predictive control with a visual update rate of 500 [Hz] (image size: 256×256 [px]).

In [51], the wing kinematics of the tethered fruit fly is analysed in real-time. As this requires an extremely high speed vision system, a camera with a dynamic region of interest (ROI) is used to achieve a visual update rate of 6250 [Hz]. In order to reach this rate the size of the image is downscaled to 3600 pixels to not exceed the bandwidth of data communication.

The obvious observation that can be made from these is the balance between a high frame rate and sufficient image detail (i.e., image size). A dedicated parallel processor will definitely facilitate a high frame rate, however, this comes with the added difficulty of programming visual processing algorithms in a straight-forward way. Moreover, at such rates different effects come in to play. For instance, lighting can become an issue due to the short exposure time of the image sensor. Another example is the property of standard indoor lighting systems. As the camera is sampled higher than the 50 [Hz] alternating current frequency, typical disturbances such as flickering will occur.

2.5 Microscale Visual Servoing

A different recent advancement in visual control is in the area of micro-manipulation. Examples of technological trends in miniaturization that would benefit from visual (automated) control are for instance cell injections [161] or manipulation of micro-electro-mechanical systems (MEMS, [150]). An overview of the control issues that typically occur in micro-manipulation, which also includes visual feedback, can be found in [71].

Particularly, the visual system senses in a planar environment where the control objective can be quite diverse. Research on microscale imaging and positioning systems is performed for example by Ogawa et al. which proposes a visual control system for tracking and directing motile cells using a high-speed tracking system [107]. Vikramaditya et al. [151] present a visual guidance technique for automated microassembly of hybrid MEMS. The visual control loop is closed at 30 [Hz] with 5 feature templates of size 16×16 [px] and processing on multiple DSP's. Focussing more on the performance of visual servo techniques in microsystem applications, Bilen et al. [12] present an experimental comparison of conventional image based visual servoing (calibrated vs. uncalibrated). Visual processing is limited (processing platform undefined) which results in a visual control rate of 33 [Hz].

A direct visual servoing scheme for automatic nanoscale positioning is presented in [141]. High positioning accuracies are obtained, however, image sensing is executed at a fairly slow rate, i.e., 25 [Hz]. Real-time visual tracking of 3-DOF and 6-DOF motion with near-nanometer precision is presented in [78] and [77] respectively. Again here, a conventional processing platform (i.e., PC) is used for visual processing and thus the visual frame rate is limited to 25 [Hz].

Despite the difference in scale of sensing and motion, similar properties hold as with the traditional visual control approaches. Therefore, the novelty in these methods does not lay in the visual or control domain, but merely in the scale of sensing.

2.6 Path and Trajectory Planning

Basic design of paths and trajectories for robot motion control is one of the earliest fields of research when considering robotic technology. Traditional approaches that are now accepted as standard implementation can be found in many well-known textbooks. See for example [87] and [134]. It is well known that path planning [88] and trajectory planning [11] are two different topics. The former considers only the geometry of positioning, while the latter considers time and can thus include constraints on for instance velocity and acceleration. This difference is of importance, as commonly replanning of motion (e.g., obstacle avoidance) is designed on the path planning level and motion is designed and constrained separately by motor controllers.

General Path Planning

Path planning as a general research field has many application areas, ranging from robotics, computer animation to even molecular biology. For its purpose in robotic motion and manipulation, many books are devoted to the general problem of planning a path in an unknown or uncertain environment, see e.g., [87] or [88]. In these, solutions are usually divided into two different categories of approaches, commonly known as either combinatorial planning or sampling-based planning. Combinatorial planning is aimed at completely capturing all information needed to perform planning, whereas sampling-based planning merely searches for a solution to solve the planning problem. As such, combinatorial planning is a complete technique, meaning that if a solution exists it will be found in finite time. Despite this, in practise, the sampling-based technique is far more popular due to its efficiency in execution.

Examples of sampling-based approaches are for instance Rapidly-exploring Random Tree (RRT, [29]) or the Probabilistic RoadMap method (PRM, [29]). In particular, PRM is a planner that can compute collision-free paths for robots in a static environment by sampling the configuration space of the robot, testing if these are collision-free and connecting these to existing configurations. RRT is a similar technique, however, based on the construction of a tree in such a way that any sample in the search space is added by connecting it to the closest sample already in the tree. RRT-based algorithms were first developed for non-holonomic and kinodynamic planning problems, and are therefore a good example of a combination of path planning with kinematic or dynamic constraints.

One example of a combinatorial technique, which is based on cell decomposition is for instance SCOUT (Simple Calculation of Useful Tracks, [149]). This is a multi-resolution approach that uses cell decomposition for path planning and bubble hierarchies for collision detection. The configuration space is locally tessellated by binary division and bubbles are used to approximate the geometry of the robot in the work space.

These methods are all aimed to plan a path which is complete (i.e., if there is a solution it will be found in finite time). Methods that consider the robot as a point in a potential field combine attraction towards a goal and repulsion away from obstacles as main objective. These methods have a low computation time, however, they can get stuck in local minima [87].

Path Planning in Visual Servoing

Planning a path for visual controlled robotics offers additional possibilities compared to traditional path planning. More specifically, the presence of a camera adds a space in which a path (or trajectory) can be planned. Compared to the planning of a path (or trajectory) in joint- or Cartesian space, image paths are fairly limited with respect to constraints. Conditions such as velocity and acceleration are largely meaningless in image-space due to the difficulty in obtaining accurate measurements. Popular methods therefore employ the planning in image space only as guidance, for either the motion towards a target or the avoidance of obstacles or both at the same time. One example is the work of Chesi et al. [28]. In this, visibility and workspace constraints are considered while minimizing a cost function such as spanned image area and trajectory

length.

The method of Kermorgant et al. [74] considers PBVS as core visual servo control approach and adds 2D visual information only when necessary. That is, when an object is about to leave the field of view, the control law is changed. This 2D information is weighted by the distance towards the image border. Similar to this is the work presented in [55]. In this, a probabilistic integration of 2D and 3D cues is proposed, where a weighted sum of IBVS and PBVS lets visual servoing start with the position-based approach and end with the image-based approach. It has to be noted that, besides path planning, these works could also be listed as hybrid/partitioned visual servoing.

A different methodology, based on model predictive control (MPC) is proposed by Chan et al. [21] and incorporates the field of view constraint into a nonlinear MPC structure. Together with dynamic collision checking, a constraint-aware control law is obtained that handles joint, visual and collision constraints.

The concept of next best view (NBV) planning is a fairly new topic for visual controlled motion [128]. NBV planning considers the determination of a next view in free-space with respect to an object, and, as such, does not consider limitations of the manipulator or constraints on the (image) path or trajectory. For instance, the primary purpose of an NBV algorithm is to plan, as efficiently as possible, the path for building highly accurate 3D models from images [122].

Trajectory Planning

One important distinction that has recently been made in trajectory design is the concept of online trajectory generation. Whereas 'traditional' trajectory generation designs a motion trajectory offline, online trajectory generation designs a motion trajectory at runtime and as such, can incorporate changes of this trajectory online. This fairly new concept is particularly of interest for sensor-based motion systems, as an integration of constraints into planning becomes possible.

Traditional trajectory generation is commonly based on the assumption that initial and final constraints (e.g., velocity and acceleration for a 5^{th} order polynomial) are equal to zero. The work of Ahn et al. [2] proposes a method, denoted arbitrary states polynomial-like trajectory (ASPOT), which designs a trajectory with arbitrary initial and final constraints. The method generates the trajectories online, however, constraints are not taken into account.

Research presented by Thompson et al. [142] describes trajectory generation which explicitly considers the presence of obstacles. The method entails adding a fourth order term to a cubic polynomial and a cost function to the state equations. Solving for the parameters of the polynomial given initial and final constraints then generates polynomial trajectories which minimise the cost function. The main limitation of this method is the fact that the minimization method is a form of gradient descent and as such is subject to local minima.

The work of Namiki et al. [106] presents an online trajectory generator for catching a flying ball in mid-air. A 5^{th} order polynomial is used to describe all possible target trajectories in the neighbourhood of the catching point. The parameters of the trajectory are optimized depending on the dynamics and the

kinematics of the manipulator and the object. A final trajectory is then generated so that the end-effector can catch the target at one point, and a match between the position, velocity, and acceleration of the target and the end-effector is satisfied. The obvious drawback of this method is the fact that the number of potential trajectories to be evaluated is limited by the computational resources of the processor. Therefore, the processing system consists of many floating-point DSP modules.

Motion planning specifically designed for obstacle avoidance is developed by Shiller et al. in [132]. The avoidance of static obstacles is generated for one obstacle at a time. The robot is treated as a point-mass and bang-bang trajectories (i.e., constant acceleration) are generated online. This limits the method to be continuous only up to the velocity level. Another drawback is that only simulation results are shown for a planar multi-obstacle avoidance task.

A different approach towards planning trajectories considers filtering techniques which alter an infeasible trajectory into a feasible trajectory [49]. In this the filter generates output signals which are continuous up to and including the second time derivative. Simultaneously, bounds can be set on first, second, and third time derivatives. A filtering technique that is designed in discrete time and guarantees constrained (asymmetric) bounds of velocity, acceleration and jerk can be found in [53]. However, the method generates trajectories with continuity only up to the second time derivative. Other examples of smooth, online trajectories are for instance [10], [90] and [57].

A complete framework for the generation of motion trajectories online is presented in the work of Kröger [83, 82]. Particularly motion systems subject to unforeseen events benefit from this approach by being able to directly react to events and switch between different control methods or domains. As such, this is a hybrid switched systems approach to robotic manipulation and is motivated to generate motion with arbitrary initial constraints. In experiments, however, a trajectory is presented in which the final constraints can be specified up to and including velocity (i.e., 3^{rd} order) and the acceleration is set to zero.

2.7 Closely Related Work

The work of Geraldo Silveira, denoted direct visual servoing [135], presents a technique that uses only non-metric visual information to guide visual servo control. A reference image is used to obtain projective parameters via a photo-geometric registration method. The method is highly accurate and robust to illumination changes, even in color images. As such, it still consists of the double control loop structure typically found in traditional approaches (see Fig. 2.1). Moreover, the considered path planning is limited to a linear (translation) and geodesic (rotation) trajectory, where online updates and motion constraints are not considered.

As described in the previous section, Torsten Kröger proposed a complete framework for online trajectory generation [82]. At runtime the trajectory generator designs the motion of the next state based on the current state and events. Although a complete framework is presented, experiments are only shown with geometrically continuous trajectories up to the 2^{nd} degree.

The approach presented by Ville Kyrki in [86] describes a hybrid visual servoing method that uses image information to control two rotational degrees of freedom (pan and tilt) and Cartesian errors for the remaining degrees of freedom. This field of view constraint combines (i.e., partitions) IBVS and PBVS in one control law, and does not consider a direct or single-loop approach.

A final work that is closely related, is the work of Namiki et al. [106] which presents an online trajectory generator for catching a flying ball in mid-air. As described in the previous section, the method is based on polynomial trajectories and an online optimization method to select a final trajectory that matches the motion of the ball and the end-effector. Due to this optimization method, the method's main limitation is that the number of trajectories to be evaluated is limited by the computational resources of the processor.

2.8 Summary

This chapter presented a global overview of existing work in visual servoing. A historical review discusses the earliest developments in visual control which started in the early 1970's. Following, the main traditional approaches of visual servoing (i.e., image-based and position-based visual control) are discussed, which includes the current analyses on both approaches as well as methods known as hybrid visual servoing which partition or switch between both.

Two novel and fairly recent advances in visual control are described in a separate section. First, high-speed visual servoing is discussed which is a particularly novel method due to the ever growing availability of computation power. One example in particular is high-lighted; the high-speed vision chip of the Ishikawa-Komuro Laboratory. This sensor is especially interesting due to the direct connection between sensor and processor and the application with respect to a 2-DOF pan-tilt unit. Second, the micro-domain as new application area for visual control is described as another topic of interest. Several examples are given and discussed accordingly. It is concluded, however, that this microscale domain does not present any novel developments, but merely utilizes a smaller scale for sensing.

A topic closely related to visual control is in the area of path and trajectory planning. Although not necessarily combined with vision, the developments in this are highly related to the planning of motion in visual control. The difference in path planning and trajectory planning is explained and several examples with respect to the online adaptation of constrained motion is presented.

A final section is devoted to research that is closely related to the work presented in this thesis. These are not necessarily related to each other but cover separate topics as developed and treated in this work.

This literature review suggests that, concerning visual motion control, a proper integration of vision with constrained motion is lacking. In particular, this involves the incorporation of a direct visual sensing scheme with a trajectory that can be constrained (spatial and kinematic) online. These developments should not be limited by the update rate of the camera or the performance of the processor (i.e., due to visual processing). As such, the combination of these topics (i.e., visual processing, kinematic robot control, trajectory generation), is the subject of interest of this thesis.

Part I

Modelling and Planning of Robotic Manipulators

CHAPTER 3

Modelling and Planning of Robotic Manipulators

Abstract. This chapter considers the basics in dynamic and kinematic modelling of robotic manipulators. The analysis is presented as a general introduction towards dynamic and kinematic robot control, where kinematic redundancy and the use of this for the self-motion of a manipulator is treated subsequently. The input for these control structures are introduced on the basics of path and trajectory generation. These developments form the basis of manipulator control for all following chapters of this thesis.

3.1 Introduction

From a system point of view, the dynamic behaviour of a multibody system (a robotic manipulator) is described on the level of accelerations (i.e., in general, second-order nonlinear models). Inputs (forces/torques) acting on such system induce motion as can be described by equations that result from Newton's second law. These equations can be written for general free rigid bodies as well as for constrained robotic systems.

However, in many cases, the system to be controlled is of such structure that a decomposition can be made between a second-order system and a first-order system. The control of a first-order system considers the kinematics of the multibody system and is designed by considering a kinematic model. The complete control structure is then realized by assuming velocity inputs for the kinematic controller and ignoring the dynamics of the multibody system. The execution of these velocity reference inputs is then realized by a separate controller with a dynamic output. The performance of this control structure is physically feasible due to the assumption that movement is usually slow and dynamic effects do not have a big influence. This approach, commonly referred to as kinematic control, motivates the derivation of kinematic relationships between base and end-effector (in pure kinematic control) or end-effector and object (in vision-based kinematic control).

3.2 Modelling of Dynamics

The dynamic model of a serial link robot formulated in a Euler-Lagrange representation is given as [134]:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_f(\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}(t), \tag{3.1}$$

where $\mathbf{q} \in \mathbb{R}^n$, $\dot{\mathbf{q}} \in \mathbb{R}^n$ and $\ddot{\mathbf{q}} \in \mathbb{R}^n$ is the vector of joint coordinates, velocities and accelerations respectively. $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n \times n}$ is the symmetric, posi-

tive definite inertia matrix and $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n \times n}$ is the matrix with centripetal and Coriolis terms. Finally, $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^n$ is the vector with gravitational terms, $\boldsymbol{\tau}(t) \in \mathbb{R}^n$ is the vector of torque inputs and $\mathbf{F}_f(\dot{\mathbf{q}}) \in \mathbb{R}^n$ denotes the vector of friction terms. A classical model of friction combines a viscous friction term and a Coulomb friction term as

$$\mathbf{F}_f(\dot{\mathbf{q}}) = \mathbf{F}_v \dot{\mathbf{q}} + \mathbf{F}_c \mathbf{sgn}(\dot{\mathbf{q}}), \tag{3.2}$$

where $\mathbf{F}_v$ denotes the $(n \times n)$ diagonal matrix of viscous friction coefficients and $\mathbf{F}_c$ denotes the $(n \times n)$ diagonal matrix of Coulomb friction coefficients. Furthermore, sgn represents the signum operator which is incorporated as

$$\mathbf{sgn}(\dot{\mathbf{q}}) = \begin{bmatrix} \mathrm{sgn}(\dot{q}_1) \\ \mathrm{sgn}(\dot{q}_2) \\ \vdots \\ \mathrm{sgn}(\dot{q}_n) \end{bmatrix}. \tag{3.3}$$

Ignoring friction, a classical approach is to derive an augmented PD control law with gravity compensation as

$$\boldsymbol{\tau} = \mathbf{K}_p \mathbf{e}_q + \mathbf{K}_d \dot{\mathbf{e}}_q + \mathbf{g}(\mathbf{q}), \tag{3.4}$$

where $\mathbf{e}_q = \mathbf{q}_d - \mathbf{q} \in \mathbb{R}^n$ is the error in joint space, $\dot{\mathbf{e}}_q = \dot{\mathbf{q}}_d - \dot{\mathbf{q}} \in \mathbb{R}^n$ is the velocity error in joint space, $\mathbf{q}_d$ and $\dot{\mathbf{q}}_d$ define the desired position and velocity reference respectively, and $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of torque inputs. Furthermore, $\mathbf{K}_p, \mathbf{K}_d \in \mathbb{R}^{n \times n}$ are symmetric positive definite gain matrices. This leads to the 'ideal' (i.e., no friction) closed loop system:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{K}_p \mathbf{e}_q + \mathbf{K}_d \dot{\mathbf{e}}_q + \mathbf{g}(\mathbf{q}), \tag{3.5}$$

for which global asymptotic stability can be proven via Lyapunov's direct method [73].

This example of an augmented PD control law with gravity compensation does not include friction. The fact that friction effects in mechanical systems depend on multiple factors (e.g., material, temperature, velocity), makes it a difficult phenomena to model. Even though many different models of friction and the estimation of these models exist in literature (see e.g., [6], [13], [145] and [72]), a rigorous analysis of these is beyond the scope of this introduction. Control laws which include friction as well as the stability analysis of such control laws has been a topic in many past researches, see for instance [144], [165], [97] and the references therein for a brief overview.

From a practical point of view, however, friction can not always be so easily ignored. As will be shown in Chapter 7, in order to achieve a higher performance in motion control, a friction compensation scheme has to be included.

3.3 Modelling of Kinematics

One common method for the modelling of robot kinematics is by adopting the Denavit-Hartenberg convention. This representation assigns coordinate frames to the robotic joints and defines four parameters according to the geometric relationship between coordinate frames.

A homogeneous transformation between two frames is nothing more than the compact matrix representation of rigid motion. Consider a serial robotic manipulator with n joints, where joint i has assigned the coordinate frame $i-1$. The rigid motion (position and orientation), of coordinate frame i with respect to coordinate frame $i-1$ of this manipulator can now be represented by the homogeneous transformation matrix:

$$\mathbf{T}_i^{i-1}(q_i) = \begin{bmatrix} \mathbf{R}_i^{i-1}(q_i) & \mathbf{p}_i^{i-1}(q_i) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, \tag{3.6}$$

where $\mathbf{R}_i^{i-1}(q_i) \in \mathbb{SO}(3)$ represents the skew-symmetric rotation matrix, with $\mathbb{SO}(3)$ the special orthogonal group and where $\mathbf{p}_i^{i-1}(q_i) \in \mathbb{R}^3$ represents the position vector, which are both dependent on joint displacement q_i (see Fig. 3.1).

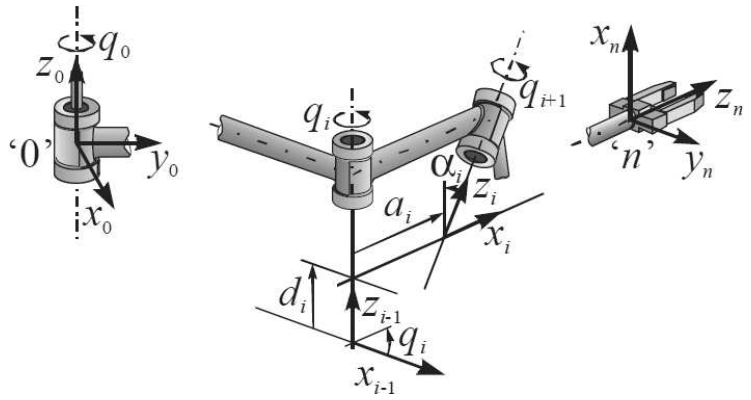


Figure 3.1: Coordinate transformations and Denavit-Hartenberg kinematic parameters in open kinematic chains [134].

The forward kinematics (FK) of a robotic manipulator defines the relationship between individual joints of the manipulator $\mathbf{q}$ and the pose (position and orientation) of the end-effector $\mathbf{x}$ by the kinematic map as

$$\mathbf{x} = \mathbf{k}(\mathbf{q}). \tag{3.7}$$

The joint variables $\mathbf{q}$ (generalized coordinates) are the angular and linear displacements of revolute and prismatic joints respectively. The FK can now be derived as a product of homogeneous transformations of subsequent coordinate frames:

$$\mathbf{T}_n^0(\mathbf{q}) = \mathbf{T}_1^0(\mathbf{q})\mathbf{T}_2^1(\mathbf{q})\dots\mathbf{T}_n^{n-1}(\mathbf{q}). \tag{3.8}$$

For the control of a manipulator, it is necessary to compute the inverse kinematics (IK), which consist of determining the manipulator's joint variables given the pose of the end-effector. Since not always a closed-form solution exists (i.e., there can be multiple or infinite solutions), a common approach is to solve this problem numerically [134]. This can be done by considering the differential kinematics which give the relationship between the joint velocities and the end-effector velocities. This will be discussed in the following section.

3.3.1 Kinematic Control

The purpose of the differential kinematics is to define a relationship between the joint velocities $\dot{\mathbf{q}}$ and the velocities (linear and angular) of the end-effector $\dot{\mathbf{x}}$ as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}}_n^0 \\ \dot{\mathbf{o}}_n^0 \end{bmatrix} = \begin{bmatrix} \mathbf{J}_p(\mathbf{q}) \\ \mathbf{J}_o(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}, \tag{3.9}$$

where the subscript p denotes position and o denotes orientation and where $\mathbf{J}(\mathbf{q})$ is the $(6 \times n)$ Jacobian matrix. The computation of this matrix commonly follows a geometric procedure, where the contributions of each joint in terms of linear and angular velocity of the end-effector are determined.

These geometric Jacobians $\mathbf{J}_p(q)$ and $\mathbf{J}_o(q)$ are based on the forward kinematics and can be derived in a systematic way as follows:

$$\mathbf{J}_p = \left[J_{p,1} \dots J_{p,n}\right], \quad J_{p,i} \in \mathbb{R}^3 \text{ and} \tag{3.10}$$

$$\mathbf{J}_o = \left[J_{o,1} \dots J_{o,n}\right], \quad J_{o,i} \in \mathbb{R}^3, \text{ where} \tag{3.11}$$

$$J_{p,i} = \begin{cases} \mathbf{z}_{i-1}^0 \times (\mathbf{p}_n^0 - \mathbf{p}_{i-1}^0) & \text{for revolute joint } i \\ \mathbf{z}_{i-1}^0 & \text{for prismatic joint } i \end{cases} \text{ and} \tag{3.12}$$

$$J_{o,i} = \begin{cases} \mathbf{z}_{i-1}^0 & \text{for revolute joint } i \\ \mathbf{0} & \text{for prismatic joint } i \end{cases} \tag{3.13}$$

In this $\mathbf{z}_{i-1}^0$, is the third column of the rotation matrix $\mathbf{R}_i^{i-1}(q_i)$.

If the pose of the end-effector can be specified in terms of a minimum number of parameters in the task space, it is also possible to determine the Jacobian matrix by differentiation of the forward kinematics:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}}_n^0 \\ \dot{\mathbf{o}}_n^0 \end{bmatrix} = \mathbf{J}_a(\mathbf{q})\dot{\mathbf{q}}, \tag{3.14}$$

where $\mathbf{J}_a(\mathbf{q}) = \partial \mathbf{k} / \partial \mathbf{q}$ and where $\mathbf{J}_a$ is termed analytic Jacobian. In general the geometric Jacobian is adopted when physical quantities are of interest, while the analytic Jacobian is adopted when task space quantities are of interest [134].

For control purposes the inverse relationship between joint and end-effector velocities is considered. Let r be the number of Cartesian space variables necessary for a task, m the number of Cartesian space variables and n the number of degrees of freedom of the manipulator. For $n = r$ (i.e., non-redundancy) the inverse kinematic solution can be found with the differential kinematics:

$$\dot{\mathbf{q}} = \mathbf{J}_a^{-1}(\mathbf{q})\dot{\mathbf{x}}. \tag{3.15}$$

However, because of drift due to the numerical implementation (i.e., numerical integration), the Cartesian space error is taken into account as $\dot{\mathbf{x}} = \dot{\mathbf{x}}_d + \mathbf{K}_k\mathbf{e}$. This leads to the following system:

$$\dot{\mathbf{q}} = \mathbf{J}_a^{-1}(\mathbf{q})[\dot{\mathbf{x}}_d + \mathbf{K}_k\mathbf{e}], \tag{3.16}$$

in which $\mathbf{e} = \mathbf{x}_d - \mathbf{x} \in \mathbb{R}^n$ is the error in task space and $\dot{\mathbf{x}}_d$ defines the desired velocity reference. The error dynamics can then be formulated as

$$\dot{\mathbf{e}} + \mathbf{K}_k\mathbf{e} = \mathbf{0}. \tag{3.17}$$

If $\mathbf{K}_k \in \mathbb{R}^{n\times n}$ is chosen as a positive definite (diagonal) gain matrix, the system is asymptotically stable [134]. The Euler integration method then allows for traditional joint position PD control:

$$\mathbf{q}(t_{k+1}) = \mathbf{q}(t_k) + \mathbf{J}_a^{-1}(\mathbf{q})[\dot{\mathbf{x}}_d(t_k) + \mathbf{K}_k\mathbf{e}(t_k)]\Delta t, \tag{3.18}$$

and

$$\boldsymbol{\tau} = \mathbf{K}_p\mathbf{e}_q + \mathbf{K}_d\dot{\mathbf{e}}_q + \mathbf{g}(\mathbf{q}), \tag{3.19}$$

where Δt is the integration interval, $\mathbf{e}_q = \mathbf{q}_d - \mathbf{q} \in \mathbb{R}^n$ and $\dot{\mathbf{e}}_q = \dot{\mathbf{q}}_d - \dot{\mathbf{q}} \in \mathbb{R}^n$ are the joint position and velocity error respectively, $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^n$ is the term to compensate for gravity effects and $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of torque inputs. As long as $\mathbf{K}_p$, $\mathbf{K}_d \in \mathbb{R}^{n\times n}$ are positive definite, this controller guarantees global asymptotic stability [134].

As the kinematic relation given by (3.15) directly allows for Cartesian velocity inputs, the following describes a joint velocity PI controller.

$$\boldsymbol{\tau} = \mathbf{K}_p\dot{\mathbf{e}}_q + \mathbf{K}_i\boldsymbol{\xi}, \tag{3.20}$$

$$\dot{\boldsymbol{\xi}} = \dot{\mathbf{e}}_q, \tag{3.21}$$

where $\mathbf{K}_p$, $\mathbf{K}_i \in \mathbb{R}^{n\times n}$ are positive definite matrices and $\boldsymbol{\xi} = \int_0^t \dot{\mathbf{e}}_q(\sigma)d\sigma$.

The desired velocity can be obtained by either (3.16) or by a separate procedure (e.g., vision sensor). The PI controller in (3.20) including (3.16) can be proven exponentially stable and uniformly, ultimately bounded [18].

3.3.2 Redundancy Formulation

For a kinematically redundant manipulator, a nonzero null space exists due to more degrees of freedom than necessary for a particular task in the Cartesian space ($r < n$). A common method of including the null space in a solution is the gradient projection formulation [134]:

$$\dot{\mathbf{q}} = \mathbf{J}_a^{\dagger}(\mathbf{q})\dot{\mathbf{x}} + (\mathbf{I} - \mathbf{J}_a^{\dagger}(\mathbf{q})\mathbf{J}_a(\mathbf{q}))\dot{\mathbf{q}}_0, \tag{3.22}$$

where $\dot{\mathbf{q}}_0$ is an arbitrary joint velocity vector, $\mathbf{J}_a^{\dagger} = \mathbf{J}_a^{\mathrm{T}}(\mathbf{J}_a\mathbf{J}_a^{\mathrm{T}})^{-1} \in \mathbb{R}^{n\times m}$ is the Moore-Penrose generalized inverse of $\mathbf{J}_a$ and $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix. The first term is the particular solution to the inverse problem ($\mathbf{J}_a\dot{\mathbf{q}} = \dot{\mathbf{x}}$), and the second term represents the homogeneous solution to the problem ($\mathbf{J}_a\dot{\mathbf{q}} = \mathbf{0}$). Note that, for the sake of clarity, $\mathbf{J}_a(\mathbf{q})$ is written as $\mathbf{J}_a$.
Thus, in the general inverse solution the matrix $(\mathbf{I} - \mathbf{J}_a^{\dagger}\mathbf{J}_a)\dot{\mathbf{q}}_0$ is a projector of the joint vector $\dot{\mathbf{q}}_0$ onto the null space of $\mathbf{J}_a$: $\mathcal{N}(\mathbf{J}_a)$. The projection operator $(\mathbf{I} - \mathbf{J}_a^{\dagger}\mathbf{J}_a)\dot{\mathbf{q}}_0$ selects the components of $\dot{\mathbf{q}}_0$ in the null space of the mapping $\mathbf{J}_a$, meaning that $\dot{\mathbf{q}}_0$ produces only joint self-motion of the structure but not task-space motion.

A problem that arises when considering redundancy in robotics is the fact that the inverse of the Jacobian, $\mathbf{J}_a^{-1}$, is not uniquely defined [16, 41, 127]. As presented in (3.22), the pseudo-inverse of the Jacobian, $\mathbf{J}_a^{\dagger}$, is a commonly used alternative. However, by definition, this pseudo-inverse has no proper geometrical meaning. That is, in physical units, this pseudo-inverse is inconsistent and should not be computed. This inconsistency originates from the computation of $\mathbf{J}_a\mathbf{J}_a^{\mathrm{T}}$ which involves an addition of units which are not similar. A

proper alternative to the inconsistent pseudo-inverse is the use of a weighted pseudo-inverse:

$$\mathbf{J}_a^{\#} = \mathbf{W}^{-1}\mathbf{J}_a^{\mathrm{T}}(\mathbf{J}_a\mathbf{W}^{-1}\mathbf{J}_a^{\mathrm{T}})^{-1}, \tag{3.23}$$

where $\mathbf{W}$ is an appropriate, positive definite weighting matrix.

One of the most widely adopted approaches to solve redundancy is by optimizing a scalar cost function $\mathbf{m}(\mathbf{q})$ using the Gradient Projection Method (GPM), i.e., choosing $\dot{\mathbf{q}}_0 = k_0\nabla\mathbf{m} = k_0(\partial\mathbf{m}/\partial\mathbf{q})^{\mathrm{T}}$. This then represents a smooth function for a secondary task in terms of some performance criteria (e.g., distance towards mechanical joint limits, distance in Cartesian space). In this, k_0 is a scalar which controls the gain of the second task and can be defined as [91]:

$$k_0 = \frac{|\mathbf{J}_a^{\#}\dot{\mathbf{x}}|}{|(\mathbf{I} - \mathbf{J}_a^{\#}\mathbf{J}_a)\nabla\mathbf{m}|}. \tag{3.24}$$

It is designed to avoid a large difference between the two terms defined in (3.22). Notice that any differentiable cost function may be used as long as the function can be reduced to an expression in terms of the joint variables only.

The redundancy task that tries to avoid singular configurations of a robotic manipulator by maximizing the manipulability index $M(\mathbf{q})$ is defined as [160]:

$$M(\mathbf{q}) = \sqrt{\det(\mathbf{J}_a(\mathbf{q})\mathbf{J}_a^{\mathrm{T}}(\mathbf{q}))}. \tag{3.25}$$

The gradient of M is found as

$$\nabla\mathbf{m} = M(\mathbf{q})\,\mathrm{tr}\left\{\frac{\partial\mathbf{J}_a}{\partial\mathbf{q}}\mathbf{J}_a^{\dagger}\right\}, \tag{3.26}$$

where $\mathrm{tr}\{.\}$ represents the trace operator.

Again here it is identified, that if redundancy is considered, the computation of $\mathbf{J}_a\mathbf{J}_a^{\mathrm{T}}$ is inconsistent [137]. Due to the different physical units of the individual elements in the Jacobian matrix, the performance index (3.25) does not give a proper index for evaluating manipulability.

Instead, similar to the weighted pseudo-inverse, a symmetric positive weighting matrix $\mathbf{W}$ can be incorporated to obtain a meaningful manipulability index as

$$M(\mathbf{q}) = \sqrt{\det(\mathbf{J}_a(\mathbf{q})\mathbf{W}^{-1}\mathbf{J}_a^{\mathrm{T}}(\mathbf{q}))}. \tag{3.27}$$

Alternative to an index which evaluates manipulability, the following derivation presents indices that evaluate a distance in Cartesian space. This includes the distance between a point on the manipulator and some geometric entity, i.e., a point, a line and a plane, which should be either minimized or maximized.

Point Distance Index

A distance index is perceived as a difference in translation. As such, the manipulator's self-motion (i.e., motion of the manipulator while keeping the end-effector fixed at a certain pose) is also controlled as a translation and orientation control is not regarded.

The vector $\mathbf{x}_{jo}$ originating from each joint (represented as point $\mathbf{x}_j$) towards a point on an obstacle $\mathbf{x}_o$ is represented as

$$\mathbf{x}_{jo} = \mathbf{x}_o - \mathbf{x}_j. \tag{3.28}$$

The distance between these two points is found as

$$d_{p,p} = |\mathbf{x}_o - \mathbf{x}_j|. \tag{3.29}$$

The vector $\mathbf{x}_{jo}$ represents the direction in which the self-motion of the manipulator should move and thus can be used as

$$\nabla \mathbf{m} = \sum_{i=1}^{n} \mathbf{J}_{a,i}^{\#} \mathbf{x}_{jo,i}. \tag{3.30}$$

$$\nabla \mathbf{m} = \left[\mathbf{J}_{a,1}^{\#} \begin{bmatrix} \mathbf{x}_{jo,1} \\ \mathbf{0}_{3\times 1} \end{bmatrix} + \mathbf{J}_{a,2}^{\#} \begin{bmatrix} \mathbf{x}_{jo,2} \\ \mathbf{0}_{3\times 1} \end{bmatrix} + \cdots + \mathbf{J}_{a,n}^{\#} \begin{bmatrix} \mathbf{x}_{jo,n} \\ \mathbf{0}_{3\times 1} \end{bmatrix} \right]^T. \tag{3.31}$$

Perpendicular Distance Index

The index that should minimize the self-motion of the manipulator is defined as the summed perpendicular distance d_p between a joint q_i for $i \in \{1, \ldots, n\}$ and the line L_{be} connecting base and end-effector of the manipulator:

$$\sum_{i=0}^{n} d_{p,i} = \sum_{i=0}^{n} |\mathbf{x}_{q,i} - \mathbf{x}_{be}|. \tag{3.32}$$

This index can be directly assigned as a weight on the nullspace of each joint in order to give higher priority (i.e., weighting) on joints with a larger offset with respect to the base-end-effector centerline L_{be}. The index and its derivative is calculated as follows.

The shortest distance between base and end-effector is defined as the line in 3D space $\mathbb{R}^3$ connected by the points: $\mathbf{x}_b$ and $\mathbf{x}_e$, which represent the position of the base and end-effector of the manipulator respectively (see Fig. 3.2). In order for the manipulator's self-motion to be as small as possible, each joint should be as close to this line as possible. The point $\mathbf{x}_p$ which is projected perpendicular on L_{be} can be calculated as

$$\begin{aligned} \mathbf{x}_p &= k_p(\mathbf{x}_e - \mathbf{x}_b), \text{ where} \\ k_p &= \frac{\mathbf{x}_j \cdot \mathbf{x}_e}{|\mathbf{x}_e - \mathbf{x}_b|^2}, \end{aligned} \tag{3.33}$$

where $\cdot$ represents the dot-product between two vectors. Since $\mathbf{x}_b = \mathbf{0}$ this is simplified to

$$\begin{aligned} \mathbf{x}_p &= k_p \mathbf{x}_e, \text{ where} \\ k_p &= \frac{\mathbf{x}_j \cdot \mathbf{x}_e}{|\mathbf{x}_e|^2}, \end{aligned} \tag{3.34}$$

from which the distance $d_{p,l}$ can be determined as

$$d_{p,l} = |\mathbf{x}_p - \mathbf{x}_j|. \tag{3.35}$$

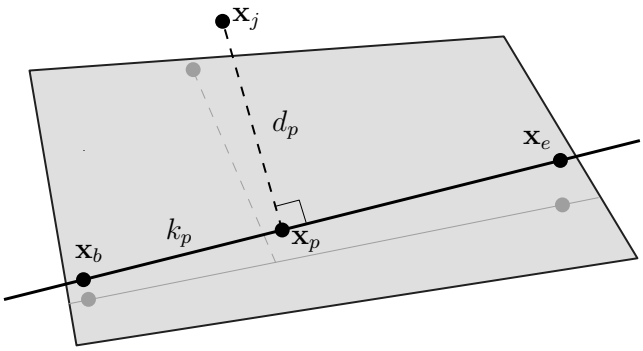


Figure 3.2: The solid line L_{be} is specified by two points which represents the position of the base of the manipulator $\mathbf{x}_b$ and the position of the end-effector $\mathbf{x}_e$. The dashed line is specified by two points which represent the position of a joint $\mathbf{x}_j$ and its perpendicular projection on L_{be} with perpendicular distance d_p.

For clarity subscripts i for each joint have been omitted.

The vector $\mathbf{x}_{jp,i}$ originating from each joint towards line l_{be} is represented as

$$\mathbf{x}_{jp,i} = \mathbf{x}_{p,i} - \mathbf{x}_{j,i}. \tag{3.36}$$

This vector represents the direction in which the self-motion of the manipulator should move and thus can be used as

$$\nabla \mathbf{m} = \sum_{i=1}^{n} \mathbf{J}_{a,i}^{\#} \mathbf{x}_{jp,i}. \tag{3.37}$$

$$\nabla \mathbf{m} = \left[\mathbf{J}_{a,1}^{\#} \begin{bmatrix} \mathbf{x}_{jp,1} \\ \mathbf{0}_{3\times 1} \end{bmatrix} + \mathbf{J}_{a,2}^{\#} \begin{bmatrix} \mathbf{x}_{jp,2} \\ \mathbf{0}_{3\times 1} \end{bmatrix} + \cdots + \mathbf{J}_{a,n}^{\#} \begin{bmatrix} \mathbf{x}_{jp,n} \\ \mathbf{0}_{3\times 1} \end{bmatrix} \right]^T. \tag{3.38}$$

Boundary Index

A boundary is chosen as an area in Cartesian space $\mathbb{R}^3$ that should be avoided. An example is, when the manipulator is mounted on a wheelchair, it should not come into contact with a user. As such, a boundary can be defined by a plane computed from the three points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ as follows (see Fig. 3.3).

First, the normal vector of the plane is determined by taking the cross product between the two vectors $\mathbf{x}_{12} = \mathbf{x}_2 - \mathbf{x}_1$ and $\mathbf{x}_{13} = \mathbf{x}_3 - \mathbf{x}_1$ as $\mathbf{n} = \mathbf{x}_{12} \times \mathbf{x}_{13}$. Since the plane passes through the origin, this is also directly the equation of the plane:

$$\mathbf{n} \cdot (\mathbf{x}_r - \mathbf{x}_1) = \mathbf{n} \cdot \mathbf{x}_r = 0, \tag{3.39}$$

where $\mathbf{x}_r = [x_p, y_p, z_p]^T$ is some point on the plane. The line starting from a point in $\mathbb{R}^3$ space $\mathbf{x}_{p,b}$ perpendicular to the plane can be parametrically represented as:

$$\mathbf{x}_b = \mathbf{x}_{p,b} + d_{p,b}\mathbf{n}, \tag{3.40}$$

where $d_{p,b}$ represents the distance along the line. To determine the crossing point $\mathbf{x}_{cp}$ of this line and the plane, $d_{p,b}$ is found by filling (3.40) into (3.39) such that:

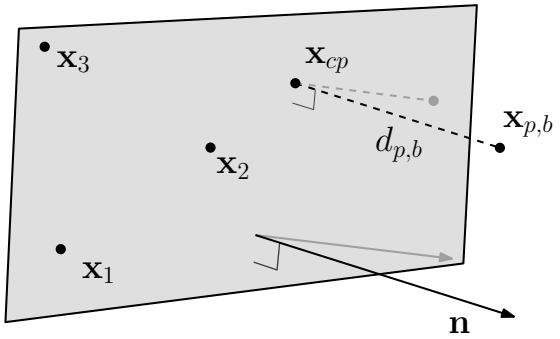


Figure 3.3: The three points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are used to form a planar boundary. The distance perpendicular towards the plane $d_{p,b}$ starting from point $\mathbf{x}_{p,b}$ (which represents a joint position) and intersecting the plane in $\mathbf{x}_{cp}$ is used as index to avoid a certain area in the workspace.

$$d_{p,b} = -\frac{\mathbf{x}_{p,b} \cdot \mathbf{n}}{\mathbf{n}^T \mathbf{n}}, \tag{3.41}$$

and filled into (3.40) to obtain the intersection point $\mathbf{x}_{cp}$.

The vector $\mathbf{x}_{bp,i}$ originating from each joint perpendicularly towards the plane is then defined as

$$\mathbf{x}_{bp,i} = \mathbf{x}_{cp,i} - \mathbf{x}_{p,b}. \tag{3.42}$$

This vector represents the negative direction in which the self-motion of the manipulator should move to avoid the plane and thus can be used as

$$-\nabla \mathbf{m} = \sum_{i=1}^{n} \mathbf{J}_{a,i}^{\#} \mathbf{x}_{bp,i}. \tag{3.43}$$

$$\nabla \mathbf{m} = \left[\mathbf{J}_{a,1}^{\#} \begin{bmatrix} \mathbf{x}_{bp,1} \\ \mathbf{0}_{3\times 1} \end{bmatrix} + \mathbf{J}_{a,2}^{\#} \begin{bmatrix} \mathbf{x}_{bp,2} \\ \mathbf{0}_{3\times 1} \end{bmatrix} + \cdots + \mathbf{J}_{a,n}^{\#} \begin{bmatrix} \mathbf{x}_{bp,n} \\ \mathbf{0}_{3\times 1} \end{bmatrix} \right]^T. \tag{3.44}$$

This index is now presented for one plane. This can be extended to include multiple planes or even a surface that acts as avoidance area.

3.4 Path Planning

In the field of motion planning, a clear difference is made between path planning and trajectory generation. While path planning only takes geometric considerations into account, a trajectory will include time and can therefore specify velocity and acceleration constraints. Furthermore, a path, ρ , is defined as a continuous, parametrized sequence of points in either the joint space, $\mathcal{J}$ or configuration space, $\mathcal{C}$, of a robot. Simply stated, the configuration space, $\mathcal{C}$, of a robot is the set of all possible configurations. This thus excludes configurations with joint values that are outside the limits of the joints, which means that the configuration space is only a subset of the joint space. Moreover, in general, this subset does not have to be a smooth manifold and may have singular points [166].

The basic motion planning problem is to find a path from an initial configuration to a final configuration without any collisions (see Fig. 3.4). This is defined in many textbooks (see e.g., [88], [29]) with the following formulations.

3.4.1 Constraints on Paths

Let $\mathcal{W}$ be the world for which $\mathcal{W} = \mathbb{R}^2$ and $\mathcal{W} = \mathbb{R}^3$ represents a 2-dimensional or 3-dimensional world respectively. Suppose $\mathcal{W}$ contains an obstacle region, $\mathcal{O} \subset \mathcal{W}$ and let a rigid body robot be defined as $\mathcal{A} \subset \mathcal{W}$ and a multi-body robot as $\mathcal{A}_i \subset \mathcal{W}$, where $i \in \{1, 2, \ldots, n\}$ denotes the link i. The configuration of $\mathcal{A}$ is denoted as $\mathbf{q} \in \mathcal{C}$, which is determined by specifying the set of all possible transformations that may be applied to the robot. The task of motion planning is then to find a collision-free path between an initial and a final configuration.

Obstacles

An obstacle in configuration space, $\mathcal{C}_{obs} \subseteq \mathcal{C}$, is defined as

$$\mathcal{C}_{obs} = \bigcup_{i=1}^{n} \{\mathbf{q} \in \mathcal{C} | \mathcal{A}_i(\mathbf{q}) \cap \mathcal{O} \neq \emptyset\}. \tag{3.45}$$

The configuration space that is collision-free is denoted as free-space, $\mathcal{C}_{free}$, and can be defined as

$$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}. \tag{3.46}$$

A path, that connects the initial configuration $\mathbf{q}_I$ and the final configuration $\mathbf{q}_f$ as $\rho : [0, 1] \to \mathcal{C}$, with $\rho(0) = \mathbf{q}_I$ and $\rho(1) = \mathbf{q}_f$, should obviously be designed collision free and can be formulated as

$$\rho \subseteq \mathcal{C}_{free}. \tag{3.47}$$

Self-Collision

When considering a robotic manipulator with multiple links, a collision can occur between different links of the manipulator. This can be defined as a set of collision pairs [88] $\mathcal{P}$, where each pair, $(i, j) \in \mathcal{P}$ represents a pair of link indices $i, j \in \{1, 2, \ldots, n\}$, such that $i \neq j$. It has to be noted that not all pairs are represented in $\mathcal{P}$, since consecutive links are already connected to each other. Formally, the self-collision space, $\mathcal{C}_{scol}$, can be defined as

$$\mathcal{C}_{scol} = \bigcup_{[i,j] \in \mathcal{P}} \{\mathbf{q} \in \mathcal{C} | \mathcal{A}_i(\mathbf{q}) \cap \mathcal{A}_j(\mathbf{q}) \neq \emptyset\}. \tag{3.48}$$

The complete collision space for the manipulator, $\mathcal{C}_{col} \subseteq \mathcal{C}$, can then be defined as the set union of the obstacle space and the self-collision space, or

$$\mathcal{C}_{col} = \mathcal{C}_{obs} \bigcup \mathcal{C}_{scol}. \tag{3.49}$$

As final note, it should be mentioned that path planning only considers spatial constraints. Kinematic or dynamic constraints are not taken into account.

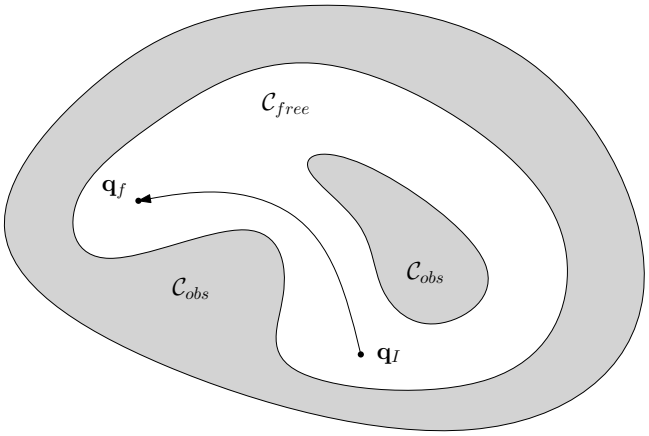


Figure 3.4: The task of path planning is to find a path from $\mathbf{q}_I$ to $\mathbf{q}_f$ in $\mathcal{C}_{free}$. The entire space is represented as $\mathcal{C} = \mathcal{C}_{free} \cup \mathcal{C}_{obs}$.

3.5 Trajectory Planning

Traditionally, trajectories can be classified into several categories, i.e., polynomial, trigonometric and exponential. The trajectories considered in this chapter are based on polynomial functions and are designed by choosing an appropriate polynomial and by setting up a (linear) system of equations. The order of this polynomial trajectory thus depends on the constraints set on each point (e.g., velocity, acceleration, jerk) or on the trajectory itself (e.g., timing, length). A further extension takes the dynamics of the system into account and is referred to as kinodynamic planning. When designing such motion profile, the traditional approach is to compute the trajectory generation off-line. Motion is then executed by comparing measurements with the known trajectory on-line (i.e., feedback control).

3.5.1 Constraints on Trajectories

When designing a trajectory for motion control, the constraints on the trajectory act in different domains. The trajectory constraints can be defined on a specific point of the trajectory (local), or on the complete trajectory (global). For example, a local constraint could involve motion constraints on specific points, while a global constraint could define the continuity of a trajectory. The following lists the most common constraints applicable for motion trajectories.

Kinematic Constraints

In practical applications, the constraints on the trajectory depend on the physical limits of the actuators. These actuation limitations can be described as constraints on the velocity and acceleration. For the velocity $\dot{\mathbf{q}}$ and the acceleration $\ddot{\mathbf{q}}$ this is defined per joint as

$$\begin{aligned} \dot{\mathbf{q}} &\in [\dot{\mathbf{q}}_{\min}, \dot{\mathbf{q}}_{\max}], \quad \dot{\mathbf{q}}_{\min}, \dot{\mathbf{q}}_{\max} \in \mathbb{R}^n, \\ \ddot{\mathbf{q}} &\in [\ddot{\mathbf{q}}_{\min}, \ddot{\mathbf{q}}_{\max}], \quad \ddot{\mathbf{q}}_{\min}, \ddot{\mathbf{q}}_{\max} \in \mathbb{R}^n, \end{aligned} \tag{3.50}$$

where the lower limit denoted with $_{min}$ is usually < 0, and the upper limit denoted with $_{max}$ is usually > 0.

When positioning with trajectories that have piecewise constant velocity or acceleration profiles, or trajectories that contain discontinuities in velocity or acceleration profiles, the jerk (derivative of acceleration) may become infinite. This can lead to effects such as vibrations and therefore wear of the mechanical system. The jerk should therefore be constrained to minimize the following functional:

$$\int_{t=0}^{T} \dddot{\mathbf{q}}^2(t)dt. \tag{3.51}$$

Another common constraint criteria is execution time. An electromechanical system should execute a motion within a certain time-period, thereby effectively constraining the velocity or acceleration. For a minimum time trajectory this is written as

$$\min \left[t_f - t_i\right], \text{where } t_f > t_i. \tag{3.52}$$

Dynamic Constraints

The physical limitations of an actuator impose an important constraint to the controller design. One example is the torque $\boldsymbol{\tau}$, which is a dynamic constraint and is bounded as

$$\boldsymbol{\tau} \in [\boldsymbol{\tau}_{\text{min}}, \boldsymbol{\tau}_{\text{max}}], \quad \boldsymbol{\tau}_{\text{min}}, \boldsymbol{\tau}_{\text{max}} \in \mathbb{R}^n, \tag{3.53}$$

where $\boldsymbol{\tau}_{\text{min}} < 0$ and $\boldsymbol{\tau}_{\text{max}} > 0$.

Furthermore, dynamic trajectory generation incorporates the dynamics of the system to be controlled into the trajectory. This is usually modelled by considering the elastic, dissipative and inertial properties of the system. To avoid vibrations and to avoid exciting resonance frequencies, the trajectory has to be free of discontinuities. In particular, the acceleration profile should be smooth, as this is directly linked to the inertial forces applied to a motion system. This can be expressed in terms of parametric continuity. A trajectory profile has a parametric continuity C^{n_p} when the ${n_p}^{th}$ derivative of the trajectory with respect to time is continuous. This not only holds for the trajectory profile itself, but also for subsequent trajectories, on connecting points. If, for example, the trajectory should have a continuous acceleration profile, then $n_p \geq 2$.

Spatial Constraints

The boundaries of a manipulator's workspace can be defined as a spatial constraint. This can be due to the finite length or manoeuvrability of the manipulator. In mathematical terms this can be expressed as

$$\mathbf{q} \in [\mathbf{q}_{\text{min}}, \mathbf{q}_{\text{max}}], \quad \mathbf{q}_{\text{min}}, \mathbf{q}_{\text{max}} \in \mathbb{R}^n.$$

These limits are not necessarily part of the trajectory generation directly, but can change a manipulator's path.

3.5.2 Basic Trajectory Profiles

Trajectories can be generated using several basic elementary functions (e.g., trigonometric, exponential, polynomial, etc.). As a complete overview of this can be found in literature (see e.g., [11]), this introduction will be limited to the function that is used extensively throughout this work.

Consider a polynomial function of degree n_t of the form

$$q(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_{n_t} t^{n_t}, \tag{3.54}$$

with $t \in [t_I,\ t_f]$, where t_I indicates the initial time instant ($t = 0$) and t_f indicates the final time instant. The $n_t + 1$ polynomial coefficients a_i can be determined while satisfying a number of required constraints.

A general solution is acquired by solving a system of linear equations:

$$\mathbf{Ta} = \mathbf{b}, \tag{3.55}$$

where $\mathbf{T}$ is the so-called Vandermonde matrix (see (3.61)) of size $(n_t + 1) \times (n_t + 1)$, $\mathbf{a}$ contains the unknown polynomial coefficients $\mathbf{a} = [a_0,\ a_1, \ldots,\ a_{n_t}]^T$ and $\mathbf{b}$ lists the $(n_t + 1)$ constraints that the polynomial should satisfy. Since matrix $\mathbf{T}$ is invertible, the coefficients $\mathbf{a}$ can be computed as

$$\mathbf{a} = \mathbf{T}^{-1}\mathbf{b}. \tag{3.56}$$

Using a polynomial interpolation method to determine a trajectory has the advantage that all points are crossed and that the trajectory is smooth. A drawback is the computational effort needed (complexity is of order: $O(n_t^2 + n_t^3)$) and the fact that for large values of n_t numerical errors may occur.

Point-to-Point Motion

Consider $q(t_I)$ and $q(t_f)$, the position at initial and final time instances of the point-to-point trajectory. Similarly, this can be written for the velocity $\dot{q}(t)$, acceleration $\ddot{q}(t)$ and the jerk $\dddot{q}(t)$.

Let at time instance t_I and t_f the constraints on the trajectory satisfy

$$\begin{aligned} q(t_I) &= q_I, & q(t_f) &= q_f, \\ \dot{q}(t_I) &= v_I, & \dot{q}(t_f) &= v_f, \\ \ddot{q}(t_I) &= \alpha_I, & \ddot{q}(t_f) &= \alpha_f. \end{aligned} \tag{3.57}$$

In motion control, the jerk negatively influences the efficiency of the control algorithm, and, as presented by Kyriakopoulos and Saridis [84], a lower jerk will lead to a lower positioning error. Furthermore, for simple trapezoidal trajectories, discontinuities occur during transition of constant to zero acceleration and velocity reversal. This jump in acceleration will cause infinite values for the jerk, leading to unwanted vibrations and electric noise in the power source. Therefore, to achieve smooth motion and a longer life-span of the robotic manipulator, minimum-jerk trajectories are a necessity. Flash and Hogan showed in [63, 45] that choosing the trajectory as a 5^{th} order polynomial, implies that the 6^{th} derivative is zero, which will minimize the integrated squared of jerk:

$$\int_{t=0}^{T} \dddot{q}^2(t)dt = \int_{t=0}^{T} \left[\frac{d^3q(t)}{dt^3}\right]^2 dt. \tag{3.58}$$

Proof is given with calculus of variations in Appendix A. The idea resembles finding the minimum of a function: find the derivative of the function with respect to a small perturbation and when that derivative is zero, the minimum is found. The trajectory has the form:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5, \tag{3.59}$$

for $0 \leq t \leq T$.

Continuing on this, the velocity and acceleration can be written as

$$\begin{aligned} \dot{q}(t) &= a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4, \\ \ddot{q}(t) &= 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3. \end{aligned} \tag{3.60}$$

In order to determine the polynomial coefficients of a trajectory, the equations (3.59) and (3.60) and the constraints of (3.57) are combined to set up a system of linear equations (i.e., the Vandermonde matrix [11]):

$$\mathbf{b} = \begin{bmatrix} q_0 & q_1 & \dots & q_{n_t-1} & q_{n_t} & v_0 & \alpha_0 & v_{n_t} & \alpha_{n_t} \end{bmatrix}^T = \mathbf{T}a =$$

$$\begin{bmatrix} 1 & t_0 & & \dots & t_0^{n_t+4} \\ 1 & t_1 & & \dots & t_1^{n_t+4} \\ & & & \vdots & \\ 1 & t_{n_t-1} & & \dots & t_{n_t-1}^{n_t+4} \\ 1 & t_{n_t} & & \dots & t_{n_t}^{n_t+4} \\ 0 & 1 & 2t_0 & \dots & (n_t+4)t_0^{n_t+3} \\ 0 & 0 & 2 & \dots & (n_t+4)(n_t+3)t_0^{n_t+2} \\ 0 & 1 & 2t_{n_t} & \dots & (n_t+4)t_{n_t}^{n_t+3} \\ 0 & 0 & 2 & \dots & (n_t+4)(n_t+3)t_{n_t}^{n_t+2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n_t-1} \\ a_{n_t} \\ a_{n_t+1} \\ a_{n_t+2} \\ a_{n_t+3} \\ a_{n_t+4} \end{bmatrix}, \tag{3.61}$$

where q_{mp}, $mp \in \{1, \dots, n_t - 1\}$ and the accompanying rows in $\mathbf{T}$ are only filled when a multipoint trajectory is to be designed. Moreover, if derivative constraints (e.g., velocity, acceleration) on via-points are also required, these can be incorporated in a similar fashion. The polynomial coefficients can then be found as $\mathbf{a} = \mathbf{T}^{-1}\mathbf{b}$, where

$$\begin{aligned} a_0 &= q_I, \\ a_1 &= v_I, \\ a_2 &= \frac{1}{2}\alpha_i, \\ a_3 &= \frac{1}{2T^3}\left[20h - (8v_f + 12v_I)T - (3\alpha_I - \alpha_f)T^2\right], \\ a_4 &= \frac{1}{2T^4}\left[-30h + (14v_f + 16v_i)T + (3\alpha_I - 2\alpha_f)T^2\right], \\ a_5 &= \frac{1}{2T^5}\left[12h - 6(v_f + v_I)T + (\alpha_f - \alpha_I)T^2\right], \end{aligned} \tag{3.62}$$

with $h = q_f - q_I$ and where it is assumed that $T = t_f - t_I$.
When regarding a trajectory with multiple points, again the concept of parametric continuity becomes important. Parametric continuity describes to what order two adjoining trajectories match. For example, C^2 indicates that adjoining trajectories have equal position, as well as velocity and acceleration at the intersection point (i.e., continuous up to the 2^{nd} derivative). Therefore, in order to avoid discontinuities in multipoint trajectories, a sufficiently high order polynomial has to be chosen.

Time Constraints on Trajectories

When leaving the execution time t_e unspecified, a minimum-time trajectory can be calculated by regarding the constraint on acceleration. This is often called a 'bang-bang' trajectory, since the acceleration will alternatively switch from maximum acceleration α_{max} to minimum acceleration α_{min} and vice-versa. If the minimum and maximum acceleration are equal in magnitude (i.e., $|\alpha_{min}| = |\alpha_{max}|$) and the trajectory is symmetric with respect to its middle point (i.e., flex point) the switching time t_s is found as $t_s = \frac{t_f - t_I}{2}$.

A different approach is to leave timing unconstrained and only regard the constraints on velocity and acceleration. The execution time of the trajectory then depends on the maximum velocity v_{max} or the maximum acceleration α_{max}. Consider the velocity trajectory, taken from a 5^{th} degree polynomial with initial constraints $q_I = \dot{q}_I = \ddot{q}_I = 0$, and final constraints $\dot{q}_f = \ddot{q}_f = 0$, as:

$$\dot{q}(t) = \left(\frac{30}{T^2} t^4 - \frac{60}{T} t^3 + 30t^2 \right) \frac{h}{T^3}. \tag{3.63}$$

Assuming a symmetric trajectory, the maximum velocity can be found at $t_{v,max} = \frac{t_f - t_I}{2}$. As $t \in [0, T]$ gives $t_{v,max} = 0.5T$, we can obtain:

$$\begin{aligned} v_{max} &= \max |\dot{q}(t)| = \frac{15}{8} \left(\frac{h}{T} \right), \text{ and thus,} \\ t_e &= \frac{15}{8} \frac{h}{v_{max}}, \end{aligned} \tag{3.64}$$

which means that the maximum velocity of a trajectory can be designed by altering the execution time.

Similarly, a maximum acceleration can be included. Consider the acceleration trajectory, again taken from a 5^{th} degree polynomial with initial constraints $q_I = \dot{q}_I = \ddot{q}_I = 0$, and final constraints $\dot{q}_f = \ddot{q}_f = 0$, as:

$$\ddot{q}(t) = \left(\frac{120}{T^2} t^3 - \frac{180}{T} t^2 + 60t \right) \frac{h}{T^3}. \tag{3.65}$$

Assuming a symmetric trajectory, the maximum acceleration can be found at $t_{\alpha,max} = \{\frac{1}{2} + \frac{\sqrt{3}}{6}, \frac{1}{2} - \frac{\sqrt{3}}{6}\}$, and we can obtain:

$$\begin{aligned} \alpha_{max} &= \max |\ddot{q}(t)| = \frac{10\sqrt{3}}{3} \frac{h}{T^2}, \text{ and thus,} \\ t_e^2 &= \frac{10\sqrt{3}}{3} \frac{h}{\alpha_{max}}, \end{aligned} \tag{3.66}$$

which means that the maximum acceleration of a trajectory can be designed by altering the execution time.

In practical situations, a maximum velocity v_{max} and a maximum acceleration α_{max} is given, or can be derived from motor specifications. The execution time can then be determined as:

$$t_e = \max \left\{ \frac{15}{8} \frac{h}{v_{max}}, \sqrt{\frac{10\sqrt{3}}{3} \frac{h}{\alpha_{max}}} \right\}. \tag{3.67}$$

Following, it can be identified from h that a linear relation exists between v_{max} and α_{max} as

$$\begin{aligned} h &= \frac{v_{max}}{\frac{15}{8}} T = \frac{\alpha_{max}}{\frac{10\sqrt{3}}{3}} T^2, \text{ and thus,} \\ v_{max} &= \frac{\alpha_{max}}{\frac{16}{9}\sqrt{3}} T, \end{aligned} \tag{3.68}$$

which means that the maximum acceleration of a trajectory can be designed by bounding the maximum velocity, and vice versa.

3.6 Summary

This chapter served as a brief recapitulation of basic modelling and planning of robotic manipulators. The presented content can be found in many well-known robotics and planning books, however due to this addition, these need not be consulted for a complete understanding of the remainder of this thesis.

First, the framework for the modelling of dynamics and kinematics is presented, which includes the topic of kinematic control with redundancy formulation. In particular, the use of kinematic redundancy for avoidance motion is presented, which includes several distance indices, i.e., towards a point, a line and a plane in Cartesian space, that can serve this purpose. The presented method is known as the gradient projection method where the gradient of a certain index can be used as secondary task. As side-note the inconsistency of the pseudo-inverse of the Jacobian, as used for redundant robot control, is addressed. This inconsistency originates from an addition of units which are not similar. Similarly, this can be found in the computation of the manipulability index for singularity avoidance. A proper alternative for this Jacobian pseudo-inverse is presented as the weighted Jacobian pseudo-inverse.

Finally, a detailed description of path and trajectory planning is presented. For path planning a formal representation of obstacles and self-collision is given. For trajectory planning, the different constraints a trajectory will encounter are addressed as well as an introduction to basic polynomial trajectory profiles. This includes the mathematical developments for polynomial point-to-point trajectories and their timing constraints. A 5^{th} order polynomial trajectory serves as example as this will minimize the integrated squared of jerk, and will therefore offer advantages for motion control (i.e., smooth motion, continuous acceleration).

Part II

Visual Control of Robotic Manipulators

CHAPTER 4

Modelling of 3D Vision

Abstract. This chapter discusses the modelling and analysis of 3 dimensional vision. As a sensor, a camera obtains measurements from which information can be extracted for the purpose of control and analysis. Relevant modelling which is considered includes modelling of the camera and lens and the relationship between two camera views in order to extract a 3D pose estimation.

4.1 Introduction

Vision-based perception considers the analysis of a 3D scene (the 'world') in the form of 2D measurements (the camera image). The modelling of all involved topics ranges from signal processing, kinematic modelling of rigid bodies, to the geometry and mathematical relations of camera models, lenses and projections. In this chapter, the modelling of 3D vision includes the pinhole camera model with geometric lens distortion and two view geometry. An accurate model of the camera and lens will correct for lens distortion and transforms image information into relevant Cartesian measurements. In order to extract 3D information from two images, two view geometry requires the decomposition of a homography matrix, which can be estimated by two sets of corresponding keypoints. As such, this involves an analysis of methods to detect keypoints in a scene (and from an object) to estimate the homography and subsequently the decomposition of this homography in order to obtain a 3D Cartesian position and orientation difference. The motivation behind this homography-based approach is the fact that an object model is not necessary for detection. As the analysis is purely image-based, a single image of an object is sufficient for localization in 3D space.

4.2 Pinhole Camera Model

Consider a point $\mathbf{x}$ in space with coordinates $\mathbf{x} = [x,\ y,\ z]^T \in \mathbb{R}^3$. Let $\mathcal{F}_{if}$ be the image frame onto which $\mathbf{x}$ is mapped as an image coordinate $\mathbf{p} = [u,\ v]^T \in \mathbb{R}^2$. This perspective projection considers the focal plane at a distance f and is defined as:

$$\mathbf{p} = [u,\ v]^T = [f\frac{x}{z},\ f\frac{y}{z}]^T. \tag{4.1}$$

Since the projection is centred through one point, this model is referred to as the pinhole camera model (see Fig. 4.1). In homogeneous coordinates (which is a convenient representation for projective geometry) and taking into account the camera's intrinsic parameters, this can be written as:

$$\mathbf{p}' = \mathbf{K}\mathbf{x}', \tag{4.2}$$

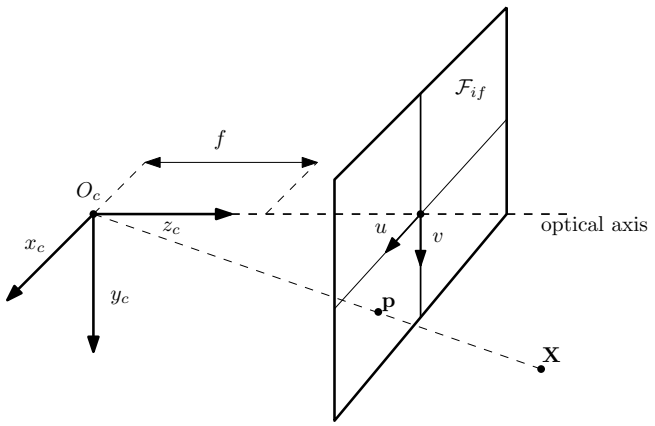


Figure 4.1: The pinhole camera model. Real world point **x** intersects the image plane at the image point **p** by a ray starting from the optical center O_c.

where $\mathbf{p}' = [u,\ v,\ 1]^T$, $\mathbf{x}' = [f\frac{x}{z},\ f\frac{y}{z},\ 1]^T$ and

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s_p & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.3}$$

where, α_x and α_y are scale factors in x and y-direction respectively, $(x_0,\ y_0)$ the coordinates of the principal point and s_p the skew parameter.

The extrinsic parameters, which describes the rotation and translation of the camera frame $\mathcal{F}_c$ with respect to the object (world) frame $\mathcal{F}_o$, are incorporated as

$$\tilde{\mathbf{p}}' = \mathbf{K}\,[\mathbf{I}_3\ \ \mathbf{0}_{3\times 1}] \begin{bmatrix} \mathbf{R}_{oc} & \mathbf{t}_{oc} \\ 0 & 1 \end{bmatrix} \bar{\mathbf{x}}, \tag{4.4}$$

where $\bar{\mathbf{x}} = [x,\ y,\ z,\ 1]^T$, $\mathbf{I}_3$ is the 3×3 identity matrix, $\mathbf{0}_{3\times 1}$ is a 3×1 zero vector and $\mathbf{R}_{oc}$ and $\mathbf{t}_{oc}$ represent the rotation and translation component.

4.2.1 Camera Calibration

In the real world, the pinhole camera model is affected by the distortion of the lens. Most commonly, lenses suffer from radial and tangential distortion, which can be modelled and corrected for (see Fig. 4.2).

In literature many different state-of-the-art camera calibration techniques are identified, e.g., a review and survey can be found in [123], a historical review can be found in [30]. These are mainly aimed at macroscopic camera calibration, where a minimization technique is used to find the optimal intrinsic and extrinsic camera parameters. Three techniques of camera calibration stand out in literature due to their accuracy and robustness; Tsai [146], Zhang [162] and Heikillä [60]. Tsai's model uses only a second order radial distortion model, while Zhang incorporates second and fourth order terms. Tsai's algorithm determines a system of n linear equations based on the radial alignment constraint [146] to solve for the extrinsic parameters. The second step uses a non-linear optimization scheme to determine the intrinsic parameters.

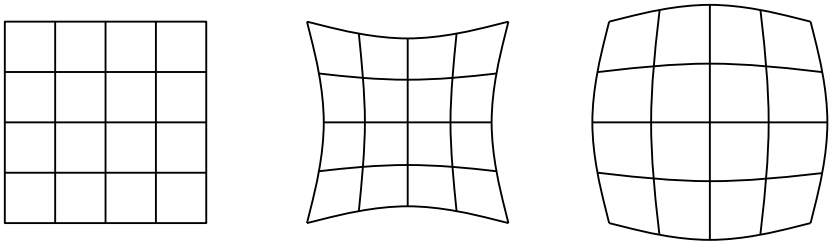


Figure 4.2: Left: no distortion, middle: pincushion distortion, right: barrel distortion.

Zhang's method uses multiple calibration pattern orientations to compute a projective transformation between image points to solve a linear set of equations and obtain the camera's internal and external parameters. A non-linear minimization of the reprojection error then optimizes all recovered parameters [162]. Heikillä's method is different for also incorporating tangential distortion components. First, a direct linear transformation (DLT) determines an initial estimate of the camera parameters. A non-linear least squares estimation then optimizes the internal parameters and computes the distortion parameters.

In this work the method of Tsai is chosen for camera calibration. The tangential distortion component is neglected and only a single radial distortion parameter (κ_1) is modelled since it is noted in several studies [146, 15] that this is sufficient when dealing with industrial machine vision lenses.

The relation between distorted image points $\mathbf{p} = [u,\ v]$ and undistorted (or corrected) image points $\mathbf{p}_{cor} = [u_{cor},\ v_{cor}]$ can then be defined as

$$\begin{aligned} u_{cor} &= u(1+\kappa_1 r^2), \\ v_{cor} &= v(1+\kappa_1 r^2), \end{aligned} \tag{4.5}$$

with $r^2 = u^2 + v^2$. By combining equations (4.1), (4.4) and (4.5), the following expression can be written for the calibration model:

$$u(1+\kappa_1 r^2) = f\frac{r_{1,1}x + r_{1,2}y + r_{1,3}z + t_x}{r_{3,1}x + r_{3,2}y + r_{3,3}z + t_z}, \tag{4.6}$$

$$v(1+\kappa_1 r^2) = f\frac{r_{2,1}x + r_{2,2}y + r_{2,3}z + t_y}{r_{3,1}x + r_{3,2}y + r_{3,3}z + t_z}, \tag{4.7}$$

with f the focal length. The entries of the rotation matrix and translation vector are taken as

$$\mathbf{R}_{oc} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix}, \quad \mathbf{t}_{oc} = \begin{bmatrix} t_x,\ t_y,\ t_z \end{bmatrix}^T. \tag{4.8}$$

For readability, the subscript oc is omitted.

Tsai's two step procedure can then be employed by first determining the extrinsic parameters through a closed form solution and a radial alignment constraint (RAC), which assumes that the lens distortion occurs only in the radial

direction from the optical axis. An overdetermined system of linear equations is set up as

$$\begin{bmatrix} vx \\ vy \\ vz \\ v \\ -ux \\ -uy \\ -uz \end{bmatrix}^T \begin{bmatrix} t_y^{-1} r_{1,1} \\ t_y^{-1} r_{1,2} \\ t_y^{-1} r_{1,3} \\ t_y^{-1} t_x \\ t_y^{-1} r_{2,1} \\ t_y^{-1} r_{2,2} \\ t_y^{-1} r_{2,3} \end{bmatrix} = u, \tag{4.9}$$

which can be solved with $\mathbf{p} > 7$ calibration points.

Step 2 consists of a nonlinear optimization routine to determine the intrinsic parameters. Equations (4.6) and (4.7) can be rewritten as

$$f(p+q) - \kappa_1(u+v)r^2 = u+v, \tag{4.10}$$

where p represents the right hand side of (4.6) and q represents the right hand side of (4.7). With $\mathbf{p} > 2$ calibration points, an overdetermined system of linear equations can be set up and solved for κ_1 and f:

$$\begin{bmatrix} p+q & -(u+v)r^2 \end{bmatrix} \begin{bmatrix} f \\ \kappa_1 \end{bmatrix} = u+v. \tag{4.11}$$

These initial estimates for κ_1 and f are then perfected with a nonlinear optimization scheme to obtain an accurate solution. Details of both steps can be found in [146].

4.3 Two View Geometry

In this section the relationship between two different views of the same 3D points is investigated. One method considers the concept of projective transformations, also known as homographies. Such homography describes the displacement (translation and rotation) of a camera and is reconstructed from two point sets of the same 3D points in the scene. The advantage of a homography-based approach is that in human-centered environments, the scene is inherently planar. This means that a rough approximation of objects and the scene itself can be modelled by a plane, making the reconstruction problem significantly easier. The following derivation of a planar homography is followed from [94], however, many other textbooks can also be consulted [43, 56].

Consider two images ($\mathcal{I}_1$, $\mathcal{I}_2$) of points $\mathbf{p}$ on a 2D plane $\mathbf{I}_p$ as depicted in Fig. 4.3. The coordinate transformation between these two planes can then be written as

$$\mathbf{x}_2 = \mathbf{R}\mathbf{x}_1 + \mathbf{t}, \tag{4.12}$$

where $\mathbf{x}_1 \in \mathbb{R}^3$ and $\mathbf{x}_2 \in \mathbb{R}^3$ are the spatial coordinates of $\mathbf{p}$ relative to camera frames 1 and 2, respectively.

Let $\mathbf{n} = [n_1\ n_2\ n_3]^T \in \mathbb{S}^2$ be the unit normal vector of the plane $\mathbf{I}_p$ with respect to the first camera frame, with $d > 0$ the depth from the optical center of the first camera towards the plane $\mathbf{I}_p$. This can then be written as

$$\mathbf{n}^T\mathbf{x}_1 = n_1x + n_2y + n_3z = d \Leftrightarrow \frac{\mathbf{n}^T}{d}\mathbf{x}_1 = 1, \quad \forall \mathbf{x}_1 \in \mathbf{I}_p. \tag{4.13}$$

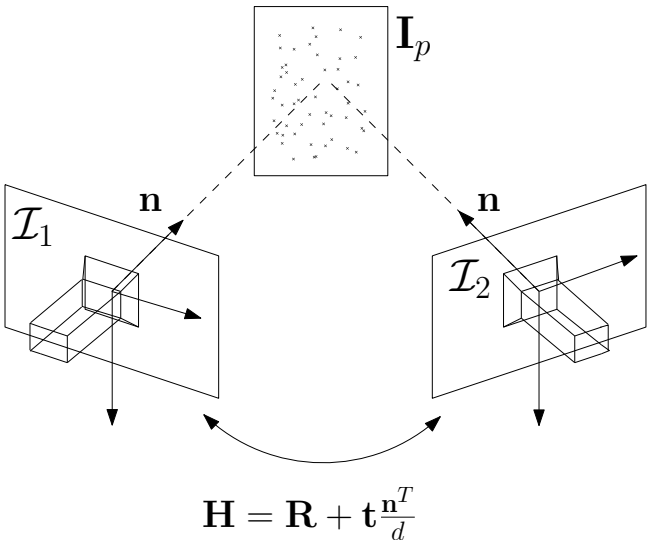


Figure 4.3: Homography transformation $\mathbf{H}$ between image $\mathcal{I}_1$ and image $\mathcal{I}_2$. $\mathbf{I}_p$ is the object plane that contains the feature points to be matched between the two frames.

Substituting (4.13) into (4.12) will give

$$\mathbf{x}_2 = \mathbf{R}\mathbf{x}_1 + \mathbf{t} = \mathbf{R}\mathbf{x}_1 + \frac{\mathbf{t}\mathbf{n}^T}{d}\mathbf{x}_1 = \left(\mathbf{R} + \frac{\mathbf{t}\mathbf{n}^T}{d}\right)\mathbf{x}_1. \tag{4.14}$$

The planar homography matrix is therefore defined as

$$\mathbf{H} \doteq \mathbf{R} + \frac{\mathbf{t}\mathbf{n}^T}{d} \in \mathbb{R}^{3\times 3}, \tag{4.15}$$

since the transformation from $\mathbf{x}_1 \in \mathbb{R}^3$ to $\mathbf{x}_2 \in \mathbb{R}^3$ is defined as

$$\mathbf{x}_2 = \mathbf{H}\mathbf{x}_1. \tag{4.16}$$

Due to the inherent scale ambiguity in the term $\frac{\mathbf{t}}{d}$ in (4.15), only a scaled translation can be recovered from $\mathbf{H}$.

4.3.1 Homography Estimation

A common method to estimate a homography is known as the Direct Linear Transform (DLT) and is derived as follows. Given a set of 2D to 2D point correspondences, $\mathbf{p}'_{1,i} \leftrightarrow \mathbf{p}'_{2,i}$, a perspective transformation is written as $\mathbf{p}'_{2,i} = \mathbf{H}\mathbf{p}'_{1,i}$, with $i \in \{1, 2, \ldots, n_{ip}\}$. As this definition involves homogeneous vector transformations, it can be expressed in the form of a vector cross product as

$$\mathbf{p}'_{2,i} \times \mathbf{H}\mathbf{p}'_{1,i} = \mathbf{0}. \tag{4.17}$$

With $\mathbf{p}'_{2,i} = [u_{2,i},\ v_{2,i},\ 1]^T$ and $\mathbf{p}'_{1,i} = [u_{1,i},\ v_{1,i},\ 1]^T$, this results in

$$\begin{bmatrix} \mathbf{0}^T & -\mathbf{p}'^T_{1,i} & v_{2,i}\mathbf{p}'^T_{1,i} \\ \mathbf{p}'^T_{1,i} & \mathbf{0}^T & -u_{2,i}\mathbf{p}'^T_{1,i} \\ -v_{2,i}\mathbf{p}'^T_{1,i} & u_{2,i}\mathbf{p}'^T_{1,i} & \mathbf{0}^T \end{bmatrix} \mathbf{h} = \mathbf{0}, \tag{4.18}$$

where $\mathbf{h} = [h_1,\ h_2,\ h_3,\ h_4,\ h_5,\ h_6,\ h_7,\ h_8,\ h_9]^T$, as to be found in the homography matrix as

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}. \tag{4.19}$$

It is common practise to omit the last row in (4.18), since only the first two rows are linearly independent[1]. A detailed expression can be found as $\mathbf{Ah} = \mathbf{0}$ or

$$\begin{bmatrix} 0 & 0 & 0 & -u_{1,i} & -v_{1,i} & -1 & u_{1,i}v_{2,i} & v_{1,i}v_{2,i} & v_{2,i} \\ u_{1,i} & v_{1,i} & 1 & 0 & 0 & 0 & -u_{1,i}u_{2,i} & -v_{1,i}u_{2,i} & u_{2,i} \end{bmatrix} \mathbf{h} = \mathbf{0}. \tag{4.20}$$

where $\mathbf{h} = [h_1,\ h_2,\ h_3,\ h_4,\ h_5,\ h_6,\ h_7,\ h_8]^T$.

Since each point correspondence provides 2 equations, 4 correspondences are sufficient to solve for the 8 degrees of freedom of $\mathbf{H}$. The resulting 8×9 matrix $\mathbf{A}$ is then formed and the 1D null space of $\mathbf{A}$ is the solution space for $\mathbf{h}$. If more than four point correspondences are given, the set of equations $\mathbf{Ah} = \mathbf{0}$ is over-determined. Commonly, a cost function is evaluated that minimizes the norm of $\mathbf{h}$ as $|\mathbf{h}| = 1$. Since $\mathbf{H}$ is determined up to a scale factor, the actual value to be minimized is not of importance. This resulting algorithm is commonly known as the basic Direct Linear Transform (DLT) [56].

RANSAC

The homography estimation algorithm assumes a perfect match between correspondences $\mathbf{p}_{1,i}$ and $\mathbf{p}_{2,i}$. It is, however, very likely that many points, as extracted from an image, are mismatched or can not be matched at all. One algorithm that separates the inliers from the outliers is known as RANSAC (RANdom SAmple Consensus). RANSAC iteratively computes a homography from 4 points and uses this to classify all other correspondences. The iteration containing the largest number of inliers is eventually chosen, from which a final $\mathbf{H}$ is recomputed. The distance measure which classifies correspondences (d_e) is chosen as the Euclidean distance between two points (i.e., $d_e = |\mathbf{p}'_{2,i} - \mathbf{H}\mathbf{p}'_{1,i}|$) [44]. A correspondence pair is chosen to be part of a homography if $d_e < d_t$, for some threshold d_t. The algorithm in recursive form is presented in pseudocode in Algorithm 4.1.

Algorithm 4.1 RANSAC

1: Compute $\mathbf{H}$ from 4 randomly selected points
2: Select all the correspondence pairs that coincide with this $\mathbf{H}$, i.e., if $d_e < d_t$
3: Repeat step 1 and 2 until a sufficient number of correspondence pairs coincide with $\mathbf{H}$
4: Compute $\mathbf{H}$ from all coincided correspondence pairs

As it can become infeasible to try every combination of correspondence points, a termination mechanism has to be employed to limit the number of iterations. This can be done by choosing a probability (indirectly set by d_t), such that at least one of the random samples of the 4 points is free from outliers. A different, practical rule of thumb can be used that terminates the loop

[1] the third row can be obtained from the first and second row

when the number of correspondence pairs is equal to the number of inliers as to be expected in the data set [56].

4.3.2 Homography Decomposition

As stated in [94], a planar homography matrix of the form $\mathbf{H} = \mathbf{R} + \frac{\mathbf{t}\mathbf{n}^T}{d}$ has at most two physically possible solutions for a decomposition into the components $\{\mathbf{R}, \mathbf{t}, \mathbf{n}\}$. From the fact that $\mathbf{H}^T\mathbf{H}$ is symmetric and has three eigenvalues $\sigma_1^2 \geq \sigma_2^2 \geq \sigma_3^2 \geq 0$, and the fact that $\sigma_2 = 1$ [94], the following decomposition can be made:

$$\mathbf{H}^T\mathbf{H} = \mathbf{V}\Sigma\mathbf{V}^T, \tag{4.21}$$

where Σ is a diagonal matrix consisting of the singular values of $\mathbf{H}^T\mathbf{H}$ sorted in decreasing order. With $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$, this can be rearranged as

$$\mathbf{H}^T\mathbf{H}\mathbf{v}_1 = \sigma_1^2\mathbf{v}_1, \;\; \mathbf{H}^T\mathbf{H}\mathbf{v}_2 = \mathbf{v}_2, \;\; \mathbf{H}^T\mathbf{H}\mathbf{v}_3 = \sigma_3^2\mathbf{v}_3. \tag{4.22}$$

Thus, $\mathbf{v}_2$ is orthogonal to both $\mathbf{n}$ and $\mathbf{t}$, and its length is preserved under the map $\mathbf{H}$. Also, it can be checked that the length of two other unit-length vectors defined as

$$\begin{aligned} \mathbf{u}_1 &\doteq \frac{\mathbf{v}_1\sqrt{1-\sigma_3^2} + \mathbf{v}_3\sqrt{\sigma_1^2-1}}{\sqrt{\sigma_1^2-\sigma_3^2}}, \\ \mathbf{u}_2 &\doteq \frac{\mathbf{v}_1\sqrt{1-\sigma_3^2} - \mathbf{v}_3\sqrt{\sigma_1^2-1}}{\sqrt{\sigma_1^2-\sigma_3^2}}, \end{aligned} \tag{4.23}$$

are also preserved under the map $\mathbf{H}$. Furthermore, it can be verified that $\mathbf{H}$ preserves the length of any vector inside each of the two subspaces

$$\mathbf{S}_1 = \text{span}\{\mathbf{v}_2, \mathbf{u}_1\}, \quad \mathbf{S}_2 = \text{span}\{\mathbf{v}_2, \mathbf{u}_2\}. \tag{4.24}$$

Since $\mathbf{v}_2$ is orthogonal to $\mathbf{u}_1$ and $\mathbf{u}_2$, $[\mathbf{v_2}]_\times\mathbf{u}_1$ is a unit normal vector to $\mathbf{S}_1$, and $[\mathbf{v_2}]_\times\mathbf{u}_2$ a unit normal vector to $\mathbf{S}_2$. $\{\mathbf{v}_2, \mathbf{u}_1, [\mathbf{v_2}]_\times\mathbf{u}_1\}$ and $\{\mathbf{v}_2, \mathbf{u}_2, [\mathbf{v_2}]_\times\mathbf{u}_2\}$ then form two sets of orthonormal bases for $\mathbb{R}^3$. In this, $[-]_\times$ denotes the skew-symmetric matrix determined from the associated vector. Notice that we have

$$\mathbf{R}\mathbf{v}_2 = \mathbf{H}\mathbf{v}_2, \quad \mathbf{R}\mathbf{u}_1 = \mathbf{H}\mathbf{u}_1, \quad \mathbf{R}([\mathbf{v_2}]_\times\mathbf{u}_1) = [\mathbf{H}\mathbf{v}_2]_\times\mathbf{H}\mathbf{u}_1, \tag{4.25}$$

$$\mathbf{R}\mathbf{u}_2 = \mathbf{H}\mathbf{u}_2, \quad \mathbf{R}([\mathbf{v_2}]_\times\mathbf{u}_2) = [\mathbf{H}\mathbf{v}_2]_\times\mathbf{H}\mathbf{u}_2, \tag{4.26}$$

if $\mathbf{n}$ is the normal to the subspace $\mathbf{S}_1$ and $\mathbf{S}_2$. Defining the matrices

$$\begin{aligned} \mathbf{U}_1 &= [\mathbf{v}_2, \mathbf{u}_1, [\mathbf{v_2}]_\times\mathbf{u}_1], \\ \mathbf{U}_2 &= [\mathbf{v}_2, \mathbf{u}_2, [\mathbf{v_2}]_\times\mathbf{u}_2], \\ \mathbf{W}_1 &= [\mathbf{H}\mathbf{v}_2, \mathbf{H}\mathbf{u}_1, [\mathbf{H}\mathbf{v}_2]_\times\mathbf{H}\mathbf{u}_1], \\ \mathbf{W}_2 &= [\mathbf{H}\mathbf{v}_2, \mathbf{H}\mathbf{u}_2, [\mathbf{H}\mathbf{v}_2]_\times\mathbf{H}\mathbf{u}_2], \end{aligned} \tag{4.27}$$

leads to

$$\mathbf{R}\mathbf{U}_1 = \mathbf{W}_1, \qquad \mathbf{R}\mathbf{U}_2 = \mathbf{W}_2, \tag{4.28}$$

which suggests that each subspace $\mathbf{S}_1$, or $\mathbf{S}_2$ may give rise to a solution to the decomposition. By taking into account the extra sign ambiguity in the term $\frac{\mathbf{t}\mathbf{n}^T}{d}$ we then obtain four solutions for the decomposition as

$$\left\{ \begin{array}{lcl} \mathbf{R}_1 & = & \mathbf{W}_1\mathbf{U}_1^T \\ \frac{\mathbf{t}_1}{d} & = & (\mathbf{H} - \mathbf{R}_1)\,\mathbf{n}_1 \\ \mathbf{n}_1 & = & [\mathbf{v_2}]_\times \mathbf{u}_1 \end{array} \right\}, \left\{ \begin{array}{lcl} \mathbf{R}_3 & = & \mathbf{R}_1 \\ \frac{\mathbf{t}_3}{d} & = & -\frac{\mathbf{t}_1}{d} \\ \mathbf{n}_3 & = & -\mathbf{n}_1 \end{array} \right\},$$

$$\left\{ \begin{array}{lcl} \mathbf{R}_2 & = & \mathbf{W}_2\mathbf{U}_2^T \\ \frac{\mathbf{t}_2}{d} & = & (\mathbf{H} - \mathbf{R}_2)\,\mathbf{n}_2 \\ \mathbf{n}_2 & = & [\mathbf{v_2}]_\times \mathbf{u}_2 \end{array} \right\}, \left\{ \begin{array}{lcl} \mathbf{R}_4 & = & \mathbf{R}_2 \\ \frac{\mathbf{t}_4}{d} & = & -\frac{\mathbf{t}_2}{d} \\ \mathbf{n}_4 & = & -\mathbf{n}_2 \end{array} \right\}. \tag{4.29}$$

Due to the fact that the camera can only perceive points in front of it (i.e., a positive depth) the number of solutions reduces to two. Obtaining the correct solution from the remaining two can be done via several methods. For instance, Vargas et al. show in [148] how the average of the two solutions (i.e., for the translation and rotation) can be used, such that the system will converge in such a way that it is always possible to discard the false solution. Different methods exist which use a second plane in the image [125] or a third image [20] to estimate the common normal vector. Obvious difficulties arise in the latter case where at start-up only two images are available. This problem can be solved by selecting a virtual reference plane from the feature points, and therefore also uses a second plane. Finally, having knowledge of the task to be executed (i.e., the positioning motion), an estimate of this motion can as well resolve the correct solution for the decomposition.

4.4 Keypoint Detection

As explained in Section 4.3.1, the motion transformation between two frames (i.e., initial-pose to end-pose) can be determined from two image views. The two main approaches towards retrieving a 3D motion transformation between two views are marker-based methods and natural feature-based methods.

Marker-Based

Markers such as barcodes or Glyphs are known to achieve a high robustness and repeatability regarding detection and matching for 3D motion estimation. ARToolKit[2] is one well-known software library and is used extensively in augmented reality (AR) applications[3]. As markers are commonly designed as a black pattern with a white background (see e.g., [17]), detection depends on intensity thresholding and template matching for recognition. Even though computationally inexpensive compared to natural-feature detection, this causes the marker detection to be linear in the number of markers. When using multiple markers in the field of view, performance may therefore be inadequate. Besides this performance issue, the main drawback of marker-based motion estimation is the presence of markers in the scene.

[2] see http://www.hitl.washington.edu/artoolkit
[3] see http://www.arlab.nl

Natural Feature-Based

The second method for 3D motion retrieval uses features that are naturally present on a object or in a scene. The approach encompasses that an object as used for reference, can be encoded and reduced to a number of 'salient' points. The transformation between these two views (or planes) can then be estimated from the two point sets. In this, the detection of natural features (i.e., keypoints) and the correspondence matching between them is far more computationally demanding. Furthermore, the presence of features on an object or in a scene is highly object dependent. Keypoints tend to cluster locally and uniform or unstructured patches can disturb or bias the decomposition of 3D motion parameters [147].

Depending on the task at hand (e.g., offline detection or classification of objects, augmented reality applications or real-time visual servoing), several detection and matching methods are available. A complete review of distinction between different methods is beyond the scope of this work (see e.g. [147] and [46] for a survey and review). Instead, a short introduction is given towards (ideal) natural keypoints and following, the two most popular algorithms for keypoint detection (i.e., SIFT and SURF) are recalled and their properties are discussed. Afterwards, experimental results present a comparison between the two detection methods.

4.4.1 Ideal Keypoints

The quality of a keypoint is based on a local patch of pixels, which means that the main measure for detecting a keypoint has a spatial nature. Furthermore, due to the discrete property of an image, the quality of feature detection decreases rapidly with decreasing image size (or feature size) and is amplified by a limited availability of computational power and computation time. The key consideration is therefore to find a balance between high-level interpretation of keypoints and the disposable computational resources. Properties that an ideal keypoint should encompass are listed as follows (see also [147]):

- **Repeatability**
  Keypoints, observed and transformed by any viewpoint change (e.g., perspective, affine, scale) should be detected in both images.
- **Distinctiveness**
  Different keypoints should differentiate from each other in the sense that their local identity can be distinguished from other (similar) keypoints.
- **Size**
  The amount of pixels used to define a keypoint, should relate to the distance (pitch) between keypoints (i.e., keypoints should not overlap).
- **Density**
  The amount of keypoints as found in an image should reflect the information present in an image. As such, the number of keypoints should be tunable with simple heuristics.
- **Time efficiency**
  Keypoint detection ideally has a deterministic timing layout and is linear with increasing number of features.

- **Memory efficiency**
  The local identity of a keypoint must be encoded in a fixed vector with limited memory resources.

- **Detector quality/accuracy**
  The combination of all properties leads to a global output of the keypoint detector that can be viewed as a quality measure. The main goal is to get an as high as possible quality with as little as possible resources utilized.

Disturbances that have the greatest impact on aforementioned properties are typical for any analog-digital conversion of signals. Noise and artefacts due to discretization, (motion) blur and compression must be included in the modelling of keypoint detection and matching such that these have an as low as possible impact. Other disturbances, which cannot be taken into account (or are not due to limited computational resources) are e.g., lighting artefacts or shadow, occlusion, geometric distortion or additional information which is not part of a reference image. Unavoidably, contradicting properties cannot be satisfied at the same time. The primary example of this is the balance between quality and computation time of the feature detector. One direction to solve this issue is by employing different processors (e.g., GPU, FPGA) for more computational resources, but even in this a limit is quickly reached.

From a historical point of view, the first detection methods mainly focussed on detecting corners, whereas more recently blob-like features are popular [147]. Both methods exploit the scale-space representation of an image, where at multiple resolutions (scales) detection operators are executed (see Fig. 4.4). Examples of corner-detectors are e.g., Harris-Laplace [103], which uses differential operators to find keypoints or FAST [121], which evaluates possible keypoints based on the surrounding intensity. Blob-detectors became of interest when the invariance to scale of (basic) corner-detectors showed not sufficient. Examples are the Laplacian of Gaussian (LoG) [8], the Difference of Gaussian (DoG) as implemented in SIFT [93] or the Determinant of Hessian (DoH) as implemented in SURF [7]. These detectors (i.e., SURF and SIFT) are typically designed with strong descriptive properties and thus have a high computational load. When the application requires descriptors with high repeatability and low computational load due to a relatively small and smooth motion between consecutive frames, other detectors can be used. For instance, in the field of augmented reality (see e.g., [3]), commonly detectors such as FAST (Features from Accelerated Segment Test) [121] or AGAST (Adaptive and Generic Accelerated Segment Test) [96] are employed. In particular, FAST is based on the accelerated segment test (AST), which is a modification of the SUSAN corner detector, as described in [136]. Similarly, the AGAST detector utilizes the same corner criterion as FAST but provides a performance increase for arbitrary environments, i.e., only the way the decision trees for the AST are built and used have been significantly improved (accelerated).

A brief comparison of different keypoint detection methods can be found in Table 4.1, a thorough review can be found in e.g., [147]. As the task at hand (i.e., real-time visual control) requires robust keypoint detection and strong descriptors, the choice of possible detection algorithms is limited to SIFT and its computationally cheaper version, SURF. These are explained in more detail in forthcoming section.

Table 4.1: Comparison of feature detectors (taken from [46])

	Harris	Shi-Tomassi [131]	FAST	SIFT	SURF	CenSurE[1]
Corner detector	x	x	x			
Blob detector				x	x	x
Rotation invariant	x	x	x	x	x	x
Scale invariant			x	x	x	x
Affine invariant				x	x	x
Repeatability	+++	+++	++	+++	+++	+++
Localization accuracy	+++	+++	++	++	++	++
Robustness	++	++	++	+++	++	+++
Efficiency	++	++	++++	+	++	+++

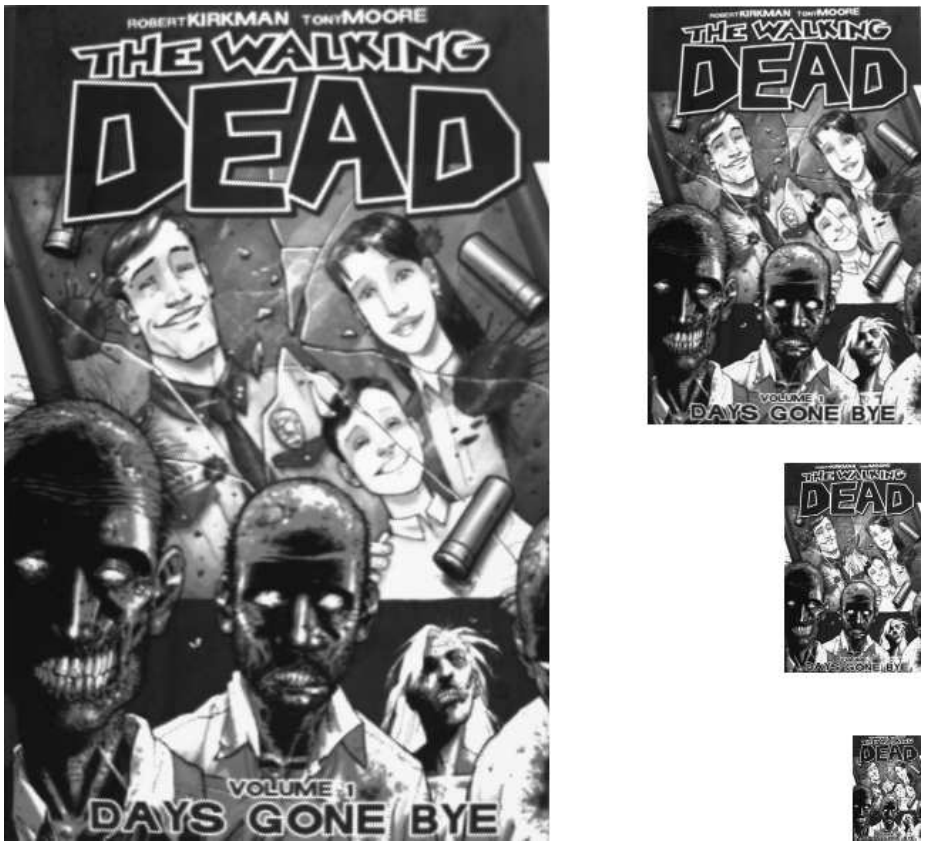


Figure 4.4: Scale-space pyramid of reference image.

4.4.2 Scale Invariant Feature Transform (SIFT)

This well-known work of David Lowe [93] can be summarized in a few steps:

1. Scale-space extrema detection,
2. Keypoint localization,
3. Orientation assignment,
4. Descriptor assembly.

The first step consists in computing the Difference of Gaussian (DoG) (which is an approximation of the Laplacian of Gaussian (LoG)) and detecting scale-space extrema across different scales. For this, a scale-space is created by convolving the image with a Gaussian blur at different octaves (i.e., images of similar size form an octave, and subsequent octaves are a down-sampling of a factor two) with different scales (i.e., level of blur). Following, DoG images are taken from the difference of adjacent blurred images per octave.

Keypoints are then localized by fitting a 3D quadratic in scale-space and taking the interpolated maximum as the actual keypoint location. Due to the sensitivity along edges (i.e., poor location, but high edge response), a measure of principal curvature m_p determines if candidate keypoints should be discarded or not:

$$m_p = \frac{\mathrm{tr}(\mathbf{H}_{im})^2}{\det(\mathbf{H}_{im})}, \quad \text{with } \mathbf{H}_{im} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}, \tag{4.30}$$

where the Hessian matrix $\mathbf{H}_{im}$ of the image patch at the location and scale of the candidate keypoint serves as curvature approximation. Moreover, D denotes the second-order partial derivative of the Difference of Gaussian, and tr and det denote the trace and determinant respectively.

The third step assigns an orientation to each keypoint based on the local image gradient direction. In addition to changes in scale, the descriptor then also becomes invariant to rotation. In order to compute this orientation, the image of the pyramid that is closest in scale to the keypoint's actual scale is chosen. The gradient magnitude $m(u,v)$ and orientation $\theta(u,v)$ at each pixel of the image $\mathcal{I}$ are then pre-computed using pixel differences:

$$\begin{aligned} m(u,v) &= \sqrt{[\mathcal{I}_{(u+1,v)} - \mathcal{I}_{(u-1,v)}]^2 + [\mathcal{I}_{(u,v+1)} - \mathcal{I}_{(u,v-1)}]^2}, \\ \theta(u,v) &= \arctan\left[\frac{\mathcal{I}_{(u,v+1)} - \mathcal{I}_{(u,v-1)}}{\mathcal{I}_{(u+1,v)} - \mathcal{I}_{(u-1,v)}}\right]. \end{aligned} \tag{4.31}$$

Based on $\theta(u,v)$, a 36-bin orientation histogram, covering 10 degrees each, is formed within a region around the keypoint's location. Each sample added to the histogram is weighted by $m(u,v)$ and a Gaussian around the keypoint. Peaks of the orientation histogram correspond to dominant directions of the local gradients and the highest peak is used as the keypoint's orientation. When other peaks within 80% of the highest occur, additional keypoints are created with the same location and scale as the original keypoint for each additional orientation.

In the last step, the distribution of local gradients are summarized in several histograms around the keypoint and concatenated to form the descriptor of length 128. After concatenating the histogram values in a single vector, normalization is applied which makes the descriptor more robust to illumination changes.

4.4.3 Speeded Up Robust Features (SURF)

The SURF [7] detector and descriptor is partly inspired by SIFT, however, designed with the intention to have comparable (or better) robustness yet faster computation. Like SIFT, SURF is based on the image representation in scale-space and on differential operators for detection. These are speeded up by using integral images (i.e., for finding the sum of a rectangular area) and box filters.

Detection

In the detection step, keypoints are localized in scale-space by approximating the Difference of Gaussian (DoG) by a Determinant of Hessian (DoH). This DoH is effectively implemented as box filters L_b which can be accelerated efficiently by using integral images.

$$\det(\widehat{\mathbf{H}}_{im}) = L_{b,uu}L_{b,vv} - w^2 L_{b,uv}^2. \tag{4.32}$$

In this, the weight w is determined from the Frobenius norm of the approximated determinant and the real determinant (i.e., $w \approx 0.6$). Keypoints are finally selected by using non-maxima suppression and scale-space interpolation.

The local orientation of each keypoint in scale-space is computed from the local neighbourhood around each keypoint in both directions. Again for faster computation, SURF employs scale-adapted Haar wavelets and integral images.

As orientation calculation can be an extra source of error, an upright version of SURF exists, denoted U-SURF. This method simply skips the orientation step by assigning each keypoint a zero orientation.

Descriptor

The descriptor of SURF keypoints are build upon the keypoint's orientation and gradient. The gradients are computed in an oriented 4×4 window of scale-adjusted size. The sums of gradients and the sums of absolute gradients are determined in both directions, yielding a 4-vector for each sub-region. Combining these produces a descriptor length of 64, which is normalized as final step for robustness against illumination changes.

4.5 Experimental Comparison

An experimental comparison of discussed keypoint detectors and descriptors will give more insight into their properties, benefits and shortcomings. Following, experiments are carried out to distinguish between SIFT and SURF as well as an analysis to assess their performance towards different translations and

rotations. Furthermore, the estimation and decomposition of the homography is also examined.

4.5.1 SURF versus SIFT

A first comparison is made between the two keypoint detectors and descriptors as presented in Section 4.4.3 (SURF) and Section 4.4.2 (SIFT). Even though rigorous analyses between both has been a popular topic (see e.g. [104] or [147]), we compare SURF and SIFT with respect to the number of detected keypoints, number of matches and computation time. This evaluation can be seen in Table 4.2.

Table 4.2: Comparison of SIFT and SURF keypoint detector and descriptor

SURF				SIFT		
Hessian threshold	800	1200	1200	Edge threshold	0.04	0.04
Octaves	4	2	2		-	-
Octave scales	2	2	1		2	3
Keypoints						
Reference	523	244	135		794	1369
Current	350	130	60		2300	4300
Matches	200	70	30		90	170
Comput. time [ms]	260	140	100		1300	1600

For the SURF detector and descriptor three experiments with different parameters are performed, for the SIFT detector and descriptor two experiments are performed. Each keypoint detection and matching experiment is executed 1000 times, from which an average number of keypoints, number of matches and computation time is computed. For the reference image only one detection step is necessary.

A first observation shows that SIFT is an order of magnitude slower than SURF. Secondly, the fact that for SIFT the number of octaves is determined automatically from the image resolution, makes it difficult to control the number of found matches and thus the computation time. For SURF a rough relation can be observed between the parameters of detection (i.e., octaves and octave scales), the number of found keypoints and consequently the time of computation. A trade-off has then to be found between the accuracy and repeatability of keypoint detection and matching and the available computation time. A comparison of SIFT and SURF with different parameters is shown in Table 4.2. In this, individual experiments are separated per column. The parameters of SURF (i.e., Hessian threshold, number of octaves and number of octave scales) can be altered in a more straight-forward way than the parameters of SIFT (i.e., edge threshold and number of octave layers). This is mostly due to the automatic computation of the number of octaves for SIFT. Furthermore, the number of found keypoints in a reference image and a current image differ quite significantly between SIFT and SURF. Despite the fact that SIFT detects more keypoints, this does not directly give an advantage in the keypoint matching process, as the number of matched points between the two methods is roughly similar. The biggest difference therefore, is the computation time. As the com-

putation time of SIFT is an order of magnitude greater than SURF, a real-time implementation with SURF would therefore be more favourable.

Fig. 4.5 shows a reference image (right) and the reference image found in a current view (left). The circles represent the keypoints, where the size of the circles depicts the scale in which the keypoint is detected. From the reference image it can be seen that certain areas in the image provide more keypoints than other areas. This local clustering of keypoints is a logical result of the keypoint detection process as images with high information content (e.g., many corners or intensity changes) can be represented in more detail than images with large uniform areas. In the current view in Fig. 4.5 (left) it can be seen that the reference image is partly blocked. A consequence of this is that not all correspondence points can be matched and only a part of the reference image is 'found'. Obviously this does not benefit the accuracy of the homography estimation and subsequently the translation and rotation decomposition. The importance of utilizing reference images with high information content over the whole image as well as over the whole scale-space range is therefore essential. A second example can be seen in Fig. 4.6.

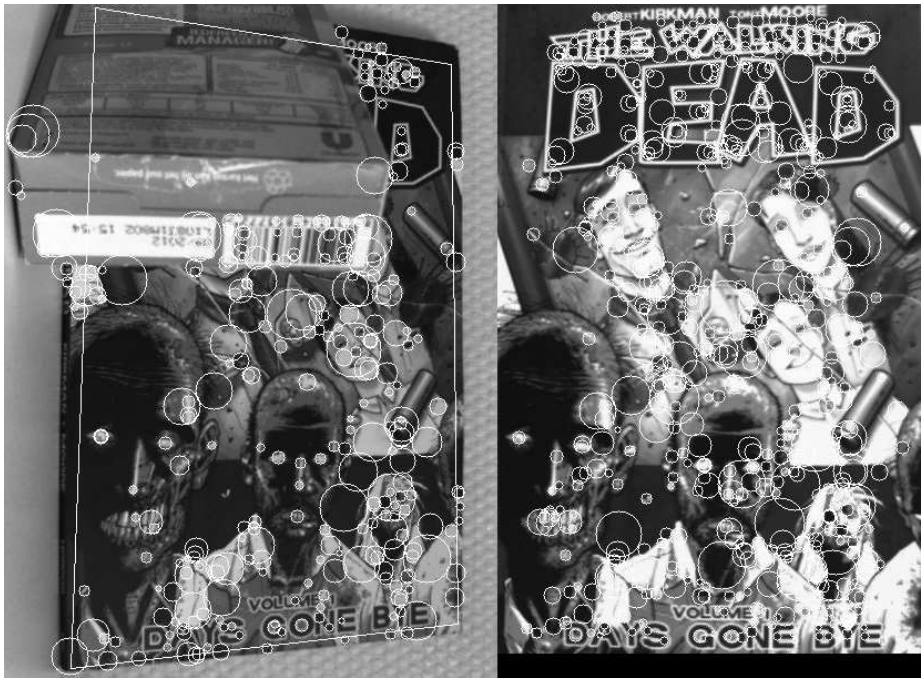


Figure 4.5: Current view image of a blocked comic book (left) and a reference image of a comic book (right). The circles represent the found keypoints, where the size of the circles depicts the scale in which the keypoint is detected. The left image also shows the found reference image which is outlined with a rectangle.

Figure 4.6: Example of SURF detector and descriptor. A relatively large number of false positive keypoints are detected which are discarded via the keypoint matching process. Despite the difference in image intensity, the reference image is found in the current image, as is indicated by the white rectangle.

Figure 4.7: Example sequence of reference images (soup box) for performance evaluation of SURF.

4.5.2 Performance of SURF

In order to asses the performance of the SURF detector and descriptor for real-time visual servo control, a number of experiments have been carried out.

The accuracy and repeatability of the homography decomposition into a rotational and translational difference is assessed by experiments with several offsets. For the rotational decomposition 7 different orientations with a 10 degree interval are used (see e.g., Fig. 4.7). For the translational decomposition 5 different translation are used. The parameters of the SURF detector are taken

as the second column in Table 4.2, which give a decent balance between computation time and performance. The object (i.e., the comic book and the soup box) is translated and rotated in each particular direction and orientation until the position and angular offset is reached. This configuration is maintained for 1000 iterations, to limit the influence of outliers. The performance is subsequently assessed by standard error evaluation measures (i.e., mean and 3σ). Table 4.3 shows the results of these experiments.

A first observation shows the limited accuracy and repeatability for determining an accurate orientation measurement. Even in an ideal case, a zero degree orientation, the average measurements are only accurate up to tenths of a degree and exacerbates with increasing orientation offset. The repeatability results show a similar trend, where a larger orientation offset results in a larger standard deviation (3σ). The decreasing accuracy for larger offset angles is due to several factors. First, an imperfectly calibrated vision system will project the scene distorted on the vision sensor and will therefore give incorrect measurements. Secondly, the local clustering of keypoints will bias the homography estimation. If more keypoints are detected in one area, this area will account for a higher accuracy than an area with less keypoints. In an ideal situation, keypoints should have a uniform distribution, however, in a real situation (e.g., with random images), this is never the case. For instance, it can be seen in Fig. 4.5 and Fig. 4.6 that the distribution of keypoints depends highly on the information content (i.e., texture or non-uniformity of patches) of the image. Moreover, the decrease in accuracy can particularly be observed for estimation involving depth. For instance, the roll angle accuracy is slightly better since depth is not involved. A logical reasoning for this is the limitation in sensor resolution as objects closer to the camera will cover more pixels than objects further away. Moreover, when the object is rotated in yaw and pitch, the covered area of the object becomes smaller which limits the number of pixels for detection as well.

As the estimation of the translation can only be observed up to a scale factor, this does not include a metric value. Experiments are carried out similarly to the orientation estimation and can be found in Table 4.3. It can be observed that the accuracy and variability (3σ) in x- and y-direction is worse than for the z-direction. This is most likely due to the fact that for a larger offset in x and y direction the object partially leaves the field of view. This leads to an incomplete reconstruction of the reference object, as a part of the object is simply missing. Again here, local clustering of keypoints (i.e., the non-uniformity in distribution of keypoints) can play a significant role, as the remaining keypoints can be biased as well. This can for instance occur when a part of the object is missing which contains the most keypoints. Clearly this does not occur as severe for a depth translation, however, similarly to the orientation estimation, the fact that objects further away cover less pixels will have an effect as well.

The choice of keypoint detector and descriptor depends highly on the task at hand and can be tailored to the expected motion and required level of invariance. A logical approach is to proceed with the most robust detector, to cover all possible types of invariance, however, this is not necessarily the best choice. In fact, as mentioned in literature [147], it is often better to rely on the robustness of the keypoint detector and descriptor rather than to increase the level of invariance. For instance, when the expected transformations are relatively

Table 4.3: Comparison of SURF keypoint detector and descriptor

angle offset	-30	-20	-10	0	10	20	30
[deg]							
roll							
mean	-29.77	-20.25	-10.27	0.78	10.18	20.53	30.32
3σ	1.54	1.26	0.89	0.41	1.14	1.42	1.72
pitch							
mean	-31.30	-20.98	-9.06	0.67	10.56	21.21	31.05
3σ	2.21	1.67	1.11	0.66	1.39	1.99	2.14
yaw							
mean	-30.44	-20.76	-10.98	0.53	10.06	20.28	30.37
3σ	2.87	1.78	1.48	0.77	1.24	1.36	2.43
position offset		-200	-100	0	100	200	
[-]							
x							
mean		-199.10	-100.37	0.21	90.41	200.56	
3σ		5.47	3.76	2.68	3.53	4.27	
y							
mean		-200.55	-99.20	0.83	100.55	198.78	
3σ		4.25	3.86	1.79	2.95	2.66	
z							
mean		-19.84	-9.94	0.24	10.54	20.21	
3σ		2.55	1.9	2.06	2.17	2.34	

small, invariance to e.g., perspective transformations are of little use. Moreover, as also claimed by Lowe [93], the additional complexity of full affine-invariant features often has a negative impact on their robustness and does not pay off, unless really large viewpoint changes are to be expected. In some cases even rotation invariance can be left out, resulting in only a scale-invariant version of the descriptor (e.g., as upright- or U-SURF).

Finally, one issue that is worth mentioning is the relationship between coverage of keypoints and a transformation between viewpoints. If the distribution of keypoints in an image is insufficient (either due to clustering or a low number of keypoints), effects due to the transformation between views may be missed or incorrectly reconstructed. A logical solution is to ensure a high amount of keypoints in the image, however, this can not always be guaranteed (e.g., due to a uniform image patch) or might be too computational intensive.

4.6 Summary

This chapter discussed the modelling of 3D vision. As such, the goal of this chapter is to determine the ideal keypoint detector and descriptor as well as to determine a method which transforms these image measurements into a useful 3D Cartesian error pose. The modelling of vision starts with a model of the camera and the lens. The pinhole camera model is presented and one method for camera calibration is discussed in more detail, i.e., Tsai's camera calibration method. The relationship between two views is analytically defined as the concept of projective transformations, also known as homographies. Such planar homography describes the displacement (translation and rotation) between a current view and a reference view, and can be reconstructed from two point sets of the same 3D points of a reference image. A mathematical definition and estimation of this homography is given and the decomposition into a translational and rotational part is derived in detail.

Following, an introduction towards keypoint detection is addressed, which includes a brief overview of ideal keypoints and existing keypoint detectors. The compared keypoint detectors can be divided as either corner detectors (i.e., Harris, Shi-Tomassi, FAST, AGAST) or as blob detectors (i.e., SIFT, SURF, CenSurE). Both methods exploit the scale-space representation of an image (i.e., at multiple resolutions (scales) detection operators are executed), which means that at the complete depth range keypoints will be found. A second differentiation is noted by the descriptive properties of a keypoint detector. Blob-detectors are typically designed with strong descriptive properties, which suggests a high computational load. When the application involves small motion between consecutive frames, weaker descriptors (and thus low computational load) are sufficient and detectors such as FAST or AGAST can be used. The application intended in this work involves large motion differences and, as such, strong descriptors are a necessity. Therefore, SIFT and SURF are experimentally compared from a computational and performance point of view with two different reference images. The results show that SURF is more suitable due to its computational advantages (i.e., robust keypoint detection with 70 matches in 140 $[ms]$ and thus an order of magnitude faster than SIFT) and the ability to tune between number of found keypoints and processing time. Finally, the subsequent processing to obtain a pose difference between current and reference image is analysed with respect to accuracy and repeatability. From this the following conclusions are drawn. The accuracy and repeatability of a rotation and translation estimation is highly affected by the clustering of keypoints. That is, if detected keypoints are not uniformly distributed over the reference image, this causes a bias on locally clustered keypoint patches. This effect can also be identified in cases where the reference object is partly outside the field of view or is located at a relatively large depth. In both cases the found images will not contain the similar amount of keypoints as detected in the reference image.

CHAPTER 5

Visual Control of Robotic Manipulators

Abstract. This chapter presents in detail the traditional approaches in vision-based robot control as well as a novel hybrid visual servoing approach. The proposed method combines traditional position-based visual servoing with image-based measurements to form a feedforward visual control law. The method is motivated to maintain objects in the field-of-view while designing motion in Cartesian space. Simulation and experimental results are presented and show the effectiveness of the novel approach.

5.1 Introduction

Visual servoing is defined as the motion control of a robot by means of visual feedback. Executed either in Cartesian space $\mathbb{R}^3$ or image space $\mathbb{R}^2$, the goal of vision-based control is to regulate a set of measured variables $\mathbf{f}(\mathbf{m}(t), \gamma)$ towards a set of desired variables $\mathbf{f}_d$:

$$\mathbf{f}(\mathbf{m}(t), \gamma) - \mathbf{f}_d = \mathbf{e} \rightarrow \mathbf{0}. \tag{5.1}$$

In this, $\mathbf{m}(t)$ represents a set of k measured features (i.e., for one 2D image point $k = 2$) and γ represents a set of parameters containing additional information (e.g., intrinsic camera parameters, 3D object model). The standard approach is to design a velocity controlled system and derive a relationship between the velocity of the measured variables $\dot{\mathbf{f}}$ and the velocity of the camera $\mathbf{v}_c$:

$$\dot{\mathbf{f}} = \mathbf{L}_\mathbf{e}\mathbf{v}_c, \tag{5.2}$$

where $\mathbf{L}_\mathbf{e} \in \mathbb{R}^{k\times 6}$ is referred to as the interaction matrix. A velocity control reference can then be derived by inverting (5.2) as:

$$\mathbf{v}_c = \mathbf{L}_\mathbf{e}^{-1}\dot{\mathbf{f}}, \tag{5.3}$$

assuming $\mathbf{L}_\mathbf{e}$ is square (i.e., when $k = 6$) and non-singular. Otherwise, when $\mathbf{L}_\mathbf{e}$ is of full rank 6, the Moore-Penrose pseudo-inverse $\mathbf{L}_\mathbf{e}^\dagger = (\mathbf{L}_\mathbf{e}^T\mathbf{L}_\mathbf{e})^{-1}\mathbf{L}_\mathbf{e}^T$ should be used [31]. As will be shown in the following chapters, this condition (i.e., $k \geq 6$), is ensured as follows. For position-based feedback, commonly, a 3D pose is measured or estimated, ensuring that $k = 6$. For image-based feedback, commonly $n > 3$ image points are measured, and, as one image points gives a 2D measurement, its is ensured that $k > 6$. From (5.1) it follows that $\dot{\mathbf{e}} = \dot{\mathbf{f}}$, which states that the feature error velocity is equal to the measured feature

velocity. Choosing an exponential decrease of the feature error (i.e., $\dot{\mathbf{e}} = -\lambda \mathbf{e}$, where λ is a positive gain factor) then results in

$$\mathbf{v}_c = -\lambda \mathbf{L}_\mathbf{e}^\dagger \mathbf{e}. \tag{5.4}$$

This control structure represents the general relationship between the velocity of the camera $\mathbf{v}_c$ and the error velocity of the observed features $\mathbf{e}$. A distinction between different visual control methods can be made by regarding different camera and robot architectures (e.g., camera location, type of image variables, see also Section 2.2), which is translated to the interaction matrix. In this chapter, the developments for visual control of robotic manipulators is limited to eye-in-hand configurations. That is, the camera is located on the end-effector of the manipulator.

The remaining contents of this chapter are as follows. A review on the basic approaches of visual servoing is presented, with a brief analysis of stability. Following, a novel approach is proposed which combines image-based and position-based visual servoing into one control law. This method is motivated to keep features in the field-of-view, while designing camera motion in Cartesian space. Finally, a stability analysis, as well as simulation and experimental results are presented.

5.2 Traditional Visual Servoing Approaches

An overview is made of a few existing visual servoing methods: Position based visual servoing (PBVS), image based visual servoing (IBVS) and a combination of both methods, also known as hybrid visual servoing. The methods discussed are known as indirect visual servoing, which implies that an outer control loop designs motion based on visual measurements, and a local joint control loop guarantees that the designed motion will be executed (see Fig. 5.1, Fig. 5.2 and Fig. 5.3). As this topic is already treated in great detail in literature, this overview follows largely the well-known tutorials [23, 24].

5.2.1 Position Based Visual Servoing

In accordance with the notation of (5.1), PBVS defines the feature vector $\mathbf{f}(\gamma)$ in Cartesian space between an initial pose $\mathbf{x}_I \in \mathbb{R}^6$ and a final pose $\mathbf{x}_f \in \mathbb{R}^6$ of the end-effector (see Fig. 5.1). As such, the feature vector $\mathbf{f}(\gamma)$ involves only intrinsic camera parameters and the 3-D model of the object.

The pose error $\mathbf{e}$ is defined as the difference between the two poses: $\mathbf{e} = [\mathbf{x}_f - \mathbf{x}_I]^T = [\mathbf{t_e},\ \theta\mathbf{u}]^T$, in which $\mathbf{t_e}$ is an error translation vector, and $\theta\mathbf{u}$ gives the angle/axis parametrization for the rotation error. One choice of $\mathbf{t_e}$ defines the translation error with respect to the camera frame $\mathcal{F}_c$ as $\mathbf{t_e} = \mathbf{t}_c - \mathbf{t}_{c,d}$, with $\mathbf{t}_{c,d} = 0$ and thus $\mathbf{t_e} = \mathbf{t}_c$. The interaction matrix, that relates the camera velocity and the error velocity as $\dot{\mathbf{e}} = \mathbf{L_e}\mathbf{v}_c$, can then be written as:

$$\mathbf{L_e} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}, \tag{5.5}$$

in which $\mathbf{R}$ represents the rotation matrix between current and desired frame

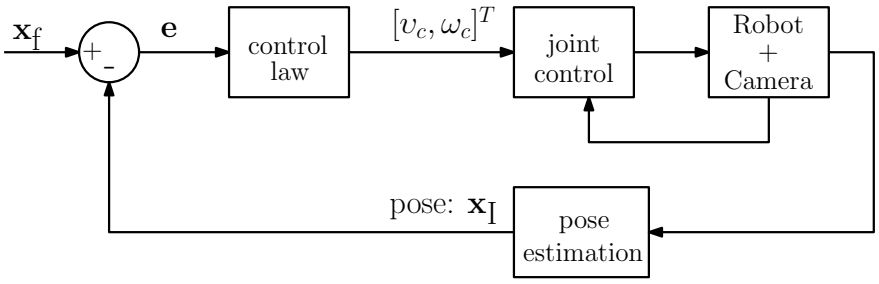


Figure 5.1: Block diagram of position-based visual control scheme (PBVS).

and $\mathbf{L}_{\theta\mathbf{u}}$ is defined as

$$\mathbf{L}_{\theta\mathbf{u}} = \mathbf{I}_3 - \frac{\theta}{2}[\mathbf{u}]_\times + \left(1 - \frac{\text{sinc}(\theta)}{\text{sinc}^2(\frac{\theta}{2})}\right)[\mathbf{u}]_\times^2, \tag{5.6}$$

where $\text{sinc}(\theta)$ is defined knowing that $\theta\text{sinc}(\theta) = \sin(\theta)$ and $\text{sinc}(0) = 1$ and $[\mathbf{u}]_\times$ is the skew symmetric matrix determined from vector $\mathbf{u}$. In this,

$$\begin{aligned} \theta &= \text{acos}\left(\frac{\text{trace}(\mathbf{R}) - 1}{2}\right), \text{ and} \\ \mathbf{u} &= \frac{1}{2\sin(\theta)}\begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix}. \end{aligned} \tag{5.7}$$

To obtain an exponential minimization of the error (i.e., $\dot{\mathbf{e}} = -\lambda\mathbf{e}$, where λ is a gain factor) the control law is set as

$$\mathbf{v}_c = -\lambda\widehat{\mathbf{L}_\mathbf{e}^{-1}}\mathbf{e}, \tag{5.8}$$

where $\widehat{\mathbf{L}_\mathbf{e}^{-1}}$ is an approximation of the real (inverted) interaction matrix $\mathbf{L}_\mathbf{e}^{-1}$ and is defined as

$$\widehat{\mathbf{L}_\mathbf{e}^{-1}} = \begin{bmatrix} \mathbf{R}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}}^{-1} \end{bmatrix}. \tag{5.9}$$

The camera velocities are then written as

$$\mathbf{v}_c = \begin{bmatrix} \boldsymbol{v}_c \\ \boldsymbol{\omega}_c \end{bmatrix} = \begin{bmatrix} -\lambda\mathbf{R}^T\mathbf{t_c} \\ -\lambda\theta\mathbf{u} \end{bmatrix}. \tag{5.10}$$

This PBVS approach lets the camera trajectory follow a straight line, while the image trajectory does not. The consequence is that configurations exist for which image features (i.e., the object) may leave the field-of-view.

A different choice of $\mathbf{t_e}$ defines the translation error with respect to the object frame $\mathcal{F}_o$ as $\mathbf{t_e} = \mathbf{t}_o - \mathbf{t}_{o,d}$. The interaction matrix, that relates the camera velocity and the error velocity as $\dot{\mathbf{e}} = \mathbf{L_e}\mathbf{v}_c$, can then be written as:

$$\mathbf{L_e} = \begin{bmatrix} -\mathbf{I}_3 & [\mathbf{t_o}]_\times \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}, \tag{5.11}$$

in which $\mathbf{I}_3$ is the 3×3 identity matrix, $[\mathbf{t_o}]_\times$ is the skew symmetric matrix determined from vector $\mathbf{t_o}$ and $\mathbf{L}_{\theta\mathbf{u}}$ is defined as (5.6).

The control law is set similar to (5.8), where the inverse of the interaction matrix is determined as

$$\widehat{\mathbf{L_e^{-1}}} = \begin{bmatrix} -\mathbf{I}_3 & [\mathbf{t_o}]_\times \mathbf{L}_{\theta\mathbf{u}}^{-1} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}}^{-1} \end{bmatrix}. \tag{5.12}$$

The camera velocities are then written as

$$\mathbf{v}_c = \begin{bmatrix} \boldsymbol{v}_c \\ \boldsymbol{\omega}_c \end{bmatrix} = \begin{bmatrix} -\lambda(\mathbf{t_e} + [\mathbf{t_o}]_\times \theta\mathbf{u}) \\ -\lambda\theta\mathbf{u} \end{bmatrix}. \tag{5.13}$$

This PBVS approach lets the translational and rotational motion decrease exponentially and lets the rotational motion follow a geodesic. Furthermore, the image trajectory follows a straight line, implying that the camera trajectory does not.

The stability of PBVS can be evaluated by considering Lyapunov's direct method [75]. Consider the squared error norm as candidate Lyapunov function: $\mathcal{L} = \frac{1}{2}\mathbf{e}^\mathrm{T}\mathbf{e}$. Using $\dot{\mathbf{e}} = \mathbf{L_e}\mathbf{v}_c$ and (5.8), the derivative is then written as

$$\dot{\mathcal{L}} = \mathbf{e}^\mathrm{T}\dot{\mathbf{e}} = -\lambda \mathbf{e}^\mathrm{T}\mathbf{L_e}\widehat{\mathbf{L_e^{-1}}}\mathbf{e}. \tag{5.14}$$

For global asymptotic stability the following condition should be satisfied:

$$\mathbf{L_e}\widehat{\mathbf{L_e^{-1}}} > 0. \tag{5.15}$$

The interaction matrix $\mathbf{L_e}$ is non-singular as long as $\theta \neq 2k_s\pi$ for $k_s \neq 0$. Practically speaking this ensures global convergence since the camera opening angle restricts θ as $|\theta| < 2\pi$. Furthermore, when the pose estimation is perfect, it follows that $\mathbf{L_e}\widehat{\mathbf{L_e^{-1}}} = \mathbf{I}_6$ and global asymptotic stability can be ensured [23].

5.2.2 Image Based Visual Servoing

In accordance with the notation of (5.1), IBVS defines the feature vector $\mathbf{f}(\mathbf{m}(t), \gamma)$ in image space and control is executed by minimizing the error between current and desired feature vector as defined by (5.1) (see Fig. 5.2). As such, $\mathbf{m}(t)$ denotes the image measurements (i.e., points, lines) and γ now only contains the camera intrinsic parameters. This means that measurements are taken directly from the image plane and used as feedback. In the most basic form these image measurements are a set of 2D feature points $\mathbf{p} = [u,\, v]^T \in \mathbb{R}^2$. Other image measurements are e.g., lines [42] or image moments [140].

The explicit derivation of the interaction matrix for point features can be found by differentiating (4.1) with respect to time and finding the relationship between the velocity of the feature point and the velocity of the end-effector as

$$\mathbf{L_e} = \begin{bmatrix} \frac{f}{z} & 0 & -\frac{u}{z} & -\frac{uv}{f} & \frac{f^2+u^2}{f} & -v \\ 0 & \frac{f}{z} & -\frac{v}{z} & \frac{-f^2-v^2}{f} & \frac{uv}{f} & u \end{bmatrix}, \tag{5.16}$$

where f is the focal length of the camera. Depending on the number of feature points n, this interaction matrix is then inverted according to (5.3) or (5.4).

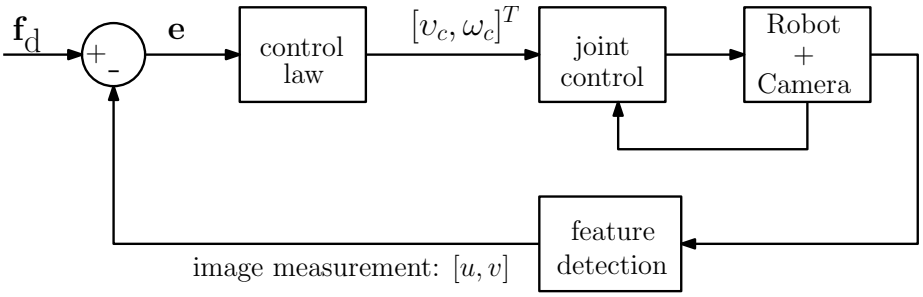


Figure 5.2: Block diagram of image-based visual control scheme (IBVS).

When considering multiple feature points, these can be stacked together in a composite point feature vector $\mathbf{f} = [\mathbf{p}_1^T, \ldots, \mathbf{p}_n^T]^T \in \mathbb{R}^{2n}$ and the interaction matrix can be formed as

$$\mathbf{L_e} = \begin{bmatrix} \mathbf{L_e}_{,1}(u_1, v_1, z_1) \\ \vdots \\ \mathbf{L_e}_{,n}(u_n, v_n, z_n) \end{bmatrix}. \tag{5.17}$$

One issue is the fact that the depth z of feature points is not directly measured and has to be estimated. This means that the estimated interaction matrix $\widehat{\mathbf{L_e}}$ must be used for feedback.

The stability analysis of IBVS follows the same method as for PBVS. With the Lyapunov function $\mathcal{L} = \frac{1}{2}\mathbf{e}^\mathrm{T}\mathbf{e}$ we obtain

$$\dot{\mathcal{L}} = \mathbf{e}^\mathrm{T}\dot{\mathbf{e}} = -\lambda \mathbf{e}^\mathrm{T}\mathbf{L_e}\widehat{\mathbf{L_e}^{-1}}\mathbf{e}. \tag{5.18}$$

Again, in order to ensure global asymptotic stability it has to hold that

$$\mathbf{L_e}\widehat{\mathbf{L_e}^{-1}} > 0. \tag{5.19}$$

To proceed, we consider the case when the number of image features is greater than three: $n > 3$. As $\mathbf{L_e}\widehat{\mathbf{L_e}^\dagger}$ can be at most rank 6, it has a non-trivial null space. Therefore, configurations such that $\mathbf{e} \in \mathcal{N}(\widehat{\mathbf{L_e}^\dagger})$ exist for IBVS with more than 3 image features. The physical consequence of this is that local minima exist for which the error $\mathbf{e}$ is nonzero (i.e., $\mathbf{f}(\mathbf{m}(t), \gamma) \neq \mathbf{f}_d$), while a zero velocity command is executed (i.e., $\mathbf{v}_c = \mathbf{0}$). In fact, when $n = 3$, four distinct and undistinguishable camera poses exist for which $\mathbf{e} = \mathbf{0}$ [102].

Because of this issue, usually more than three feature points are used. Furthermore, it can only be proven that in some neighbourhood of $\mathbf{e} = \mathbf{0}$, no local minima can be encountered. The conclusion is that for IBVS only local asymptotic stability can be ensured.

5.2.3 Hybrid/Partitioned Approaches

Hybrid visual servoing entails that both previous methods (IBVS and PBVS) are combined into one new visual control law [40, 48], meaning that the feature vector $\mathbf{f}$ now contains a mixture of 2D image information and 3D Cartesian information.

2-½D Visual Servoing

A typical hybrid method, known as 2-1/2D visual servoing, decouples the rotational DOF from the translational DOF. The rotational part is then controlled by PBVS and the translational part is controlled by IBVS. As such, the algorithm takes advantage of the robustness properties of IBVS and the stability properties of PBVS. This is therefore the reasoning behind its name; a combination of control in 2D image space and 3D Cartesian space.

Let $\mathbf{f}(\mathbf{m}(t), \gamma) = [\mathbf{p}^T, \log(z), \theta\mathbf{u}]^T \in \mathbb{R}^6$ be the feature vector, where $\theta\mathbf{u}$ is again the parametrization of the rotation error, $\mathbf{p} = [u,\ v]$ the image measurement in pixels and z the associated depth. The error $\mathbf{e} = \mathbf{f}(\mathbf{m}(t), \gamma) - \mathbf{f}_d$ can then be written as $\mathbf{e} = [\mathbf{p} - \mathbf{p}_d^T, \log(z/z_d), \theta\mathbf{u}]^T$ and the interaction matrix is found as

$$\mathbf{L_e} = \begin{bmatrix} \mathbf{L_v} & \mathbf{L}_\omega \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}, \tag{5.20}$$

where

$$\mathbf{L_v} = \frac{1}{z_d(z/z_d)} \begin{bmatrix} -1 & 0 & u \\ 0 & -1 & v \\ 0 & 0 & -1 \end{bmatrix}, \text{ and } \mathbf{L}_\omega = \begin{bmatrix} uv & -(1+u^2) & v \\ 1+v^2 & -uv & -u \\ -v & u & 0 \end{bmatrix}, \tag{5.21}$$

in which z/z_d is defined as $\det \mathbf{H}$; the determinant of the homography matrix and $\mathbf{L}_{\theta\mathbf{u}}$ is given by (5.6). As the depth parameter z_d is not directly measured, this has to be estimated or can be adapted online, for which methods can be found in [37] and [95].

The stability conditions of 2-1/2D visual servoing can be evaluated quite straightforward. The fact that $\mathbf{L_e}$ is a 6×6 upper triangular matrix, global asymptotic stability can be proven when ideal conditions apply (i.e., perfect pose estimation) [98].

Kyrki's method

A hybrid approach similar to 2-1/2D visual servoing proposed by Kyrki et al. in [86] controls 2 rotational degrees of freedom (i.e., $\mathbf{R}_x$ and $\mathbf{R}_y$) purely from image data. The error vector for visual control is defined as $\mathbf{e} = \mathbf{L_e}\mathbf{v}_c = \left[\mathbf{t}_e,\ t_{e,x},\ t_{e,y},\ \theta u_z\right]^T$, where $\mathbf{L_e}$ is defined as

$$\mathbf{L_e} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{L}_{ib} & \\ \mathbf{l}_r & \end{bmatrix}, \tag{5.22}$$

in which $\mathbf{I}_3$ is 3×3 identity matrix, $\mathbf{0}_3$ is a 3×3 zero matrix, $\mathbf{L}_{ib}$ is identical to (5.16) and $\mathbf{l}_r = [0,\ 0,\ 0,\ 0,\ 0,\ 1]$. This hybrid method is motivated by guaranteeing both a shortest Cartesian trajectory and object visibility. Evaluating the

determinant of the interaction matrix (in terms of the object position) shows that one degenerate case exists [86]:

$$\det \mathbf{L_e} = \frac{x^2 + y^2 + z^2}{z^2}, \tag{5.23}$$

which is zero when all x, y and z are zero, that is, when the camera is exactly at the origin of the object. Fortunately, this configuration is not physically possible.

5.2.4 Comparison of Traditional Methods

The differences between the traditional methods in visual servo control can be easily explained from their method of control. The obvious fact that for PBVS the control is executed in Cartesian space results in a motion that is favourable in Cartesian space. Similar conclusions can be drawn from IBVS, where the image trajectory is a pure straight line and the camera motion is not taken into account. Besides these obvious differences, the robustness properties of both methods against noise is another point of interest. The fact that IBVS methods are more robust against image noise and camera calibration errors makes it particularly attractive for practical implementation. A downside, however, is the existence of local minima with large camera motions or singularities in the interaction matrix.

For hybrid methods the motion is mixed due to the design of the control in both image and Cartesian space. In particular, considering 2-1/2D visual servoing [99, 98], the translational motion is controlled by measurements in image space and the rotational motion is controlled by position-based measurements. As such, this decoupled method takes advantage of the robustness properties of IBVS and the stability properties of PBVS. The hybrid method developed by Kyrki et al. [86] controls the translational motion with PBVS and two rotational DOFs (pan and tilt) with IBVS. The remaining rotational DOF is also controlled with PBVS. The resulting motion has the property to maintain objects in the field of view. This, however, is also the drawback of the approach, as a different desired rotational motion becomes particularly difficult to design.

This last conclusion can be drawn for most image-based methods. When an image-based approach is solely used for control, the design of motion is limited to the image space. As this space practically only consists of two dimensions and can not directly be transformed to the Cartesian space, the resulting motion will be relatively limited.

Consider for instance the task, where an object should be kept in the field of view, while motion in Cartesian space should be deterministic (or at least a straight line in translation). The traditional methods (IBVS and PBVS) both do not fulfil these requirements. In short PBVS does design shortest paths, however, image features might leave the field of view, while IBVS disregards Cartesian space completely. The hybrid methods (2-1/2D visual servoing and Kyrki's method) do not suffice as well. For 2-1/2D visual servoing, as the translation motion is controlled by image data, trajectories in Cartesian space may not be designed as a shortest path. On the other hand, while Kyrki's method should give appropriate results, the fact that no Cartesian offset can be added to the pan- and tilt DOFs, motion design is fairly limited.

This analysis therefore gives rise to the development of a novel method that takes these requirements as main objective: a Cartesian intuitive motion design while guaranteeing the field-of-view constraint.

5.3 Feedforward Visual Servoing

As discussed in Section 5.2 both image-based and position-based visual servoing has its positive and negative properties. In short, image-based feedback lacks the design of motion in Cartesian space, while position-based feedback has no control over the trajectories in image space. Combining both methods in one approach can therefore result in an improved performance as seen by hybrid or partitioned approaches. In this section a method is proposed that merges an image-based feedforward control action with position-based visual servoing.

Similar to encoder-based motion systems, a feedforward control action is commonly applied when disturbances are present which are known beforehand (e.g., gravity, friction) and can therefore be compensated for. Extending this concept to the visual domain will result in a similar compensation response. More specifically, if a 2D image-based feedforward action is added to rotational control of PBVS, this will maintain objects in the field-of-view. This approach is developed in more detail in forthcoming section.

5.3.1 Field-of-View Constraint

The constraint of maintaining features in the field-of-view can be defined as a rectangular bound of the image sensor, which, for one image feature point $\mathbf{p} = [u,\ v]^T \in \mathbb{R}^2$, is stated as

$$\begin{aligned} u &\in [u_{min}, u_{max}], \quad u_{min}, u_{max} \in \mathbb{R}, \\ v &\in [v_{min}, v_{max}], \quad v_{min}, v_{max} \in \mathbb{R}. \end{aligned} \tag{5.24}$$

As long as the feature point $\mathbf{p}$ stays within this bound the constraint is satisfied. Fig. 5.3 shows the block diagram of this control scheme. In this, $\mathbf{x}_t$ is the target pose, which changes with every visual update, depending on the current pose $\mathbf{x}_c$. Together with an image-based angle towards the target $\hat{\mathbf{e}}_\theta$, $\mathbf{x}_t$ is changed into a reference pose $\mathbf{x}_r$, which is used for PBVS.

5.3.2 Image-Based Feedforward

2D image measurements can be obtained as a single point or a set of points, from which a mean point is computed. Considering a single point, basic image processing algorithms can be employed to obtain robust measurements (e.g., color blob detection, circle detection). A more complex approach employs advanced image processing algorithms such as SIFT or SURF for feature point detection (see Section 4.4). Although fairly computational intensive, this approach also enables a reliable computation of the pose error (see Section 4.3), which can also be used for vision-based control. The latter is the method of choice for our hybrid visual servoing approach where SURF features are used

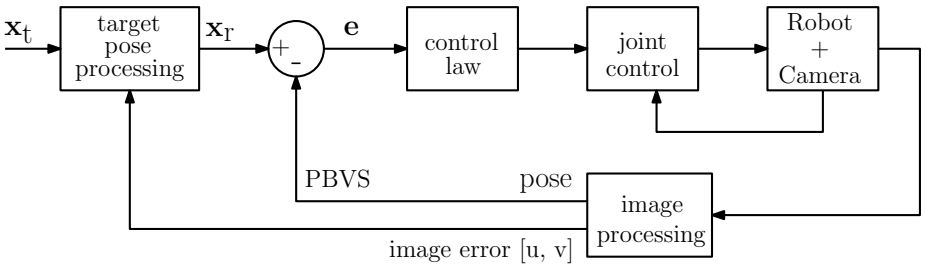


Figure 5.3: Block diagram of proposed visual control scheme (FF PBVS).

for both image- and position-based feedback. The object as used for reference is highly textured and planar (see Fig. 4.4 and Fig. 4.7 in Section 4.5).
For the image based error, the mean of a set of n object points $\bar{\mathbf{p}}_o$ is defined as

$$\bar{\mathbf{p}}_o = \frac{1}{n}\sum_{i=0}^{n} \mathbf{p}_{o,i}, \tag{5.25}$$

where $\mathbf{p}_{o,i} = [u_{o,i}, v_{o,i}]^\mathrm{T}$.

As the image-based error is combined (i.e., added) with a position-based control law, the image point has to be transformed to Cartesian space. More precisely, the 2D image-based error is used as a 2D rotation feedforward term.

The angle relative towards the target $\mathbf{e}_\theta = [\theta_x, \theta_y]^\mathrm{T}$ is estimated as the difference in image coordinates from the mean of the object points $\bar{\mathbf{p}}_o$ and the center of the image $\mathbf{p}_0 = [0, 0]^\mathrm{T}$ as $\hat{\mathbf{e}}_\theta = [\hat{\theta}_x, \hat{\theta}_y]^\mathrm{T} = [\bar{\mathbf{p}}_o - \mathbf{p}_0]$, since it is assumed that $\hat{\mathbf{e}}_\theta \propto \mathbf{e}_\theta$ for the camera's FOV. Therefore, the 2D rotation term is bounded by a hyperbolic tangent function defined as

$$\begin{aligned} e_{\theta,x} &= \hat{\theta}_x \pm \theta_{x,f}, \text{ where } \hat{\theta}_x = c_{\theta,x} \tanh(c_{o,u}\bar{p}_{o,u}), \\ e_{\theta,y} &= \hat{\theta}_y \pm \theta_{y,f}, \text{ where } \hat{\theta}_y = c_{\theta,y} \tanh(c_{o,v}\bar{p}_{o,v}), \end{aligned} \tag{5.26}$$

where $c_{\theta,x}$ and $c_{\theta,y}$ are constant scalars that limit $\hat{\theta}_x$ and $\hat{\theta}_y$, since $\lim_{x\to\infty} \tanh = 1$, and $\lim_{x\to-\infty} \tanh = -1$. Furthermore, $c_{o,u}$ and $c_{o,v}$ are constant scalars tuned such that $\hat{\mathbf{e}}_\theta \approx \mathbf{e}_\theta$. The fixed angles $\theta_{x,f}$ and $\theta_{y,f}$ are computed from the difference between current pose $\mathbf{x}_c$ and the target pose $\mathbf{x}_t$ by simple trigonometry. If only fixed angles where given, the control law would essentially be pure PBVS. With the estimated angles $\hat{\mathbf{e}}_\theta$, a control system is created that acts as a feedforward term on the reference pose. This creates an overshoot of the reference pose which dies out due to the decrease of the image error.

5.3.3 Feedforward and Position-Based Visual Servoing

The combination of the feedforward control action with PBVS is developed as follows. For position-based visual servoing, the control scheme is chosen with the translation which designs a straight line in Cartesian space. The camera velocities which achieve this are defined as (5.8) or

$$\mathbf{v}_c = -\lambda \widehat{\mathbf{L}_\mathbf{e}^{-1}} \mathbf{e}, \tag{5.27}$$

where the interaction matrix is defined as in (5.9) and $\mathbf{e} = [\mathbf{t}_e,\ \theta\mathbf{u}]^T = [\mathbf{e}_\nu,\ \mathbf{e}_\omega]^T$. The components for translational control are unaltered.

As stated earlier, the fact that the image trajectory will not be a straight line can cause the camera to lose the image features. This loss of features is now ensured by adapting the rotational motion $\theta\mathbf{u}$ as:

$$\theta\mathbf{u} = \begin{bmatrix} \hat{\theta}_x \pm \theta_{x,f} \\ \hat{\theta}_y \pm \theta_{y,f} \\ \theta u_z \end{bmatrix}. \tag{5.28}$$

The details of the individual entries can be found from (5.26).

This results in a rotation reference trajectory $\mathbf{e}_\omega = [e_{\theta,x},\ e_{\theta,y},\ \theta u_z]^T = \theta\mathbf{u}$ that can oscillate and not necessarily points towards the object. The camera's normal vector $\mathbf{n}$, however, does point towards the object at all times, enabling a continuous fixation on the object for recognition or exploration.

Concluding, as the core control method is position based visual servoing, the interaction matrix for control is therefore similar as presented in Section 5.2.1. To ensure the field-of-view constraint, only the reference for rotational control is altered to include an image-based feedforward term.

5.3.4 Stability Analysis

Consider the squared error norm as candidate Lyapunov function: $\mathcal{L} = \frac{1}{2}\mathbf{e}^{\mathrm{T}}\mathbf{e}$. Using $\dot{\mathbf{e}} = \mathbf{L}_\mathbf{e}\mathbf{v}_c$ and (5.8), the derivative becomes

$$\begin{aligned} \dot{\mathcal{L}} &= \mathbf{e}^{\mathrm{T}}\dot{\mathbf{e}}, \\ &= -\lambda\mathbf{e}^{\mathrm{T}}\mathbf{L}_\mathbf{e}\widehat{\mathbf{L}_\mathbf{e}^{-1}}\mathbf{e}. \end{aligned} \tag{5.29}$$

In this, the error vector $\mathbf{e}$ is expressed as $\mathbf{e} = [\mathbf{t_h},\ \theta\mathbf{u}]^{\mathrm{T}}$, where $\theta\mathbf{u} = [\hat{\theta}_x \pm \theta_{x,f},\ \hat{\theta}_y \pm \theta_{y,f},\ \theta u_z]^{\mathrm{T}}$. Although $\mathbf{e}$ is different from normal PBVS and $\mathbf{L}_{\theta\mathbf{u}}$ and therefore $\mathbf{L}_\mathbf{e}$ is incorporated differently due to an added estimated rotation, still the stability proof for traditional PBVS can be used. Although already restricted by the opening angle of the camera, it is enforced that

$$|\hat{\theta}_x| + |\theta_{x,f}| < 2\pi, \quad \text{and} \quad |\hat{\theta}_y| + |\theta_{y,f}| < 2\pi, \tag{5.30}$$

which gives a singularity in $\mathbf{L}_{\theta\mathbf{u}}$. For global asymptotic stability the following condition should be satisfied:

$$\mathbf{L}_\mathbf{e}\widehat{\mathbf{L}_\mathbf{e}^{-1}} > 0. \tag{5.31}$$

If $\widehat{\mathbf{L}_\mathbf{e}^{-1}} = \mathbf{L}_\mathbf{e}^{-1}$, meaning the pose estimation is perfect, it follows that $\mathbf{L}_\mathbf{e}\widehat{\mathbf{L}_\mathbf{e}^{-1}} = \mathbf{I}_6$, and global asymptotic stability can be ensured [23].

5.4 Simulation and Experimental Results

Simulations are carried out by using the Robotics [32] and the Epipiolar Geometry [100] Toolboxes for Matlab. To simulate visual feature detection, a set of 30 random points is generated, from which two views are created with a perspective transformation. These two perspective point sets are then input to the

homography estimation and decomposition, which determines a rotation and translation difference for control. Normally distributed random noise is added to the points with zero mean and 5% standard deviation.

5.4.1 Experimental Setup

Experimental results are obtained with a 7-DOF redundant manipulator, where the camera is located on the end-effector (eye-in-hand, see Fig. 5.4). The implementation of the direct and differential kinematics, as well as the homogeneous solution for redundancy, are translated to and optimized for `C/C++`, and implemented using the Eigen library for vector and matrix manipulation. This is wrapped inside a ROS [117] node for high-level functionality and low-level device control. As all processing is executed on a standard notebook, communication with the manipulator is done via a CAN-USB device, with different threads managing the data exchange with the CAN device (one for reading and one for writing). As is typical for vision-based control, a low-level PD joint controller is executed for each joint, with an update rate of 1 $[kHz]$. More details on the manipulator, the modelling and the implementation can be found in Section 8.3.1 and Appendix B.

For visual processing, an industrial camera (Prosilica GE680M, communicating via Gigabit Ethernet) takes grayscale images which are processed using the computer vision library Opencv [14]. The SURF feature detector as presented in Section 4.4.3 and subsequent homography estimation and decomposition as presented in Section 4.3.1 and Section 4.3.2 provides a rotation and scaled translation between two views. These detected features are also used to estimate the image-based rotation measurement for the feedforward control action. This vision algorithm is executed at 10 Hz with an image size of 640×480 $[px]$ (VGA).

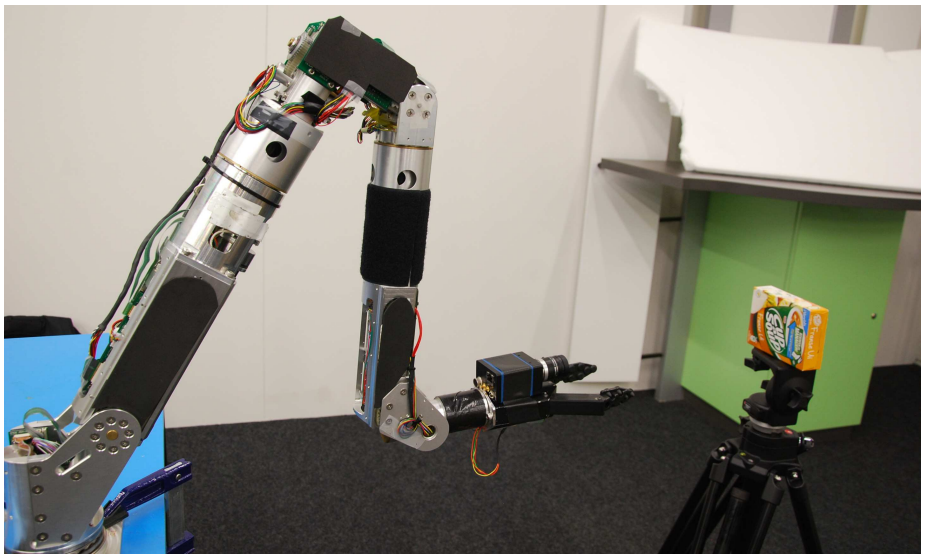


Figure 5.4: 7-DOF redundant robotic manipulator with eye-in-hand camera. The object as used for visual control and exploration can be seen on the right.

5.4.2 Task Definition

The defined task is divided in three steps: the first step positions the camera and manipulator orthogonally and centred in front of the object. The second and third step are defined as a translation offset in Cartesian space, respectively to the front-left and front-right of the object. This translation can be freely chosen; in simulation and experimental setting a sideways translation of 0.2 $[m]$ and forward translation of 0.1 $[m]$ was chosen (Fig. 5.5).

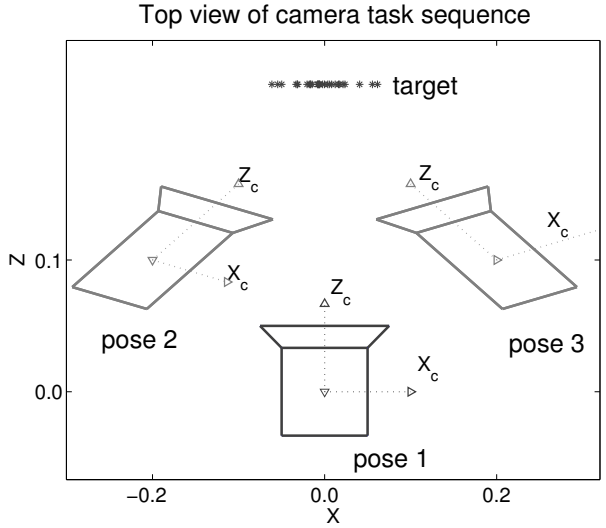


Figure 5.5: Task sequence for comparison of visual control algorithms and exploration motion. Left and right translations can be chosen freely.

The first pose is also used as safety; if a direct motion from pose 2 to pose 3 is defined, this may lead to unwanted or unstable behaviour. The camera may come too close to the object, and thus the easiest solution is therefore to employ an extra via-point. With the object taken as center, the camera covers roughly 60 degrees (i.e., $\sim$1 $[rad]$) with respect to the center of the image plane. Taking into account the opening angle of the camera itself, this covers a large part of the object for exploration.

5.4.3 Simulation Results

In simulation the different control methods are evaluated with the same exploration task. Each iteration the visual control law is executed and it is assumed that the robot follows this desired motion perfectly. This assumption (i.e., visual loop runs at a similar rate as the local control loop) is obviously not possible in a real world situation. Fig. 5.6 compares the references for PBVS and the proposed method. For position based visual servoing, the reference is a constant offset. The proposed method employs the same reference with the extra feature that a term is added that takes the image error into account. This can be seen in the figure as a continuously changing reference, with an overshoot at

the start of every new pose, to obtain the best centred view. This feedforward action dies out when the image error goes to zero.

Fig. 5.7 shows a comparison between different control solutions. Our method (FF PBVS), is compared to traditional PBVS and Kyrki's method [85]. The comparison is evaluated with the yaw angle error of the camera, which executes the largest motion. It is shown that our proposed method has equal or better performance (in terms of error) compared to all others. Although PBVS gives a decent performance, there is no guarantee that the object remains in the field-of-view.

5.4.4 Experimental Results

For an experimental comparison, the same task is executed. From first observations of the experimental results (Fig. 5.9, and Fig. 5.8) it can be clearly seen that vision is only updated every few iterations. This leads to a step-wise profile as reference, however, the real motion of the robot is smooth. A second difference between simulations and experiments is that due to external disturbances (e.g., friction in joints, gravity), the feedforward reference term of our method does not completely die out towards the reference term of PBVS.

Fig. 5.8 compares the reference poses for the yaw angle of the different methods. PBVS (dashed-dot) has a constant reference, and Kyrki's method (dashed) has a reference only based on the image error. The proposed method (solid) shows a bigger yaw angle reference, due to the feedforward term, which dies out when the image error goes to zero. A rough analysis also reveals that when the PBVS reference and the reference of Kyrki's method would be combined, the reference of the proposed method would be obtained. This corresponds with the basic concept of the proposed method.

Fig. 5.9 compares the proposed method (FF PBVS, solid) with Kyrki's method (dashed) for the yaw angle. The proposed method has a clear greater range of motion for exploration around an object. Since in Kyrki's method only an image error defines the range of motion, external disturbances (friction in joints, noisy measurements) have a great impact. Due to the fact that the proposed method uses a feedforward term combined with a position based reference, the effect of these disturbances is overcome. Moreover, the measured motion (red lines) reveals that the executed motion is smooth, despite the stepwise input due to the slow update rate of the vision sensor.

Fig. 5.10 shows the error in image space (u- or x-axis) for moving from pose 1 to pose 2 and back. The response of the proposed method (solid) compared to Kyrki's method (dashed-dot) can be considered comparable or better. PBVS however, has a clear worse performance as it does not execute the same error decrease. Moreover, with PBVS it may occur that the object leaves the field-of-view. The large error of Kyrki's method can be explained from the fact that rotational control only relies on an image error. This is not sufficient to compensate for large disturbances (e.g., friction, image noise).

These results correspond with the traditional vision-based control methods. IBVS designs a straight line in image space and does not consider Cartesian space. Kyrki's method proves this, as the reference and the motion itself in Cartesian space for the yaw angle (Fig. 5.9 and Fig. 5.8) are minimal compared to PBVS and FF PBVS. On the other hand, PBVS designs a straight line in Cartesian space and does not consider image space. This is shown in Fig. 5.10

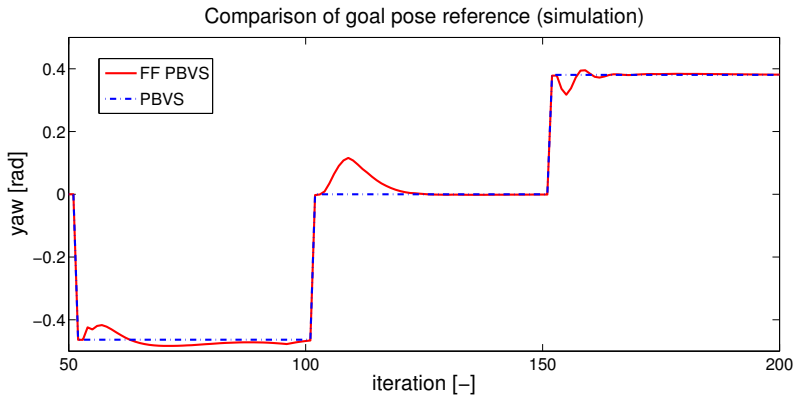


Figure 5.6: Reference pose comparison between the proposed method (FF PBVS) and PBVS in simulation. It is shown that PBVS executes a constant reference. The reference pose of our method changes continuously and has an overshoot due to the feedforward term which dies out when the image error goes to zero.

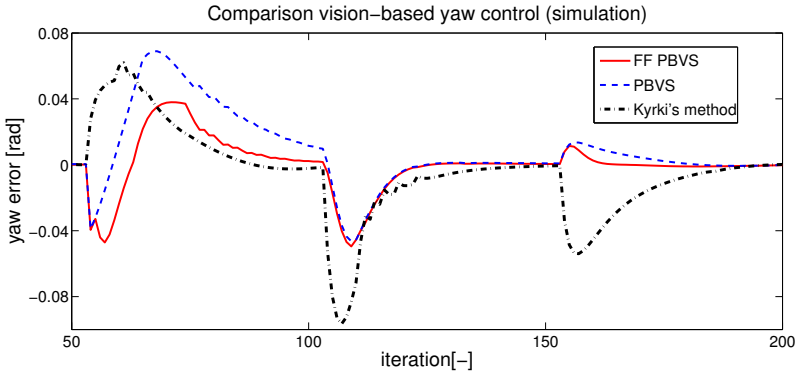


Figure 5.7: Performance comparison (in terms of error) between the proposed method (FF PBVS), PBVS and Kyrki's method [85] in simulation. The overall performance of the proposed method (FF PBVS) can be considered equal or better than the others.

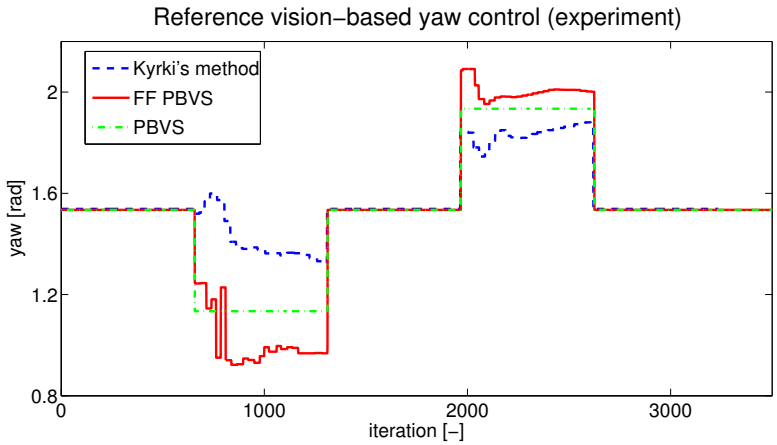


Figure 5.8: Experimental comparison of reference trajectories. PBVS executes a constant reference. Kyrki's method designs a trajectory only based on image error data, which causes disturbances (e.g., friction, noise) to have a great impact (i.e., small range of motion). The proposed method (FF PBVS) designs motion with PBVS and a feedforward term obtained from image-based measurements, and as such, achieves a greater range of motion.

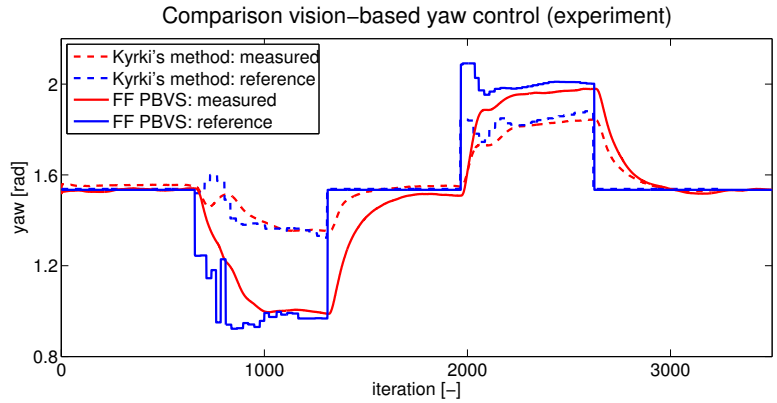


Figure 5.9: Experimental comparison of yaw angle control between the proposed method (FF PBVS) and Kyrki's method. Due to the feedforward scheme, the proposed method has a clear greater range for exploring an object. Despite the step-wise input due to a low update rate of the vision sensor, the measured motion is smooth.

Figure 5.10: Image error response for motion from pose 1 to pose 2 and back. The response of the proposed method (FF PBVS) compared to Kyrki's method can be considered comparable or better. PBVS however, does not consider image error data and thus performs worse.

where PBVS does not achieve an exponential decrease nor reaches zero error in a similar time span. Concluding, the proposed method takes advantage of both PBVS and Kyrki's method and shows smooth motion in Cartesian space while maintaining the field-of-view constraint (see Fig. 5.11).

5.5 Summary

This chapter presented several visual control algorithms in more detail and elaborates on their advantages and disadvantages. The analysis includes the traditional approaches (i.e., image-based and position-based visual servoing) as well as several hybrid methods. A general comparison concludes that for position-based visual servoing motion is designed in Cartesian space, while for image-based control motion is designed in image space. Hybrid methods are designed to take advantage of both. For example, the method designed by Kyrki et al. designs a shortest path in Cartesian space while guaranteeing object visibility. Following, a novel approach is proposed that eliminates the short-comings of the traditional as well as the hybrid approaches. In particular, if object visibility would only be guaranteed by an image error, disturbances typical for motion control systems (i.e., friction, gravity) could play a large role. This then result in motion which has a fairly limited range as is shown in experimental setting. The proposed method overcomes these issues by combining position-based visual servoing with a rotational image-based feedforward. This effectively ensures the field-of-view constraint and adds a greater range of motion for e.g. exploration around an object. It is shown that by definition the stability properties of the proposed method are similar to the stability properties of traditional position-based visual servoing. Simulations and experiments are carried out which show that the proposed method results in equal or better performance compared to existing methods.

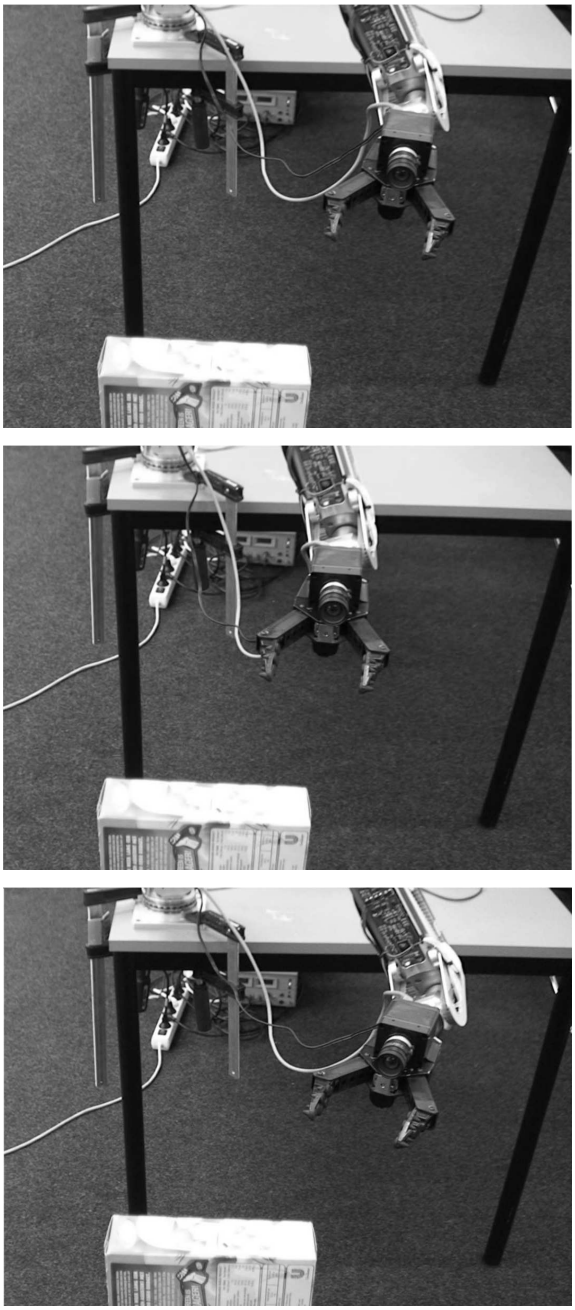


Figure 5.11: End of pose 2 (i.e., ∼iteration 46 in Fig. 5.10), of exploration motion using different visual control laws. Upper figure shows PBVS, where the object is not kept in the field of view. Middle figure shows Kyrki's method, which due to the use of only an image error for rotation, is highly disturbed by friction and results in a small range of motion. Lower figure shows the proposed method, which combines PBVS and an image feedforward error and achieves a greater range of motion (than the other methods) while keeping the object in the field of view.

CHAPTER 6

Direct Trajectory Generation for Vision-Based Control

Abstract. This chapter discusses the topic of direct trajectory generation for vision-based robot control. The proposed method is based on a combination of traditional trajectory generation and vision-based control. As such, direct trajectory generation can incorporate changes of constraints online as obtained by visual measurements. The method is explained and analysed in detail, its properties are discussed accordingly and experimental results are presented for a single degree of freedom system.

6.1 Introduction

Trajectory generation is one of the most basic topics in robotic motion control. Much research has been carried out (for an introduction see e.g., [11]) and due to its proven history, commonly the focus of research in control systems lies more on the improvement and development of controllers rather than focussing on the generation of motion itself. For the generation of such trajectories many different elementary functions can be used. Examples include trigonometric, exponential or polynomial functions. Common trajectories for industrial robots are e.g., linear segments with parabolic blends (LSPB), cubic splines for multi-point trajectories or B-splines for trajectories with higher degree of continuity. The developments of these methods can be found in [11]. Due to the simplicity of incorporating constraints on a local level (i.e., on individual points and their time derivatives) as well as on a global level (i.e., constraints on the complete trajectory and its time derivatives), the method of polynomial trajectory generation is used in this work.

A recent advancement, which has been gaining interest over the last decade, is on the topic of online generation of motion trajectories (see Section 2.6 and e.g., [82], [106], [2], [142]). The general idea is that by focussing more effort on the design of appropriate trajectories, controller design can become effectively less demanding. This is motivated by the fact that if constraints of a trajectory change online, offline planning is not suitable to handle this when the trajectory is already being executed. Before a new motion trajectory can be started, the current trajectory has to be finished. An abrupt change from an executed trajectory to a new trajectory without taking into account the current state will cause discontinuities in motion and will therefore lead to difficulties in motion control (e.g., vibrations, wear, large error, etc.). If a trajectory can be altered directly after a change is detected, this will result in a better performance.

In order to provide a solution to this motion and trajectory planning problem, this chapter proposes an extension to trajectory generation by incorporating a direct and online method for constrained motion planning.

6.1.1 Vision-Based versus Offline Motion Planning

Even-though the concept of both sensor-based planning and offline motion planning is treated in detail in Chapter 5 and Section 3.5 respectively, a short summary is given stating their advantages and disadvantages.

Offline motion planning designs a trajectory before any motion is executed. This trajectory can not be changed at runtime, however, constraints on the trajectory can be easily considered. A common procedure is to execute multiple trajectories successively, where subsequent trajectories can account for changes in constraints. This implies that while executing motion, the system is blind to any changes. Vision-based motion planning considers the motion of a system to be dependent on the sensor at hand. This means that motion is directly modified based on the (visual) measurements of the sensor. The design of this motion is usually highly simplified as incorporation of sudden events is fairly complex or too time-consuming. Examples include only error minimization (i.e., no kinematic constraints) for vision-based control or the planning of a path instead of a trajectory for obstacle avoidance.

Moreover, a further difference between both traditional methods can be identified in its execution time. Traditional offline motion planning defines a single control structure known as trajectory tracking, which can be executed at a fairly high rate (e.g., 1 $[kHz]$). On the other hand, vision-based control requires more processing time to compute a motion command. This gives rise to a local control loop to guarantee stability (i.e., ensuring a motion command is reached) and a global loop that computes the motion command (see e.g., the traditional visual servo approaches in Section 5.2). In particular, for vision-based control it holds that

$$T_v > T_l, \tag{6.1}$$

where T_v, the visual update time, commonly lies in the range of tens of milliseconds (e.g., 20 $[Hz]$ or 50 $[ms]$), and T_l, the local control update time, commonly does not exceed one millisecond. A path is then defined by a visually processed position error and set to be minimized: $\dot{\mathbf{e}} = -\lambda \mathbf{e}$, with λ a positive constant. Combined with the interaction matrix $\mathbf{L_e}$, which relates the image feature velocities to the velocities of the camera, a velocity controlled system is achieved which executes an exponential decrease in error as motion: $\mathbf{v}_c = -\lambda \mathbf{L_e}^{-1}\mathbf{e}$ (see Fig. 6.1 and Section 5.1). This velocity input can lead to non-smooth or undesirable robot motion. In particular, the initial image error (at $t = 0$) acts as a step-function (i.e., a discontinuity) for the velocity signal, which in turn implies an infinite acceleration. Moreover, any constraints on motion (i.e., spatial, kinematic or dynamic) are not directly included. Furthermore, missing, noisy or delayed measurements have to be dealt with by e.g., a state observer (which estimates the state of the next step based on current and past information), otherwise instability of the system may occur.

Closer inspection suggests that if both approaches could be adapted into one, the advantages of both could account for an improved motion design. This approach fits perfectly in a motion control scheme where direct reactions

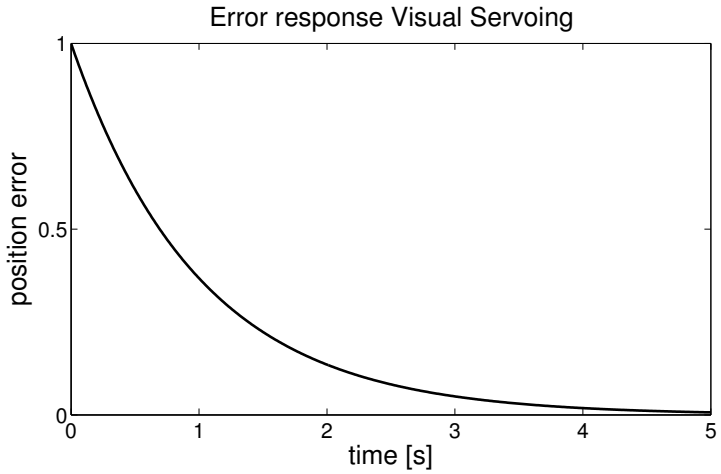


Figure 6.1: Traditional visual servoing executes motion with an exponential error decrease. This error is the velocity input to the system, and as such, serves as a path, which leads to an inherently non-smooth and unconstrained motion profile.

to sensor readings are eminent. More specifically, the approach of direct trajectory generation could serve as solution to the problem of obstacle avoidance. Where path planning would direct an avoidance procedure merely on path planning level, the direct trajectory generation method considers the avoidance procedure on trajectory planning level and, as such, can incorporate motion (i.e., kinematic) constraints online.

6.2 Direct Trajectory Generation

The general idea of direct trajectory generation is that each iteration (k) a new motion profile for the next iteration ($k+1$) is made, depending on the current constraints $\mathbf{q}_c$, the current state $\mathbf{S}_k$ and the current trajectory synchronization times $\mathbf{t}_s$. As initial conditions, certain choices have to be made regarding the type of trajectory and the constraints. These are listed as follows:

- Trajectory type
  Depending on the task at hand, it has to be specified how the overall shape of the trajectory (and its time derivatives) should be designed. In particular, the degree of continuity $\mathcal{C}^{n_p}$ and whether the trajectory includes constraints on (via)-points and their time derivatives has to be decided beforehand. These choices can be easily incorporated in a polynomial trajectory by simply including or omitting constraint equations and by expanding or reducing the polynomial function. The order of the polynomial, as well as the complexity of trajectory generation is therefore determined by these choices.

- Trajectory constraints
  The motion constraint vector $\mathbf{q}_c$ of a trajectory specifies the position constraints and its time derivatives on (via-)points (see also Section 3.5.2) as:

$$\mathbf{q}_c = \begin{bmatrix} q_I & q_v & q_f & v_I & \alpha_I & v_f & \alpha_f \end{bmatrix}, \tag{6.2}$$

  for a 6^{th} order polynomial trajectory (i.e., with 3 points). This includes position, velocity and acceleration constraints on the initial- and final-point and only a position constraint on the via-point. For a 5^{th} order polynomial trajectory (i.e., point-to-point), the via-point constraint q_v would be omitted.

- Trajectory timing
  The execution time t_e of each trajectory, depends on the limits of the task and the system, and defines the synchronization time t_s of the complete motion. These limits (e.g., maximum velocity or acceleration) have to be predefined.

The type of trajectory determines the complexity of trajectory generation and is directly related to the number of (via-)points and the number of constraints. For instance, a trajectory can be designed with only start- and end-point, or can be composed of several via-points. Alternatively, a motion profile can consist out of several, smaller trajectories, each with different complexity. Together with the local constraints on these points, the complexity of the complete trajectory can easily grow relatively large. A trajectory with $\mathcal{C}^2$ continuity and three points with equal number of constraints (i.e., a position, velocity and acceleration constraint on all points) designs an 8^{th} order trajectory. If the two derivative constraints on the via-point would be omitted (as these are not necessary to guarantee $\mathcal{C}^2$ parametric continuity), the order of the trajectory would reduce to 6 (see (6.2)).

In general, a trajectory $\mathcal{T}$ is defined as:

$$\mathcal{T}(\mathbf{q}_c, \mathbf{S}_k, \mathbf{M}, \mathbf{t}_s), \tag{6.3}$$

where $\mathbf{q}_c$ is the constraint vector as defined in (6.2), $\mathbf{S}_k = [q_k,\ \dot{q}_k, \ldots, q_k^{(p)}]^T$, is the current state of the system (measured or from previous trajectory step) with highest order of derivative p and

$$\mathbf{M} = \begin{bmatrix} q_{max,1} & \dot{q}_{max,1} & \cdots & q^{(p)}_{max,1} \\ \vdots & \vdots & \vdots & \vdots \\ q_{max,N} & \dot{q}_{max,N} & \cdots & q^{(p)}_{max,N} \end{bmatrix}, \tag{6.4}$$

is the matrix of maximum motion constraints on all N points. $\mathbf{t}_s = [t_{s,1}, \ldots, t_{s,N}]^T$ is the vector containing the timing constraints of all N points. The output is the vector $\mathbf{a}$ containing the polynomial coefficients of the trajectory: $\mathbf{a} = [a_0,\ a_1, \ldots,\ a_{n_t}]^T$. Depending on the control method (e.g., position, velocity, torque control), this vector is converted into a motion profile. For instance, for position control the motion profile would be defined by

$$q(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_{n_t} t^{n_t}, \tag{6.5}$$

where the degree of the polynomial n_t depends on predefined choices (see also (3.54)).

As a trajectory depends on the current state, as well as on events that are not known before they occur, only the state of the next step of the trajectory $\mathcal{T}_{k=1}$ has to be computed. More precisely, for every iteration, **a** is recalculated and the execution time is updated. In this way, the trajectory is redesigned every iteration to changing constraints and motion execution. This is elaborated in more detail in the following subsections.

6.2.1 Event-Based versus Rate-Based

The proposed algorithm allows for event-based or rate-based trajectory generation (see Algorithm 6.2: line 1). For event-based generation, a trajectory update is incorporated only when an event occurs. This is motivated by the fact that events can occur at any moment, and should be detected and dealt with as soon as possible. Examples are for instance the avoidance of an obstacle or a safety manoeuvre. For rate-based trajectory generation, the trajectory is updated continuously at a certain rate enabling even small disturbances to be incorporated. A downside on the latter approach is that noise can affect the trajectory generation quite significantly. Rate-based generation is motivated by the fact that a sensor will not necessarily execute measurements at the same rate as the local control loop. A logical implementation is to update the trajectory at the same rate as the sensor.

As both methods are largely similar (the only difference lies in the triggering of the trajectory generator), a general algorithm for direct, online trajectory generation is shown in pseudo-code in Algorithm 6.2. Depending on an update due to an event-trigger or rate-trigger, a trajectory is generated as follows. First, depending on predefined (or changed) constraints, the execution time of the trajectory is determined (Algorithm 6.2: line 2). Following, all initial and final constraints as well as the trajectory timing are updated (Algorithm 6.2: line 3). Finally, the polynomial coefficients **a** are computed, from which the state of the next step $\mathbf{S}_{k+1}$ is determined.

Determining the actual values for the final constraints depends on a separate mechanism. For example, when considering an avoidance motion with visual obstacle detection, the proper values for the final constraints (e.g., a position that avoids the obstacle) are determined by visual processing.

6.2.2 Point-to-Point versus Multi-Point

When designing a trajectory with two points, the evolution of the final point (i.e., the position of the point and its time derivatives) is a variable that can be altered. A change of this variable can be applied at any moment in time. If the trajectory is designed to contain multiple points, more design choices (i.e., more variables which can be altered) become available. For instance, the constraints on the via-point can be limited to only position or velocity as a continuous $\mathcal{C}^2$ trajectory is already guaranteed. Moreover, this choice is preferable as with higher order trajectories, the behaviour becomes more oscillatory (i.e., Runge's phenomenon). With the addition of via-points, the degree of the trajectory will grow depending on the number of constraints. Unfortunately, when a

Algorithm 6.2 Direct Trajectory Generation (DTG)

Input: $\mathcal{C}^{n_p}, \mathbf{q}_c, q_s, \mathbf{S}_k$ — initial conditions ◁
Output: $\mathbf{S}_{k+1}$ — next step state ◁
1: **if** $q_s > 0 \;||\; \text{mod}\;(i, 10) = 0$ **then** — event or rate-based ◁
2: compute t_{ev}, $t_{e\alpha}$ — see Algorithm 6.3 ◁
3: $q_I = q_{k-1}$ — update $\mathbf{q}, \mathbf{T}, t_f$ ◁
$q_f = q_f + q_s$
$v_I = v_{k-1}$
$\alpha_I = \alpha_{k-1}$
$t_f = t_s + t_e - \Delta t_{sum}$ — see also (6.8) ◁
$\mathbf{a} = \mathbf{T}^{\dagger}\mathbf{q}_c$ — see also (3.61) ◁
$q(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_{n_t} t^{n_t}$ — see also (3.59) ◁
$\mathbf{S}_{k+1} = [q_{k+1}, \dot{q}_{k+1}, \ddot{q}_{k+1}]^T$
4: **end if**

via-point is included, the order of the polynomial trajectory will increase (from 5^{th} to 6^{th}), and a minimum-jerk trajectory is no longer guaranteed.

Besides these local constraints on the points, the trajectory itself can also be constrained. In particular this involves bounding the motion of the complete trajectory (e.g., maximum velocity, acceleration), which can also be altered during runtime. The method for guaranteeing such constraint is discussed in the following subsection.

6.2.3 Constraint Optimization

When considering that a new trajectory can be generated at any arbitrary state and time, the symmetry as found in traditional trajectories (i.e., the polynomial trajectory can be mirrored around the middle point for odd order polynomials) does not hold any more and a relation between execution time and constraints is difficult to obtain. This difficulty originates from the order of the trajectory, as finding roots for higher order polynomials becomes a cumbersome and computational intensive task. A much simpler solution is to optimize the constraint online. This implies that every iteration the constraints are evaluated, and if, due to a redesign of the trajectory, these would be violated, extra time is added to the trajectory. On the other hand, when a trajectory is altered such that a constraint is not reached, time can be subtracted from the total execution time.

The location of a current constraint (maximum or minimum) is found by computing the zero-crossings of the derivative (roots) of the considered polynomial and its magnitude by evaluating the original polynomial at the found roots. A simple steepest descent optimization routine [116] is sufficient to accommodate for an eventual constraint mismatch and does not need to be executed in one iteration. For a velocity and acceleration constraint this is respectively expressed as

$$\begin{aligned} t_{ev} &= d_v(|v_m| - v_{max}), \\ t_{e\alpha} &= d_\alpha(|\alpha_m| - \alpha_{max}), \end{aligned} \tag{6.6}$$

in which $d_v > 0$ and $d_\alpha > 0$ defines the rate of convergence, v_{max} and α_{max} the predefined constraints and v_m and α_m the computed constraint (maximum or

minimum) of the current trajectory.

As the number of iterations is fairly limited, the computation can be spread out over several iterations. Algorithm 6.3 presents more details of the optimization procedure in pseudo-code for point-to-point motion. For multi-point trajectories the root solving problem becomes higher order, however, the method of solution remains the same.

Algorithm 6.3 Constraint Optimization for 5^{th} order polynomial

Input: $\mathbf{a}, \mathbf{M}$ ◁ trajectory and constraints
Output: $t_{ev} \mid\mid t_{e\alpha}$ ◁ extra time to satisfy constraint

$\mathcal{T}_{vc} = 2a_3 + 6a_4 t + 12a_5 t^2 + 20a_6 t^3$ ◁ velocity constraint
$\mathcal{T}_{\alpha c} = 6a_4 + 24a_5 t + 60a_6 t^2$ ◁ acceleration constraint
if $\mathcal{T}_{vc}$ **then**
  $\mathcal{T}_{vc} = 0$ ◁ find roots and sort descending in $\mathbf{r}$
  $t_m = \arg\max\{\mathcal{T}_{vc} = 0\}$ ◁ time of maximum
  $v_m = a_2 + 2a_3 t_m + 3a_4 t_m^2 + 4a_5 t_m^3 + 5a_6 t_m^4$
  if $v_m > v_{max}$ **then**
    $t_{ev} = d_v(|v_m| - v_{max})$ ◁ steepest descent
  end if
end if
if $\mathcal{T}_{\alpha c}$ **then**
  $\mathcal{T}_{\alpha c} = 0$ ◁ find roots and sort descending in $\mathbf{r}$
  $t_m = \arg\max\{\mathcal{T}_{\alpha c} = 0\}$ ◁ time of maximum
  $\alpha_m = 2a_3 + 6a_4 t_m + 12a_5 t_m^2 + 20a_6 t_m^3$
  if $\alpha_m > \alpha_{max}$ **then**
    $t_{e\alpha} = d_\alpha(|\alpha_m| - \alpha_{max})$ ◁ steepest descent
  end if
end if

The extra time needed to avoid violating a constraint is added to (or subtracted from) the originally designed trajectory time. The fact that every iteration a new trajectory is generated implies that the trajectory time is continuously decreasing (accept when t_{ev} or $t_{e\alpha}$ is added) and is equal to zero at the end of the trajectory. More specifically, at $t = 0$ and $t = t_f$ it holds that

$$t(0) = t_f, \quad \text{and} \quad t(t_f) = 0. \tag{6.7}$$

When computing the trajectory online, the initial and final time are defined as

$$t_I = 0, \quad \text{and} \quad t_f = t_s + t_e - \Delta t_{sum}, \tag{6.8}$$

where t_s is obtained from (6.9) and t_e is obtained from (6.6). Δt_{sum} is the ascending trajectory time and can be approximated as $\Delta t_{sum} = T_l n_{it}$, in which T_l is the local loop time with iteration count n_{it}.

In general if a motion duration is not given, the optimal motion duration for a minimum jerk trajectory is infinite. This can be easily verified by noticing that the jerk cost approaches zero as the duration of a minimum-jerk trajectory approaches infinity. Adding a secondary term (i.e., time) is therefore the most straight-forward way to avoid such prediction [62]. By guaranteeing that a kinematic constraint is always reached (with constraint optimization), a minimum jerk and time-optimal trajectory is achieved.

It has to be noted that for the evaluation of the roots of a polynomial, the complexity depends on the order of the polynomial. As the order of the polynomial increases, it becomes more complex to determine these roots. A closed-form solution for higher order polynomials becomes easily infeasible (i.e., too many terms) for implementation (e.g., for quartic polynomials) or even do not exist (i.e., for quintic polynomials and higher, as stated by the Abel-Ruffini theorem[1]). Solutions, however, can be approximated by numerical methods for root-finding (e.g., the method of Newton-Raphson) or numerical methods which locate local minima (or maxima), where prior knowledge of the shape of the polynomial can be taken into account (e.g., optimization methods). A drawback, however, is the fact that such methods can take considerable computation time.

6.2.4 Trajectory Synchronization

When considering a multiple degree of freedom (MDOF) trajectory where motion between DOFs is uncoupled, it is unlikely that all motions will be finalized at the same time instant t_f. In the case of direct trajectory generation where replanning requires altering the execution time, this synchronization needs to be evaluated at runtime. This involves determining which DOF has the lowest maximum constraint (i.e., velocity, acceleration, etc.) and adapting all trajectories to its execution time. For a point-to-point 5^{th} order trajectory this is evaluated as

$$t_{s,l} = \left\{ \frac{15}{8}\frac{h}{v_{max}}, \sqrt{\frac{10\sqrt{3}}{3}\frac{h}{\alpha_{max}}} \right\}, \tag{6.9}$$

where $t_{s,l}$, $l \in \{v, \alpha\}$ is the execution time, $h = q_f - q_I$ and v_{max} and α_{max} are the maximum velocity and acceleration respectively.

When considering a multi-point trajectory, the shape of the trajectory determines the relation between execution time and constraints. For a 3-point trajectory with only a position constraint on the via-point (i.e., a 6^{th} order trajectory), equation (6.9) can be used. For a 3-point trajectory with equal constraints on all points (i.e., an 8^{th} order trajectory) the relation is found, similar to the developments in Section 3.5.2, as

$$t_{s,l} = \left\{ v_8\frac{h}{v_{max}}, \alpha_8\frac{h}{\sqrt{\alpha_{max}}} \right\}, \tag{6.10}$$

where $v_8 = 1.9444$ and $\alpha_8 = 2.6925$ (expressed numerically as the solution is determined experimentally). These numerical results are obtained by filling in arbitrary values for v_8 or α_8 (e.g., $v_8 = 1$ or $\alpha_8 = 1$), running a simulation, determining the maximum velocity and acceleration, and computing the real numeric value:

$$v_8 = t_{s,l}\frac{\max|\dot{q}(t)|}{h}, \quad \text{and} \quad \alpha_8 = t_{s,l}\frac{\sqrt{\max|\ddot{q}(t)|}}{h}. \tag{6.11}$$

[1]The Abel-Ruffini theorem states that there is no general algebraic solution to polynomial equations of degree five or higher. It does, however, not assert that higher-degree polynomial equations are unsolvable.

When during runtime a trajectory is altered and the final time t_f is changed due to an addition or subtraction of t_e, this is passed on to all other trajectories. Violations due to this addition is again dealt with by the constraint optimization for all other DOFs.

6.3 Experimental Results

In order to show that the method can generate from an arbitrary state the desired motion profiles as explained in Section 6.2, first results are shown for a single DOF. Experimental results carried out with a 7-DOF redundant, anthropomorphic robotic manipulator, where motion is designed in Cartesian space, can be found in Chapter 8.

6.3.1 Experimental Setup

For experimental results, the first joint q_1 of the 7-DOF robotic manipulator AMOR[2] is used (see Fig. 6.2). For safety, this base joint is limited to a maximum angular velocity and acceleration of $\nu_{1,max} = 0.5\ [rad/s]$ and $\alpha_{1,max} = 1\ [rad/s^2]$ respectively.The local joint controller (i.e., PD-control with an update rate of 1 $[kHz]$) is implemented in `C/C++`, using the Eigen library for vector and matrix manipulation. All processing is executed on a standard notebook where communication with the manipulator is done via a CAN-USB device, with different threads managing the data exchange with the CAN device (one for reading and one for writing).

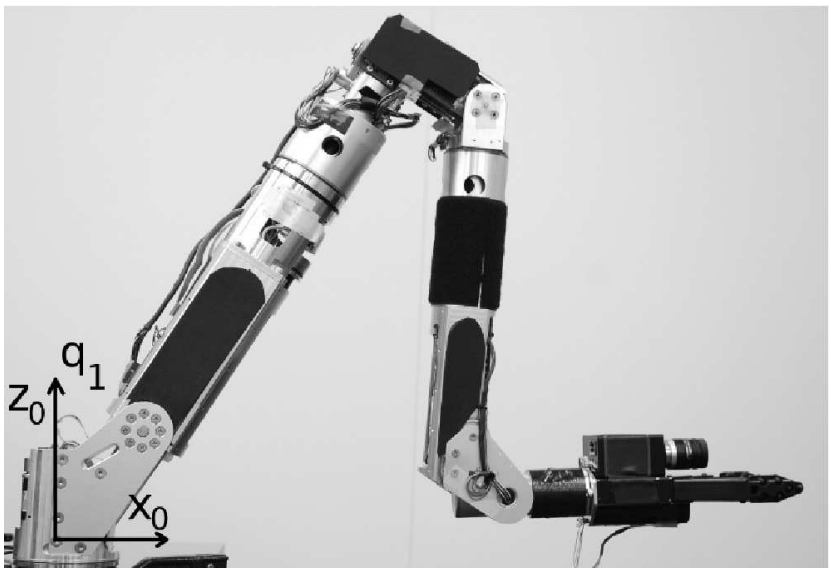


Figure 6.2: DTG is experimentally verified with a single degree of freedom, i.e., the base joint q_1 of the 7-DOF redundant manipulator AMOR.

[2]http://www.amorrobot.com/

6.3.2 Experimental Results for a Single Degree of Freedom

As task a single DOF trajectory is designed with $\mathcal{C}^2$ continuity from initial point $q_I = 0\ [rad]$ to final point $q_f = 0.5\ [rad]$. This implies a 5^{th} order (quintic) trajectory where the timing is determined by a maximum constraint (i.e., maximum velocity or acceleration). The constraints on the initial and final point are designed as $\dot{q}_I = \ddot{q}_I = \dot{q}_f = \ddot{q}_f = 0$. At time $t = 1.2\ [s]$ the final point $q_f = 0.5\ [rad]$ is changed to $q_f = 0.8\ [rad]$. This shows that at any time and in any state motion can smoothly be directed to new constraints. Direct trajectory generation is shown in Fig. 6.3 and Fig. 6.4, for constraint optimization of velocity and acceleration respectively. For the velocity case it can be seen that the maximum velocity does not exceed $v_{max} = 0.5\ [rad/s]$. For the acceleration case the bound of $|\alpha_{max}| = 1\ [rad/s^2]$ is not exceeded. Closer inspection of the final time t_f shows that directly after the change of the final point the final time is increased (i.e., incrementally optimized) to comply with predefined bounds. As final note it should be mentioned that the acceleration profile is $\mathcal{C}^0$ continuous, the velocity profile is $\mathcal{C}^1$ continuous and the position profile is $\mathcal{C}^2$ continuous.

6.3.3 Constraint Optimization

As explained in Section 3.5.2 and Section 6.2.3 the relationship between the execution time of a trajectory and the imposed constraint is difficult to obtain for trajectories that can be altered at any time instant. Instead, an optimization routine is developed that incrementally extends (or shortens) the final time when at any time instant a constraint would be violated. In Fig. 6.5 this can be seen as at $t = 1.2\ [s]$ a new final condition is imposed for which the predefined constraints would be violated if the execution time of the trajectory would not be altered. Fig. 6.6 shows how the constraint optimization can be solved in either one iteration (dashed black line) or spread out over several iterations (solid black line). The resulting velocity trajectory is the same for both methods. The only difference lies in the execution time per iteration. As the solution for finding the roots of a 3^{rd} order polynomial is relatively simple (i.e., 3^{rd} order due to a 5^{th} order polynomial position trajectory with global velocity constraint; see Algorithm 6.3), this optimization routine can also be implemented in one iteration online (i.e., the computation time is negligible). In the case of a 6^{th} order polynomial, the order of the polynomial for which the roots have to be determined is maximum 4, for a global velocity constraint. As a closed-form solution for this is too complex for real-time implementation (i.e., too many terms), numerical methods have to be employed. Consequently, the computation time is of such magnitude that a solution can no longer be determined in a similar time-span. A solution can be found within one iteration (i.e., within $1\ [ms]$), however, as a typical optimization routine requires dozens of steps to achieve convergence, the solution for constraint optimization has to be spread out over multiple iterations. In this experimental case and when tuned properly, the number of iterations necessary for convergence does not exceed 40 (see Fig. 6.6).

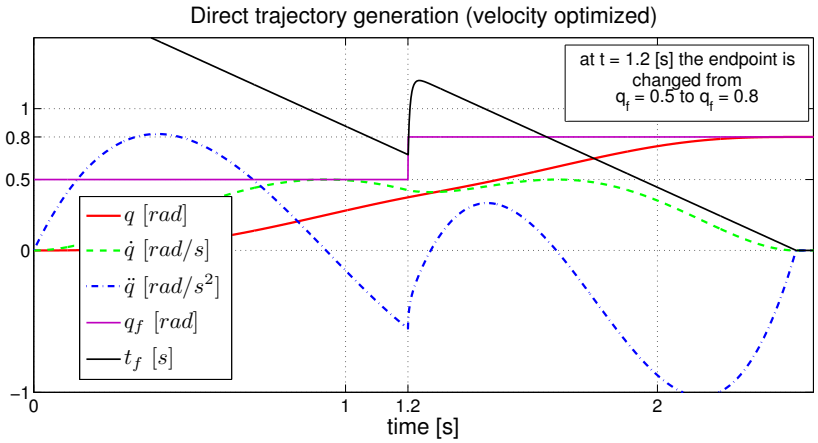


Figure 6.3: Direct trajectory generation with online end-point change. In order to comply with desired constraints ($|v_{max}| = 0.5\ [rad/s]$), the final time of the trajectory t_f is iteratively extended directly after $t = 1.2\ [s]$ (black line). Also note that the acceleration profile is $\mathcal{C}^0$ continuous, the velocity profile is $\mathcal{C}^1$ continuous and the position profile is $\mathcal{C}^2$ continuous.

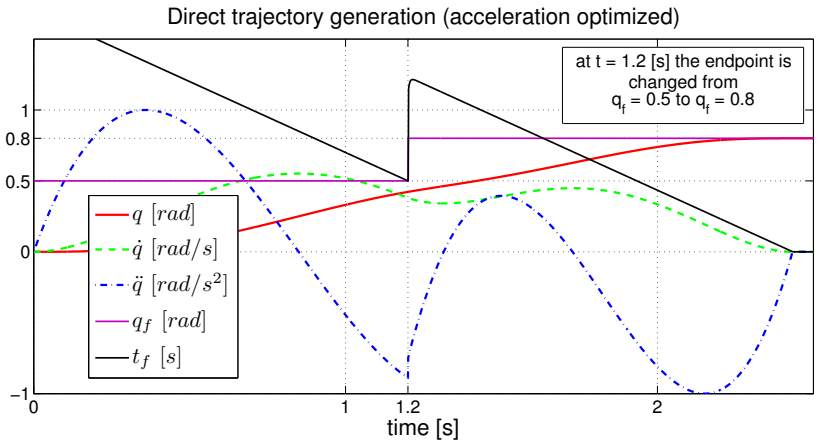


Figure 6.4: Direct trajectory generation with online end-point change. In order to comply with desired constraints ($|\alpha_{max}| = 1\ [rad/s^2]$), the final time of the trajectory t_f is iteratively extended directly after $t = 1.2\ [s]$ (black line). Also note that the acceleration profile is $\mathcal{C}^0$ continuous, the velocity profile is $\mathcal{C}^1$ continuous and the position profile is $\mathcal{C}^2$ continuous.

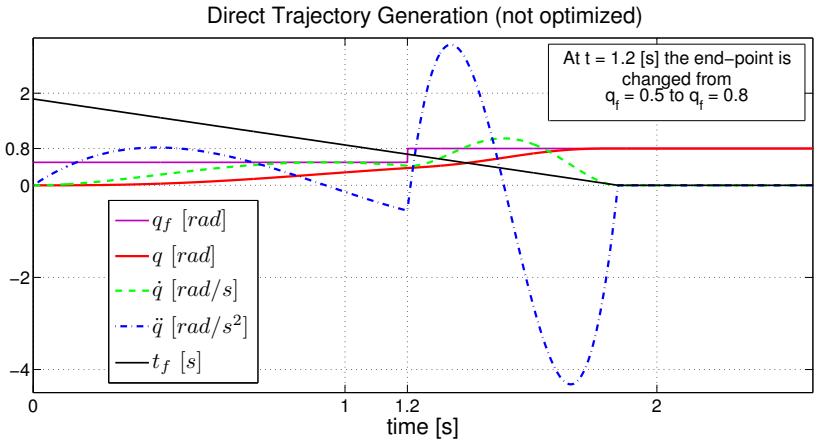


Figure 6.5: Direct trajectory generation with online change of the final point. In this example the final time t_f is not altered, resulting in the violation of the predefined velocity and acceleration constraints ($|v_{max}| = 0.5\ [rad/s]$ and $|\alpha_{max}| = 1\ [rad/s^2]$ respectively).

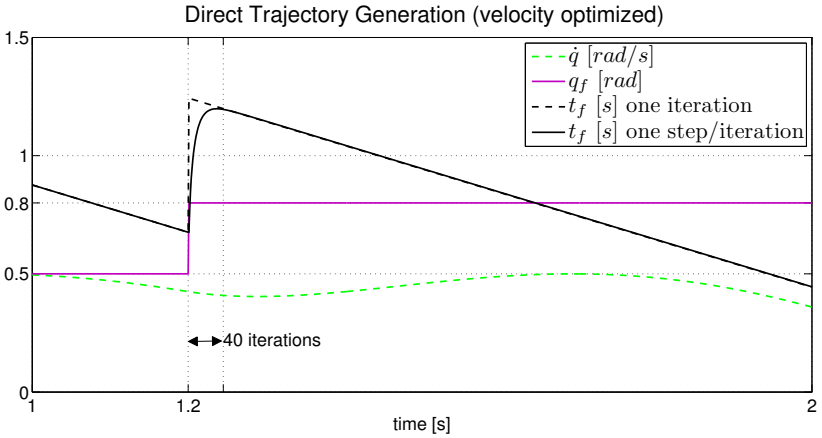


Figure 6.6: Magnification of direct trajectory generation with constraint optimization for velocity. The timing of constraint optimization is shown as executed in one iteration (dashed black) and as executed with one optimization step per iteration (solid black). Both methods result in a similar velocity trajectory.

6.4 Summary

This chapter proposed a method for the direct generation of trajectories for vision-based control. As introduction the properties of vision-based motion planning and traditional trajectory planning are addressed with respect to constraints and update rates. Vision-based planning is known to execute motion on a path planning level where changes of constraints are incorporated at a fairly slow rate. For traditional trajectory planning the opposite properties hold, i.e., constraints can be easily incorporated (however, not online) and control rates are fairly high compared to vision-based planning. This conflict of properties is the motivation for the developed approach. The proposed direct trajectory generation method is presented from a general motion point-of-view where the different design choices are explained in more detail. In particular, motion is predefined based on the type of trajectory (i.e., order), which constraints apply (i.e., spatial and kinematic) and how the trajectory timing should be incorporated (i.e., based on constraints or fixed). Following, several design possibilities for trajectory generation (i.e., event-based versus rate-based and point-to-point versus multi-point) are clarified and detailed in pseudo-code. Event-based versus rate-based trajectory generation entails that the trajectory should be either updated based on an event (e.g., due to a visual measurement) or updated at a fixed rate (e.g., at the rate of the vision sensor). Furthermore, similar to traditional trajectory generation, motion can be designed with 2 points (i.e., point-to-point) or with multiple points. Additionally, an online optimization procedure is proposed which guarantees that a constraint is ensured whenever a trajectory is redesigned. This is achieved by monitoring the predefined constraints online and altering the execution time of the trajectory (i.e., either adding or subtracting time), whenever a constraint would be violated. The alteration of this execution time can be effected in one iteration or in multiple iterations depending on the available computational resources. Finally, the proposed method is validated in experimental setting for a single degree of freedom. It is shown that the proposed method can deal with changes of constraints online at a trajectory planning level and at a traditional motion planning rate.

Part III

Application and Implementation

CHAPTER 7

Product Pattern-Based Visual Servoing

Abstract. This chapter presents the application and implementation details of product pattern-based visual servoing. The method is denoted as such due to the use of the repetitive pattern of the product for visual control. Methods as proposed in previous chapters are adapted to fit the purpose of direct visual control of a planar motion stage. In particular, algorithms for the detection of the product pattern and the generation of the image-based trajectory as well as the control structure are explained in detail. Following, experimental result show the effectiveness of the approach.

7.1 Introduction and Motivation

The continuous consumer demand for better and faster electronics (e.g., displays for smart-phones, televisions and cameras) has led to the development of displays with increasingly higher resolution and increasingly smaller pixel size. The technology for manufacturing these displays has to be improved or reinvented as well. In particular, current state-of-the art display technology offers products which have a flexible or non-rigid nature (see Fig. 1.1). The fabrication of these devices then becomes a clear challenge as these flexibilities cause high inaccuracies in the manufacturing process.

As explained in the introduction of this thesis in Section 1.2.1, the performance of state-of-the-art positioning systems still depends on the rigid design of a measurement and fixation system. With the manufacturing of displays this solution can be problematic. For example, Organic Light Emitting Diode (OLED) displays need a printing task on every pixel, however, when the locations for printing are inaccurate or unknown, the display will not be manufactured correctly. These inaccuracies occur when the display is flexible, causing a mismatch between measurements of the display location and the actual pixel location. Direct visual measurements can circumvent this problem and can determine accurately where a printing task should be executed. However, when vision becomes part of a control system, a number of problems may arise. Foremost, the fact that visual processing can take considerably more time to execute than a local control loop, demands the use of a double control loop structure (see Fig. 2.1). The local controller is executed at a fast rate (e.g., 1 $[kHz]$) to control the motion of the system, while a slow (e.g., 25 $[Hz]$) visual loop designs the motion of the system. This control structure is necessary to ensure stability and at the same time allow vision to be part of the control loop.

The drawbacks associated to this control structure directly affect the performance of the system. In particular, the delay induced by the vision system

deteriorates the overall performance. This not only includes the update rate of the image sensor, but also the transport of the image to the processing board and the processing itself. Furthermore, as visual servoing is a sensor-based control methodology, typical design of motion is executed on a path-planning level, where constraints are not directly taken into account.

In order to avoid the delay as induced by visual processing on standard processing platforms, this chapter presents a visual control system that *directly* takes visual measurements into the control loop. This is achieved with a high-speed vision system where the image sensor is directly connected to an FPGA for visual processing and control feedback. This means that a double control loop structure is *not* present and a traditional approach towards visual motion control is attainable (i.e., trajectory tracking).

As such, this chapter presents the application and implementation details of the developed method; product pattern-based visual servoing. The benefits of using direct high-speed visual feedback are discussed, which, for our micrometer positioning system, can be exemplified by a trade-off between frame rate and image size to obtain useful information.

The developed methodologies include visual processing to detect individual features and the subsequent design of motion. The direct trajectory generation method as proposed in Chapter 6 is applied for smooth and constrained motion which can be adapted online. Finally, experimental results are presented to validate the developed method.

7.1.1 High-Speed Visual Control Trade-off

As mentioned in Chapter 5 and Chapter 6, a basic visual servoing control architecture is divided into a slow visual reference loop (e.g., video rate) and a fast local joint control loop (e.g., kHz rate). Due to the slow visual update rate, the control of the end-effector is stable, but the delay between a disturbance and a control action can be dozens of sample-times in joint control reference. However, as stated in [64], the availability of more and more computing power has enabled researchers to use vision for feedback at higher rates. If the camera is thus sampled high enough (e.g., 1 $[kHz]$), this can reliably be used as single feedback for motion control. This gives rise to some design choices that have to be made:

Ogawa et al. [107] introduced a trade-off relationship between magnification and trackability in microscopic object tracking which can be written as

$$n_f M = \frac{fps}{v_r}, \tag{7.1}$$

with M the magnification index (i.e. the ratio of the target length to the visual field width), fps the frame rate of the vision system, n_f the trackability index (defined as the number of frames for the target to cross the visual field) and $v_r = \frac{v}{l}$, a velocity measure with l the diameter (in $[m]$) of the target and v the physical velocity (in $[m/s]$) of the target. For a sufficiently high $n_f M$ (Ogawa et al. obtained $n_f M > 20$), the system ensures a sufficient magnification and trackability.

A similar classification of performance is stated by finding a trade-off between frame rate and image size. In this, the velocity parameter is disregarded

and instead the focus lies more on the measurement accuracy that can be obtained. The relation between the frame rate (fps) and the cell size of the image sensor for image processing (number of pixels c) is defined as

$$\frac{1}{fps} \propto c, \tag{7.2}$$

where c is measured in pixels (or $[\mu m]$) and assumed as square sized. Of course it is highly application dependent what image size (in pixels) is too excessive for real-time image processing. Fig. 7.1 shows a comparison of several high-speed vision systems, as well as the theoretical communication limits of several standard industrial communication protocols for machine vision cameras.

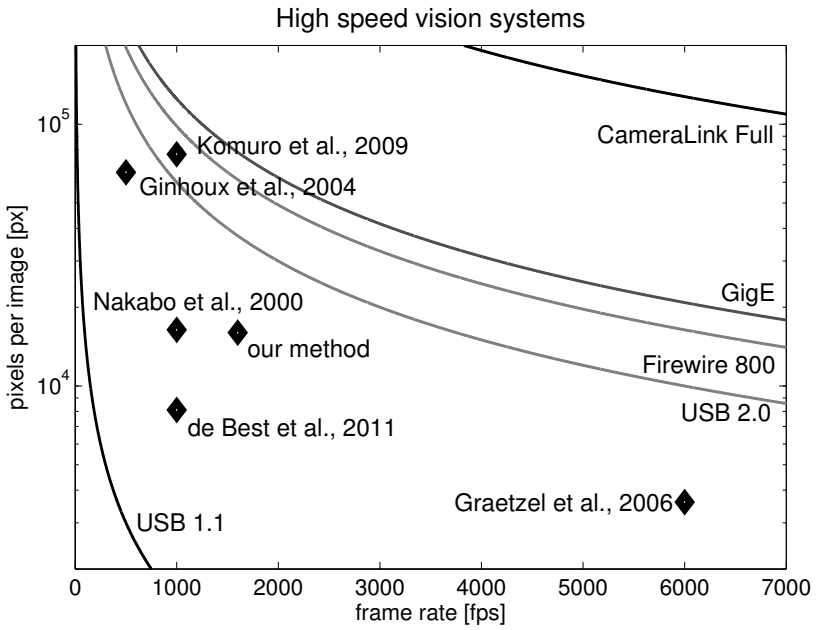


Figure 7.1: Comparison of high-speed vision systems. Solid lines depict the theoretical communication transfer bandwidth limit. The compared references can be found in Komuro et al. [80], Ginhoux et al. [50], Nakabo et al. [105], de Best et al. [34] and Graetzel et al. [51].

Clearly, the trade-off between image size and frame rate determines the properties of the visual control system, as is shown by all references. For example, the largest image size for visual feedback is 320×240 $[px]$ as is shown by Komuro et al. [80], where a frame rate of 1 $[kHz]$ is achieved. A predecessor of this system was developed by Nakabo et al. [105], with an image size of 128×128 $[px]$ and similar frame rate. The other extreme is developed by Graetzel et al. [51], where a frame rate of 6 $[kHz]$ is achieved, however, with an image size of 60×60 $[px]$. In fact, visual control is not executed in this, as a static camera observes a fixed target (i.e., real-time wing beat analysis of drosophila). The method of Ginhoux et al. [50] shows a model predictive control scheme combined with visual servoing to track a beating heart in robotic

surgery, which is achieved with a frame rate of 500 $[Hz]$ and an image size of 256×256 $[px]$. From this comparison it might be obvious that a high frame rate is chosen at the cost of a high resolution, which implies a lower accuracy as less pixels are available for analysis. This trade-off is therefore the limiting factor in vision-based control systems. The two remaining methods, i.e., from de Best et al. [34] and our method, are discussed and presented in the following section.

7.1.2 Repetitive Product Pattern

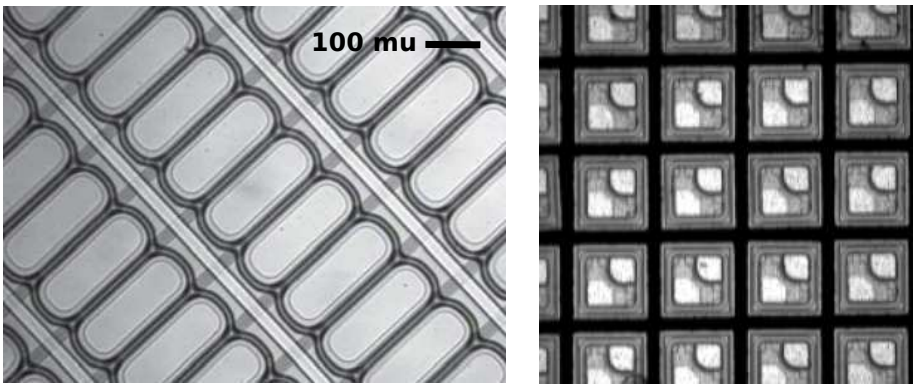
The advantage of using the product directly for positioning can be explained best by a comparison. In traditional motion control, the motor encoder determines in part the overall system accuracy. This is true for systems in which the product is fixed with respect to the encoder and if the product itself can not deform. If the location of the production head is relatively far from the motor encoder, vibrations can play a role in this large measurement loop, and the positioning accuracy can decrease. Similarly, if measurements are obtained to control the position of a table, the transformation between production head and control reference (e.g., kinematic or dynamic model) has to be known with high accuracy. This knowledge is usually difficult to obtain and can even change over time. Furthermore, if measurements are done to control the position of a table, the fixation of the product on the table has to be rigid and identical for every new product, which implies a costly motion and fixation system. A direct, visual measurement system will effectively relate the position of the product with respect to the production head since these have the same coordinate frame. Moreover, when the product has a repetitive pattern, this can act repetitively as (visual) reference encoder and, if sampled fast enough, separate motor encoders become redundant. In that sense, it would be highly beneficial if the image (i.e., the product pattern) that acts as encoder is relatively simple and needs few pixels for accurate measurements. Basic operations and simple geometry on low resolution images would then result in sub-pixel accurate measurements and a high frame rate and thus fast feedback.

Examples of repetitive patterns are for instance organic LED displays (OLED, Fig. 7.2.a and Fig. 1.1) or semiconductors on a wafer substrate (Fig. 7.2.b). In both cases a positioning task has to align the production head with respect to a repetitive pattern feature and perform a task. In the case of OLED manufacturing, this additional task consists of inkjet printing. In the case of semiconductor manufacturing, a pick-and-place task has to be carried out. Despite the difference in manufacturing, a similar approach towards using the product as encoder can be taken.

These two industrial production processes are the topic of interest in the research project 'Fast Focus on Structures' (FFOS)[1]. This project is carried out by a joint consortium of industrial and academic partners and focussed on vision-based motion control regarding a repetitive pattern. For this, both industrial cases were considered and contributions were made to develop a flexible, low-cost, miniaturized measurement system for accurate positioning with respect to a product.

In particular, for the case of semiconductor manufacturing, de Best et al. [35] presents a visual control system with a frame rate of 1 $[kHz]$ and an image

[1]supported by Agentschap NL - IOP Precision Technology - Fast Focus On Structures (FFOS).



(a) OLED substrate (b) Wafer substrate

Figure 7.2: Repetitive product patterns. High resolution image of OLED substrate (a). High Resolution image of transistors on a wafer (b).

size of 90×90 $[px]$. In [34] results are shown where a 2D stop-and-go positioning task is executed with a positioning accuracy of ± 10 $[\mu m]$ (3σ measurement variation: 0.3 $[\mu m]$) and a delay of 2.5 $[ms]$. A repetitive pattern on a wafer consisting of semiconductor products (see Fig. 7.2.b) is used for direct feedback. Moreover, in [35] the theory of iterative learning control (ILC) is used to handle scale varying set-points. This is particularly useful for a repetitive pattern which does not have a perfectly identical pitch between products (e.g., due to the flexibility of the wafer). Experimental results with an industrial XY-wafer stage show that a positioning error can be reduced to less than 5 $[\mu m]$ after convergence of the ILC algorithm.

As the development of a (miniaturized) visual control system involves the mapping of image processing algorithms on dedicated hardware, the research project 'Embedded Vision Architecture' (EVA)[2] has adopted the FFOS case (i.e., industrial inkjet printing in particular) as practical application. As such, research has been carried out to implement the complete vision pipeline on a FPGA (Field-Programmable Gate Array) processor [157, 158, 159] and a SIMD (Single Instruction Multiple Data) processor [59].

7.1.3 Inkjet Printing of Near-Repetitive Patterns

Following, the case of industrial inkjet printing is motivated by the current state-of-the-art and solutions for improvement and their developments are proposed.

The manufacturing of Organic Light Emitting Diode (OLED) displays requires an inkjet printing task on each individual OLED display pixel (or cell). As such, each pixel (or cell) has to be aligned with the printing nozzle (printhead) and a printing action shoots a droplet of polymer into each cell (see Fig. 7.3 and [89]). For cost reasons, the manufacturing of such displays has to be done as fast as possible, implying that also the printing should be carried out as fast as possible. The obvious solution of printing in a stop-and-go manner,

[2]supported by the Dutch Ministry of Economic Affairs - Embedded Vision Architecture (EVA).

therefore, does not suffice. Instead, a printing task has to be executed on-the fly, where the print-head moves with a fixed velocity over each OLED cell. If a display is a rigid structure (i.e., the pitch between OLED cells is equal) and the location of the display is known at all times, the printing task could be executed with a constant velocity and a constant drop-on-demand (DOD) print-frequency [163]. Existing research adopting this technique can be found in e.g., [36], [118], an overview of inkjet-based micro-manufacturing is given in [68]. However, due to the flexible nature of the display and the absence of a proper fixation system, a designed trajectory is a necessity. An additional reason for designing motion with a trajectory instead of a constant reference is the quality of the printing process. As the printing quality deteriorates with a higher velocity of the print-head with respect to the motion stage [153], a low velocity, when a printing action is executed, is desirable. In order to obtain a higher average velocity, a constant reference velocity should be avoided, and a designed trajectory should be employed.

Fig. 7.2(a) shows a microscopic view of an OLED display. The size of one OLED cell is 220×80 $[\mu m]$, with a pitch in horizontal and vertical direction of 220 $[\mu m]$ and 80 $[\mu m]$ respectively. For the printing task, the print-head shoots a polymer droplet, which has a diameter of 50 $[\mu m]$, at the centre of each OLED cell (see Fig. 7.3 and [118]). In order to execute an accurate printing task, the delay of the printing task itself needs to be taken into account. The travel time of a droplet depends on the velocity of the droplet v_{print} and the distance it has to travel d_{print}, and can be determined as

$$t_{travel} = \frac{d_{print}}{v_{print}}. \tag{7.3}$$

Assuming a droplet print velocity of $v_{print} = 5$ $[m/s]$ and a printing height of $d_{print} = 1$ $[mm]$, the travel time of a droplet equals $t_{travel} = 0.2$ $[ms]$. Consider that a printing action is triggered when the table is moving with a velocity of $v_{table} = 4$ $[px/frame] = 28.8$ $[mm/s]$ (for a 4.5 $[\mu m]$ sized pixel and a frame rate of 1600 fps). This implies that from the droplet leaving the nozzle to the droplet hitting the OLED cell, a distance of 5.76 $[\mu m]$ or 1.28 $[px]$ has been travelled by the motion stage. Similarly, the bounds on the velocity over the centre of the OLED cell can then be determined. Assume that the position error for the printing process is tolerable at ± 10 $[\mu m]$ from the centre of the OLED cell. If the printing task is triggered exactly at the centre of the OLED cell, the velocity which violates this error is then found as 6.9 $[px/frame]$. As such, the tolerance for the velocity of the table at each OLED cell centre is thus specified as: $v_{table} = 4 \pm 2.9$ $[px/frame] = 28.8 \pm 20.9$ $[mm/s]$.

As there already exists an error of 5.76 $[\mu m]$ if the printing task is triggered exactly at the centre of the OLED cell with perfect velocity tracking, a better solution is to predict when the print-head should be triggered. This can be done by a linear predictor such as an α-β filter [70], which should take into account the delay due to the travel time of the droplet (i.e., $t_{travel} = 0.2$ $[ms]$) as well as the delay between the trigger of the print-head and the droplet leaving the nozzle.

This analysis assumes that the print-head is located at the centre of the image, which might not be the case. For the actual printing task the delay due to this mismatch has to be taken into account as well.

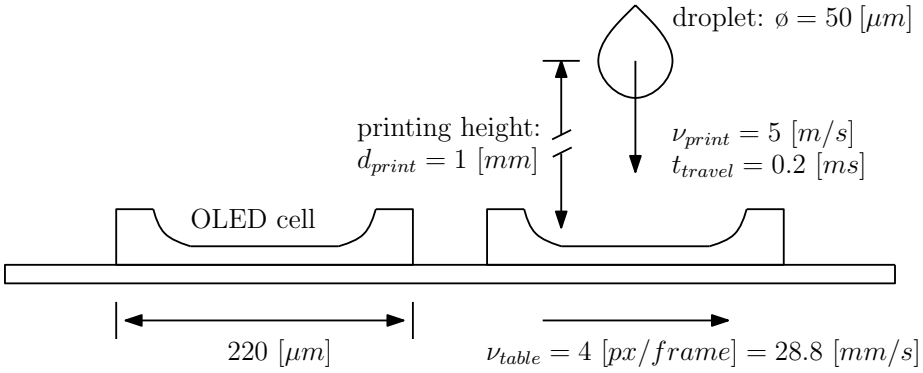


Figure 7.3: Close-up of inkjet printing process. The travel time of the droplet causes a position error of 5.76 $[\mu m]$. In order for the printing task to stay within the defined positioning tolerance of ± 10 $[\mu m]$, the velocity of the table at each OLED cell centre should ensure $\nu_{table} = 4 \pm 2.9$ $[px/frame]$.

With an average velocity of $\bar{v}_{table} = 5$ $[px/frame] = 36$ $[mm/s]$ and an assumed display size of 640×480 $[px]$, one row of 480 OLED cells will be covered in 2.93 seconds, which implies a print-frequency of $\sim$160 $[Hz]$.

This brief analysis assumes very conservative assumptions, and does therefore not represent the state-of-the-art. For instance, print frequencies up to 10-20 $[kHz]$ are attainable, and a higher speed of the printing task are therefore feasible.

7.2 Product Pattern-Based Visual Control

The use of a repetitive pattern for control can be summed up in a few consecutive steps. First, it has to be determined if distortions due to the lens have to be taken into account. Subsequently, image processing algorithms are employed which compute the location of the OLED cells in the field of view. These measurements are then used as input for the design of the motion trajectory. These developments are presented in more detail in forthcoming sections.

7.2.1 Planar Microscopic Camera calibration

When regarding the complete image solely as visual information for control (e.g., as encoder), high demands must be set to obtain correct visual measurements. From visual systems it is known that lens distortion causes image information to be misplaced on the image sensor. A first step is therefore to employ a camera calibration algorithm to determine if the distortion due to the lens causes a disturbance. As the object can only be observed when the product plane is parallel to the image plane and the field of view is limited (i.e., $\sim$6.2 mm^2 for 640×480 $[px]$), macroscopic camera calibration techniques [123] and printed calibration patterns can not be used. A method is proposed that simplifies traditional macro calibration by the assumption that the depth parameter z is fixed.

Due to the fact that the used microscopic optical system has a small depth of focus, a fixed depth and only planar motions will be made, a few assumptions can be made that simplify the parametric microscope model [164]. Tsai's well-known calibration algorithm [146] is therefore modified for the parallel case as follows. The transformation between global coordinates: (x_g, y_g, z_g) and object coordinates (x, y, z) can be written as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{R} \begin{bmatrix} x_g \\ y_g \\ z_g \end{bmatrix} + \mathbf{t} = \begin{bmatrix} r_{1,1} & r_{1,2} & t_x \\ r_{2,1} & r_{2,2} & t_y \\ r_{3,1} & r_{3,2} & t_z \end{bmatrix} \begin{bmatrix} x_g \\ y_g \\ 1 \end{bmatrix}, \tag{7.4}$$

with z_g equal to zero for coplanar points. The entries of the rotation matrix and translation vector are taken as defined in (4.8).

For microscopic lenses, only the first radial distortion parameter (κ_1) needs to be modelled, which relates the distorted image points $\mathbf{p} = [u,\ v]$ and the undistorted (or corrected) image points $\mathbf{p}_{cor} = [u_{cor},\ v_{cor}]$ image points as:

$$\begin{aligned} u_{cor} &= u(1 + \kappa_1 r^2), \\ v_{cor} &= v(1 + \kappa_1 r^2), \end{aligned} \tag{7.5}$$

with $r^2 = u^2 + v^2$. Furthermore, in (7.4), z can be written as:

$$z = r_{3,1} x_g + r_{3,2} y_g + t_z. \tag{7.6}$$

When now assuming the parallel relation between global coordinates and image coordinates (i.e., implying $r_{3,1} = r_{3,2} \simeq 0$), z can be simplified to $z = t_z$.

Finally combining all above equations, the following expression can written for the microscopic calibration model:

$$\begin{aligned} u(1 + \kappa_1 r^2) &= M_o(r_{1,1} x_g + r_{1,2} y_g + t_x), \\ v(1 + \kappa_1 r^2) &= M_o(r_{2,1} x_g + r_{2,2} y_g + t_y), \end{aligned} \tag{7.7}$$

with $M_o = f/t_z$ the optical magnification factor and f the focal length.

The algorithm to solve for the calibration parameters is adopted from Tsai [146] with the adapted definition for the microscope model. Tsai's two-step procedure first determines all extrinsic parameters through a closed-form solution and a radial alignment constraint (RAC) by setting up an overdetermined system of linear equations:

$$\begin{bmatrix} v x_g & v y_g & v & -u x_g & -u y_g \end{bmatrix} \begin{bmatrix} t_y^{-1} r_{1,1} \\ t_y^{-1} r_{1,2} \\ t_y^{-1} t_x \\ t_y^{-1} r_{2,1} \\ t_y^{-1} r_{2,2} \end{bmatrix} = u, \tag{7.8}$$

which can then be solved with $\mathbf{p} > 5$ calibration grid points.

Step 2 consists of a nonlinear optimization routine to determine the intrinsic parameters. (7.7) can be rewritten as

$$M_o(p + q) - \kappa_1(u + v)r^2 = u + v, \tag{7.9}$$

where

$$p = r_{1,1}x_g + r_{1,2}y_g + t_x,$$
$$q = r_{2,1}x_g + r_{2,2}y_g + t_y. \tag{7.10}$$

With $\mathbf{p} > 2$ calibration points, an overdetermined system of linear equations can be set up and solved for κ_1 and M_o:

$$\begin{bmatrix} p+q & -(u+v)r^2 \end{bmatrix} \begin{bmatrix} M_o \\ \kappa_1 \end{bmatrix} = u + v. \tag{7.11}$$

These initial estimates for κ_1 and M_o are then filled into (7.4) and optimized with a nonlinear optimization scheme.

The extraction of calibration points from images is done with the image processing algorithm presented in the following section.

7.2.2 Feature Localization

In order to use the product as reference, a fast image processing algorithm has to extract individual reference points from an image. From a grayscale (8 bit) image obtained from the camera, the first step is to segment the image into a binary format. Many thresholding techniques have been presented over the years, an extensive survey can be found in literature [130]. The main problem in detecting the features is separating the foreground from the background. Since two main intensity distributions can be distinguished, the solution is to find the most optimum threshold value. A few clustering-based thresholding techniques are evaluated with respect to performance, threshold method and complexity (see Table 7.1).

The 'Isodata' method [120] is an iterative procedure based on foreground and background classification. The clustering converges when the difference between iterative threshold values becomes sufficiently small. The method is very similar to the K-means algorithm with the difference that the Isodata method allows for different numbers of clusters, while K-means assumes the number known a priori. Otsu's method is an optimum global thresholding technique in the sense that it maximizes the inter-class variance, and equivalently, minimizes the intra-class variance [108]. The 'Minimizing error' thresholding technique by Kittler and Illingworth [79] assumes grayscale (i.e., 8 bit) data and is designed to optimize the average pixel classification error rate directly, using either an exhaustive search or an iterative algorithm.

Table 7.1: Thresholding algorithm comparison

	Isodata	Min. error	Otsu
Method	k-means clustering	minimizing error	minimizing intraclass variance
Timing	fast	medium	fast
Complexity	high	medium	low
Applicable	yes	no	yes

From a thorough comparison, Otsu's method seems most advantageous for our application. Compared with the other two methods, it has the lowest complexity and is easiest to implement. The 'Isodata' method has similar performance in timing, however, the amount of parameters and the number of steps per iteration make it less preferable with respect to the real-time requirements. Furthermore, due to the large deviations in threshold value, compared to aforementioned methods, and the fact that it is much slower in computation time, the 'Minimizing error' algorithm is also not applicable.

The studied thresholding algorithms all compute a global threshold value, which is sufficient due to the small size of the image.

After the image is binarized, the next step is to detect and identify individual features while ignoring noise. By employing mathematical morphology (i.e., boundary following), an image processing algorithm iteratively locates each feature.The final step is then to compute the sub-pixel accurate centre location inside each feature. The two vertical edges of each structure are known from boundary following and used to determine a pixel accurate x-position for each of the 4 points (see Fig. 7.4):

$$\begin{aligned} hx[1] &= box.x + 0.2 * box.width, \\ hx[2] &= box.x + 0.8 * box.width. \end{aligned} \tag{7.12}$$

This is then the start point to localize the maximum edge gradient position in sub-pixel accuracy in vertical (y) direction. A local maximum is interpolated using five neighbouring points and calculating their gradient norm. Simply said, this is fitting three points (i.e., representing the edge gradient) to a quadratic equation and finding its maximum.

To determine which gradient approximation method is most beneficial, a comparison is made between several derivative operators (see Table 7.2). The overall performance of the four edge detectors is very similar, with exception of Roberts' operator. This is due to the simplicity of the kernel and the inhibition of suppressing noise. Because the remaining operators have similar output, the simplest (i.e., symmetric) of all three was chosen for gradient approximation on OLED cell edges.

With $L = \{l(j_p)|j_p \in \mathbb{Z}\}$ an infinite line of pixels with a peak at coordinate 0 corresponding to the middle pixel $l(0)$ and ∇ a general derivation operator (symmetric) the gradient norm becomes:

$$\begin{aligned} a &= |\nabla l(-1)|, \\ b &= |\nabla l(0)|, \\ c &= |\nabla l(1)|. \end{aligned} \tag{7.13}$$

Table 7.2: Derivative operator comparison

	Prewitt	Roberts	Sobel	Symmetric
Kernel size	3×3	2×1	3×3	3×1
Performance	high	low	high	medium
Noise suppression	medium	low	high	medium

The maximum of the parabola (i.e., the highest slope in intensity) passing through $(-1, a)$, $(0, b)$ and $(1, c)$ is now found by:

$$y_m = \frac{a - c}{2(a - 2b + c)}. \tag{7.14}$$

This maximum y_m is calculated for $hx[i]$ and its direct neighbouring pixel columns, i.e., $hx[i] - 1$ and $hx[i] + 1$, and the average of these three values is then passed as local y-maximum (see Fig. 7.5). From the 4 found sub-pixel accurate points, a line is fit from the two pairs of opposite points on both vertical and horizontal edges:

$$y_1 = a_1 x_1 + b_1, \quad a_1 = \frac{y_{m,rd} - y_{m,lt}}{x_{m,rd} - x_{m,lt}}, \quad b_1 = y_{lt} - x_{m,lt} a_1, \tag{7.15}$$

$$y_2 = a_2 x_2 + b_2, \quad a_2 = \frac{y_{m,ru} - y_{m,ld}}{x_{m,ru} - x_{m,ld}}, \quad b_2 = y_{ld} - x_{m,ld} a_2, \tag{7.16}$$

where the subscripts l, r, d and u denote left, right, down and up respectively for the location of the 4 points. The crossing of these lines then determines the final centre coordinates $\mathbf{p}_c = [p_{c,x},\ p_{c,y}]^T$ of the feature (Fig. 7.4):

$$p_{c,x} = \frac{b_2 - b_1}{a_1 - a_2}, \quad \text{and} \quad p_{c,y} = p_{c,x} a_2 + b_2. \tag{7.17}$$

The presented centre detection algorithm is designed for rectangular features. When the feature is square shaped, a similar method can be used to determine each centre location. For instance, the gradient approximation method can be applied in both directions (i.e., on all edges) with one or two approximations per edge, depending on available computation time and required accuracy. In Algorithm 7.4 all steps of the centre detection algorithm are summarized.

Obviously, a weighted centre of gravity calculation would be an easier and faster method to determine the location of a feature. However, this has the disadvantage that, whenever there is a slight deviation in intensity inside the feature, this affects the repeatability and outcome of the algorithm. In other words, the algorithm is not robust against lighting changes when similar feature conditions apply. The presented method uses only a few pixels, reducing the chance that a small lighting deviation affects the outcome of the complete algorithm.

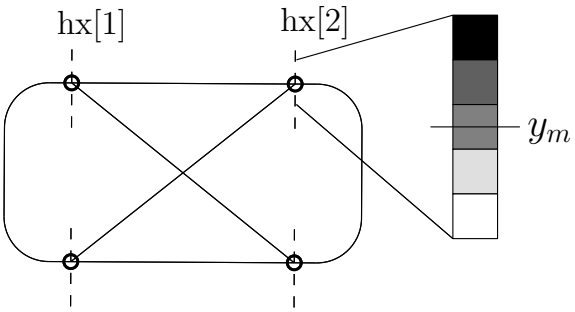


Figure 7.4: Outline of OLED cell with sub-pixel accurate points to determine centre point. hx[1] and hx[2] are determined from the vertical edges of the OLED structure (bounding box). The right grid shows the intensity value changes for the edge and the point where the slope has a maximum (y_m).

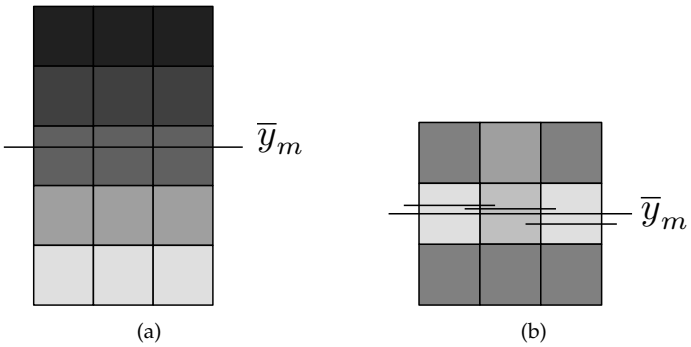


Figure 7.5: (a) shows the intensity values for each vertical edge pixel-line ($hx[i]-1, hx[i]$ and $hx[i]+1$). The long horizontal line represents the mean highest slope of all three vertical pixel-lines. (b) shows the derivative (1D symmetric) of (a) in vertical direction . The short horizontal lines represent the maximum of each vertical grid line, the long line represents the mean of the three short lines.

Algorithm 7.4 Repetitive Feature Localization

Input: $\mathcal{I}$ grayscale image ◁
Output: $\mathbf{p}_c$ feature centres ◁

1: Segment image with Otsu's method
2: **for all** features **do**
3: Follow boundary of feature
4: **for all** 4 edge points hx **do**
5: Calculate gradient $\nabla(l)$ of edge pixels
6: Calculate sub-pixel edge point $\bar{y}(m)$
7: **end for**
8: Use opposite points to form 2 lines
9: Crossing of lines determines feature centre $\mathbf{p}_c$
10: **end for**

7.2.3 Direct Trajectory Generation

The inkjet printing task that the visual control system has to accomplish can be stated formally as follows. In an ideal situation (constant pitch between OLED cells and perfect fixation), a constant velocity signal with fixed drop on demand print frequency would suffice. However, due to the design of the display (i.e., flexible display) or due to disturbances on the structure (e.g., heat, vibrations, fixation), this does not hold any more. Moreover, in order to obtain a higher throughput (i.e., more displays manufactured per hour), a constant velocity should be replaced by a velocity profile with fixed velocity on OLED cell centres and a higher velocity in between OLED cell centres. This results in a higher average velocity for the manufacturing of a display.

The motion of the print-head is designed such that at the centre of each OLED cell the velocity and acceleration is equal to a predefined value:

$$\begin{aligned} t_{drop} &= \text{constant, or} \\ t_{drop} &= t_e, \\ \dot{q}_f &= \nu_{drop}, \quad \text{and} \\ \ddot{q}_f &= \alpha_{drop}, \end{aligned} \tag{7.18}$$

in which, subscript f indicates final. t_e can be set as a constant time between cell centres or taken as (6.9), which indicates that the timing is constrained by either a maximum velocity or acceleration. This effectively enables the trajectory generation to obtain a higher average velocity while ensuring constraints on the centres of the OLED cells.

In order to not excessively excite the system, a $\mathcal{C}^2$ continuous point-to-point trajectory is chosen, for which the constraint vector $\mathbf{q}_c$ is obtained as

$$\begin{aligned} \mathbf{q}_c &= [\, q_I,\, q_f,\, \dot{q}_I,\, \ddot{q}_I,\, \dot{q}_f,\, \ddot{q}_f \,]^T, \\ &= [\, q_{k-1},\, q_f,\, \nu_{k-1},\, \alpha_{k-1},\, \nu_{drop},\, \alpha_{drop} \,]^T, \end{aligned} \tag{7.19}$$

in which image measurements are incorporated as $q_f = p_{c,x}$ (see (7.17)) for motion in x-direction.

The timing of each trajectory is thus fixed and a rate-based generation is effected (i.e., based on the visual measurements). This in effect implies that a trajectory is no longer designed with respect to a global kinematic constraint, but is rather designed to ensure the temporal constraint, i.e., arrival at fixed times with fixed local kinematic constraint:

$$t_I = 0, \quad \text{and} \quad t_f = t_{drop} - \Delta t_{sum}, \tag{7.20}$$

in which t_{drop} is taken from (7.18) and $\Delta t_{sum} = T_l n_{it}$ is the ascending trajectory time with T_l the local loop time with iteration count n_{it}. The routine for computing the trajectory online (see Algorithm 6.2) is proposed in Section 6.2.

As an OLED display typically has hundreds of OLED structures in sequence, a continuous succession of trajectories has to be generated with matching constraints at end- and start-point. More specifically, it has to hold that the start-point of a new trajectory matches with the end-point of the previous trajectory. When generating a trajectory with $\mathcal{C}^2$ parametric continuity, this should also hold for the velocity and acceleration at the end- and start-point.

The direct trajectory generator (DTG) generates a velocity trajectory to control the table to move from one OLED cell to the next one. For motion in x-direction, from initialization the left most OLED cell (of the three horizontal cells in the field of view) is tracked and motion is generated to move it to the print-head position $p_{print,x}$ with predefined constraints. When this position is crossed (i.e., before the next iteration), a new trajectory is designed. Similarly, the print-head should be triggered by the event of an OLED cell centre crossing $\mathbf{p}_{print}$. However, as this crossing most likely occurs in between iterations, the exact trigger time t_{print} is predicted by as:

$$t_{print} = t_{k+1} + \Delta t_{cc} - t_{travel} - t_{rem}, \tag{7.21}$$

$$\Delta t_{cc} = \frac{p_{print,x} - \hat{p}_{c,x,k+1}}{\hat{p}_{c,x,k+2} - \hat{p}_{c,x,k+1}} T, \tag{7.22}$$

where $t_{k+1} + \Delta t_{cc}$ is the estimate of the time until a cell centre crossing, t_{travel} is the delay due to the travel time of the droplet through the air (see Section 7.1.3 and (7.3)) and t_{rem} is the remaining delay (e.g., due to data communication, position difference print-head and image centre). The future position and velocity are predicted with an α-β filter [70] as:

$$\begin{cases} p_{c,x,k} &= \hat{p}_{c,x,k-1} + T_l \hat{\dot{p}}_{c,x,k-1}, \\ \hat{\dot{p}}_{c,x,k} &= \hat{\dot{p}}_{c,x,k-1}, \\ \hat{r}_{c,x,k} &= p_{c,x,k} - \hat{p}_{c,x,k}, \\ \hat{p}_{c,x,k} &= \hat{p}_{c,x,k} + \alpha \hat{r}_{c,x,k}, \\ \hat{\dot{p}}_{c,x,k} &= \hat{\dot{p}}_{c,x,k} + \beta / T_l \hat{r}_{c,x,k}. \end{cases} \tag{7.23}$$

The future positions are then found as:

$$\begin{cases} \hat{p}_{c,x,k+1} &= \hat{p}_{c,x,k} + T_l \hat{\dot{p}}_{c,x,k}, \\ \hat{p}_{c,x,k+2} &= \hat{p}_{c,x,k} + 2T_l \hat{\dot{p}}_{c,x,k}. \end{cases} \tag{7.24}$$

The algorithm to detect both the trigger for DTG as well as for the print-head is described in Algorithm 7.5.

Algorithm 7.5 Event trigger of DTG and print-head

Input: $\mathbf{p}_c = [p_{c,x},\ p_{c,y}]$, $\mathbf{p}_{print}$ OLED cell centres and print-head position ◁
Output: Trigger signal for DTG and print-head

```
Initialize: track left most cell
for each iteration do
    if (p_{c,x} < p_{print,x}) and (p_{c,x} + p̂) > p_{print,x} then
                    If crossing p_{c,x} = p_{print,x} will occur before next iteration ◁
        Trigger DTG                                   Generate new trajectory ◁
        Trigger print-head                  determine t_{print}, see (7.21) ◁
    end if
end for
```

7.2.4 Visual Control Law

The motion of the xy-table is velocity controlled due to the importance of a fixed velocity at each OLED cell centre. This is necessary to guarantee a fast cycle time when manufacturing a display. In order to take into account the differences in pitch with the designed trajectory, a velocity PID controller (3.20) as presented in Section 3.3.1 is employed (see also Fig 7.6). Additionally, a feedforward compensation term for the mass and the friction of the table is added as presented in Section 3.2. The friction is modelled with a Coulomb and viscous friction term as

$$\mathbf{F}_w(\dot{\mathbf{q}}_d) = \mathbf{F}_v\dot{\mathbf{q}}_d + \mathbf{F}_c\mathbf{sgn}(\dot{\mathbf{q}}_d), \tag{7.25}$$

where $\mathbf{F}_v > 0$ is the matrix containing viscous friction terms, $\mathbf{F}_c > 0$ is the matrix containing Coulomb friction terms and $\mathbf{sgn}(\dot{\mathbf{q}}_d)$ denotes the vector containing the signum operator as presented by (3.2). This classical model of friction (see e.g., [6] for a survey on friction models, [13] for an overview of friction compensation in robotics, as well as Section 3.2) is sufficient to compensate for the major friction disturbance as occurs in the prescribed task. This will be shown in the experimental results in Section 7.3.4.

Including the mass feedforward term, the complete feedforward control term can be described as

$$\mathbf{F}_{ff} = \hat{\mathbf{M}}_t\ddot{\mathbf{q}}_d + \mathbf{F}_w, \tag{7.26}$$

where $\hat{\mathbf{M}}_t$ is the estimated mass matrix of the table and $\mathbf{F}_w$ is taken from (7.25).

The feedback obtained from vision are position measurements, which subsequently are converted to velocity estimates. Such velocity estimate is determined as:

$$\hat{\dot{x}}_k = \frac{1}{n_d}\sum_{i=1}^{n_d}\left(x_{k,i} - x_{k-1,i}\right), \tag{7.27}$$

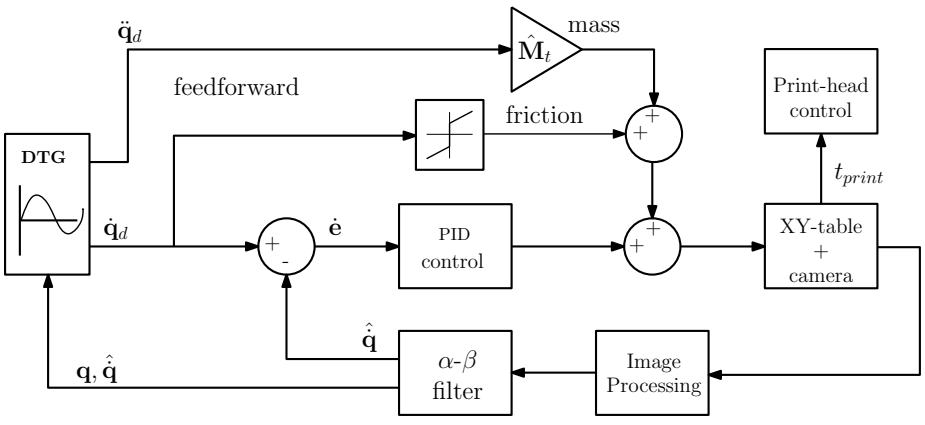


Figure 7.6: Control scheme for planar product pattern-based visual servoing. The DTG block generates a trajectory online based on image measurements. Trajectory tracking is achieved with a velocity PID controller, with an additional feedforward term to compensate for the mass and the friction of the motion stage.

where n_d is the number of detected OLED cells in the field of view. This estimate is valid as the velocity is expressed in pixels per frame $[px/frame]$. Finally, the measurement noise is filtered out by using an α-β filter.

7.3 Experimental Results

The algorithms for visual and trajectory processing as presented in previous sections are integrated into an experimental setup for evaluation. This setup is explained, and results are shown for camera calibration and pattern detection. Following, results are given for the methodology of direct trajectory generation and control, as well as a comparison with a constant trajectory reference.

7.3.1 Experimental Setup

The experimental setup as developed for experiments can be seen in Fig. 7.7, a schematic representation in Fig. 7.8. The system consists of 2 linear actuators (Dunkermotoren ServoTube STA11), a stationary camera (SVS-Vistek-340) and an FPGA (Xilinx Virtex-5 xc5vsx50t) for processing. Compared to gear-reduced systems (e.g., ball-screw actuators), direct-drive systems are known to have reduced friction, no backlash and high stiffness. This setup was developed by Jeroen de Best as part of the Fast Focus on Structures (FFOS) project and presented in [34]. The embedded processing system (camera, FPGA and processing) was developed by Zhenyu Ye as part of the Embedded Vision Architecture (EVA) project and will be presented in [156]. The camera sends monochrome images with a frame rate of 1600 fps and image size of 160×100 pixels directly to the FPGA via a CameraLink interface. Combined with a 1.5x magnifying lens (Opto-engineering MC1.50x) the images have a pixel size of 4.5 $[\mu m]$. A coaxial lighting system is incorporated which has the advantage that the light that enters the camera sensor is reflected mainly from axial illumination. This is due to the use of a beamsplitter which directs illumination from a power LED source downwards onto the OLED substrate which subsequently is reflected

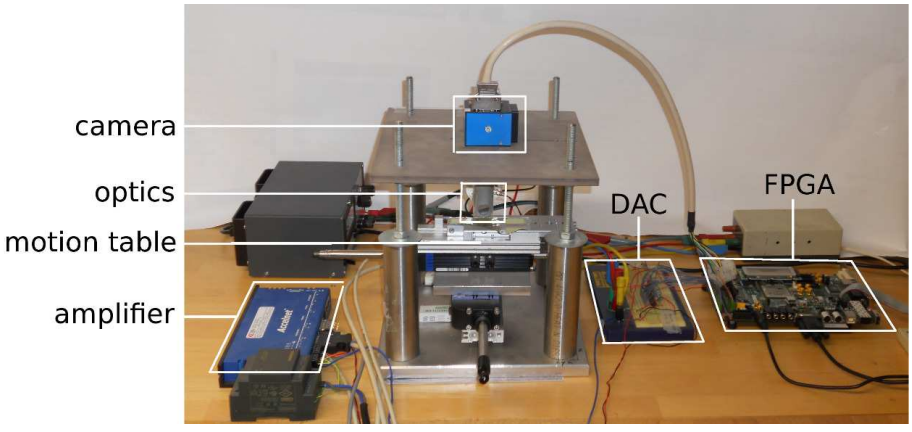


Figure 7.7: Experimental setup of the visual control system for product pattern-based visual servoing.

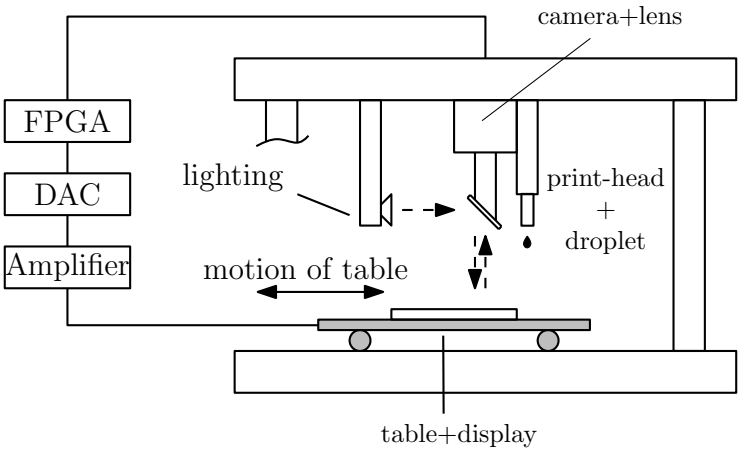


Figure 7.8: Diagram of the visual control system for inkjet printing.

up into the camera. The camera and lighting are static, while the motion table including the OLED display are actuated by the two linear motors. As one pixel is represented by one byte, the effective network load for transferring the images at 1.6 kHz is roughly 26 MB/s (see also Fig. 7.1). As the proposed control method is a direct visual servoing approach, the control frequency is similar to the camera frequency, i.e., 1600 Hz. It has to be mentioned that feedback is solely obtained from visual measurements, the local motor encoders which are present in the linear motion system are *not* used.

With these parameters the trade-off relationship between magnification and trackability as defined by Ogawa et al. in [107] is found as $n_f M = \frac{fps}{v_r} \approx 12$, for a target velocity of $v = 28.8\ [mm/s]$ and a target length of $l = 220\ [\mu m]$. Compared to the results of Ogawa et al., who obtained $n_f M > 20$, this is a fairly decent value.

7.3.2 Implementation Details

The complete visual processing pipeline, including control algorithm, is executed on an embedded processing platform. Visual processing is accelerated and optimized on a Field-Programmable Gate Array (FPGA) to utilize parallel processing as much as possible. The image sensor is directly connected to the processor such that processing starts directly when the first line of the image is received. Fig. 7.9 and Table 7.3 show the timing breakdown of the complete image pipeline. It shows that the update rate is dominated (i.e., limited) by the transfer (readout) of image data to the processor.

The resource usage of the whole system is less than 30% of a mid-range FPGA (Xilinx Virtex-5 xc5vsx50t). The remaining resources could be used to even further accelerate the vision pipeline, however, as shown in the timing analysis of Fig. 7.9 the start-to-end delay is dominated by the readout time. Unfortunately, these cannot be accelerated by an improved FPGA implementation. Furthermore, utilizing an even faster frame rate should be supported by the camera, which might not be the case, and a shorter exposure time brings further difficulties regarding sufficient light for image exposure. The imple-

mentation details of the vision pipeline on FPGA[3] is not part of this work, but can be found in [157], [158], [159] and [156].

Table 7.3: Timing of vision pipeline

	Time
Camera frame rate	1.6 [kHz]
Camera update	∼600 [μs]
Exposure	50 [μs]
Readout	500 [μs]
Visual Processing	950 [μs]
Start-to-end delay	1000 [μs]

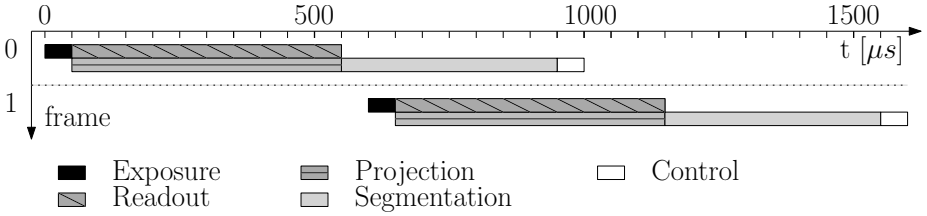


Figure 7.9: Timing pipeline for the vision and control algorithm on FPGA. As can be seen in the figure, processing starts directly when the first line of the image is received. Foremost, the trade-off between image size and update rate determines the delay between exposure and control action.

7.3.3 Calibration and Detection Results

The derived planar calibration procedure is tested with two different calibration patterns; an industrial high accuracy calibration grid and the OLED display itself which is also a repetitive pattern.

Calibration

To evaluate the necessity for camera calibration, the position error due to radial distortion for several values of κ_1 is shown in Fig. 7.10. The exponential growth when further away from the centre proves the necessity for position correction when using even a highly accurate lens. For instance, if 1% distortion at 200 pixels centre offset results in 2 pixels deviation, then a maximum of 0.1% distortion (a common distortion value for low-distortion lenses) will give a 0.2 pixel mismatch at the same centre offset. When keeping in mind that each pixel can be as large as 4.5 $[\mu m]$, a measurement error of 0.9 $[\mu m]$ (i.e., 20%) is added to the measurement accuracy.

Camera calibration is performed with single images for two patterns (i.e., an OLED product pattern image and an industrial calibration pattern image). Multiple calibration experiments were carried out with the patterns shifted

[3]These developments where carried out in close collaboration with Zhenyu Ye.

slightly, however, since these are continuous grids, the patterns fill the complete field of view (i.e., 640×480 $[px]$). Due to image processing and lighting conditions, the measurements of the product pattern contain more noise (i.e., higher standard deviation and maximum error), which subsequently causes a less uniform grid then the industrial pattern. This results in slightly different calibration parameters as to be seen in Table 7.4. κ_1 shows to be a very small value, confirming a very precisely machined lens, as stated by the manufacturer. Standard error evaluation measures (mean, standard deviation, maximum error) also verify the lesser accuracy for the product pattern, due to the higher complexity in measurements.

As the image for control only extends a maximum of 80 $[px]$ from the centre, this results in a 0.03 % image error mismatch, which, with a pixel size of 4.5 $[\mu m]$ is 0.15 $[\mu m]$. This value is much smaller then the noise observed from measurements (i.e., $3\sigma = 2.55$ $[\mu m]$) and implies that a calibration procedure is not necessary.

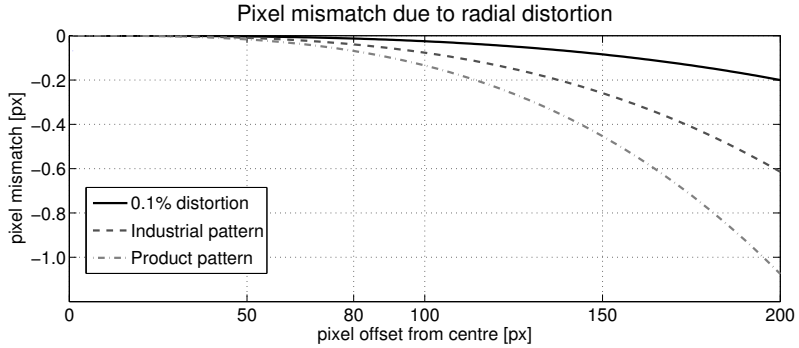


Figure 7.10: Mismatch of pixel locations for different radial distortion values.

Table 7.4: Calibration results

Pattern	Industrial	Product
κ_1 $[\mu m^{-2}]$	-3.84e-10	-6.72e-10
magnification: $M_0 = T_z/f$	4.03	4.26
image error		
mean	0.6 $[\mu m]$	1.3 $[\mu m]$
standard deviation	0.3 $[\mu m]$	0.7 $[\mu m]$

Feature Detection

Each 160×100 $[px]$ image (i.e., $\sim$0.3 $[mm^2]$) contains 3×5 OLED cells. Fig. 7.11 shows a close-up of the result of the image processing steps as explained in Section 7.2. The measurement noise has a standard deviation of $\sigma = 0.18$ $[px] = 0.85$ $[\mu m]$. As such, 99.7% of the measurements lie within the deviation of $3\sigma = 0.56$ $[px] = 2.55$ $[\mu m]$, which is quite a substantial value considering the required accuracy of 10 $[\mu m]$.

Figure 7.11: Output of the centre detection algorithm. Left figure shows the output after thresholding with Otsu's method. Right figure shows the found OLED cells outlined with a rectangle. On horizontal lines the points are shown where the optimal vertical edges are detected.

7.3.4 Trajectory Generation Results

As explained in the previous sections, the control method consists of the tracking of a trajectory which is generated online, directly based on visual measurements. This section presents results of the proposed method and compares this with a constant velocity reference trajectory.

Results for Constant Trajectory Tracking

As initial experiment, the response of the control system to a step input is evaluated (see Fig. 7.12). Control of this step input consists of a simple PID controller with experimentally tuned control parameters. It can be seen that the actuators are subject to considerable friction. This friction originates from the internal dry bearing of the linear actuators and is dominated by viscous (dynamic) friction and Coulomb (static) friction. For viscous friction, this can be identified by the relatively low rate of acceleration (i.e., slope of velocity)[4]. Static friction (or stiction) is noticeably present at velocities close to zero, as it takes several iterations before the platform starts moving. Finally, the constant velocity reference reveals that the friction of the motion system is also position dependent. This can be identified in the figure as the steady-state behaviour in the range of $0.075 - 0.15$ $[s]$ compared to the steady-state behaviour in the range of $0.15 - 0.3$ $[s]$ is not similar. An explanation for this could be that the latter range is subject to a greater amount of viscous friction.

Results for Point-to-point Trajectory Tracking

To show the effectiveness of using a near-repetitive pattern for motion control the trajectory is designed as follows. From standstill a smooth velocity is designed to a fixed velocity (i.e., $v_{drop} = 4\ [px/frame] = 28.8\ [mm/s]$) and acceleration (i.e., $a_{drop} = 0\ [px/frame^2]$) at an OLED cell centre. The velocity in between the cell centres is chosen higher to obtain a higher printing throughput, and obtained by setting a maximum velocity for each velocity profile. This results in an average velocity for the trajectory of about $\bar{v} = 5\ [px/frame] = 36\ [mm/s]$, while for a constant velocity trajectory this would be equal to the velocity at the OLED cell centres, i.e., $v_{drop} = 4\ [px/frame] = 28.8\ [mm/s]$. This directly motivates the use of a online generated trajectory for motion control as a speed increase for printing of 25% is easily obtained. Depending on

[4]Of course the mass of the system also plays a role in this.

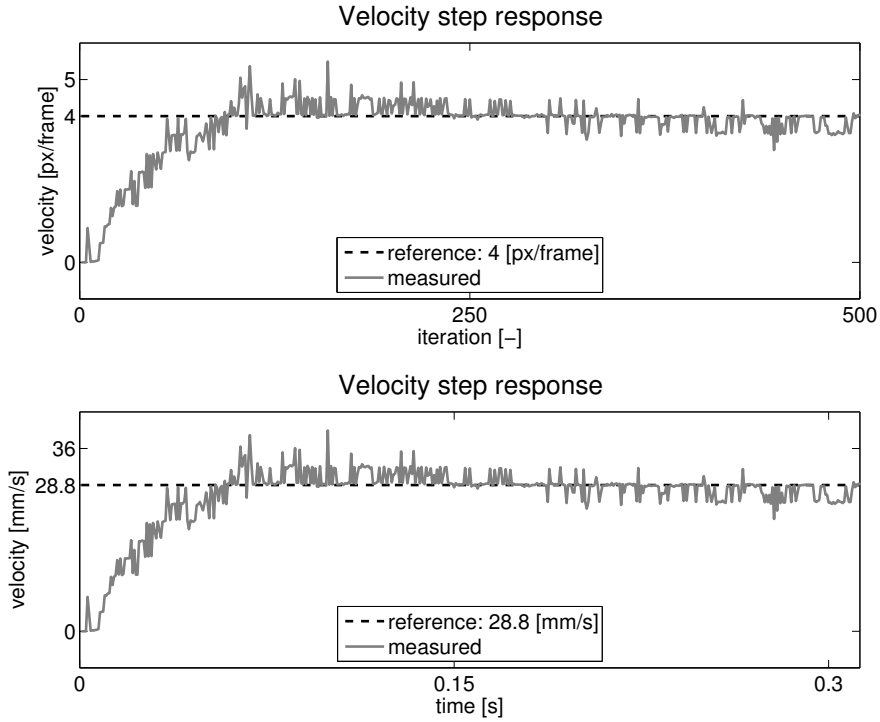


Figure 7.12: Response of the visual control system (i.e., with PID control) on a velocity step input. Upper and lower figure show the response with dimensions for velocity in pixels per frame and millimetres per second respectively.

the limits of the actuators, this can be increased even more. Finally, it has to be mentioned that the parameters of the PID controller are retuned for experiments with different control structure. To be more specific, the parameters of the PID gains for control with feedforward action compared to control without feedfoward action, are different.

Fig. 7.13 presents the tracking results of the online generated trajectory with only a PID controller. It can be seen, at the start of the trajectory, that the static friction (stiction) again takes several iterations to overcome. Furthermore, it shows that the viscous friction creates a delay between the reference velocity and the real (or estimated) velocity. This is particularly visible at relatively low ($< 4\ [px/frame] =< 28.8\ [mm/s]$) velocities. Furthermore, the friction of the system causes large disturbances at OLED cell centres (i.e., the local minima where $v_{drop} = 4\ [px/frame] = 28.8\ [mm/s]$), and is most likely caused by the switching of sign of the acceleration (i.e., negative to positive). This delay and disturbance can be compensated for with a feedforward term which includes the mass of the table as well as a friction compensation term as proposed in Section 7.2.4. A final friction effect can be seen in the velocity range of $0 - 4\ [px/frame] = 0 - 28.8\ [mm/s]$ and reveals a stick-slip-like phenomena. This spontaneous jerking motion is caused by alternating sticking and sliding regimes in the lower velocity range.

Fig. 7.14 presents the tracking of the online generated trajectory with a PID controller and the mentioned compensation terms. The parameters for friction compensation are obtained via the method presented in [72] and via experimental tuning. In particular, the individual parameters of the Coulomb and viscous friction (i.e., $\mathbf{F}_c$ and $\mathbf{F}_v$) are estimated based on open-loop measurements. A velocity ramp trajectory is executed as reference and from the resulting measurement response (i.e., velocity vs. time) an initial estimate of the friction parameters can be retrieved. This initial guess is then tuned online (i.e., closed-loop) to obtain a better motion performance. The mass of the system $\hat{\mathbf{M}}_t$ is estimated by weighing the motion system and tuned to obtain a decent performance. It can be seen that by compensation for the mass of the system as well as the viscous friction, the measured (or estimated) velocity follows the reference velocity more close. This is especially visible at relatively low ($< 4\ [px/frame] =< 28.8\ [mm/s]$) velocities. In the same velocity range, however, the stick-slip-like phenomenon is still visible. A compensation for this is not incorporated as the performance of motion control in this velocity range is not particularly of interest. This also holds for the stiction effect close

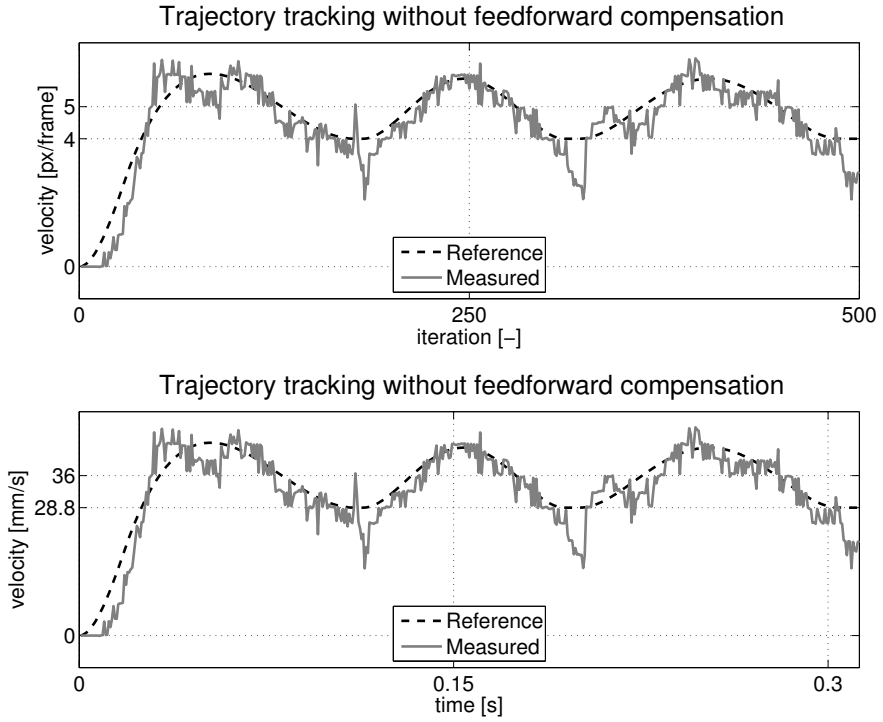


Figure 7.13: Velocity trajectory control with DTG without compensation ($\dot{e}_{rms\_no\_FF} = 0.57\ [px/frame] = 2.56\ [\mu m/frame] = 4.1\ [mm/s]$). Especially at OLED cell centres (local minima where $v_{drop} = 4\ [px/frame] = 28.8\ [mm/s]$) only a PID controller proves not to be sufficient. Upper and lower figure show the tracking results with dimensions for velocity in pixels per frame and millimetres per second respectively.

to zero velocity.

The performance of trajectory tracking is evaluated by the root mean square (RMS) of the error velocity in Cartesian space. Without compensation of the friction and the mass of the system this is found as: $\dot{e}_{rms\_no\_FF} = 0.57\ [px/frame] = 2.56\ [\mu m/frame] = 4.1\ [mm/s]$ (see Fig. 7.13). When the compensation scheme is included the error RMS value is found as $\dot{e}_{rms} = 0.40\ [px/frame] = 1.8\ [\mu m/frame] = 2.88\ [mm/s]$, indicating a clear advantage of the compensation scheme (see Fig. 7.14).

A different important performance measure is the actual velocity on the centre of the OLED cell. As can be seen in Fig. 7.13, there is a relatively large error between the reference velocity and the actual velocity on the OLED cell centres (i.e., local minima where $v_{drop} = 4\ [px/frame] = 28.8\ [mm/s]$), due to a poor controller. Fig. 7.14 shows that with a properly designed controller (i.e., including the feedfoward compensation) this error is clearly lower. Even though the velocity response has some delay, this amount of delay stays within bounds (i.e., $\pm 2.9\ [px/frame] = 20.9\ [mm/s]$ as determined in Section 7.1.3) when considering the moment of printing: $\dot{e}_{rms} = 0.40\ [px/frame] = 2.88\ [mm/s]$.

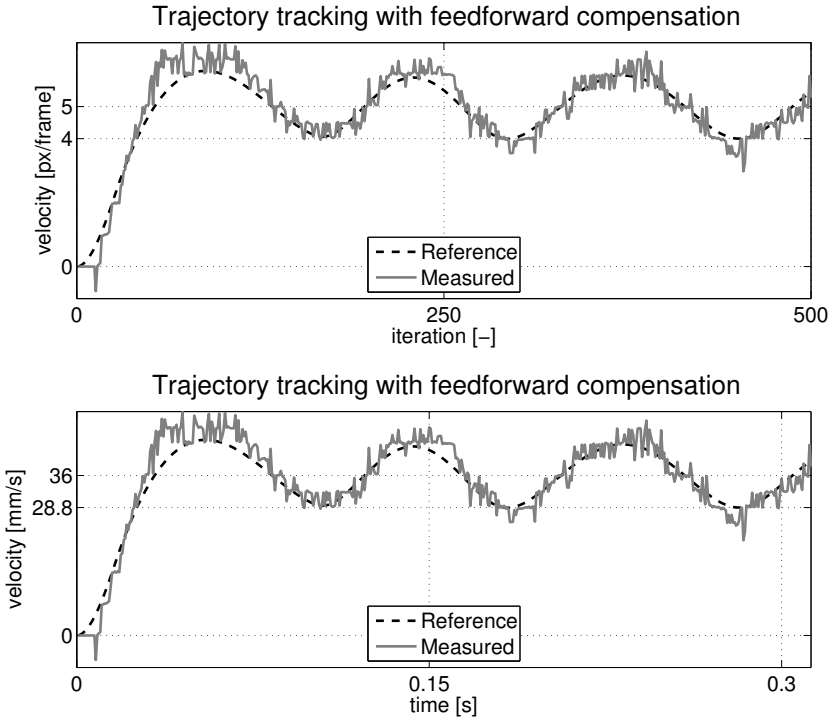


Figure 7.14: Velocity trajectory control with DTG with feedforward compensation ($\dot{e}_{rms} = 0.40\ [px/frame] = 1.8\ [\mu m/frame] = 2.88\ [mm/s]$). The estimated velocity stays closer to the reference velocity compared to DTG without compensation. Upper and lower figure show the tracking results with dimensions for velocity in pixels per frame and millimetres per second respectively.

Even though a fair amount of noise can be seen in the figures, it must be noted that this does not necessarily originate from control. The estimation process of the velocity itself acts as a clear noise source. Therefore, in order to obtain a decent motion control performance, an α-β filter [70] is incorporated to filter out measurement noise (i.e., all experiments include this filter). In fact, if this α-β filter would be omitted, the motion of the visual control system will become unstable. Fig. 7.15 shows an experiment with a poorly tuned α-β filter, which results in delay and overshoot: $\dot{e}_{rms\_poor\_\alpha\beta} = 0.64\ [px/frame] = 2.86\ [\mu m/frame] = 4.6\ [mm/s]$. Finally, it has to be mentioned that the estimated velocity shown in all figures is taken directly after the output of the plant (i.e., before the α-β filter, see Fig. 7.6), to show the real measurements.

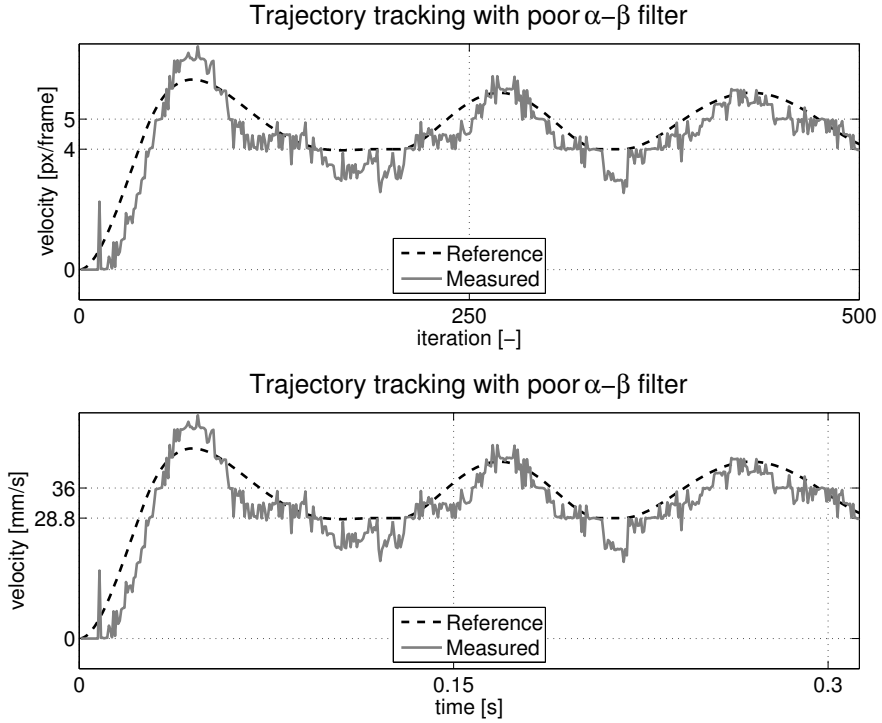


Figure 7.15: In order to filter out the measurement noise and obtain a better velocity estimation, an α-β filter is incorporated for all experiments. To show the necessity of this, this figure shows a poorly tuned filter. A clear delay and overshoot can be seen which results in a decreased motion performance: $\dot{e}_{rms\_poor\_\alpha\beta} = 0.64\ [px/frame] = 2.86\ [\mu m/frame] = 4.6\ [mm/s]$. Upper and lower figure show the tracking results with dimensions for velocity in pixels per frame and millimetres per second respectively.

7.4 Summary

This chapter presented the application and implementation details of product pattern-based visual servoing. The method of using the product pattern as visual encoder for motion control is motivated by regarding the current state-of-the-art in visual motion control and industrial inkjet printing. As comparison, the trade-off between image size and update rate is shown for several high-speed vision systems. For industrial inkjet printing (i.e., OLED display manufacturing by printing a droplet into each display cell), the current state-of-the-art assumes that the pitch (or time) between individual printing actions is fixed and a constant print-frequency combined with motor-encoder feedback for control is sufficient for the manufacturing of displays. However, when this assumption no longer holds (i.e., a varying pitch due to a flexible display) current methods no longer suffice. This is mainly due to the fact that the product (i.e., the location for printing) is not directly measured. The proposed method takes this into account by designing a trajectory online based on direct visual measurements. The developments for this include a camera calibration method, a feature detection method for the detection of individual display elements and the visual control method with direct trajectory generation. In more detail, it is shown that, due to the lens, the narrow field-of-view and the measurement noise, a calibration for lens distortion is not necessary. The detection of individual display elements is presented for a 160×100 $[px]$ image (i.e., ~ 0.3 $[mm^2]$) containing 3×5 OLED cells where each pixel is 4.5 $[\mu m]$ square sized. This allows for a camera and control update rate of 1600 $[Hz]$. The measurement noise shows to have a standard deviation of $\sigma = 0.18$ $[px] = 0.85$ $[\mu m]$. The method for visual control consists of velocity trajectory tracking, where the trajectory is generated online based on the position of individual OLED cells. As such, at each iteration the next state of the trajectory is generated based on a predefined 5^{th} order point-to-point polynomial trajectory with predefined (i.e., 4 $[px/frame] = 28.8$ $[mm/s]$) velocity on OLED cell centres, and a higher velocity in between OLED cell centres. This allows for a higher average velocity for the overall motion, which would be impossible for a constant velocity trajectory if a similar quality of printing should be ensured. The precise moment of printing is predicted by an α-β filter as there still exists a mismatch due to the delay of the system and the spatial difference between print-head and image centre. The feedback for velocity trajectory tracking is obtained by estimating the velocity of the OLED cells in between frames. Furthermore, a feedforward control action is added to compensate for the mass and friction (i.e., a Coulomb and viscous term) of the motion system. This complete framework is implemented on an experimental setup consisting of a 2D planar table, a static camera and a FPGA for processing. The details of this setup as well as the implementation of the developed methodologies are explained and results have been shown which motivate the proposed method. In particular, the root means square of the velocity error trajectory is found as $\dot{e}_{rms} = 0.40$ $[px/frame] = 2.88$ $[mm/s]$, indicating a clear advantage of the compensation scheme and the effectiveness of this visual control method.

CHAPTER 8

Vision-Based Obstacle Avoidance

Abstract. This chapter presents the application and implementation details of vision-based obstacle avoidance. In particular, vision-based obstacle avoidance is developed by integrating direct trajectory generation into Cartesian motion design. For comparison, a basic reactive motion scheme is presented that generates motion on a path planning level based on the distance towards an obstacle. In addition, for a redundant manipulator, avoidance of the self-motion of the manipulator towards certain objects is incorporated into the kinematic control design.

8.1 Introduction and Motivation

With the increasing demand of integrating robotics into every day life and industry, safety requirements are still a driving factor. Especially in a human-centred environment, robot motion has to be as smooth as possible and safety has to be guaranteed. This implies a safe replanning of motion when obstacles are detected. As current state-of-the-art approaches differentiate between obstacle avoidance (i.e., path planning) and traditional motion control (i.e., trajectory planning), the problem of avoidance is usually solved by designing a new path (see also Chapter 3 and Section 6.1.1). This means that predefined kinematic constraints for the trajectory are not taken into account for obstacle avoidance and only a reactive motion guides the robot away from objects (e.g. potential field, roadmap [88]).

This chapter presents the application and implementation details of obstacle avoidance of a n-DOF manipulator ($n \geq 6$) in Cartesian space and considers the direct trajectory generation method as proposed in Chapter 6 as novel solution to this problem. In particular, the approach generates a new trajectory at every iteration, even when no obstacle is detected. Direct trajectory design is presented for point-to-point and multi-point positioning, for different levels of constraints. This enables the possibility of incorporating different trajectory shapes in real-time motion design.

This novel direct trajectory planning approach is compared to a reactive path planning approach which designs motion without any kinematic constraints. A potential field is developed which guides motion away from an obstacle by distributing the weight for each task (i.e., either positioning or obstacle avoidance) depending on the distance towards an obstacle.

In addition, robotic manipulators are commonly designed with more degrees of freedom than necessary for task execution in Cartesian space (i.e., DOF

of a manipulator > 6). This gives the manipulator the ability to design and execute complex motion with respect to a secondary task as is explained in Section 3.3.2. As the self-motion of a manipulator (i.e., motion of the manipulator while keeping the end-effector fixed at a certain pose) can also collide with obstacles (or itself), the redundancy property is now exploited for the avoidance of these obstacles. Three different avoidance indices, as presented in Section 3.3.2 are experimentally validated and show the effectiveness of avoidance for the self-motion of the manipulator.

8.1.1 Task and Kinematic Constraints

The task of robotic manipulation is divided in two separate motion solutions. This includes the solution to generate motion for the end-effector and the solution to exploit motion for the redundant degrees of freedom. In particular, this is explained in Section 3.3.2, where the solution for redundancy does not generate motion for the end-effector.

The considered task involves a point-to-point planar (i.e., XY-plane in Cartesian space, see Fig. 8.1) positioning task, where orientation is not taken into account. Due to the difficulty in determining a 3D translation difference from a monocular camera, the avoidance scheme is designed and experimentally verified in 2D Cartesian space. This difficulty originates from the fact that an image sensor is essentially a 2D measurement array. As such, a homography-based approach determines only a scaled translation and a simple object detection approach (e.g., color blob detection) will only give accurate measurements parallel to the 2D image plane (i.e., depth is only estimated). However, as methods exists that can give highly accurate 3D measurements (e.g., by stereo-vision algorithms or depth sensors like the Kinect), this approach can be easily extended to full 3D obstacle avoidance.

Considering the task, only the initial and final position (a point in Cartesian space) of the task are known. The position of a static object which should be avoided is measured and thus obtained during runtime. The velocity and acceleration of the initial and final point are defined to be zero ($\dot{x}_I = \ddot{x}_I = \dot{x}_f = \ddot{x}_f = 0$). Furthermore, the general motion for positioning is constrained by either a velocity or acceleration constraint (i.e., limitations of the manipulator). Obviously, for path planning these kinematic constraints are not taken into account.

Finally, kinematic redundancy is exploited on the basis of several geometric indices (i.e., a distance towards a point, a line and a plane). These variables are geometric entities in Cartesian space and represent an obstacle (i.e., a perpendicular distance towards it) for avoidance (see also Section 3.3.2).

8.2 Obstacle Avoidance

In this section the mathematical details of obstacle avoidance by both path planning and trajectory planning is presented. The kinematic control law is schematically depicted and the additional kinematic redundancy resolution is discussed.

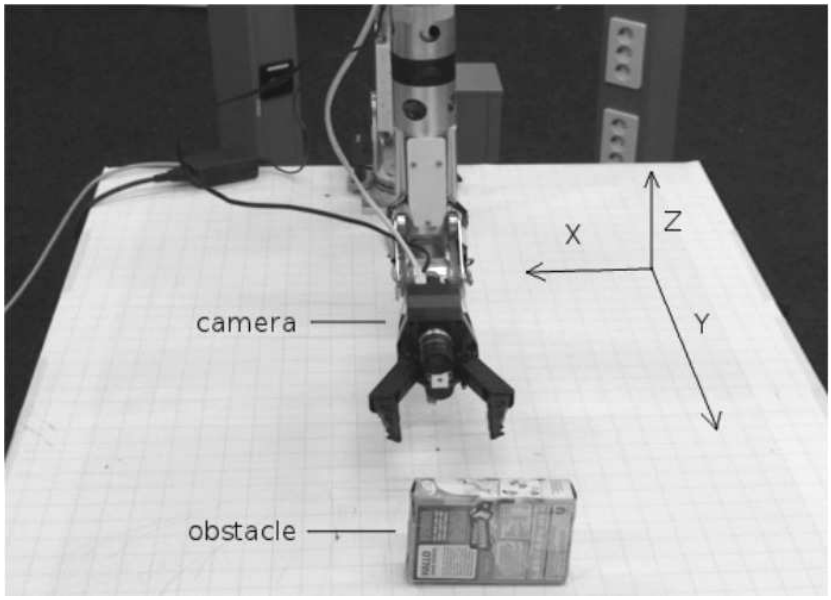


Figure 8.1: The Cartesian plane for obstacle avoidance is the XY-plane.

8.2.1 Path Planning

In Section 2.6 a brief overview was given concerning path planning. Methods such as Probabilistic Roadmaps (PRM) or Rapidly-exploring Random Trees (RRT) are a popular approach to be implemented on real robotic systems. However, despite the fact that these methods are sampling-based (i.e., developed to be implemented in real-time), motion planning is commonly the only task that has to be executed. When another computationally intensive task has to be executed at the same time (i.e., visual processing), compromises have to be made to guarantee real-time performance. In this respect, it was chosen to utilize a potential field-based approach which is computationally low demanding. Potential field-based approaches typically consist of an attractive or repulsive function that pulls or pushes a robot towards a goal or away from an obstacle (see e.g., [76] or [87]). The method presented here is very similar to the concept of potential fields and defined to achieve a smooth transition between obstacle avoidance and target following. Two similar techniques can be found in [25] and [109].

The tasks which are pursued are a smooth transition between obstacle avoidance and target following and is defined as follows (see also Section 3.3).

$$\dot{\mathbf{q}} = \mathbf{J}_a^{\#}(\mathbf{w}\dot{\mathbf{x}}_{oa} + (\mathbf{I}_w - \mathbf{w})\dot{\mathbf{x}}_t) + (\mathbf{I} - \mathbf{J}_a^{\#}\mathbf{J}_a)\dot{\mathbf{q}}_0. \tag{8.1}$$

In this, a division is made between end-effector control (i.e., $\dot{\mathbf{e}} = \mathbf{J}_a\dot{\mathbf{q}}$, where there error vector is $\dot{\mathbf{e}} = \dot{\mathbf{x}} = \mathbf{w}\dot{\mathbf{x}}_{oa} + (\mathbf{I}_w - \mathbf{w})\dot{\mathbf{x}}_t$) and redundancy control (i.e., $(\mathbf{I} - \mathbf{J}_a^{\#}\mathbf{J}_a)\dot{\mathbf{q}}_0$). End-effector control is performed with the minimum-norm solution of the joint velocities, while redundancy control makes use of the homogeneous solution. A smooth transition between obstacle avoidance velocity $\dot{\mathbf{x}}_{oa}$ and target positioning velocity $\dot{\mathbf{x}}_t$ is created by weighting each velocity according to the vicinity towards a certain obstacle.

As mentioned earlier, due to the difficulty of determining an accurate 3D translation difference, the path planning scheme is shown in experiments for 2 Cartesian translational degrees of freedom. Therefore, in 2 dimensions, $\mathbf{I}_w = [1, 1, 0, 0, 0, 0]$ and $\mathbf{w} = [w_u, w_v, 0, 0, 0, 0]$ where w_u and w_v are defined as a double sigmoid function as

$$w_l = w_{s,l} \left[\tanh\left(\frac{d_l + d_{o,l}}{k_s} \right) - \tanh\left(\frac{d_l - d_{o,l}}{k_s} \right) \right], \tag{8.2}$$

for $l \in \{u, v\}$ (see Fig. 8.2).

d_l is the distance in pixels towards an obstacle and thus decides the weight of both the obstacle avoidance velocity $\dot{\mathbf{x}}_{oa}$ as well as the target positioning velocity $\dot{\mathbf{x}}_t$. Furthermore, k_s is the slope of transition between the two velocities and $d_{o,l}$ is a parameter that determines the center location of the slope k_s, which, when equal for both hyperbolic tangent functions, returns a symmetric function. When $w_{s,l} = 0.5$, the function has a range $w_l : \mathbb{R} \rightarrow [0, 1]$. An extra weight $w_{s,l} > 0.5$ can be added to create an overshoot around an obstacle. This gives more space to avoid collisions, since the obstacle cannot be perceived if it is next to the end-effector. A side effect is that the extra weight works both ways (i.e., an added negative $\dot{\mathbf{x}}_t$), which can be solved, however, with simple heuristics.

This method is essentially similar to a potential field-based approach. Depending on the location of the obstacle in the image (i.e., distance in pixels from center) a certain weight is distributed over the both tasks (i.e., positioning or obstacle avoidance). This creates a motion which guides the end-effector away from objects in the field of view. The weight distribution function shown in Fig. 8.2 is applied for every DOF that can be directly controlled. As such, this obstacle avoidance scheme is designed for 2 DOF, for 2D Cartesian obstacle avoidance motion.

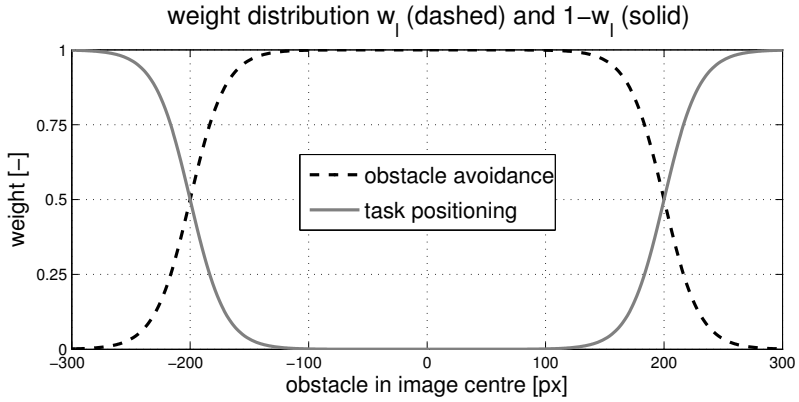


Figure 8.2: Two double sigmoid functions represent the weight of each task for visual end-effector control (task positioning $\dot{\mathbf{x}}_t$ or obstacle avoidance $\dot{\mathbf{x}}_{oa}$) as defined in (8.1). These weights depend on the obstacle position in the field of view. Such function is applied for each DOF that is directly controlled (i.e., in this case twice for obstacle avoidance in 2D).

8.2.2 Direct Trajectory Planning

Incorporating the avoidance of obstacles into a direct trajectory planner implies that constraints can be directly taken into account (see also Chapter 6). This includes the change of positions of (via-)points to avoid obstacles while maintaining certain constraints on these (via-)points as well as the trajectory itself. Similar to offline trajectory design, the flexibility of direct generation of trajectories suggests that more complex trajectories can be used for constrained avoidance. More specifically, two methods are presented, i.e., obstacle avoidance for point-to-point motion and multi-point motion.

For both methods, each iteration a new trajectory is generated, which implies that only the next state has to be computed. The conjunction of these trajectories is designed with $\mathcal{C}^2$ continuity. This means that the trajectory itself, its first time derivative $\mathcal{T}^{'}$ and the second time derivative $\mathcal{T}^{''}$ are continuous. In fact, the initial point x_I of a trajectory $\mathcal{T}$ is also the final point x_f of the previous trajectory. The same holds for both first and second time derivatives of a trajectory.

Point-to-Point Avoidance

The direct generation of a $\mathcal{C}^2$ continuous trajectory with initial (I) and final point (f) is designed as follows. The constraints which are variable are the final position, velocity and acceleration:

$$q_f \in \mathcal{C}_{free}, \quad \dot{q}_f = v_f, \quad \text{and} \quad \ddot{q}_f = \alpha_f,$$

as well as the global kinematic constraints

$$\dot{q}_{max} = v_{max}, \quad \text{and} \quad \ddot{q}_{max} = \alpha_{max}. \tag{8.3}$$

$\mathcal{C}_{free}$ is denoted as the collision-free space (or free-space) as defined by (3.46). The constraint vector $\mathbf{q}_c$ is obtained as

$$\begin{aligned} \mathbf{q}_c &= [\, q_I,\, q_f,\, \dot{q}_I,\, \ddot{q}_I,\, \dot{q}_f,\, \ddot{q}_f\,]^T, \\ &= [\, q_{k-1},\, q_f,\, v_{k-1},\, \alpha_{k-1},\, v_f,\, \alpha_f\,]^T, \end{aligned} \tag{8.4}$$

where the final constraints $q_f,\ \dot{q}_f,\ \ddot{q}_f$ are defined as an avoidance motion. The timing of the trajectory is obtained as

$$t_I = 0, \quad \text{and} \quad t_f = t_s + t_e - \Delta t_{sum}, \tag{8.5}$$

where t_s and t_e is obtained from (6.9) and (6.6) respectively and, as explained in Section 6.2.3, $\Delta t_{sum} = T_l n_{it}$ is the ascending trajectory time with T_l is the local loop time with iteration count n_{it}. The complete routine for computing the trajectory online (see Algorithm 6.2) is proposed in Section 6.2.

Multi-Point Avoidance

The direct generation of a $\mathcal{C}^2$ continuous 6^{th} order trajectory with initial- (I), via- (v) and final-point (f) is designed as follows. The constraints which are variable are the final position, velocity and acceleration, and the position of the via-point:

$$\{q_v, q_f\} \in \mathcal{C}_{free}, \quad \dot{q}_f = 0, \quad \text{and} \quad \ddot{q}_f = 0,$$

as well as the global kinematic constraints

$$\dot{q}_{max} = v_{max}, \quad \text{and} \quad \ddot{q}_{max} = \alpha_{max}. \tag{8.6}$$

The constraint vector $\mathbf{q}_c$ is now obtained as

$$\begin{aligned} \mathbf{q}_c &= [\, q_I,\, q_v,\, q_f,\, \dot{q}_I,\, \ddot{q}_I,\, \dot{q}_f,\, \ddot{q}_f]^T, \\ &= [\, q_{k-1},\, q_v,\, q_f,\, v_{k-1},\, \alpha_{k-1},\, v_f,\, \alpha_f]^T. \end{aligned} \tag{8.7}$$

One option is to use the via-point position q_v for avoidance motion and designing the final constraints as a stopping motion (e.g., $\dot{q}_f = \ddot{q}_f = 0$). The timing of this multi-point trajectory is obtained as

$$\begin{aligned} t_I = 0, \quad t_v &= t_{s,v} + t_{e,v} - \Delta t_{sum}, \quad \text{and} \\ t_f &= t_{s,f} + t_{e,f} - \Delta t_{sum}, \end{aligned} \tag{8.8}$$

where

$$t_{s,v} = \left\{ \frac{15}{8}\frac{h_v}{v_{max}}, \sqrt{\frac{10\sqrt{3}}{3}\frac{h_v}{\alpha_{max}}} \right\}, \quad t_{s,f} = \left\{ \frac{15}{8}\frac{h_f}{v_{max}}, \sqrt{\frac{10\sqrt{3}}{3}\frac{h_f}{\alpha_{max}}} \right\}, \tag{8.9}$$

and where $h_v = q_v - q_I$ and $h_f = q_f - q_I$ and v_{max} and α_{max} are the maximum velocity and acceleration respectively. $t_{e,v}$ and $t_{e,f}$ are both obtained from (6.6) as $t_{e,l} = \max\{t_{ev}, t_{e\alpha}\}$. The complete routine for computing the trajectory online (see Algorithm 6.2) is proposed in Section 6.2.

When an obstacle is detected and a change of final-point (or via-point) constraints is incorporated, the global constraint on the trajectory is most likely changed as well. To guarantee that a constraint is maintained, the method as proposed in Section 6.2.3 is executed. This method monitors the global constraints of the updated trajectory and, if a violation will occur, alters the execution time (i.e., by adding or subtracting time iteratively).

For consistency the direct point-to-point and multi-point trajectory were defined with 'q' as motion variable. If motion is to be designed in Cartesian space, the design is identical to the mentioned procedure. Furthermore, both methods are implemented as an event-based scheme, i.e., whenever an obstacle is present in the field of view a new motion is designed.

8.2.3 Visual Control Law

Kinematic control is achieved by using the control scheme as represented by (3.16) (see Fig. 8.3). The inputs for direct trajectory generation are developed as presented in Section 8.2.2. Local joint control is achieved by using a PD controller plus gravity compensation term as described in (3.19). For the obstacle avoidance via path planning a desired velocity $\dot{\mathbf{x}}_d$ is not computed and thus not used. To estimate the joint velocities of the manipulator, the method proposed in [138] is used. This method is based on the fact that numerical integration can provide more accurate results than numerical differentiation in the presence of noise.

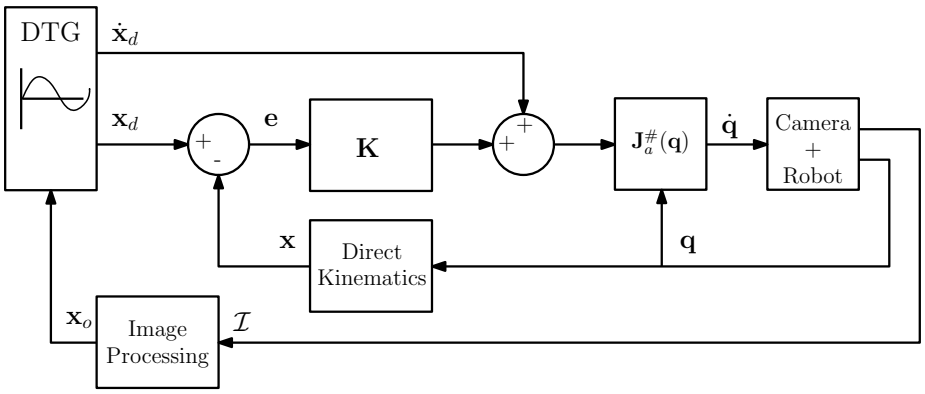


Figure 8.3: Kinematic control scheme for obstacle avoidance with direct trajectory generation.

8.2.4 Self-Motion Control

As explained in Section 3.3.2, for a kinematically redundant manipulator, a non-zero null space exists due to more degrees of freedom n than necessary for a particular task in the Cartesian space r (i.e., $r < n$). The method used here is projecting a secondary task onto the nullspace of the main task, i.e., by adding the homogeneous term $(\mathbf{I} - \mathbf{J}_a^{\#}\mathbf{J}_a)\dot{\mathbf{q}}_0$ with the minimum-norm term. In this, $\dot{\mathbf{q}}_0$ is an arbitrary joint velocity vector, $\mathbf{J}_\mathbf{a}^{\#}$ is de weighted pseudo-inverse of $\mathbf{J}_a$ as defined by (3.23) and $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix. One of the most widely adopted approaches is to solve redundancy by optimizing a scalar cost function $\mathbf{m}(\mathbf{q})$ using the Gradient Projection Method (GPM), i.e., choosing $\dot{\mathbf{q}}_0 = k_0 \nabla \mathbf{m} = k_0(\partial \mathbf{m}/\partial \mathbf{q})^{\mathrm{T}}$. This then represents a smooth function for a secondary (or third) task in terms of some performance index. Several performance or distance indices are presented in Section 3.3.2.

For safe operation, the redundancy of the manipulator (i.e., the manipulator's self-motion) is also controlled to avoid obstacles. This can be defined by $\dot{\mathbf{q}}_0$, to move towards or away from objects. This method is defined as follows.

In robotic manipulators consecutive joint axes may intersect (i.e., rotate in the same point $\mathbf{x}_{in}$). This means that not all joints are necessary or even appropriate for redundancy control. This number of intersecting joints n_{in} is limited by the total number of joints in the manipulator but practically never exceeds 3 (i.e., $n_{in} = \{2, \ldots, n\}$). Furthermore, a base joint position is always unsuitable since it is fixed to the world, and certain joints may be located so close to each other that evaluation of both is redundant.Let n be the number of joints of the manipulator and $n_{in} - 1$ the number of joints in an intersection point. The number of suitable joint points for evaluation n_s is then defined as $n_s = n - n_{in} - 1$, where the number 1 represents the base joint.

The redundancy formulation to avoid or approach a point in $\mathbb{R}^3$ space is defined in (8.1), where $\dot{\mathbf{q}}_0 = k_0 \nabla \mathbf{m}$. In this,

$$\nabla \mathbf{m} = \sum_{i=1}^{n_s} \mathbf{J}_{a,i}^{\#} \dot{\mathbf{x}}_{q,i}, \tag{8.10}$$

where $\mathbf{J}_{a,i}^{\#}$ is the weighted Jacobian pseudo-inverse. k_0 is a scalar which controls

the gain due to the second and can be defined as [91]:

$$k_0 = \frac{|\mathbf{J}_a^{\#}(\mathbf{w}\dot{\mathbf{x}}_{oa} + (\mathbf{I}_w - \mathbf{w})\dot{\mathbf{x}}_t)|}{|(\mathbf{I} - \mathbf{J}_a^{\#}\mathbf{J}_a)\nabla\mathbf{m}|}. \tag{8.11}$$

This k_0 is designed to avoid a large difference between the two main terms in (8.1). In essence, the redundancy formulation tries to minimize the euclidean distance d_e between a control pose $\mathbf{x}_c$ and the current pose of a joint q_{i,n_s} (i.e., defined as $\mathbf{x}_{i,n_s}$):

$$\min_{d_e}\{|\mathbf{x}_{i,n_s} - \mathbf{x}_c|\}. \tag{8.12}$$

When the value of k_0 is negative (i.e., $k_0 \in [-1, 0)$), the control pose acts as an attractor, i.e., d_e is minimized. Similarly, when the value of k_0 is positive (i.e., $k_0 \in (0, 1]$), the control pose acts repulsive, i.e., d_e is maximized. This measure should be chosen with great care, as a too large weight on the nullspace formulation can result in unstable configurations.

8.3 Experimental Results

Experiments are conducted to show the difference between avoidance via path planning and avoidance via trajectory planning. Prior to these results the experimental setup is presented and explained in detail. Finally, the additional kinematic redundancy scheme for avoidance with respect to self-motion is presented separately.

8.3.1 Experimental Setup

The selected robotic manipulator is the AMOR anthropomorphic arm[1] from Exact Dynamics, B.V.[2] (see Fig. 8.4). The manipulator has 7-DOF and is equipped with a gripper at its end-effector for the manipulation of objects.

Fully stretched the manipulator has a spherical range of 1.1 [m] and can rotate unlimitedly around its base (i.e., the range of joint q_1 is 360°). The camera is located on the end-effector (eye-in-hand), with the gripper's z-axis aligned and parallel to the camera's optical axis. The Denavit-Hartenberg parameters for modelling the manipulator and deriving its forward and inverse kinematics can be found in Appendix B, together with the joint ranges for $\mathbf{q} = [q_1, q_2, q_3, q_4, q_5, q_6, q_7]^T$.

Simulations are carried out by using the Robotics [32] and the Epipiolar Geometry [100] Toolboxes for Matlab. To simulate visual feature detection, a set of 30 random points is generated, from which two views are created with a perspective transformation. These two perspective point sets are then input to the homography calculation, which determines a rotation and scaled translation difference for control. Normally distributed random noise is added to the points with zero mean and 5% standard deviation.

For the experimental implementation, all the matrices required by the inverse kinematics algorithm are derived using the Robotics Toolbox for Matlab.

[1] http://www.amorrobot.com/
[2] http://www.exactdynamics.com

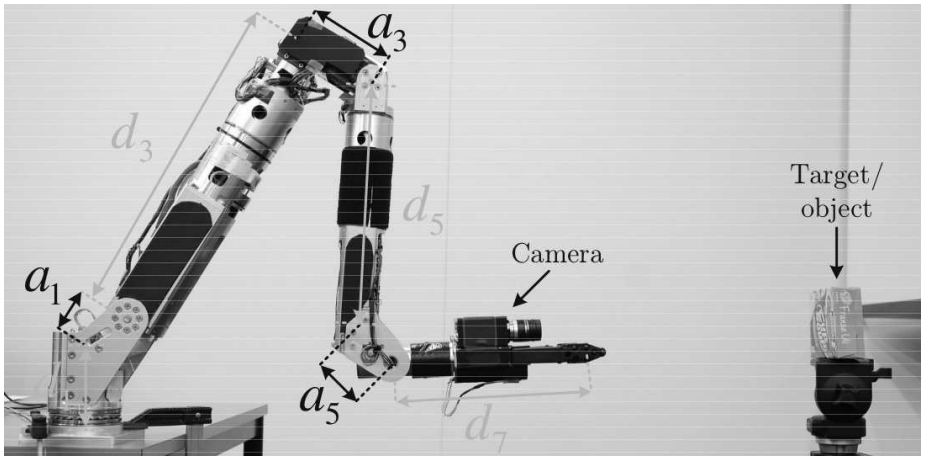


Figure 8.4: Redundant 7 DOF AMOR robotic manipulator.

The inverse differential kinematics are combined with the homogeneous solution for redundancy. These are translated to and optimized for `C/C++`, and implemented using the Eigen library for vector and matrix manipulation. This is wrapped inside a ROS [117] node for high-level functionality and low-level device control. Communication with the arm is done via a CAN-USB device, with different threads managing the data exchange with the CAN device (one for reading and one for writing) and the inverse kinematics algorithm (with an update rate of 200 [Hz] and including the computation of the low-level PD individual joint control). The manipulator's on-board electronics generate the actuator's PWM signals and process sensor data with an update rate of 1000 [Hz]. The camera (Prosilica GE680M) is connected via a Gigabit Ethernet interface (GigE Vision) to a standard notebook with 2 GB of RAM and 2.4 GHz Intel Core 2 Duo CPU running Linux. The software framework was (partly) developed by Alejandro Alvarez-Aguirre [4] as part of the Teleoperated Service Robot (TSR) project[3].

8.3.2 Vision-Based Obstacle Detection

For visual processing, the camera takes grayscale images which are processed using the computer vision library Opencv [14]. Obstacles are detected using the algorithms as presented in Chapter 4. In particular, the SURF feature detector and descriptor (see Section 4.4.3) is employed to find and match correspondence points in a reference image and a current image. This therefore includes a database of images (i.e., sets of keypoints of reference images), with potential obstacles, which are precomputed and prestored in the memory of the computer. The keypoints that are found in a current view are therefore continuously compared for a potential match. Subsequently, a homography is estimated and decomposed (see Section 4.3.1 and Section 4.3.2) which finally gives a rotation and scaled translation between the end-effector (i.e., the camera) and an obstacle. The vision algorithm is executed at 10 Hz with an image

[3]Further developments can be found on the website: http://www.win.tue.nl/tsr/

size of 640×480 $[px]$ (VGA), which is fast enough to detect slow moving obstacles in the field of view. Fig. 8.5 shows an example of the vision algorithm executed in an office environment with the detected object outlined with a white rectangle.

Figure 8.5: Surf feature detector executed in an office environment. The obstacle is detected and outlined with a white rectangle.

8.3.3 Obstacle Avoidance via Path Planning

The path planning technique with obstacle avoidance presented in Section 8.2.1 is experimentally verified with the robotic manipulator shown in Fig. 8.4. For clarity one Cartesian degree of freedom (x-direction) is affected by an obstacle, which is detected by the camera as explained in Section 8.3.2. The motion task that is assigned to the manipulator is defined as follows. From the initial point (start-point in Fig. 8.6) a path is defined through 2 via-points and 1 final-point. This motion is unconstrained, i.e., no local or global constraints are defined.

The result of this reactive obstacle avoidance scheme can be seen in Fig. 8.6. Clearly the effected motion is not directly kinematically constrained and only a reactive motion (i.e., a path) is planned. This is shown as the path in between points is different compared to the motion without obstacle avoidance. Moreover, the required via-points (and end-point in the upper figure) are not reached due to the obstacle avoidance motion.

The difference in response between the two figures is due to a difference in parameters of the obstacle avoidance function (the sigmoid function in (8.2)). These values (i.e., $d_{o,l}$ and k_s) can be changed to obtain a different obstacle avoidance motion.

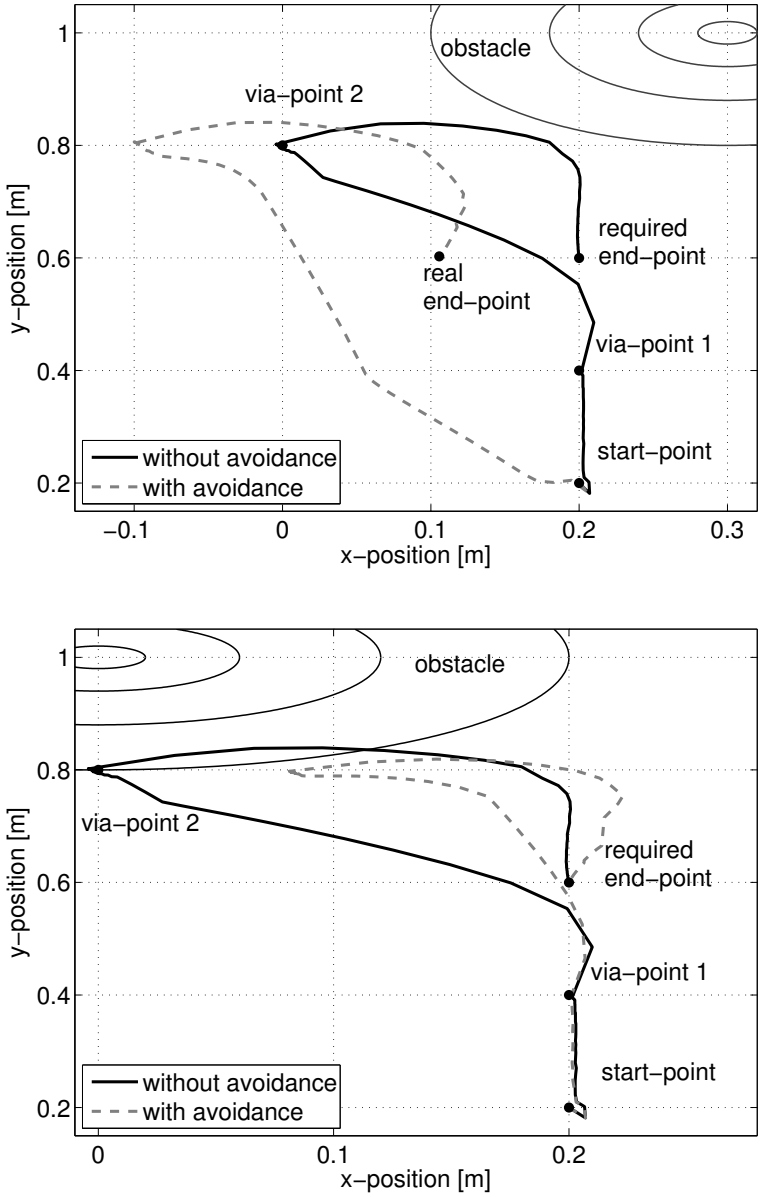


Figure 8.6: Two examples of obstacle avoidance via path planning. The circles denote the distance towards the object, and thus determine the weight of the tasks (i.e., positioning or avoidance). The lower figure shows that when executing motion with less strict parameters (i.e., smaller $d_{o,l}$ and k_s compared to upper figure) the realized path stays closer to the required (or desired) path. This can be seen as the realized and required end-point are the same. For the upper figure the parameters for avoidance are more strict (compared with the lower figure) and the realized motion is further away from the required motion. In particular, the realized motion does not reach the required end-point due to the obstacle.

8.3.4 Obstacle Avoidance via Direct Trajectory Generation

In order to assess the avoidance scheme, a scenario is developed in which a robotic manipulator should execute a predefined planar positioning task, and is blocked by an obstacle at certain time and location. This predefined motion trajectory is thus altered at an arbitrary time and location (i.e., when the obstacle is detected). Results for obstacle avoidance are shown with a point-to-point trajectory and a multi-point trajectory, and include a constraint optimization for velocity. Prior to this DTG approach, the performance of trajectory tracking is evaluated.

Trajectory Tracking Performance

The performance of trajectory tracking is evaluated by the root mean square (RMS) of the error (e_{RMS}) in joint space and Cartesian space (see Table 8.1). Compared to any high-end industrial manipulator (which typically expresses accuracy or repeatability in the range of 0.1 to 0.01 [mm]), these results are an order of magnitude lower in performance. The dominant reason for this is the relatively low joint update rate (i.e., 200 [Hz]) due to the consideration of a trade-off between tracking performance and visual updates. If the tracking performance of the individual joints is required to be higher, a faster update rate and a more complex compensation to disturbances could be employed, however, at the cost of an even lower visual update rate. In the case of visual control, a higher priority is given to the (robust) processing of images, as this directly accounts for a higher performance in Cartesian space as well (i.e., direct visual measurements).

Table 8.1: Tracking Performance

joint	e_{RMS} [rad]	Cartesian	e_{RMS}
q_1	0.0064	X	0.0030 [m]
q_2	0.0086	Y	0.0056 [m]
q_3	0.0052	Z	0.0052 [m]
q_4	0.0125	roll	0.0062 [rad]
q_5	0.0082	pith	0.0039 [rad]
q_6	0.0085	yaw	0.0072 [rad]
q_7	0.0064		

Results for Point-to-point Whole-arm Movements

Direct trajectory generation means that from any arbitrary state the motion of the manipulator should be guided to an online updated goal state, while maintaining certain kinematic constraints. Fig. 8.7 and Fig. 8.8 show the simulation and experimental results of this scenario for a point-to-point motion. From an initial start-point (i.e., $\mathbf{x}_I = [0.0,\ 0.0]^T$), a 5^{th} degree polynomial trajectory is designed to execute a straight-line motion in Cartesian space with predefined constraints. New final constraints are determined (i.e., position to the right) when the obstacle is detected (indicated by the arrow) and adapted in the constraint vector $\mathbf{q}_c$ accordingly. In order to not violate the predefined constraints,

the constraint optimization procedure monitors if a constraint will be violated in the future. In more detail, Fig. 8.7 simulates a motion that is interrupted at $t = 0.1$ $[s]$ by an obstacle. At this moment new final constraints are updated for the trajectory (i.e., $\mathbf{x}_f = [0.2,\ 0.3]^T$). It can be seen that predefined motion bounds, i.e., $v_{max} = 0.5$ $[m/s]$ are not violated. Fig. 8.8 shows a similar motion in experimental setting. Motion, with initial point $\mathbf{x}_I = [0.05,\ 0.63]^T$, is now interrupted at $t = 0.95$ $[s]$ and new final constraints are updated for the trajectory (i.e., $\mathbf{x}_f = [0.225,\ 0.93]^T$). Moreover, the predefined bounds of $v_{max} = 0.5$ $[m/s]$ are not violated. Snapshots of this point-to-point method in experimental setting can be seen in Fig. 8.11.

Results for Multi-point Whole-arm Movements

A similar scenario is developed that generates motion to avoid an obstacle with a multi-point trajectory containing 3 points (one via-point is added with only a position constraint, thus still ensuring $\mathcal{C}^2$ continuity). From an initial start-point, this 6^{th} degree polynomial trajectory is designed to execute a straight-line motion in Cartesian space with predefined constraints. The extra via-point makes it possible to control more variables of the trajectory when compared with point-to-point motion (i.e., one extra position). Fig. 8.9 and Fig. 8.10 show the simulation and experimental results of this scenario. Again here, new final constraints are computed when the obstacle is detected (indicated by the arrow) and adapted in the constraint vector $\mathbf{q}_c$ accordingly. In order to not violate the predefined constraints, the constraint optimization procedure monitors if a constraint will be violated in the future and alters the execution time accordingly. As the constraint is reached, time-optimality is guaranteed.

In more detail, Fig. 8.9 simulates a motion (i.e., $\mathbf{x}_I = [0.0,\ 0.0]^T$) that is interrupted at $t = 0.55$ $[s]$ by an obstacle. At this moment the constraints are updated for the trajectory at the via-point (i.e., $\mathbf{x}_v = [0.18,\ 0.15]^T$) and final-point (i.e., $\mathbf{x}_f = [0.125,\ 0.3]^T$). It can be seen that predefined motion constraints, i.e., $v_{max} = 0.5$ $[m/s]$ are not violated. Fig. 8.10 shows a similar motion (i.e., $\mathbf{x}_I = [0.05,\ 0.63]^T$) in experimental setting. Motion is now interrupted at $t = 0.95$ $[s]$ and $t = 1.15$ $[s]$ and the constraints are updated for the trajectory at the via-point (i.e., $\mathbf{x}_v = [0.18,\ 0.8]^T$) and final-point (i.e., $\mathbf{x}_f = [0.125,\ 0.93]^T$). Moreover, the predefined constraints of $v_{max} = 0.5$ $[m/s]$ are not violated. Snapshots of this multi-point method in experimental setting can be seen in Fig. 8.12. Comparing the simulation and experimental results, it can be seen that the via-point is at a different location on the trajectory. This is most likely due to the time difference when updating the constraint vector when the obstacle is detected. Also the fact that both via-points differ in value might play a role.

One issue that remains when designing a multi-point trajectory is the fact that, due to the addition of a via-point, the trajectory is now a 6^{th} degree polynomial, which no longer implies a minimum-jerk trajectory.

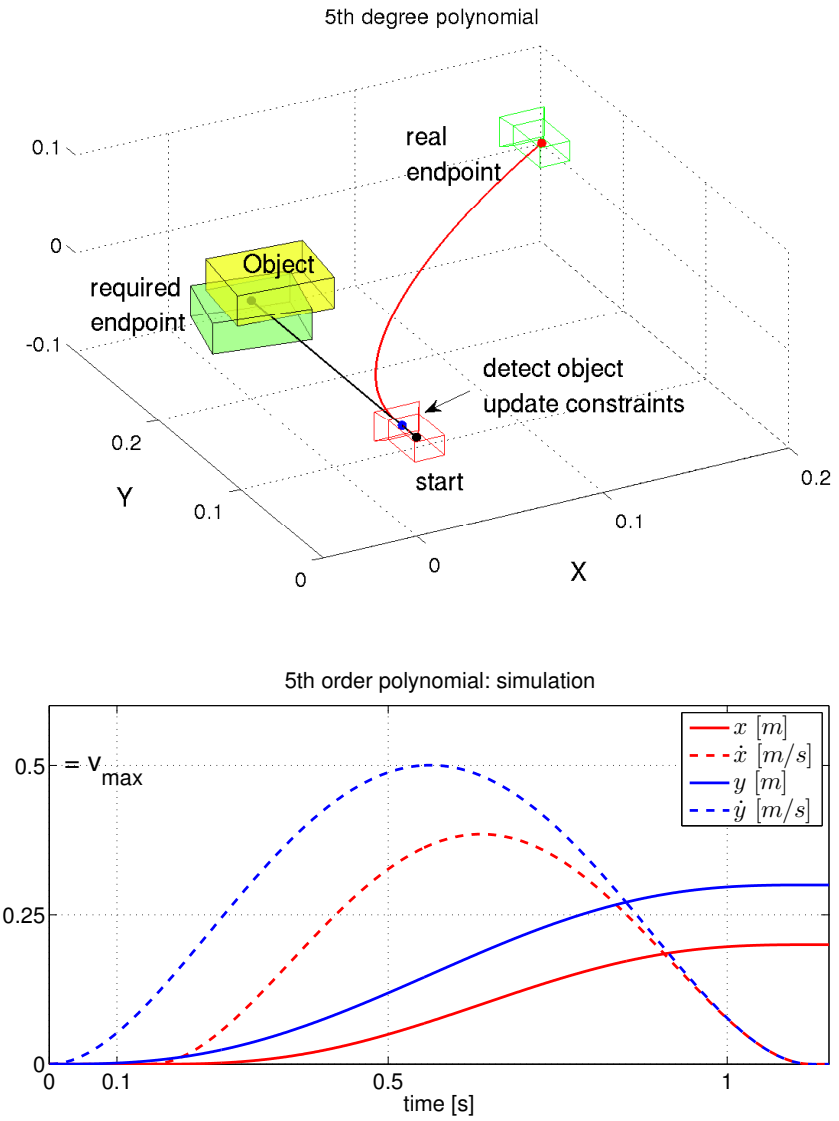


Figure 8.7: Simulation for direct, online obstacle avoidance with a 5^{th} degree polynomial (2 points, 3 constraints each). The initial trajectory with initial point $\mathbf{x}_I = [0.0,\ 0.0]^T$ is shown as a straight line. The object is smoothly avoided when detected (as indicated by the arrow in the trajectory, i.e., at $t = 0.1\ [s]$) with constrained motion $v_{max} = 0.5\ [m/s]$. In particular, for avoidance motion the final constraints are updated as $\mathbf{x}_f = [0.2,\ 0.3]^T$.

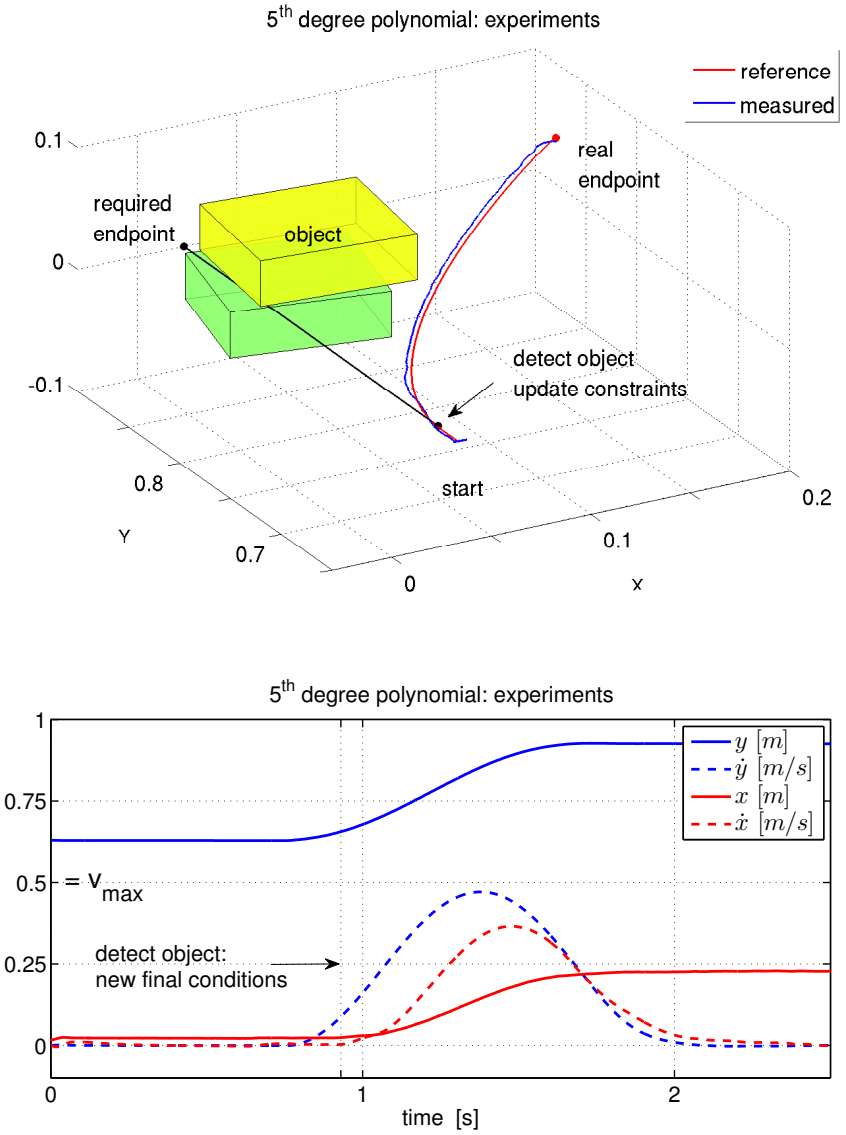


Figure 8.8: Experiment for direct, online obstacle avoidance with a 5^{th} degree polynomial (2 points, 3 constraints each). The initial trajectory with initial point $\mathbf{x}_I = [0.05,\ 0.63]^T$ is shown as a straight line. The object is smoothly avoided when detected (as indicated by the arrow in the trajectory, i.e. $t = 0.95\ [s]$) with constrained motion $v_{max} = 0.5\ [m/s]$. In particular, for avoidance motion the final constraints are updated as $\mathbf{x}_f = [0.225,\ 0.93]^T$.

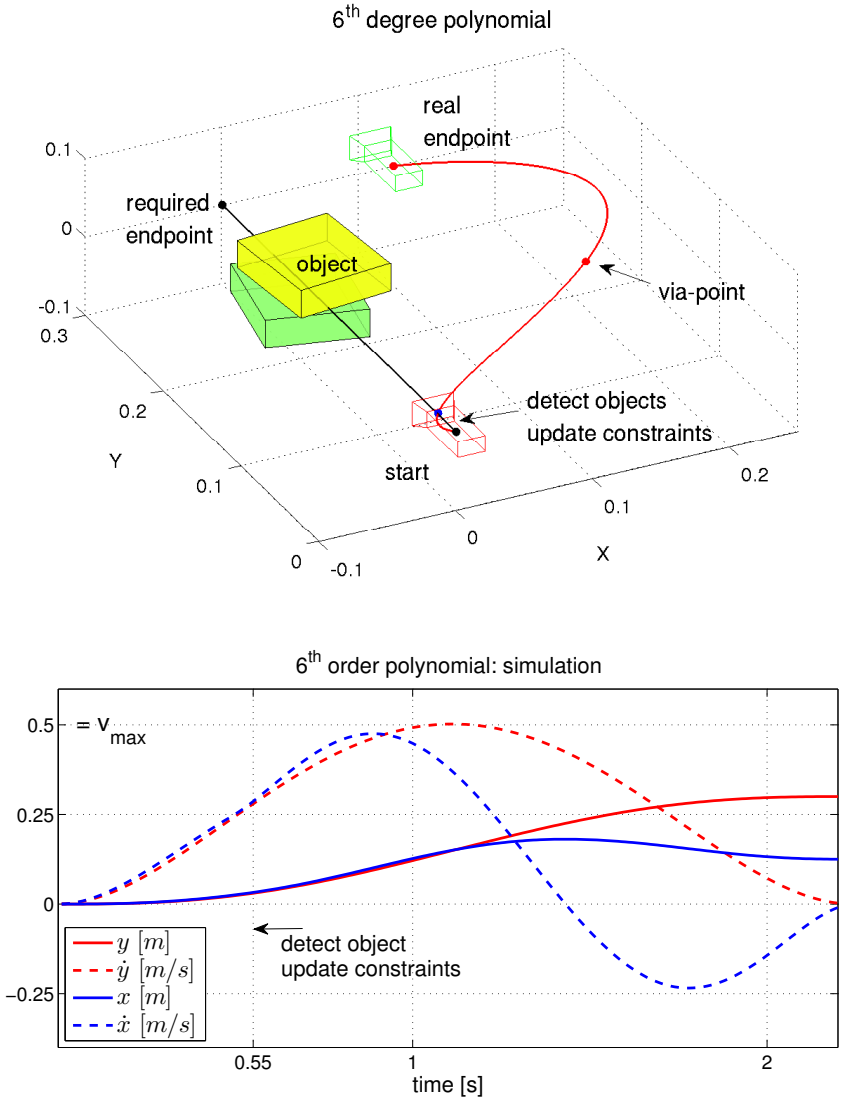


Figure 8.9: Simulations for direct, online obstacle avoidance with a 6^{th} degree polynomial (3 points, 3 constraints on extremal points, only position on via-point). The initial trajectory is shown as a straight line. The object is smoothly avoided when detected (indicated by the arrows, which occurs at $t = 0.55\ [s]$) with constrained motion $v_{max} = 0.5\ [m/s]$. In particular, for avoidance motion with initial point $\mathbf{x}_I = [0.0,\ 0.0]^T$, the via-point is determined as $\mathbf{x}_v = [0.18,\ 0.15]^T$ and final constraints are updated as $\mathbf{x}_f = [0.125,\ 0.3]^T$.

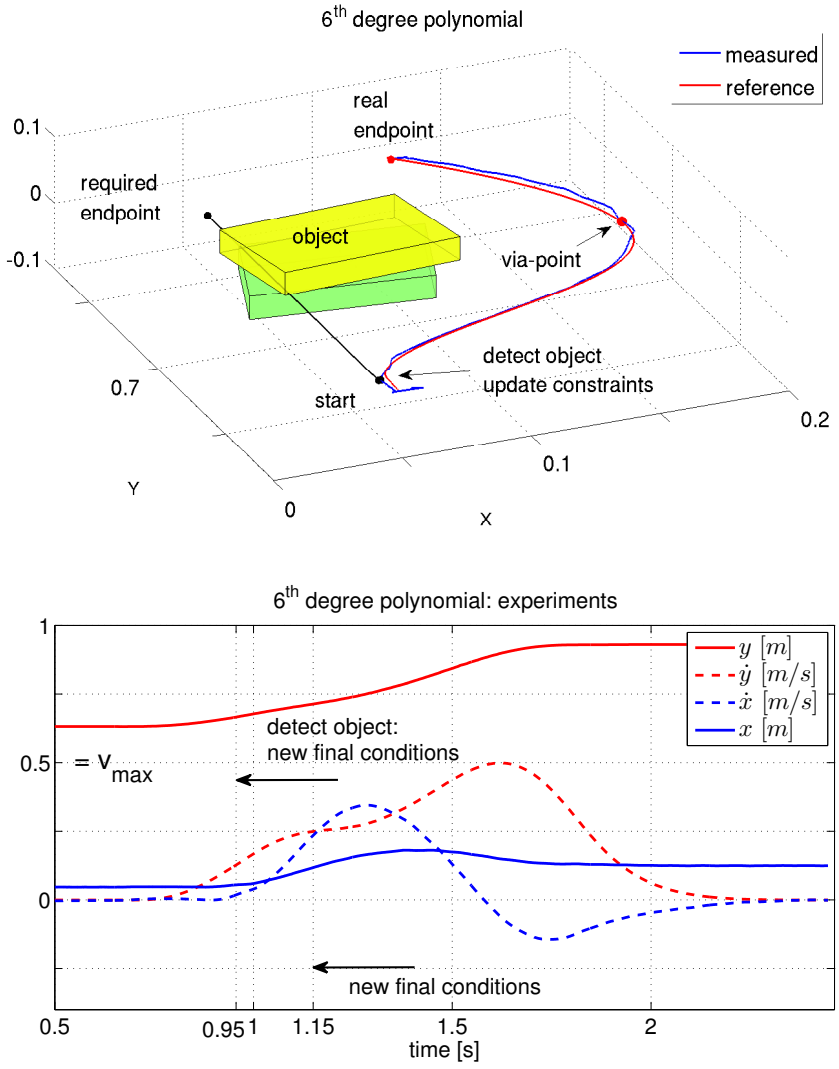


Figure 8.10: Experiment for direct, online obstacle avoidance with a 6^{th} degree polynomial (3 points, 3 constraints on extremal points, only position on via-point). The initial trajectory is shown as a straight line. The object is smoothly avoided when detected (indicated by the arrows, i.e., at $t = 0.95\ [s]$ as well as at $t = 1.15\ [s]$) with constrained motion $v_{max} = 0.5\ [m/s]$. In particular, for avoidance motion with initial point $\mathbf{x}_I = [0.05,\ 0.63]^T$, the via-point is determined as $\mathbf{x}_v = [0.18,\ 0.8]^T$ and final constraints are updated as $\mathbf{x}_f = [0.125,\ 0.93]^T$.

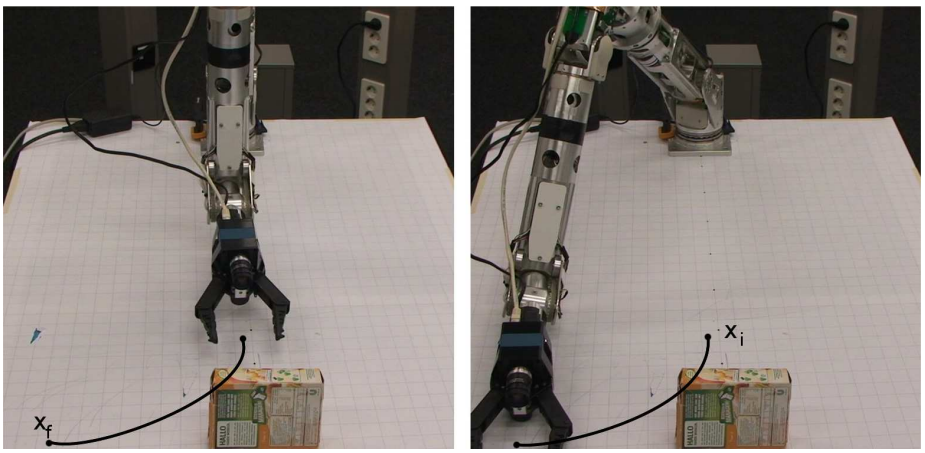


Figure 8.11: Obstacle avoidance via DTG with point-to-point motion.

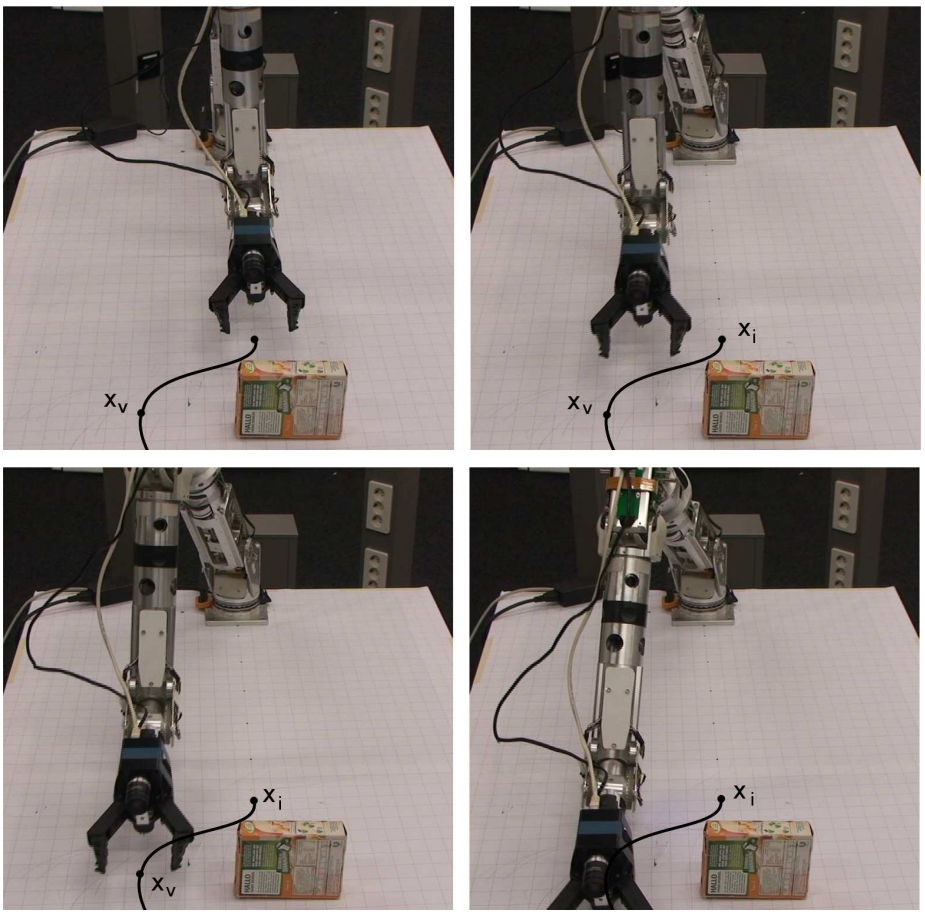


Figure 8.12: Obstacle avoidance via DTG with multi-point motion.

Simulation Results for Dynamic Object following

To show the potential of the DTG method towards a dynamic target, a simulation example is presented. In this context, dynamic means that the final point of the trajectory is continuously changing, at the rate of for instance a vision sensor. As such, a task is defined which involves the following of a moving object while ensuring a motion that stays within predefined bounds on the acceleration. A 5^{th} order polynomial is generated every control cycle, with a bound on the acceleration of $|\alpha_{max}| = 1\ [m/s^2]$. This trajectory is then updated at a fixed rate (i.e., every 20 iterations, which translates to 50 $[Hz]$ at a 1 $[kHz]$ local control rate). At $t = 0\ [s]$ the final point of the trajectory is $\mathbf{x}_f = [0.1,\ 0.3]^T$. The object moves towards $\mathbf{x}_f = [0.23,\ 0.13]^T$ between $t = 0.5\ [s]$ and $t = 0.9\ [s]$ and towards $\mathbf{x}_f = [0.14,\ 0.04]^T$ between $t = 1.7\ [s]$ and $t = 1.9\ [s]$. To show that the DTG method can cope with noisy measurements, the final point update is subject to noise with 50% standard deviation. This change of $\mathbf{x}_f$ is adapted in the constraint vector $\mathbf{q}_c$ accordingly. To ensure that the predefined bound on the acceleration trajectories are both kept, the optimization procedure monitors if a constraint will be violated in the future and alters the execution time t_f accordingly. As the constraint is reached, time-optimality is guaranteed. The results of this simulation can be seen in Fig. 8.13.

8.3.5 Self-Motion Control

As explained in Section 8.2.4, it has to be analysed which joints qualify as suitable points for nullspace control. For the 7-DOF redundant manipulator, the joint pairs $q_2 - q_3$, $q_4 - q_5$ and $q_6 - q_7$ are intersecting. The joints q_2, q_4 and q_6 are therefore chosen for evaluation of boundary crossing. Each iteration the position in Cartesian space (i.e., $\mathbf{x}_{q,i}$ for $i \in \{2, 4, 6\}$) is computed and, accordingly, the shortest (i.e., perpendicular) distance towards several geometric objects is evaluated as follows. The trajectory of the end-effector is chosen as defined in Section 8.3.3.

Point Distance Index

The point distance index $d_{p,p}$ defined in Section 3.3.2 with the accompanying gradient projection as defined by (3.31) is used to show the control of self-motion of the 7-DOF redundant manipulator. This point distance index is defined as the shortest distance between a point on the manipulator and a point in Cartesian space. Joint q_4 has the most freedom for self-motion and is therefore chosen for avoidance. Fig. 8.14 and Fig. 8.15 presents two experiments with different avoidance points (i.e., $\mathbf{x}_{o,1}$ and $\mathbf{x}_{o,2}$ respectively) and the changing of the gain k_0. If this gain is not used (i.e., $k_0 = 0$), the direction for self-motion of the manipulator is given no particular preference. By giving k_0 a negative value, a negative velocity is generated for joint q_4 towards the avoidance point. This in effect results in a motion which pushes joint q_4 away from the avoidance point. It can be seen in Fig. 8.14 that such avoidance motion is highly dependent on the configuration of the robot and the desired end-effector motion. In particular, Fig. 8.14 (i.e., $k_0 = -0.6$) shows that for a greater overall avoidance motion, a compromise has to be found at different times.

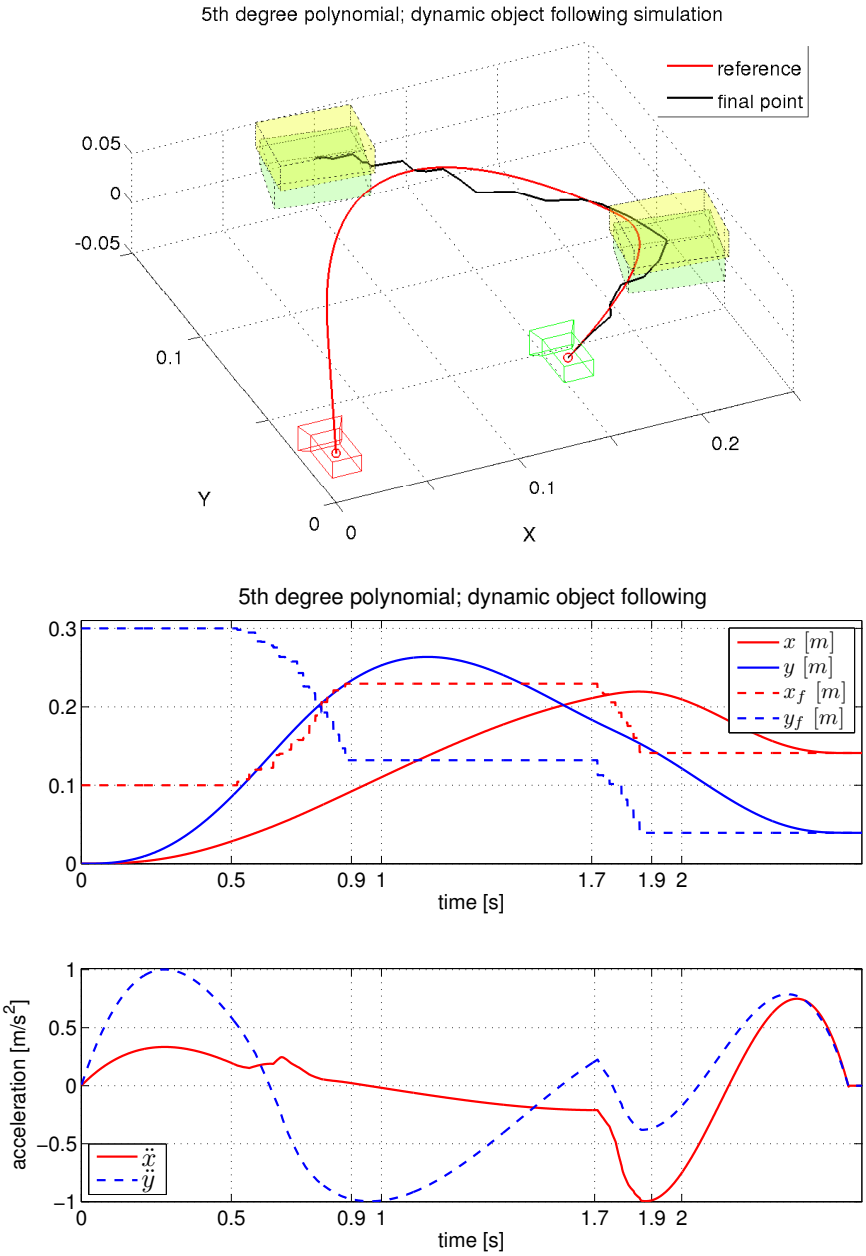


Figure 8.13: Simulation of direct trajectory generation for the following of a dynamic object with a 5^{th} degree polynomial. The start position of the object is the final point $\mathbf{x}_f = [0.1,\ 0.3]^T$ of the trajectory at $t = 0\ [s]$. The object moves towards $\mathbf{x}_f = [0.23,\ 0.13]^T$ between $t = 0.5\ [s]$ and $t = 0.9\ [s]$ and towards $\mathbf{x}_f = [0.14,\ 0.04]^T$ between $t = 1.7\ [s]$ and $t = 1.9\ [s]$. Upper and middle figure shows that the object measurements are noisy, however, the position trajectories are not. Lower figure shows a $\mathcal{C}^0$ continuous acceleration trajectory with a bound on the acceleration: $|\alpha_{max}| = 1\ [m/s^2]$ guaranteed by the constraint optimization procedure.

Perpendicular Distance Index

The perpendicular distance index $d_{p,l}$ defined in Section 3.3.2 with the accompanying gradient projection as defined by (3.38) is used to show the control of self-motion of the 7-DOF redundant manipulator. This perpendicular distance index is defined as the perpendicular distance between a point on the manipulator and a line in Cartesian space, i.e., from the base of the manipulator to the end-effector; L_{be}. Joint q_4 is again chosen as this has the largest freedom for self-motion. In this experiment, joint q_4 is controlled to be attracted towards the line L_{be}. In Fig. 8.16 the perpendicular distance towards this line $d_{p,l}$, shows to be smaller when a larger gain k_0 is executed. One drawback is that the cumulative distance of all joints (i.e., d_{total}, the covered distance of all joints) turns out to be larger.

Boundary Index

The boundary distance index $d_{p,b}$ defined in Section 3.3.2 with accompanying gradient projection as defined by (3.44) is used to show the self-motion of the 7-DOF redundant manipulator. This boundary distance index is defined as the perpendicular distance between a point on the manipulator and a boundary surface (plane) in Cartesian space. Three points are selected; the base of the manipulator $\mathbf{x}_1 = [0, 0, 0]^T$, a point directly above the base $\mathbf{x}_2 = [0, 0, 1]^T$ and a point in space completing a plane that separates the manipulator from a certain boundary, $\mathbf{x}_3 = [1, 1, 0.5]^T$. The distance of joint q_4 perpendicular towards this plane, $d_{p,b}$ is shown in Fig. 8.17. It is shown that for larger positive values of k_0 the joint q_4 is attracted towards the plane, and conversely, for larger negative values of k_0 the joint q_4 is pushed away from the plane.

These results obviously depend highly on the task at hand (i.e., the trajectory) and the configuration of the robot. The simplicity of this approach, however, is a high motivation to include self-motion avoidance into an obstacle avoidance scheme for robotic manipulators.

Furthermore, these indices are all determined as fixed points in Cartesian space. Of course it is possible to include the avoidance of self-collision, by including indices that represent points on the manipulator. However, a general definition and representation of this is not easily determined and can be a cumbersome task. A more logical solution towards self-collision avoidance is to restrict the motion of the end-effector such that self-collision will not occur.

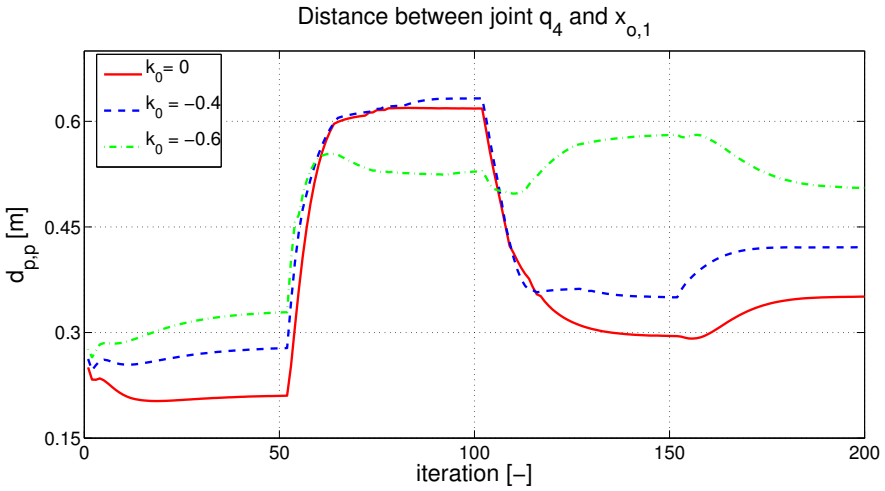


Figure 8.14: By altering the gain factor k_0 that weights the effect of the velocity of self-motion, it is shown that a stronger gain corresponds to a bigger repulsion force towards a point. This in effect creates a larger distance $d_{p,p}$ between joint q_4 and avoidance point $x_{o,1}$. Obviously this is dependent on the trajectory and the configuration of the robot, as is shown by the experiment where $k_0 = -0.6$ that has to compromise at different times (i.e., lower distance between the iteration range $50 - 100$) for a greater overall avoidance distance.

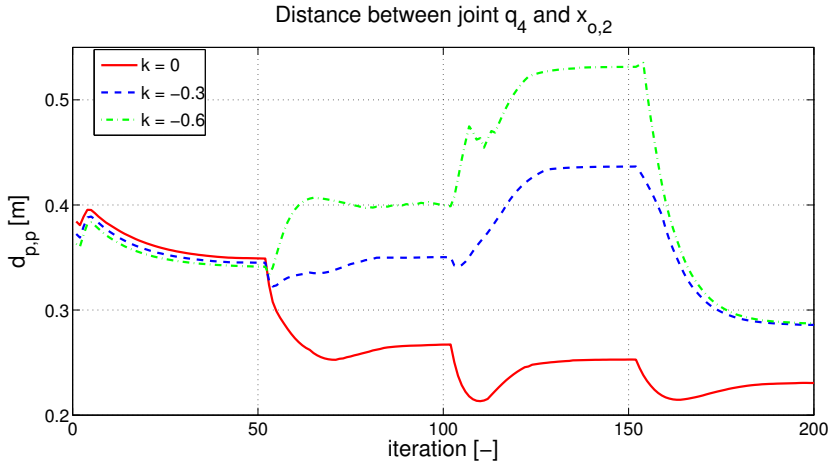


Figure 8.15: Similar example of repulsion by altering the gain k_0 that weights the effect of the velocity of self-motion. The point $x_{o,2}$ is chosen directly underneath joint q_4 so that it will be pushed upwards.

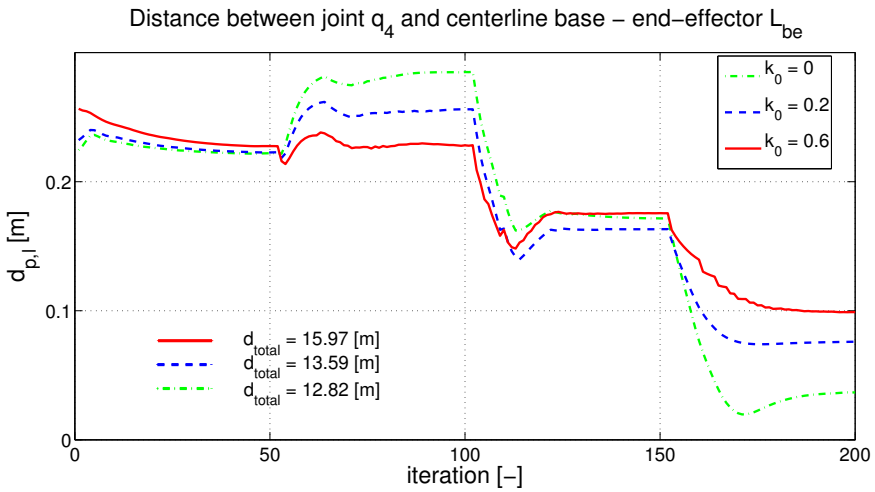


Figure 8.16: The distance index $d_{p,l}$ used is the shortest (perpendicular) distance from joint q_4 towards base - end-effector centreline L_{be}. For increasing values of k_0, the joint q_4 stays closer to this line L_{be}. The total travelled distance d_{total} of all joints, however, turns out to be larger in this case.

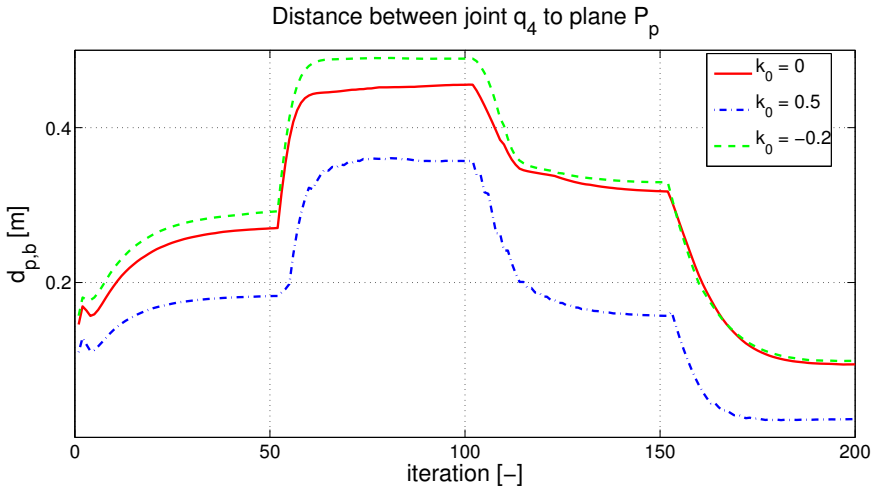


Figure 8.17: Avoidance or attraction towards a plane is realized for different values of k_0. For larger positive values of k_0, the perpendicular distance of joint q_4 towards the plane, $d_{p,b}$ becomes smaller (i.e., attraction), and conversely, for smaller values of k_0, the perpendicular distance of joint q_4 towards the plane, $d_{p,b}$ becomes larger (i.e., avoidance).

8.4 Summary

This chapter presented and proposed several solutions towards the application of vision-based obstacle avoidance. Methods that were presented in previous chapters are elaborated with respect to this similar task. In particular, a reactive path planning method is shown which avoids an obstacle by weighting the velocity of the end-effector (i.e., the camera) according to the vicinity towards an obstacle. This is essentially a path planning approach as constraints on a kinematic level are not directly taken into account. In order to incorporate such constraints online, a direct trajectory generation method is developed which designs or alters a trajectory online based on direct visual measurements. The method is presented for point-to-point and multi-point trajectories and includes an optimization scheme which guarantees that a predefined global kinematic constraint is maintained. Besides this obstacle avoidance method for the end-effector of a manipulator, obstacle avoidance is also included for the self-motion of a redundant manipulator. This avoidance motion is based on the gradient projection method, and includes several indices for avoidance, i.e., the perpendicular distance towards a point, a line and a plane.

This complete framework is implemented on a 7-DOF redundant robotic manipulator, consisting of an eye-in-hand camera with processing on a standard notebook. The details of this setup, as well as the implementation of the developed methodologies are explained and results are shown which motivate the proposed method. In particular, visual processing is executed at 10 $[Hz]$ with an image size of 640×480 $[px]$, the kinematic controller (which includes the direct and differential kinematics) is executed at 200 $[Hz]$ and local joint control is executed at 1 $[kHz]$. Due to this relatively low update rate (i.e., 200 $[Hz]$), the performance of trajectory tracking (i.e., $\bar{e}_{RMS} = 0.008$ $[rad]$ in joint space) is less accurate than high-end industrial manipulators (i.e., an order of magnitude lower), however, the direct visual sensing approach accounts for an improved performance in Cartesian space. Finally, simulations and experiments are conducted to show the difference between obstacle avoidance on a path-planning level compared to obstacle avoidance on a trajectory-planning level. For obstacle avoidance with direct trajectory generation obstacles are smoothly avoided and kinematic constraints are maintained due to the optimization scheme. Experimental results also show the effectiveness of obstacle avoidance for the self-motion of the manipulator by regarding the aforementioned distance indices.

CHAPTER 9

Conclusions and Recommendations

Abstract. In this final chapter, the main conclusions of this research are presented. Following, several recommendations for improvements and future work are given.

9.1 Conclusions

The main objective of this work has been the development and implementation of methodologies that provide robots with the ability to use visual measurements in their activities in a direct and constrained way. This is motivated by the fact that visual controlled motion designs a path for positioning (instead of a trajectory which includes time) and changes of constraints (spatial or kinematic) can not (directly) be taken into account by current state-of-the-art solutions. The developments that take these issues into consideration are as follows.

Feedforward Visual Servoing

Traditional visual servo control systems can be divided into two common approaches, image-based (IBVS) and position-based visual servoing (PBVS). As the names suggest, IBVS designs motion based on image-space feedback and PBVS designs motion based on 3D Cartesian-space feedback. A combination of both methods is known as hybrid visual servoing. For example, the method designed by Kyrki et al. [86] designs a shortest path in Cartesian space while guaranteeing object visibility. In this, object visibility is only guaranteed by an image error, and thus, disturbances typical for motion control systems (i.e., friction, gravity) could play a large role. The result is that the range of motion can be fairly limited, as is shown in experimental setting. A method is proposed, denoted feedforward visual servoing, that overcomes these issues by combining position-based visual servoing with a rotational image-based feedforward control action. This effectively ensures the field-of-view constraint and adds a greater range of motion for e.g. exploration around an object. It is shown that by definition the stability properties of the initial position-based method do not change.

The approach is validated in simulation and experimental setting with a 7-DOF redundant manipulator with eye-in-hand camera. A comparison with two other methods (i.e., traditional position-based visual servoing and the partitioned approach by Kyrki et al. [86]) is given where the advantage of the proposed approach is clearly shown.

For the development of this method several other contributions are proposed. These are listed as follows.

- The modelling of 3D vision, consisting of the pinhole camera model, a camera calibration method and the modelling of projective transformations (i.e., a homography) is discussed and developed in software.
- The implementation and comparison of feature detectors for planar object detection. A review is made of existing detection algorithms (i.e., several corner and blob detectors), which are analysed towards a real-time implementation. Due to their strong descriptive properties, SIFT and SURF are chosen for a more thorough comparison. SURF is finally chosen because of its computational advantages (i.e., robust keypoint detection with 70 matches in 140 $[ms]$ and thus an order of magnitude faster than SIFT), and the ability to tune between number of found keypoints and processing time.
- The implementation and evaluation of homography estimation and decomposition techniques for 3D visual measurements between two views. These transform the two points-sets obtained from SURF (i.e., from a reference image and a current view) into a 3D error pose.
- The implementation of an image-based, a position-based and a hybrid visual servoing technique in simulation and experimental setting.

Direct Visual Servoing

A method is developed that incorporates direct visual measurements into the design of motion. For traditional encoder-based control systems, motion is designed based on the readings of the motor-encoder. This implies that the control performance is dependent on several factors which are unrelated to the sensor measurements (e.g., stiffness and inaccuracies of the motion and fixation system, uncertainties in system modelling). Furthermore, as the positioning is executed with respect to a fixation system and not directly with respect to the product itself, encoder-based measurements are indirect. This leads to an inherent mismatch between the location of measurements and the location of task execution. By controlling a motion system with feedback directly obtained from visual measurements, typical traditional difficulties as found in encoder-based motion systems can be avoided.

The approach is motivated by considering the industrial application of inkjet printing. In this, a near-repetitive product pattern (an OLED display with an array of pixels/cells) needs to be manufactured by printing a droplet of polymer into each display pixel. Current state-of-the-art solutions solve the motion control problem with encoder-based feedback, which fail if the pitch (or time) between print actions is not identical. For the proposed approach, motion is designed with a velocity trajectory which is designed online, based on direct visual measurements. On OLED cell centres the velocity is predefined ($4\ [px/frame] = 28.8\ [mm/s]$), while in between centres the velocity is designed higher. This results in a higher average velocity for the overall motion, which would be impossible for a constant velocity trajectory if a similar quality of printing should be ensured.

For the development of this method several other contributions are proposed. These are listed as follows.

- A method for the calibration of a lens with short depth-of-view is developed. Short depth-of-view means that traditional macro-calibration methods do not suffice, as only a planar calibration pattern exactly perpendicular to the image sensor can be utilized. It is shown that, due to the lens, the narrow field-of-view and the measurement noise calibration for lens distortion is not necessary.

- A robust method for the visual detection of individual display cells is developed. This method is robust against lighting changes and faults of the substrate (e.g., surface cracks, dust) or in individual cells. The images have a size of 160×100 $[px]$, contain 3×5 OLED cells and have a pixel size of 4.5 $[\mu m]$. This allows for a visual update rate of 1600 $[fps]$. The measurement noise shows to have a standard deviation of $\sigma = 0.18$ $[px] = 0.85$ $[\mu m]$.

- A direct visual control method for display manufacturing is proposed. The method consists of a PID with velocity feedback and a feedforward compensation scheme for known disturbances (i.e., mass of the table and friction of the motors). The velocity trajectory is generated online based on direct visual measurements, where the time for triggering the printhead is predicted based on an α-β filter.

- The developed method is implemented on an experimental setup, consisting of a 2D planar table, a static camera and an FPGA for processing. The direct visual control structure is executed at 1.6 $[kHz]$, and includes the processing of images, as well as the computation of the trajectory and the control law. The performance of the control system (i.e., velocity trajectory tracking) is determined by the root mean square of the error velocity: $\dot{e}_{rms} = 0.40$ $[px/frame] = 1.8$ $[\mu m/frame] = 2.88$ $[mm/s]$.

Direct Trajectory Generation

A method is developed that designs a trajectory directly based on the current state and events. This implies that (changes of) constraints (i.e., spatial or kinematic) can be incorporated at each iteration and a fast response to disturbances is possible. Traditional motion control designs a trajectory offline (which remains unchanged during execution), where kinematic constraints can be incorporated in a straight-forward manner. Sensor-based motion control on the other hand directly reacts to sensor-readings (e.g., visual detection of an obstacle), however, kinematic constraints can not be incorporated in a straightforward manner. The proposed approach combines both methods into one, where a new trajectory is generated each iteration, which takes direct measurements and constraints into account.The method is validated in two experimental settings.

First, when considering the industrial application of inkjet printing, a near-repetitive product pattern serves as visual encoder which is used as input for trajectory generation. The fact that the pitch between display cells is varying motivates the benefit of the direct trajectory generation method over an

encoder-based approach. Experimental results are presented with a 2D planar positioning table with sensing and positioning at micrometer scale.

Second, current state-of-the-art solutions for obstacle avoidance for robotic manipulators commonly design and execute motion on a path planning level. The proposed approach designs and executes an obstacle avoidance motion on a trajectory planning level. Experiments with a 7-DOF anthropomorphic manipulator show smooth constrained motion for obstacle avoidance, compared to a reactive potential field-based approach. This robotic manipulator (with eye-in-hand camera and processing on a standard notebook) executes visual processing at 10 $[Hz]$ with an image size of 640×480 $[px]$. The kinematic controller (which includes the direct and differential kinematics) is executed at 200 $[Hz]$ and local joint control is executed at 1 $[kHz]$.

For the development of this method several other contributions are proposed. These are listed as follows.

- The development and implementation of an event- and rate-based method for direct trajectory generation. For rate-based trajectory generation a new trajectory is computed at a fixed rate, whereas for event-based trajectory generation a new trajectory is computed whenever an event occurs.

- The development and implementation of direct trajectory generation for point-to-point and multi-point motion. Trajectories with multiple points can be beneficial when more complex motion has to be designed (for e.g., multiple obstacles). This also includes the order of constraints at each point, as well as the order of continuity of the trajectory.

- The development and implementation of a constraint optimization procedure that alters the execution time of the trajectory online based on changed constraints. When constraints of a trajectory are changed online, the predefined constraints (e.g., maximum velocity or acceleration) will also change. Ensuring that the optimization method will guarantee to reach a given constraint effectively makes the trajectory time-optimal.

- The development and implementation of visual obstacle detection. Based on the visual detection methods for 3D visual measurements, a similar approach is developed to detect obstacles in the manipulator's field of view. Descriptors of several images are pre-stored in memory and continuously sought for in the current image. A homography estimation and decomposition then computes the 3D position of the object in the field of view with respect to camera.

- The implementation of a potential field-based obstacle avoidance technique. In this, the weight between the positioning task and the avoidance task depends on the vicinity towards an object.

- The development and implementation of an obstacle avoidance technique for the self-motion of a redundant manipulator. This includes the derivation of several distance indices (i.e., towards a point, a line and a plane in 3D Cartesian space) and the development of these in the gradient projection method for self-motion control.

9.2 Recommendations

As directions for future work several recommendations are made that could improve or build upon the proposed developments.

Distributed Processing

The developed methods for visual control are all executed on a single processing platform. For visual control of the robotic manipulator this is a notebook running Linux, for visual control of the OLED display this is an FPGA. Besides the processing of the visual data, also the control algorithm (local and global) and the scheduling of these is executed on this processor. The consequence of this is that a large amount of processing power is not devoted to the process which requires it most. It would be more desirable if a processor is completely devoted to its task at hand. For instance, one processor (e.g., an FPGA directly connected to the image sensor) could be completely devoted to process images, while a separate processor takes care of the scheduling of tasks and the global control law. As such, this enables the miniaturization of local, distributed controllers with individual processing abilities (e.g., low-cost FPGA). In this way, utilization of processing abilities is fully exploited and delays due to one process will not interfere with other processes.

Limitation of Inkjet Printing

The performance of the developed sensing and motion planning method in terms of printing speed is limited by the properties of the motion system. Consider for instance the resolution of the camera. This setting restrains the frame rate of the camera and as such the update rate for control and the design of motion. This in effect limits the drop-on-demand print frequency for the manufacturing of a display. Current standards in drop-on-demand printing systems (i.e., frequencies of printing in the range of $10 - 40$ $[kHz]$ are possible) as well as the developments in vision systems design (i.e., frame rates > 10 $[kHz]$ for area-scan cameras, > 50 $[kHz]$ for line-scan cameras), suggest that a development of a visual control system with limits closer to this state-of-the-art is possible. However, as the choices for the vision system as well as the motion system can be interdependent, and the fact that the parameter space for such design can be quite large, a straight-forward analysis does not give a clear optimal path. A more preferable solution is to utilize an automated method (i.e., design space exploration) which makes an optimal design choice based on certain predefined requirements (e.g., accuracy, update rate) or system properties (e.g., sensor resolution, frame rate).

Performance Improvement

The introduction of depth cameras such as the Microsoft Kinect enables an improvement for sensing compared to mono-vision cameras. For avoidance motion such sensing ability is essential, as a homography-based approach only provides a translation up to a scale factor. A second addition which would improve performance of the developed methods in short term is the use of available sensors for estimation. More specifically, standard industrial motors

are equipped with motor encoders, and can be used to attain a higher performance in motion control. For example, the developed planar motion table is controlled with visual feedback. This feedback consists of an estimated velocity, obtained at the same rate as the camera, which contains a fair amount of noise (due to visual processing). A more accurate estimation could be obtained from the available motor encoders, which are sensing in the same plane (i.e., coordinate frame), assuming that the linear velocity of the motor is equal to the velocity of the table. One condition, however, is that this motor encoder should have a higher resolution and update rate than the image sensor. For reference, the encoder currently present in the actuator has a 10-bit data update rate of 10 $[kHz]$, with a resolution of 8 $[\mu m]$. If this is sufficient for an improved velocity estimate has to be determined experimentally.

Dynamic Obstacles

The obstacle avoidance method as proposed with direct trajectory generation is only assessed with static obstacles. From a computational point of view, the limits for the avoidance of obstacles is (to some extend) dependent on the computational resources. The detection of fast moving obstacles is therefore limited by the sampling rate of the camera and the processing power of the system. In this thesis one simulation example of the tracking of a dynamic object is presented. In order to explore the full potential of the DTG method regarding dynamic objects, a thorough analysis with experimental implementation should be carried out. This is directly related to the implementation of kinematic control. As in such case it is assumed that the motion of the manipulator will not be executed at high velocities, this therefore also limits the response when fast motions are detected. Future work should therefore be focussed on control which includes dynamics instead of kinematics.

Combining DTG with a Sampling-based Planner

In the proposed solution towards obstacle avoidance, the path (i.e., positions in free space) is determined from visual processing. The fact that obstacles are assumed (or simplified) as simple shapes (i.e., polyhedrons) is, however, quite restricting. In fact, in order to take advantage of the configuration of objects (i.e., the space these occupy), a better sensing or modelling is necessary. One direction of approach could be to employ a sampling-based planner which determines a free path in a static and cluttered environment. This should therefore include a sensing system which captures the whole 3D space (e.g., a depth camera such as the Kinect). As these methods are known to have a high computational load, distributed processing (as mentioned earlier) should be applied.

By taking these developments and considerations into account, the true benefit of vision in robot control can effectively be utilized. As intended, these benefits reach further than simple advantages in performance of control or robustness of task execution. When such system is finally combined with a service robot, where a multitude of tasks are widely available, it is then to be proven useful for actual societal issues. As an assistant for tasks in human care environments, with safe operation as main objective, the goal of integrating robotics in everyday life becomes a clear possibility.

APPENDIX A

Minimum Jerk Trajectory: Proof

Hogan showed in [63] that smoothness of human arm movement can be quantified as a function of jerk. Such motion between two points is coordinated by minimizing the functional F, which is the sum of squared jerk along its trajectory:

$$F(q(t)) = \frac{1}{2}\int_{t=0}^{T} q^{(3)}(t)dt^2 = \frac{1}{2}\int_{t=0}^{T}\left[\frac{d^3q(t)}{dt^3}\right]^2 dt. \tag{A.1}$$

To find the minimum of this functional, calculus of variations is employed. In essence this involves determining the derivative of the functional with respect to a small perturbation. The minimum is then found when that derivative is zero. Let the variation be a function $\eta(t)$ which has the properties that it vanishes smoothly at the boundaries. That is:

$$\eta(t) : \begin{cases} \eta(t_0) = 0 & \eta(t_1) = 0 \\ \dot{\eta}(t_0) = 0 & \dot{\eta}(t_1) = 0 \\ \ddot{\eta}(t_0) = 0 & \ddot{\eta}(t_1) = 0 \end{cases} \tag{A.2}$$

In order to minimize $F(q(t))$, we replace $q(t)$ with $q(t) \mapsto q(t) + c\eta(t)$, where c is the differentiation variable. We now proceed with

$$F(q + c\eta) = \frac{1}{2}\int_{t=0}^{T}(q^{(3)} + c\eta^{(3)}dt^2. \tag{A.3}$$

Differentiation with respect to c yields

$$\begin{aligned} \frac{dF(q+c\eta)}{dc} &= \int_{t=0}^{T}(q^{(3)} + c\eta^{(3)})\eta^{(3)}dt, \text{ and} \\ \left.\frac{dF(q+c\eta)}{dc}\right|_{c\to 0} &= \int_{t=0}^{T} q^{(3)}\eta^{(3)}dt. \end{aligned} \tag{A.4}$$

Using integration by parts, this is rewritten as

$$\int_{t=0}^{T} q^{(3)}\eta^{(3)}dt = \int_{t=0}^{T} u\,dv = uv\Big|_0^T - \int_{t=0}^{T} v\,du \tag{A.5}$$

where $u = q^{(3)}$, $dv = \eta^{(3)}\ dt$, $du = q^{(4)}dt$, and $v = \ddot{\eta}$. In this, $q^{(3)}$ represents the third and $q^{(4)}$ represents the fourth derivative of q. This leads to

$$\int_{t=0}^{T} q^{(3)}\eta^{(3)}dt = q^{(3)}\ddot{\eta}\Big|_0^T - \int_{t=0}^{T} \ddot{\eta}q^{(4)}dt = -\int_{t=0}^{T} \ddot{\eta}q^{(4)}dt$$
$$-\int_{t=0}^{T} \ddot{\eta}q^{(4)}dt = -\int_{t=0}^{T} u\ dv = -uv\Big|_0^T + \int_{t=0}^{T} v\ du \tag{A.6}$$

where in this case $u = q^{(4)}$, $dv = \ddot{\eta}\ dt$, $du = q^{(5)}dt$, and $v = \dot{\eta}$. Continuing, we get

$$-\int_{t=0}^{T} \ddot{\eta}q^{(4)}dt = -q^{(4)}\dot{\eta}\Big|_0^T + \int_{t=0}^{T} \dot{\eta}q^{(5)}dt = \int_{t=0}^{T} \dot{\eta}q^{(5)}dt$$
$$\int_{t=0}^{T} \dot{\eta}q^{(5)}dt = q^{(5)}\eta\Big|_0^T - \int_{t=0}^{T} \eta q^{(6)}dt = -\int_{t=0}^{T} \eta q^{(6)}dt \tag{A.7}$$

It shows that the final integral is the derivative of our perturbed functional. That is:

$$\frac{dF(q+c\eta)}{c}\Big|_{c\to 0} = -\int_{t=0}^{T} \eta q^{(6)}dt \equiv 0. \tag{A.8}$$

As this property has to hold for any function $\eta(t)$, we can reduce (A.8) to

$$q^{(6)} = 0 \tag{A.9}$$

which means that any function which has its 6^{th} derivative equal to zero will minimize the jerk.

Richardson discussed in [119] why a functional that minimizes a higher order derivative would not be more smooth for reaching movements. It was found that with increasing order of the derivative n_{od}, the solution to the functional $q(t)$ approaches a step function. This means that with increasing n_{od}, the peak speed also increases, with regard to the average speed. A ratio r_v can then be defined which relates the average speed and the peak speed. Table A.1 lists this ratio for different orders of derivative. Psychophysical experiments done by Flash et al. in [45] revealed that human reaching movements have a ratio r_v equal to 1.75, which most resembles a minimum jerk trajectory (i.e., where $n_{od} = 3$).

Table A.1: ratio average-peak velocity for order of derivative n_{od}

	$n_{od} = 2$	$n_{od} = 3$	$n_{od} = 4$
ratio r_v	1.5	1.875	2.186
human ratio		$r_v \approx 1.75$	

APPENDIX B

7-DOF Redundant Manipulator AMOR

Table B.1: DH parameters for redundant manipulator AMOR

i	α_i	$a_i\ [mm]$	$d_i\ [mm]$	range
1	$-\frac{\pi}{2}$	$a_1 = 62.3$	$d_1 = 155$	∞
2	$-\frac{\pi}{2}$	0	0	150°
3	$\frac{\pi}{2}$	$a_3 = 97$	$d_3 = 419.46$	240°
4	$-\frac{\pi}{2}$	0	0	165°
5	$\frac{\pi}{2}$	$a_5 = 50.2$	$d_5 = 358.2$	∞
6	$-\frac{\pi}{2}$	0	0	175°
7	0	0	$d_7 = 70$	∞

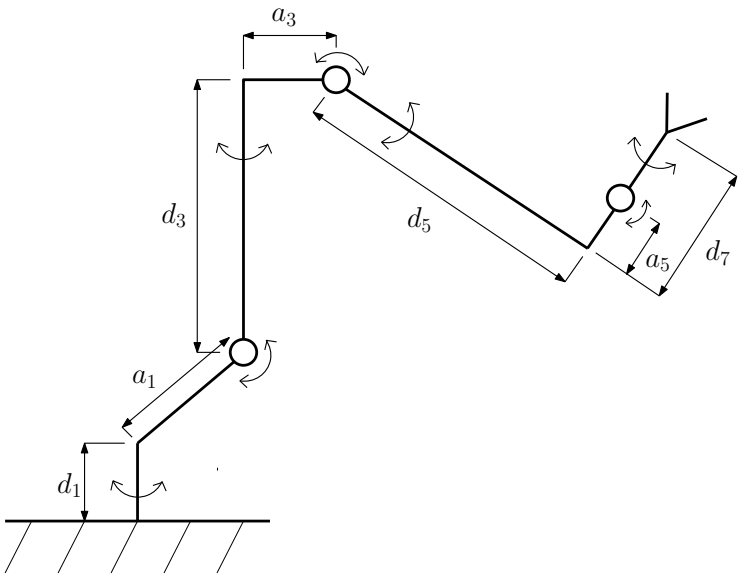


Figure B.1: Redundant 7-DOF AMOR (anthropomorphic arm) robotic manipulator developed by Exact Dynamics B.V.[1].

[1] http://www.amorrobot.com
http://www.exactdynamics.com

Bibliography

[1] M. Agrawal, K. Konolige, and M. R. Blas. Censure: Center surround extremas for realtime feature detection and matching. In *Proc. of European Conf. on Computer Vision (ECCV)*, volume 5305, pages 102–115, 2008. (53)

[2] K. Ahn, W.K. Chung, and Y. Yourn. Arbitrary states polynomial-like trajectory (ASPOT) generation. In *Proc. of Annual Conf. of the IEEE Industrial Electronics Society*, volume 1, pages 123–128, 2004. (19, 81)

[3] O. Akman. *Robust Augmented Reality*. PhD thesis, Delft University of Technology, 2012. (52)

[4] A. Alvarez-Aguirre. *Remote Control and Motion Coordination of Mobile Robots*. PhD thesis, Eindhoven University of Technology, 2011. (131)

[5] N. Andreff, B. Espiau, and R. Horaud. Visual servoing from lines. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2070–2075, 2000. (13)

[6] B. Armstrong-Hélouvry, P. Dupont, and C. Canudas de Wit. A survey of models, analysis tools and compensation methods for the control of machines with friction. *Automatica*, 30(7):1083–1138, 1994. (26, 111)

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Vangool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008. (52, 55)

[8] P. R. Beaudet. Rotationally invariant image operators. In *Proc. of the Int. Joint Conf. on Pattern Recognition (IJCPR)*, pages 579–583, 1978. (52)

[9] S. Benhimane and E. Malis. Homography-based 2D visual servoing. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2397–2402, 2006. (15)

[10] Y. Bestaoui. On line motion generation with velocity and acceleration constraints. *Robotics and Autonomous Systems*, 5(3):279–288, 1989. (20)

[11] L. Biagiotti and C. Melchiorri. *Trajectory Planning for Automatic Machines and Robots*. Springer Berlin Heidelberg, 2008. (17, 37, 38, 81)

[12] H. Bilen, M. Hocaoglu, E. Ozgur, M. Unel, and A. Sabanovic. A comparative study of conventional visual servoing schemes in microsystem applications. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1308–1313, 2007. (17)

[13] B. Bona and M. Indri. Friction compensation in robotics: an overview. In *Proc. of IEEE Conf. on Decision and Control (CDC)*, pages 4360–4367, 2005. (26, 111)

[14] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. (73, 131)

[15] D. C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971. (45)

[16] H. Bruyninckx. Some invariance problems in robotics, *Internal report*, katholieke universiteit leuven, 1991. (29)

[17] J. Caarls. *Pose estimation for mobile devices en Augmented Reality*. PhD thesis, Delft University of Technology, 2009. (50)

[18] K. Camarillo, R. Campa, V. Santibáñez, and J. Moreno-valenzuela. Stability analysis of the operational space control for industrial robots using their own joint velocity PI controllers. *Robotica*, 26(6):729–738, 2008. (29)

[19] E. Cervera, A. P. Del Pobil, F. Berry, and P. Martinet. Improving image-based visual servoing with three-dimensional features. *Int. Journal of Robotics Research*, pages 821–840, 2003. (13)

[20] A Chan. *Constraint-Aware Visual Servoing for Teaching Practical Robot Motion*. PhD thesis, The University of British Columbia, 2009. (50)

[21] A. Chan, S. Leonard, E.A. Croft, and J.J. Little. Collision-free visual servoing of an eye-in-hand manipulator via constraint-aware planning and control. In *Proc. of American Control Conference (ACC)*, pages 4642–4648, 2011. (19)

[22] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In *The Confluence of Vision and Control*, pages 66–78. LNCIS Series, No 237, 1998. (13, 15)

[23] F. Chaumette and S. Hutchinson. Visual servo control, part I: Basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006. (11, 12, 13, 14, 64, 66, 72)

[24] F. Chaumette and S. Hutchinson. Visual servo control, part II: Advanced approaches. *IEEE Robotics & Automation Magazine*, 14(1):109–118, 2007. (11, 12, 14, 64)

[25] A. Cherubini and F. Chaumette. A redundancy-based approach for obstacle avoidance in mobile robot navigation. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 5700–5705, 2010. (125)

[26] G. Chesi and K. Hashimoto. *Visual Servoing via Advanced Numerical Methods*. Springer Publishing Company, Incorporated, 1st edition, 2010. (11)

[27] G. Chesi, K. Hashimoto, D. Prattichizzo, and A. Vicino. A switching control law for keeping features in the field of view in eye-in-hand visual servoing. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 3, pages 3929–3934, 2003. (15)

[28] G. Chesi and Y.S. Hung. Global path-planning for constrained and optimal visual servoing. *IEEE Trans. on Robotics*, 23(5):1050–1060, 2007. (18)

[29] H. Choset, W. Burgard, S. Hutchinson, G. Kantor, L. E. Kavraki, K. Lynch, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2005. (18, 34)

[30] T. A. Clarke and J. G. Fryer. The development of camera calibration methods and models. *The Photogrammetric Record*, 16(91):51–66, 1998. (44)

[31] P.I. Corke. Visual control of robot manipulators - a review. In *Visual Servoing*, pages 1–31. World Scientific, 1993. (11, 12, 63)

[32] P.I. Corke. A robotics toolbox for matlab. *IEEE Robotics & Automation Magazine*, 3(1):24–32, 1996. (72, 130)

[33] P.I. Corke and S.A. Hutchinson. A new partitioned approach to image-based visual servo control. *IEEE Trans. on Robotics and Automation*, 17(4):507–515, 2001. (15)

[34] J.J.T.H de Best. *Feature-Based Motion Control for Near-Repetitive Structures*. PhD thesis, Eindhoven University of Technology, 2011. (9, 99, 100, 101, 112)

[35] J.J.T.H. de Best, R. van de Molengraft, and M. Steinbuch. High speed visual motion control applied to products with repetitive structures. *IEEE Trans. on Control Systems Technology*, 20(6):1450–1460, 2012. (100, 101)

[36] B.J. de Gans, P.C. Duineveld, and U.S. Schubert. Inkjet printing of polymers: State of the art and future developments. *Advanced Materials*, 16(3):203–213, 2004. (102)

[37] A. De Luca, G. Oriolo, and P.R. Giordano. On-line estimation of feature depth for image-based visual servoing schemes. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2823–2828, 2007. (68)

[38] K. Deguchi. Optimal motion control for image-based visual servoing by decoupling translation and rotation. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 705–711, 1998. (15)

[39] D.F. DeMenthon and L.S. Davis. Model-based object pose in 25 lines of code. *Int. Journal of Computer Vision*, 15:123–141, 1995. (12)

[40] L. Deng, F. Janabi-Sharifi, and W.J. Wilson. Hybrid motion control and planning strategies for visual servoing. *IEEE Trans. on Industrial Electronics*, 52(4):1024–1040, 2005. (68)

[41] K.L. Doty, C. Melchiorri, and C. Bonivento. A theory of generalized inverses applied to robotics. *Int. Journal of Robotics Research*, 12(1):1–19, 1993. (29)

[42] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326, 1992. (66)

[43] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, Cambridge, MA, USA, 1993. (46)

[44] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. (48)

[45] T. Flash and N. Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of Neuroscience*, 5(7):1688–1703, 1985. (37, 154)

[46] F. Fraundorfer and D. Scaramuzza. Visual odometry: Part II: Matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, 19(2):78–90, 2012. (51, 53)

[47] N.R. Gans, P.I. Corke, and S.A. Hutchinson. Performance tests of partitioned approaches to visual servo control. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1616–1623, 2002. (12, 15)

[48] N.R. Gans and S.A. Hutchinson. Stable visual servoing through hybrid switched-system control. *IEEE Trans. on Robotics*, 23(3):530–540, 2007. (15, 68)

[49] O. Gerelli and C.G.L. Bianco. A discrete-time filter for the on-line generation of trajectories with bounded velocity, acceleration, and jerk. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3989–3994, 2010. (20)

[50] R. Ginhoux, J.A. Gangloff, M.F. de Mathelin, L. Soler, M.M.A. Sanchez, and J. Marescaux. Beating heart tracking in robotic surgery using 500 hz visual servoing, model predictive control and an adaptive observer. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 274–279, 2004. (16, 99)

[51] C.F. Graetzel, S.N. Fry, and B.J. Nelson. A 6000 hz computer vision system for real-time wing beat analysis of drosophila. In *Proc. of IEEE/RAS-EMBS Int. Conf. on Biomedical Robotics and Biomechatronics (BioRob)*, pages 278–283, 2006. (16, 99)

[52] A. Gruss, S. Tada, and T. Kanade. A VLSI smart sensor for fast range imaging. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 349–358, 1992. (16)

[53] C. Guarino Lo Bianco and F. Ghilardelli. Third order system for the generation of minimum-time trajectories with asymmetric bounds on velocity, acceleration, and jerk. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Workshop on Robot Motion Planning: Online, Reactive, and in Real-time*, 2012. (20)

[54] I.J. Ha, D.H. Park, and J.H. Kwon. A novel position-based visual servoing approach for robust global stability with feature points kept within the field-of-view. In *Proc. of Int. Conf. on Control Automation Robotics Vision (ICARCV)*, pages 1458–1465, 2010. (15)

[55] A.H.A. Hafez and C.V. Jawahar. Visual servoing by optimization of a 2D/3D hybrid objective function. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1691–1696, 2007. (19)

[56] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2nd edition, 2003. (46, 48, 49)

[57] R. Haschke, E. Weitnauer, and H. Ritter. On-line planning of time-optimal, jerk-limited trajectories. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3248–3253, 2008. (20)

[58] K. Hashimoto and T. Noritsugu. Enlargement of stable region in visual servo. In *Proc. of IEEE Conf. on Decision and Control (CDC)*, pages 3927–3932, 2000. (15)

[59] Y. He, Z. Ye, D. She, B. Mesman, and H. Corporaal. Feasibility analysis of ultra high frame rate visual servoing on FPGA and SIMD processor. In *Proc. of LNCS Int. Conf. on Advanced Concepts for Intelligent Vision Systems (ACIVS)*, pages 623–634, 2011. (101)

[60] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1106–1112, 1997. (44)

[61] J. Hill and W. T. Park. Real time control of a robot with a mobile camera. In *Proc. of Int. Symp. on Industrial Robot*, pages 233–246, 1979. (11)

[62] B. Hoff. A model of duration in normal and perturbed reaching movement. *Biological Cybernetics*, 71(6):481–488, 1994. (87)

[63] N. Hogan. Adaptive control of mechanical impedance by coactivation of antagonist muscles. *IEEE Trans. on Automatic Control*, 29(8):681–690, 1984. (37, 153)

[64] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, 1996. (11, 12, 14, 98)

[65] Y. Imai, A. Namiki, K. Hashimoto, and M. Ishikawa. Dynamic active catching using a high-speed multifingered hand and a high-speed vision system. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1849–1854, 2004. (16)

[66] I. Ishii, Y. Nakabo, and M. Ishikawa. Target tracking algorithm for 1 ms visual feedback system using massively parallel processing. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2309–2314, 1996. (16)

[67] M. Ishikawa, A. Morita, and N. Takayanagi. High speed vision system using massively parallel processing. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 373–377, 1992. (16)

[68] D.Y. Shin J.G. Korvink, P.J. Smith, editor. *Overview of Inkjet-Based Micromanufacturing*. Wiley-VCH Verlag GmbH & Co. KGaA, 2012. (102)

[69] S. Kagami. High-speed vision systems and projectors for real-time perception of the world. In *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 100–107, 2010. (16)

[70] P.R. Kalata. The tracking index: A generalized parameter for α-β and α-β-γ target trackers. *IEEE Trans. on Aerospace and Electronic Systems*, AES-20(2):174–182, 1984. (102, 110, 120)

[71] P. Kallio, Quan Zhou, and H.N. Koivo. Control issues in micromanipulation. In *Proc. of Int. Symp. on Micromechatronics and Human Science*, pages 135–141, 1998. (17)

[72] R. Kelly, J. Llamas, and R. Campa. A measurement procedure for viscous and coulomb friction. *IEEE Trans. on Instrumentation and Measurement*, 49(4):857–861, 2000. (26, 118)

[73] R. Kelly, V. Santibáñez, and A. Loría. *Control of Robot Manipulators in Joint Space*. Springer-Verlag London Limited, 2005. (26)

[74] O. Kermorgant and F. Chaumette. Combining IBVS and PBVS to ensure the visibility constraint. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2849–2854, 2011. (19)

[75] H.K. Khalil. *Nonlinear Systems*. Prentice Hall, 2003. (66)

[76] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 500–505, 1985. (125)

[77] J.H. Kim, S.K Kuo, and C.H. Menq. An ultraprecision six-axis visual servo-control system. *IEEE Trans. on Robotics*, 21(5):985–993, 2005. (17)

[78] J.H. Kim and C.H. Menq. Visual servo control achieving nanometer resolution in X-Y-Z. *IEEE Trans. on Robotics*, 25(1):109–116, 2009. (17)

[79] J. Kittler and J. Illingworth. Minimum error thresholding. *Pattern Recognition*, 19(1):41–47, 1986. (105)

[80] T. Komuro, A. Iwashita, and M. Ishikawa. A QVGA-size pixel-parallel image processor for 1000-fps vision. *IEEE Micro*, 29(6):58–67, 2009. (16, 99)

[81] D. Kragic and H.I. Christensen. Survey on visual servoing for manipulation. Technical report, Computational Vision and Active Perception Laboratory, 2002. (11)

[82] T. Kröger. *On-Line Trajectory Generation in Robotic Systems*, volume 58 of *Springer Tracts in Advanced Robotics (STAR)*. Springer, Berlin, Heidelberg, Germany, 2010. (20, 81)

[83] T. Kröger and F.M. Wahl. Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. *IEEE Trans. on Robotics*, 26(1):94–111, 2010. (20)

[84] K.J. Kyriakopoulos and G.N. Saridis. Minimum jerk path generation. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 364–369, 1988. (37)

[85] V. Kyrki, D. Kragic, and H.I. Christensen. Measurement errors in visual servoing. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1861–1867, 2004. (75, 76)

[86] V. Kyrki, D. Kragic, and H.I. Christensen. New shortest-path approaches to visual servoing. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 349–354, 2004. (15, 21, 68, 69, 147)

[87] J.-C. Latombe. *Robot Motion Planning*. Kluwer, Boston, MA, 1991. (17, 18, 125)

[88] S.M. Lavalle. *Planning Algorithms*. Cambridge University Press, 2006. (17, 18, 34, 123)

[89] S.H. Lee, J.Y. Hwang, K. Kang, and H. Kang. Fabrication of organic light emitting display using inkjet printing technology. In *Proc. of Int. Symp. on Optomechatronic Technologies*, pages 71–76, 2009. (101)

[90] S. Liu. An on-line reference-trajectory generator for smooth motion of impulse-controlled industrial manipulators. In *Proc. of Int. Workshop on Advanced Motion Control (AMC)*, pages 365–370, 2002. (20)

[91] Y. Liu, J. Zhao, and B. Xie. Obstacle avoidance for redundant manipulators based on a novel gradient projection method with a functional scalar. In *IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, pages 1704–1709, 2010. (30, 130)

[92] D.G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395, 1987. (12)

[93] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004. (52, 54, 60)

[94] Y. Ma, S. Soatto, J. Kosecka, and S.S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer Publishing Company, Incorporated, 2003. (46, 49)

[95] R. Mahony, A.v. Brasch, P. Corke, and T. Hamel. Adaptive depth estimation in image based visual servo control of dynamic systems. In *Proc. of IEEE Conf. on Decision and Control (CDC)*, pages 5372–5378, 2005. (68)

[96] E. Mair, G.D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 183–196, 2010. (52)

[97] C. Makkar, G. Hu, W.G. Sawyer, and W.E. Dixon. Lyapunov-based tracking control in the presence of uncertain nonlinear parameterizable friction. *IEEE Trans. on Automatic Control*, 52(10):1988–1994, 2007. (26)

[98] E. Malis and F. Chaumette. Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods. *IEEE Trans. on Robotics and Automation*, 18(2):176–186, 2002. (15, 68, 69)

[99] E. Malis, F. Chaumette, and S. Boudet. 2-1/2D visual servoing. *IEEE Trans. on Robotics & Automation*, 15(2):238–250, 1999. (15, 69)

[100] G.L. Mariottini and D. Prattichizzo. EGT: A toolbox for multiple view geometry and visual servoing. *IEEE Robotics & Automation Magazine*, 12(4):26–39, 2005. (72, 130)

[101] C.A. Mead and M.A. Mahowald. A silicon model of early visual processing. *Neural Networks*, 1(1):91–97, 1988. (16)

[102] H. Michel and P. Rives. Singularities in the determination of the situation of a robot effector from the perspective view of 3 points. In *INRIA Research Report, Tech. Rep. 1850*. 1993. (67)

[103] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 128–142, 2002. (52)

[104] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005. (56)

[105] Y. Nakabo, M. Ishikawa, H. Toyoda, and S. Mizuno. 1 ms column parallel vision system and its application of high speed target tracking. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 650–655, 2000. (16, 99)

[106] A. Namiki and M. Ishikawa. Vision-based online trajectory generation and its application to catching. In A. Bicchi, D. Prattichizzo, and H.I. Christensen, editors, *Control Problems in Robotics*, volume 4 of *Springer Tracts in Advanced Robotics (STAR)*, pages 249–264. Springer Berlin Heidelberg, 2003. (19, 21, 81)

[107] N. Ogawa, H. Oku, K. Hashimoto, and M. Ishikawa. Microrobotic visual control of motile cells using high-speed tracking system. *IEEE Trans. on Robotics*, 21(4):704–712, 2005. (17, 98, 113)

[108] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. on Systems, Man and Cybernetics*, 9(1):62–66, 1979. (105)

[109] T. Petric and L. Zlajpah. Smooth transition between tasks on a kinematic control level: Application to self collision avoidance for two kuka LWR robots. In *IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, pages 162–167, 2011. (125)

[110] R.S. Pieters, A. Alvarez-Aguirre, P.P. Jonker, and H. Nijmeijer. Direct trajectory generation for vision-based obstacle avoidance. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Workshop on Robot Motion Planning: Online, Reactive, and in Real-time*, 2012. (9)

[111] R.S. Pieters, A. Alvarez-Aguirre, P.P. Jonker, and H. Nijmeijer. Feed forward visual servoing for object exploration. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1702–1707, 2012. (8)

[112] R.S. Pieters, P.P. Jonker, and H. Nijmeijer. Real-time center detection of an OLED structure. In *Proc. of LNCS Int. Conf. on Advanced Concepts for Intelligent Vision Systems (ACIVS)*, pages 400–409, 2009. (9)

[113] R.S. Pieters, P.P. Jonker, and H. Nijmeijer. High performance visual servoing for controlled μm-positioning. In *Proc. of IEEE World Congress on Intelligent Control and Automation (WCICA)*, pages 379–384, 2010. (9)

[114] R.S. Pieters, P.P. Jonker, and H. Nijmeijer. Product pattern-based camera calibration for microrobotics. In *Proc. of IEEE Int. Conf. of Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6, 2010. (9)

[115] R.S. Pieters, P.P. Jonker, and H. Nijmeijer. Trajectory generation for 1000 fps direct visual servoing. In *Proc. of IAPR Int. Conf. on Machine Vision Applications (MVA)*, pages 39–42, 2011. (9)

[116] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition, 2007. (86)

[117] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source Robot Operating System. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), Workshop on Open Source Robotics*, 2009. (73, 131)

[118] M. Ren, H. Gorter, J. Michels, and R. Andriessen. Ink jet technology for large area organic light-emitting diode and organic photovoltaic applications. *Journal of Imaging Science and Technology*, 55(4):1–6, 2011. (102)

[119] M.J.E. Richardson and T. Flash. Comparing smooth arm movements with the two-thirds power law and the related segmented-control hypothesis. *Journal of Neuroscience*, 22:8201–8211, 2002. (154)

[120] T.W. Ridler and S. Calvard. Picture thresholding using an iterative selection method. *IEEE Trans. on Systems, Man and Cybernetics*, 8(8):630–632, 1978. (105)

[121] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proc. of European Conf. on Computer Vision (ECCV)*, pages 430–443, 2006. (52)

[122] S.D. Roy, S. Chaudhury, and S. Banerjee. Active recognition through next view planning: a survey. *Pattern Recognition*, 37(3):429–446, 2004. (19)

[123] J. Salvi, X. Armangue, and J. Batlle. A comparative review of camera calibrating methods with accuracy evaluation. *Pattern Recognition*, 35(7):1617–1635, 2002. (44, 103)

[124] A.C. Sanderson and L.E. Weiss. Image-based visual servo control using relational graph error signals. In *Proc. of IEEE Int. Conf. on Cybernetics and Society*, pages 1074–1077, Cambridge, Massachusetts, 1980. (11)

[125] D. Santosh Kumar and C. V. Jawahar. Robust homography-based control for camera positioning in piecewise planar environments. In *Proc. of the Indian conference on Computer Vision, Graphics and Image Processing (ICVGIP)*, pages 906–918, 2006. (50)

[126] F. Schramm and A. Morel, G.and Lottin. Image based visual servoing from groups of 3D points. In *Proc. of Int. Symp. on Robotics (ISR)*, 2004. (13)

[127] E.M. Schwartz. *Algebraic properties of noncommensurate systems and their applications in robotics*. PhD thesis, University of Florida, 1995. (29)

[128] W. Scott, G. Roth, and J.F. Rivest. View Planning for Automated Three-Dimensional Object Reconstruction and Inspection. *ACM Computing Surveys*, 35(1):64–96, 2003. (19)

[129] T. Senoo, A. Namiki, and M. Ishikawa. High-speed batting using a multi-jointed manipulator. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1191–1196, 2004. (16)

[130] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168, 2004. (105)

[131] J. Shi and Tomasi. Good features to track. In *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994. (53)

[132] Z. Shiller and S. Sharma. High speed on-line motion planning in cluttered environments. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 596–601, 2012. (20)

[133] Y. Shirai and H. Inoue. Guiding a robot by visual feedback in assembling tasks. *Pattern Recognition*, 5:99–108, 1973. (11)

[134] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer-Verlag, 1st edition, 2008. (17, 25, 27, 28, 29)

[135] G. Silveira and E. Malis. Direct visual servoing: Vision-based estimation and control using only nonmetric information. *IEEE Trans. on Robotics*, 28(4):974–980, 2012. (20)

[136] S. M. Smith and J. M. Brady. SUSAN - a new approach to low level image processing. *International Journal of Computer Vision*, 23:45–78, 1995. (52)

[137] E. Staffetti, H. Bruyninckx, and J. De Schutter. *On The Invariance Of Manipulability Indices*, pages 57–66. Advances in Robot Kinematics. Springer-Verlag, 2002. (30)

[138] Y.X. Su, C.H. Zheng, P.C. Mueller, and B.Y. Duan. A simple improved velocity estimation for low-speed regions based on position measurements only. *IEEE Trans. on Control Systems Technology*, 14(5):937–942, 2006. (128)

[139] O. Tahri and F. Chaumette. Image moments: generic descriptors for decoupled image-based visual servo. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1185–1190, 2004. (13)

[140] O. Tahri and F. Chaumette. Point-based and region-based image moments for visual servoing of planar objects. *IEEE Trans. on Robotics*, 21(6):1116–1127, 2005. (13, 66)

[141] B. Tamadazte, N. Le-Fort Piat, and E. Marchand. A direct visual servoing scheme for automatic nanopositioning. *Trans. on IEEE/ASME Mechatronics*, 17(4):728–736, 2012. (17)

[142] S. Thompson and S. Kagami. Continuous curvature trajectory generation with obstacle avoidance for car-like robots. In *Proc. of Int. Conf. on Computational Intelligence for Modelling, Control and Automation (CIMCA)*, pages 863–870, 2005. (19, 81)

[143] B. Thuilot, P. Martinet, L. Cordesses, and J. Gallice. Position based visual servoing: keeping the object in the field of vision. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 2, pages 1624–1629, 2002. (13)

[144] P. Tomei. Robust adaptive friction compensation for tracking control of robot manipulators. *IEEE Trans. on Automatic Control*, 45(11):2164–2169, 2000. (26)

[145] M.-C. Tsai, I.-F. Chiu, and M.-Y. Cheng. Design and implementation of command and friction feedforward control for CNC motion controllers. *IEE Proc. on Control Theory and Applications*, 151(1):13–20, 2004. (26)

[146] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, 1987. (44, 45, 46, 104)

[147] T. Tuytelaars and K. Mikolajczyk. *Local Invariant Feature Detectors: A Survey*. Now Publishers Inc., Hanover, MA, USA, 2008. (51, 52, 56, 59)

[148] M. Vargas and E. Malis. Visual servoing based on an analytical homography decomposition. In *Proc. of IEEE Conf. on Decision and Control and European Control Conf. (CDC-ECC)*, pages 5379–5384, 2005. (50)

[149] B.J.H. Verwer. A multiresolution work space, multiresolution configuration space approach to solve the path planning problem. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2107–2112, 1990. (18)

[150] B. Vikramaditya, J.G. Lord, and B.J. Nelson. Visually servoed microspositioning for assembly of hybrid mems: theory and experiments. In *Proc. of IEEE Conf. on Decision and Control (CDC)*, pages 96–101, 1997. (17)

[151] B. Vikramaditya and B.J. Nelson. Visually guided microassembly using optical microscopes and active vision techniques. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3172–3177, 1997. (17)

[152] L. Weiss, A. Sanderson, and C. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal of Robotics and Automation*, 3(5):404–417, 1987. (12)

[153] Herman Wijshoff. The dynamics of the piezo inkjet printhead operation. *Physics Reports*, 491(4-5):77–177, 2010. (102)

[154] W.J. Wilson, C.C. Williams Hulls, and G.S. Bell. Relative end-effector control using Cartesian position based visual servoing. *IEEE Trans. on Robotics and Automation*, 12(5):684–696, 1996. (12)

[155] J.L. Wyatt, D.L. Standley, and W. Yang. The MIT vision chip project: analog VLSI systems for fast image acquisition and early vision processing. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1330–1335, 1991. (16)

[156] Z. Ye. PhD thesis, Eindhoven University of Technology, to appear, 2013. (112, 114)

[157] Z. Ye, Y. He, R.S. Pieters, B. Mesman, H. Corporaal, and P.P. Jonker. Bottlenecks and tradeoffs in high frame rate visual servoing : a case study. In *Proc. of IAPR Int. Conf. on Machine Vision Applications (MVA)*, pages 55–58, 2011. (101, 114)

[158] Z. Ye, Y. He, R.S. Pieters, B. Mesman, H. Corporaal, and P.P. Jonker. Demo: An embedded vision system for high frame rate visual servoing. In *Proc. of ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–2, 2011. (101, 114)

[159] Z. Ye, R.S. Pieters, B. Mesman, H. Corporaal, and P.P. Jonker. FPGA implementation of 1000 fps visual servoing for repetitive structures. In *Proc. of STW.ICT conference on Research in Information and Communication Technology*, 2010. (101, 114)

[160] T. Yoshikawa. Dynamic manipulability ellipsoid of robot manipulators. *Journal of Robotic Systems*, 2:113–124, 1985. (30)

[161] S. Yu and B.J. Nelson. Microrobotic cell injection. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 620–625, 2001. (17)

[162] Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000. (44, 45)

[163] J. Zhou, J. Fuh, H. Loh, Y. Wong, Y. Ng, J. Gray, and S. Chua. Characterization of drop-on-demand microdroplet printing. *The International Journal of Advanced Manufacturing Technology*, 48:243–250, 2010. (102)

[164] Y. Zhou and Bradley J. Nelson. Calibration of a parametric model of an optical microscope. *Optical Engineering*, 38(12):1989–1995, 1999. (104)

[165] Y. Zhu and P.R. Pagilla. Static and dynamic friction compensation in trajectory tracking control of robots. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2644–2649, 2002. (26)

[166] D. Zlatanov, I.A. Bonev, and C.M. Gosselin. Constraint singularities as C-space singularities. In *Proc. of Int. Symposium on Advances in Robot Kinematics (ARK)*, 2002. (33)

Samenvatting

Met de groeiende belangstelling voor de integratie van robotica in het dagelijks leven en de industrie, nemen de eisen met betrekking tot de kwaliteit en de kwantiteit van toepassingen even snel toe. Deze trend kan in het bijzonder worden herkend in toepassingen met visuele perceptie.

Aangezien in een huiselijke omgeving visuele perceptie voornamelijk wordt gebruikt voor herkenning en lokalisatie, is veiligheid de drijvende factor voor het ontwikkelen van intelligente, visuele regelalgoritmes. In het bijzonder, een robot die werkzaam is in een menselijke omgeving mag niet in botsing komen met obstakels en de uitgevoerde beweging moet zo soepel als mogelijk zijn. Bovendien, aangezien de omgeving niet op voorhand bekend is, zijn hoge eisen aan de robuustheid van beeldverwerkingsalgoritmes een noodzaak.

Anderzijds, in een industriële setting is de omgeving op voorhand bekend en wordt veiligheid hoofdzakelijk gewaarborgd door uitsluiting van een menselijke operator. Bovendien krijgen visuele regelstrategieën veel aandacht van de industrie om een standaard oplossing te worden voor robotische automatiseringstaken. Ondanks deze redenen worden applicaties sterk vereenvoudigd. Bijvoorbeeld, methoden zoals visuele foutdetectie zijn al een volwassen techniek in industriële automatisering, waar een statische camera een product observeert (bijvoorbeeld op een lopende band) en controleert of deze aan bepaalde eisen voldoet. Deze handelingen kunnen worden uitgevoerd op een relatief hoog tempo vanwege de eenvoud van het systeem (bv. statische camera) en de vereenvoudiging van de verwerkingstaak (bv. binaire beelden).

Voor beide gebieden zijn de geïdentificeerde problemen vergelijkbaar. Voornamelijk is dit het trage karakter van (robuuste) beeldverwerking, met betrekking tot de steeds groeiende vraag naar een toename in snelheid en een vermindering van vertraging. Deze twee toepassingsgebieden met overeenkomstige beperkingen motiveren het ontwerp van een meer directe benadering van vision in visuele regelsystemen. Om te voldoen aan de eisen voor volgende generatie visuele regelsystemen, worden in dit proefschrift methoden gepresenteerd die visuele metingen gebruiken als directe terugkoppeling voor bewegingsplanning.

Ten eerste, voor industriële robotica, om de vereiste positioneringsnauwkeurigheid te verkrijgen, dient het meet- en bevestigingssysteem zeer stijf en doordacht te zijn ontworpen, wat hoge kosten en een lange ontwerptijd met zich meebrengt. Door het meten van de positie van objecten direct met een camera, in plaats van indirect door motor encoders worden de eisen van het meet- en bevestigingssysteem minder veeleisend. Bovendien motiveert dit de miniaturisatie van het complete regelsysteem. Deze aanpak is experimenteel gevalideerd op een vereenvoudigde 2-dimensionale positioneertafel (dat wil zeggen, met aanzienlijke wrijving en slechte fixatie), en bereikt vergelijkbare prestaties in vergelijking met encoder-gebaseerde positioneersystemen.

Ten tweede, in een menselijke omgeving kan deze directe waarneming traditionele visuele regelsystemen verbeteren wanneer deze onderhevig zijn aan

bepaalde verstoringen. In het bijzonder wordt een methode voorgesteld die gebruik maakt van een beeld-gebaseerde feedforward regelaar bovenop een traditionele positie-gebaseerde visuele servoregeling om verstoringen zoals wrijving of slecht ontworpen lokale regelaars te overwinnen. Deze visuele feedforward regelactie is alleen actief wanneer een beeld-gebaseerde fout aanwezig is en verdwijnt wanneer deze fout naar nul gaat. De methode wordt gevalideerd op een antropomorfe robotische manipulator met 7 vrijheidsgraden, bedoeld voor gebruik in de menselijke zorg-omgeving.

Ten derde, het direct waarnemen van een product geeft aanleiding tot het direct ontwerpen van beweging. Terwijl bij traditionele methoden het bewegingstraject offline wordt ontworpen en niet kan worden gewijzigd tijdens uitvoering, kan bij directe trajectorie generatie de beweging van de volgende tijdstap berekend worden aan de hand van de huidige toestand en gebeurtenissen. Dit betekent dat op elk moment, de trajectorie van een bewegingssysteem kan worden gewijzigd met betrekking tot bepaalde gewenste kinematische of dynamische beperkingen. Voor industriële toepassingen maakt dit de productie van bijna-repetitieve of niet-starre structuren (bijvoorbeeld flexibele beeldschermen) mogelijk. Wanneer toegepast op een robotische manipulator, worden obstakels niet langer vermeden op een pad-planning niveau, maar op een trajectorie-planning niveau waar kinematische of dynamische beperkingen kunnen worden meegenomen. Dit resulteert in een beweging die soepeler is dan de beweging die wordt verkregen met het vermijden van obstakels door middel van pad-planning. Voor beide toepassingsgebieden is deze directe trajectorie generatie methode uitgevoerd en toont een hoge flexibiliteit in bewegingsontwerp.

Acknowledgements

Even though it might have seemed I've been living in isolation for the last 4 years, there are many people who have contributed in one way or the other. Therefore, I would like to express my gratitude.

Foremost, I want to thank Henk Nijmeijer and Pieter Jonker for giving me the chance to do research in this very exciting and promising field.
Henk, the freedom you gave me, the always on-the-spot comments and your eagle-eye view are all admiring. I feel privileged to have been part of your group. Pieter, your decision a long time ago to spend one day a week in Eindhoven is probably one of the reasons why I decided to stay around for another four years. Despite the chaos, it was always a pleasure to have discussions and build upon your many great ideas. I am very grateful that spending these years under both your supervision was an invaluable opportunity in my professional and scientific career.

A special thanks goes to Zhenyu for all your hard work and commitment. Your level of helpfulness is impressive; I will no longer bother you for that one extra experiment. The same appreciation goes out to Alejandro. All our discussions (robotics related or not), the (not) fixing of the robot, the constant wondering if you would show up or not; I have enjoyed every moment.

Much appreciation also goes out to Henk Corporaal, Dragan Kostić and everyone else (Zhenyu, Mark and all students) involved in our Embedded Visual Control project. Even though things did not always go as planned, it was one big learning experience for me, which I greatly appreciate.

The projects I have been part of would not have been possible without the help of all the students, researchers and companies involved. Many thanks therefore goes out to all members of Fast Focus on Structures (FFOS), Embedded Vision Architecture (EVA) and Teleoperated Service Robot (TSR).

I would also like to thank Prof. Herman Bruyninckx, Prof. Stefano Stramigioli and Prof. Kees van Hee for being part of my PhD committee and all the valuable comments and feedback.

These last years would not have been bearable without some distraction (and proper complaining?) from work. I would like to thank Nandra for putting up with me for all those years and all the rest of the group (Alejandro, Alper, America, Carlos, Jonatan, Mark, Zhenyu) for making a great work atmosphere.

A decent break from work can be inspiring. A decent break from work to visit Finnish grounds is priceless. For that I thank Kari & Liisa, Tapio, Tuomo & Rosa and Leena. Besides north, many thanks also go out to the south. Mam & pap, Romy & Bart, Marise & Rick, although it was sometimes a mystery what I did all day, and that I was sometimes missing for weeks, without you I would not be where I stand today.

Most of all, I thank Elina. Oh, the hardships I have put you through; I am forever in your debt. I am truly excited what the future will bring. Wherever it will be, home is wherever I'm with you.

Roel Pieters
March 2013

Curriculum Vitae

December 22, 1982
Born in Meerssen, The Netherlands.

1995-2000
Secondary school ('HAVO') at Stella Maris College, Meerssen, The Netherlands.

2000-2004
Bachelor degree (ing.) in Mechanical Engineering, Hogeschool Zuyd (University of Applied Sciences), Heerlen, The Netherlands. Section: Industrial Automation. Graduation project: 'Development of an automated pump test rig'. Diploma July 2004.

2004-2009
Master degree (ir.) in Mechanical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands. Section: Dynamics and Control. Graduation project: 'Active Vision; Directing Visual Attention'. Diploma January 2009.

2009-2013
Ph.D. at Eindhoven University of Technology, department of Mechanical Engineering, section Dynamics and Control. Thesis title: 'Direct Methods for Vision-Based Robot Control: Application and Implementation'. The project involved visual control of robotics, with a focus on two main applications and their implementation: vision-based industrial inkjet printing and vision-based obstacle avoidance for service robotics.

Stellingen

behorende bij het proefschrift

Direct Methods for Vision-Based Robot Control

Application and Implementation

1. Het meten en terugkoppelen van visuele informatie in een regelsysteem biedt voordelen wanneer de transformatie tussen product en meetlocatie onzeker of onbekend is. Een directe, relatieve meting tussen product en gereedschap omzeilt deze transformatie. (Dit proefschrift)

2. Wanneer uitgevoerd op voldoende hoge snelheid, kan de terugkoppeling van een regelsysteem uit slechts een camera-gebaseerd signaal bestaan. (Dit Proefschrift)

3. Het instantaan en 'real-time' genereren van een trajectorie voor robotische regelsystemen, maakt het mogelijk om deze trajectorie instantaan en 'real-time' aan te passen. Hiermee kan de beweging van een visueel regelsysteem worden uitgevoerd met kinematische restricties. (Dit proefschrift)

4. In analogie tot een feedforward regelactie voor verstoringen in de Cartesische of joint-ruimte, kan een feedforward regelactie ook worden toegepast voor verstoringen die optreden in camerabeelden (beeldruimte). (Dit proefschrift)

5. Kunstmatige intelligentie is geen partij voor natuurlijke domheid.

6. Kennis en kunde op een bepaald gebied wordt veelal verkregen door het maken van fouten. Ervaring is het herkennen van dezelfde fouten als deze opnieuw worden gemaakt.

7. Wat onmisbaar is, is onzichtbaar voor het oog.
   - aangepast van Antoine de Saint-Exupéry, Le petit Prince.

8. Bij een samenwerking tussen de academische wereld en de industrie dienen de wetenschappelijke voordelen niet ten koste te gaan van de financiële voordelen.

9. De paradigmaverschuiving van feitenkennis-cultuur naar opzoek-cultuur zal onze relatie tot kennis drastisch veranderen.

10. De opkomst van de zogenaamde smart-phone met het gebruik van sociale media om te allen tijde beschikbaar te zijn, heeft een averechts effect.

Roel Pieters
Maart, 2013

Propositions

accompanying the thesis

Direct Methods for Vision-Based Robot Control

Application and Implementation

1. The measurement and feedback of visual information in a control system provides advantages when the transformation between the product and the measurement location is unknown or uncertain. A direct, relative measurement between product and tool circumvents this transformation. (This thesis)

2. If sampled at a sufficiently high rate, feedback of a control system can consist of only a camera-based signal. (This thesis)

3. The instantaneous and 'real-time' generation of a trajectory for robotic control systems, enables the instantaneous and 'real-time' adaptation of this trajectory. As a result, the motion of a visual control system can be executed with kinematic constraints. (This thesis)

4. Similar to a feedforward control action to account for disturbances in Cartesian or joint-space, a feedforward control action can also be applied to account for disturbances that occur in camera images (image-space). (This thesis)

5. Artificial intelligence is no match for natural stupidity.

6. Knowledge and expertise in a particular field is often obtained by making mistakes. Experience is recognizing the same mistakes as they are made again.

7. What is essential is invisible to the eye.
   -Antoine de Saint-Exupéry, The Little Prince

8. In a collaboration between academia and industry, the scientific benefits should not be at the expense of the financial benefits.

9. The paradigm shift from 'factual knowledge'-culture to 'lookup'-culture, will change our relationship to knowledge drastically.

10. The rise of the so-called smart-phone with the use of social media to be available at all times, has the opposite effect.

Roel Pieters
March, 2013