Dribbling Control of an Omnidirectional Soccer Robot

Dissertation
der Fakultät für Informations- und Kognitionswissenschaften
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
M. Sc. Xiang Li
aus Hanzhong

Tübingen

2009

Tag der mündlichen Qualifikation: 17.06.2009	
Dekan:	Prof. Dr.-Ing. Oliver Kohlbacher
1. Berichterstatter:	Prof. Dr. Andreas Zell
2. Berichterstatter:	Prof. Dr. Andreas Schilling

Abstract

This thesis is concerned with motion control of omnidirectional robots. From developing a robot control system to designing different controllers, this thesis focuses on achieving high control performance with consideration of important issues, such as actuator dynamics, actuator saturation and constraints of robot systems. As a testbed, the motion control of an omnidirectional robot of the Tübingen Attempto robot soccer team, especially the ball dribbling control of the soccer robot, has been considered in this thesis.

Before designing motion control methods, a control system combining dynamics and kinematics is adopted for the Attempto soccer robot. This architecture allows to design and test low-level controllers according to the dynamic model and high-level controllers based on the kinematic model separately. Taking actuator saturation and actuator dynamics into account, the proposed control system builds a foundation to design high-level controllers with consideration of the low-level system's performance.

Based on the robot control system, path following and orientation tracking problems of omnidirectional robots are addressed in this thesis. Since these two problems are all formulated in the form of error kinematics, the designed nonlinear controllers in this thesis can be applied to other omnidirectional robots. In order to improve the control performance and satisfy constraints of the robot system, Nonlinear Model Predictive Control (NMPC) was employed to solve the motion control problem of the omnidirectional robot. The designed NMPC scheme guarantees closed-loop stability. With the selected numerical solutions, the results of real-world experiments show the feasibility of applying NMPC on a fast moving omnidirectional robot and better control performance compared to the designed nonlinear controllers.

With respect to the dribbling control problem, this thesis focuses on two problems: ball tracking and ball dribbling. A robust H_∞ filter is first developed to estimate the ball's relative position and velocity with respect to a soccer robot when the ball is pushed by the robot. The relative position denotes whether the ball is moving away and results in changing the robot behaviors of ball dribbling and ball catching. To achieve good ball dribbling, an analytical dribbling control strategy has been developed. With the analysis of the ball's movement relative to the robot, a sufficient constraint for keeping the ball is deduced, which gives clues to choose the desired robot orientations. Then the dribbling task is achieved by controlling a reference point denoting the desired ball's center to follow a planned path and steering the robot orientation to track the desired one. This dribbling control strategy is fulfilled with the proposed nonlinear motion control method and the NMPC scheme. Real-world experiments show the high performance and efficiency of the dribbling control method.

Zusammenfassung

Die Arbeit behandelt die Regelung der Bewegung eines omnidirektionalen Roboters. Unter Berücksichtigung der Dynamik und Sättigung des Antriebs, werden ein Regelungssystem und Regler mit hoher Güte entwickelt. Als Testumgebung wird die Bewegungsregelung eines Roboters aus dem Tübingen Attempto Roboter Fußball Team, insbesondere die Regelung des Roboters den Ball zu dribbeln, vorgestellt.

Um Methoden zur Bewegungsregelung vozustellen, wird ein Regelungssystem angenommen, das auf die Kinematik und Dynamik des Attempto Fußballroboters basiert. Diese Architektur ermöglicht die separate Entwicklung und das Testen von low-level Reglern für das Dynamik-Modell und high-level Regler für das Kinematik-Modell. In Anbetracht der Dynamik und Auslastung des Antriebs, bildet das Regelsystem eine Basis für das Design von high-level Reglern, die die Güte des low-level Systems miteinbeziehen.

Auf das Regelungssystem des Roboters basierend, wird in dieser Arbeit die Pfadplanung und die Regelung der Orientierung behandelt. Beide Probleme sind so behandelt, dass sie auch auf andere omnidirektionale Robotersysteme anwendbar sind. Um die Reglergüte zu erhöhen und den Einschränkungen des omnidirektionalen Robotersystems gerecht zu werden, wurde Nonlinear Model Predictive Control (NMPC) eingesetzt. Das entwickelte NMPC-System garantiert die Stabilität des geschlossen Regelkreises. Die Ergebnisse der experimentellen Validierung ausgewählter numerischer Algorithmen beweist die Anwendbarkeit von NMPC auf sich schnell bewegenden omnidirektionalen Robotern bei verbesserter Performanz im Vergleich zu den entwickelten nichtlinearen Reglern.

In Bezug auf die Regelung des Roboters den Ball zu dribbeln, konzentriert sich diese Arbeit auf zwei Probleme, Ball Tracking und das Dribbeln des Balls. Zunächst wird ein robuster H_∞ Filter wird erstellt, um die relative Position und Geschwindigkeit zwischen dem Roboter und dem Ball zu schätzen. Diese relative Information zeigt, ob sich der Ball vom Roboter entfernt und wechselt das Verhalten des Roboters in Ball dribbeln oder Ball annnehmen. Um den Ball verlässlich zu dribbeln, wurde eine analytische Regelungsstrategie angewendet. Durch die Analyse der relativen Bewegung des Balls zum Roboter konnten Bedingungen formuliert werden, den Ball zu führen und eine bestimmte Orientierung einzunehmen. Das Dribbeln wird durch die Regelung eines Referenzpunktes erreicht, der den gewünschten Mittelpunkt des Balls angibt und einen geplanten Pfad folgt, den der Roboter mit gewünschter Orientierung verfolgt. Diese Dribbelregelung wird durch die vorgestellte nichtlineare Bewegungsregelung und das NMPC-System erreicht. Experimentelle Ergebnisse zeigen die große Performanz und die Effizienz der Dribbelregelung.

Acknowledgements

First of all I would like to thank my supervisor, Prof. Dr. Andreas Zell, for his help, support, encouragement and confidence throughout my work, especially for giving me this chance which extended my experience not only of the research work but also of life. I would also thank my external examiner, Prof. Andreas Schilling, for his assessment of this work. I gratefully acknowledge the financial support of the PhD scholarship from the Landesgraduiertenförderungsgesetz and Landesgraduiertenförderungsverordnung of Baden-Württemberg.

I would like to thank Dr. Patrick Heinemann and all members of the Attempto robot soccer team, especially Jügen Haase, Hannes Becker and Sebastian Scherer for their help in experiments with the Attempto soccer robots.

Furthermore, I have to say thanks a lot to my colleagues, to Kiattisin Kanjanawanishkul and Marius Hofmeister for the fruitful discussion and cooperation, to Marcel Kronfeld, Karsten Bohlmann, Christian Weiss, Philippe Komma and Kiattisin Kanjanawanishkul for reading and commanding my papers and the drafts of this thesis, to Karl E. Wenzel for the interesting discussions when we learned the ”Informatik III” together, to Klaus Beyreuther for his technical support of computers and software, to Vita Serbakova for her encouragement.

Finally, I thank my family for their unquestioning support and the great person who came along with me through all the years: Ning.

Contents

Chapter 1

Introduction

1.1 Motivation

Wheeled Mobile Robots have received considerable attention and achiev-ed tremendous progress in industries and service robotics because of their flexible motion capabilities on reasonably smooth grounds and surfaces [153]. From the view of controllable degrees of freedom on the plane, Wheeled Mobile Robots can be categorized into two types: holonomic (or omnidirectional) robots and nonholonomic robots. Omnidirectional robots are able to move in any direction at any time regardless of their orientation. In contrast, nonholonomic robots have less than three simultaneous degrees of freedom. The highly maneuverable characteristics make omnidirectional robots very attractive in wheeled mobile robot applications. For example, some kinds of omnidirectional wheelchairs have been developed for human assistance in public environments such as residences, offices and hospitals [156, 159]. In the annual RoboCup competition, which deals with highly dynamic environments, omnidirectional soccer robots have been employed successfully since 2000 in the Small Size League [29, 133] and in nearly all the Middle Size League RoboCup teams in recent years [120, 19, 92]. Therefore, designing high-performance motion controllers is always an important and attractive topic for omnidirectional robots.

Normally, mobile robot control systems are built on robot models. The combination of kinematic and dynamic models has been widely used in robot control systems [156, 116, 140]. The main advantage of this control system is that kinematic models have simple structure and dynamic models are simplified by only taking the inputs of kinematic models as their output variables. By assigning a control task to different parts of the control system, controllers for the kinematic and dynamic models can be designed separately.

Assuming that no wheel slippage occurs, that all sensors have high accuracy and that the ground is planar enough, kinematic models have been well employed in robot motion control. As the inputs of kinematic models are wheel velocities and outputs are linear and angular velocities, the actuator dynamics of a robot are assumed to be fast enough to be ignored, which means that the desired wheel velocities can be achieved immediately. However, the actuator dynamics are impossible to be omitted in real situations, and limit and degrade the robot performance. Moreover, motor speeds of the robot wheels are constrained. When the desired robot wheel velocities exceed their maximum values, actuator saturation appears, which affects the robot performance and even destroys the stability of the controlled robot system [73, 27]. Therefore, actuator saturation is another important practical issue to be coped with when controlling mobile robots.

With respect to the nonlinear characteristics of kinematic models of omnidirectional robots, nonlinear controllers are widely used to achieve satisfying performance in the motion control problems, for example, in the path following problem [7, 125, 53, 40, 105, 35, 98]. However, these controllers rarely take robot constraints into account, which are crucial factors capable of degrading the robot performance. Moreover, only the errors between the current robot states and the desired states are considered in most control laws, while improving the control performance by considering more information of the control task is ignored.

Motivated by the practical issues of controlling mobile robots, this thesis is aiming to design such motion controllers of omnidirectional robots: the controllers are designed based on robot kinematic models and consider actuator dynamics, they guarantee closed-loop stability even though actuator saturation occurs, they achieve high control performance and satisfy constraints of robot systems. As a testbed, the motion control problems of an omnidirectional robot of the Tübingen Attempto robot soccer team, especially the ball dribbling problem of the soccer robot, have been treated in this thesis.

1.2 RoboCup

The Tübingen Attempto robot soccer team belongs to the RoboCup Middle Size League.

RoboCup (Originally called Robot World Cup Initiative) is an international research and education initiative. It is an attempt to foster AI and intelligent robotics research by providing a standard problem where a wide range of technologies can be integrated and examined, as well as being used for integrated project-oriented education.

For this purpose, RoboCup chose to use the soccer game as a primary domain, and organizes RoboCup. In order for a robot team to actually perform a soccer game, various technologies must be incorporated, including design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning, robotics, and sensor-fusion. RoboCup is a task for a team of multiple fast-moving robots under a dynamic environment. RoboCup also offers a software platform for research on the software aspects of RoboCup. [139]

After the first public announcement in September 1993, RoboCup has been held every year from 1997 to 2008 in different places around the world and has become the most successful international robot tournament. From 38 participating teams in the first RoboCup held in 1997 to about 400 teams and 2000 participants in RoboCup 2008, RoboCup has received more and more attention in the areas of robotics and artificial intelligence, and provides an international platform to show and compare the scientific progress in different teams.

The ultimate goal of the RoboCup project [138] is:

"*By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, comply with the official rule of the FIFA, against the winner of the most recent World Cup.*"

Compared to the other four leagues, *Small Size League*, *Simulation League*, *Four-legged League* and *Humanoid League*, in the *RoboCupSoccer* domain, the *Middle Size league* (MSL) was the most sophisticated league before the introduction of the *Humanoid League*. The body size of 50 cm x 50 cm x 85 cm gives MSL robots enough space to carry powerful sensors, actuators, computer systems and energy supplies. With distributed intelligence, MSL robots are able to play the soccer game autonomously, interacting with human referees. Via a wireless network, the commands of referees are submitted to the robots with a graphical user interface called *Referee Box* [41]. This communication network also enables the information exchange among teammates.

From the establishment of MSL in 1999 to RoboCup 2008, the MSL robot soccer teams have achieved great progress in the hardware development. Particularly the robot platforms have evolved from commercial differential drive systems, such as the Pioneer robot [70], to self-developed omnidirectional systems. Equipped with more efficient sensors, powerful computer and actuator systems, omnidirectional soccer robots have higher maneuverability and mobility. However, there were still many disappointing scenes happened in the past RoboCup games, for example, many robots drove out of the boundary lines and persisted moving in the false direction, many robots did not react to the ball even when it was in the robots' neighborhood, few robots were able to dribble the ball along a curve. Most of these observations show that the improvement of motion control are highly requested by the basic soccer playing skills of the RoboCup soccer robots. One such crucial basic skill is ball dribbling, which means that a robot is capable of handling a rolling soccer ball and maneuvering it to a desired position. Ball dribbling has been a special challenge topic in recent RoboCup tournaments, with the following setup and tasks [2]:

"*Six to eight black obstacles (length/width 40 cm, height 60 cm) are put at arbitrary positions on the field. The ball is put on the middle of the penalty area line, and a robot inside the same goal. The robot should dribble the ball into the opposite goal within 90 seconds, while it avoids all obstacles.*"

As the rules admit that a robot can only cover up to 30% of the ball's diameter and forbid the robot to hold the ball, it is difficult for the robot to handle the ball when the ball must move along a curve. If the robot can not provide the ball with enough centripetal force, the ball may get lost from the robot and towards the outside of the curve. Therefore, dribbling control includes the substantial requirement for the robot movement to take a proper orientation and exert suitable force on the ball.

1.3 Contributions

This thesis is concerned with the motion control of an omnidirectional soccer robot and focused in particular on the dribbling control problem. Besides the RoboCup domain, the control system and control methods presented in this theses can also be applied to other omnidirectional robots. The main contributions of this thesis include:

- The introduction of a robot control system, which allows the designer to divide the control tasks into different parts and assign them to different levels in the control system's architecture.

- The consideration of robot constraints and the identified dynamics of the low-level system and into the controller design of the high-level control system. This guarantees stability of the whole system respecting the system constraints.

- The formulation of the path following problem based on the following error kinematics, which makes the designed path following control methods independent from the specific platforms of omnidirectional robots.

- The application of Nonlinear Model Predictive Control (NMPC) to the robot motion control problem, which shows good control performance and feasibility with a fast moving omnidirectional soccer robot.

- The development of a robust H_∞ filter to track a rolling ball when it is dribbled by a soccer robot. Compared to a Kalman filter in real world experiments, the H_∞ filter shows better performance.

- The development of an analytical dribbling control strategy, which divides the dribbling problem into path following and ball keeping problems and is achieved with the high mobility of omnidirectional robots.

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

- Chapter 2 presents the hardware and software systems of the Attempto Tübingen soccer robots, which are used as a basis for real world experiments.

- Chapter 3 presents the motion control system of the Attempto Tübingen soccer robots, which combines the kinematic and dynamic models of the robot. With the identification of actuator dynamics and the analysis of actuator saturation, the overall control system builds a foundation to design controllers with consideration of the system constraints.

- Chapter 4 focuses on designing the robot motion control laws of the path following and orientation tracking problems. According to the different ways of choosing the desired robot position, the path following problem is formulated in the orthogonal projection-based case and the *Virtual Vehicle*-based case. The control laws with respect to these two formulations have been addressed. A PD controller was designed for a robot to track the desired orientations which takes the maximum wheel velocity into account.

- Chapter 5 addresses Nonlinear Model Predictive Control (NMPC) applied to the Attempto Tübingen soccer robots. As two important issues in the application of NMPC, stability and numerical solutions of NMPC are considered after introducing the mathematical formulation of NMPC.

- Chapter 6 presents a H_∞ filter to estimate the ball's relative position and velocity with respect to a soccer robot, when the ball is dibbled by the robot. The performance of the H_∞ filter is evaluated by comparing the estimation values with those from a Kalman filter.

- Chapter 7 addresses the dribbling control strategy of the Attempto Tübingen soccer robots. The motion control laws presented in Chapter 4 and Chapter

5 were used to fulfill the strategy. Their control performance is evaluated with real world experiments.

- Chapter 8 summarizes the achieved results and provides an outlook on possible future research directions.

- Appendix A shows some image scenarios of dribbling experiments in the robot laboratory and two scenarios in the games of RoboCup 2006 in Bremen.

Chapter 2

Robot System

This chapter presents hardware and software systems of the Attempto Tübingen soccer robots. The newly developed components of the Attempto Tübingen soccer robots from 2003 to 2006 mainly consist of the omnidirectional robot platforms, the omnidirectional vision system and the electro-magnetic kick system. The 20 ms cycle time of the new software system is a great benefit for the robots playing soccer in a highly dynamic environment.

2.1 Hardware System

The Attempto Tübingen robot soccer team has evolved from differential-drive robots to omnidirectional robots from 1997 to 2006. After several years participation in the international robot soccer competitions, the differential-drive soccer robots showed big difficulties to play against fast and agile opponents. At the same time, the updated rules of RoboCup Middle Size League promoted the improvement of sensors and computational systems of the soccer robots. In 2003, the old goalkeeper based on the Pioneer 1-AT platform from *MobileRobots Inc.* [71] was renewed with a new omnidirectional platform. After real tests of the omnidirectional robot at the RoboCup 2004 in Lisbon, the good mobility performance prompted to renew the old field players, which were built on the Pioneer 2-DX platform from *MobileRobots Inc.* [72]. Figure 2.1 shows the Attempto Tübingen soccer robots at the RoboCup 2006 in Bremen. They are not only reformed with the omnidirectional platforms, but also equipped with new sensors and a new computational system.

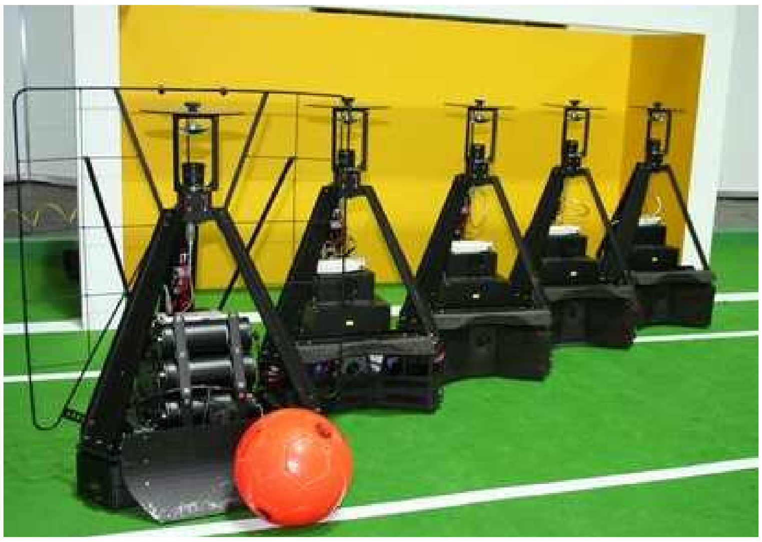


Figure 2.1: The Attempto Tübingen soccer robots in RoboCup 2006 in Bremen.

2.1.1 Platform

Considering a stable structure and the higher mobility, a triangular omnidirectional robot platform was adopted for the new generation of the Attempto Tübingen robot soccer team. The main feature of the platform is three *Swedish 90-degree* wheels [151] from *TRAPOROL GmbH* [56], which have a diameter of 80 mm as shown in Figure 2.2. The six small rollers mounted along the wheel's periphery enable a movement of the wheel perpendicular to the normal rotating direction of the wheel's axis. When the three wheels are driven separately by three DC motors, and the wheel-motor combinations are assembled symmetrically with 120 degrees between each other in a solid frame as illustrated in Figure 2.3, an omnidirectional drive results. The frame can move into any direction while tracking any orientation.

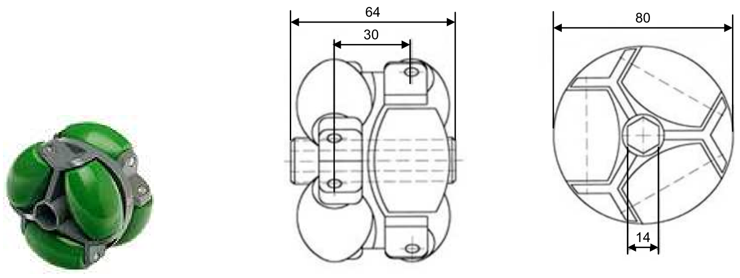
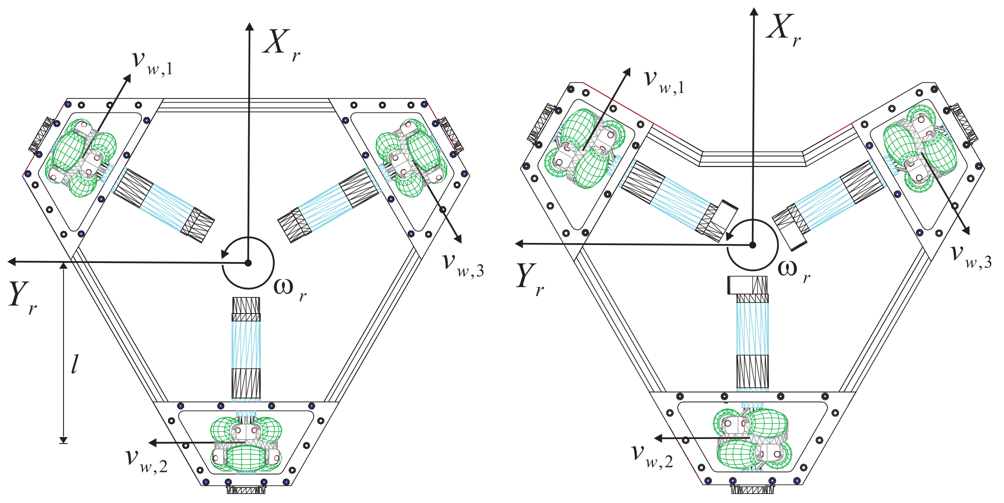


Figure 2.2: The ARG 80 *Swedish* wheel [56].

The size of the frame is determined by choosing the distance l from the wheel's center to the robot center as shown in Figure 2.3. For the goalkeeper, l is 21 cm, which is fitting the maximum size of 50×50 cm^2, seen in Figure 2.4. For the field players, it is decreased to 19.5 cm, which permits a higher rotation velocity for ball handling.



(a) Motor frame of the goalkeeper. (b) Motor frame of a field player.

Figure 2.3: Motor frames of the Attempto Tübingen soccer robots [64].

Surrounding the frame, aluminum profiles were chosen to form a stable and lightweight body. As it is noticed in Figure 2.3, the form of the field player is slightly different from the goalkeeper. There is an indentation at the front of the field player with a depth of one third of the ball's diameter. This indentation helps keeping the ball near the center of the robot's front when a field player dribbles the ball.

2.1.2 Equipment

On-Board Computer

Instead of the old computer system, which used a *coolMONSTER/P3* PISA slot CPU from JUMPtec AG [3] equipped with an *Intel Pentium-III* 850 MHz CPU and 512 MB RAM, the new on-board computer system adopts a *Thunderbird Mini-ITX* motherboard from *Lippert GmbH* [55] equipped with a 2.0 GHz Intel Pentium M processor and 1 GB RAM. *Scientific Linux 4.0* was taken as the operating system and installed with a minimum size. In order to bring the hard disk into a safe state withstanding the shocks in real robot soccer games, the operating system codes and user programs are loaded into a 32 MB RAM disk at the startup of the computer. This RAM disk is a specially reserved part of the computer's RAM and

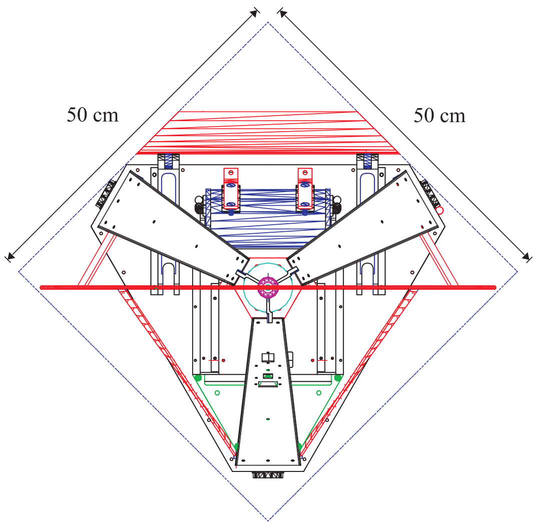


Figure 2.4: The size of the goalkeeper [64].

can be addressed like a normal disk drive. The computer system is fast enough to keep a stable 20 ms cycle time of the software system with only a 25 Watt power consumption.

The other new equipment in the computer system is an external IEEE 802.11b and 802.11a compatible WLAN bridge, which supports using the IEEE 802 11a standard with a possibly higher bandwidth of 54 Mbit/s to reduce the interference from other leagues at RoboCup tournaments.

Sensors

Because the boundary walls on the play field had to be removed according to the new rules of the RoboCup Middle Size League, the laser scanner assembled on the old robot platform became useless for the robot self-localization. The new omnidirectional soccer robot adopted an omnidirectional vision system as the sole sensor. Supported by efficient image processing algorithms, the omnidirectional vision system has been successfully used and become a trend in robot soccer teams of the RoboCup Middle Size League.

The omnidirectional vision system of the Attempto soccer robot consists of a hyperbolic mirror from the *Fraunhofer Institute for Autonomous Intelligent System* [48] and a *Marlin F-046C* camera from *Allied Vision Technologies GmbH* [54]. The camera is assembled pointing up to the hyperbolic mirror, which is on the top of the robot as shown in Figure 2.5. The omnidirectional images of the surrounding environment are transmitted to the on-board computer through an *IEEE 1394a FireWire* bus system. The *Marlin F-046C* camera provides a maximum resolution of 780×580 pixels and is able to capture and transmit images with 780×580 pixels in the 16 bit *YUV 4:2:2* format up to 50 times per second. This characteristic is appropriate for the target software cycle time of 20 ms and makes use of the full bandwidth of the *IEEE 1394a FireWire* bus system.

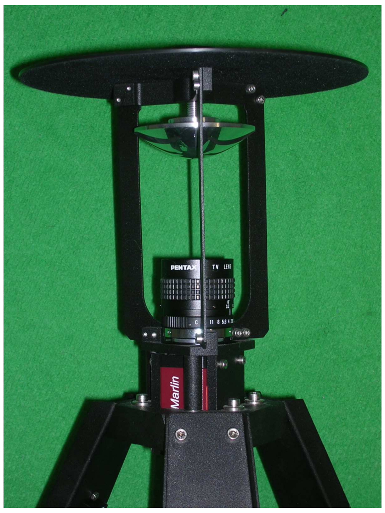


Figure 2.5: The omnidirectional vision system with a hyperbolic mirror and a *Marlin F-046C* camera.

Actuator

The actuator system is composed of three *RE 30* DC motors with a power of 60 Watt and a maximum 8200 revolutions per minute from *Maxon Motor AG* [4]. This motor is equipped with a *GP 32 C* ceramic planetary gear box with a gear ratio of 18:1, and an *MR* wheel encoder with 500 impulses per revolution. Each Swedish wheel is driven by a Maxon DC motor with the maximum wheel velocity of 1.9 m/s, which gives the robot a maximum moving velocity of 2.2 m/s according to the robot kinematics.

The control of DC motors is implemented by the triple motor controller board TMC200 developed by the *Fraunhofer Institute for Autonomous Intelligent Systems* [87]. This controller board has three independent channels supporting up to three DC motors with a maximum continuous load of 200 Watt. It not only allows speed control, torque control, thermal motor protection and operating voltage monitoring, but also offers plenty of feedback messages, for example, the actual velocity, the actual current and the odometry value. The two alternative communication interfaces: CAN bus and RS232 serial interface, make TMC200 easily connect to the on-board computer. All the parameters of the control board

TMC200, such as controller constants, motor parameters and modes of operation, can be set via the respective communication interface. The RS232 serial interface was chosen by the Attempto soccer robot with the setup of 57,600 baud rate, which is able to handle fifty commands per second and supports the 20 ms cycle time of the whole system.

Dribblers

The robot dribbling system consists of dribblers aiming to increase the robot's ability of controlling the ball. The main contribution of dribblers is exerting some force onto the ball. This force can prevent the ball from sliding away from the robot when the robot has a fast rotation. The dribbling system of the Attempto soccer robot shown in Figure 2.6 and 2.7 is based on the indentation of the robot front. Three spongy blocks are pasted on the indentation to damp the collisions between the ball and the robot, and prevent the ball from sliding away. A rubber foam pad is assembled at the top of the indentation, which exerts pressure onto the ball and keeps it from leaving the robot along the forward direction.

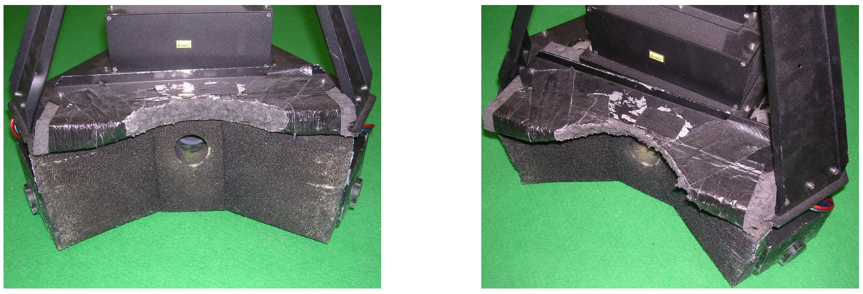

(a) Front view of dribblers. (b) Side view of dribblers.

Figure 2.6: The dribbling system composed of three spongy blocks pasted on the indentation and a rubber foam pad at the top of the indentation.

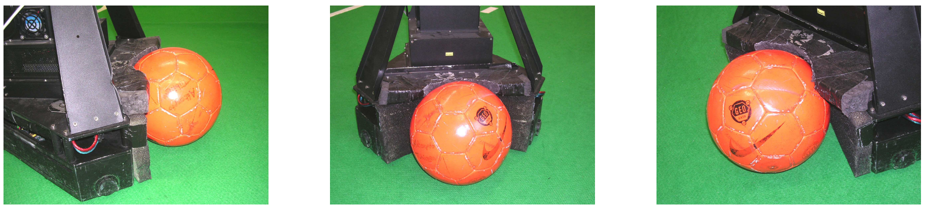

(a) Left side view. (b) Front view. (c) Right side view.

Figure 2.7: Photos of dribblers keeping the ball.

Kicker

The new electro-magnetic kick system shown in Figure 2.8 has the advantage of controlling the strength of the kick, which is of benefit for passing the ball. The main component of the kicker system is a coil with an inductance of 9.6 mH, which is made by winding 690 circles of a magnet wire of 0.8 mm diameter around a plastic tube of 100 mm length and 40 mm diameter. A rob is inside the coil, which is composed of a steel cylinder of 100 mm length, a nylon cylinder in the front and a thin appendix in the back. When a high current passes through the windings, the steel part of the rob will be accelerated into the coil by the produced magnetic field and the nylon part will move outside to kick the ball.

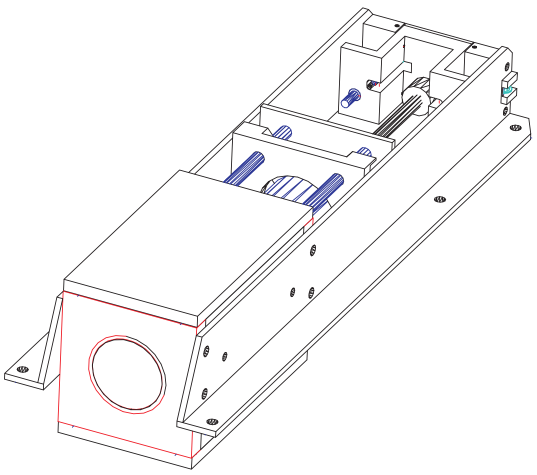


Figure 2.8: The CAD model of the electro-magnetic kicker [64].

Controlling the strength of the kick is achieved through the I/O port of the motor controller board TMC200. 6 bits of the port are used to determine the duration of the current flow. The other 2 bits control the voltage of the electronic circuit to exert 85% or 95% of the maximum value. As a consequence, the electonic circuit enables the kicker to have 128 different strengths and to accelerate the ball to a maximum speed of nearly 10 m/s.

After kicking the ball, the kicker has to go back to the home position. A rubber spring with a nearly constant spring force for a certain range of deflection is attached to the thin appendix of the rob. This spring is able to return the rob from any position. To keep the rob at the home position when a kick is not requested, a small permanent magnet is fixed at the rear side of the kicker frame.

2.2 Software System

The software system of the Attempto Tübingen robot soccer team was designed as a common *Programming Interface*, which is not only used in RoboCup tournaments but also suitable for other mobile robot applications. To simplify the design and test, this software system consists of several independent functional processes. Utilizing a client/server architecture, the data transmission among processes allows access to data sources of each process. The current computer system uses a single processor, which matches the single data flow in the current software system with a global cycle time of 20 ms. When more sensors or a shorter global cycle time is required, the multi-processes software system could support the parallel computation on a multi-processor system. As shown in Figure 2.9, the

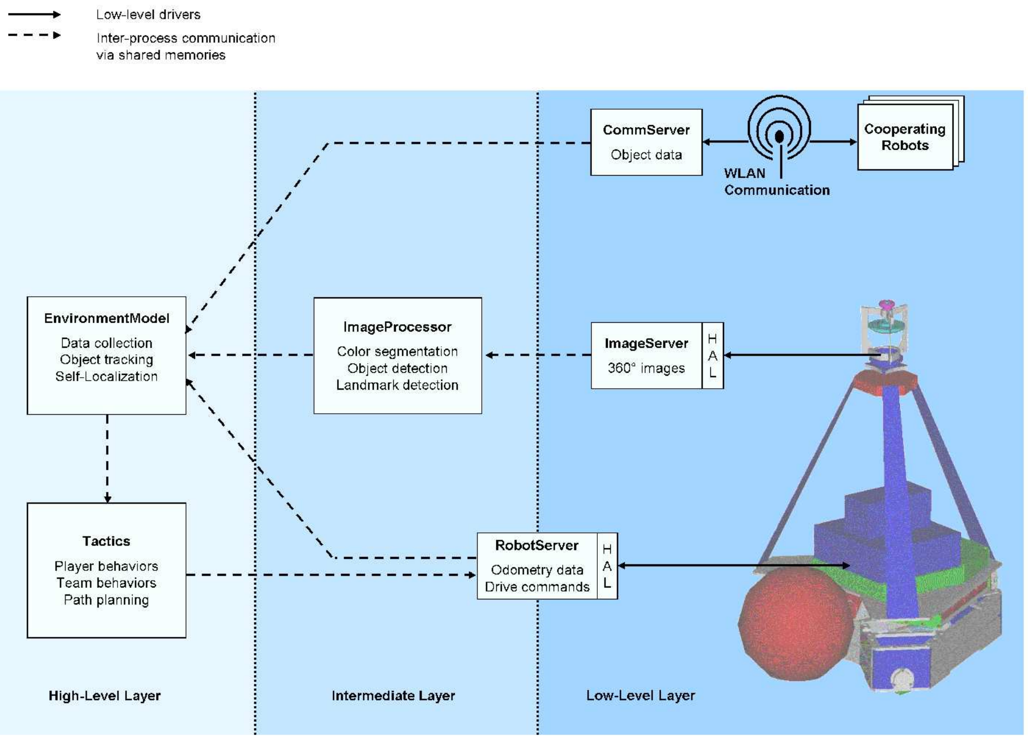


Figure 2.9: An overview of the software system [64].

software system can be divided into three functional levels. The low-level system aims to access all sensors' data and to perform pre-processing. The middle-level system processes the pre-processed data from the low-level system, such as extracting landmarks and obstacles from the processed images. The high-level system builds an environment model and fulfills the high-level control of the robot, for example, catching the ball, dribbling the ball and shooting a goal.

2.2.1 Low-Level System

The low-level system consists of three processes that provide access to the raw sensor's data and the data from teammates [64].

ImageServer uses the open source library *libdc1394* [38] controlling the *Marlin F-046C* camera to capture raw images and serves the image data to other processes.

RobotServer communicates with the TMC200 motor controller board and has two functions. One is to act as a client of the robot control process **Tactics**, receive control commands and send them to the controller board. The other is to work as a server providing robot data to other processes. The robot data includes the rotational speed and the encoder count of the motors, which are used to compute the odometry information of the robot. Moreover, **RobotServer** collects the battery voltage and the kicker capacity in each second, which are needed by the process **Tactics**.

CommServer is responsible for the communication with teammates. It acts as a client of the process **EnvironmentModel** and sends the own robot's data to teammates. On the other hand, it receives messages from teammates and serves them to the higher level system.

2.2.2 Middle-Level System

The middle-level system includes only one process **ImageProcessor** [68]. It acts as a client of **ImageServer** to get the raw image data, and processes images to extract all necessary features, such as landmarks and objects on the play field, then serves these features to other processes.

2.2.3 High-Level System

The high-level system is composed of two processes **EnvironmentModel** and **Tactics**.

EnvironmentModel integrates the processed image data from **ImageProcessor** and teammates' information from **CommServer** to build an environment model of the own robot. Using the landmarks in the images from the omnidirectional vision system, a self-localization algorithm [65] works to find the robot position in the environment model. With the objects information extracted from the images, an objects tracking algorithm [66] provides the information of the ball, opponents and teammates, such as their positions and velocities.

The process **Tactics** focuses on the high level control problems [64]. It processes the information from **EnvironmentModel** and sends driving commands to the motor controller, such that the robot can successfully play a soccer game in cooperation with its teammates. For the goal keeper, a reactive behavior-based system based on simple condition-action rules was designed, which mainly depends on the ball's position in the environment model. The software of field players utilizes a hybrid control system in controlling the robot to fulfill designed behaviors, for example, dribble, pass and shoot the ball. A path planning algorithm is also designed to plan an efficient collision-free path to navigate the robot movement.

Chapter 3

Robot Control System

This chapter presents the first step of controlling a mobile robot, which is to build up a robot control system. A control system is the foundation of designing robot control laws, and a suitable control system can benefit more the controller design with respect to control tasks. After a short introduction of wheeled mobile robots, the control system of the Attempto soccer robots is addressed in Section 3.2. Taking a reasonable architecture, the adopted control system of the Attempto soccer robot consists of a high-level control system and a low-level control system, which are based on the robot kinematic model and dynamic model, respectively. The robot models and corresponding control laws are detailed in sections 3.3 and 3.4. While the actuator characteristics may have severe impact on the control strategy based on kinematic models, the actuator dynamics and actuator saturation have to be considered as well, which are described in Section 3.5.

3.1 Wheeled Mobile Robots

Wheeled Mobile Robots have received considerable attention and achiev-ed tremendous progress in industries and service robotics because of their flexible motion capabilities on reasonably smooth grounds and surfaces [153]. A wheeled mobile robot can be modeled as a planar rigid body that rides on an arbitrary number of wheels, some or all of which can be steered [8]. Ignoring the DOF (Degrees of Freedom) of the wheel axles and wheel joints relative to the robot body, a wheeled mobile robot has maximum of three DOF: two of them are with respect to the translation on the plane, the other one is the rotation along the vertical axis, which is orthogonal to the plane.

The maneuverability of a wheeled mobile robot depends on wheel types and configurations [151]. From the view of controllable DOF, wheeled mobile robots can be categorized into two types: omnidirectional (or holonomic) robots and nonholonomic robots.

Nonholonomic robots have less than three controllable DOF, because the commonly used wheels are unable to move parallel to their axes. For example, an unicycle has only an upright wheel rolling on the plane, which is the simplest example of the nonholonomic wheeled mobile robot. Such constraints are nonholonomic, i.e. they cannot be integrated to give a constraint only on the robot poses [121]. With the generalized coordinates (x, y, θ), the nonholonomic constraint can be expressed as

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0, \tag{3.1}$$

where (x, y) denotes the robot position and θ is the robot orientation. Nonholonomic constraints allow the robot to only move forward and backward after changing the direction of its movement. Due to the wide application environments covering from research laboratories and factories to the daily life, the development of the nonlinear control theory and practice with respect to nonholonomic systems is still an attractive and challenging research topic.

Omnidirectional robots have higher maneuverability than nonholonomic robots, because the DOF of an omnidirectional robot can be totally controlled. The omnidirectional movement means that a robot can move in any direction at any time regardless of its orientation. Although the omnidirectional mobility can be achieved by using conventional wheels, for example caster wheels and steering wheels, these designs are not truly omnidirectional because of the nonholonomic nature of these wheels [43]. Therefore, true omnidirectional wheels which can move in parallel to the direction of their axes are widely used in omnidirectional robot platforms, for example Swedish wheels, meccanum wheels and Ball wheels [160, 36, 37]. The highly maneuverable characteristics makes omnidirectional robots very attractive in wheeled mobile robots applications. For example, some kinds of omnidirectional wheelchairs have been developed for human assistance in public environments such as residences, offices and hospitals [156, 159]. In the annual RoboCup competition which deals with highly dynamic environments, omnidirectional soccer robots have been employed extremely successfully since 2000 in the Small Size League [29, 133] and in nearly all the Middle Size League RoboCup teams in recent years [120, 19, 92].

3.2 Control System

Mobile robot control systems are built on robot models. As a direct description of the relationship between the forces exerted on the wheels and the robot movement, robot dynamic models have been used in many robot control systems [160, 158, 51, 102, 132, 157]. Taking the torque and force or the applied voltage of wheels motors as inputs and the robot linear and angular accelerations as outputs, dynamic models are complex in structure and rely on many mechanical or physical parameters. Due to dynamic variations caused by changes in the robot's inertia moment and perturbations from the mechanic components, obtaining a perfect dynamic model of a mobile robot is a very hard task, which increases the complexity of controllers based only on dynamic models [51, 148]. Another widely adopted control system combines robot kinematic with dynamic models [156, 116, 140]. With assignments of control tasks, controllers with respect to the kinematic and dynamic models are designed separately. The main advantage of this control system is that kinematic models have a simple structure and also simplify the dynamic model by only using the kinematic model's inputs as output variables.

A common architecture of control systems combing kinematic and dynamic models, Navigation-Guidance-Control (NGC), is adopted to control the omnidirectional Attempto soccer robot, which is organized in a three-level hierarchy [103, 118]. The three levels cover the three basic problems of a mobile robot: to know where it is, to determine suitable maneuvers for desired tasks and to execute such maneuvers as well as possible. Navigation is in charge of the knowledge of robot position and attitude based on the sensors measurement. Guidance is built from high level tasks and robot kinematic models to generate desired angular and linear velocities with respect to the output of the navigation system. Control is responsible for keeping the robot at the desired velocities specified by the guidance system as close as possible. According to the hierarchical structure, the control system is also considered a low-level control system which commands the robot actuators, such as motor torques, based on the planned velocities from the guidance system. The guidance system is also taken as a high-level control system, whose objective is to plan the desired velocity trajectory and send the reference velocity to the low-level control system.

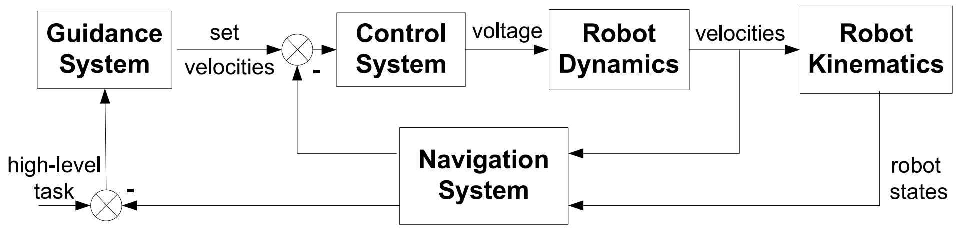


Figure 3.1: Diagram of NGC control architecture

With the assumption that no slippage of wheels occurs, all sensors have high accuracy and the ground is planar enough, kinematic models have been well used in designing robot behaviors. As the inputs of kinematic models are robot wheel velocities and outputs are robot linear and angular velocities, the actuator dynamics of the robot are assumed to be fast enough to be ignored, which means that the desired wheel velocities can be achieved immediately. However, the actuator dynamics is impossible to be omitted in real situations, and even limits and degrade the robot performance. Moreover, motor speeds of the robot wheels are bounded by saturation limits. When the desired robot wheel velocities exceed their maximum values, the actuator saturation appears which affects the robot performance and can even destroy the stability of the robot control system [73, 27]. Therefore, actuator saturation is another important practical issue to be considered in controlling mobile robots.

The next sections in this chapter describe the control system of the Attempto soccer robot in detail. High-level control, low-level control, the robot kinematic and dynamic models and the corresponding control methods will be presented. To guarantee control stability, the dynamic actuator saturation and identified actuator dynamics have been considered in the control system, which are addressed in Section 3.5.

3.3 High-Level Control

The high-level control is based on the robot kinematic model and works as the guidance system in an NGC architecture. It determines the desired robot velocities based on the measured robot information so as to fulfill the high-level control tasks, then forwards these values to the robot actuators as the objective of the low-level control system.

3.3.1 Kinematic Model

Figure 3.2 shows the base of an Attempto soccer robot. Besides the fixed world coordinate system $\{W\}$ composed of axes X_w and Y_w, a moving robot coordinate system $\{M\}$ consisting of axes X_m and Y_m is defined. The angle θ between the axes X_m and X_w denotes the robot orientation. Angles α and φ denote the direction of robot motion in the world and robot coordinate systems, respectively. Each wheel has the same distance L_w to the robot's center of mass R. δ refers to the wheel orientation in the robot coordinate system and has a constant value of 30 degrees.

The kinematic model of the robot is as follows:

$$\dot{\mathbf{x}} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{v}, \tag{3.2}$$

where $\dot{\mathbf{x}}$ is the robot velocity vector $(\dot{x}, \dot{y}, \dot{\theta})^T$ with respect to the world coordinate

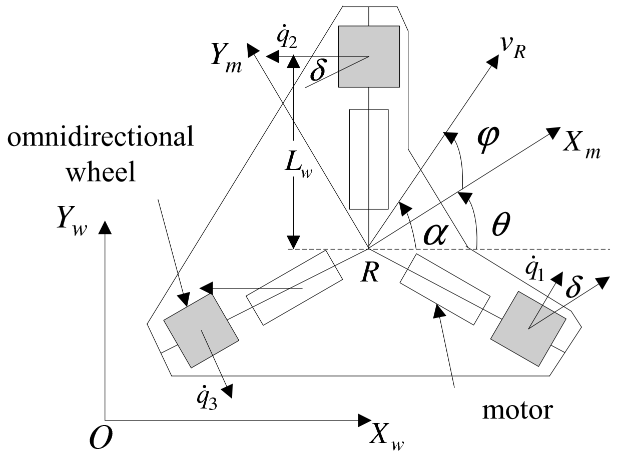


Figure 3.2: Kinematics diagram of the base of an Attempo soccer robot.

system, which is composed of the robot translation velocity $(\dot{x}, \dot{y})$ and the robot rotation velocity $\dot{\theta}$. The input is a velocity vector $\mathbf{v} = (\dot{x}_R^m, \dot{y}_R^m, \omega)^T$ with respect to the robot coordinate system, where the velocities $\dot{x}_R^m$ and $\dot{y}_R^m$ are along the axes X_m and Y_m, respectively, ω is the robot rotation velocity along the axis perpendicular to the plane.

When we consider the wheel velocities, the lower level kinematic model of the robot can be deduced as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \frac{2}{3}cos(\theta+\delta) & -\frac{2}{3}cos(\theta-\delta) & \frac{2}{3}sin\theta \\ \frac{2}{3}sin(\theta+\delta) & -\frac{2}{3}sin(\theta-\delta) & -\frac{2}{3}cos\theta \\ \frac{1}{3L_w} & \frac{1}{3L_w} & \frac{1}{3L_w} \end{bmatrix} \dot{\mathbf{q}}, \qquad (3.3)$$

where $\dot{\mathbf{q}}$ is the vector of wheel velocities $[\dot{q}_1\ \dot{q}_2\ \dot{q}_3]^T$. Here $\dot{q}_i\ (i = 1, 2, 3)$ denotes the i-th wheel velocity, which is equal to the wheel radius multiplied by the wheel angular velocity. As the motor's voltage and current are limited, the maximum wheel velocity is limited by $\dot{q}_m$, namely $|\dot{q}_i| \leq \dot{q}_m$.

It is important to notice that the transformation matrices in the kinematic models (3.2) and (3.3) are all full rank, which implies that the translation and rotation of the omnidirectional robot are completely decoupled, and allows the separate control of these two movements.

3.3.2 Control Law

It is clear that the robot kinematic model is nonlinear. When the model Eq. (3.2) changes to

$$\dot{\mathbf{x}} = \mathbf{G}\mathbf{v}, \qquad (3.4)$$

by defining the transformation matrix as $\mathbf{G}$, the trigonometric functions of angle θ in $\mathbf{G}$ show the system's nonlinearities. As $\mathbf{G}$ is full rank, its inverse can be introduced as a compensator $\mathbf{C}$ to exactly linearize this nonlinear model as shown in Figure 3.3 and generates the following linearized system

$$\dot{\mathbf{x}} = \mathbf{u} \tag{3.5}$$

with a new input vector $\mathbf{u} = (u_1\ u_2\ u_3)^T$.

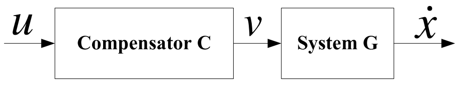


Figure 3.3: Linearized system with the compensator C.

This linearized system inherits the decoupled controllability of robot translation and rotation. When a controller K is designed based on this simple linear system, the controller of the original system is generated as CK. The overall control loop, which consists of the nonlinear system, the compensator and the controller, is shown in Figure 3.4, where $\mathbf{x}$ denotes the robot state vector $(x_R\ y_R\ \theta)^T$ and $\mathbf{x_d}$

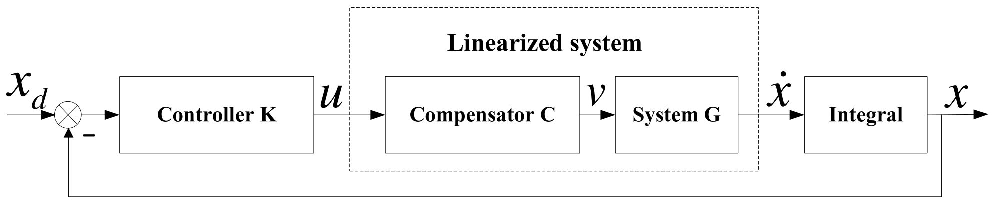


Figure 3.4: Closed-loop control system.

is the desired state vector.

With the simple system (3.5), linear control technologies can be easily used to design suitable controllers with respect to high-level control tasks. In Chapter 4, path following and orientation tracking are chosen as high-level control tasks, and the corresponding control laws are addressed in detail.

3.4 Low-Level Control

The low-level control is based on the robot dynamic model. It takes the desired robot velocities coming from the high-level control system as control objectives and controls the real robot velocities to approach the desired ones as closely as possible.

3.4.1 Dynamic Model

The dynamic model is derived from Newton's Second Law,

$$m\ddot{x}_w = F_x, \tag{3.6}$$

$$m\ddot{y}_w = F_y, \tag{3.7}$$

$$I_v\ddot{\theta} = m_I, \tag{3.8}$$

where $\ddot{x}_w$ and $\ddot{y}_w$ denote the robot translation accelerations with respect to the world coordinate system, $\ddot{\theta}$ is the robot rotation acceleration, F_x and F_y are the forces with respect to the world coordinate system exerted on the center of gravity of the robot, m denotes the robot mass, I_v is the moment of inertia of the robot, m_I is the moment around the center of gravity of the robot.

When we transfer (3.6)-(3.8) into the robot coordinate system with the following transformation matrix,

$$^{m}R_w = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix},$$

we get the following dynamic equations,

$$m(\ddot{x}_m - \dot{y}_m\dot{\theta}) = f_x, \tag{3.9}$$

$$m(\ddot{y}_m + \dot{x}_m\dot{\theta}) = f_y, \tag{3.10}$$

where $\ddot{x}_m$ and $\ddot{y}_m$ denote the robot translation accelerations with respect to the robot coordinate system, $\dot{x}_m$ and $\dot{y}_m$ are the corresponding robot translation velocities, f_x and f_y are the forces with respect to the robot coordinate system exerted on the center of gravity of the robot.

f_x, f_y and m_I can be calculated from

$$f_x = -D_1\sin\delta - D_2\sin\delta + D_3, \tag{3.11}$$

$$f_y = D_1\cos\delta - D_2\cos\delta, \tag{3.12}$$

$$m_I = (D_1 + D_2 + D_3)L_w, \tag{3.13}$$

where D_i for $i = 1, 2, 3$ is given by the following driving system's dynamics for each wheel [142],

$$I_w\ddot{\theta}_i + c\dot{\theta}_i = k_f u_i - rD_i. \tag{3.14}$$

D_i is the driving force for each wheel, I_w is the moment of inertia of the wheel, θ_i denotes the angular position of each wheel, $\dot{\theta}_i$ and $\ddot{\theta}_i$ are the corresponding angular velocity and acceleration, u_i is the driving input torque on each wheel, c

is the viscous friction factor of the wheel assembly, k_f is the driving gain factor and r is the wheel radius.

Substituting the following inverse kinematic equations referring to the relationship between wheel velocities and robot velocities,

$$\dot{\mathbf{q}} = \begin{bmatrix} -\sin\delta & \cos\delta & L_w \\ -\sin\delta & -\cos\delta & L_w \\ 1 & 0 & L_w \end{bmatrix} \mathbf{v}, \tag{3.15}$$

and their time derivatives into equations (3.9)-(3.14), the dynamic model of the robot is deduced as,

$$\ddot{x}_m = a_1\dot{x}_m + a_2\dot{y}_m\omega - b_1(u_1 + u_2 - 2u_3), \tag{3.16}$$

$$\ddot{y}_m = a_1\dot{y}_m - a_2\dot{x}_m\omega + \sqrt{3}b_1(u_1 - u_2), \tag{3.17}$$

$$\ddot{\theta} = a_3\omega + b_2(u_1 + u_2 + u_3), \tag{3.18}$$

with

$$\begin{aligned} a_1 &= -3c/(3I_w + 2mr^2) \\ a_2 &= 2mr^2/(3I_w + 2mr^2) \\ a_3 &= -3cL_w^2/(3I_wL_w^2 + I_vr^2) \\ b_1 &= k_fr/(3I_w + 2mr^2) \\ b_2 &= k_frL_w/(3I_wL_w^2 + I_vr^2). \end{aligned}$$

3.4.2 Control Law

Based on the dynamic model (3.16) - (3.18), three Proportional-Integral-Derivative (PID) controllers have been adapted to steer the three wheels to track the desired velocities independently. Figure 3.5 shows the control block diagram, where the set velocities are the desired wheel velocities coming from the inverse kinematic model (3.15) with the desired robot velocities. The PID controller is implemented by the TMC200 Triple Motorcontroller, which measures the motors' and wheels' data and controls the wheels velocities to approach the set values as closely as possible. The PID controller employs the following driving input torque to steer each wheel,

$$u_i = k_{p_i}e_{\dot{q}_i} + k_{i_i}\int_0^t e_{\dot{q}_i}d\tau + k_{d_i}, \dot{e}_{\dot{q}_i}$$

where the parameters k_{p_i}, k_{i_i} and k_{d_i} are proportional, integral and derivative gains with respect to the i-th wheel. They are parameters and can be modified according to the requirement of the control performance.

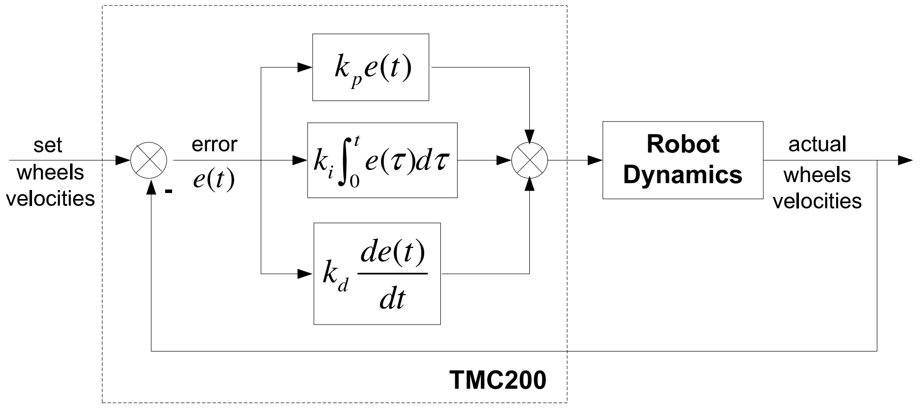


Figure 3.5: Diagram of the low-level control system

3.5 Actuator Influence

An actuator is typically a mechanical device used to convert energy into motions. In wheeled mobile robot systems, actuators normally are motors, which consume electric energy and control the wheel velocities. Actuator dynamics is an important component of robot dynamics. Especially in the cases of high-velocity movement and highly varying load, actuator dynamics has a high impact on the robot performance. Based on the kinematic model, designing robot motion control laws must assume that the robot actuator dynamics is fast enough to be ignored, which means the desired input values of the kinematic model can be achieved immediately. However, the actuator dynamics can not be completely omitted due to the materials, mechanisms and the limited power of motors. Therefore, it is necessary to think about the actuator influences on the kinematic model-based controller design. As two important aspects of the actuator, actuator dynamics and actuator saturation are considered in the motion control system of the Attempto soccer robot, which will be presented in the following subsections.

3.5.1 Actuator Dynamics

Actuator dynamics denotes the performance of the low-level control system. Although the actuators are normally controlled by the low-level controller with good performance, the actuator dynamics might degrade the whole system performance, especially in the case of high-velocity movement and highly varying load. It is necessary to analyze the actuator dynamics and take them into account in controller design.

In order to learn about the actuator dynamics, actuator dynamics are modeled based on the observed input-output data. The model is identified based on the observed input-output data so that a performance criterion is minimized. Because the full rank transformation matrix in the inverse low-level dynamics model (3.15) denotes the outputs $\dot{x}_R^m$, $\dot{y}_R^m$ and ω are decoupled, the models with respect to these three values are identified separately. The inputs of each model are required velocity values ($\dot{x}_{R_c}^m$, $\dot{y}_{R_c}^m$ and ω_c), respectively. The outputs are the corresponding measured values. As one commonly used parametric model, the autoregressive moving average with exogenous variable (ARMAX) model is chosen as the identified model. It has the following structure:

$$\begin{aligned} A(z)y(t) &= B(z)u(t-n_k) + C(z)e(t), \\ A(z) &= 1 + a_1 z^{-1} + ... + a_{n_a} z^{-n_a}, \\ B(z) &= b_1 z^{-1} + ... + b_{n_b} z^{-n_b+1}, \\ C(z) &= 1 + c_1 z^{-1} + ... + c_{n_c} z^{-n_c}. \end{aligned}$$

n_k denotes the delay from input $u(t)$ to output $y(t)$. $e(t)$ is white noise. z is the shift operator resulting in $z^{-1}u(t) = u(t-1)$. n_a, n_b and n_c are the orders of polynomials $A(z)$, $B(z)$ and $C(z)$, respectively. To choose the optimal parameters of this model, we use the prediction error method, which tries to find the optimal n_k and parameters of $A(z)$, $B(z)$ and $C(z)$ such that the prediction error E is minimized, namely

$$[A(z), B(z), C(z), n_k]_{opt} = argmin \sum_{t=1}^{N} E^2,$$

$$E = y_o(t) - A^{-1}(z)(B(z)u(t-n_k) + C(z)e(t)),$$

where $y_o(t)$ denotes the measured output data.

The system identification toolbox of Matlab has been used to identify the actuator dynamic models. After comparing prediction errors of ARMAX models with different values of n_a, n_b, n_c and n_k chosen from the positive integer set $[1, 5]$, the smallest prediction errors have been found using the following parameters,

- Model: $\dot{x}_{Rc}^m \rightarrow \dot{x}_R^m$

$$\begin{aligned} A(z) &= 1 - 0.642z^{-1} - 0.1997z^{-2}, \\ B(z) &= 0.1603z^{-1}, \\ C(z) &= 1 - 0.6444z^{-1}; \end{aligned}$$

- Model: $\dot{y}_{Rc}^m \rightarrow \dot{y}_R^m$

$$\begin{aligned} A(z) &= 1 - 1.296z^{-1} - 0.1821z^{-2} + 0.4937z^{-3}, \\ B(z) &= -0.01212z^{-1} + 0.1521z^{-2} - 0.1247z^{-3}, \\ C(z) &= 1 - 1.53z^{-1} + 0.6439z^{-2}; \end{aligned}$$

- Model: $\omega_c \rightarrow \omega$

$$\begin{aligned} A(z) &= 1 - 1.845z^{-1} + 0.8736z^{-2}, \\ B(z) &= 0.04546z^{-1} - 0.01723z^{-2}, \\ C(z) &= 1 - 1.794z^{-1} + 0.854z^{-2}. \end{aligned}$$

Figures 3.6, 3.7 and 3.8 show the comparisons between model outputs and measured outputs with respect to the actual inputs.

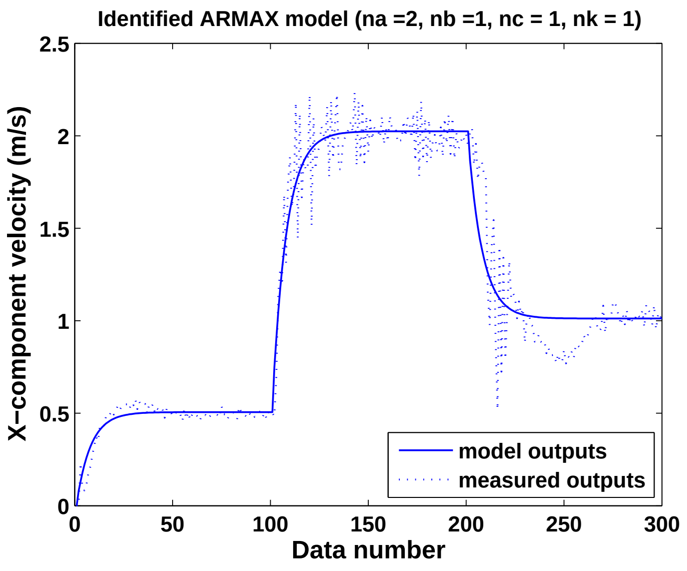


Figure 3.6: Identified model for $\dot{x}_R^m$.

To coincide with the robot's continuous model, the identified models are transformed from discrete ones into continuous ones using the 'zoh' (zero-order hold) method with the sampling time of 20 ms,

$$\dot{x}_R^m = \frac{8.7948(s + 58.47)}{(s + 73.66)(s + 6.897)} \dot{x}_{R_c}^m, \tag{3.19}$$

$$\dot{y}_R^m = \frac{2.4525(s + 48.83)(s + 6.185)}{(s + 28.45)(s^2 + 6.837s + 25.97)} \dot{y}_{R_c}^m, \tag{3.20}$$

$$\omega = \frac{1.667(s + 45.37)}{(s^2 + 6.759s + 76.11)} \omega_c. \tag{3.21}$$

With the identified actuators dynamics, the guidance system is designed in the following motion control system.

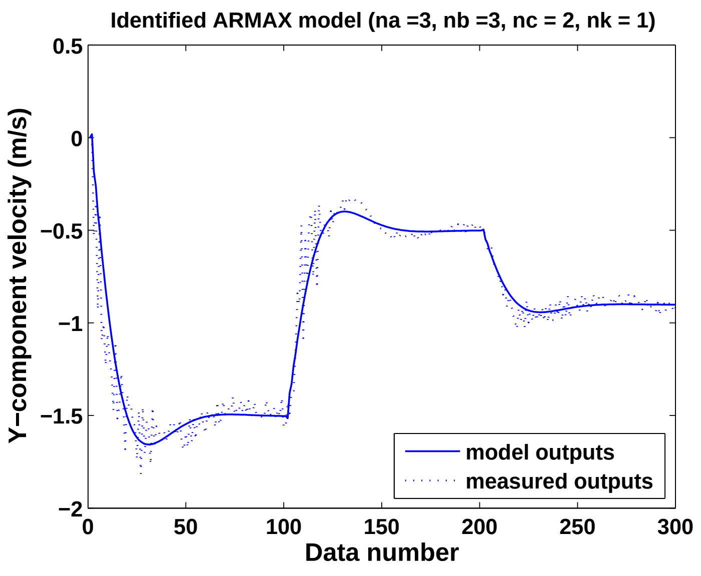


Figure 3.7: Identified model for $\dot{y}_R^m$.

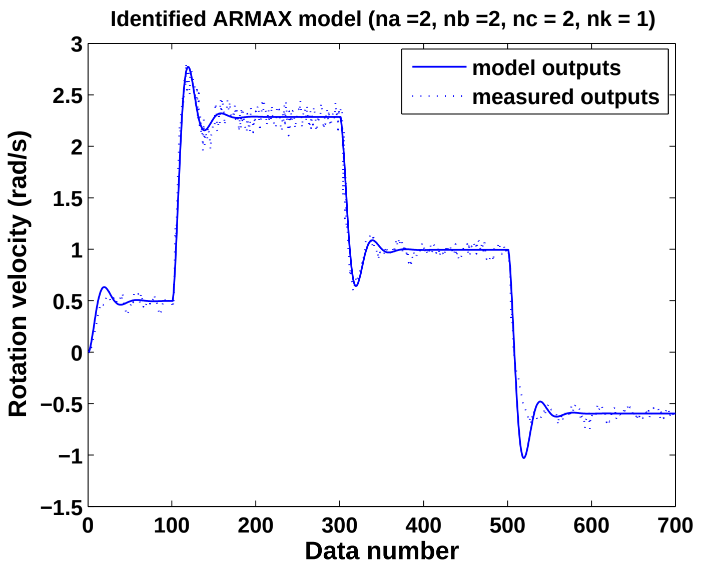


Figure 3.8: Identified model for ω.

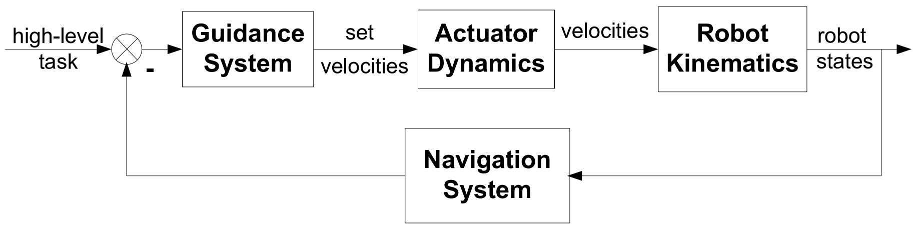


Figure 3.9: Closed-loop control system with actuator dynamics.

3.5.2 Actuator Saturation

Actuator saturation occurs if the commanded wheel velocities are larger than the maximum values, which are limited by the power of the wheels motors. At this moment, the actuator can only generate the highest admissible velocities, but not the desired ones, which can severely influence the overall control performance. Although more powerful actuators can be chosen to equip the robot platforms, the desired velocities coming from the high-level controller are also possible larger than the velocities which the actuators are able to generate. The decoupled maneuverability of translation and rotation can be broken for an omnidirectional robot losing its flexible mobility. Taking the actuator saturation into account, the robot control system is illustrated by the following block diagram,

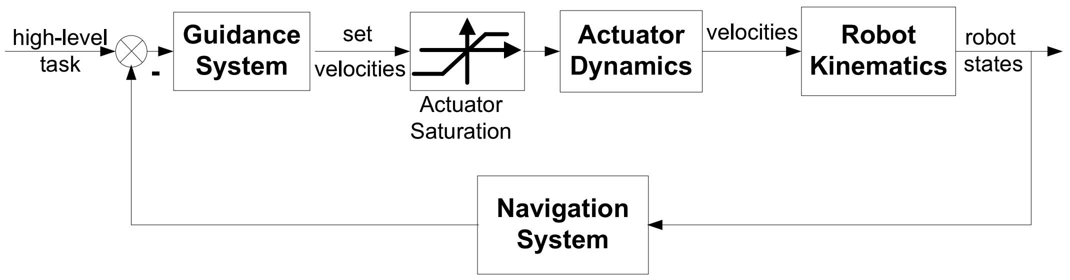


Figure 3.10: Closed-loop control system with actuator saturation and actuator dynamics.

When the robot translation velocities are projected into the world coordinate system, we get the control values of the linearized system Eq. (3.5) as

$$u_1 = v_d \cos\alpha, \tag{3.22}$$

$$u_2 = v_d \sin\alpha, \tag{3.23}$$

where α denotes the robot moving direction with respect to the world coordinate system. Substituting equations (3.22) and (3.23) into the following inverse robot kinematics,

$$\dot{\mathbf{q}} = \begin{bmatrix} \cos(\theta+\delta) & \sin(\theta+\delta) & L_w \\ -\cos(\theta-\delta) & -\sin(\theta-\delta) & L_w \\ \sin\theta & -\cos\theta & L_w \end{bmatrix} \dot{\mathbf{x}}, \tag{3.24}$$

the wheel velocities are computed as:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} v_d cos(\varphi-\delta) + L_w u_3 \\ -v_d cos(\varphi+\delta) + L_w u_3 \\ -v_d sin\varphi + L_w u_3 \end{bmatrix}, \tag{3.25}$$

where φ is the robot moving direction with respect to the robot coordinate system with $\varphi = \alpha - \theta$.

As the omnidirectional robot has the maximum velocity of each wheel $\dot{q}_m$, namely $|\dot{q}_i| \leq \dot{q}_m$, the actuator constraints are deduced as:

$$|v_d cos(\varphi-\delta) + L_w u_3| \leq \dot{q}_m, \tag{3.26}$$

$$|-v_d cos(\varphi+\delta) + L_w u_3| \leq \dot{q}_m, \tag{3.27}$$

$$|-v_d sin\varphi + L_w u_3| \leq \dot{q}_m, \tag{3.28}$$

These three inequalities dynamically constrain the values of control variables v_d, φ and u_3. Explicit analysis of the nonlinear constraints is not apparent, but the actuator saturation can also be handled by allocating them to the robot translation and rotation separately. In real applications, robot translation and rotation control are normally assigned with priorities based on pre-designed behaviors. For example, in the RoboCup scenarios, when the robot is required to block a rolling ball, the translation has to be as fast as possible. Therefore, the translation control has priority over rotation control. But for catching the ball and preparing a good shot, the rotation control is more important. With the different priorities of translation and rotation, the motion with higher priority should be ensured to be realized, and the other motion must be controlled under the corresponding saturation.

When translation control has higher priority, the desired translation velocity's magnitude v_d is assumed to be not bigger than $\dot{q}_m$ in order to achieve feasible control of the robot orientation. Based on the controlled values of u_1 and u_2 from equations (3.22) and (3.23), the lower and upper boundaries (l_{b_i} and u_{b_i}, $i = 1, 2, 3$) of u_3 with respect to each wheel can be calculated as follows,

$$\begin{gathered} l_{b_1} = -\dot{q}_m - v_d cos(\varphi-\delta) \leq L_w u_3 \leq \dot{q}_m - v_d cos(\varphi-\delta) = u_{b_1}, \\ l_{b_2} = -\dot{q}_m + v_d cos(\varphi+\delta) \leq L_w u_3 \leq \dot{q}_m + v_d cos(\varphi+\delta) = u_{b_2}, \\ l_{b_3} = -\dot{q}_m + v_d sin(\varphi) \leq L_w u_3 \leq \dot{q}_m + v_d sin(\varphi) = u_{b_3}. \end{gathered}$$

Then the dynamic boundary values of u_3 are computed as

$$l_b = \frac{max(l_{b_1}, l_{b_2}, l_{b_3})}{L_w},$$
$$u_b = \frac{min(u_{b_1}, u_{b_2}, u_{b_3})}{L_w},$$

where l_b and u_b are the lower and upper boundaries, and the resulting control system has the structure shown in Figure 3.11.

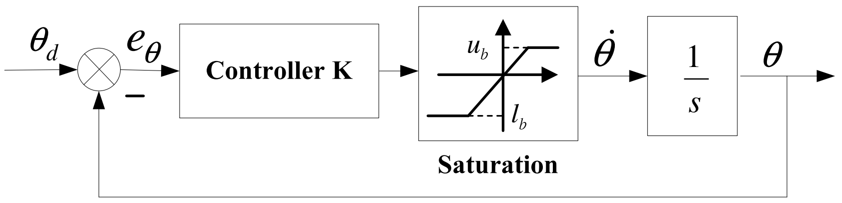


Figure 3.11: Closed-loop of robot orientation control.

On the other hand, when the rotation control is more important, the input values of φ and v_d should be bounded by the following constraints:

$$l_b \leq cos(\varphi - \delta) \leq u_b, \tag{3.29}$$

$$l_b \leq cos(\varphi + \delta) \leq u_b, \tag{3.30}$$

$$l_b \leq sin\varphi \leq u_b, \tag{3.31}$$

with the saturation limits

$$l_b = -\frac{\dot{q}_m - L_w \mid \omega|}{v_d},$$
$$u_b = \frac{\dot{q}_m - L_w \mid \omega|}{v_d}.$$

Figure 3.12 shows the curves of functions $\cos(\varphi + \delta)$, $\cos(\varphi - \delta)$ and $\sin\varphi$ from left to right, with φ valued from $-\pi$ to π. Two lines denote the boundaries with the absolute value equal to $\cos\delta$. It can be concluded that feasible values of φ exist only when the absolute value of l_b (or u_b) is not less than $\cos\delta$, namely,

$$|l_b| = \left|\frac{\dot{q}_m - L_w \mid \omega|}{v_d}\right| \leq \cos\delta = \frac{\sqrt{3}}{2}. \tag{3.32}$$

Therefore, when ω is determined to control the robot rotation, v_d should satisfy constant (3.32) such as to get the feasible boundary values of φ according to the constraints (3.29) - (3.31).

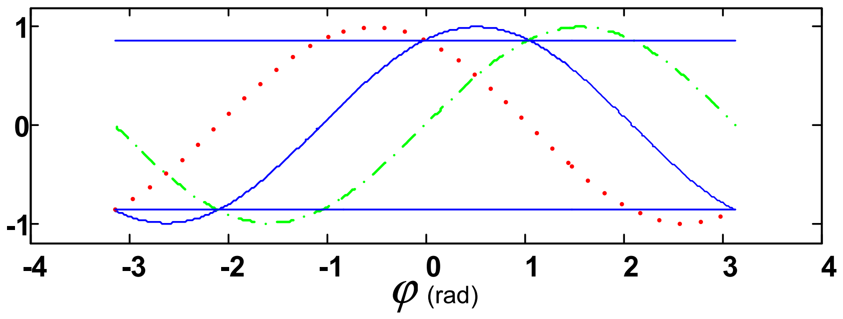


Figure 3.12: Solid, dotted and dashdotted curves illustrate the constraints from the left wheel, the right wheel and the back wheel, respectively. Two lines show the upper and lower boundaries with the absolute value $\frac{\sqrt{3}}{2}$.

3.6 Summary

This chapter presented the motion control system of the Attempto soccer robot. Considering the complex structure and dynamically variable mechanic and physical parameters in the robot dynamic model, a control system combining a kinematic and a dynamic model has been designed for our robot. The control system adopts the well-known Navigation-Guidance-Control structure, where the kinematic model-based high-level control focuses on accomplishing robot behaviors to fulfill high-level tasks, and the low-level control is concerned with the robot dynamic model, which takes the outputs of the high-level control as objectives and keeps the robot with the desired movement as well as possible.

Compared with nonholonomic robots, omnidirectional robots have higher maneuverability because of their full motion DOF on the plane. This advantage makes omnidirectional robot attractive in wheeled mobile robots applications, for example, in the Middle Size League of RoboCup. Especially the completely decoupled translation and rotation of omnidirectional robots enable the separate control of these two motions according to the requirement of different control tasks. However, this decoupled control strategy must assume that the robot actuators are powerful enough for the desired control values and the robot actuator dynamics are fast enough to be ignored. For a real mobile robot, the motors can only have limited power and the robot actuator dynamics are impossible to be omitted. Therefore, considering the influences of the robot actuator dynamics in the robot kinematics-based control is necessary and very important.

In sections 3.3 and 3.4, the kinematic and dynamic models of the Attempto soccer robot and the corresponding control laws have been described in detail. In order to guarantee closed-loop stability and good control performance, the actuator influence has been analyzed in Section 3.5. The identified actuator dynamics and the dynamically calculated actuator saturation are added into the control system, which provides the basis of designing the robot motion control laws.

Chapter 4

Robot Motion Control

Robot motion control refers to controlling the position and/or velocity of mobile robots by servo mechanisms. The tasks of robot motion control addressed in the literature can be roughly classified in the following three groups:

- Point stabilization: a robot is required to move from an initial point to a goal point with desired posture, no matter how the robot moves between these two points;
- Trajectory tracking: a robot is required to track a time parameterized reference trajectory;
- Path following: a robot is required to converge to a geometric reference path.

Point stabilization

The main challenge of point stabilization lies in stabilizing a nonholonomic robot at the goal point with the desired orientation because Brockett has shown in [20] that there is no smooth or even continuous state feedback control law able to solve the point stabilization problem for such kind of robot. The underlying reason is nonholonomic robots can not satisfy the necessary condition of the point stabilization problem: smooth stabilizability of a driftless regular system requires that the number of inputs is equal to the number of states. As a consequence, either discontinuous or time-varying control laws have been proposed to stabilize the robot at the desired posture, for example, smooth time-varying control laws taking the time variable into account, piecewise continuous time-varying control laws keeping continuous except at the equilibrium [31, 57], hybrid continuous or discontinuous feedback control laws composed of time-varying feedback controllers and either continuous or discontinuous controllers [5, 6, 10, 33]. With respect to omnidirectional robots, their full DOF on the plane determines the number of inputs equal to or larger than the number of states and makes the point stabilization problem easier to be solved.

Trajectory tracking

Trajectory tracking is aimed to control a robot to track a pre-designed feasible trajectory, which specifies the time evolutions of the position, orientation and translation and rotation velocities of the robot. As a necessary condition for the trajectory tracking problem, a feasible trajectory is the time based solution of the robot kinematic model with respect to a set of feasible input values. In trajectory tracking, the pose and velocity at each specified time instant in the future which the robot should track are exactly determined. To facilitate the controller design, the tracking error kinematics is normally developed as the first task based on the robot kinematics. Then, the trajectory tracking problem becomes to find suitable control laws which asymptotically stabilize the tracking error $(x_r(t) - x(t), y_r(t) - y(t), \theta_r(t) - \theta(t))$ at zero, where $(x(t), y(t), \theta(t))$ denotes the robot state and $(x_r(t), y_r(t), \theta_r(t))$ presents the state of the reference trajectory. By linearizing the tracking error kinematic model at the equilibrium point, the linearized system is controllable as long as the reference trajectory is non-stationary. This implies that linear control techniques can be used to achieve the local stability of the trajectory tracking problem, for example, Lyapunov stable time-varying feedback control laws [145, 144] and chained form based feedback control laws [111, 115]. Besides researching the preliminary local stability guaranteed controllers, many control laws have achieved global trajectory stabilization, such as the nonlinear feedback controllers based on the original nonlinear error kinematics [111] or the dynamic feedback linearized system [30, 124]. Based on the basic ideas of above trajectory tracking controllers, many variations and improvements have been proposed in the dedicated robotics literature. Some of them focus on increasing robustness due to the robot model uncertainties and environmental perturbations [127, 136], some try to keep control stability and good control performance with consideration of saturation constraints of the robot input values [73, 77].

Path following

The path following problem is the other important robot motion control problem, where a robot is required to move to a geometric reference path as close as possible. Unlike trajectory tracking, the robot moving speed is decoupled from the robot convergence movement in the path following problem. This means the robot can follow the reference path with a freely defined velocity magnitude according to the application requirements. Therefore, trajectory tracking can be regarded as a special case of the path following problem because the desired velocity history is directly specified with time in the trajectory tracking problem. A fundamental difference between the path following problem and the trajectory tracking problem can be demonstrated in non-minimum phase systems [7], where the $\mathcal{L}_2$-norm[1] of the tracking error can not be decreased arbitrarily small, but the flexibility of speed assignment allows the path following problem to have arbitrarily small $\mathcal{L}_2$-norm of the following error. Moreover, transforming or combining trajectory tracking and path following have been studied to yield better trajectory tracking performance [42, 125]. By steering the robot speed to track a desired speed profile, path following has been involved in many complex tasks, for example, multi robots formation control [53, 78], obstacle avoidance control [93], etc..

In Chapter 7, path following has been formulated in the soccer robot dribbling task. The most challenging task of my research is to control an omnidirectional robot to dribble a rolling ball. During the dribbling process, the ball can only be pushed but not be pulled. To decrease the ball's speed, the robot has to move ahead and hinder the ball's movement. When the ball's speed has to be increased, the robot needs to stay behind the ball and push it. This means that varying the ball's speed will unsmooth the robot movement and increase the possibility of losing the ball. Therefore, a desired constant high speed is chosen for the ball in the path following task, which not only facilitates the robot motion control, but also ensures the ball's fast moving in the RoboCup matches.

According to the omnidirectional mobility, the motion control of our soccer robot can be achieved by separately controlling robot translation and rotation. Sections 4.2 and 4.3 introduce the control methods of robot translation and rotation, respectively. The path following problem is taken as the task of robot translation control. The robot rotation control concerns the actuator dynamics and actuator saturation, such as to keep a stability guaranteed robot motion control [99].

4.1 Nonlinear Systems Theory

Before the introduction of motion control methods, this section recalls some necessary theories about stability of the equilibrium points of nonlinear systems. The reported theorems and definitions are borrowed from [84].

[1]$\mathcal{L}_2$-space is the set of square integrable functions.

4.1.1 Lyapunov Stability

Suppose the autonomous[2] system

$$\dot{\mathbf{x}} = f(\mathbf{x}) \tag{4.1}$$

has an equilibrium point, which is assumed to be at the origin of $\mathbb{R}^n$, i.e. $f(\mathbf{0}) = 0$. There is no loss of generality in doing so because any equilibrium point can be shifted to the origin via a change of variables.

Definition 1 *The equilibrium point $\mathbf{x} = 0$ of (4.1) is*

- *stable, if for each $\epsilon > 0$, there is $\delta = \delta(\epsilon) > 0$ such that*

$$\|\mathbf{x}(0)\| < \delta \Rightarrow \|\mathbf{x}(t)\| < \epsilon, \;\; \forall\, t \geq 0$$

- *unstable, if it is not stable*
- *asymptotically stable, if it is stable and δ can be chosen such that*

$$\|\mathbf{x}(0)\| < \delta \Rightarrow \lim_{t\to\infty} \mathbf{x}(t) = 0$$

In order to demonstrate that the origin is a stable equilibrium point, for each selected value of ϵ one must produce a value of δ, possibly dependent on ϵ, such that a trajectory starting in a δ neighborhood of the origin will never leave the ϵ neighborhood. It is possible to determine stability by examining the derivatives of some particular functions without having to know explicitly the solution of (4.1).

4.1.2 Lyapunov's Stability Theorem

Theorem 1 *Let $\mathbf{x} = 0$ be an equilibrium point for (4.1) and $D \subset \mathbb{R}^n$ be a domain containing $\mathbf{x} = 0$. Let $V : D \to \mathbb{R}$ be a continuously differentiable function such that*

$$V(\mathbf{0}) = 0 \; and \; V(\mathbf{x}) > 0 \; in \; D - \{0\} \tag{4.2}$$

$$\dot{V}(\mathbf{x}) \leq 0 \; in \; D \tag{4.3}$$

Then, $\mathbf{x} = 0$ is stable. Moreover, if

$$\dot{V}(\mathbf{x}) < 0 \; in \; D - \{0\}$$

then $\mathbf{x} = 0$ is asymptotically stable.

[2]A system in which the function f does not depend explicitly on time.

A function $V(\mathbf{x})$ satisfying condition (4.2) is said to be *positive definite*. If it satisfies the weaker condition $V(\mathbf{x}) \geq 0$ for $\mathbf{x} \neq 0$, it is said to be *positive semidefinite*. A function $V(\mathbf{x})$ is said to be *negative definite* or *negative semidefinite* if $-V(\mathbf{x})$ is *positive definite* or *positive semidefinite*, respectively. A continuously differentiable function $V(\mathbf{x})$ satisfying (4.2) and (4.3) is called a Lyapunov function, after the Russian mathematician who laid the base of this theory.

4.1.3 Barbalat's Lemma

Lemma 1 *For a time-varying system, if a Lyapunov function $V(\mathbf{x}, t)$ satisfies the following conditions:*

1. *$V(\mathbf{x}, t)$ is lower bounded*
2. *$\dot{V}(\mathbf{x}, t)$ is negative semidefinite*
3. *$\dot{V}(\mathbf{x}, t)$ is uniformly continuous in time (satisfied if $\ddot{V}$ is finite)*

then $\dot{V}(\mathbf{x}, t) \to 0$ as $t \to 0$.

Barbalat's Lemma will be used to proof stability of the quasi-infinite horizon nonlinear model predictive control in Section 5.3.

4.2 Translation Control

4.2.1 Path Following Problem

As one important control problem of mobile robots, path following aims to find a feedback control law such that the robot's center of mass converges asymptotically to a geometric reference path. Unlike trajectory tracking, which can be formulated to decrease the error between the real robot states and the desired robot states parameterized by time, the aim of the path following problem is to decrease the *distance* between the robot and the reference path. The different definitions of the *distance* results in different formulations of the path following problem.

A kinematic level formulation of the path following problem was first presented in [111, 143], where the *distance* value is described in a path frame which is moving along the given path. When the distance is equal to the radius of curvature of the given path, the singularity problem will occur in this formulation. By linearizing the formulation, the proposed controller can only keep the robot locally converging to the given path. In order to get a global convergence to the given path and to avoid the singularity problem, a more general formulation of the path following problem was utilized in many control laws ([34, 152, 39, 40, 105]), where the concept of *Virtual Vehicle* is introduced to refer to the robot's desired position on the given path and a new variable is introduced to control the movement of the *Virtual Vehicle*. The main difference between these controllers focuses on the ways of controlling the *Virtual Vehicle*'s movement, which results in different control performance of the robot following the reference path. Normally the velocity of the *Virtual Vehicle* is controlled by a nonlinear controller based on the distance errors and the characteristics of the reference path. With respect to the controlled *Virtual Vehicle*'s velocity, the corresponding robot motion controllers can provide the global convergence to the reference path.

Besides the variable controlling the *Virtual Vehicle*'s movement, the direction of robot translation is another important control variable in the formulations of the path following problem. For nonholonomic robots, the robot orientation angle denotes the direction of robot translation, which can only be controlled by the robot rotation velocity around the axis perpendicular to the plane. In the case of omnidirectional robots, robots are able to translate and rotate simultaneously, which means the robot's translation direction can be regulated regardless of the robot rotation velocity. In the next subsections, orthogonal projection-based and *Virtual Vehicle*-based formulations of the path following problem have been described. The path following control methods with respect to nonholonomic and omnidirectional robots are presented in details.

4.2.2 Orthogonal Projection-based Control

Figure 4.1 illustrates the orthogonal projection-based formulation of the path following problem. P denotes the reference path. Point R is the robot's center of mass, and point Q is the orthogonal projection of R on the path P. A path coordinate system $\{P\}$ is composed of axes $\mathbf{X}_t$ and $\mathbf{X}_n$, which are the tangent and normal unit vectors at Q, respectively. x_n is the signed distance between the robot and the path P. s is the signed curvilinear abscissa denoting the position of Q. θ_p is the angle between axis $\mathbf{X}_t$ and axis $\mathbf{x}_w$. v_R is the robot translation velocity, whose direction with respect to the world coordinate system is α.

Based on the previous definitions, the path following problem can be parameterized as

$$\dot{s} = v_R \cos\alpha_e \frac{1}{1 - c x_n}, \tag{4.4}$$

$$\dot{x}_n = v_R \sin\alpha_e, \tag{4.5}$$

$$\dot{\alpha}_e = \omega_v - v_R \cos\alpha_e \frac{c}{1 - c x_n} \tag{4.6}$$

where c is the path curvature at point Q, α_e is the angular error with $\alpha_e = \alpha - \theta_p$, ω_v is equal to $\dot{\alpha}$. Therefore, the aim of path following is to find suitable control

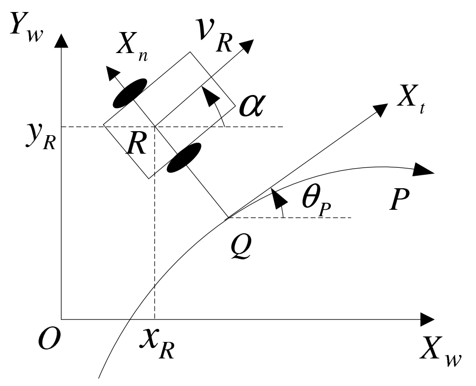


Figure 4.1: Illustration of the orthogonal projection-based formulation of the path following problem

values of v_R and ω_v such that the deviation x_n and angular error α_e tend to zero.

According to the kinematic equations (4.4) - (4.6), two widely used linear and nonlinear feedback control laws for nonholonomic robots were introduced in [32]. v_R is set to be the desired translation velocity v_d, which is assumed to be different from zero. Then the task of path following is to find a suitable ω_v such that the deviation distance and angular error tend to zero.

In the neighborhood of the origin $(x_n = 0, \alpha_e = 0)$, linearizing (4.5) and introducing an auxiliary control variable u result in

$$\dot{x}_n = v_d \alpha_e, \tag{4.7}$$

$$\dot{\alpha_e} = u, \tag{4.8}$$

with

$$u = \omega_v - v_R \cos \alpha_e \frac{c}{1 - c x_n}. \tag{4.9}$$

When v_d is constant and different from zero, the system described by equations (4.7) and (4.8) is controllable and can be stabilized with the following linear state feedback controller

$$u = -k_1 v_d x_n - k_2 \left| v_d \right| \alpha_e, \tag{4.10}$$

where $k_1 > 0$ and $k_2 > 0$. As a consequence, the closed-loop equation of x_n becomes

$$\ddot{x}_n + k_2 \left| v_d \right| \dot{x}_n + k_1 v_d^2 x_n = 0.$$

This is a typical second-order system, whose performance requirement can be directly used to choose the controller parameters k_1 and k_2,

$$k_1 = \frac{2\xi a}{v_d^2},$$
$$k_2 = \frac{a^2}{|v_d|},$$

where a and ξ are the undamping natural frequency and the damping ratio of the second order system, respectively.

Without the linearisation, a nonlinear feedback control law with little difference from (4.10) can also asymptotically stabilize the distance deviation and the angular error to zero [32]. It is designed as

$$u = -k_1 v_d x_n \frac{\sin \alpha_e}{\alpha_e} - k_2(v_d)\alpha_e,$$

where k_1 is a positive constant and $k(v_d)$ is a positive continuous function when $v_d \neq 0$.

The disadvantage of the formulation with equations (4.4) - (4.6) is the singularity problem at $cx_n = 1$ [143]. That limits the controller to situations, where x_n is smaller than $\frac{1}{c_{max}}$. c_{max} denotes the maximum curvature of the path.

Unlike nonholonomic robots, omnidirectional robots have the capability of translating to any direction regardless of their orientation. This means the translation direction α of an omnidirectional robot can be controlled directly to achieve any desired value. Therefore, α can replace ω_v to be controlled for solving the path following problem of omnidirectional robots.

Mojaev *et al.* [114] presented a simple control law based on the deviation x_n, where robot's center of mass R is controlled to converge to the axis x_t along an exponential curve expressed as

$$x_n = x_{n_0} exp(-kx_t).$$

x_{n_0} is the initial deviation and the positive constant k determines the convergence speed of the deviation. Figure 4.2 shows the basic control idea.

Differentiating (4.2.2) with respect to x_t, we get the tangent direction of the exponential curve as

$$\alpha_e = \arctan(\frac{dx_n}{dx_t}) = \arctan(-kx_n). \tag{4.11}$$

Therefore, for a non-zero constant desired velocity v_d, the translation velocity of the robot in the path coordinate system $\{P\}$ results in

$$\dot{x}_n = v_d \sin \alpha_e, \tag{4.12}$$
$$\dot{x}_t = v_d \cos \alpha_e. \tag{4.13}$$

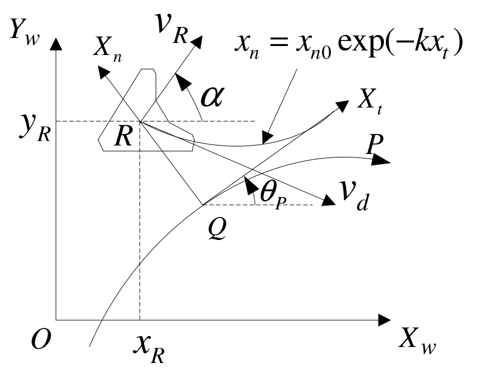


Figure 4.2: Illustration of path following control with an exponential convergence trajectory.

To prove the stability of the closed-loop system, a Lyapunov candidate function

$$V = \frac{1}{2}k_d x_n^2 + \frac{1}{2}k_\theta \alpha_e^2$$

can be considered, where k_d and k_θ are positive constants. The time derivation of V results in

$$\dot{V} = k_d x_n \dot{x}_n + k_\theta \alpha_e \dot{\alpha}_e. \tag{4.14}$$

Substituting the time derivative of α_e from controller (4.11) into (4.14), we get

$$\dot{V} = k_d x_n \dot{x}_n + kk_\theta \arctan(-kx_n)\frac{-\dot{x}_n}{1+(kx_n)^2} < 0. \tag{4.15}$$

Because $x_n\dot{x}_n = x_n v_d \sin(\arctan(-kx_n)) < 0$ and $\dot{x}_n \arctan(kx_n) < 0$, this solution of $\dot{V}$ guarantees the global stability of the equilibrium at $(x_n = 0, \alpha_e = 0)$, which means this control law solves the path following problem.

Transforming the robot velocity into the world coordinate system $\{W\}$, we get the control values of the Attempto soccer robot as

$$\begin{aligned} u_1 &= v_d \cos\alpha, \\ u_2 &= v_d \sin\alpha, \end{aligned}$$

with $\alpha = \alpha_e + \theta_P$.

The input of controller (4.11) is the distance between point R and the given path, which normally can be directly obtained by the sensors on the robot. Moreover, the convergence speed is controlled only by parameter k, which can be chosen according to the performance requirement.

4.2.3 *Virtual Vehicle*-based Control

The *Virtual Vehicle*-based formulation of the path following problem presents the kinematics of the *distance* value between the robot and the *Virtual Vehicle*. The corresponding controller aims to decrease the *distance* to zero.

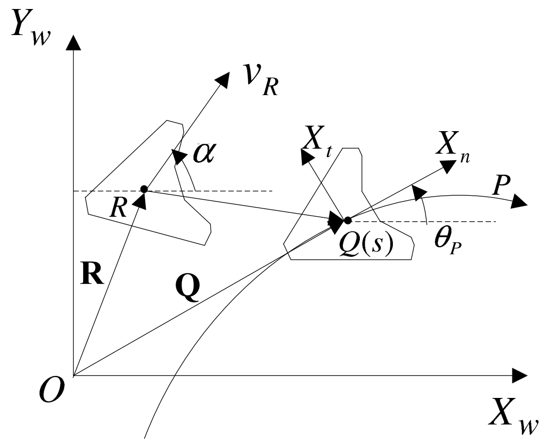


Figure 4.3: Illustration of *Virtual Vehicle*-based formulation of the path following problem.

As shown in Figure 4.3, the path coordinate system $\{P\}$ is located at point Q, the center of mass of the *Virtual Vehicle*, which moves along the reference path P and is determined by the curvilinear abscissa s. Let vectors $\mathbf{R}$ and $\mathbf{Q}$ describe the positions of R and Q in the world coordinate system, ${}^wR_m$ and ${}^wR_p$ present the transformation matrices from $\{M\}$ to $\{W\}$ and from $\{P\}$ to $\{W\}$, respectively, the following relationship holds:

$$\mathbf{R} = \mathbf{Q} + {}^wR_p(\begin{matrix} x_e \\ y_e \end{matrix}), \tag{4.16}$$

where x_e and y_e denote the robot positions with respect to the path coordinate system $\{P\}$. Computing the time derivatives of (4.16)

$${}^wR_m \begin{pmatrix} u \\ v \end{pmatrix} = {}^wR_p \begin{pmatrix} \dot{s} \\ 0 \end{pmatrix} + {}^w\dot{R}_p \begin{pmatrix} x_e \\ ye \end{pmatrix} + {}^wR_p \begin{pmatrix} \dot{x}_e \\ \dot{y}e \end{pmatrix},$$

and after some simplifications, the error kinematic model of the path following problem is given by,

$$\dot{\mathbf{x}}_e = \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\alpha}_e \end{bmatrix} = \begin{bmatrix} (y_e c(s) - 1)\dot{s} + v_R \cos \alpha_e \\ -x_e c(s)\dot{s} + v_R \sin \alpha_e \\ \omega_v - c(s)\dot{s} \end{bmatrix}, \tag{4.17}$$

where the error vector $\mathbf{x}_e$ is with respect to the path coordinate system $\{P\}$, α_e presents the angular error between the robot moving direction α and the path tangent direction θ_p, the $c(s)$ denotes the path curvature at point Q and ω_v is the corresponding angular velocity of α.

It is noticed in model (4.17) that controlling v_R is decoupled from controlling $\dot{s}$ and ω_v, because the errors can stay on the equilibrium $(\mathbf{x}_e = 0)$ when $\dot{s}$ approaches v_R. Therefore, the path following problem is to find suitable values of $\dot{s}$ and ω_v to minimize errors x_e, y_e and α_e, while v_R is assigned with any desired velocity.

Model (4.17) is really independent of the robot platforms. The control values of error kinematics (4.17) are high-level control inputs of the robot platforms. For the omnidirectional robots, the value α_e can be steered directly. But for the nonholonomic robots, the angular movement can only be controlled by $\dot{\alpha}_e$.

The direct Lyapunov function method has been used to design a nonlinear controller. Defining the following Lyapunov candidate function

$$V = \frac{1}{2}k_x x_e^2 + \frac{1}{2}k_y y_e^2 + \frac{1}{2}k_\theta \alpha_e^2,$$

where parameters k_x,k_y and k_θ are positive constants, and substituting (4.17) into the time derivation of V,

$$\dot{V} = k_x x_e \dot{x}_e + k_y y_e \dot{y}_e + k_\theta \alpha_e \dot{\alpha}_e,$$

we obtain

$$\begin{aligned} \dot{V} = \; & k_x x_e((y_e c(s) - 1)\dot{s} + v_R \cos\alpha_e) \\ & + k_y y_e(-x_e c(s)\dot{s} + v_R \sin\alpha_e) + k_\theta \alpha_e \dot{\alpha}_e. \end{aligned} \tag{4.18}$$

When $k_x = k_y = k_p$ is selected, equation (4.18) results in

$$\dot{V} = -k_p x_e \dot{s} + k_p x_e v_R \cos\alpha_e + k_p y_e v_R \sin\alpha_e + k_\theta \alpha_e \dot{\alpha}_e. \tag{4.19}$$

To keep $\dot{V}$ being negative, the control variables of (4.17) can be defined as follows:

- for non-omnidirectional robots

$$\dot{\alpha}_e = \; -\alpha_e - \frac{k_p y_e v_R \sin\alpha_e}{k_\theta \alpha_e}, \tag{4.20}$$

$$\dot{s} = \; v_R \cos\alpha_e + k_2 x_e, \tag{4.21}$$

- for omnidirectional robots

$$\alpha_e = \; \arctan(-k_1 y_e), \tag{4.22}$$

$$\dot{s} = \frac{v_R \cos\alpha_e + k_2 x_e}{1 + \frac{c(s)\alpha_e k_1 k_\theta}{k_p(1+(k_1 y_e)^2)}}, \tag{4.23}$$

where, k_1 and k_2 are positive constants. For nonholonomic robots, controller (4.20) - (4.21) only gives the control value of $\dot{alpha_e}$ to steer the robot rotation velocity with $\omega = \omega_v = \dot{\alpha}_e + c(s)\dot{s}$. In order to get better following performance, a desired approach angle shaping transient maneuvers during the path approach phase has been added to direct the robot moving direction [111, 152]. In the case of omnidirectional robots, the moving direction α is directly controlled with $\alpha = \alpha_e + \theta_p$, where α_e is from controller addressed in Eq. (4.22) and θ_p is related to the controlled value $\dot{s}$. To control the robot translation with the desired velocity v_d , the input values of the robot control system (3.5) are given by,

$$\begin{aligned} u_1 &= v_d \cos\alpha, \\ u_2 &= v_d \sin\alpha. \end{aligned}$$

4.3 Rotation Control

Unlike the nonholonomic robot, the orientation of an omnidirectional robot can be different from the direction of the robot translation velocity by any angle φ. This relationship is denoted as $\alpha = \theta + \varphi$. That means the robot orientation can track any angle no matter how the robot translates. Based on the robot kinematic model (3.2), the rotation control is to find a suitable robot rotation velocity ω, which is u_3 in the linearized system (3.5), such that the robot orientation converges to the desired value, i.e.

$$\lim_{t\to\infty}(\theta_d(t) - \theta(t)) = 0, \tag{4.24}$$

where $\theta_d(t)$ is the desired orientation.

As a simple first-order differential system, many control laws can be utilized to control the robot orientation. In order to get fast transient response and decrease overshoot, we chose the Proportional-Derivative (PD) controller to fulfill the rotation control task, which is given by

$$\omega = k_1(e_\theta + k_2\dot{e}_\theta), \tag{4.25}$$

where e_θ is the orientation error defined as $e_\theta = \theta_d(t) - \theta(t)$, $\dot{e}_\theta$ is the corresponding derivative term, k_1 and k_2 are the proportional and derivative gains, respectively.

As described in Chapter 3, when the robot translation control is assigned with higher priority, the controlled value of ω has to satisfy the system constraints. Considering the saturation function

$$x_2 = \begin{cases} u_b & x_1 > u_b \\ x_1 & l_b \le x_1 \le u_b \\ l_b & x_1 < l_b \end{cases}, \tag{4.26}$$

and its gain characteristics illustrated in Figure 4.4, the closed-loop system of controlling the robot orientation is shown in Figure 4.5.

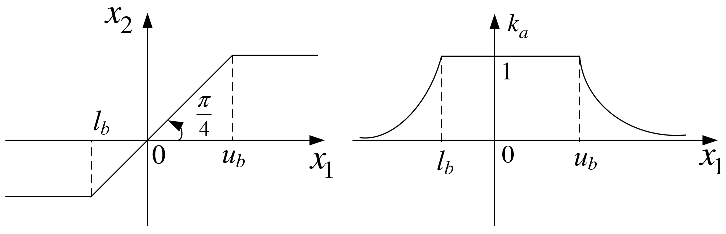


Figure 4.4: Saturation function and its gain characteristics.

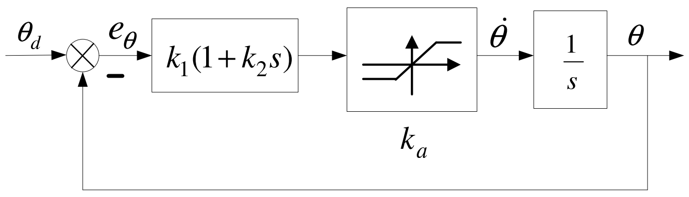


Figure 4.5: Closed-loop of the robot orientation control.

The saturation function works as a dynamic gain block k_a, which has the maximum value of one and converges to zero when the input saturates. By computing the transformation function of the closed-loop system, it can be obtained that there exist one pole at $\frac{-k_a k_1}{1+k_a k_1 k_2}$ and one zero at $-1/k_2$. Therefore, when k_2 and k_1 are positive, the stability of the closed-loop system can be guaranteed whenever k_a decreases.

The other important issue to be considered is the actuator dynamics, which has been identified and shown in subsection 3.5.1. As shown in Figure 4.6, the dynamic system (3.21) adds another two poles to the closed-loop system and the controller parameters are chosen again such as to guarantee the stability.

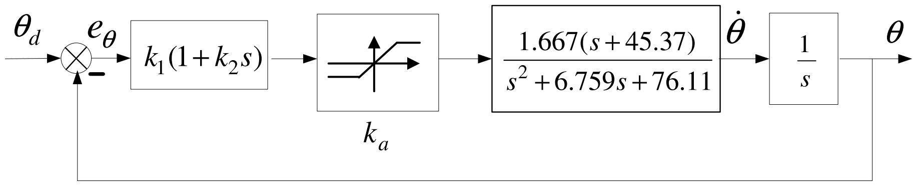


Figure 4.6: Closed-loop of the robot orientation control including actuator dynamics.

The locus technique is used here to set the positions of poles and zeros of the closed-loop system. Analyzing the root locus of the controlled system, the necessary conditions to guarantee the closed-loop stability can be found at $k_1 > 0$ and $k_2 > 0.0515$. Figure 4.7 shows the root locus of an open-loop system in the critical situation with $k_2 = 0.0515$, where all the poles of the closed-loop system locate in the left-half plane whatever positive value $k_a k_1$ is. Otherwise, when k_2 is less than 0.0515, the root locus may cross the imaginary axis, and the poles of the closed-loop system may move to the right-half plane when k_a goes to zero.

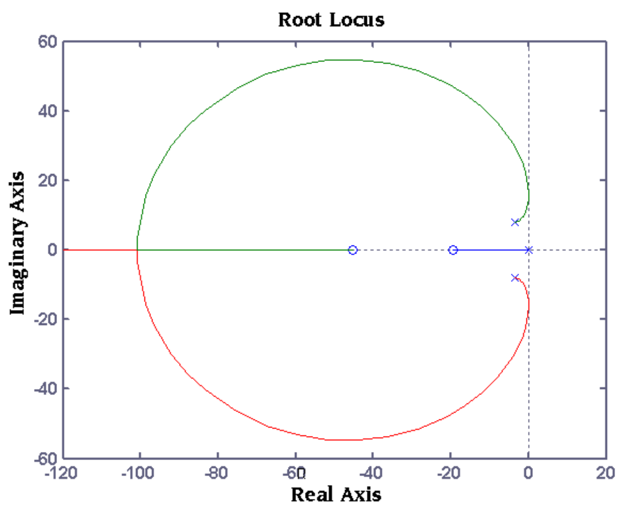


Figure 4.7: Root locus of robot orientation control.

4.4 Experimental Results

The control algorithms discussed above have been tested in our robot laboratory having a half-field of the RoboCup Middle Size league. An eight-shaped path is adopted as the reference path, whose geometrical symmetry and sharp changes in curvature make the test challenging. With a scale variable p, the chosen eight-shaped path is calculated as

$$x_r = 1.8\sin(2p), \tag{4.27}$$

$$y_r = 1.2\sin(p). \tag{4.28}$$

The robot was controlled to follow the eight-shaped path with a constant translation velocity $v_d = 1$ m/s. In the view of orientation control, two kinds of desired robot orientation are designed in the experiments. One is a constant angle of zero degree with respect to the world coordinate system, which is a typical test to show the decoupled translation and rotation of the omnidirectional robot. Another is to compute the desired robot orientation with

$$\theta_d = \theta_P + 0.9 c_P v_d^2, \tag{4.29}$$

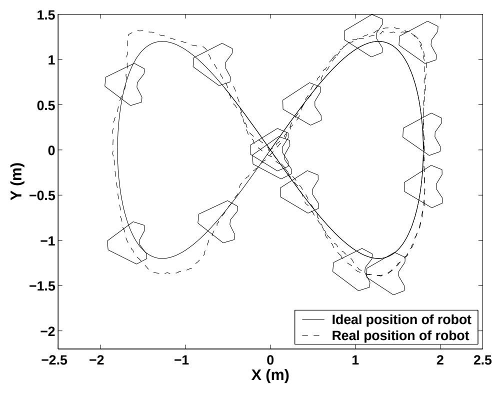
where c_P is the path curvature at point Q. These high values of the desired orientation are employed to check the control stability when the actuator saturation appears.

With respect to the two different formulations of the path following problem, two sets of experiments have been done with the real robot. Figures 4.8 and 4.9 show the results of the orthogonal projection-based control, where the parameters in the control algorithm were chosen as $k = 2.5$, $k_1 = 4.15$ and $k_2 = 3$. Figures 4.10 and 4.11show the results of the *Virtual Vehicle*-based control taking the following parameters $k_1 = 2.5,\ k_2 = 2.5,\ k_p = 2.0$ and $k_\theta = 1.0$.

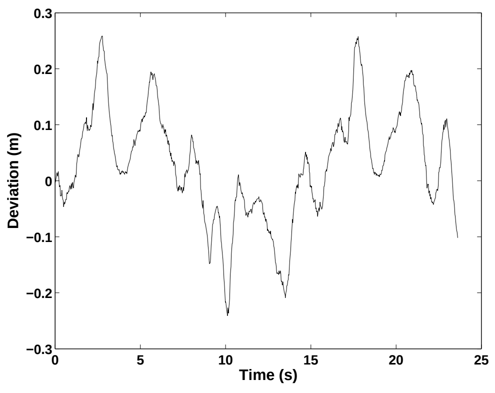
Figures 4.8 and 4.9 illustrate the results with respect to the different desired robot orientations. Figures 4.8(a), 4.8(b) and 4.8(c) show us that the proposed control method steers the robot center R converging to the given path and the robot orientation tracking the constant value with good performance. The maximum distance error is less than 0.26 m, and most angular errors are less than 0.2 rad. Figure 4.8(d) shows the measured wheel velocities are less than the maximum value 1.9 m/s, which means that the actuator saturation did not appear. When the desired orientation has more requirement for controlling the robot orientation, the actuator saturation appeared in the second experiment. In Figure 4.9(d), the measured wheel velocities reach the maximum value, when the robot is around the sharp turning segments of the reference path. At these moments, the orientation errors become much bigger, but still converging to zero, as shown in Figure 4.9(c). The distance errors illustrated in Figure 4.9(b) are also decreasing to zero, although the robot moves away from the reference path after the sharp turning segments because of slide caused by the large translation and rotation accelerations, which can be seen in Figure 4.9(a).

The similar following errors can be seen in figures 4.10(b) and 4.11(b)), where the *Virtual Vehicle*-based control method was employed. Figures 4.10(c) and 4.11(c) imply the function of the control value $\dot{s}$, which dynamically changes the positions of the *Virtual Vehicle* according to the following errors. In Figure 4.10(c), $\dot{s}$ slows down at the sharp turning segments of the reference path because of the large distance errors outside the reference path shown in Figure 4.10(a). But in Figure 4.11(c), $\dot{s}$ increases around the sharp turning segments because of the large orientation errors. This changed $\dot{s}$ also made the robot have a more smooth following performance comparing to that in the case of the orthogonal projection-based control, which can be seen in figures 4.9(a) and 4.11(a).

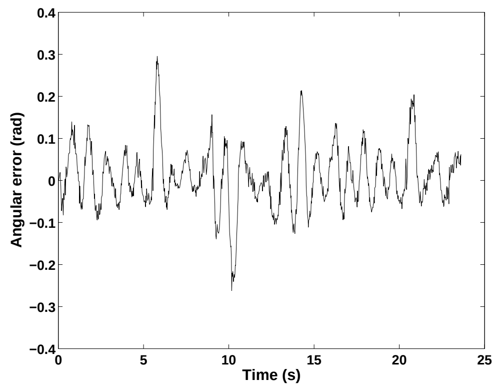
All the experimental results show that the path following control methods can guarantee closed-loop stability of the path following problem and the robot orientation can track the desired ones, even though the wheel velocities reach saturation when the robot makes a sharp turn.



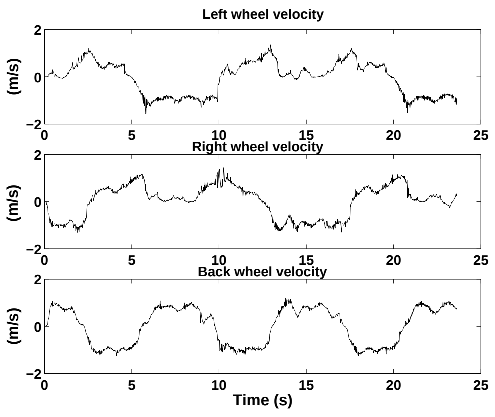
(a) Reference and real paths of the robot



(b) Distance error.

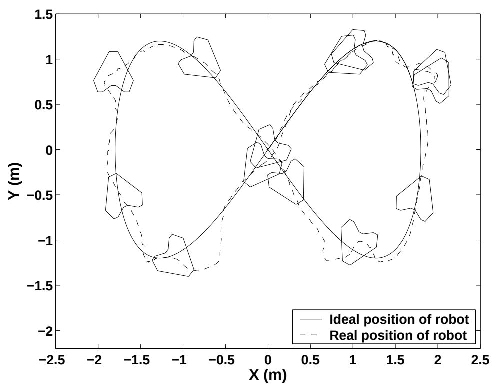


(c) Orientation error.

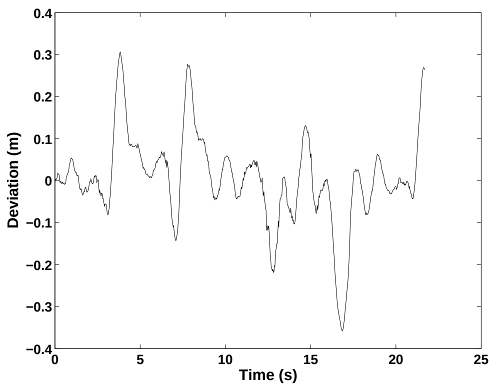


(d) Real wheel velocities.

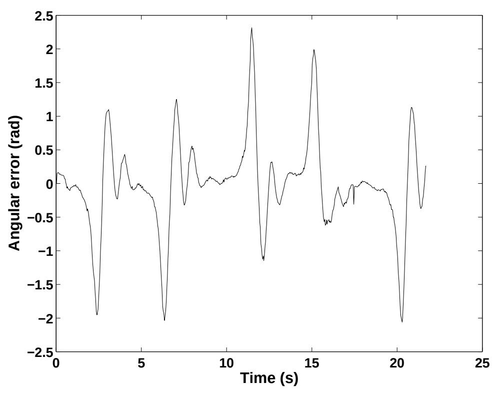
Figure 4.8: Orthogonal projection-based path following control with constant desired robot orientation



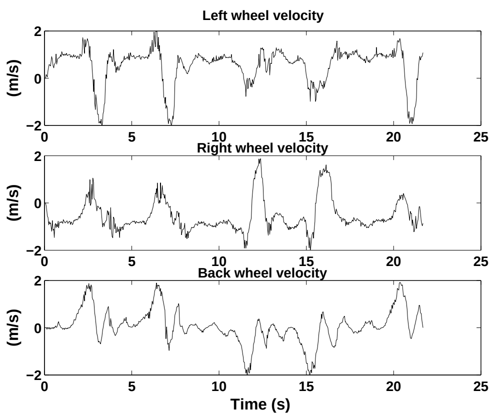
(a) Reference and real paths of the robot



(b) Distance error.

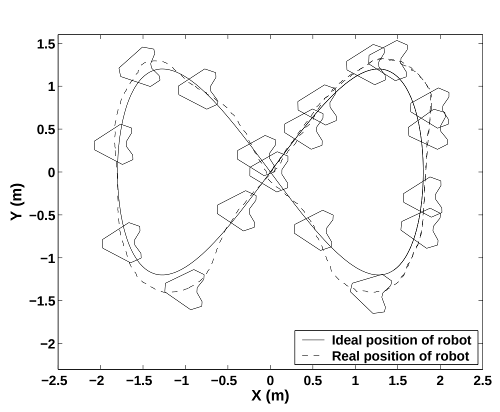


(c) Orientation error.

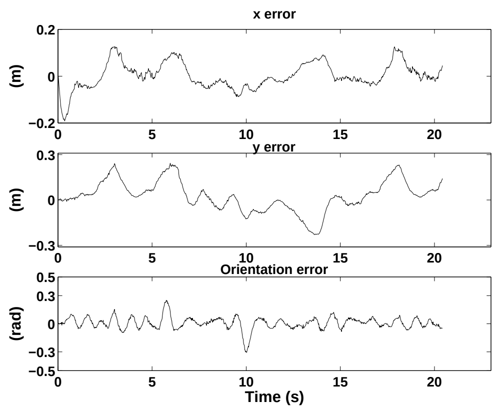


(d) Real wheel velocities.

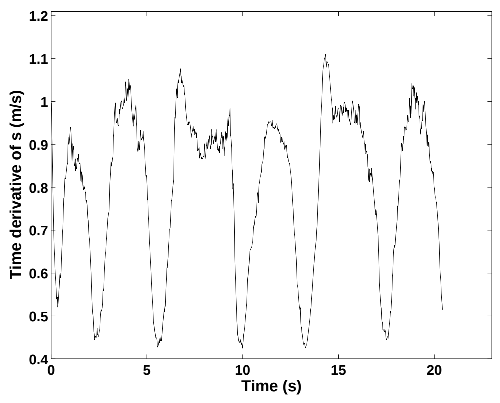
Figure 4.9: Orthogonal projection-based path following control with dynamic desired robot orientation



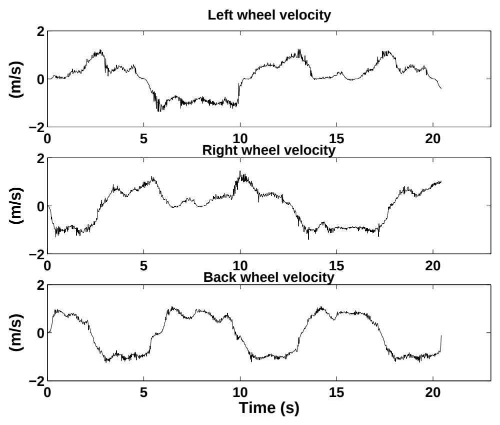
(a) Reference and real paths of the robot



(b) Following error.

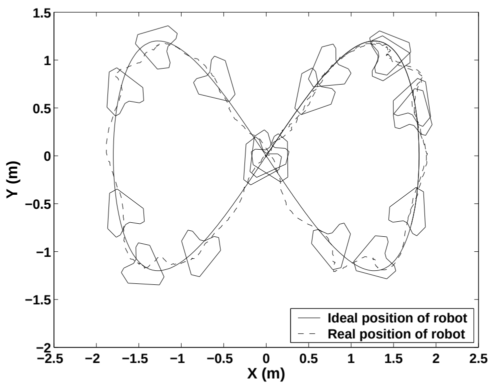


(c) Controlled values of $\dot{s}$.

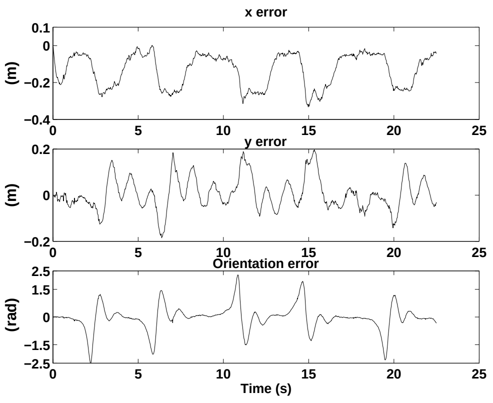


(d) Real wheel velocities.

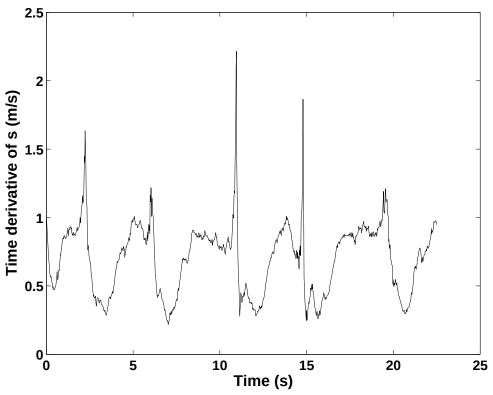
Figure 4.10: *Virtual Vehicle*-based path following control with constant desired robot orientation



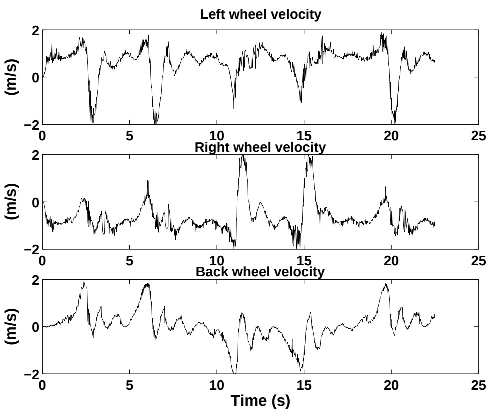
(a) Reference and real paths of the robot



(b) Following error.



(c) Controlled values of $\dot{s}$.



(d) Real wheel velocities.

Figure 4.11: *Virtual Vehicle*-based path following control with constant desired robot orientation

4.5 Summary

Based on the robot control system introduced in Chapter 3, this chapter focuses on designing the robot motion control laws. In the three basic robot motion control problems, path following has been chosen as the main control task of this research. The reasons are: (1) The path following problem gives a more general formulation. Trajectory tracking and point stabilization can be taken as special cases of the path following problem. The specialties are that the time parameterized desired velocities are designed in the trajectory tracking problem, and the point stabilization problem only stabilizes the robot at one desired position with desired posture. (2) The main research challenge in this thesis is the dribbling control of the Attempto soccer robot, which requires that the robot can always keep the ball moving in a dynamic environment without any collisions with other objects. The adopted dribbling strategy, which will be detailed in Chapter 7, involves the path following formulation, where a path planner designs a collision free path, then the robot dribbles the ball along the reference path with fast speed.

The key issue of formulating the path following problem is to choose the desired positions on the reference path, which results in different formulations of the path following problem. Orthogonal projection-based and *Virtual Vehicle*-based formulations present the two basic categories, which aim to select the static and dynamic desired positions, respectively. For the Attempto soccer robot, the path following control laws with respect to these two formulations have been addressed in this chapter. The designed nonlinear controllers are able to guarantee the closed-loop stability, which is proven by Lyapunov's stability theorem. Besides the path following control, the omnidirectional robot has another DOF to regulate its orientation. Considering the actuator dynamics and actuator saturation detailed in Chapter 3, the designed PD controller can keep the robot tracking the desired orientations, even though the actuator goes into saturation. To check the performance of the designed control laws, real-world experiments with the Attempto soccer robot were done in our robot laboratory. Experimental results show the good performance of the controlled system, and the guaranteed closed-loop stability regardless of the appearance of the actuator saturation.

Chapter 5

Nonlinear Model Predictive Control

With respect to the nonlinear characteristics of error kinematic models (4.4) - (4.6) and (4.17) of the path following problem shown in Chapter 4, many nonlinear controllers have been presented [7, 125, 53, 40, 105, 35, 98]. However, they rarely take the robot constraints into account, which are crucial factors capable of degrading robot performance, even destroying control stability [73, 27]. Moreover, only the errors between the current robot states and the desired states are considered in most control laws, which ignores the potential opportunity of improving the control performance by considering more information of the given path.

Motivated by the above considerations, the Nonlinear Model Predictive Control (NMPC) method has been adopted to solve the path following problem of the Attempto soccer robots. As NMPC can easily take robot constraints into account and utilize the future information to get current control inputs, NMPC has been used in many robotics applications. Considering the high computational requirement of NMPC, some works eliminate the computations which are necessary to keep control stability [13, 83, 86]. Some methods linearize the error kinematics, but they can only guarantee local stability [11, 149]. Many researchers presented detailed analysis of NMPC with mobile robots, but their applications were only in simulation [59, 88, 89]. The main contributions of this chapter are the analysis and design of stability guaranteed NMPC schemes with respect to nonlinear kinematic models, and the proof of the feasibility of applying NMPC to a real omnidirectional robot [95, 79].

5.1 Introduction

The theory of the Nonlinear Model Predictive Control (NMPC) has been reported in many documentations. To give an introduction, the following two sections refer [46], where the details of NMPC could be found.

Model Predictive Control (MPC), also known as Receding Horizon Control (RHC), has been an attractive optimal control law since the 1970s. For linear systems, MPC has shown its great benefit and been widely used in industry, specially in the process industries [134, 135]. As the inherent nonlinearities of general systems and constraints, applying MPC to nonlinear systems with nonlinear constraints is strongly motivated, and Nonlinear Model Predictive Control (NMPC) has become popular since the 1990s.

The task of NMPC is to on-line solve a finite horizon optimal control problem subject to the system models and constraints at each time step. This optimal control problem is called the open-loop optimal control problem, because it is solved based on each measurement and only the first part of the optimal control inputs is implemented until the new measurement becomes available. While the open-loop optimal control problem has to be solved again with the new measurements, NMPC is a feedback control law. The basic principle of NMPC is shown in Figure 5.1. Based on the measurements at time t, the future behavior of the system over

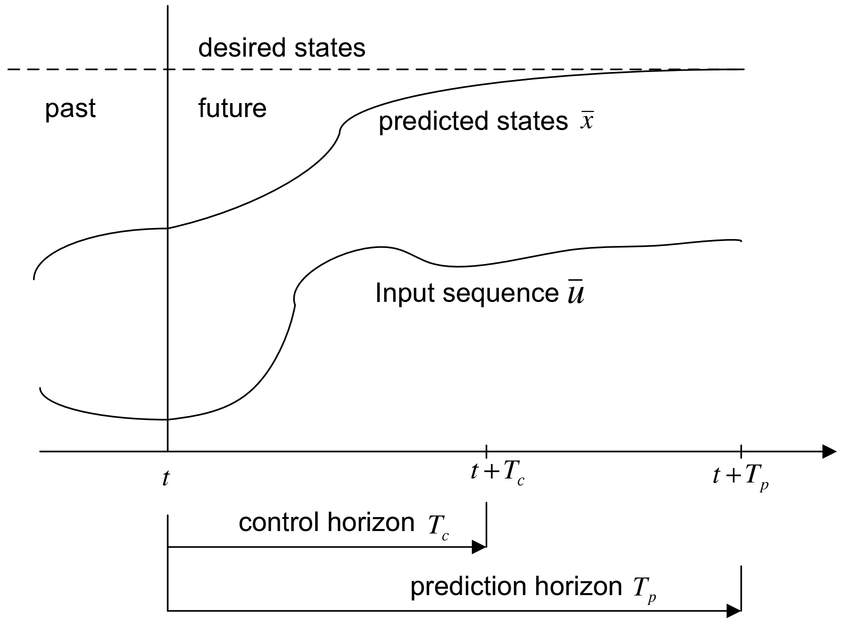


Figure 5.1: Principle of Nonlinear Model Predictive Control [46]

a prediction horizon T_p is predicted, and optimal inputs during a control horizon T_c ($T_c \leq T_p$) are calculated such that a predefined open-loop objective function is optimized under the system and input constraints, then the first optimal input value is taken as the current input.

Inheriting the advantages of normal optimal control laws, the formulation of the open-loop optimal control problem in NMPC can easily handle the system constraints and specify the desired control performance. By only using finite horizons, NMPC avoids the complex computation in the normal optimal control with an infinite horizon. But utilizing a finite horizon can make the predicted input and state trajectories differ from the actual trajectories even though no model mismatch and disturbances are present [9]. This trajectories' difference results in no stability guarantee of the closed-loop system. Therefore, the stability problem has to be emphasized in NMPC. On the other hand, to solve the open-loop optimal control problem of a nonlinear system is not an easy task, which makes the powerful optimization solver very important in NMPC. The following sections first formulate the NMPC problem, and then discuss the closed-loop stability problem and numerical solutions of NMPC. At the end, the implementations of NMPC in the motion control of the Attempto soccer robot show the feasibility and efficiency of NMPC in real-time applications.

5.2 Mathematical Formulation

A normal nonlinear system is described by the following differential equation:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{5.1}$$

subject to the constraints:

$$\mathbf{u}(t) \in U, \ \ \mathbf{x}(t) \in X, \ \ \forall t \geq 0, \tag{5.2}$$

where $\mathbf{x}(t) \subseteq \mathbb{R}^n$ and $\mathbf{u}(t) \subseteq \mathbb{R}^m$ are the n-dimensional state vector and m-dimensional input vector, respectively. X and U denote the sets of feasible states and inputs, respectively. Without loss of generality, if the system equilibrium is at $\mathbf{x}(t) = 0$ and $\mathbf{u}(t) = 0$, it should be included in sets X and U. The basic idea of NMPC is to iteratively execute the following steps :

1. predict the system's future behavior over a prediction horizon T_p at each time step t;

2. find optimal inputs $\bar{\mathbf{u}}(\cdot)\colon \ [t, t+T_p] \rightarrow U$ to minimize the value of the following objective function,

$$J(t, \mathbf{x}(t), \bar{\mathbf{u}}(\cdot)) = \int_t^{t+T_p} F(\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau, \tag{5.3}$$

   subject to:

   - $\dot{\bar{\mathbf{x}}}(\tau) = \mathbf{f}(\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau)), \ \bar{\mathbf{x}}(0) = \mathbf{x}(0),$

- $\bar{\mathbf{u}}(\tau) = \bar{\mathbf{u}}(t + T_c),\ \forall\tau \in [t + T_c, t + T_p]\,,$
- $\bar{\mathbf{u}}(\tau) \in U,\ \ \bar{\mathbf{x}}(\tau) \in X,\ \ \forall t \in [t, t + T_p]\,,$

where T_c is the control horizon with $T_c \leq T_p$, F is the cost function specifying the desired control performance, the bar denotes that the predicted values in the future are not the same as the real values;

3. take the first optimal input value $\bar{\mathbf{u}}(t)$ as the current input.

The design of cost function F directly reflects the desired control performance. Normally, the controlled system is expected to track the desired trajectories of system states and inputs. Thus F takes the form of a function of the errors between the real and desired system states and inputs. Because of the simple form, the standard quadratic function is often used as a cost function,

$$F(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = (\mathbf{x}_d - \bar{\mathbf{x}})^T \mathbf{Q}(\mathbf{x}_d - \bar{\mathbf{x}}) + (\mathbf{u}_d - \bar{\mathbf{u}})^T \mathbf{R}(\mathbf{u}_d - \bar{\mathbf{u}}). \tag{5.4}$$

$\mathbf{x}_d$ and $\mathbf{u}_d$ denote the desired states and inputs, which are contained in X and U, respectively. $\mathbf{Q}$ and $\mathbf{R}$ are positive definite and symmetric weighting matrices with corresponding dimensions. Moreover, a special formed cost function, such as a Lyapunov function, can also help to get additional necessary constraints for keeping closed-loop stability.

5.3 Stability

With respect to the set-up of NMPC, the optimal control inputs are computed based on the predicted system behavior. But in general, the predicted system behavior will differ from the actual closed-loop behavior although no model uncertainties and unknown disturbances occur, when a finite horizon is used in the open-loop optimal control problem [9]. It is not true that a repeated minimization of an objective function with a finite horizon leads to an optimal solution for the minimization of the objective function over the infinite horizon [17]. Therefore, using a finite horizon in the open-loop optimal control problem can not guarantee the closed-loop stability.

Many schemes have been proposed to achieve guaranteed stability. The most intuitive way to keep closed-loop stability is using an infinite horizon in the open-loop optimal control problem [17, 110]. It follows from Bellman's Principle of Optimality [14]: at one instance in time, the predicted state and input trajectories based on the solution of the open-loop optimal control problem with the infinite horizon are the same as the trajectories of the closed-loop system, and the remaining trajectories of the closed-loop system after a sampling interval are the predicted trajectories based on the optimal solution of the open-loop optimal control problem at the next sampling instance. Although the nonlinear optimal problem with infinite horizon can be reduced to solve a Hamilton-Jacobi-Bellman partial differential equation, the difficulties in finding such solutions make the infinite horizon NMPC not practical in real applications [21]. The relationship between the finite horizon length and closed-loop stability has also been studied. The results in [25, 147] show that for a constrained linear system, if the prediction horizon is sufficiently long, the terminal stability can be implicitly satisfied. Reference [130] presents, for a constrained discrete-time linear system, there always exists a finite horizon length guaranteeing the stability without any terminal penalties and constraints. The similar research of nonlinear system was done by Grimm *et al.* and Jadbabaie *et al.* [58, 74], which show that there exists also a finite horizon guaranteeing the stability for nonlinear model predictive control without terminal penalties and constraints. But there is no general way to find this finite horizon.

In practice, many schemes with respect to the finite horizon NMPC have been proposed [59, 24, 122, 75, 112, 113, 129, 131]. They guarantee closed-loop stability with modifications of the normal setup of NMPC by adding a terminal constraint,

$$\bar{\mathbf{x}}(t + T_p) \in \Omega \subseteq X, \tag{5.5}$$

and/or a terminal penalty $E(\bar{\mathbf{x}}(t + T_p))$ in the objective function,

$$J(t, \mathbf{x}(t), \bar{\mathbf{u}}(\cdot)) = \int_t^{t+T_p} F(\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau + E(\bar{\mathbf{x}}(t + T_p)). \tag{5.6}$$

Because these modifications are specially designed to keep closed-loop stability, and have no relationship with system restrictions and performance requirements, they are named as *stability constraints* [107, 109].

The most simple scheme uses a *zero terminal equality* as the constraint of the predicted terminal state, i.e. $\bar{\mathbf{x}}(t + T_p) = 0$. Although the implementation of this constraint is straightforward, finding an optimal solution satisfying the zero equality leads to a high computational burden on solving the corresponding nonlinear optimal problem [81, 24]. Michalska and Mayne reduced the *zero terminal equality* constraint to an inequality state constraint Ω, which is a neighbourhood around the origin. And a *dual-mode* control law is designed to guarantee control stability. When the system state enters the region, a local linear state feedback controller based on a linearized system is utilized. When the system state is outside of the region, a receding horizon control law is executed [112, 113, 108]. Without switching between different controllers, a Control Lyapunov Function (CLF) based scheme is proposed in [129, 131]. Once a global CLF is obtained, the derivative of the CLF along the predicted states and inputs trajectories is negative. If the corresponding decrease in the value of the CLF is greater than the decrease in the output of a pointwise min-norm controller [131], the stability of NMPC can be guaranteed. However, finding such a pointwise min-norm controller and a global CLF is not straightforward.

In order to lower the computational burden, some schemes remove the state inequality constraints and add a terminal penalty into the objective function. De Nicolao *et al.* [122] proposed an NMPC scheme guaranteeing the exponential stability of the equilibrium by adding a non quadratic terminal state penalty, which is the cost of the objective function incurred by applying a locally stabilizing linear controller from time $t+T_p$ to infinity. But the requirement of a large attraction region of the linear controller may not be easily satisfied. Jadbabaie *et al.* proposed to use a CLF based terminal penalty to achieve closed-loop stability [75], which is attractive because of plenty methods of obtaining a CLF.

From the viewpoint of computational cost of on-line solving the open-loop nonlinear optimal control problem, combining a terminal penalty and terminal constraints has been proven to be a feasible method to lower the computational burden [59, 46, 44]. The quasi-infinite horizon NMPC is one of such widely used control schemes. With terminal constraints, the system's terminal states are limited into a terminal region, where the terminal penalty gives an upper bound on the infinite horizon cost, i.e.

$$\int_{t+T_p}^{\infty} F(\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau \leq E(\bar{\mathbf{x}}(t+T_p)). \tag{5.7}$$

This implies the cost value of the infinite horizon problem is bounded by that of the corresponding finite horizon problem, i.e.

$$\min_{\mathbf{u}(\cdot)} \int_{t}^{\infty} F(\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau \leq \min_{\mathbf{u}(\cdot)} \int_{t}^{t+T_P} F(\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau + E(\bar{\mathbf{x}}(t+T_p)), \tag{5.8}$$

and the finite horizon extends to a quasi-infinite horizon, which denotes the name of this NMPC scheme. Therefore, the open-loop optimal control problem of the quasi-infinite horizon NMPC is to find optimal inputs $\bar{\mathbf{u}}(\cdot)\colon [t, t+T_p] \rightarrow U$ to minimize the value of the following objective function,

$$J(t, \mathbf{x}(t), \bar{\mathbf{u}}(\cdot)) = \int_{t}^{t+T_p} F(\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau + E(\bar{\mathbf{x}}(t+T_p)), \tag{5.9}$$

subject to:

$$\dot{\bar{\mathbf{x}}}(\tau) = \mathbf{f}(\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau)),\ \bar{\mathbf{x}}(0) = \mathbf{x}(0), \tag{5.10a}$$

$$\bar{\mathbf{u}}(\tau) = \bar{\mathbf{u}}(t+T_c),\ \forall \tau \in [t+T_c, t+T_p], \tag{5.10b}$$

$$\bar{\mathbf{u}}(\tau) \in U,\ \ \bar{\mathbf{x}}(\tau) \in X,\ \ \forall t \in [t, t+T_p], \tag{5.10c}$$

$$\bar{\mathbf{x}}(t+T_p) \in \Omega \subseteq X. \tag{5.10d}$$

To guarantee the stability, the following theorem presents the necessary conditions [44]:

Theorem 5.3.1 *Suppose*

- $\mathbf{f}:\mathbb{R}^n\times\mathbb{R}^m$ *is continuous and satisfies* $\mathbf{f}(\mathbf{0},\mathbf{0})=\mathbf{0}$. $(\mathbf{0},\mathbf{0})$ *is an equilibrium of the system,*
- $U\subset\mathbb{R}^m$ *is compact,*$X\subseteq\mathbb{R}^n$ *is connected and* $(\mathbf{0},\mathbf{0})\in X\times U$*,*
- *system (5.1) has a unique solution for any initial condition* $\mathbf{x}_0\in\mathbb{R}^n$ *and any piecewise continuous* $\mathbf{u}(\cdot):[0,\infty)\to U$*,*
- *the cost function is continuous with* $F(\mathbf{0},\mathbf{0})=0$ *and* $F(\mathbf{x},\mathbf{u})>0$ *for every state* $\mathbf{x}$ *and input* $\mathbf{u}$*,*
- *the open-loop optimal control problem has a solution at time* $t=0$*,*

for a continuously differentiable terminal penalty $E(\mathbf{x})$ *with* $E(\mathbf{0})=0$*, and a closed region* $\Omega\subseteq X$ *including the origin, if there is a control law* $\mathbf{k}(\mathbf{x})\in U$ *with* $\mathbf{k}(\mathbf{0})=0$ *such that*

$$\dot{E}(\mathbf{x})+F(\mathbf{x},\mathbf{k}(\mathbf{x}))\leq 0,\forall\mathbf{x}\in\Omega, \tag{5.11}$$

the closed-loop system is asymptotically stable with the attraction region being the set of states for which the open-loop optimal control problem has a feasible solution.

Proof: The proof consists of two steps [44]. The first step shows that initial feasibility implies feasibility afterwards. The second one proves the decreasing of a Lyapunov function of the closed-loop system.

Feasibility

Consider any instant t_i (e.g. t_0), there exists an optimal solution $\bar{\mathbf{u}}^*(\tau;\mathbf{x}(t_i))$ with $\tau\in\left[t_i,t_{i+T_p}\right]$ of the open-loop control problem depicted in (5.9) and (5.10). When $\bar{\mathbf{u}}^*(\tau;\mathbf{x}(t_i))$ is implemented between t_i and t_{i+1} and no model mismatch nor disturbances are present, the equation of $\mathbf{x}(t_{i+1})=\bar{\mathbf{x}}(t_{i+1};\mathbf{x}(t_i),\bar{\mathbf{u}}^*(\tau;\mathbf{x}(t_i))$ holds. $\bar{\mathbf{x}}(t_{i+1};\mathbf{x}(t_i),\bar{\mathbf{u}}^*(\tau;\mathbf{x}(t_i))$ denotes the state at time t_{i+1} resulted from the state $\mathbf{x}(t_i)$ and the control $\bar{\mathbf{u}}^*(\tau;\mathbf{x}(t_i))$ with $\tau\in[t_i,t_{i+1})$. Furthermore, the predicted terminal state satisfies $\bar{\mathbf{x}}(t_{i+T_p};\mathbf{x}(t_i),\bar{\mathbf{u}}^*(\tau;\mathbf{x}(t_i))\in\Omega$. It follows from Theorem 5.3.1 that there exists at least one input $\mathbf{k}(\mathbf{x})$ for the predicted state $\bar{\mathbf{x}}$

over $[t_i + T_p, t_{i+1} + T_p]$. Taking any such input we obtain an admissible input for $t_i + \sigma$ with $\sigma \in (0, t_{i+1} - t_i]$:

$$\tilde{\mathbf{u}}(\tau; \mathbf{x}(t_i + \sigma)) = \begin{cases} \bar{\mathbf{u}}^*(\tau; \mathbf{x}(t_i)) & \tau \in [t_i + \sigma, t_i + T_p] \\ \mathbf{k}(\mathbf{x}(\tau)) & \tau \in (t_i + T_p, t_i + T_p + \sigma]. \end{cases} \tag{5.12}$$

When $\sigma = t_{i+1} - t_i$ holds, $\tilde{\mathbf{u}}(\tau; \mathbf{x}(t_{i+1}))$ is an admissible input at time t_{i+1}. This means that admissibility at time t_i implies admissibility at time t_{i+1}. Therefore, if the open-loop control problem depicted in (5.9) and (5.10) has a solution at $t = 0$, it will have a solution for all $t > 0$.

Convergence

Considering the following value function as a Lyapunov function

$$\begin{aligned} V(\mathbf{x}(t)) = & \\ & \int_t^{t+T_p} F(\bar{\mathbf{x}}(\tau; \mathbf{x}(t), \bar{\mathbf{u}}^*(\cdot; \mathbf{x}(t))), \bar{\mathbf{u}}^*(\tau; \mathbf{x}(t)))d\tau \\ & + E(\bar{\mathbf{x}}(t + T_p\, ; \mathbf{x}(t), \bar{\mathbf{u}}^*(\cdot; \mathbf{x}(t)))), \end{aligned} \tag{5.13}$$

where $\bar{\mathbf{x}}(\tau; \mathbf{x}(t), \bar{\mathbf{u}}^*(\cdot; \mathbf{x}(t)))$ denotes the state at time τ resulting from the state $\mathbf{x}(t)$ and the control $\bar{\mathbf{u}}^*(\tilde{t}; \mathbf{x}(t))$ with $\tilde{t} \in [t, \tau)$. The value of the function V for the state $\mathbf{x}(t_i)$ is given by:

$$\begin{aligned} V(\mathbf{x}(t_i)) = & \\ & \int_{t_i}^{t_i+T_p} F(\bar{\mathbf{x}}(\tau; \mathbf{x}(t_i), \bar{\mathbf{u}}^*(\cdot; \mathbf{x}(t_i))), \bar{\mathbf{u}}^*(\tau; \mathbf{x}(t_i)))d\tau \\ & + E(\bar{\mathbf{x}}(t_i + T_p\, ; \mathbf{x}(t_i), \bar{\mathbf{u}}^*(\cdot; \mathbf{x}(t_i)))). \end{aligned} \tag{5.14}$$

The cost resulting from (5.12) starting from any state $\bar{\mathbf{x}}(t_i + \sigma; \mathbf{x}(t_i), \bar{\mathbf{u}}^*(\cdot; \mathbf{x}(t_i)))$ with $\sigma \in (0, t_{i+1} - t_i]$ is given by:

$$\begin{aligned} J(\mathbf{x}(t_i + \sigma), \tilde{\mathbf{u}}(\cdot; \mathbf{x}(t_i + \sigma))) = & \\ & \int_{t_i+\sigma}^{t_i+\sigma+T_p} F(\bar{\mathbf{x}}(\tau; \mathbf{x}(t_i + \sigma), \tilde{\mathbf{u}}(\cdot; \mathbf{x}(t_i + \sigma))), \tilde{\mathbf{u}}(\tau; \mathbf{x}(t_i + \sigma)))d\tau \\ & + E(\bar{\mathbf{x}}(t_i + \sigma + T_p\, ; \mathbf{x}(t_i + \sigma), \tilde{\mathbf{u}}(\cdot; \mathbf{x}(t_i + \sigma)))) \\ = & \int_{t_i+\sigma}^{t_i+T_p} F(\bar{\mathbf{x}}(\tau; \mathbf{x}(t_i + \sigma), \tilde{\mathbf{u}}(\cdot; \mathbf{x}(t_i + \sigma))), \tilde{\mathbf{u}}(\tau; \mathbf{x}(t_i + \sigma)))d\tau \\ & + \int_{t_i+T_p}^{t_i+\sigma+T_p} F(\bar{\mathbf{x}}(\tau; \mathbf{x}(t_i + \sigma), \tilde{\mathbf{u}}(\cdot; \mathbf{x}(t_i + \sigma))), \tilde{\mathbf{u}}(\tau; \mathbf{x}(t_i + \sigma)))d\tau \\ & + E(\bar{\mathbf{x}}(t_i + \sigma + T_p\, ; \mathbf{x}(t_i + \sigma), \tilde{\mathbf{u}}(\cdot; \mathbf{x}(t_i + \sigma)))). \end{aligned} \tag{5.15}$$

Substituting (5.14) into (5.15), we obtain

$$
\begin{aligned}
&J(\mathbf{x}(t_i+\sigma), \tilde{\mathbf{u}}(\cdot;\mathbf{x}(t_i+\sigma))) = \\
&\qquad V(\mathbf{x}(t_i)) - \int_{t_i}^{t_i+\sigma} F(\bar{\mathbf{x}}(\tau;\mathbf{x}(t_i), \tilde{\mathbf{u}}^*(\cdot;\mathbf{x}(t_i))), \tilde{\mathbf{u}}^*(\tau;\mathbf{x}(t_i)))d\tau \\
&\qquad\qquad - E(\bar{\mathbf{x}}(t_i+T_p\,;\mathbf{x}(t_i), \tilde{\mathbf{u}}^*(\cdot;\mathbf{x}(t_i)))) \\
&\quad + \int_{t_i+T_p}^{t_i+\sigma+T_p} F(\bar{\mathbf{x}}(\tau;\mathbf{x}(t_i+\sigma), \tilde{\mathbf{u}}(\cdot;\mathbf{x}(t_i+\sigma))), \tilde{\mathbf{u}}(\tau;\mathbf{x}(t_i+\sigma)))d\tau \\
&\qquad\qquad + E(\bar{\mathbf{x}}(t_i+\sigma+T_p\,;\mathbf{x}(t_i+\sigma), \tilde{\mathbf{u}}(\cdot;\mathbf{x}(t_i+\sigma)))). \qquad (5.16)
\end{aligned}
$$

Integrating inequality (5.11) from t_i+T_p to $t_i+\sigma+T_p$ starting from $\mathbf{x}(t_i+T_p)$, we obtain that the last three terms at the right side of (5.16) are upper bounded by zero. Thus, the following inequality holds

$$
\begin{aligned}
&J(\mathbf{x}(t_i+\sigma), \tilde{\mathbf{u}}(\cdot;\mathbf{x}(t_i+\sigma))) - V(\mathbf{x}(t_i)) \\
&\qquad \le -\int_{t_i}^{t_i+\sigma} F(\bar{\mathbf{x}}(\tau;\mathbf{x}(t_i), \tilde{\mathbf{u}}^*(\cdot;\mathbf{x}(t_i))), \tilde{\mathbf{u}}^*(\tau;\mathbf{x}(t_i)))d\tau \qquad (5.17)
\end{aligned}
$$

Because $\tilde{\mathbf{u}}$ is only a feasible input for $\mathbf{x}(t_i+\sigma)$ but not necessary to be the optimal input, the following inequality holds

$$
\begin{aligned}
&V(\mathbf{x}(t_i+\sigma)) - V(\mathbf{x}(t_i)) \\
&\qquad \le -\int_{t_i}^{t_i+\sigma} F(\bar{\mathbf{x}}(\tau;\mathbf{x}(t_i), \tilde{\mathbf{u}}^*(\cdot;\mathbf{x}(t_i))), \tilde{\mathbf{u}}^*(\tau;\mathbf{x}(t_i)))d\tau \le 0. \qquad (5.18)
\end{aligned}
$$

Repeatedly using the inequality (5.18) yields

$$
\begin{aligned}
&V(\mathbf{x}(t)) - V(\mathbf{x}(0)) \\
&\qquad \le -\int_{0}^{t} F(\bar{\mathbf{x}}(\tau;\mathbf{x}(0), \tilde{\mathbf{u}}^*(\cdot;\mathbf{x}(0))), \tilde{\mathbf{u}}^*(\tau;\mathbf{x}(0)))d\tau \le 0. \qquad (5.19)
\end{aligned}
$$

This inequality establishes that the value function $V(\mathbf{x}(t))$ is decreasing. Considering the cost function F is continuous and the integral term at the right side of (5.19) is lower bounded, we can obtain that the state $\mathbf{x}$ converges to the origin when time converges to infinity by using Barbalat's lemma (1). This means the closed-loop system is asymptotically stable.

It is noticed in Theorem 5.3.1, the feedback controller $\mathbf{k}(\mathbf{x})$ is not used to control the system, but used to select the suitable terminal penalty and terminal constraints. Although the terminal penalty and terminal constraints can be selected off-line, it is not a easy task. In [24, 22, 23], a quadratic form terminal penalty is selected. The terminal constraint is designed based on a linearized feedback controller with respect to the Jacobian linearization of the original nonlinear system. The necessary condition is that the Jacobian linearization is stabilizable. Without linearization, recent works in [131, 47, 59] give the idea to directly chose the terminal penalty and terminal constraints with the original nonlinear system according to Theorem 5.3.1, where the chosen terminal penalty also takes the quadratic form and the feedback controller is also linear.

5.4 Numerical Solutions

On-line solving the open-loop nonlinear optimal control problem plays a key role in NMPC. Although the high computational demands of solving the nonlinear finite optimization problem make NMPC hard to be implemented in applications with fast sampling time and limited computational resources [46], many research results show the feasibility of applying NMPC in real-time processes [83, 82]. Referring [46], this section introduces solution methods for the optimal control problem (5.9) of the quasi-infinite NMPC subject to the conditions in (5.10).

5.4.1 Solution Methods

In principle, there are three basic approaches.

- **Hamilton-Jacobi-Bellmann Partial Differential Equations, Dynamic Programming**
  This approach directly obtains a feedback control law $\mathbf{u}^* = \mathbf{k}(\mathbf{x})$ based on the solution of the so called Hamilton-Jacobi-Bellmann partial differential equations. Although the closed-loop controller works for the whole horizon and is valid for every initial condition, the high computational requirement of solving such partial differential equations limits this approach only to very small dimensional systems. This is also the main motivation of researching receding horizon control laws.

- **Euler-Lagrange Differential Equations, Calculus of Variations, Maximum Principles**
  This approach utilizes the necessary conditions for constrained optimization problems and gets a time-based control value, which is only valid for the specified initial condition $\mathbf{x}(t)$. As a boundary value problem has to be solved, the high computational burden makes this approach not suitable for on-line implementation.

- **Direct Solution Algorithm**
  The direct solution algorithm transfers the original control problem over a finite horizon into a finite dimensional Nonlinear Programming problem (NLP). As the NLP can be solved with standard static optimization techniques, direct solution algorithms have been proven to be most successful for the large scale optimal control problems, and are normally used for calculating the on-line solution of NMPC.

5.4.2 Direct Solution Algorithm

The direct solution algorithm uses a finite parameterization of the control trajectory to solve the finite dimensional NLP. The basic parameterization method utilizes a piecewise constant control input on each partition over the prediction horizon $[t, t + T_p]$, i.e. $\mathbf{u}(\delta) = \mathbf{u}(t_i)$ with $\delta \in [t_i, t_{i+1})$. When the prediction horizon is divided by a constant interval τ shown in Figure 5.2, the optimization problem becomes

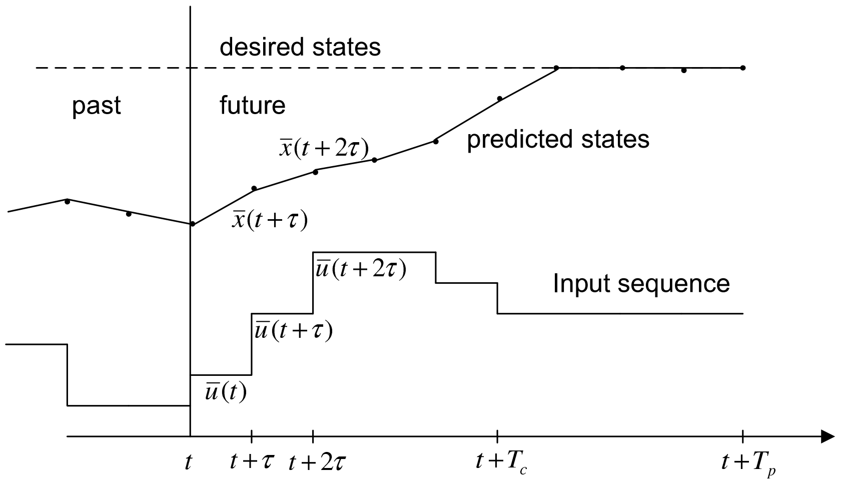


Figure 5.2: Parameterization of the direct solution for the open-loop optimal control problem

$$\min_{\{\mathbf{u}_1,\mathbf{u}_2,\ldots\mathbf{u}_{\frac{T_p}{\tau}}\}} \mathbf{J}(\mathbf{x}(t), \{\mathbf{u}_1, \mathbf{u}_2, \ldots\mathbf{u}_{\frac{T_p}{\tau}}\}), \tag{5.20}$$

subject to the constraints in (5.10), where $\mathbf{u}_j$ denotes $\mathbf{u}(t + (j - 1)\tau)$ with $j = 1, 2, \ldots, \frac{T_p}{\tau}$. There are two basic solution strategies for this optimization problem [16, 104].

1. **Sequential approach (single shooting)**
   At each sampling time t, the sequential approach updates the system's future behavior with input series $\mathbf{u}_1, \mathbf{u}_2, ...\mathbf{u}_{\frac{T_p}{\tau}}$, and a numerical integration of system model (5.1) based on the current system state $\mathbf{x}(t)$. Single shooting represents a pure sequential approach. The numerical effort of the single shooting method is highly based on the complexity of the discretization of the control trajectory. The solution of single shooting depends on the sensitivity of the states with respect to the control variables. As a small number of the control variables is required in solving the NLP, single shooting is easier to implement.

2. **Simultaneous approach (multiple shooting)**
   The simultaneous approach solves the optimization problem with stabilizing endpoint constraints. Multiple shooting is one of most widely used simultaneous approaches, where the system states at the sampling points are taken as additional optimization variables to keep $\bar{\mathbf{s}}_{i+1} = \bar{\mathbf{x}}(t_{i+1}; \bar{\mathbf{s}}_i, \bar{\mathbf{u}}_i)$. $\bar{\mathbf{s}}_{i+1}$ is the system state at time t_i. $\bar{\mathbf{x}}(t_{i+1}; \bar{\mathbf{s}}_i, \bar{\mathbf{u}}_i)$ denotes the predicted system states resulted from the system state $\bar{\mathbf{s}}_i$ with the control $\mathbf{u}_i)$. Another popular simultaneous approach is direct collocation, whose details can be found in [16, 15]. Although simultaneous approaches are applicable to the highly unstable systems, which direct shooting can not handle, the large number of optimization variables increases the computational cost.

5.4.3 Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) is an efficient iterative method for the solution of the NLP arising from NMPC, such as the problem (5.20). Considering an NLP as

$$\min_{\boldsymbol{\xi}} \varphi(\boldsymbol{\xi})$$

subject to

$$\mathbf{a}(\boldsymbol{\xi}) = 0,\ \mathbf{b}(\boldsymbol{\xi}) \geq 0,$$

where $\mathbf{a} \in \mathbb{R}^{n_a}$ and $\mathbf{b} \in \mathbb{R}^{n_b}$ denote equality and inequality constraints, respectively. SQP solves this problem based on the line search method. The vector of optimization variables $\boldsymbol{\xi}_k \in \mathbb{R}^n$ and the vector of multipliers $\mathbf{v}_k = (\boldsymbol{\mu}, \boldsymbol{\sigma})_k \in \mathbb{R}^{n_a+n_b}$ are updated from iteration number k to $k+1$ by

$$\begin{pmatrix} \boldsymbol{\xi}_{k+1} \\ \mathbf{v}_{k+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\xi}_k \\ \mathbf{v}_k \end{pmatrix} + \alpha_k \begin{pmatrix} \mathbf{d}_k \\ \mathbf{u}_k - \mathbf{v}_k \end{pmatrix}, \quad k = 0, 1, 2, ... \ ,$$

where the *search direction* $(\mathbf{d}_k, \mathbf{u}_k)$ comes from the solution of a linearly constrained Quadratic Problem (**QP**),

$$\min_{\mathbf{d}\in\mathbb{R}^n} \frac{1}{2}\mathbf{d}^T\mathbf{C}_k\mathbf{d} + \nabla\varphi(\boldsymbol{\xi})^T\mathbf{d}$$

subject to

$$\begin{aligned} \nabla a_i(\boldsymbol{\xi}_k)^T\mathbf{d} + a_i(\boldsymbol{\xi}_k) = 0, \;\; i = 1, ..., n_a, \\ \nabla b_j(\boldsymbol{\xi}_k)^T\mathbf{d} + b_j(\boldsymbol{\xi}_k) \geq 0, \;\; j = 1, ..., n_b, \end{aligned}$$

based on the following quadratic approximation of the Lagrangian $\mathcal{L}$,

$$\mathcal{L}(\boldsymbol{\xi}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \varphi(\boldsymbol{\xi}) - \sum_{i=1}^{n_a}\mu_i a_i(\boldsymbol{\xi}) - \sum_{i=1}^{n_a}\sigma_i b_i(\boldsymbol{\xi}), \;\; \boldsymbol{\mu}\in\mathbb{R}^{n_a}, \;\; \boldsymbol{\sigma}\in\mathbb{R}^{n_b}.$$

$\mathbf{C}_k$ is a positive definite approximation of the Hessian $\mathbf{H}_k$ of the Lagrangian $\mathcal{L}(\boldsymbol{\xi}_k, \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)$. The quadratic problem is solved by an iterative method to get $\mathbf{d}_k$ and the corresponding multiplier $\mathbf{u}_k$.

The *step size* $\alpha_k \in \mathbb{R}$ is obtained by minimizing a merit function (*line search*)

$$\psi_m\left(\begin{pmatrix}\boldsymbol{\xi}\\ \mathbf{v}\end{pmatrix} + \alpha\begin{pmatrix}\mathbf{d}\\ \mathbf{u}-\mathbf{v}\end{pmatrix}\right)$$

As a suitable choice, the merit function can be an augmented Lagrangian as,

$$\begin{aligned} \psi_m(\boldsymbol{\xi}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \varphi(\boldsymbol{\xi}) &- \sum_{i=1}^{n_a}\left(\mu_i a_i(\boldsymbol{\xi}) - \frac{1}{2}r_i a_i^2(\boldsymbol{\xi})\right) \\ &- \sum_{j\in\mathbf{J}}\left(\sigma_j b_j(\boldsymbol{\xi}) - \frac{1}{2}r_{n_a+j}b_j^2(\boldsymbol{\xi})\right) - \frac{1}{2}\sum_{j\in\mathbf{K}}\frac{\sigma_j^2}{r_{n_a}+j}, \end{aligned} \tag{5.21}$$

where the sets $\mathbf{J}$ and $\mathbf{K}$ are chosen as $\mathbf{J} = \{j|1 \leq j \leq n_b, b_j(y) \geq \frac{\sigma_j}{r_{n_a+j}}\}$, $\mathbf{k} = \{1, .., n_b\}$ $\mathbf{J}$ with $r_i > 0, i = 1, ..., n_a + n_b$.

The details of basic SQP are introduced in [61, 128, 18]. A recent research of application of SQP to NMPC gives comparisons among implementations of SQP with different methods, such as feasible and infeasible path methods, sequential and simultaneous methods and reduced and full space methods [104].

5.5 Implementation

Based on the error kinematic models of the path following problem and the orientation tracking problem introduced in Chapter 4, the quasi-infinite NMPC has been successfully used in the motion control of the Attempto soccer robot. With respect to the different formulations of the path following problem, Subsection 5.5.1 describes the details of choosing the terminal penalty and terminal constraints of NMPC. Subsection 5.5.2 presents the numerical integration method of the robot model equations. Subsection 5.5.3 emphasizes a practical issue about computational delays in applying NMPC.

5.5.1 Terminal penalty and constraints

Orthogonal Projection-based Case

With respect to the error kinematic model (4.5) of the path following problem ,

$$\dot{y}_e = v_R \sin \alpha_e, \tag{5.22}$$

the aim of the path following control is to drive y_e and α_e to zero. By introducing a new input $u_{e1} = v_R \sin \alpha_e$, and combining the kinematics of the orientation error θ_e, the error kinematics of the robot motion is given by the following equations,

$$\begin{bmatrix} \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} u_{e1} \\ u_{e2} \end{bmatrix} \tag{5.23}$$

It is clear that this model has the equilibrium at $y_e = 0$, $\theta_e = 0$, $u_{e1} = 0$ and $u_{e2} = 0$, which is a necessary condition in Theorem 5.3.1. In order to stabilize the errors around the equilibrium as close as possible, a quadratic form cost function is selected in NMPC,

$$F(\mathbf{x}, \mathbf{u}) = \mathbf{x}_e^T \mathbf{Q} \mathbf{x}_e + \mathbf{u}_e^T \mathbf{R} \mathbf{u}_e, \tag{5.24}$$

where $\mathbf{x}_e$ is the error vector $(y_e, \theta_e)^T$, $\mathbf{u}_e$ is the input vector $(u_{e1}\ u_{e2})^T$, $\mathbf{Q}$ and $\mathbf{R}$ are positive diagonal matrices with corresponding dimensions.

In the quasi-infinite horizon NMPC, a terminal penalty and terminal constraints are required to guarantee closed-loop stability. Based on Theorem 5.3.1, the following terminal penalty has been chosen and added into the objective function,

$$E(t + T_p) = \frac{1}{2} \mathbf{x}_e(t + T_p)^T \mathbf{x}_e(t + T_p), \tag{5.25}$$

where $\mathbf{x}_e(t + T_p)$ denotes the terminal error state vector $(y_{eT_p}\ \theta_{eT_p})^T$. When the terminal feedback controller is chosen as

$$u_{e1}^L = -\alpha y_{eT_p}, \tag{5.26}$$

$$u_{e2}^L = -\beta \theta_{eT_p}, \tag{5.27}$$

with $\alpha \geq 0$ and $\beta \geq 0$, the left side of stability condition (5.11) results in

$$\begin{aligned}\dot{E}(t+T_p)+F(t+T_p) = \ & (-\alpha+q_{11}+r_{11}\alpha^2)y_{eT_p}^2 + \\ & (-\beta+q_{22}+r_{22}\beta^2)\theta_{eT_p}^2.\end{aligned} \tag{5.28}$$

Therefore, the following inequalities

$$\alpha - q_{11} - r_{11}\alpha^2 \geq 0, \tag{5.29}$$

$$\beta - q_{22} - r_{22}\beta^2 \geq 0, \tag{5.30}$$

make (5.28) satisfy stability condition (5.11). The terminal controller (5.26) and (5.27) are only used to get the terminal constraints considering closed-loop stability. They are very simple based on only two parameters α and β, which control the convergence speed of errors y_e and θ_e. The advantage of (5.29) and (5.30) is that the values of q_{11}, q_{22}, r_{11}, r_{22}, α and β can be easily selected off-line.

On the other hand, to guarantee the existence of the terminal feedback controllers, the following constraint should be satisfied,

$$-1 \leq \sin\alpha_{eT_p} = \frac{u_{e1}^L}{v_R} \leq 1, \tag{5.31}$$

and the system constraints should not be broken, i.e.

$$-\begin{bmatrix}\dot{q}_m\\ \dot{q}_m\\ \dot{q}_m\end{bmatrix} \leq \begin{bmatrix}\cos\delta & \sin\delta & L_w\\ -\cos\delta & \sin\delta & L_w\\ 0 & -1 & L_w\end{bmatrix}\begin{bmatrix}v_R cos(\alpha_{eT_p}+\theta_p-\theta)\\ v_R sin(\alpha_{eT_p}+\theta_p-\theta)\\ \omega_d^L-u_{e2}^L\end{bmatrix} \leq \begin{bmatrix}\dot{q}_m\\ \dot{q}_m\\ \dot{q}_m\end{bmatrix}, \tag{5.32}$$

where α_{eT_p} is from the control value u_1 with $\alpha_{eT_p} = \arcsin\frac{u_{e1}^L}{v_R}$, ω_d^L denotes the desired rotation velocity at the terminal time $t+T_p$.

With simple transformations, the control values of the Attempto soccer robot are given by

$$\begin{aligned}u_1 = \ & v_d\cos\alpha,\\ u_2 = \ & v_d\sin\alpha,\\ u_3 = \ & \omega_d - u_{e2}^*,\end{aligned}$$

with $\alpha = \arcsin\frac{u_{e1}^*}{v_R} + \theta_P$. $(u_{e1}^*, u_{e2}^*)^T$ is the first vector of the solution of the open-loop optimal control problem at each time.

***Virtual Vehicle*-based Case**

Combining the error kinematic model (4.17) and the kinematics of the orientation error θ_e, the following error kinematic model having the equilibrium at $\mathbf{x}_e = 0$ and $\mathbf{u}_e = 0$ comes into being,

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} 0 & c(s)\dot{s} & 0 \\ -c(s)\dot{s} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ 4y_e \\ \theta_e \end{bmatrix} + \begin{bmatrix} u_{e1} \\ u_{e2} \\ u_{e3} \end{bmatrix}, \tag{5.33}$$

with

$$\begin{bmatrix} u_{e1} \\ u_{e2} \\ u_{e3} \end{bmatrix} = \begin{bmatrix} -\dot{s} + v_R \cos\alpha_e \\ v_R \sin\alpha_e \\ \omega_d - \omega \end{bmatrix}, \tag{5.34}$$

To drive the errors approaching zero, the selected cost function also has the quadratic form,

$$F(\mathbf{x}, \mathbf{u}) = \mathbf{x}_e^T \mathbf{Q} \mathbf{x}_e + \mathbf{u}_e^T \mathbf{R} \mathbf{u}_e, \tag{5.35}$$

$\mathbf{x}_e = (x_e, y_e, \theta_e)^T$ is the error vector. $\mathbf{u}_e = (u_{e1}\ u_{e2}\ u_{e3})^T$ is the input vector. $\mathbf{Q}$ and $\mathbf{R}$ are positive diagonal weight matrices with corresponding dimensions. To guarantee the control stability, the following Lyapunov function is selected as the terminal penalty:

$$E(t + T_p) = \frac{1}{2}\mathbf{x}_e(t + T_p)^T \mathbf{x}_e(t + T_p), \tag{5.36}$$

where $\mathbf{x}_e(t+T_P) = (x_{eT_p}\ y_{eT_p}\ \theta_{eT_p})^T$ denotes the terminal state. When the terminal feedback controllers are designed as:

$$u_{e1}^L = -\kappa x_{eT_p}, \tag{5.37}$$
$$u_{e2}^L = -\beta y_{eT_p}, \tag{5.38}$$
$$u_{e3}^L = -\gamma \theta_{eT_p}, \tag{5.39}$$

with parameters $\kappa \geq 0$, $\beta \geq 0$, and $\gamma \geq 0$, the left side of stability condition (5.11) becomes

$$\begin{aligned} \dot{E}(t + T_p) + F(t + T_p) = & x_{eT_p}^2(-\kappa + q_{11} + \kappa^2 r_{11}) + y_{eT_p}^2(-\beta + q_{22} + \beta^2 r_{22}) \\ & + \theta_{eT_p}^2(-\gamma + q_{33} + \gamma^2 r_{33}). \end{aligned} \tag{5.40}$$

Therefore, the following constraints can satisfy the stability condition (5.11),

$$\kappa - q_{11} - \kappa^2 r_{11} \geq 0, \tag{5.41}$$
$$\beta - q_{22} - \beta^2 r_{22} \geq 0, \tag{5.42}$$
$$\gamma - q_{33} - \gamma^2 r_{33} \geq 0. \tag{5.43}$$

Furthermore, constraint

$$-1 \leq \frac{u_{e2}^L}{v_R} \leq 1, \tag{5.44}$$

should be satisfied to obtain a reasonable input value u_{e2}^L. Moreover, the controlled values (5.37)-(5.39) have to satisfy the system constraints, which are the bounded wheel velocities. Combining (3.2), (5.34) and (5.37)-(5.39), the second part of terminal constraints is deduced as

$$-\begin{bmatrix} \dot{q}_m \\ \dot{q}_m \\ \dot{q}_m \end{bmatrix} \leq \begin{bmatrix} \cos\delta & \sin\delta & L_w \\ -\cos\delta & \sin\delta & L_w \\ 0 & -1 & L_w \end{bmatrix} \begin{bmatrix} -\kappa x_{eT} + \dot{s} \\ -\beta y_{eT} \\ -\gamma\theta_{eT} + \omega_d^L \end{bmatrix} \leq \begin{bmatrix} \dot{q}_m \\ \dot{q}_m \\ \dot{q}_m \end{bmatrix}. \tag{5.45}$$

Similar to the orthogonal projection-based case, the control values of the omnidirectional robot are given by

$$\begin{aligned} u_1 &= v_d \cos\alpha, \\ u_2 &= v_d \sin\alpha, \\ u_3 &= \omega_d - u_{e3}^*, \end{aligned}$$

with $\alpha = \arcsin \frac{u_{e2}^*}{v_R} + \theta_p$. $(u_{e1}^*, u_{e2}^*, u_{e3}^*)^T$ is the first vector of the solution of the open-loop optimal control problem at each time. u_{e1}^* gives the optimal value of $\dot{s}$ which determines the desired robot position on the reference path.

5.5.2 Formulation

The single shooting approach is chosen to solve the NLP from the NMPC formulations of controlling the Attempto soccer robot because of its low computational burden. The robot model has to be updated with numerical integration to obtain the predicted states in the future. The robot kinematic model (3.2) is discretized as:

$$\theta(k+1) = \theta(k) + \omega(k)\tau, \tag{5.46}$$

$$\begin{aligned} x_R^w(k+1) = \ & x_R^w(k) + \frac{\dot{x}_R^m(k)}{\omega(k)}[\sin(\theta(k+1)) - \sin(\theta(k))] \\ & + \frac{\dot{y}_R^m(k)}{\omega(k)}[\cos(\theta(k+1)) - \cos(\theta(k))], \end{aligned} \tag{5.47}$$

$$\begin{aligned} y_R^w(k+1) = \ & y_R^w(k) - \frac{\dot{x}_R^m(k)}{\omega(k)}[\cos(\theta(k+1)) - \cos(\theta(k))] \\ & + \frac{\dot{y}_R^m(k)}{\omega(k)}[\sin(\theta(k+1)) - \sin(\theta(k))], \end{aligned} \tag{5.48}$$

if $\omega(k) = 0$:

$$x_R^w(k+1) = x_R^w(k) + [\dot{x}_R^m(k)\cos(\theta(k)) - \dot{y}_R^m(k)\sin(\theta(k))]\tau, \tag{5.49}$$

$$y_R^w(k+1) = y_R^w(k) + [\dot{x}_R^m(k)\sin(\theta(k)) + \dot{y}_R^m(k)\cos(\theta(k))]\tau. \tag{5.50}$$

Taking the updated robot states as well as the reference path and desired robot orientations into account, the error values in model (5.23) or (5.33) can be consequently obtained. If this update process is modeled by a function $\mathbf{g}(\mathbf{x_e}, \mathbf{u_e})$, the NMPC scheme used to solve the path following problem of the Attempto soccer robot is formulated as follows,

$$\min_{\{\mathbf{u}_e(k), \mathbf{u}_e(k+\tau), \ldots \mathbf{u}_e(k+T_p)\}} \mathbf{J}(\mathbf{x}_e(k), \{\mathbf{u}_e(k), \mathbf{u}_e(k+\tau), \ldots \mathbf{u}_e(k+T_p)\}), \tag{5.51}$$

with

$$\mathbf{J}(k) = \sum_{j=1}^{\frac{T_p}{\tau}} \mathbf{x}_e(k+j\tau)^T \mathbf{Q} \mathbf{x}_e(k+j\tau) + \mathbf{u}_e(k+(j-1)\tau)^T \mathbf{R} \mathbf{u}_e(k+(j-1)\tau) + E(k+T_p), \tag{5.52}$$

subject to

$$\bar{\mathbf{x}}_e(k+j\tau) = \mathbf{g}(\mathbf{x}_e(k+(j-1)\tau), \mathbf{u}_e(k+(j-1)\tau)), \tag{5.53a}$$

$$\bar{\mathbf{u}}_e(k+j\tau) = \bar{\mathbf{u}}_e(k+T_c), \ \forall\, j \in \left[\frac{T_c}{\tau}, \frac{T_p}{\tau}\right], \tag{5.53b}$$

$$c(\bar{\mathbf{x}}_e(k+j\tau), \bar{\mathbf{u}}_e(k+(j-1)\tau)) \leq 0, \tag{5.53c}$$

$$\bar{\mathbf{x}}_e(k+T_p) \in \Omega. \tag{5.53d}$$

k denotes the kth time step. τ is the constant prediction sampling time. Condition (5.53d) denotes the terminal constraints. Inequality constraint (5.53c) represents the constraints of $\mathbf{x}_e$ and $\mathbf{u}_e$, which are related to the robot constraints of wheel velocities, and can be calculated from (5.32) or (5.45) with $\mathbf{u}_e$ instead of the terminal feedback control values $\mathbf{u}_e^L$.

To solve the above open-loop optimal control problem, the software *donlp2-intv-dyn* written by P. Spellucci is used. It is a general purpose nonlinear optimizer and can be found at *http://plato.la.asu.edu/donlp2.html*. This optimizer implements a quadratic programming method constrained by a sequential equality with an active set technique. When the linearly dependent gradients of active constraints occur, a fully mixed constrained subproblem is used alternatively. This optimizer also uses following methods: a slightly modified version of the Pantoja-Mayne update for the Hessian of the Lagrangian, variable dual scaling, an improved Armjijo-type stepsize algorithm. Their details can be found in [155, 154].

5.5.3 Delays Compensation

Although NMPC is well studied from the theory side, to apply NMPC in practice meets some challenges. Delays are the main problem in practical applications, which roughly include measurement delays, communication delays and computational delays [45]. With respect to the controller, computational delays are more crucial for applying NMPC. It is well known that on-line solving the nonlinear optimization problem requires some time τ_t^d at each time t, although the faster computer and efficient mathematical methods are used in NMPC. Not taking computational delays into account may significantly decrease the control performance and even lead to instability in the practical applications of NMPC. A simple delay compensation approach proposed in [45] is able to guarantee the control stability which is the same as in the case without delays, which is shown in Figure 5.3 and has following steps:

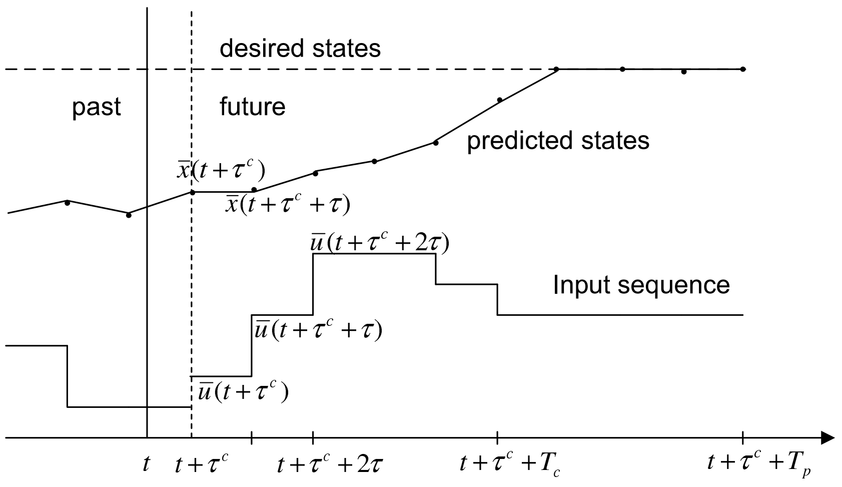


Figure 5.3: Delay compensation in the open-loop optimal control problem

- estimate the maximum computational delay τ^c, i.e. $\tau_t^d \leq \tau^c$,
- predict the system state $\bar{\mathbf{x}}(t+\tau^c)$ with $\mathbf{x}(t)$ and $\mathbf{u}(t)$ at time step t,
- solve the open-loop optimal control problem based on $\bar{\mathbf{x}}(t+\tau^c)$,
- take the first optimal control value $\bar{\mathbf{u}}^*(t+\tau^c)$ as the current control input.

This method is easy to implement and guarantees control stability. The same idea can also be utilized to deal with the measurement delays and communication delays.

5.6 Experimental Results

Real-world experiments with the Attempto soccer robot have been done to test the performance of the designed NMPC schemes. The set-up of experiments is the same as that in Section 4.4. The parameters used in the orthogonal projection-based NMPC were chosen as

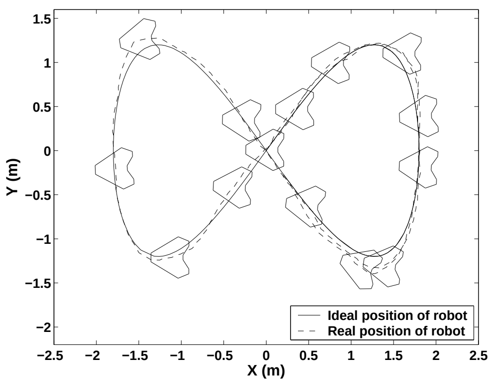
$$\mathbf{Q} = \begin{bmatrix} 0.4 & 0 \\ 0 & 1.0 \end{bmatrix}, \ \mathbf{R} = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.1 \end{bmatrix}, \ \alpha = \beta = 2.0.$$

The parameters in the *Virtual Vehicle*-based NMPC chosen the following values

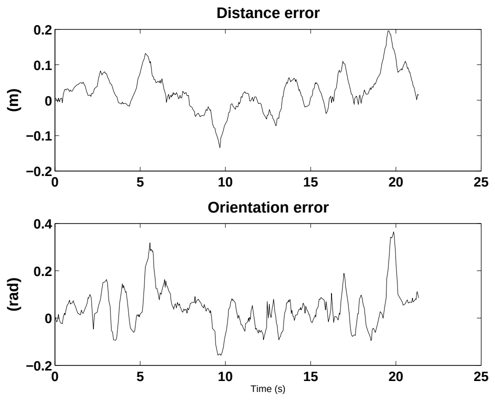
$$\mathbf{Q} = \begin{bmatrix} 1.2 & 0 & 0 \\ 0 & 1.4 & 0 \\ 0 & 0 & 1.4 \end{bmatrix}, \ \mathbf{R} = \begin{bmatrix} 0.08 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}, \ \kappa = \beta = \gamma = 2.0.$$

The control horizon was chosen same as the prediction horizon with $T_P = T_c = 3\tau$, where the prediction sampling time τ was assigned with special values in different experiments.

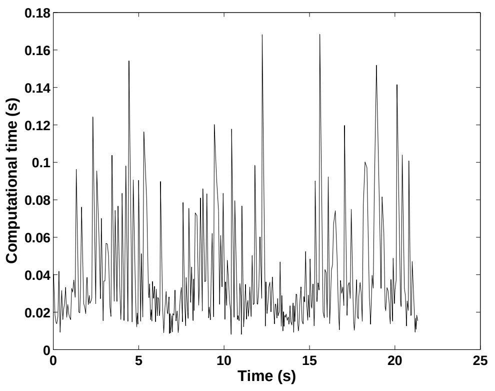
Figures 5.4 and 5.5 show the results with respect to the orthogonal projection-based NMPC. Figures 5.6 and 5.7 illustrate the results with respect to the *Virtual Vehicle*-based NMPC. Comparing the following errors shown in figures 5.4(b), 5.5(b) and 5.6(b) with those shown in Section 4.4, it can be seen that NMPC has similar performance in the robot translation control, but better performance in the robot orientation control. Moreover, the robot traveled paths shown in figures 5.4(a), 5.5(a) and 5.6(a) are smoother than those controlled by the nonlinear control methods described in Chapter 4. Especially around the sharp turning segments of the reference path, NMPC has handled the difficulty at sharp tuning in advance. This smooth control performance can make a great benefit for the robot dribbling control task. In the view of NMPC's formulation, the weighting matrices can be used to specify the control performance, for example, large elements in $\mathbf{Q}$ emphasize the path following errors in the objective function, large elements in $\mathbf{R}$ make the controlled values more important. But designing the weighting matrices has to compromise among different requirements of the control performance. Figures 5.4(d), 5.5(d), 5.6(d) and 5.7(d) imply the wheel velocities are always bounded by the maximum value 1.9 m/s. As the most important issue, computational times of NMPC are shown in figures 5.4(c), 5.5(c) and 5.6(c). The average and maximal values imply the computational time are acceptable by the robot motion control problem. In the fourth experiment, the *Virtual Vehicle*-based NMPC did not achieve good performance. The high value of desired robot orientation, i.e. $\theta_d = \theta_P + 0.5 c_P v_d^2$, makes the orientation tracking more difficult, especially at the sharp turning segments of the reference path. The NMPC scheme requires long computational time with the average value of 0.1081 s and the maximum value of 0.65 s, which results a long time interval between successive control commands and an unsmooth robot trajectory.



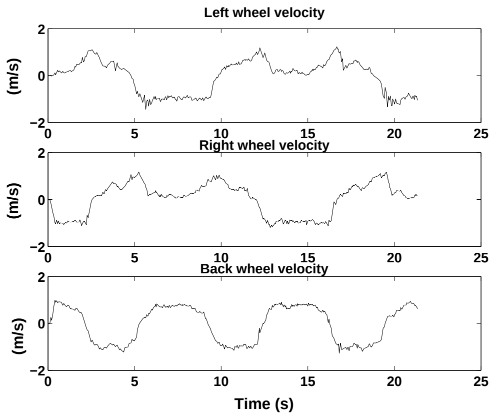
(a) Reference and real paths of the robot



(b) Following errors.

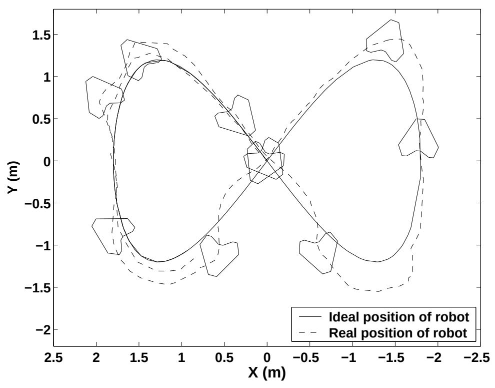


(c) Computational time of NMPC.

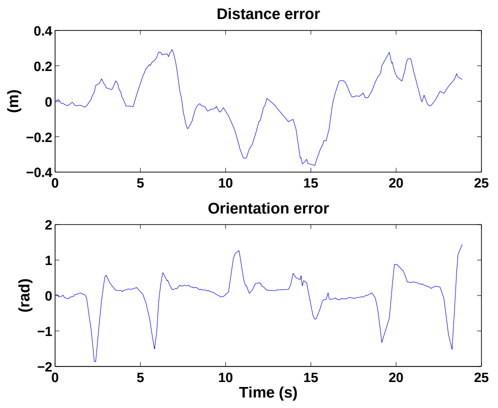


(d) Real wheel velocities.

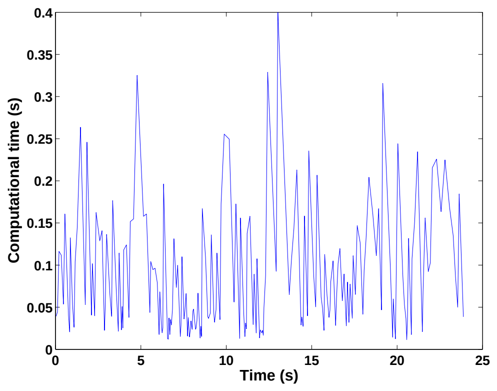
Figure 5.4: Orthogonal projection-based path following control with the constant desired robot orientation of 0 degree. The prediction sampling time τ and the estimated computational delay τ^c were selected as $\tau = \tau^c = 0.2$ s. The maximum and average computational time of NMPC are 0.168 s and 0.035 s, respectively.



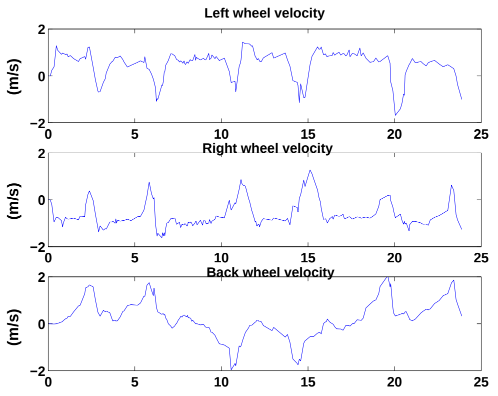
(a) Reference and real paths of the robot



(b) Following errors.

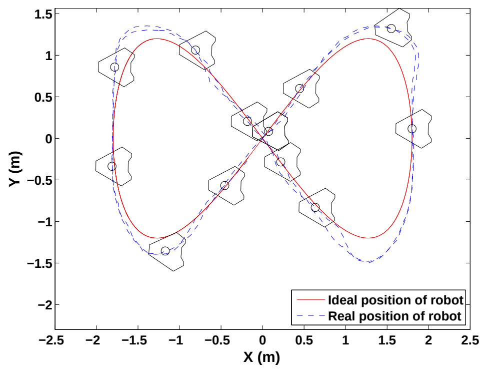


(c) Computational time of NMPC.

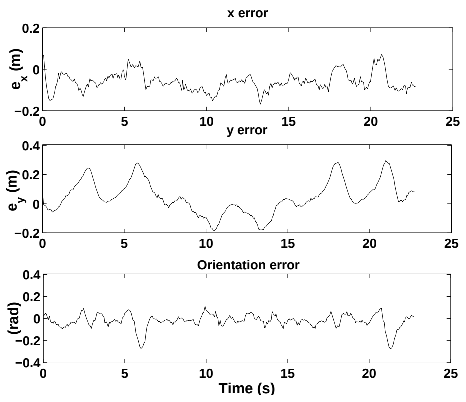


(d) Real wheel velocities.

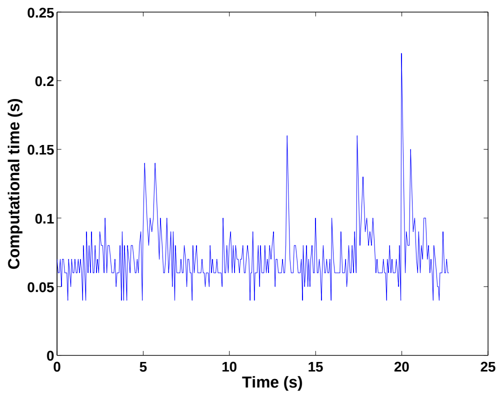
Figure 5.5: Orthogonal projection-based path following control with the desired robot orientation determined by $\theta_d = \theta_P + 0.9c_P v_d^2$. The selected prediction sampling time was $\tau = 0.2$ s and the estimated computational delay τ^c was chosen as $\tau^c = 0.25$ s. The maximum and average computational time of NMPC are 0.399 s and 0.0879 s, respectively.



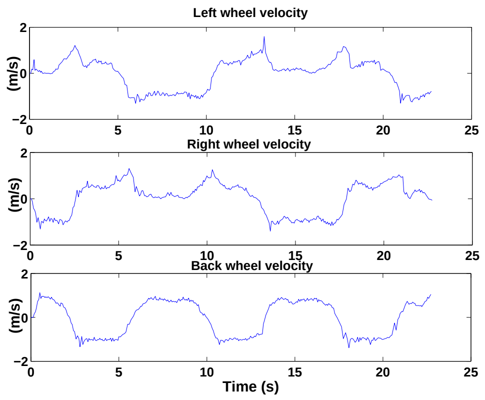
(a) Reference and real paths of the robot



(b) Following errors.

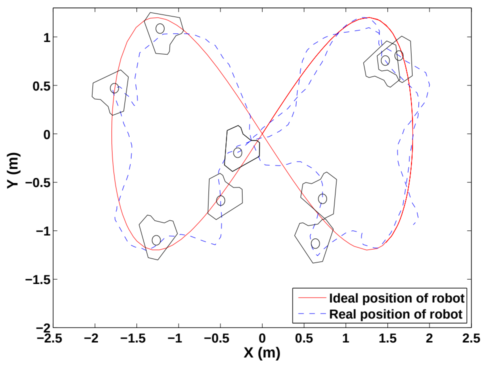


(c) Computational time of NMPC.

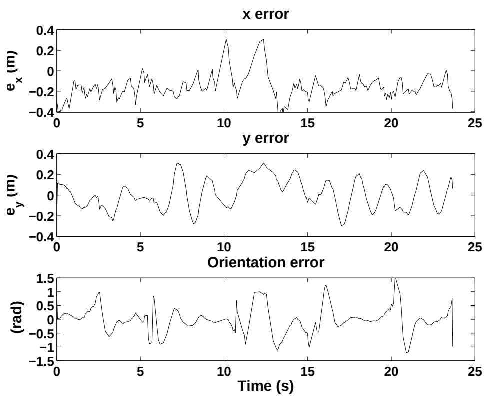


(d) Real wheel velocities.

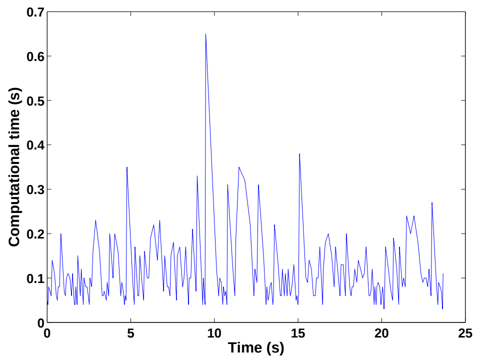
Figure 5.6: *Virtual Vehicle*-based path following control with the constant desired robot orientation of 0 degree. The selected prediction sampling time was $\tau = 0.2$ s and the estimated computational delay τ^c was chosen as $\tau^c = 0.25$ s. The maximum and average computational time of NMPC are 0.22 s and 0.0691 s, respectively.



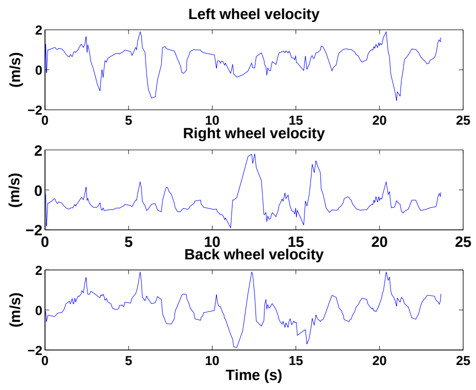
(a) Reference and real paths of the robot



(b) Following errors.



(c) Computational time of NMPC.



(d) Real wheel velocities.

Figure 5.7: *Virtual Vehicle*-based path following control with the desired robot orientation determined by $\theta_d = \theta_P + 0.5c_P v_d^2$. The selected prediction sampling time was $\tau = 0.2$ s and the estimated computational delay τ^c was chosen as $\tau^c = 0.25$ s. The maximum and average computational time of NMPC are 0.65 s and 0.1081 s, respectively.

5.7 Summary

This chapter addresses the successfully used Nonlinear Model Predictive Control of the Attempto soccer robot. The main reasons for utilizing NMPC come from two aspects. The first one is that the robot constraints and the control performance specification can be easily considered in NMPC. The other one is from the concerned control task in this work, which is the robot following a reference path and tracking desired orientations. While the reference path and desired orientations are pre-designed, it is possible to use more of this known information to improve the control performance.

After introducing the mathematical formulation of NMPC, the stability problem and numerical solutions of NMPC have been emphasized in Section 5.3 and 5.4. These two issues are very important in the application of NMPC. The control stability has to be guaranteed with additional effort, because the normal setup of NMPC only considers a finite prediction horizon and can not guarantee closed-loop stability. Moreover, on-line solving the open-loop optimal control problem for a nonlinear system requires long computational time, which is the main block for applying NMPC in fast systems and requires efficient numerical solutions. The main contribution of the work presented in this chapter is that NMPC has been successfully used to solve the path following and orientation tracking problems for a fast moving omnidirectional wheeled robot, where the control tasks are formulated in the NMPC's framework, the closed-loop stability is guaranteed by designing suitable terminal constraints and penalties and computational delays are considered by using a delay compensation method.

All solutions are shown by real-world experiments with the Attempto soccer robot. The more interesting point is that the robot traveled trajectories controlled by NMPC are smoother than those controlled by the nonlinear controllers addressed in Chapter 4. This advantage plays a great role in the dribbling control of the soccer robot, which will be addressed in Chapter 7. As mentioned above, the computational burden is a block of using NMPC with a long prediction horizon. Finding and applying more powerful optimization methods is still a hard task in NMPC. Although the stability problem currently is quite well solved , finding better terminal constraints and penalties to increase the feasibility of the open-loop optimal control problem is still an active topic in the research of NMPC.

Chapter 6

Ball Tracking

In the robot control system described in Chapter 3, the navigation system is in charge of the knowledge of robot position and heading based on the sensor measurements. The software system of the Attempto soccer robot uses an **Image-Processor** process to extract landmarks and objects from images captured by the omnidirectional vision system. An **EnvironmentModel** process in the software system estimates the robot pose on the field with a self-localization algorithm [64], and locates objects in a world coordinate system [67]. Because misreadings and failures in the image processing may occur and objects' velocities can not be obtained by image processing directly, an object tracking algorithm is designed in the **EnvironmentModel** process. The task of object tracking aims to model an object's movement based on a series of past and present measurements of the object's position, in order to decrease the measurement errors of the object's position, obtain the object's current velocity, even predict the object's movement over a short time horizon.

Locating the ball's position and predicting the ball's movement are central for a soccer robot and the cooperation of a robot soccer team. With good knowledge of the ball's status relative to the robot, the robot can initiate a suitable behavior to achieve a good control of the rolling ball, for example, catch the ball, push it around obstacles, and shoot it into the goal. The ball tracking problem in the RoboCup domain is challenging due to the interactions between the robots and the ball. Especially when the ball is dribbled by a robot, the frequent interactions usually result in a highly non-linear movement of the ball, and it is difficult to precisely estimate the interactions. Moreover, the measurement accuracy of the ball's position is also limited by sensors and corresponding signal processing algorithms.

This chapter focuses on tracking a rolling ball when it is consecutively pushed by an Attempto soccer robot [101]. After a short overview of the related work, two filter techniques, the Kalman filter and the H_∞ filter, are addressed in sections

6.2 and 6.3, respectively. Considering that the assumptions of the process noise and measurement noise are hardly satisfied in the dribbling process, the H_∞ filter was successfully implemented in ball tracking with the Attempto soccer robot, which is presented in Section 6.4. Taking the Kalman filter as a benchmark, the comparison of the experimental results with the Kalman filter and the H_∞ filter are discussed in Section 6.5.

6.1 Related Work

The Kalman filter is the most widely used technology to estimate the ball's position and velocity in the RoboCup domain [12, 49, 52, 141, 126], where the ball is assumed to have a linear movement and the variation of ball's velocity is modeled as random noise. The Kalman filter provides efficient and convenient minimum-mean-square-error solutions for the state estimation problem, considering that both the process noise and the measurement noise of the target system are assumed as Gaussian with known statistical properties. When the ball's movement is tracked in polar coordinates [90] or modeled with nonlinear dynamics incorporating the retardation of a ball on the carpet [80], the extended Kalman filters can be used for the ball tracking problem. Besides a single filter, multiple model filters based on Kalman filters revealed better performance in some applications. For example, the Interacting Multiple Model (IMM) algorithm utilizes a Kalman filter for each mode of the target's movement model [66], the Multiple Hypothesis Tracking (MHT) algorithm keeps a set of object hypotheses, each hypothesis corresponding to a Kalman filter describes a unique real object [146]. However, in practical situations, the noise of the target system and the measurement usually do not satisfy the Gaussian assumption, and the noise statistics is usually not available.

To avoid the Gaussian assumption and estimate the states of a nonlinear process, particle filters [91, 123] and a predictive model based method [94] have been applied in object tracking in the RoboCup domain. Although the sample-based representation makes particle filters more robust and the updated parameters enable the predictive model to react to jerky changes of the ball's movement, the increased memory consumption and computational complexity make these methods inefficient for higher-dimensional estimation problems.

Motivated by the possibilities to avoid assumed statistical properties of noises and simultaneously reduce the computational cost, a robust H_∞ filter was implemented for the Attempto soccer robots to track a rolling ball during the dribbling process. The H_∞ filter does not require a priori knowledge of the noise statistics. It only assumes that the noise signals have finite energy. At the same time, the proposed H_∞ filter has similar recursive equations to those of the Kalman filter, and thus it inherits the efficiency and accuracy of Kalman filters.

6.2 Kalman Filter

The Kalman filter is used to estimate the state $\mathbf{x} \in \mathbb{R}^n$ of the following general linear discrete-time process,

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k,$$

with a measurement $\mathbf{y} \in \mathbb{R}^m$ given by

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k,$$

where k is the index of the time step. The known matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ have corresponding dimensions, and may remain constant or change at each time step. Variables $\mathbf{w}$ and $\mathbf{v}$ represent the process noise and the measurement noise, respectively. They are assumed to be independent of each other and have Gaussian probability distributions with zero means, i.e.

$$p(\mathbf{w}) \sim N(0, \mathbf{Q}). \tag{6.1}$$

$$p(\mathbf{v}) \sim N(0, \mathbf{R}). \tag{6.2}$$

The *process noise covariance* $\mathbf{Q}$ and *measurement noise covariance* $\mathbf{R}$ can be different at each time step, but they are mostly assumed to be constant in practice.

With definitions of the *a priori* state estimate $\hat{\mathbf{x}}_k^- \in \mathbb{R}^n$ at time step k based on the knowledge of the process prior to time step k, and the *a posteriori* state estimate $\hat{\mathbf{x}}_k \in \mathbb{R}^n$ at time step k incorporating all the knowledge of the process including the measurement $\mathbf{y}_k$, the goal of the Kalman filter is to minimize the covariance of the *a posteriori* estimation error,

$$\mathbf{P}_k = E[\mathbf{e}_k \mathbf{e}_k^T], \tag{6.3}$$

where $\mathbf{e}_k = x_k - \hat{\mathbf{x}}_k$ is the *a posteriori* estimation error.

The solution of the Kalman filter involves two steps. The prediction step aims to obtain the *a priori* estimations of the state and the error covariance,

$$\begin{aligned} \hat{\mathbf{x}}_k^- &= \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_{k-1}, \\ \mathbf{P}_k^- &= \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}, \end{aligned}$$

where $\mathbf{P}_k^-$ is the *a priori* estimation error covariance defined as $\mathbf{P}_k^- = E[\mathbf{e}_k^- \mathbf{e}_k^{-T}]$. $\mathbf{e}_k^- = x_k - \hat{\mathbf{x}}_k^-$ is the *a priori* estimation error.

The correction step is responsible for getting improved *a posteriori* estimations with a weighted difference between the measurement $\mathbf{y}_k$ and a measurement prediction $\mathbf{C}\hat{\mathbf{x}}_k^-$,

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{C}^T (\mathbf{C}\mathbf{P}_k^- \mathbf{C}^T + \mathbf{R})^{-1}, \tag{6.4}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-), \tag{6.5}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{C})\mathbf{P}_k^-. \tag{6.6}$$

Matrix $\mathbf{K}$ is called a *gain* or *blending factor* and results from minimizing (6.3). Normally, the *gain* is computed by substituting equation (6.5) into (6.3) and setting the derivation with respect to $\mathbf{K}$ equal to zero. A more detailed derivation of $\mathbf{K}$ can be found in [106].

As long as the noises satisfy the assumptions (6.1) and (6.2), the *a posteriori* state estimate $\hat{\mathbf{x}}_k$ and the *a posteriori* estimation error covariance $\mathbf{P}_k$ reflect the mean and variance of the state distribution, respectively:

$$E[\mathbf{x}_k] = \hat{\mathbf{x}}_k,$$

$$E[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T] = \mathbf{P}_k.$$

6.3 H_∞ Filter

The optimality of the Kalman filter relies on the knowledge of the statistical properties of the noises $\mathbf{w}$ and $\mathbf{v}$. Although the Gaussian assumption can be approximated and suitable covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ can be chosen by trial and error, the resulting Kalman filter can not guarantee to achieve a certain level of performance. Unlike the Kalman filter obtaining the minimum variance of the estimation error, the H_∞ filter obtains the minimal effect of the worst noise on the estimation error. The H_∞ filter is robust against the noise and gives an upper boundary on the estimation errors based on the assumption of a finite disturbance energy no matter what the noise distributions are.

Consider the following linear system:

$$\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{w}_k,$$
$$\mathbf{y}_k = \mathbf{C}_k\mathbf{x}_k + \mathbf{v}_k,$$

where $\mathbf{x}_k \in \mathbb{R}^n$, $\mathbf{w}_k \in \mathbb{R}^m$, $y_k \in \mathbb{R}^p$ and $\mathbf{v}_k \in \mathbb{R}^p$. $\mathbf{A}_k, \mathbf{B}_k$ and $\mathbf{C}_k$ are matrices with appropriate dimensions. $(\mathbf{A}_k, \mathbf{B}_k)$ is controllable and $(\mathbf{C}_k, \mathbf{A}_k)$ is detectable. Compared to the Kalman filter aiming to estimate the system state $\mathbf{x}_k$, the H_∞ filter concerns a linear combination of $\mathbf{x}_k$:

$$\mathbf{z}_k = \mathbf{L}_k\mathbf{x}_k.$$

The output matrix $\mathbf{L}_k$ is selected by the user according to different applications. The H_∞ filter computes the estimated state $\hat{\mathbf{z}}_k$ based on the measurement $\mathbf{y}_k$, where $\mathbf{y}_k = \{y_k, 0 \le k \le N\}$, and evaluates the estimation error by a performance measurement J which can be regarded as an energy gain:

$$J = \frac{\sum_{k=0}^{N} \|\mathbf{z}_k - \hat{\mathbf{z}}_k\|^2_{\mathbf{Q}_k}}{\|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|^2_{\mathbf{p}_0^{-1}} + \sum_{k=0}^{N} \left(\|\mathbf{w}_k\|^2_{\mathbf{W}_k^{-1}} + \|\mathbf{v}_k\|^2_{\mathbf{V}_k^{-1}}\right)}.$$

Herein N is the size of the measurement history. $\mathbf{Q}_k, \mathbf{p}_0, \mathbf{W}_k, \mathbf{V}_k$ are the weighting matrices for the estimation error, the initial condition, the process noise and the measurement noise, respectively. Moreover, $\mathbf{Q}_k \ge 0$, $\mathbf{p}_0^{-1} > 0$, $\mathbf{W}_k > 0$, $\mathbf{V}_k > 0$ and $((\mathbf{x}_0 - \hat{\mathbf{x}}_0), \mathbf{w}_k, \mathbf{v}_k) \neq 0$. The notation $\|\mathbf{x}_k\|^2_{\mathbf{Q}_k}$ is defined as $\|\mathbf{x}_k\|^2_{\mathbf{Q}_k} = \mathbf{x}_k^T \mathbf{Q}_k \mathbf{x}_k$. The denominator of J can be considered as the energy of the unknown noises, and the numerator is the energy of the estimation error. The H_∞ filter aims to provide a uniformly small estimation error $\mathbf{e}_k = \mathbf{z}_k - \hat{\mathbf{z}}_k$ for any $\mathbf{w}_k, \mathbf{v}_k \in \mathcal{L}_2$ and $\mathbf{x}_0 \in \mathbb{R}^n$, such that the energy gain J is bounded by a prescribed value:

$$\sup J < 1/\gamma$$

where $\sup$ denotes the supremum and $1/\gamma$ is the noise attenuation level with $\gamma > 0$. This formulation leads to the robustness of the H_∞ filter, because the estimation energy gain is limited by $1/\gamma$ no matter what the bounded energy noises are.

To solve this optimal estimation $\hat{\mathbf{z}}$ due to the bounded energy gain J, the H_∞ filter can be interpreted as a *minimax* problem [150]

$$\begin{aligned}\min_{\hat{\mathbf{z}}_k} \max_{(\mathbf{w}_k, \mathbf{v}_k, \mathbf{x}_0)} J = & -\frac{1}{2\gamma} \|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|^2_{\mathbf{p}_0^{-1}} + \\ & \frac{1}{2} \sum_{k=0}^{N} \left[\|\mathbf{z}_k - \hat{\mathbf{z}}_k\|^2_{\mathbf{Q}_k} - \frac{1}{\gamma} \left(\|\mathbf{w}_k\|^2_{\mathbf{W}_k^{-1}} + \|\mathbf{v}_k\|^2_{\mathbf{V}_k^{-1}} \right) \right]\end{aligned}$$

where the estimation value $\hat{\mathbf{z}}_k$ plays against the bounded energy noises $\mathbf{w}_k$ and $\mathbf{v}_k$. “min” stands for minimization and “max” denotes maximization.

Many strategies have been proposed for solving this *minimax* problem [119, 62]. Reference [150] proposed a linear quadratic game approach, which gave a complete solution to this *minimax* problem without checking the positive definiteness and inertia of the Riccati difference equations for every step. This approach is implemented through recursive updating the filter gain $\mathbf{H}_k$, the solution $\mathbf{P}_k$ of the Riccati difference equations, and the state estimation $\hat{\mathbf{x}}_k$ with the following updating equations:

$$
\begin{aligned}
\bar{\mathbf{Q}}_k &= \mathbf{L}_k^T \mathbf{Q}_k \mathbf{L}_k, \\
\mathbf{H}_k &= \mathbf{A}_k \mathbf{P}_k \left(I - \gamma \bar{\mathbf{Q}}_k \mathbf{P}_k + \mathbf{C}_k^T \mathbf{V}_k^{-1} \mathbf{C}_k \mathbf{P}_k \right)^{-1} \mathbf{C}_k^T \mathbf{V}_k^{-1}, \\
\hat{\mathbf{x}}_k &= \mathbf{A}_k \hat{\mathbf{x}}_{k-1} + \mathbf{H}_k \left(\mathbf{y}_k - \mathbf{C}_k \mathbf{A}_k \hat{\mathbf{x}}_{k-1} \right), \\
\hat{\mathbf{z}}_k &= \mathbf{L}_k \hat{\mathbf{x}}_k, \\
\mathbf{P}_{k+1} &= \mathbf{A}_k \mathbf{P}_k \left(I - \gamma \bar{\mathbf{Q}}_k \mathbf{P}_k + \mathbf{C}_k^T \mathbf{V}_k^{-1} \mathbf{C}_k \mathbf{P}_k \right)^{-1} \mathbf{A}_k^T + \mathbf{B}_k \mathbf{W}_k \mathbf{B}_k^T,
\end{aligned}
$$

where $\mathbf{P}_0 = \mathbf{p}_0$ and $\mathbf{P}_k > 0$. I is the identity matrix with corresponding dimensions.

Apparently, these recursive equations have similar forms to those of the classic Kalman filter. Although the statistics of noises $\mathbf{w}_k$ and $\mathbf{v}_k$ are not required in the H_∞ filter, tuning the weight matrices $\mathbf{Q}_k, \mathbf{p}_0, \mathbf{W}_k, \mathbf{V}_k$ should be done carefully, because these values determine the estimation error in the performance criterion. The weight matrices $\mathbf{W}_k, \mathbf{V}_k$ can be chosen according to the experience with the noise. For example, if the noise $\mathbf{w}$ is known to be smaller than the noise $\mathbf{v}$, $\mathbf{W}_k$ should have smaller elements than those of $\mathbf{V}_k$ and vice versa. $\mathbf{p}_0$ is based on the initial estimation error. If the initial estimation $\hat{\mathbf{Z}}_0$ has higher creditability, $\mathbf{p}_0$ should be small. Similarly, if estimations of some elements in the state have received more attention, or some elements have bigger magnitude in their physical definition, the corresponding elements in the matrix $\mathbf{Q}_k$ can be set larger than others. The performance criterion γ is hoped to be as large as possible. Yet too large γ may make some eigenvalues of the matrix P larger than one, which makes the H_∞ filter's mathematical deviation become invalid. Therefore, the estimation error of the H_∞ filter can not be arbitrarily small.

6.4 Implementation

To implement the H_∞ filter, the ball's movement is modeled by the following linear discrete system,

$$
\begin{aligned}
\dot{p}_{k+1} &= \dot{p}_k + \ddot{p}_k T, \\
p_{k+1} &= p_k + \dot{p}_k T + \frac{1}{2} \ddot{p}_k T^2,
\end{aligned}
$$

where p is the position of the ball, while $\dot{p}$ and $\ddot{p}$ are the velocity and acceleration of the ball, respectively. T is the sampling interval and k is the index of the sampling time. Defining a state vector $\mathbf{x}_k$ consisting of the position and velocity as $\mathbf{x}_k = [p_k \ ; \ \dot{p}_k]$, and taking the ball's position as the measurement value, the ball's movement is described by:

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} u_k, \tag{6.7}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}_k, \tag{6.8}$$

where the system input u equals the acceleration $\ddot{p}$, which is completely determined by the friction of the ground and the pushing operation from the robot. In practice situation, equation (6.7) can not give the precise state values because of the noise due to the rugged carpet ground. The precise output values can not be obtained from equation (6.8), since measurement noise decreases the reliability of the measured data. Therefore, the ball's model should take process noise $\mathbf{w}$ and measurement noise $\mathbf{v}$ into account,

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} u_k + \mathbf{w}_k,$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}_k + \mathbf{v}_k.$$

Several effects, such as the friction of the ground, the moment when the robot collides with the ball and the corresponding effect of the collision on the ball's movement can not be obtained exactly, so the system input u is not available when the robot is dribbling the ball. However, u can be taken as additional process noise and unified with the process noise $\mathbf{w}$. Then a more realistic system model is deduced as

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} \mathbf{w}_k, \tag{6.9}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}_k + \mathbf{v}_k. \tag{6.10}$$

In the context of the H_∞ filter, $\mathbf{L}_k$ is specified as an identity matrix in this case, because the ball's location and velocity are all required to be estimated, i.e.

$$\mathbf{z}_k = \mathbf{x}_k.$$

6.5 Experimental Results

The ball's observation data come from the omnidirectional vision system of the Attempto soccer robots. Pointing up towards a hyperbolic mirror mounted on the top of a robot, an AVT Marlin F-046C color camera can capture surrounding images of a robot up to 50 times per second. After obtaining these color images, a color calibration process maps the colors to different classes in the RoboCup

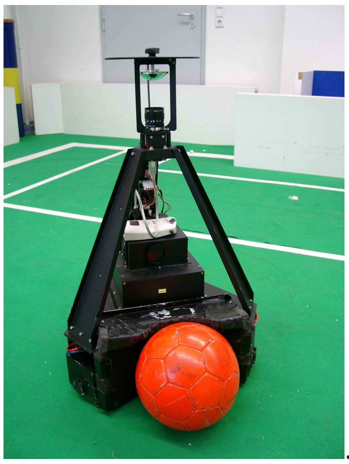


Figure 6.1: An Attempto soccer robot with an orange match ball.

domain and extracts the landmarks and objects from the images. Then, a distance calibration process transfers pixel positions in the image into the real world coordinate system [67].

While the object detection algorithm always outputs the ball's relative position to the robot, the ball's relative position and velocity with respect to the robot coordinate system can be estimated directly by using the ball's observation values. When the ball's absolute position and velocity are required, the estimated ball's relative values can be transformed into the world coordinate system using the robot orientation values. Figure 6.2 illustrates the ball's position in the robot coordinate system.

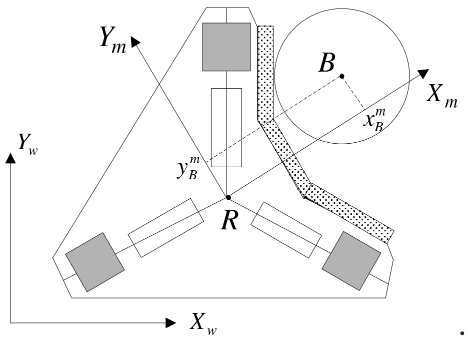


Figure 6.2: Ball's relative position (x_B^m, y_B^m) in the robot coordinate system.

To prove the feasibility and the robustness of the H_∞ filter in the ball tracking problem, two real experiments with an Attempto soccer robot were performed in the robot laboratory. Figure 6.1 shows an Attempto soccer robot with an orange ball. Considering the limited size of the robot laboratory, the soccer robot was controlled to dribble the ball along a linear path and a circular path. The following commands were sent to the robot in these two experiments, respectively:

$$a)\ \dot{x}_R^m = 1.5m/s,\ \dot{y}_R^m = 0,\ \omega = 0;$$
$$b)\ \dot{x}_R^m = 0.8m/s,\ \dot{y}_R^m = 0,\ \omega = 0.5rad/s.$$

In case a), the robot drives linearly with a speed of 1.5 m/. In case b), it forwards along a circle with a speed of 0.8 m/s. The ball did not slide away from the robot during the whole dribbling process because of the consecutive collisions with the robot. At every sampling time, two H_∞ filters estimated the x and y components of the ball's relative position and velocity with respect to the robot coordinate frame. The noise attenuation level and weight matrices for estimating the x and y components of the ball's movement in both cases were chosen as follows:

$$\gamma^x = 2.0,\ \mathbf{p}_0^x = \begin{bmatrix} 30 & 0.004 \\ 30 & 2 \end{bmatrix},\ \mathbf{Q}_k^x = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix},\ \mathbf{W}_k^x = 1,\ \mathbf{V}_k^x = 10\ ;$$

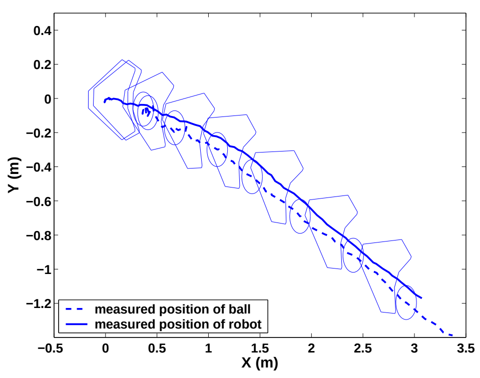
$$\gamma^y = 1.5,\ \mathbf{p}_0^y = \begin{bmatrix} 10 & 0.05 \\ 30 & 0.8 \end{bmatrix},\ \mathbf{Q}_k^y = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix},\ \mathbf{W}_k^y = 10,\ \mathbf{V}_k^y = 1\ .$$

To evaluate the performance of the H_∞ filter, a Kalman filter with assumed noise variance was selected as a benchmark to estimate the ball's position and velocity with the same observation values. The initial estimation error covariance matrices $\mathbf{P}_o$ and the probability distributions of the process noise and the measurement noise were chosen by trial and error. The following parameters gave the optimal estimations:

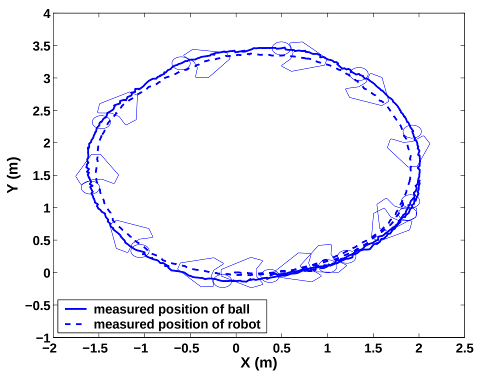
$$\mathbf{p}_0^x = \begin{bmatrix} 0.01 & 0.0001 \\ 0.003 & 0.005 \end{bmatrix},\ p(\mathbf{w}^x) \sim N(0, 0.01),\ p(\mathbf{v}^x) \sim N(0, 0.0001);$$

$$\mathbf{p}_0^y = \begin{bmatrix} 0.01 & 0.0001 \\ 0.01 & 0.005 \end{bmatrix},\ p(\mathbf{w}^y) \sim N(0, 1),\ p(\mathbf{v}^y) \sim N(0, 0.0001).$$

Figure 6.3(a) show the traveled paths of the robot and the ball in these two experiments. The results illustrated in figures 6.4 and 6.5 show that the H_∞ filter eliminated the high frequency components of the measurement and estimated the ball's relative positions and velocities successfully. The estimated ball's relative positions shown in figures 6.4(b) and 6.5(b) imply that the ball did not slide away from the robot, while the maximum position along the $\mathbf{Y}_m$ direction is far from the boundary value 0.15 m. Figures 6.4(a), 6.4(b), 6.5(a) and 6.5(b) show that the estimated ball's relative positions from the H_∞ filter are slightly better than those from the Kalman filter. Moreover, figures 6.4(c), 6.4(d), 6.5(c) and 6.5(d) show that the H_∞ filter gave smoother estimations of the ball's relative velocities than the Kalman filter.

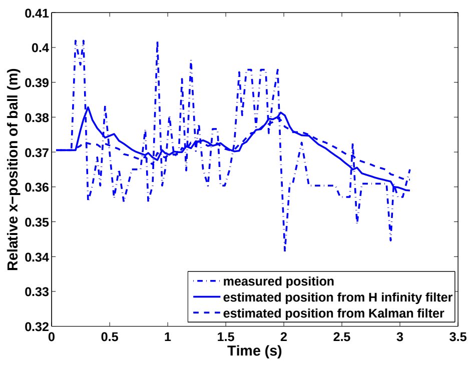


(a) Linear paths of robot and ball.

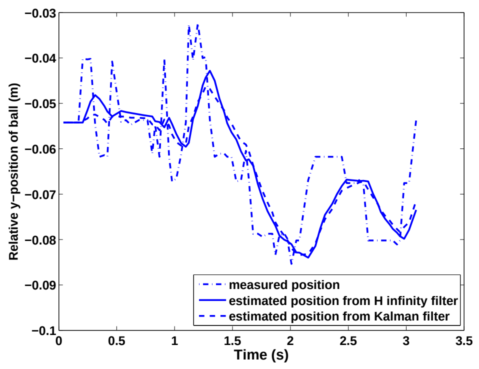


(b) Circular paths of robot and ball.

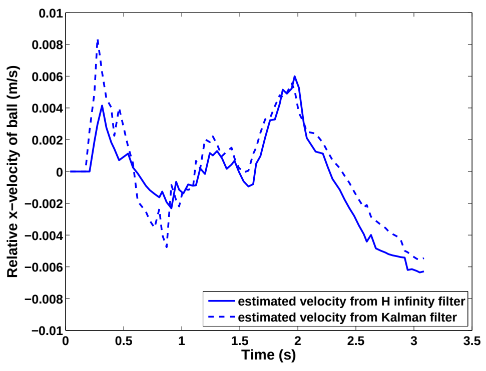
Figure 6.3: Traveled paths of the robot and the ball in the dribbling experiments.



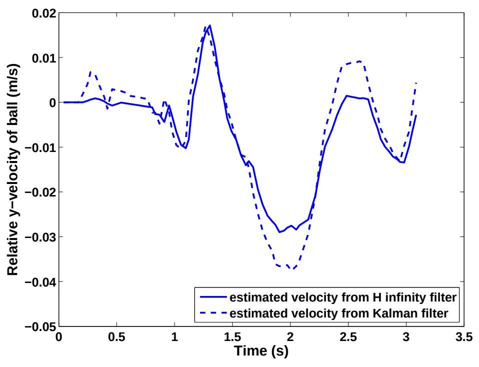
(a) Relative x-positions of ball.



(b) Relative y-positions of ball.

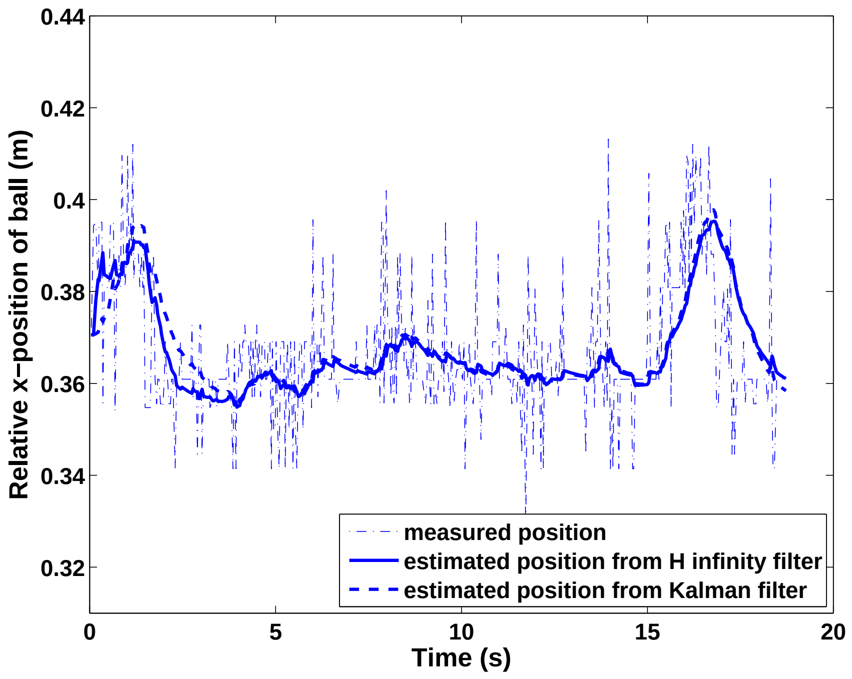


(c) Relative x-velocities of ball.

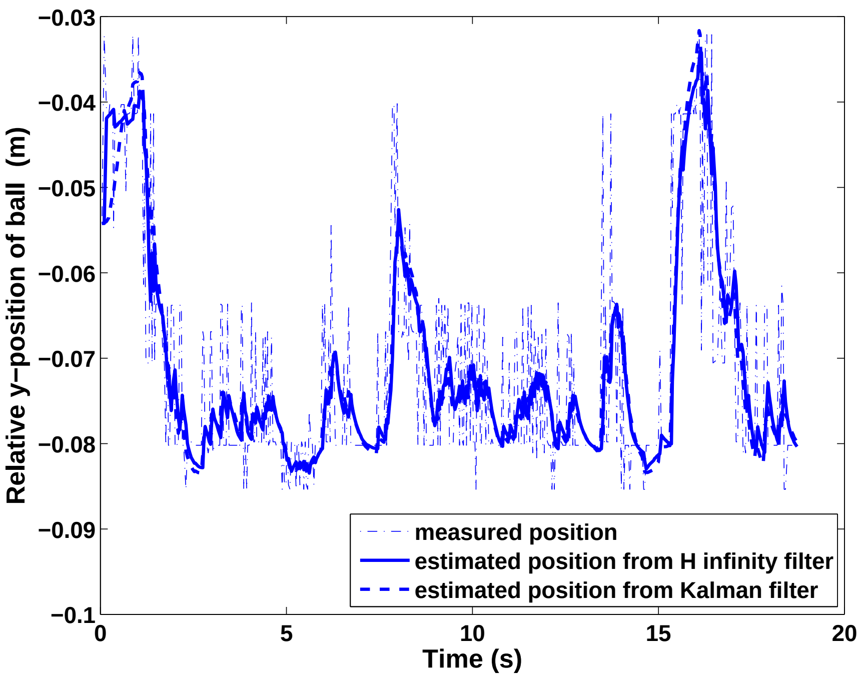


(d) Relative y-velocities of ball.

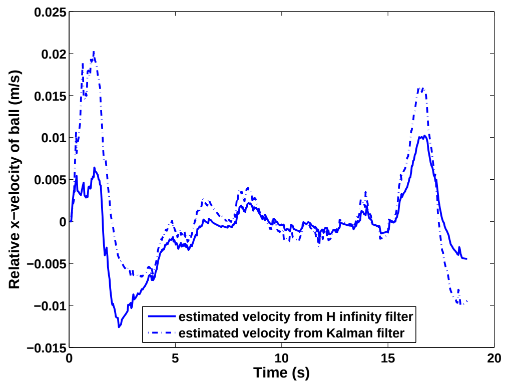
Figure 6.4: Relative ball's positions and velocities with respect to the robot coordinate system when robot moves along a linear path.



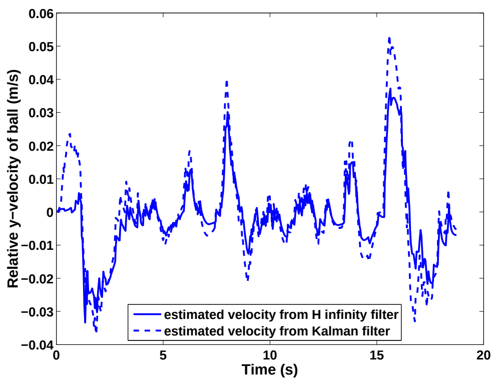
(a) Relative x-positions of ball.



(b) Relative y-positions of ball.



(c) Relative x-velocities of ball.



(d) Relative y-velocities of ball.

Figure 6.5: Relative ball's positions and velocities with respect to the robot coordinate system when the robot moves along a circle.

6.6 Summary

In this chapter, a robust H_∞ filter is adopted to estimate the ball's relative position and velocity when the ball is dibbled by a soccer robot. Unlike the Kalman filter, which relies on suitable assumptions of noise variances, the H_∞ filter does not require a priori knowledge about the statistical properties of the process noise and the measurement noise, but only depends on the assumption of finite noise power. The Kalman filter aims to minimize the expected estimation error covariance and yields maximum-likelihood estimations, while the H_∞ filter minimizes the worst possible effects of noise on the estimation errors. This guarantees that, if noise is small in energy, the estimation error will be as small as possible. The recursive equations of the H_∞ filter have similar forms to those of the Kalman filter, thus the H_∞ has similarly low computational cost and is feasible for real time estimation problems.

In two real-world experiments, where the ball was pushed consecutively by an omnidirectional soccer robot, the performance of the H_∞ filter was evaluated by comparing the estimation values to those of the Kalman filter. The results of the estimated ball's relative positions and velocities show that the H_∞ filter eliminates the high frequency noise components of the measurements and estimates the ball's position and velocity robustly during the pushing process. Although the H_∞ filter involves regulating some weighting matrices, the real world experimental results show that it has better performance than a Kalman filter and the independence of noise statistics makes the H_∞ filter more robust.

Chapter 7

Dribbling Control

For a soccer robot, ball control is one of the most important and essential skills. Concerning offensive and defensive tactics, controlling the ball consists of three tasks. The first task denotes ball capturing, where a robot is able to catch the ball whenever the ball is moving or resting. The second task refers to ball dribbling, which involves the maneuvering of the ball through consecutive and short contacts of a robot in a n environment with dynamic obstacles. The third one is ball keeping, which enables a soccer robot to prevent the ball from being stolen by the opponents.

Compared to other tasks, ball dribbling is more important from the offensive standpoint, because a soccer robot has to control the ball and shoot a goal after catching the ball. Moreover, dribbling control is more challenging than the normal motion control of an autonomous robot, while dribbling control has to consider the ball's movement in steering the robot movement. Although efficient dribbling mechanisms help a soccer robot to achieve a good dribbling skill, design and execution of appropriate dribbling strategies have attracted attention in the RoboCup robot soccer teams.

The dribbling process is actually a consecutive impact process of high frequency and low magnitude. Because of the difficulty of determining the impact time, the contact position, and corresponding impact influence, many RoboCup teams either use a simple model to approximate the interaction, or let the soccer robot learn the dribbling skills where the interaction is regarded as a black-box system.

In dribbling learning, artificial neural networks (ANN) have been adopted by many RoboCup teams [26, 63, 117], because they are able to approximate an arbitrary function by learning from observed data. Normally, the inputs of an ANN refer to the environment information, such as the positions of the robot and the ball, the positions of opponents, the direction of the goal, and so on. The outputs of the ANN are the desired robot actions, for example, the desired robot velocities and accelerations. The selection of a suitable structured ANN given the training data is not straightforward. Especially using a real robot, the training process has to start with positioning the ball and the robot at arbitrary positions of the play field. Then, all measurements are recorded, which is followed by training an ANN based on suitable learning algorithms. The whole process generally needs a very long time to obtain a trained ANN with optimal performance. Therefore, most soccer robot teams use adequate simulators to achieve a considerable faster training time. Simulators have the advantage of not being constrained to time limitations, such as to tune the parameters of the ANN and the learning algorithms for an optimal performance. Although the learning procedure benefits from simulators, many problems occur in real experiments when the trained ANN is used on real robots. It is because the trained ANN is constrained to a small subset of robot behaviors. For example, the interactions between a robot and the ball are difficult to describe in mathematical terms when the robot dribbles the ball. As a result, the learning method directly operating on real robots with efficient learning algorithms became attractive recently. Concerning learning algorithms, reinforcement learning became popular for learning of soccer robot behaviors, as it dose not need the correct input/output pairs, but only the information about the behaviors' success or failure [137, 76, 50, 60]. As a successful application, an off-line neural fitted Q iteration scheme based reinforcement learning approach has been proposed in [137] to learn the dribbling on a real soccer robot. This learning method allows the application of advanced supervised learning methods, and has a faster convergence than the on-line gradient descent methods.

Learning the dribbling of a soccer robot avoids building complex physical models, but the learning process requires a long time and results in high computational cost, especially when there is a large number of parameters to optimize. Moreover, the collection of training data is difficult to be completed. In a new environment, new training data may result in a new learning process. Therefore, an analytical dribbling control method is necessary to decrease the time spent on designing a dribbling controller. Damas *et al.* addressed some analytical constraints for a nonholonomic soccer robot dribbling a rolling ball in [28]. Based on a simple description of the interaction between the robot and the ball, these constraints are used to avoid loosing the ball by limiting the robot translation and rotation velocities. Another analytical method is presented in [69] with respect to an omnidirectional soccer robot. It approximates the interaction by a spring kinematic model, and assumes that the ball does not leave the robot but always compresses the spring. According to this assumption, the robot is controlled to track the desired poses, which are computed from the ball's desired trajectory based on the suitable values of a weight factor and a damping ratio.

To avoid the long learning process, this work focuses on designing an analytical dribbling control method for the Attempto soccer robot. Inspired by [28], a constraint of robot movement in the dribbling process for keeping the ball is deduced by analyzing the force exerted on the ball. Without modeling the interaction between a robot and the ball during the dribbling process, the analysis of the relative movement of the ball with respect to the robot results in the dribbling control method. It accomplishes the dribbling task by introducing a reference point to follow a given path and keeping the ball near this point simultaneously [97, 96, 100].

7.1 Dribbling Mechanisms

The dribbling system of a soccer robot is composed of dribblers, which are built of special materials to increase the robot's ability of controlling the ball. The main contribution of dribblers is to exert a certain amount of force onto the ball. The force can not only give a backwards spin to the ball such that the ball can move back when it loses contact with the robot, but also prevent the ball from sliding away from the robot when the robot rotates quickly.

Designing dribbling mechanisms of soccer robots in the RoboCup Middle Size League has to obey the following rules [1]:

- *During a game the ball must not enter the convex hull of a robot by more than a third of its diameter except when the robot is stopping the ball;*
- *Forces exerted onto the ball that hinder the ball from rotating in its natural direction of rotation are allowed for no more than four seconds and a maximal distance of movement of one meter.*

Considering many hardware challenges in the RoboCup Middle Size League, most teams pay more attention to improving and executing dribbling strategies. They mostly adopt simple and flexible dribbling systems. Passive dribblers are widely adopted by the Middle Size League teams due to the simple mechanisms. As illustrated in figures 7.1(c), 7.1(d), 7.0(e) and 7.0(f), passive dribblers have no actuators such as motors and gears to be controlled actively, but influence the ball's movement by special structures and materials. In order to improve the dribbling capability, some teams designed active dribblers as shown in Figures 7.1(a) and 7.1(b), usually represented by wheels which are controlled by DC motors to influence the ball's rotation.

Consisting of dribblers built from materials with high friction properties and good damping qualities, the popular dribbling system is designed with a concave front and a top component. The concave shape either stems from the concave front of the robot base as shown in figures 7.1(a), 7.1(b), 7.0(e) and 7.0(f), or is formed with some separated dribblers, which are illustrated in figures 7.1(c) and

7.1(d). The main benefit of the concave shape is that the ball can be easily forced toward the center of the robot front. The contribution of the top component is to exert pressure onto the ball from the above, such as to keep the ball from rolling away. The top component can consist of active wheels, plastic bars or a rubber string, as shown in figures 7.1(c) - 7.0(f), respectively.

The dribbler system of the Attempto soccer robot is shown in Figure 7.0(f). There are three spongy blocks attached to the concave front to damp the collisions of the ball and to prevent the ball from sliding away from the robot. A rubber foam pad is assembled at a higher position, which exerts pressure onto the ball and keep it from leaving the robot along the lateral and longitudinal directions. The advantage of this dribbling system is that the bigger facing size enables the robot to easily capture the ball.

7.2 Dribbling Analysis

In a dribbling task, the main challenge is that the ball should be kept and be pushed by a robot when the robot moves and passes obstacles. Dribbling control needs to consider not only the robot's movement, but also the ball's movement. Therefore, it is necessary to analyze the relative movement between the robot and the ball. When the ball is considered as a mass point B located at the sphere center, the relationship between the ball's accelerations observed in the world coordinate system and the robot coordinate system is described as follows:

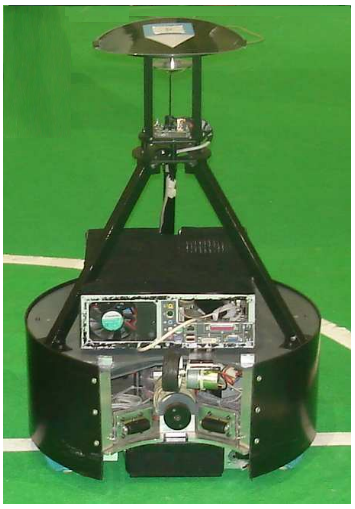
$$\mathbf{a}_B = \mathbf{a}_R + \mathbf{a}_B^m + 2\boldsymbol{\omega} \times \mathbf{v}_B^m + \dot{\boldsymbol{\omega}} \times \mathbf{r}_B^m + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}_B^m). \tag{7.1}$$

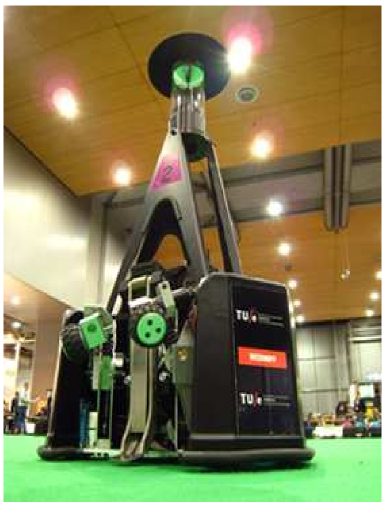
$\mathbf{a}_R$ denotes the robot's translation acceleration observed in the world coordinate system. $\mathbf{a}_B$ and $\mathbf{a}_B^m$ are the ball's accelerations observed in the world and robot coordinate systems, respectively. ω and $\dot{\omega}$ are the robot rotation velocity and the corresponding rotation acceleration. $\mathbf{v}_B^m$ and $\mathbf{r}_B^m$ are the ball's velocity and position observed in the robot coordinate system, respectively. In (7.1), the term $2\boldsymbol{\omega} \times \mathbf{v}_B^m$ is called Coriolis' acceleration; the term $\dot{\boldsymbol{\omega}} \times \mathbf{r}_B^m$ is due to the robot rotation acceleration; the term $\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}_B^m)$ is called centripetal acceleration, which always points towards the axis of robot rotation.

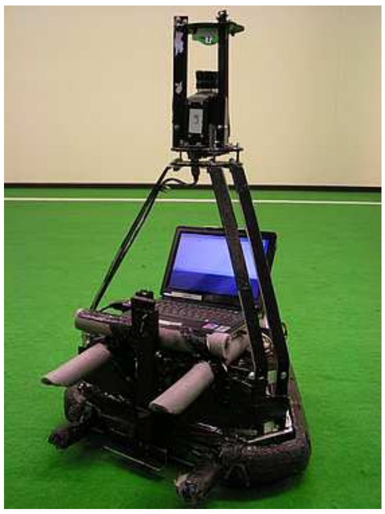
Multiplying (7.1) by the mass of the ball m_B, the extended Newton's second law with respect to the robot coordinate system is given by

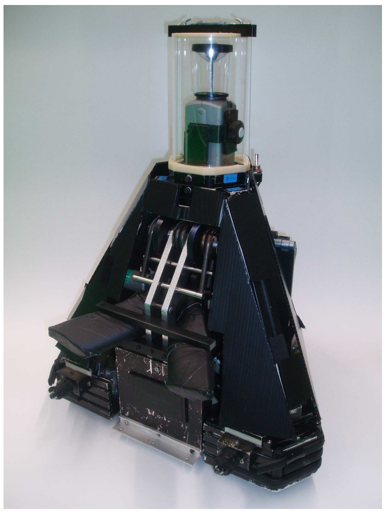
$$\mathbf{F}_B^m = \mathbf{F}_B + \mathbf{F}_{in}, \tag{7.2}$$

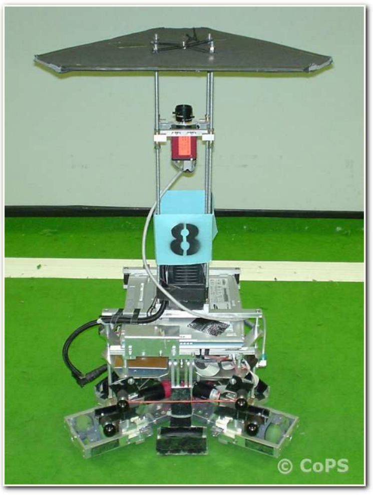
where $\mathbf{F}_B = m_B \mathbf{a}_B$ and $\mathbf{F}_B^m = m_B \mathbf{a}_B^m$. $\mathbf{F}_B$ is the vector sum of all the external force acting on the ball with respect to the world coordinate system. $\mathbf{F}_B$ is composed of the force exerted by the robot and the friction between the ball and


(a) CAMBADA robot 2007. One active wheel presses on the ball from above. Two inactive wheels are mounted on the front of the robot base.

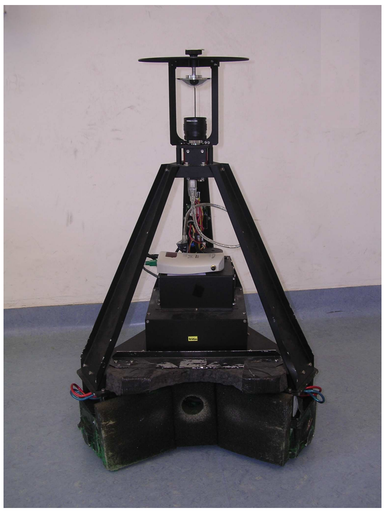

(b) Tech United Eindhoven robot 2008 . Two active wheels act as top dribblers, which can change the contact points on the ball and drive the ball in the longitudinal and lateral directions.


(c) Tribots robot 2006. The dribbling system consists of four rubber cylindrical bars representing the corners of an isosceles trapezoid.


(d) Hibikino-Musashi robot 2006. The Dribblers are similar to those of the Tribots robot, but the upper two dribblers have a flat form.



(e) Cops robot 2006. Four short mental bars are used. A rubber spring is connected between the upper two dribblers to damp collisions of the ball.



(f) Attempto robot 2006. A rubber foam pad is assembled above the concave robot front. Three spongy blocks are attached to the robot front to improve the damping property.

Figure 7.0: Soccer robots of the RoboCup Middle Size League

the floor. $\mathbf{F}_B^m = m_B\mathbf{a}_B^m$ is the vector sum of the force with respect to the robot coordinate system. $\mathbf{F}_{in}$ is the inertial force calculated as

$$\mathbf{F}_{in} = -m_B\mathbf{a}_R - m_B(2\boldsymbol{\omega} \times \mathbf{v}_B^m + \dot{\boldsymbol{\omega}} \times \mathbf{r}_B^m + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}_B^m)). \tag{7.3}$$

Equation (7.2) implies that not only the external force but also the inertial force is exerted on the ball, when the ball is observed in the robot coordinate system. The inertial force stems from the acceleration of the reference coordinate system, which is the robot coordinate system. Although the inertial force manifests itself as a real force, it is not the real one while it results from the non-inertial reference coordinate system but not from interactions with other bodies.

If the ball is moving along a curve with a clockwise turning as shown in Figure 7.1, it is only possible for the robot to keep the ball if the force $\mathbf{F}_B^m$ has nonpositive projection on the line $\overrightarrow{BL}$, which is parallel to the left border of the robot's front. That yields

$$(\mathbf{F}_B^m)_{\overrightarrow{BL}} = (\mathbf{F}_B + \mathbf{F}_{in})_{\overrightarrow{BL}} \leq 0. \tag{7.4}$$

When the ball follows a curve, the external force $\mathbf{F}_B$ can be projected on the tangent and normal directions of the curve. The tangent part $\mathbf{F}_t$ can be calculated as $\mathbf{F}_t = m_B\mathbf{a}_t$ with a acceleration $\mathbf{a}_t$. The normal part $\mathbf{F}_n$ is pointing to the center of curvature and has the magnitude $|m_B c \mathbf{v}_B^2|$. c is the curvature of the curve and

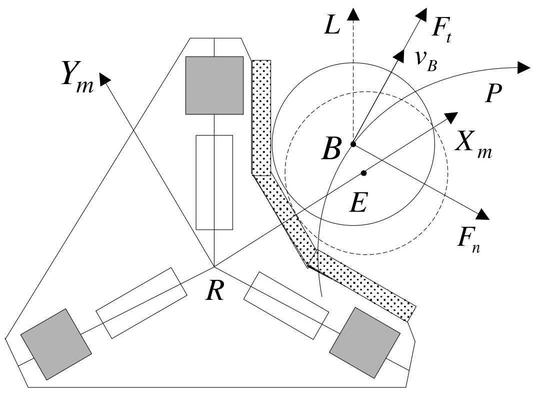


Figure 7.1: The ball's relative position in the robot coordinate system. Three spongy blocks are pasted on the robot's front to increase the friction. Point E denotes the desired position of the ball's center B.

$\mathbf{v}_B$ is the ball's moving velocity. $\mathbf{F}_{in}$ is related to the ball's relative state and the robot movement. Inequality (7.4) represents the constraint of robot movement in the dribbling process, under which the robot can keep the ball moving along curved paths avoiding losing the ball. The analysis presented above assumes that the robot rotates in the clockwise direction, but similar results can be obtained in the non-clockwise case. However, the constraint (7.4) can not be easily satisfied. While the interaction between the robot and the ball is difficult to be modeled in mathematical equations, it is hard to analytically determine the desired robot movement, which guarantees suitable force exerted on the ball in the dribbling process.

Based on the analysis of the relative movement between the ball and the robot, an analytical dribbling control strategy is designed for the Attempto soccer robot. Considering the fully free mobility of the omnidirectional robot, the dribbling control strategy assigns different tasks to the translation control and rotation control of the robot. Without exact modeling the interaction between the ball and the robot, the analytical results yield an efficient condition of the robot movement for a successful dribbling control.

7.3 Dribbling Control Strategy

In an environment with obstacles, the dribbling control problem considered here refers to a robot moving along some obstacle free paths and keeping the ball in the whole dribbling process. Figure 7.2 illustrates an ideal situation in the dribbling process, where the ball moves along a curve P and the ball's center B matches point E located at the front of the robot. Point E is depicted in Figure 7.1, which has a fixed distance L to the robot's center of mass R and the coordinate $(L, 0)$ in the robot coordinate system. Since point E is the ideal position of the ball's center B, the dribbling control can be formulated to control point E to follow the desired path and keep the ball near point E simultaneously. With respect to the advantage of the omnidirectional robot, i.e. the decoupled controllability of translation and rotation, the dribbling control strategy can be achieved by assigning the path following task and the ball keeping task to the robot translation and rotation control, respectively. The following contents in this subsection will detail the dribbling control strategy from the aspects of the controlled kinematic system, the translation control of point E' and the rotation control of the robot.

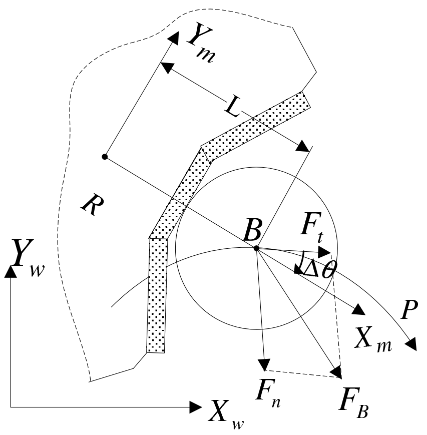


Figure 7.2: Force analysis in an ideal dribbling situation, where the ball's center B moves along a curve and matches point E located at the front of the robot. $\Delta\theta$ denotes the angular deviation between θ and θ_P, i.e. $\Delta\theta = \theta - \theta_P$.

7.3.1 Kinematic model in dribbling control

The definition of point E implies that it has a fixed coordinate $(L, 0)$ with respect to the robot coordinate system, which results in the relationship between point E and the robot center of mass R defined as,

$$x_E = x_R + L\cos\theta, \tag{7.5}$$

$$y_E = y_R + L\sin\theta, \tag{7.6}$$

where x_E and y_E denote the position of point E with respect to the world coordinate system. Substituting the robot kinematic model (3.2) into the following derivatives of (7.5) and (7.6),

$$\dot{x}_E = \dot{x}_R - L\omega\sin\theta, \tag{7.7}$$

$$\dot{y}_E = \dot{y}_R + L\omega\cos\theta, \tag{7.8}$$

the kinematic model of Point E is given by

$$\dot{x}_E = \dot{x}_R^m \cos\theta - \dot{y}_R^m \sin\theta - L\omega \sin\theta, \tag{7.9}$$

$$\dot{y}_E = \dot{x}_R^m \sin\theta + \dot{y}_R^m \cos\theta + L\omega \cos\theta, \tag{7.10}$$

where $\dot{x}_E$ and $\dot{y}_E$ denote the point E's velocity with respect to the world coordinate system.

Combining point E's kinematics with the robot rotation, the following system is employed in the dribbling control strategy,

$$\begin{bmatrix} \dot{x}_E \\ \dot{y}_E \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & -L\sin\theta \\ \sin\theta & \cos\theta & L\cos\theta \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_R^m \\ \dot{y}_R^m \\ \omega \end{bmatrix}, \tag{7.11}$$

This system has the same inputs as those of the robot kinematic model (3.2), but the outputs consist of point E's velocity and robot rotation velocity. Note that the transformation matrix in (7.11) is also full rank, which means the decoupled controllability of translation and rotation is inherited. Therefore, the control of point E to follow the reference path and the robot orientation to track the desired orientations can be achieved separately.

7.3.2 Translation Control

The path following problem of point E can be solved with the presented control methods of Chapter 4. The two inputs $\dot{x}_R^m$ and $\dot{y}_R^m$ in (7.11) can completely control the values of $\dot{x}_E$ and $\dot{y}_E$ whatever the value of ω is. During the dribbling process, the ball can only be pushed but not be pulled. To decrease the ball's speed, the robot has to move ahead and to hinder the ball's movement. When the ball's speed is required to increase, the robot needs to stay behind the ball and push it. Therefore, varying the ball's speed will unsmooth the robot movement and increase the possibility of the robot to lose the ball. In the dribbling strategy, the ball is required to move along a reference path with a highly constant speed, which not only facilitates the robot motion control, but also ensures a fast movement of the ball in the RoboCup matches.

7.3.3 Rotation Control

Besides the robot translation control, the degree of freedom of robot rotation can be used to keep the ball near point E. When point E is controlled along a reference path, point E's acceleration with respect to the world coordinate system is given by

$$\ddot{x}_E = cv_E^2 \sin(\theta - \theta_P), \tag{7.12}$$
$$\ddot{y}_E = cv_E^2 \cos(\theta - \theta_P). \tag{7.13}$$

v_E is point E's velocity, c and θ_P are the curvature and tangent direction at the position of point E on the reference path. Differentiating (7.5) and (7.6) twice yields,

$$\ddot{x}_E = \ddot{x}_R - L\dot{\omega}\sin\theta - L\omega^2\cos\theta, \tag{7.14}$$
$$\ddot{y}_E = \ddot{x}_R + L\dot{\omega}\cos\theta - L\omega^2\sin\theta. \tag{7.15}$$

The robot acceleration can be directly calculated from point E's acceleration with the following equations,

$$\ddot{x}_R = cv_E^2 \sin(\theta - \theta_P) + L\dot{\omega}\sin\theta + L\omega^2\cos\theta, \tag{7.16}$$
$$\ddot{y}_R = cv_E^2 \sin(\theta - \theta_P) - L\dot{\omega}\cos\theta + L\omega^2\sin\theta. \tag{7.17}$$

$\ddot{x}_R$ and $\ddot{y}_R$ denote the robot translation acceleration with respect to the world coordinate system.

Substituting (7.16) and (7.17) into the projection of (7.1) into the robot coordinate system, the ball's kinematics is determined as,

$$\ddot{x}_B^m = \ddot{x}_{Bm} - cv_E^2 \sin(\theta - \theta_P) - L\omega^2 + 2\omega\dot{y}_B^m + \dot{\omega}y_B^m + \omega^2 x_B^m, \tag{7.18}$$
$$\ddot{y}_B^m = \ddot{y}_{Bm} - cv_E^2 \cos(\theta - \theta_P) + L\dot{\omega} - 2\omega\dot{x}_B^m - \dot{\omega}x_B^m + \omega^2 y_B^m, \tag{7.19}$$

where x_B^m and y_B^m denote the ball's position with respect to the robot coordinate system. $\dot{x}_B^m$, $\dot{y}_B^m$, $\ddot{x}_B^m$ and $\ddot{y}_B^m$ are the corresponding velocities and accelerations. $\ddot{x}_{Bm}$ and $\ddot{y}_{Bm}$ denote the projection of vector $\mathbf{a}_B$ in the robot coordinate system, $\ddot{\omega}$ is the robot rotation acceleration. It is noticed in (7.18) and (7.19) that the ball's states with respect to the robot coordinate system are determined by the robot orientation, the robot rotation velocity and rotation acceleration. v_E is determined by the translation control. c and θ_P are derived from the pre-designed reference path.

According to the constraint (7.4) of the robot to keep the ball, $\ddot{x}_B^m$ is required to be less and equal to zero along the axis $\mathbf{X}_m$, and $\ddot{y}_B^m$ is required to drive the ball to the position of point E. When the ball is near point E, which means $(x_B^m, y_B^m) \rightarrow (L, 0)$ and $(\dot{x}_B^m, \dot{y}_B^m) \rightarrow (0, 0)$, (7.18) and (7.19) have the following approximations:

$$\ddot{x}_B^m \approx \ddot{x}_{Bm} - cv_E^2 \sin(\theta - \theta_P), \tag{7.20}$$
$$\ddot{y}_B^m \approx \ddot{y}_{Bm} - cv_E^2 \cos(\theta - \theta_P), \tag{7.21}$$

where the robot orientation plays an important role on controlling the ball's relative acceleration $\ddot{x}_B^m$ and $\ddot{y}_B^m$. Because the robot can only push the ball, the value of $\Delta\theta = \theta - \theta_P$ is bounded to the interval $-\frac{\pi}{2} \leq \Delta\theta \leq \frac{\pi}{2}$. If $\Delta\theta$ has the same sign as that of c, the inertial acceleration $-cv_E^2 \sin(\Delta\theta)$ always points in the negative direction of the axis $\mathbf{X}_m$, which presses the ball to the robot. However, the inertial acceleration $-cv_E^2 \cos(\Delta\theta)$ causes the ball to slide away from the robot. Therefore, selecting $\Delta\theta$ with the same sign of c can be beneficial for the ball keeping task. But the magnitude of $\Delta\theta$ is a trade-off between the inertial accelerations $-cv_E^2 \sin(\Delta\theta)$ and $-cv_E^2 \cos(\Delta\theta)$. If $\Delta\theta$ is larger, there is more pressure on the ball but less force to prevent the ball from sliding away, and vice versa.

On the other hand, $\ddot{x}_{Bm}$ and $\ddot{y}_{Bm}$ refer to the external force acting on the ball, which is mainly influenced by the robot movement. Although the interaction between the robot and the ball is hard to know, the angular deviation $\Delta\theta$ as shown in Figure 7.2 is necessary for the robot to provide the ball with enough pushing force F_t and centripetal force F_n. Moreover, $\Delta\theta$ has a relationship with the centripetal acceleration cv_E^2.

As a consequence of above analysis, $\Delta\theta$ is selected proportional to the centripetal acceleration with a positive parameter k_θ:

$$\Delta\theta = k_\theta c v_d^2. \tag{7.22}$$

This results in the following desired robot orientation

$$\theta^d = \theta_P + k_\theta c v_d^2. \tag{7.23}$$

Then, the rotation control problem is to steer the robot orientation to track the desired ones.

7.4 Following a Static Path

In the dribbling task, a pre-designed reference path connects the ball's position and the target position through the free space on the play field. When obstacles are static or move slowly, a static path can be adopted in the whole dribbling process. To test the performance of the dribbling strategy, a sinusoidal path and an eight-shaped path were chosen for the dribbling task, because the varying curvature and symmetry of these paths are complex enough to validate the robot's agility of dribbling a rolling ball. Considering the space limitation of the robot laboratory, the sinusoidal path is parameterized as,

$$\begin{aligned} x_r &= p, \\ y_r &= sin(1.5p), \end{aligned}$$

and the eight-shaped path is given by,

$$\begin{aligned} x_r &= 1.8sin(2p), \\ y_r &= 1.0sin(p). \end{aligned}$$

p is a scale variable. Also the value of 0.2 was chosen for k_θ, which determines the following desired robot orientation

$$\theta^d = \theta_P + 0.2cv_d^2.$$

The NMPC method was utilized here to fulfill the dribbling control task, while the results presented in Chapter 5 show that the NMPC approach can not only keep the robot working under the system constraints, but also get smooth traveling trajectories of the robot by specifying the corresponding objective function. Compared to the control problem presented in Chapter 5, the only difference is that point E is controlled in the path following problem instead of the robot center of mass R. To reduce the computational burden of NMPC, the orthogonal projection-based formulation of the path following problem was adopted, where only one control variable α_e is required. Therefore, the NMPC method aims to find suitable values of α_e and ω, such that point E follows a reference path and the robot orientation tracks the desired orientation. The formulation of the NMPC approach is described by equations (5.51) - (5.53). The cost function is from 5.24, the terminal penalty is from 5.25 and terminal constraints are designed as 5.29 - 5.32. α_e denotes the moving direction of point E with respect to the path coordinate system, which generates the desired moving velocities of point E in the world coordinate system as follows,

$$\dot{x}_E = v_E \cos(\alpha_e + \theta_p), \tag{7.24}$$
$$\dot{y}_E = v_E \sin(\alpha_e + \theta_p). \tag{7.25}$$

Substituting (7.24) and (7.25) into (7.7) and (7.8), the inputs of the linearized kinematic model (3.5), which are the desired translation velocities of the robot in the robot coordinate system, are given by,

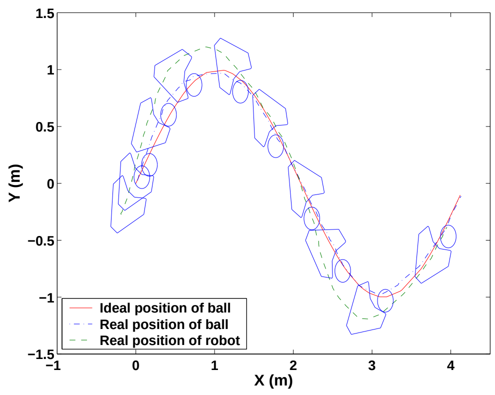
$$u_1 = \dot{x}_R^m = v_E \cos(\alpha_e + \theta_p) + L\omega \sin\theta, \tag{7.26}$$
$$u_2 = \dot{y}_R^m = v_E \sin(\alpha_e + \theta_p) - L\omega \cos\theta. \tag{7.27}$$

We chose the values of 1 m/s and 0.8 m/s for v_E in the experiments with the sinusoidal reference path and the eight-shaped reference path, respectively. The same parameters of the NMPC approach were used in the two experiments, which are

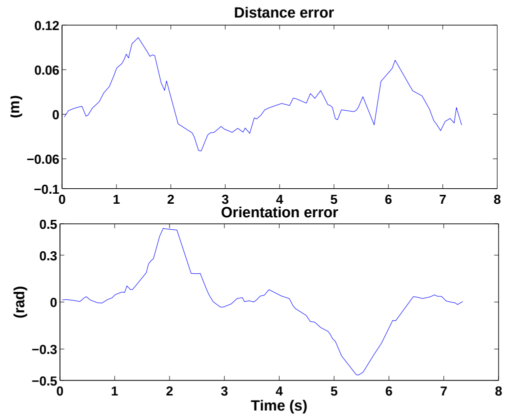
$$\mathbf{Q} = \begin{bmatrix} 0.3 & 0 \\ 0 & 0.5 \end{bmatrix}, \ \mathbf{R} = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.1 \end{bmatrix}, \ \alpha = \beta = 2.0.$$

The control horizon was chosen the same as the prediction horizon with $T_P = T_c = 3\tau$. The prediction sampling time τ and the estimated computational delay τ^c were selected as $\tau = 0.2$ s and $\tau^c = 0.25$ s.

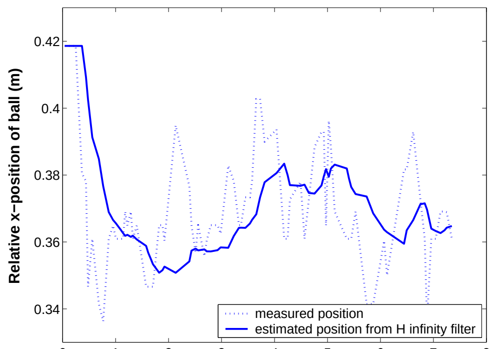
The experimental results with respect to the two reference paths are shown in figures 7.3 and 7.4. Figures 7.3(a) and 7.4(a) show the traveled paths of the ball and the robot. The dribbling errors are illustrated in figures 7.3(b) and 7.4(b), where the maximum deviation from point E to its projection on the reference path is less than 0.1 m and the largest angular errors with respect to the robot orientation are not more than 0.4 rad. Figures 7.3(c), 7.3(d), 7.4(c) and 7.4(d) show the filtered ball's relative positions with respect to the robot coordinate system based on H_∞ filters. They imply that the ball was kept during the dribbling processes, while the relative x-positions are mostly less than 0.4 m, and the relative y-positions are far from the boundary value 0.15 m. Figures 7.3(e), 7.4(e), 7.3(f) and 7.4(f) show the performance of the NMPC approach. The wheel velocities were kept under the boundary value 1.9m/s. Although the computational time is more than 0.2 s in some cases, the average computational times are 0.092 s and 0.091 s in both experiments, which is appropriate for dribbling the ball with the speed of 1 and 0.8 m/s, respectively.



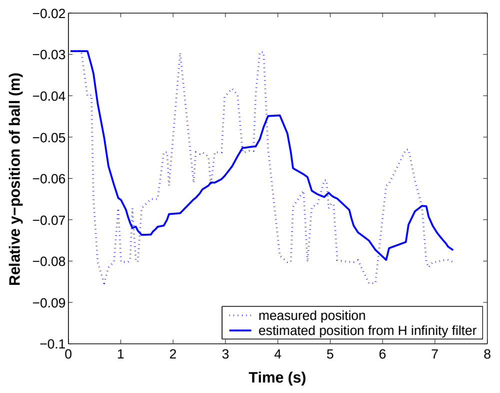
(a) Travelled paths of the robot and the ball.



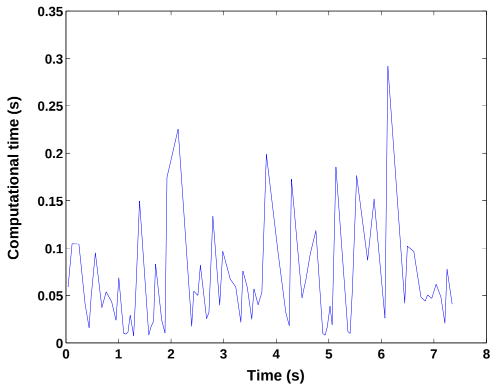
(b) Following errors.



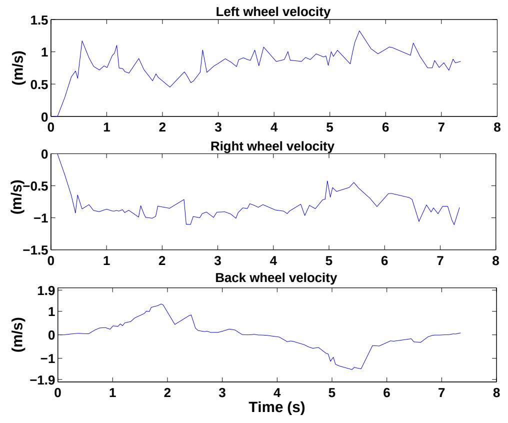
(c) Relative x-positions of the ball.



(d) Relative y-positions of the ball.

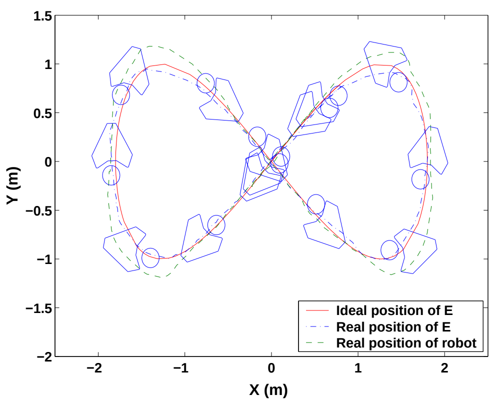


(e) Computational time of NMPC.

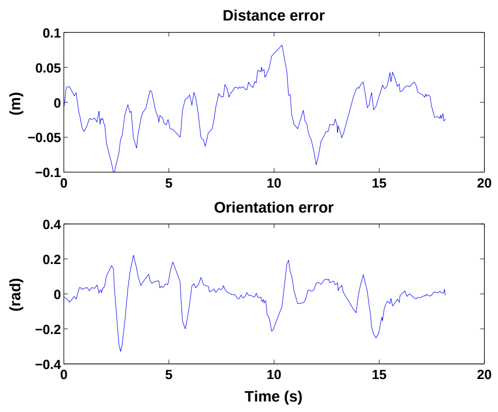


(f) Real wheel velocities.

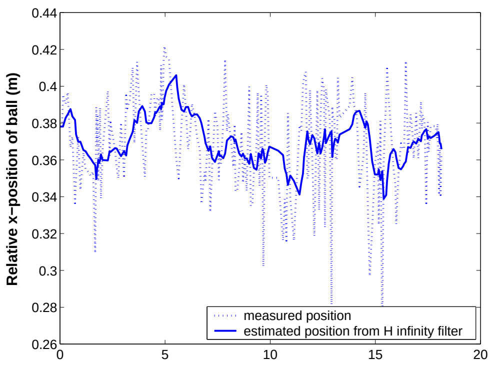
Figure 7.3: The NMPC-based dribbling control along the sinusoidal reference path.



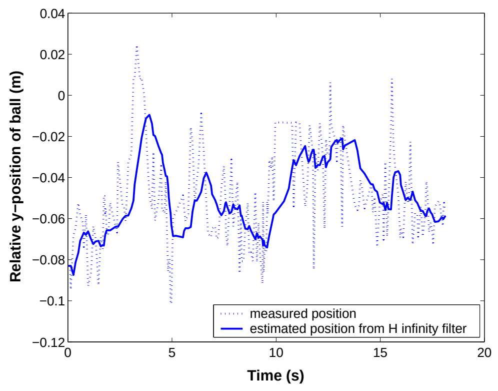
(a) Travelled paths of the robot and the ball.



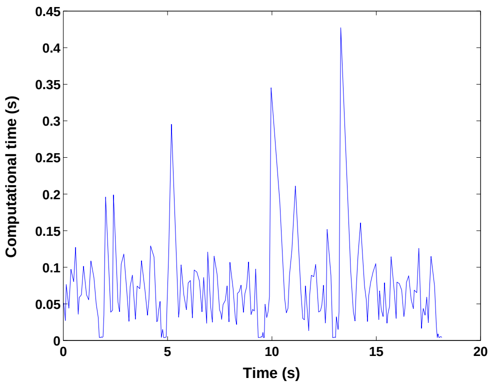
(b) Following errors.



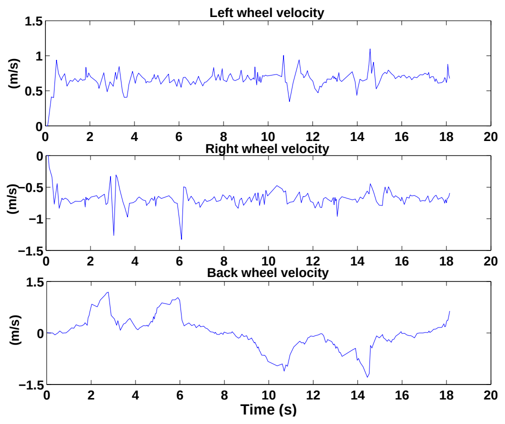
(c) Relative x-positions of the ball.



(d) Relative y-positions of the ball.



(e) Computational time of NMPC.



(f) Real wheel velocities.

Figure 7.4: The NMPC-based dribbling control along the eight-shaped reference path.

7.5 Following a Dynamic Path

The RoboCup match is a highly dynamic environment involving moving soccer robots and the ball. Although some situations enable a robot to dribble the ball along a static path and shoot a goal, the planned path should be updated with respect to the changed environment most of the time. Therefore, dribbling the ball along a dynamically planned path is more essential and offers a great challenge for the soccer robot.

7.5.1 Path Planning

The path planning method presented by Weigel et al [161] is used in the Attempto soccer robot, which is one of the most efficient path planning approaches in the RoboCup domain. This method is based on the potential field technique, and is able to navigate a robot out of a local minimum by introducing a grid-based planning method. Moreover, it is able to design a smooth path by reversing the positions of the target and the start position. The following subsections present more details of this path planning method.

7.5.2 Potential Field based Planner

Khatib [85] first reported a potential field for path planning of mobile robots. The idea behind the approach is to navigate a mobile robot as a charged particle moving in a magnetic field. This article is attracted to particles with the same sign and repelled by particles with the opposite sign. In a potential field, a robot has an attractive potential to the target and repulsive potentials away from obstacles and field boundaries. The planned moving direction of the robot is along the negative gradient of the potential field, which always points to the position with lower potential.

Each position $\mathbf{x} = (x, y)$ in the potential field is composed of an attractive potential well $p_T(\mathbf{x})$ and repulsive potential barriers $p_{O,i}(\mathbf{x})$ and $p_{B,j}(\mathbf{x})$. $p_T(\mathbf{x})$ is computed around the target $\mathbf{g} = (g_x, g_y)$. $p_{O,i}(\mathbf{o}_i)$ and $p_{B,j}(\mathbf{b}_j)$ are around the obstacles $\mathbf{o}_i = (o_{x,i}, o_{y,i})$ and the field boundaries $\mathbf{b}_j = (b_{x,j}, b_{y,j})$, respectively.

A conic well is chosen to model the attractive potential well, that is,

$$p_T(\mathbf{x}) = \rho_T \left\| \mathbf{d}_T \right\| , \tag{7.28}$$

where $\|\mathbf{d}_T\| = \|\mathbf{d} - \mathbf{g}\|$, $\|\cdot\|$ denotes the Euclidean norm. The corresponding negative gradient is calculated as

$$-\nabla p_T(\mathbf{x}) = -\frac{\rho_T}{\|\mathbf{d}_T\|}\mathbf{d}_T, \tag{7.29}$$

which causes the robot to move towards the target.

Repulsive potential barriers drive the robot away from obstacles and field boundaries. For the obstacle $\mathbf{o}_i$, the repulsive potential is computed as

$$p_{O,i}(\mathbf{x}) = \begin{cases} \rho_{O,i}(\mathbf{x}) & \|\mathbf{d}_{O,i}(\mathbf{x})\|^2 \leq \mu_O^2 \\ \rho_{O,i}\kappa_O \left(\frac{1}{\|\mathbf{d}_{O,i}(\mathbf{x})\|^2} - \frac{1}{M_O^2} \right) & \mu_O^2 < \|\mathbf{d}_{O,i}(\mathbf{x})\|^2 < M_O^2 \\ 0 & M_O^2 \leq \|\mathbf{d}_{O,i}(\mathbf{x})\|^2 , \end{cases} \tag{7.30}$$

which is reversely proportional to the distance $\|\mathbf{d}_{O,i}(\mathbf{x})\| = \|\mathbf{x} - \mathbf{o}_i\|$. To keep $p_{O,i}(\mathbf{x})$ continuous at $\|\mathbf{d}_{O,i}(\mathbf{x})\|^2 = \mu_O^2$, a normalization parameter κ_O is introduced with

$$\kappa_O = \frac{M_O^2 \mu_O^2}{M_O^2 - \mu_O^2}. \tag{7.31}$$

μ_O is the minimum distance used to hinder $p_{O,i}(\mathbf{x})$ from increasing infinitely. For the positions having less distance than μ_O to an obstacle, the corresponding potential barrier has the maximum value ρ_O. When obstacles are assumed to have a radius r_O, μ_O can be chosen as

$$\mu_O = r_O + r_R + \epsilon, \tag{7.32}$$

where r_R is the robot radius and ϵ is a security distance. Moreover, a maximum distance M_O is defined to reduce the amount of local minima resulting from a field containing many obstacles. If the obstacles are far away from the robot, they do not influence the robot movement. The negative gradient of the repulsive potential $p_{O,i}$ is given by

$$-\nabla p_{O,i}(\mathbf{x}) = \begin{cases} 2\frac{\rho_O \kappa_O \mu_O^2}{\|\mathbf{d}_{O,i}(\mathbf{x})\|^4} \mathbf{d}_{O,i}(\mathbf{x}) & \text{if} \quad \mu_O^2 < \|\mathbf{d}_{O,i}(\mathbf{x})\|^2 < M_O^2 \\ (0,0) & \text{if} \quad \|\mathbf{d}_{O,i}(\mathbf{x})\|^2 \leq \mu_O^2 \\ & \quad \vee \|\mathbf{d}_{O,i}(\mathbf{x})\|^2 \geq M_O^2 \end{cases} \tag{7.33}$$

Besides obstacles, the play field in RoboCup is limited by boundary lines. All boundaries are considered as potential barriers, whose repulsive potentials are computed similar to those of the obstacles.

$$p_{B,j}(\mathbf{x}) = \begin{cases} \rho_{B,j}(\mathbf{x}) & \|\mathbf{d}_{B,j}(\mathbf{x})\|^2 \leq \mu_B^2 \\ \rho_{B,j}\kappa_B \left(\frac{1}{\|\mathbf{d}_{B,j}(\mathbf{x})\|^2} - \frac{1}{M_B^2} \right) & \mu_B^2 < \|\mathbf{d}_{B,j}(\mathbf{x})\|^2 < M_B^2 \\ 0 & M_B^2 \leq \|\mathbf{d}_{B,j}(\mathbf{x})\|^2 , \end{cases} \tag{7.34}$$

where $\mathbf{d}_{B,j} = (0, y - b_{y,j})$ and $\mathbf{d}_{B,j} = (x - b_{x,j}, 0)$ are with respect to the boundary in the x-direction and y-direction, respectively. The normalization parameter κ_B

is defined as

$$\kappa_B = \frac{M_B^2 \mu_B^2}{M_B^2 - \mu_B^2}. \tag{7.35}$$

The minimum distance μ_B is only concerned with the robot radius r_R and a security distance ϵ,

$$\mu_B = r_R + \epsilon. \tag{7.36}$$

The negative gradient of $p_{B,j}(\mathbf{x})$, $-\nabla p_{B,j}(\mathbf{x})$, is calculated as

$$-\nabla p_{B,j}(\mathbf{x}) = \begin{cases} 2\frac{\rho_B \kappa_B \mu_B^2}{\left\|\mathbf{d}_{B,j}(\mathbf{x})\right\|^4} \mathbf{d}_{B,j}(\mathbf{x}) & \text{if} \quad \mu_B^2 < \left\|\mathbf{d}_{B,j}(\mathbf{x})\right\|^2 < M_B^2 \\ (0,0) & \text{if} \quad \left\|\mathbf{d}_{B,j}(\mathbf{x})\right\|^2 \leq \mu_B^2 \\ & \quad \vee \left\|\mathbf{d}_{B,j}(\mathbf{x})\right\|^2 \geq M_B^2 \end{cases} \tag{7.37}$$

Merging the attractive and repulsive potentials, the final potential field has the following potential:

$$P(\mathbf{x}) = p_T(\mathbf{x}) + \sum_i p_{O,i}(\mathbf{x}) + \sum_j p_{B,j}(\mathbf{x}). \tag{7.38}$$

The negative gradient of $P(\mathbf{x})$ is also a superposition of the negative gradients in each potential field, i.e.

$$-\nabla P(\mathbf{x}) = -\nabla p_T(\mathbf{x}) - \sum_i \nabla p_{O,i}(\mathbf{x}) - \sum_j \nabla p_{B,j}(\mathbf{x}). \tag{7.39}$$

7.5.3 Grid-based Planner

Although the negative gradient direction $-\nabla P(\mathbf{x})$ navigates the robot towards the target and away from obstacles and field boundaries, a local minimum problem may occur when a robot is trapped in a dead end. Therefore, methods for a robot to escape local minima are required. The grid-based planner is an example of such methods, which divides the whole space into square-shaped grid cells and designs a path composed of a set of grid cell's centers. The gradient of a grid cell's center at position (u, v) is approximated by evaluating the local potential field as

$$grad(u,v) = \frac{1}{2\alpha}\left[P(u+1,v) - P(u-1,v), P(u,v+1) - P(u,v-1)\right]^T$$

with a positive parameter α. If the next grid cell's center directed by $-\nabla P(\mathbf{x})$ goes into a local minimum, a recursive best-first search is started and terminated if either an adjacent grid cell with a lower potential or the target cell is found.

7.5.4 Experimental Results

To test the dribbling control method, some real world experiments were done in the robot laboratory. The size of the play field is 5.1×4.2 m^2, which is nearly the size of the old half play field of the RoboCup Middle Size league. A soccer robot was required to dribble the ball to the target field without colliding with one or two moving obstacles. At the start, the robot and the ball were static, and the robot was placed with an arbitrary orientation. The target field was defined by a circle centered at $(3, 1)$ having the radius of 0.3 m. The obstacles moved to and fro along a linear path of nearly 1 m length. The path was chosen such that it crosses the direction connecting the ball's initial position and the target field. The moving speed of the obstacles was 0.4 m/s.

As point E is controlled to follow a reference path in the dribbling strategy, the path planner was set to design a collision-free path of point E from its current position to the target. Then the nonlinear motion control method introduced in Chapter 4 and the NMPC law addressed in Chapter 5 were used in the experiments. For the path following control of point E, the orthogonal projection-based formulation was adopted. The desired velocity of point E was selected as 1 m/s.

Table 7.5.4 shows the selected values of the path planner's parameters. Because

ρ_T	ρ_O	ρ_B	r_R	r_O	ϵ	M_O	M_B	α
$1 \cdot 10^6$	$4 \cdot 10^5$	$2 \cdot 10^5$	30 cm	30 cm	5 cm	80 cm	50 cm	10

Table 7.1: Parameter values used in the path planner.

of the discretization of the grid, the planed path is very square-edged. To smooth the path, the average over the first m path points from the path point P_i is used to calculate the tangent direction $\theta_{P,i}$ of the path at point $\mathbf{P}_i$, i.e.

$$\theta_{P,i} = \frac{1}{m} \sum_{j=i}^{i+m-1} \mathbf{P}_j - \mathbf{P}_i.$$

The value of m was chosen as 10 in the experiments. When the number of points on the designed path is less than m, m takes the value of the number of path points. The curvature of the path at point $\mathbf{P}_i$ is approximated by

$$c_i = k_c(\theta_{P,i} - \theta_{P,i-1}),$$

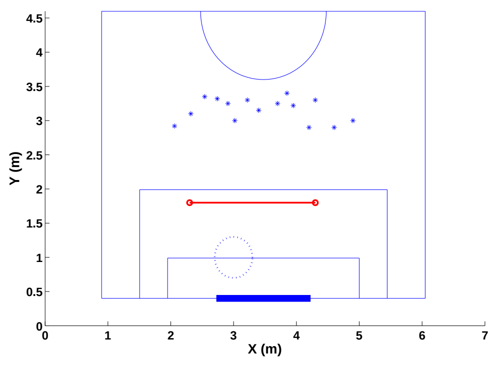
where k_c is a positive parameter and selected as 0.5 in the experiments.

To verify the performance of the dribbling control strategy, several initial positions of the ball were tested in the experiments as shown in Figure 7.5. The choice of initial positions considered the size of the laboratory and the trajectories of obstacles' movement, such that the obstacles' movement really have influence on the dribbling tasks. Table 7.5.4 gives a summary of the experimental results. In the case of one moving obstacle, the nonlinear motion control method and the NMPC method were tested. In 15 experiments with different initial positions, the NMPC-based dribbling control performed 9 successful dribblings, and the nonlinear motion control-based dribbling control succeeded 12 times. The successful ratios are 60% and 80%, respectively. But the computational time of NMPC is much longer than the one of the nonlinear motion control method, as the average computational time of eight experiments based on each control method are 0.138 s and 0.0387 s, respectively. When there were two moving obstacles, only the nonlinear motion control method yielded appropriate results. In 15 tests, the robot succeeded 10 times to dribble the ball from the initial position to the target field. The success ratio is 66.7%. Moveover, the computational time also remains very short (average value = 0.0392 s). These experimental results show that the dribbling control strategy worked efficiently and successfully in the dribbling tasks, and the nonlinear motion control-based dribbling control shows better performance than the NMPC-based one.

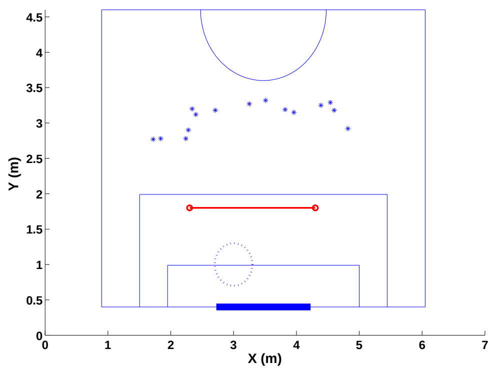
	One Obstacle (NMPC)	One Obstacle (Nonlinear Control)	Two Obstacles (Nonlinear Control)
Number of experiments	15	15	15
Number of successes	9	12	10
Success ratio (%)	60	80	66.7
Computational time (s)	0.138	0.0387	0.0392
Failure reason	planned paths: 6	planned paths:3	control method: 2 planned paths: 3

Table 7.2: Summary of experimental results. In the last row a failure reason " planned paths" means dribbling experiments failed because of the big change of two successive planned paths, a failure reason "control method" denotes dribbling experiments failed because the dribbling control method did not give efficient control values.

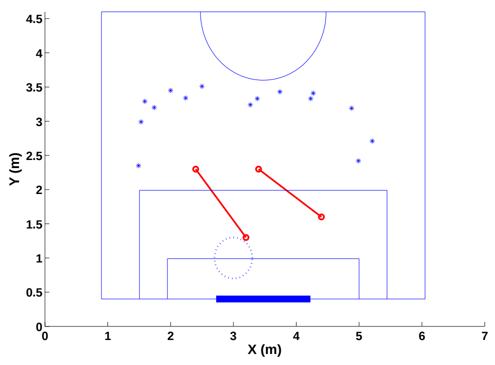
Figures 7.6, 7.7 and 7.8 show the detailed results of three successful dribbling experiments. The traveled paths of the robot and the ball in figures 7.6(a), 7.7(a) and 7.8(a) show that the robot successfully dribbled the ball to the target field, although the moving obstacles influenced the dribbling process. Figures 7.6(c),



(a) Initial positions of the ball in the NMPC-based experiments with one moving obstacle.



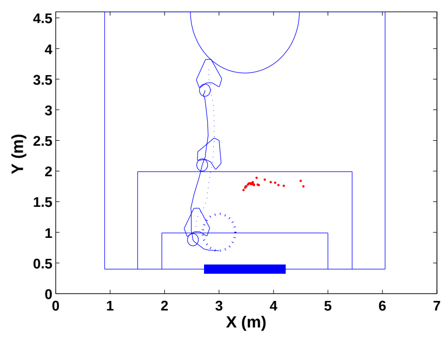
(b) Initial positions of the ball in the nonlinear motion control-based experiments with one moving obstacle.



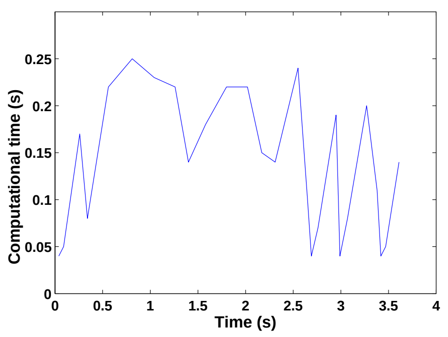
(c) Initial positions of the ball in the nonlinear motion control-based experiments with two moving obstacles.

Figure 7.5: Initial positions of the ball and the desired moving trajectories of obstacles in the experiments. The star symbols denote the ball's initial positions, which were randomly chosen in each experiment. The lines between two small circles are the desired moving trajectories of the obstacles. The dotted circles show the target area.

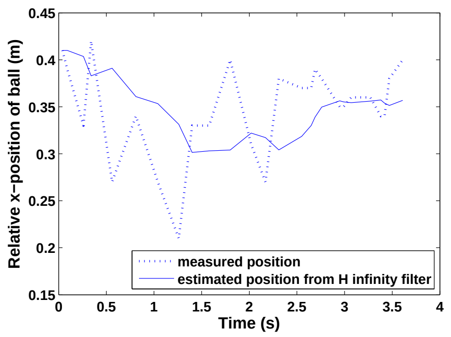
7.6(d), 7.7(c), 7.7(d), 7.8(c) and 7.8(d) show filtered relative positions of the ball using the H_∞ filter. These results indicate that the robot always succeeded in keeping the ball in the dribbling process. Figures 7.6(b), 7.7(b) and 7.8(b) show the NMPC method is of higher computational complexity than the nonlinear control method. This is the main reason that the NMPC-based dribbling control has a lower success ratio in the experiments. When the controller takes a longer computational time, the robot reacts to a changing environment more slowly. This delay may lead the robot and the ball near the moving obstacles. Then the new planned path may have a sharp turning from the old planned direction in order to avoid the collision with the obstacles. This sharp turning results in a big change of two successive planned paths. Therefore, this big change of planned path requires very sharp turns of the robot, which causes the ball to slide away from the robot.



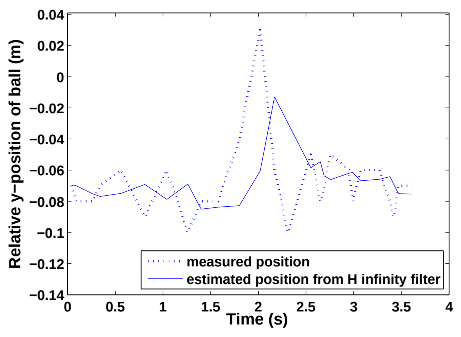
(a) Traveled paths of the robot and the ball.



(b) Computational time.

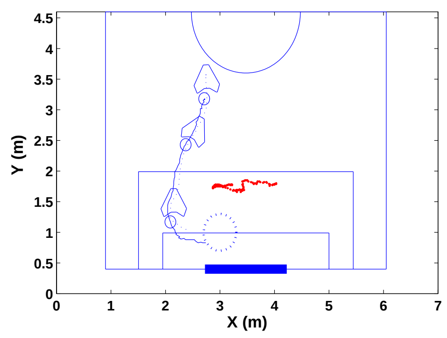


(c) Relative x-position of the ball.

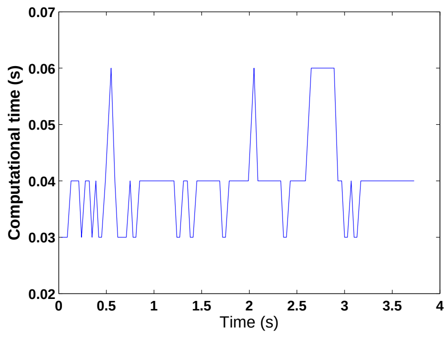


(d) Relative y-position of the ball.

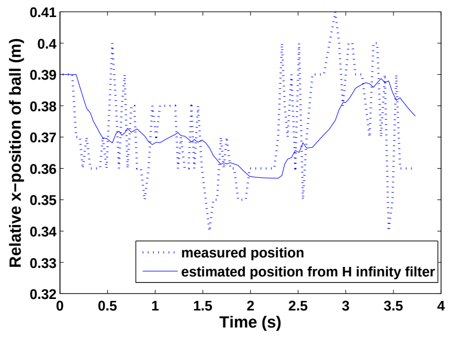
Figure 7.6: Experimental results of the NMPC-based dribbling control with one moving obstacle.



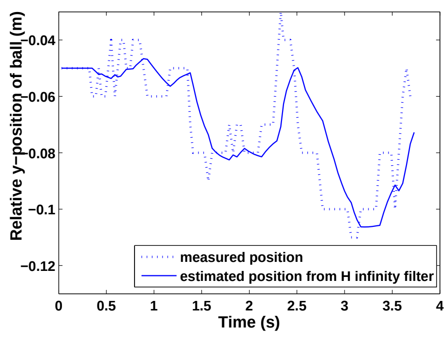
(a) Traveled paths of the robot and the ball.



(b) Computational time.

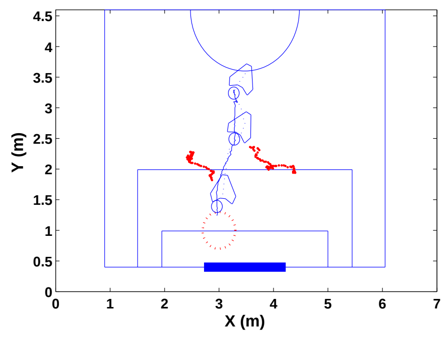


(c) Relative x-position of the ball.

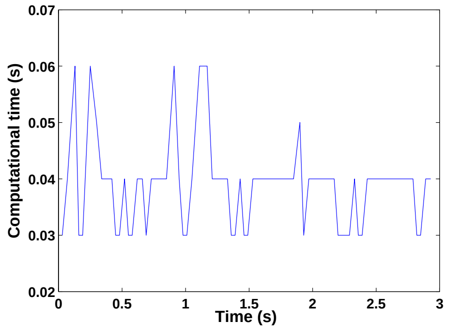


(d) Relative y-position of the ball.

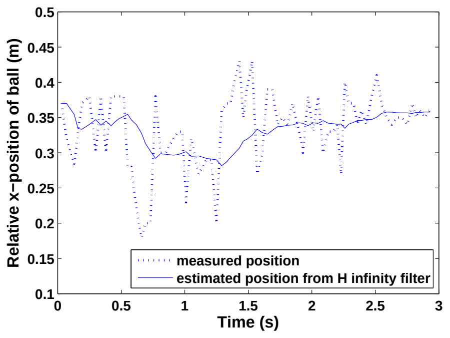
Figure 7.7: Experimental results of the nonlinear motion controller-based dribbling control with one moving obstacle.



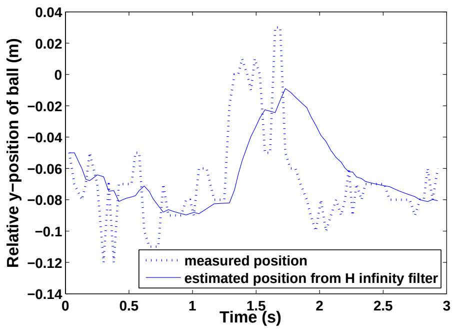
(a) Traveled paths of the robot and the ball.



(b) Computational time.



(c) Relative x-position of the ball.



(d) Relative y-position of the ball.

Figure 7.8: Experimental results of the nonlinear motion controller-based dribbling control with two moving obstacles.

7.6 Summary

This chapter addressed the dribbling control problem of Attempto soccer robots. From a hardware perspective, the dribbling mechanism used in the RoboCup Middle Size League teams is mostly designed with a concave front and a top component, which help forcing the ball towards the center of the robot's front and preventing the ball from rolling away. Although active dribblers may improve ball handling, passive dribbling mechanisms are widely used to decrease the complex hardware construction. The passive dribbling mechanism of the Attempto soccer robot also consists in a concave form and a top component. The specially designed bigger facing size enables the robot to easily catch the ball.

The main challenge of dribbling control is the difficulty of modeling the interactions between the robot and the ball, because it is hard to predict the friction coefficients and collisions between these two objects. Although some teams take the interactions as a black-box system and use an ANN to model it, the tedious process of collecting the training data and training the ANN motivates our research of designing an analytical dribbling controller for the Attempto soccer robots. Unlike the usual motion controller taking the robot center of mass as the controlled object, a reference point represented by the desired ball's center is used as the controlled object in the addressed dribbling control strategy. Analyzing the ball's movement related to the robot, a sufficient constraint of keeping the ball is deduced, which indicates an appropriate choice for the desired robot orientations. Making use of the advantage of omnidirectional robots, i.e. the decoupled translation and rotation, the dribbling task is achieved by controlling the reference point to follow a pre-designed path and steering the robot orientation to track the desired orientations simultaneously.

The dribbling strategy was fulfilled with the nonlinear motion control method introduced in Chapter 4 and the nonlinear model predictive control scheme addressed in Chapter 5. Real experiments showed the high performance and efficiency of the dribbling control strategy. Comparing the two motion control methods, the nonlinear motion control proved to be more adequate in relation to performance and run-time efficiency due to its low computational time. But the nonlinear model predictive control method showed very good performance in the case of ball dribbling along static paths. Besides real experiments in the robot laboratory, the dribbling strategy was successfully used by the Attempto soccer robots in the RoboCup 2006 in Bremen and revealed very good performance. Appendix A shows some image sequences of the dribbling experiments taken in the laboratory and during the games of the RoboCup 2006 in Bremen.

Chapter 8

Conclusions and Future Work

This thesis is concerned with motion control of omnidirectional robots. Considering important issues of mobile robots, such as actuator dynamics, actuator saturation and constraints of robot systems, this thesis focuses on achieving high control performance. As a testbed, the motion control of an omnidirectional robot of the Tübingen Attempto robot soccer team, especially the ball dribbling control of the soccer robot, has been considered in this thesis.

8.1 Conclusions

Before designing motion control methods, a control system combining dynamics and kinematics is adopted for the Attempto soccer robot. This architecture allows to design high-level controllers based on the kinematic model and low-level controllers according to the dynamic model. Although this hierarchy enables to design and test the control law of each level's system separately, the influence between each level has to be considered. For example, the high-level controller design has to take into account the performance of the low-level controlled system. Taking actuator saturation and actuator dynamics into account, the control system presented in Chapter 3 builds a foundation to design high-level controllers with consideration of the low-level system's performance.

Based on the robot control system, path following of omnidirectional robots was addressed in Chapter 4. The other two basic problems of robot motion control, trajectory tracking and point stabilization, can be regarded as special cases of the path following problem. The specialties are that the desired time parameterized velocities are designed in the trajectory tracking problem, and the point stabilization problem only needs to stabilize the robot at one desired pose. According to different ways of choosing the desired robot positions on the reference path, two formulations of the path following problem for omnidirectional robots have been introduced, i.e. the orthogonal projection-based formulation and the *Virtual Vehicle*-based formulation. Nonlinear controllers for these two formulations are designed based on Lyapunov's stability theorem. As the formulations of the path following problem are only based on the path following errors, but do not depend on specific robot platforms, the proposed controllers can be applied to other omnidirectional robots. Besides the path following control, an omnidirectional robot has another degree of freedom to control its orientation. A PD controller was designed to keep the robot tracking the desired orientations even though the actuators reach saturation.

Although the proposed nonlinear motion controllers guarantee closed-loop stability even though actuator saturation appears, there is still an opportunity to improve the control performance by considering more information about the given path. In Chapter 5, Nonlinear Model Predictive Control (NMPC) was adopted to the motion control problem of the Attempto soccer robot. The designed NMPC scheme guarantees closed-loop stability by choosing the suitable terminal penalty and constraints. With the selected numerical solutions, the results of real-world experiments show the feasibility of applying NMPC on a fast moving omnidirectional robot and better control performance, compared to the nonlinear controllers addressed in Chapter 4.

Before considering the dribbling control problem, this thesis first focused on tracking the ball's relative position with respect to a soccer robot when the ball is pushed by the robot. The relative position denotes whether the ball is moving away from the robot and results in changing the robot behaviors of ball dribbling and ball catching. A robust H_∞ filter was developed to estimate the ball's relative position and velocity, which does not require a priori knowledge about the statistical properties of the process noise and the measurement noise. It only depends on the assumption of finite noise power. The performance of the H_∞ filter was evaluated and compared to a Kalman filter. Although the performance of these two filters is similar, the independence of noise statistics makes the H_∞ filter more robust.

For the dribbling problem, this thesis focuses on designing analytical dribbling control methods. With the analysis of the ball's movement relative to the robot, a sufficient constraint of keeping the ball is deduced, which indicates an appropriate choice for the desired robot orientations. Then the dribbling task is achieved by controlling a reference point denoting the desired ball's center to follow a pre-designed path and steering the robot orientation to track the desired orientations. Thanks to the decoupled mobility of the omnidirectional soccer robot, these two subtasks can be assigned to the control of the robot translation and rotation, respectively. This dribbling control strategy is fulfilled with the nonlinear motion control method introduced in Chapter 4 and the NMPC scheme addressed in Chapter 5. Real experiments in the robot laboratory show the high performance and efficiency of the dribbling control strategy. Moreover, the dribbling control strategy was successfully used by the Attempto soccer robots at RoboCup 2006 in Bremen and revealed very good performance, where the reference point was controlled to follow the desired moving direction, and meanwhile the robot orientation was controlled to track the desired orientation with a PD controller.

Overall, the results in this thesis provide solutions to the motion control problem of an omnidirectional soccer robot. The control system and control methods presented in this thesis can also be applied to other omnidirectional robots. Moreover, several results addressed in this thesis offer the opportunity for further research.

8.2 Future Work

In the control system, the motion controllers are designed based on the robot kinematic model and take actuator dynamics and the maximum wheel velocity into account. Real-world experimental results show the good performance of the controlled system and the guaranteed closed-loop stability regardless of the appearance of the actuator saturation. However, there are other important issues that can be coped with to improve the robot motion control, for example, modeling the wheel slippage on the ground and taking the maximum wheel rotation acceleration into account.

The NMPC scheme was successfully used in controlling the movement of the Attempto soccer robot. Its good control performance is shown in the experiments in Chapter 5. However, finding more efficient NMPC schemes in the applications of controlling fast moving robots is still an attractive research direction. The main difficulty of applying NMPC is the computational effort, which was shown in the experimental results of dribbling control against moving obstacles. Therefore, to find and to apply more powerful optimization methods is required by NMPC schemes. Although the stability problem is quite well solved currently, finding better terminal constraints and penalties to increase the feasibility of the open-loop optimal control problem is still an active topic in NMPC research.

To achieve more proficient dribbling, the first attempt might be to design more advanced dribbling mechanisms, for example, using active dribblers, mounting special sensors to measure collisions between the robot and the ball. Although the dribbling strategy presented in this thesis serves for designing analytical dribbling controllers, there are also controller parameters in need of adaption. This denotes another future research of combining the dribbling control strategy with learning methods to obtain the optimal values of these parameters. Moreover, the dribbling strategy depends on a planned path. The path planner used in this work only designs a collision-free path without considering the constraints of successful dribbling. As a consequence, some designed paths have very sharp turns, which always result in the ball sliding away from the robot. Therefore, merging the constraints of dribbling into the path planner is also future work.

Appendix A

Dribbling Control Results

This appendix shows image scenarios of dribbling experiments in the robot laboratory with a size of 5.1×4.2 m^2. Two scenarios in the games of the RoboCup 2006 in Bremen are illustrated at the end.

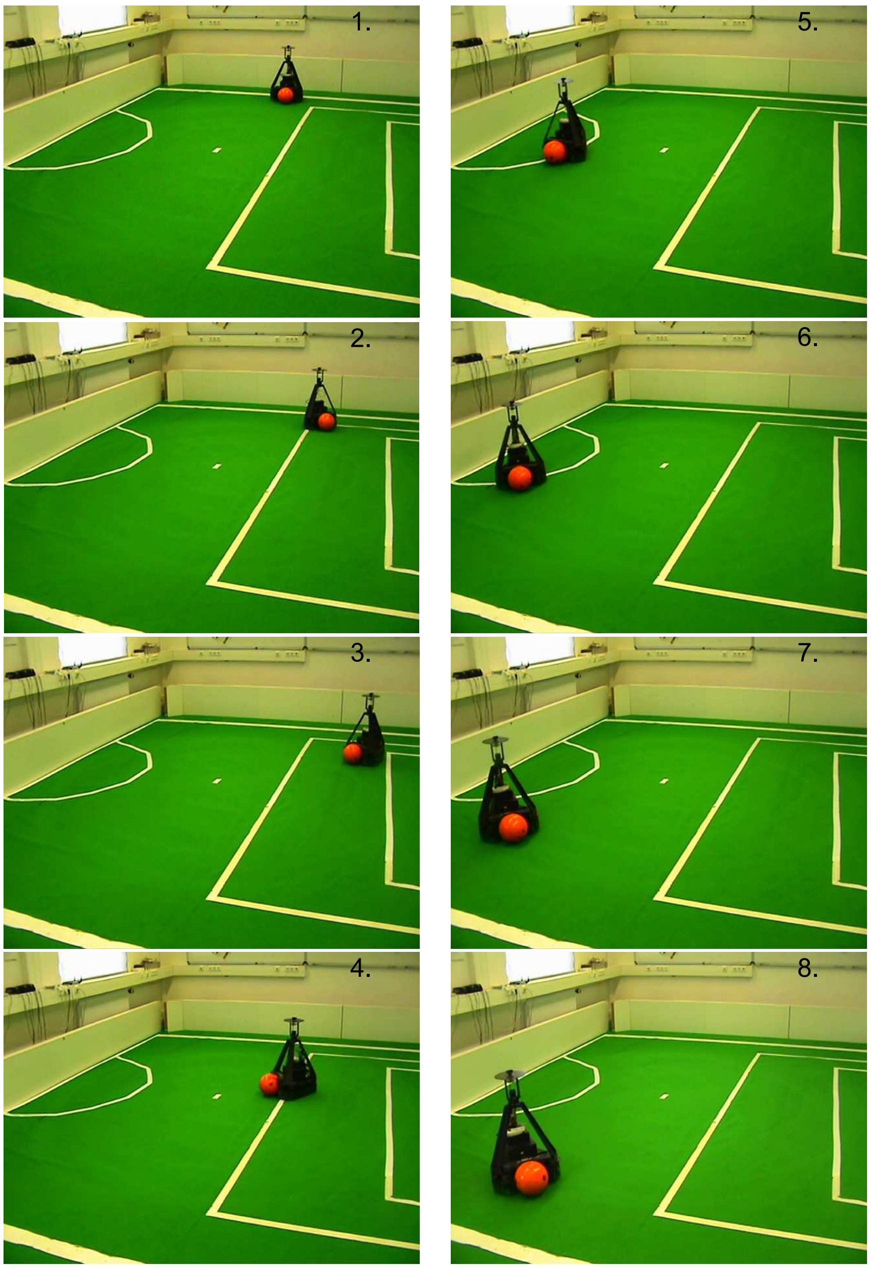


Figure A.1: Scenarios of the NMPC-based dribbling control along the sinusoidal reference path described in 7.4.

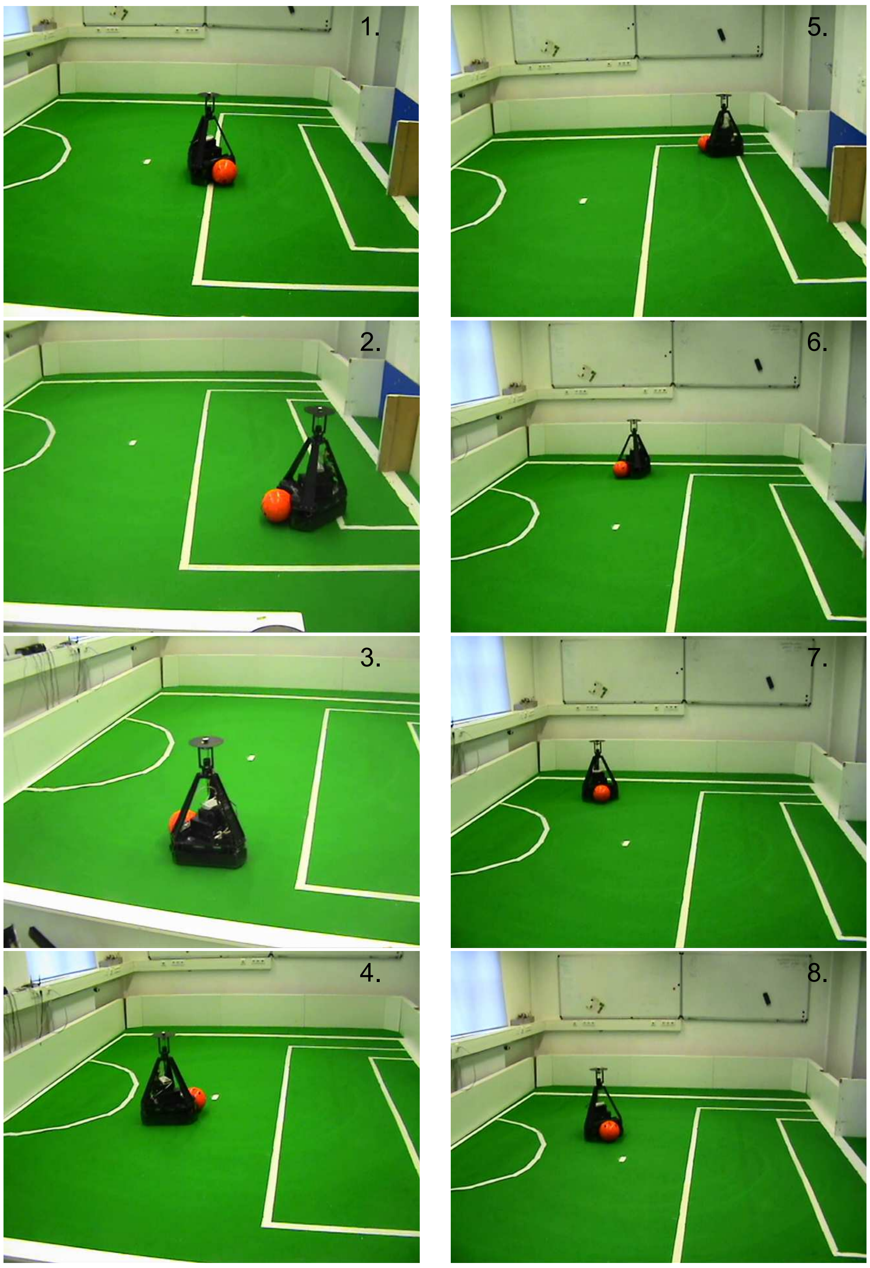


Figure A.2: Scenarios of the NMPC-based dribbling control along the eight-shaped reference path described in 7.4.

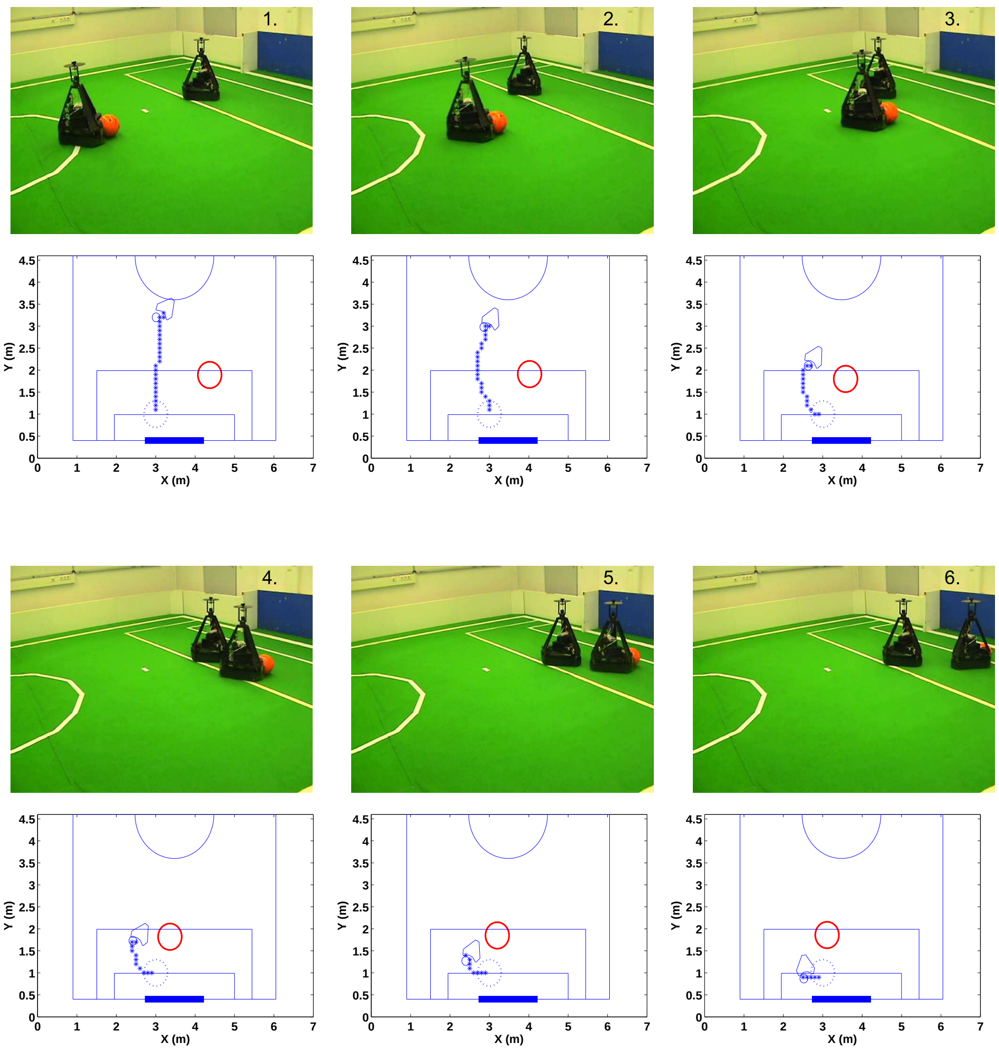


Figure A.3: Scenarios of a successful dribbling experiment based on the NMPC method with one moving obstacle. The line of asterisks denotes the planned path. The solid circle denotes the opponent. The dotted circle shows the target field.

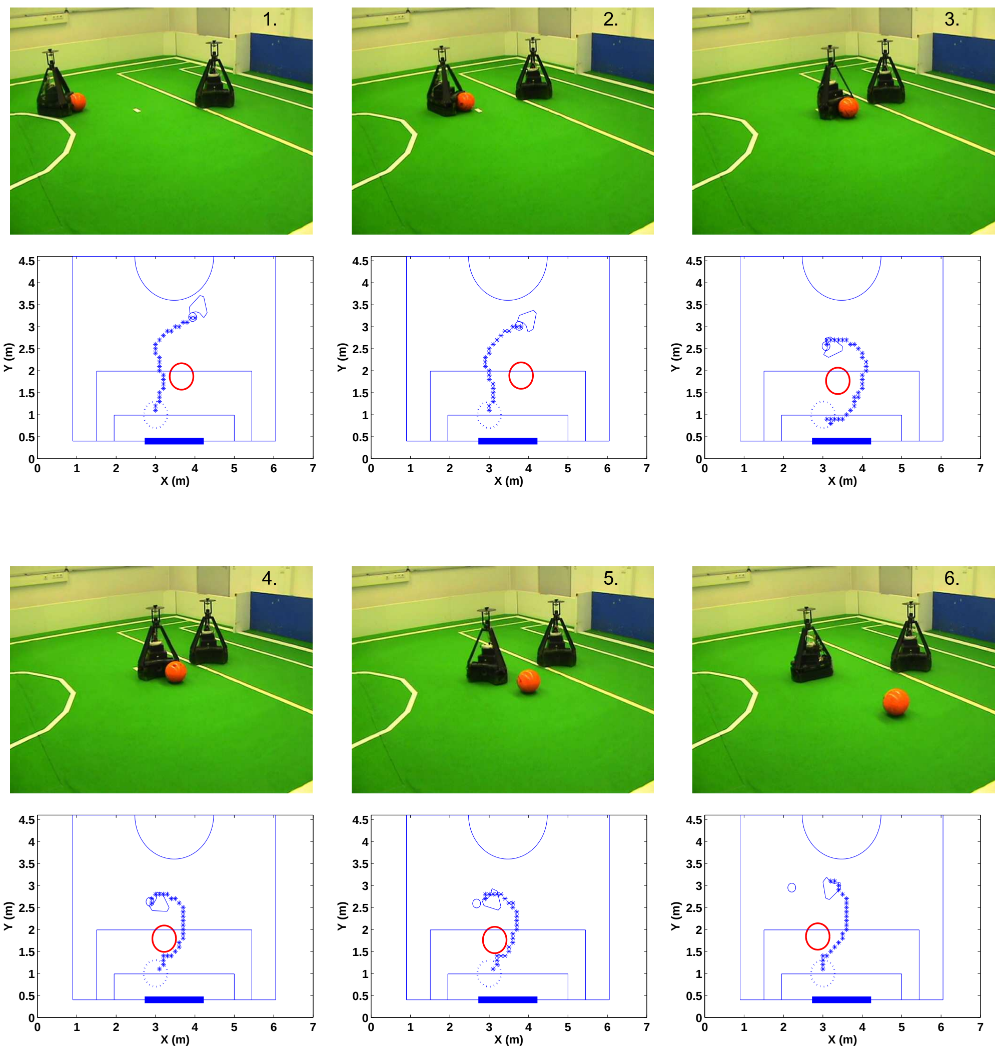


Figure A.4: Scenarios of a failing dribbling experiment based on the NMPC method with one moving obstacle. The line of asterisks denotes the planned path. The solid circle denotes the opponent. The dotted circle shows the target field. In the third image, the robot was required to turn nearly 180 degrees to follow the planned path. The resulting sharp turning of the robot lead to the loss of the ball.

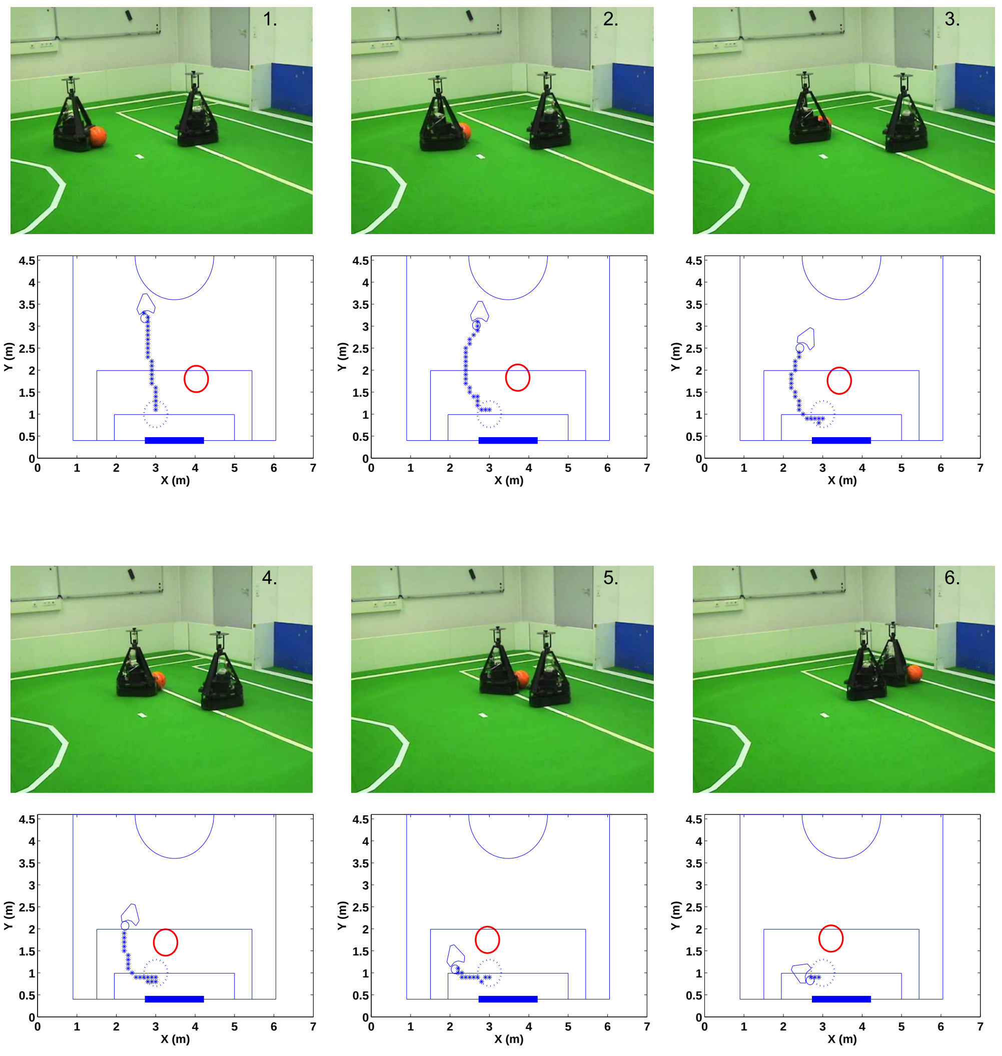


Figure A.5: Scenarios of a successful dribbling experiment based on the nonlinear control method with one moving obstacle. The line of asterisks denotes the planned path. The solid circle denotes the opponent.

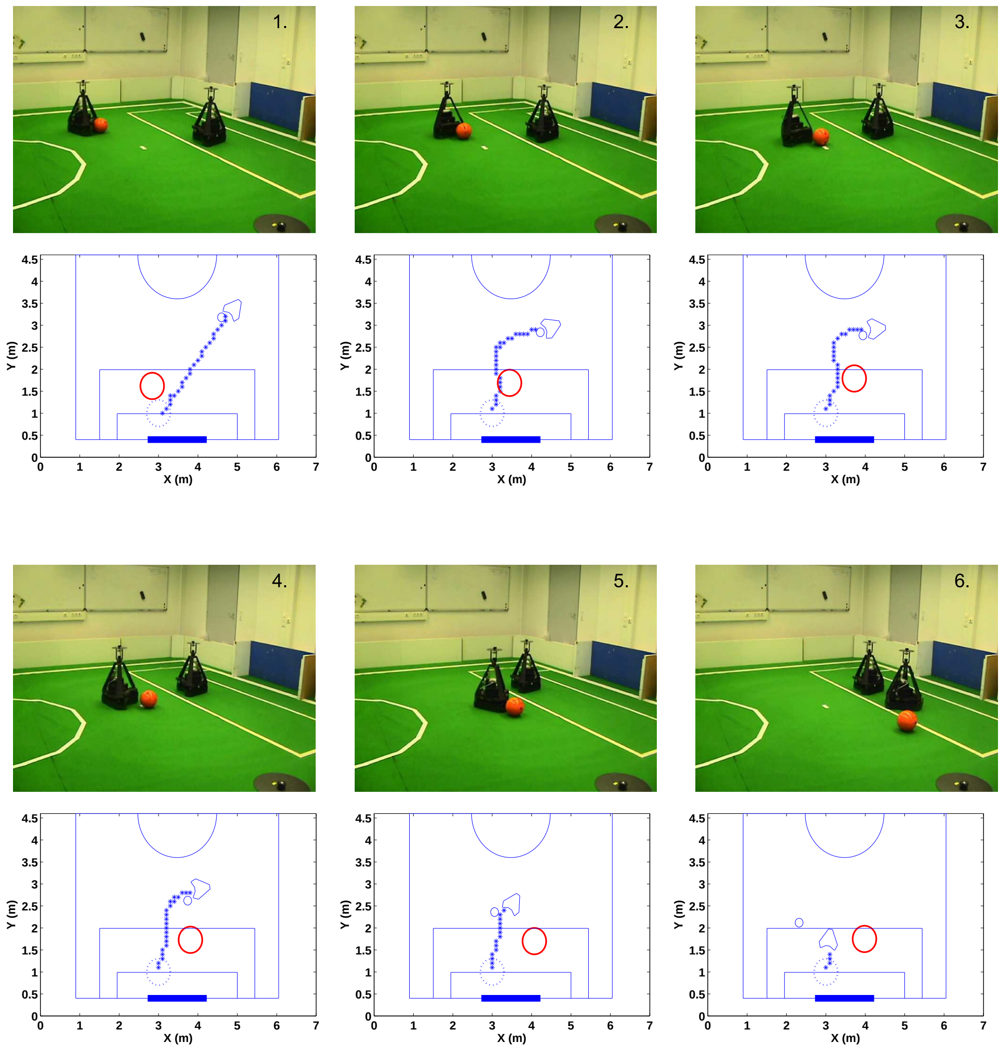


Figure A.6: Scenarios of a failing dribbling experiment based on the nonlinear control method with one moving obstacle. The line of asterisks denotes the planned path. As shown in the second and third images, the ball slided away from the robot when the robot tuned its orientation according to the changes of the planned paths. The resulting sharp turning of the robot lead to the loss of the ball.

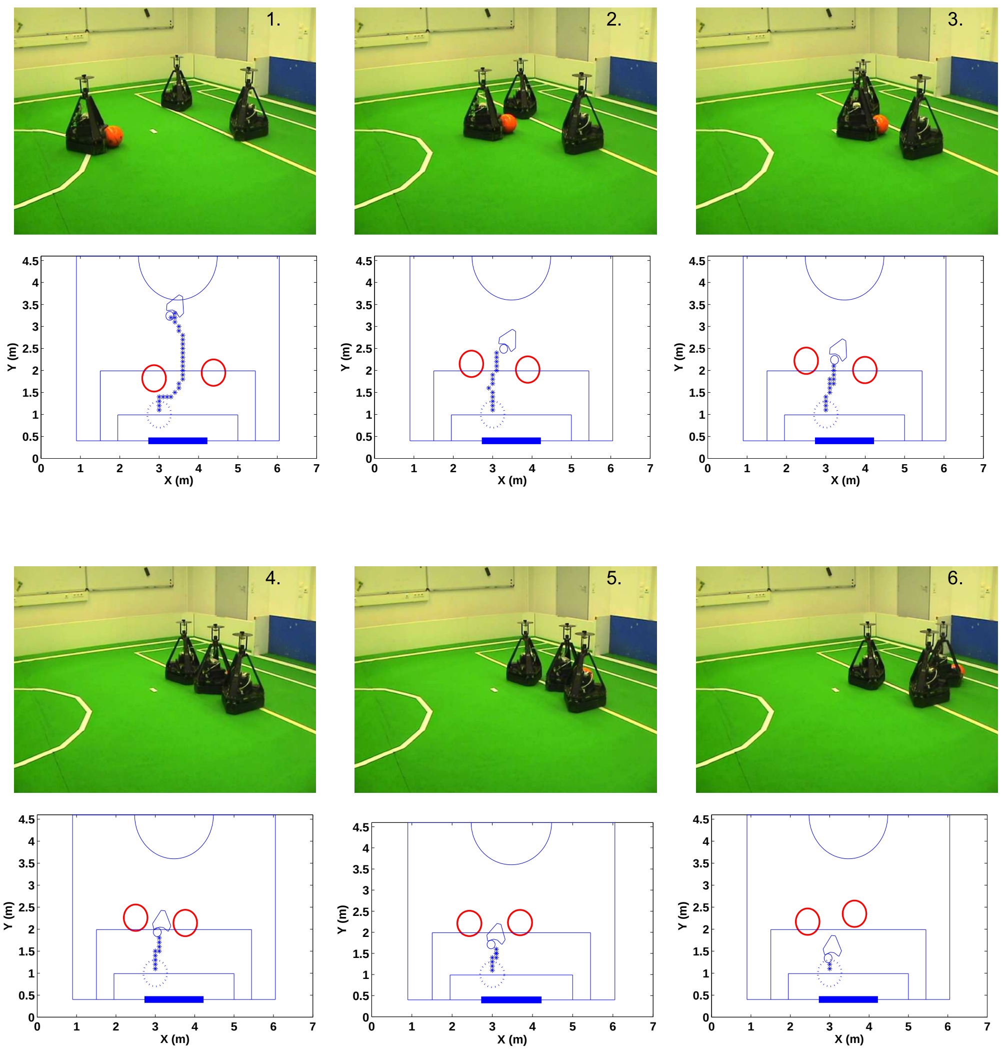


Figure A.7: Scenarios of a successful dribbling experiment based on the nonlinear control method with two moving obstacles. The line of asterisks denotes the planned path. The solid circles denote the opponents. The dotted circle shows the target field.

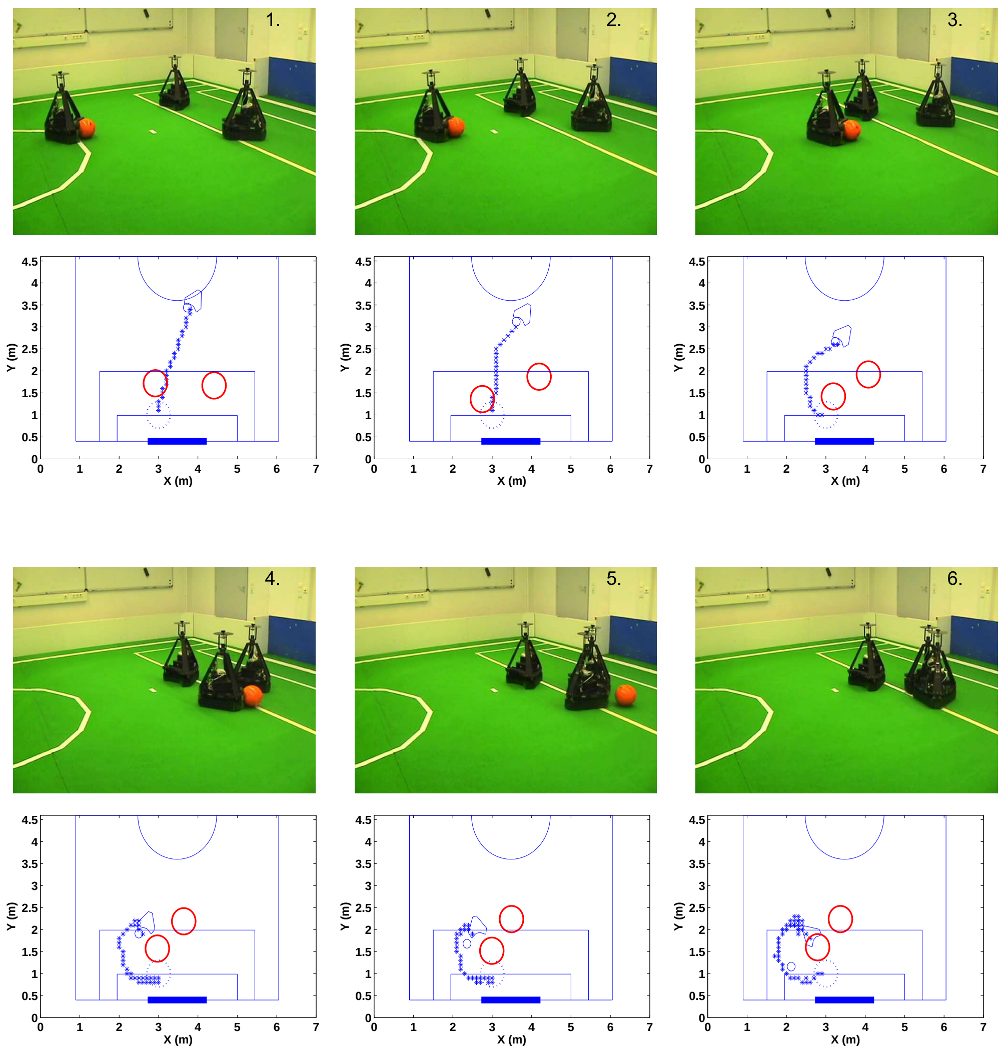


Figure A.8: Scenarios of a failing dribbling experiment based on the nonlinear control method with two moving obstacles. The line of asterisks denotes the planned path. In the fourth image, the robot was required to turn more than 90 degrees for following the planned path. The resulting sharp turning of the robot lead to the loss of the ball.

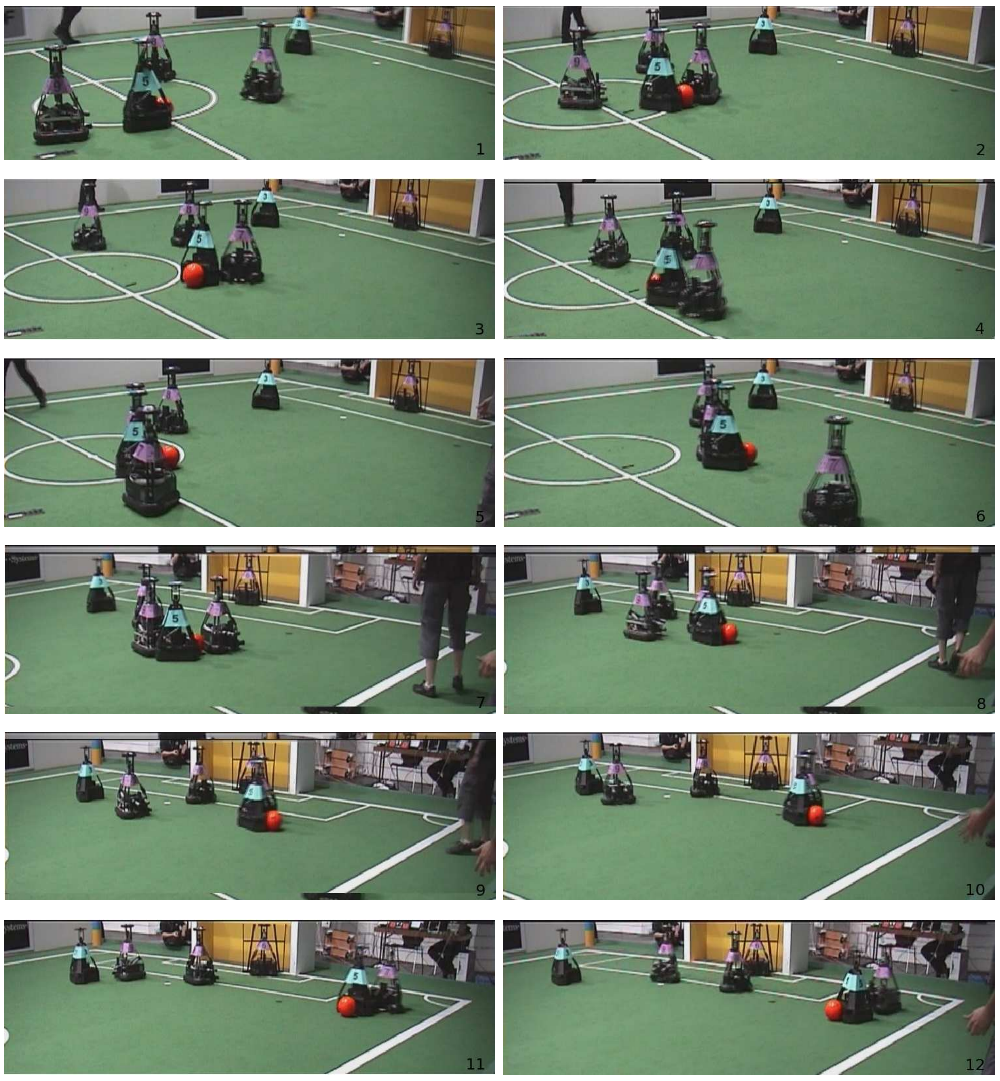


Figure A.9: Scenarios of a successful dribbling in the game against the Brainstormers Tribots at the RoboCup World Cup 2006 in Bremen. The Attempto soccer robot No. 5 makes two full rotations around the ball to shield it from the opponents at the beginning and the end of the scenarios. Between the two rotations, the robot dribbles the ball to approach the goal. The images are sorted from left to right and from top to bottom.

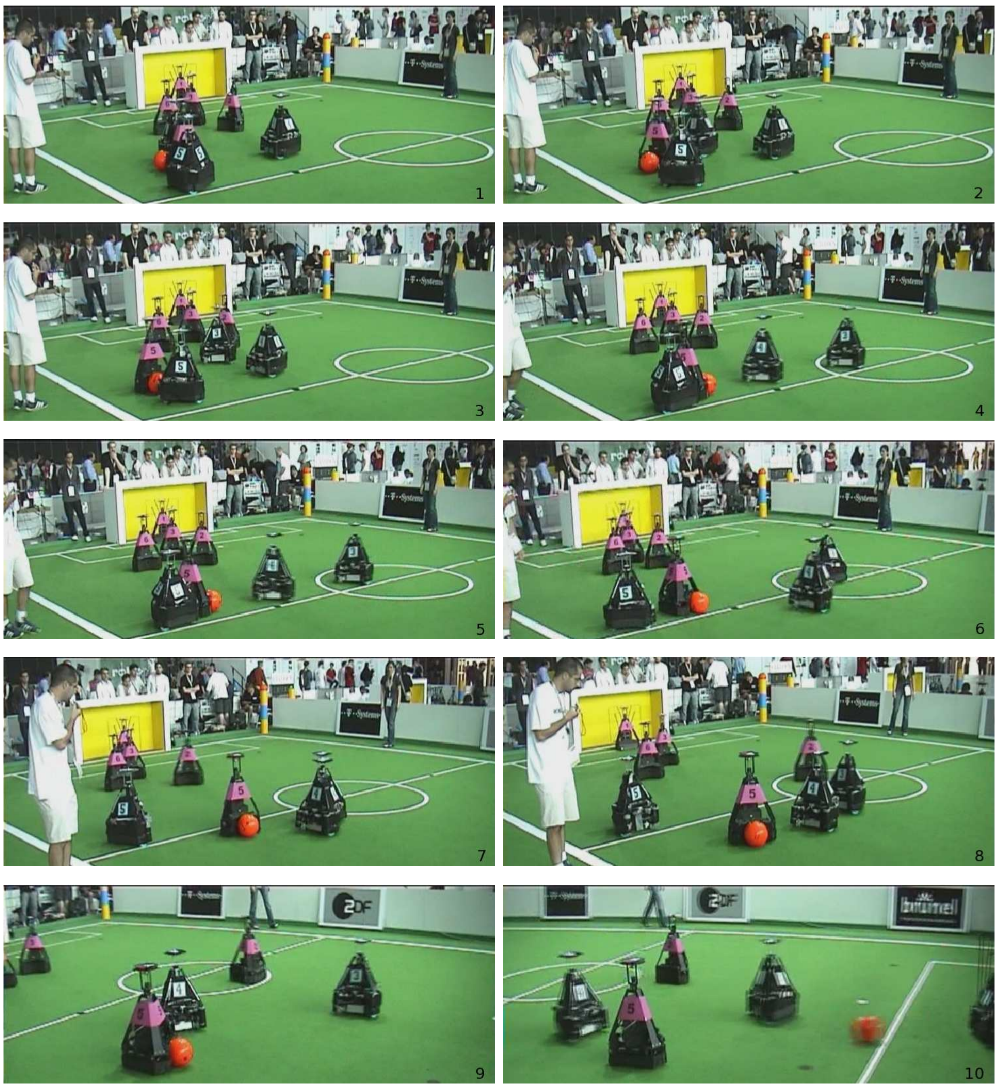


Figure A.10: Scenarios of a successful dribbling in the game against the AIS/BIT robots at the RoboCup World Cup 2006 in Bremen. The Attempto soccer robot No. 5 catches the ball in the neighborhood of two opponents, dribbles the ball out of this tight situation, avoids the third approaching opponent and finally shoots towards the goal. The images are sorted from left to right and from top to bottom.

Bibliography

[1] RoboCup MSL Technical Committee 1997-2006. Middle size robot league rules and regulations for 2006. http://www.sparrows.uni-ulm.de/docs/msl2006final.pdf, September 2006.

[2] RoboCup MSL Technical Committee 1997-2008. Middle size robot league rules and regulations for 2008. http://www.er.ams.eng.osaka-u.ac.jp/robocup-mid/, September 2008.

[3] Kontro AG. coolmonster/p3&c3 manual pisa slot pc pentium. http://de.kontron.com/support/?productId=232, September 2008.

[4] Maxon Motor AG. Dc motors online catalogue. http://www.maxonmotor.de/en/dc_motor.asp, September 2008.

[5] A. Aguiar, A. Atassi, and A. Pascoal. Regulation of a nonholonomic dynamic wheeled mobile robot with parametric modeling uncertainty using lyapunov functions. In *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, December 2000.

[6] A. Aguiar and A. Pascoal. Stabilization of the extended nonholonomic double integrator via logicbased hybrid control: An application to point stabilization of mobile robots. In *Proceedings of the 6th International IFAC Symposium on Robot Control*, Vienna, Austria., 2000.

[7] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotovic. Path-following for non-minimum phase systems removes performance limitations. In *IEEE Transactions on Automatic Control*, volume 50(2), pages 234–239, 2005.

[8] J. C. Alexander and J. H. Maddocks. On the kinematics of wheeled mobile robots. *Autonomous robot vehicles*, pages 5–24, 1990.

[9] F. Allgöwer, T. A. Badgwell, J. S. Qin, J. B. Rawlings, and S. J. Wright. Nonlinear predictive control and moving horizon estimation-an introductory overview. *Advances in control, highlights of ECC'99*, pages 391–449, 1999.

[10] A. Astolfi. Exponential stabilization of a wheeled mobile robot via discontinuous control. *Dynamic Systems, Measurement, and Control*, 121(1), 1999.

[11] M. Bak, N. K. Poulsen, and O. Ravn. Path following mobile robot in the presence of velocity constraints. Technical report, Informatics and Mathematical Modeling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2001.

[12] Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley, 2001.

[13] A. Bauchspiess. A predictive algorithm with fine interpolation for vision-guided robots. In *Proceedings of the 5th IFAC Workshop*, pages 97–102, 1998.

[14] R. Bellman. *Dynamic Programming*. Princeton Univ Pr, Princeton, New Jersey, 1957.

[15] J. T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[16] T. Binder, L. Blank, H.G. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J.P. Schlöder, and O.v. Stryk. Introduction to model based optimization of chemical processes on moving horizons. In M. Grötschel, S.O. Krumke, and J. Rambau, editors, *Online Optimization of Large scale Systems: State of the Art*, pages 295–340. Springer, 2001.

[17] R. P. Bithmead, V. Wertz, and M. Gerers. *Adaptive Optimal Control: The Thinking Man's G.P.C.* Prentice Hall Professional Technical Reference, New York, 1991.

[18] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, pages 1–51, 1995.

[19] A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, editors. *RoboCup 2005: Robot Soccer World Cup IX*. Springer-Verlag, 2006.

[20] R. W. Brockett. Asymptotic stability and feedback stabilization. *Differential Geometric Control Theory*, pages 181–191, 1983.

[21] A. E. Bryson and Y. C. Ho. *Applied Optimal Control*. Hemisphere Publ. Corp., Washington D.C., 1975.

[22] H. Chen. Stability and robustness considerations in nonlinear model predictive control. *Technical Report VDI Reihe. 8 Nr. 674*, 1997.

[23] H. Chen and F. Allgöwer. nonlinear model predictive control schemes with guaranteed stability. *Nonlinear Model Based Process Control*, pages 465–494, 1998.

[24] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1218, 1998.

[25] D. Chmielewski and V. Manousiouthakis. On constrained infinite-time linear quadratic optimal control. *Systems and Control Letters*, 29(3):121–129, 1996.

[26] V. Ciesielski and S. Y. Lai. Developing a dribble-and-score behaviour for robot soccer using neuro evolution. In *Proceedings of the Workshop on Intelligent and Evolutionary Systems*, pages 70–78, Dunedin, New Zealand, 2001.

[27] A. S. conceicao, A. P. Moreira, and P. j. Costa. Trajectory tracking for omni-directional mobile robots based on restrictions of the motor's velocities. In *Proceedings of the 8th International IFAC Symposium on Robot Control*, 2006.

[28] B. D. Damas, P. U. Lima, and L. M. Custódio. A modified potential fields method for robot navigation applied to dribbling in robotic soccer. In *RoboCup 2002: Robot Soccer World Cup VI*, pages 65–77, Fukuoka, Japan, 2003.

[29] R. DÁndrea, T. Kalmár-Nagy, P. Ganguly, and M. Babish. The cornell robocup team. In *RoboCup 2000: Robot Soccer World Cup IV*, pages 41–51, Berlin, 2001. Springer-Verlag.

[30] B. dÁndrea Novel, G. Bastin, and G. Campion. Control of nonholonomic wheeled mobile robots by state feedback linearization. *International Journal of Robotics Research*, 14(6):543–559, 1995.

[31] C. Canudas de Wit, H. Khennouf, C. Samson, and O. J. Sordalen. Recent trends in mobile robotics. *Nonlinear Control Design for Mobile Robots*, 11:121–156, 1993.

[32] C. Canudas de Wit, B. Siciliano., and G. Bastin. *Theory of Robot control*. Springer-Verlag, London, 1996.

[33] C. Canudas de Wit and O. J. Srdalen. Exponential stabilization of mobile robots with nonholonomic constraints. *IEEE Transactions on Automatic Control*, 37, 1992.

[34] F. Diaz del Rio, G. Jimenez, J. L. Sevillano, S. Vicente, and A. Civit Balcells. A generalization of path following for mobile robots. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, volume 1, pages 7 – 12, 1999.

[35] F. Díaz del Río, G. Jiménez, J. L. Sevillano, S. Vicente, and A. Civit-Balcells. A generialization of path following for mobile robots. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, volume 1, pages 7–12, 1999.

[36] O. Diegel, A. Badve, G. Bright, J. Potgieter, and S. Tlale. Improved mecanum wheel design for omni-directional robots. In *Proceedings of the 2002 Australasian Conference on Robotics and Automation*, Auckland, 27-29 November 2002.

[37] I. Doroftei, V. Grosu, and V. Spinu. Omnidirectional mobile robot - design and implementation. *Bioinspiration and Robotics: Walking and Climbing Robots*, pages 511–528, September 2007.

[38] D. Douxchamps, D. Dennedy, and G. Peters. Sourceforge.net: 1394-based dc control library. http://sourceforge.net/projects/libdc1394, September 2008.

[39] M. Egerstedt, X. Hu, and A. Stotsky. Control of a car-like robot using a virtual vehicle approach. In *Proceedings of the 37th Conference on Decision and Control*, pages 1503–1507, 1998.

[40] M. Egerstedt, X. Hu, and A. Stotsky. Control of mobile platforms using a virtual vehicle approach. *IEEE Transactions on Automatic Control*, 46:1777–1782, 2001.

[41] J. Van Eijck and G. Corrente. Sourceforge.net: The robocup msl refbox. http://sourceforge.net/projects/msl-refbox, September 2008.

[42] P. Encarnacao and A. Pascoal. Combined trajectory tracking and path following: an application tothe coordinated control of autonomous marine craft. In *Proceedings of the 40th IEEE Conference on Decision and Control*, volume 1, pages 964–969, 2001.

[43] J. Borenstein B. Everett and L. Feng. *Navigating Mobile Robots: Systems and Techniques*. A K Peters, Ltd, MA, USA, 1996.

[44] R. Findeisen. *Nonlinear Model Predictive Control a Sampled-Data Feedback Perspective*. PhD thesis, Universität Stuttgart, 2004.

[45] R. Findeisen. Challenges and opportunities in predictive sampled data open-loop feedback control. *OPTEC Tutorial Workshop on Nonlinear Model Predictive Control*, 2007.

[46] R. Findeisen and F. Allgöwer. An introduction to nonlinear model predictive control. In *21st Benelux Meeting on Systems and Control*, Veldhoven, Netherlands, 2002.

[47] F.A.C.C. Fontes. A general framework to design stabilizing nonlinear model predictive controllers. *Systems and Control Letters*, 42(2):127–143, 2001.

[48] Fraunhofer Institute for Autonomous Intelligent Systems. The aisvision omnidirectional camera system. http://www.ais.fraunhofer.de/BE/volksbot/AISVision-eng.pdf, September 2008.

[49] L. Freeston. Applications of the kalman filter algorithm to robot localization and world modeling. Technical report, University of Newcastle, NSW, Australia, 2002.

[50] T. Gabel and M. Riedmiller. Learning a partial behavior for a competitive robotic soccer agent. *Künstliche Intelligenz*, 20(2):18–23, 2006.

[51] S. S. Ge and F. L. Lewi. *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications*. CRC Press, 2006.

[52] A. Gelb. *Applied Optimal Estimation*. Cambridge, 1974.

[53] R. Ghabcheloo, A. Pascoal, and C. Silvestre. Nonlinear coordinated path following control of multiple wheeled robots with communication constraints. In *Proceedings of the 12th International Conference on Advanced Robotics*, pages 657–664, 2005.

[54] Allied Vision Technologies GmbH. The marlin f-046c vga camera ieee 1394. http://www.alliedvisiontec.com/avt-products/cameras/marlin/f-046-b-bs-c.html, September 2008.

[55] Lippert Automationstechnik GmbH. The thunderbird mini-itx motherboard. http://www.lippert-at.com/index.php?id=95, September 2008.

[56] TRAPOROL GmbH. All-side roller ∅ 80 mm - driven. http://www.traporol.de/arg-80-01.html, September 2008.

[57] J. Godhavn and O. Egeland. A lyapunov approach to exponential stabilization of nonholonomic systems in power form. *IEEE Transactions on Automatic Control*, 42(7):1028–1032, 1997.

[58] G. Grimm, M.J. Messina, S.E. Tuna, and A.R. Teel. Model predictive control: for want of a local control lyapunov function, all is not lost. *IEEE Transactions on Automatic Control*, 50:546–558, 2005.

[59] D. Gu and H. Hu. Receding horizon tracking control of wheel mobile robots. *IEEE Transaction on Control Systems Technology*, 14(4), July 2006.

[60] R. Hafner and M. Riedmiller. Neural reinforcement learning controllers for a real robot application. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 2098–2103, Roma, Italy, 2007.

[61] S.-P. Han. Superlinearly convergent variable metric algorithms for general nonlinear programming problems. *Mathematical Programming 11*, pages 263–282, 1976.

[62] B. Hassibi, A. Sayed, and T. Kailath. *Indefinite Quadratic Estimation and Control - A Unified Approach to H2 and H Infinity Theories*. Society for Industrial and Applied Mathematics, Philadelphia, 1999.

[63] B. Hecht, M. Simon, and O. Tenchio. A neural coach for teaching robots using diagrams. http://page.mi.fu-berlin.de/∼rojas/2005/NeuralCoach_Hecht_Rojas.pdf/, 2005.

[64] P. Heinemann. *Cooperative Multi-Robot Soccer in a Highly Dynamic Environment*. PhD thesis, Faculty of Information and Cognition Science, University of Tübingen, September 2008.

[65] P. Heinemann, J. Haase, and A. Zell. A combined monte-carlo localization and tracking algorithm for robocup. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1535–1540, Beijing, China, 2006.

[66] P. Heinemann, M. Plagge, A. Treptow, and A. Zell. Tracking dynamic objects in a robocup environment in a robocup environment - the attempto tübingen robot soccer team. *RoboCup - 2003: Robot Soccer World Cup VII, Lecture Notes in Computer Science*, 2004.

[67] P. Heinemann, T. Rückstieß, and A. Zell. Fast and accurate environment modeling using omnidirectional vision. *Dynamic Perception*, 2004.

[68] P. Heinemann, T. Rückstiess, and A. Zell. Fast and accurate environment modelling using omnidirectional vision. *Dynamic Perception*, pages 9–14, 2004.

[69] R. Immel. *Development and Validation of a Mobile Autonomous Omnidirectional Soccer Robot*. PhD thesis, Faculty of Mechanical Engineering, University of Darmstadt, 2006.

[70] MobileRobots Inc. Mobilerobots: Research & university robots, software & accessories. http://www.activrobots.com/ROBOTS/, September 2008.

[71] MobileRobots Inc. The pioneer at platform. http://www.activrobots.com/ROBOTS/, September 2008.

[72] MobileRobots Inc. The pioneer dx platform. http://www.activrobots.com/ROBOTS/, September 2008.

[73] G. Indiveri, J. Paulus, and P. G. Plöger. Motion control of swedish wheeled mobile robots in the presence of actuator saturation. In *Proceedings of the 10th annual RoboCup International Symposium*, 2006.

[74] A. Jadbabaie and J. Hauser. On the stability of receding horizon control with a general terminal cost. *IEEE Transactions on Automatic Control*, 50:674–678, 2005.

[75] A. Jadbabaie, J. Yu, and J. Hauser. Stabilizing receding horizon control of nonlinear systems: a control lyapunov function approach. In *Proceedings of the American Control Conference*, volume 3, pages 1535–1539, San Diego, CA, 1999.

[76] P. Jonker, B. Terwijn, J. Kuznetsov, and B. van Driel. Algorithmic function of the clockwork orange robot soccer team. In *Proceedings of the 6th International Workshop on the Algorithmic Foundations of Robotics*, pages 1–10, Zeist/Utrecht, Netherland, 2004.

[77] T. Kalmár-Nagyaand, R. DÁndrea, and P. Ganguly. Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle. *Robotics and Autonomous Systems*, 46(1):47–64, 2004.

[78] K. Kanjanawanishkul, X. Li, and A. Zell. Nonlinear model predictive control of omnidirectional mobile robot formations. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems,* pages 41–47, 2008.

[79] K. Kanjanawanishkul, X. Li, and A. Zell. Nonlinear model predictive control of omnidirectional mobile robot formations. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems,* pages 41–48, Baden-Baden, Germany, 2008.

[80] A. Karol and M. Williams. Distributed sensor fusion for object tracking. In *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *Lecture Notes in Computer Science,* pages 504–511. Springer, 2006.

[81] S. S. Keerthi and E. G. Gilbert. Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations. *Optimization Theory and Applications*, 57(2):265–293, 1988.

[82] T. Keviczky and G. J. Balas. Software-enabled receding horizon control for autonomous uav guidance. In *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, San Francisco, California USA, 2005.

[83] T. Keviczky, P. Falcone, F. Borrelli, J. Asgari, and D. Hrovat. Predictive control approach to autonomous vehicle steering. In *Proceedings of the American Control Conference*, Minneapolis, Minnesota USA, 2006.

[84] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, Upper Saddle River, New Jersey, 3. edition edition, 2002.

[85] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 500–505, 1985.

[86] B. Kim, D. Necsulescu, and J. Sasiadek. Model predictive control of an autonomous vehicle. In *Proceedings of the 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, volume 2, pages 1279–1284, 2001.

[87] S. Kubina. *The TMC200 Triple Motor Controller for DC Motors - Overview*. Fraunhofer Institute for Autonomous Intelligent Systems, August 2006. http://www.ais.fraunhofer.de/BE/volksbot/TMC200-eng.pdf.

[88] F. Kühne, J.M.G. da Silva, and W.F. Lages. Mobile robot trajectory tracking using model predictive control. In *Proceedings of the 2005 Latin-American Robotics Symposium*, 2005.

[89] F. Kühne, W.F. Lages, and J.M.G. da Silva. Point stabilization of mobile robots with nonlinear model predictive control. In *Proceedings of the 2005 IEEE International Conference on Mechatronics and Automation*, volume 3, pages 1163–1168, 2005.

[90] K. Kurihara, N. Nishiuchi, J. Hasegawa, and K. Masuda. Mobile robots path planning method with the existence of moving obstacles. In *Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation*, Catania, Italy, September 2005.

[91] C. C. T. Kwok and D. Fox. Map-based multiple model tracking of a moving object. In *RoboCup 2004: Robot Soccer World Cup VIII*, pages 18–33. Springer, 2005.

[92] G. Lakemeyer, E. Sklar, D. G. Sorrenti, and T. Takahashi, editors. *RoboCup 2006: Robot Soccer World Cup X*. Springer-Verlag, 2007.

[93] L. Lapierre, P. Lépinay, and R. Zapata. Simultaneous path following and obstacle avoidance control of a unicycle-type robot. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 2617–2622, Roma, Italy, 2007.

[94] M. Lauer, S. Lange, and M. Riedmiller. Modeling moving objects in a dynamically changing robot application. *Advances in Artificial Intelligence*, pages 219–303, 2005.

[95] X. Li, K. Kanjanawanishkul, and A. Zell. Nonlinear model predictive control of an omnidirectional mobile robot. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems*, pages 92–99, Baden-Baden, Germany, 2008.

[96] X. Li, M. Wang, and A. Zell. Dribbling control of omnidirectional soccer robots. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 2623–2628, Roma, Italy, 2007.

[97] X. Li and A. Zell. Path following control for a mobile robot pushing a ball. In *Proceedings of the 8th IFAC Symposium on Robot Control*, Bologna, Italy, 2006.

[98] X. Li and A. Zell. Motion control of an omnidirectional mobile robot. In *Proceedings of the 4th International Conference on Informatics in Control, Automation and Robotics*, Angers, France, 2007.

[99] X. Li and A. Zell. Motion control of an omnidirectional mobile robot. In *Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics, Robotics and Automation*, pages 125–132, Angers, France, 2007.

[100] X. Li and A. Zell. Nonlinear predictive control of an omnidirectional robot dribbling a rolling ball. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 1678–1683, Pasadena, California, USA, 2008.

[101] X. Li and Andreas Zell. $_{\text{infinity}}$ filtering for a mobile robot tracking a free rolling ball. In *RoboCup 2006: Robot Soccer World Cup X*, pages 296–303, Bremen, Germany, 2006.

[102] Y. Liu, X. Wu, J. J. Zhu, and J. Lew. Omni-directional mobile robot controller design by trajectory linearization. In *Proceedings of the 2003 American Control Conference*, 2003.

[103] J. Majohr and T. Buch. *Advances in Unmanned Marine Vehicles*. IEEBook, London, UK, 2006.

[104] F. Martinsena, L. T. Bieglerb, and B. A. Foss. A new optimization algorithm with application to nonlinear mpc. *Journal of Process Control*, 14:853–865, 2004.

[105] K. Maček, I. Petrović, and R. Riegwart. A control method for stable and smooth path following of mobile robots. In *Proceedings of the 2nd European Conference on Mobile Robots - ECMR 2005*, pages 128–133, Ancona, Italy, 2005.

[106] Peter S. Maybeck. *Stochastic models, estimation, and control*, volume 1 of *Mathematics in Science and Engineering*. Academic Press, 1979.

[107] D. Mayne. Nonlinear model predictive control: Challenges and opportunities. *Nonlinear Model Predictive Control*, pages 24–44, 2000.

[108] D.Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, pages AC–35(7):814–824, 1990.

[109] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, pages 26(6):789–814, 2000.

[110] E. S. Meadows and J. B Rawlings. Receding horizon control with an infinite horizon. In *Proceedings of the 1993 American Control Conference*, San Francisco, 1993.

[111] A. Micaelli and C. Samson. Trajectory tracking for unicycle-type and two-steering-wheels mobile robots. Technical Report RR-2097, INRIA, Sophiia-Antipolis, France, 1993.

[112] H. Michalska and D. Q. Mayne. Receding horizon control of nonlinear systems without differentiability of the optimal value function. *Systems & Control Letters*, 8(1):123–130, 1991.

[113] H. Michalska and D.Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control*, pages 38(11):1623–1633, 1993.

[114] A. Mojaev and A. Zell. Tracking control and adaptive local navigation for nonholonomic mobile robot. In *Proceedings of the IAS-8 conference*, 2004.

[115] P. Morin and C. Samson. Motion control of wheeled mobile robots. *Springer Handbook of Robotics*, pages 799–826, 2008.

[116] P. F. Muir and C. P. Neuman. Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot. *Autonomous Robot Vehicles*, 1990.

[117] H. Müller, M. Lauer, R. Hafner, S. Lange, A. Merke, and M. Riedmiller. Making a robot learn to play soccer using reward and punishment. *KI 2007: Advances in Artificial Intelligence*, pages 220–234, 2007.

[118] W. Naeem, T. Xu, R. Sutton, and A. Tiano. The design of a navigation, guidance, and control system for an unmanned surface vehicle for environmental monitoring. *Proceedings of the I MECH E Part M*, 222:67–79, 2008.

[119] K.M. Nagpal and P. P. Khargonekar. Filtering and smoothing in an h infinity setting. *IEEE Transactions on Automatic Control*, 36:152 – 166, 1991.

[120] D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, editors. *RoboCup 2004: Robot Soccer World Cup VIII*. Springer-Verlag, 2005.

[121] J. I. Neimark and N. A. Fufaev. Dynamics of nonholonomic systems. *Translations of mathematical monographs*, 33, 1972.

[122] G. De Nicolao, L. Magni, and R. Scattolini. Stabilizing receding-horizon control of nonlinear timevarying systems. *IEEE Transactions on Automatic Control*, 43(7):1030–1036, 1998.

[123] W. Nisticò, M. Hebbel, T. Kerkhof, and C. Zarges. Cooperative visual tracking in a team of autonomous mobile robots. In *RoboCup 2006: Robot Soccer World Cup X*, pages 146–157. Springer, 2007.

[124] G. Oriolo, A. Luca, and M. Vendittelli. Wmr control via dynamic feedback linearization: Design, implementation and experimental validation. *IEEE Transactions on Control Systems Technology*, 10(6), 2002.

[125] A. A. Pedro and H. J. Pedro. Logic-based switching control for trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. In *Proceedings of the 2004 American Control Conference ACC*, volume 6, pages 3004–3010, 2004.

[126] S. Petrov. Computer vision, sensorfusion und verhaltenssteuerung für fussball-roboter. Master's thesis, Freie Universität Berlin, Institute für Informatik, 2004.

[127] F. Pourboghrat and M. P. Karlsson. Adaptive control of dynamic mobile robots with nonholonomic constraints. *Computers and Electrical Engineering*, 28(4), 2002.

[128] M. J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. *Lecture Notes in Mathematics*, 630:144–157, 1978.

[129] J. Primbs, V. Nevisti, and c Doyle. Nonlinear optimal control: A control lyapunov function and receding horizon perspective. *Asian Journal of Control*, 1(1):14–24, 1999.

[130] J. Primbs and V. Nevistic. Feasibility and stability of constrained finite receding horizon control. *Automatica*, 36:965–971, 2000.

[131] J. Primbs, V. Nevistic, and J. Doyle. A receding horizon generalization of pointwise min-norm controllers. *IEEE Transactions on Automatic Control*, 45:898–909, May 2000.

[132] O. Purwin and R. D' Andrea. Trajectory generation and control for four wheeled omnidirectional vehicles. *Robotics and Autonomous Systems*, 54(1):13–22, 2006.

[133] O. Purwin and R. D'Andrea. Cornell big red 2003. *Robocup 2003: Robot Soccer World Cup VII, Lecture Notes in Artificial Intelligence*, 2004.

[134] S. J. Qin and T. A. Badgwell. An overview of nonlinear model predictive control applications. *Nonlinear Model Predictive Control*, pages 369–393, 2000.

[135] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

[136] F. M. Raimondi and M. Melluso. A new fuzzy dynamics controller for autonomous vehicles with nonholonomic constraints. *Robotics and Autonomous Systems*, 52, 2005.

[137] M. Riedmiller and A. Merke. Using machine learning techniques in complex multi-agent domains. http://lrb.cs.uni-dortmund.de/ merke/research/-papers/zif_book02.pdf, September 2003.

[138] RoboCup. Official homepage. http://www.robocup.org, September 2008.

[139] RoboCup. Overview: What is robocup. http://www.robocup.org/overview/21.html, September 2008.

[140] R. Rojas and A. G. Förster. *Holonomic Control of a Robot with an Omnidirectional Drive*. BöttcherIT Verlag, Bremen, 2006.

[141] J. Ruizdel-Solar and P. Vallejos. Motion detection and tracking for an aibo robot using camera motion compensation and kalman filtering. *Lecture Notes in Computer Science 3276 (RoboCup Symposium 2004)*, 2004.

[142] M. Saito and T. Tsumura. Collision avoidance among multiple mobile robots–a local approach based on non-linear programming. *Trans. of the Institute of Systems, Control and Information Engineers*, 3(8):252–260, 1990.

[143] C. Samson. Path following and time-varying feedback stabilization of a wheel robot. In *Proceedings of the International Confdrence ICARCV92*, pages 172–179, Singapore, 1992.

[144] C. Samson. Time-varying feedback stabilization of car like wheeled mobile robot. *International Journal of Robotics Research*, 12(1):55–64, 1993.

[145] C. Samson and K. Ait-abderrahim. Feedback control of a nonholonomic wheeled cart in cartesian space. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, volume 2, pages 1136 – 1141, 1991.

[146] T. Schmitt, R. Hanek, S. Buck, and M. Beetz. Cooperative probabilistic state estimation for vision-based autonomous soccer robots. In *Proceedings of the 23rd DAGM-Symposium*, pages 321–328, 2001.

[147] P.O.M. Scokaert and J.B. Rawlings. Constrained linear quadratic regulation. *IEEE Transactions on Automatic Control*, 43:1163–1169, 1998.

[148] A. Scolari Conceição, P. j. Costa, and A. P. Moreira. Control and model identification of a mobile robot's motors based in least squares and instrumental variable methods. In *Proceedings of the 11st International Conference on Methods and Models, in Automation and Robotics*, 2005.

[149] M. Seyr and S. Jakubek. Mobile robot predictive trajectory tracking. In *Proceedings of the 2005 ICINCO*, pages 112–119, 2005.

[150] X. Shen and L. Deng. Game theory approach to discrete H_∞ filter design. *IEEE Transaction on Signal Processing*, 45(4), 1997.

[151] R. Siegwart and I. R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. The MIT Press, 2004.

[152] D. Soetanto and A. Pascoal L. Lapierre. Adaptive, non-singular path-following control of dynamic wheeled robots. In *Proceedings of the 42th IEEE International Conference on Decision and Control*, volume 2, pages 1765–1770, 2003.

[153] Appin Knowledge Solutions. *Robotics*. Infinity Science Press, 2007.

[154] P. Spellucci. A new technique for inconsistent problems in the sqp method. *Mathematical Methods of Operations Research*, 47:355–500, 1998.

[155] P. Spellucci. An "sqp" method for general nonlinear programs using only equality constrained subproblems. *Mathematical Programming*, 82:413–448, 1998.

[156] K. Terashima, T. Miyoshi, J. Urbano, and H. Kitagawa. Frequency shape control of omni-directional wheelchair to increase user's comfort. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, 2004.

[157] C.-C. Tsai, H.-C. Huang, T.-S. Wang, and C.-M. Chen. System design, trajectory planning and control of an omnidirectional mobile robot. In *Proceedings of the 2006 CACS Automatic Control Conference*, 2006.

[158] S. G. Tzafestas. *Advances in Intelligent Autonomous Systems*. Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[159] J. Urbano, K. Terashima, T. Miyoshi, and H. Kitagawa. Velocity control of an omni-directional wheelchair considering user's comfort by suppressing vibration. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3169 – 3174, 27-29 November 2005.

[160] K. Watanabe. Control of an omnidirectional mobile robot. In *Proceedings of the 2th International Conference on Knowledge-Based Intelligent Electronic Systems*, 1998.

[161] T. Weigel, A. Kleiner, F. Diesch, M. Dietl, J.-S. Gutmann, B. Nebel, P. Stiegeler, and B. Szerbakowski. Cs freiburg 2001. In Andreas Birk, Silvia Coradeschi, and Satoshi Tadokoro, editors, *RoboCup 2001 : Robot Soccer World Cup V*, volume 2377 of *Lecture Notes in Computer Science*, pages 26–38. Springer, 2001.