ISSN 0280-5316
ISRN LUTFD2/TFRT--5854--SE

Real-time Trajectory Generation and Control of a Semi-Omnidirectional Mobile Robot

Sofie Nilsson

Department of Automatic Control
Lund University
May 2010

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
May 2010

Document Number
ISRN LUTFD2/TFRT--5854--SE

Author(s)
Sofie Nilsson

Supervisor
Christian Conette Fraunhofer IPA, Stuttgart Germany.
Rolf Johansson Automatic Control, Lund Sweden (Examiner)

Sponsoring organization

Title and subtitle
Real-time Trajectory Generation and Control of a Semi-Omnidirectional Mobile Robot.
(Realtidstrajektoriegenerering och reglering av semi-omnidirektionell mobil robot)

Abstract
When controlling a wheeled mobile robot with four independently steerable driving wheels, the control of the wheel coordination must be handled. Both the direction and velocity of the wheels must be coordinated to allow for proper operation of the robot. The focus of this work is on the coordination of the wheel directions. Such coordination is mostly done by solving constraint equations of the system kinematics, but when the demands on the coordination are high, it is sometimes necessary to include the steering dynamics in the coordination control. With dynamics included the complexity of the wheel coordination increases, since constraints dependent on required angle changes and current velocities must be fulfilled. By calculating the dynamic limitations in each control cycle, the steering limit for the whole wheel base within the current control cycle can be found. With use of such wheel base limit, followable and coordinated wheel trajectories can be generated online. This thesis includes the construction of a dynamic model for inclusion of the steering dynamic limitations affecting the performance the most, the construction of the online trajectory generation idea, as well as implementation and validation on the real target wheeled mobile robot platform.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
70

Recipient's notes

Security classification

Contents

Acknowledgements

The work behind this report was carried out at Fraunhofer Institute of Production technology and Automation (IPA) in Stuttgart, Germany, during the spring and summer of 2009. I would specially like to thank my supervisor at IPA Christian Connette for giving me the opportunity to do this thesis, for guiding me through the process, and for always being willing to answer my questions. I would also like to thank Prof. Karl-Erik Årzen in Lund and PhD student Karl Berntorp for many valuable comments on this report. And at last many thanks to all my colleges at IPA for making my time in Stuttgart a great one.

Sofie Nilsson

Chapter 1

Introduction

The main work of this thesis was done during 6 months at Fraunhofer Institute of Production technology and Automation (IPA) in Stuttgart, Germany. The target robot Care-O-bot©3, described in detail in Chapter 3, is developed in an internal project at IPA, with the objective to design a service robot able to assist humans in everyday household work. Care-O-bot©3 is a wheeled mobile robot, made to be a challenging but realistic stage of development between the household robots currently on the market, which are mainly autonomous vacuum cleaners, and more advanced walking robotic housekeepers.

In order to make a wheeled robot as flexible as possible, some kind of wheel platform that is able to move and rotate in multiple directions, without a need for time and space consuming maneuvers, is preferred as the mobile base. Since all other components of the mobile robot are mounted on the wheel base, the accuracy of the mobile platform is highly critical for a good overall performance. It is therefore desirable to control the mobile base in a way that guarantees well known behavior and optimal performance.

This thesis focuses on the lowest level of the mobile platform control, the under-carriage control, which handles the transformation between, and control of, the robot velocities and the wheel motions. Robot velocity commands generated in supervisory control layers are inputs to the under-carriage control. Other robot information, such as the robot position, is covered by other parts of the robot control and therefore unknown and disregarded in the under-carriage control.

1.1 Task

The given mission of this thesis was to including system dynamics to the Care-O-bot©3 wheel platform control, a vast assignment with large possibilities to define the exact task freely within the given field. In order to choose what to focus on, the work was started with an investigation of the entire platform control and dynamics. This exploratory phase was necessary to find and limit the final task statement, but lies outside the main work presented in this report. Since all the encountered research issues are highly relevant for the overall project, they are mentioned briefly as suggestions to future work in Chapter 7.

The final task is strongly related to the above mentioned statement: A well known platform behavior is critical for the entire system performance. In the existing version of the platform control, only the system kinematics is taken into account, which sometimes causes loss of wheel coordination and undeterministic behavior. The task therefore became to improve wheel coordination by integrating steering dynamics to the control system. The work was to specify relevant dynamics, determine how to account for the dynamics in the control structure, implement and verify the proposed principles on the real system.

For system analysis and evaluation a combination of recordings from tests on the robot, analysis of the control implementation in C++, and simulations in Matlab/Simulink mostly based on existing models, are used. The complexity of the platform causes difficulties to formulate a complete mathematical system model when dynamics are included, the analysis and evaluation is therefore mostly based on logical reasoning.

1.2 Outline

As a more technical, yet general introduction, a short overview of the field of mobile robotics is given in Chapter 2. Chapter 3 gives an overview of the targeted robot and a detailed description of the mobile platform concerning characteristics, control structure, and problem formulation.

Chapter 4 contains a presentation of the existing system model, selection of dynamics to include, and construction of the dynamic model. Chapter 5 presents the new platform control system including specifications, control architecture, and description of the implemented algorithm. Results of the implementation on the robot are presented in Chapter 6, recorded measurements from both the new and old control implementation is presented and compared.

In Chapter 7 conclusions drawn from evaluation of the new control idea are presented together with ideas regarding future work.

Chapter 2

Wheeled Robots

This chapter gives an introduction to the fields of mobile and service robotics, including current technical standpoints, products, and goals; common approaches to mobile robot control systems; highlighted areas of research; an overview of relevant related fields.

2.1 Background

When mobile robots and service robots are mentioned, people tend to refer to science fiction where humanoid robots more or less takes over the world. Even though humanoids exist, they are far from fully adaptable to the society and intelligent enough to be compared to a human, so the people dreaming about having a robot doing everything for them, will most likely have to wait for some more decades. Walking robots are getting more and more advanced, and some quite impressive concepts exist, like BostonDynamics Big Dog [22] and Hondas ASIMO [23]. Even though those might appear very impressive, especially in advertisement movies, they are still years from being complete and robust products.

When it comes to the field of service robotics, the products on the market today can barely be defined as robots; advanced household equipments would be a better denomination. Examples of such robots are autonomous vacuum cleaners such as iRobots Roomba [24] and Electrolux Trilobite [25]. A generation shift straight from today's fairly unintelligent machines to a fully compatible humanoid serving as well as a human employee is quite unrealistic.

Instead, an intermediate generation that combines robustness and flexibility will be needed. The concept of wheeled robots might not be as flexible as a walking robot in its fully functional state, but offers a good solution

on the way, and can for most environments, given that many premises and households are handicap friendly, be flexible enough to perform most tasks.

Since the mobile base in context of mobility is in focus in this thesis, and the research of wheeled service robotics is quite narrow, the whole area of wheeled robots is considered a valid theoretical base. In order to draw parallels to techniques used in other types of robots, a summary of some different fields are given with relating parallels to the Care-O-bot©3 platform. Service robotics is only summarized shortly and a wider range of mobile robots are considered as related concept studies.

2.2 Structures and Classifications

The structure of wheeled robots can vary widely, not only in the task they are made to perform, but also in the number and type of wheels. In many cases the system model is constructed for an individual robot and the control routine is highly adapted and optimized to the specific application.

As for most other systems it is desirable to have a general representation even for wheeled mobile robots, so that the same derivations do not have to be reinvented for each new system. Such a general representation is the bicycle model, where each wheel pair is modeled as one single wheel. The bicycle model is widely used for simple kinds of wheeled robots that follow car like structures. When the wheel structure is composed with multiple independent steerable wheels of different types, the general bicycle model is no longer a sufficient representation of the full system kinematics [7].

The degree of mobility for a robot is a returning concept that describes the possible movements of the robot platform. The classification covers both fully omnidirectional platforms to platforms with very limited steering angles. The description also mentions the difference between omnidirectional and semi-omnidirectional platforms, which is important to consider when choosing a system model [7].

For a platform to be fully omnidirectional, it must at each instant of time be able to move in any direction, with no adaption needed. Such a platform can for example be achieved by the use of so called Swedish wheels, which is flexible but not so smooth due to uneven wheel ground contact. A more popular wheel is the caster wheel, which is not as complex as the Swedish wheels, and offers an even wheel ground contact. Conventional wheels are also widely used, both in combination with, and without caster wheels. With unlimited steering, conventional wheels can also move in any direction, but a reorientation procedure is needed. A platform that needs to reorient the wheels in order to move in any direction is defined as pseudo-omnidirectional.

2.3 Models

Despite the many possible different wheel compositions, wheeled mobile robots (WMR) have a common ground in that they all are mechanical systems including kinematic constraints that cannot be eliminated from the model equation [7].

The platform kinematic model is constructed from the platform design and the kinematic constraints. System representations with such purely kinematic models are widely used, see [7][4][8], and the derivation procedure is quite standardized.

In some cases it is desired to capture even the mobile platform dynamics in the system model. Within the publications in this area, the dynamics included are normally some variant of the energy flow, or the wheel ground contact forces and wheel torques, see [9][10][11]. For more extreme applications, more detailed dynamic models might be needed; one example is driving on loose soil, which is addressed in [14].

2.4 Control

The flexible structure of wheeled robots creates an interesting field for control research. In the case of omnidirectional and pseudo-omnidirectional platforms, there is a need for quite advanced control systems. The task is to control both the robot motions and the wheel coordination.

In many cases a system representation where the robot motion is translated to a motion around a rotation center at each instant, is used. Both the robot motion and the wheel coordination control is then done by control of the instantaneous center of motion (ICM). The ICM representation holds a number of singularities, some which must be considered and avoided in the control. The control can be done in numerous ways, for example with dynamic feedback linearization, as in [8][4]. The ICM can also be controlled via introduction of a pseudo-velocity together with constraint force control. The constraint forces are then defined from the wheel velocity coordination errors. The last mentioned approach originates from [10] where it was developed for general constrained mechanical systems, and was then applied in control of a wheeled robot by Reister and Unseren in [3].

When the platform dynamics is considered, it normally, as mentioned in Section 2.3, involves some variant of wheel ground contact forces. The dynamics is then controlled by some variant of force control [16][15].

2.5 Related Topics

When a robot is operating in dynamic environments, different levels of addaptable motion planning are needed. When the path is changing or not known from start, generation of motion trajectories online in real-time during operation, is sometimes the most efficient approach. The field of such trajectory computation is not at all limited to mobile robotics, thus publications in the area are mostly general for constrained mechanical systems. In [13] trajectories for reduced maps of constrained mechanical systems are generated by use of splines and nonlinear control. In [12], on-line trajectory generation for synchronization of multiple degrees of freedom under dynamic constraints is presented and applied in robot manipulator control. The work in this thesis is similar to the work in [12], but the target system differences cause substantial effects in the trajectory generation procedure.

Chapter 3

Care-O-bot©3

The first generation of Care-O-bot, built in 1998, was equipped with a touch screen but no arm. The robot was already at this stage able to move safely around humans performing simple transportation and information sharing tasks. This version, shown in Figure 3.1(a), was successfully used as museum guide robot [20].

For the next generation an arm, tiltable sensors, and adjustable walking supporters were added. The arm was equipped with a gripper, which enabled the robot to grasp objects, like plates and cups, as well as performing simple manipulator tasks. Care-O-bot II that can be seen in Figure 3.1(b), was built in 2002 and could be used as walking assistance [21].

(a) Care-O-bot I [20]. (b) Care-O-bot II [21].

Figure 3.1: The previous Care-O-bot generations.

The current version, Care-O-bot©3, is more or less an interactive butler, developed with the goal of a mobile robot able to assist humans in their daily life. Today the robot is able to move safely around humans, detect, grasp, and move household objects in an interactive way. With the ability to plan a collision free path in an environment with both stationary and dynamical objects, in combination with a highly flexible arm, the robot is a useful tool in any household. It can reach objects both from the floor and from high shelves, and even open doors blocking the path. Despite that the robot is intentionally designed to not look like a human, it is able to perform some human gestures like nodding and bowing for the purpose of intuitive user feedback [19].

(a) Care-O-bot III overview [18].

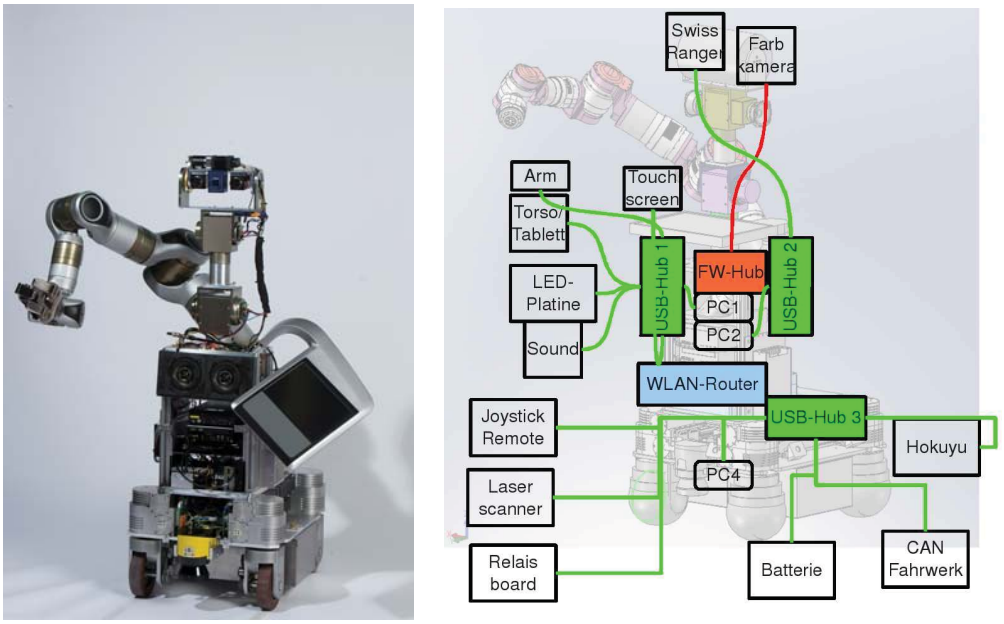
(b) Care-O-bot©3 in the laboratory kitchen.

Figure 3.2: Care-O-bot III.

In Figure 3.3(a) the robot is shown without its soft cover, allowing for a more detailed view of the robot's subsystems. As can be seen in the figure, Care-O-bot©3 consists of mainly 4 parts head, tray, arm, and mobile platform. The head is equipped with tiltable cameras, which with help of image processing provides the robot with information about the surrounding. The tray can be automatically positioned either in front of the robot or folded to the side, it is useful both for carrying objects on, and for human interaction via a built in touch screen. The light weight arm is equipped with a highly flexible hand, designed for grasping various things. The wheel platform, that is responsible for the mobility of the robot, is the target system for this thesis and described in detail in Section 3.1.

The robot is controlled by three onboard PCs, connected according to Figure 3.3(b). PC1 is responsible for the arm and tray control, PC2 handles

the head, including camera motion and image processing, and PC3 (PC4 in the figure) manages the control and planning of the mobile platform. All onboard PCs are using a Linux kernel and can be reached via WLAN.



(a) Care-O-bot©3 without its cover. (b) Subsystem connection diagram.

Figure 3.3: Care-O-bot III subsystem view.

3.1 The Mobile Platform

The Care-O-bot©3 platform is rectangular and approximately 70 cm long and 50 cm wide with one driven steerable wheel module in each corner. Each wheel module holds two motors, one for driving and one for steering, which are connected to the wheel via worm gears, allowing unlimited rotation in both steering and driving. The wheels have a radius of 75 mm, are supplied with solid rubber tires, and are positioned approximately 7,5 cm from their steer rotation center, as in Figure 3.5. The offset implies that each steering motion must be coordinated with a compensating drive motion [2].

The requested platform motions can be specified in different ways; by a person with a joystick, from a graphical user interface where paths in the surrounding are specified, or ordered by the arm control.

The rotational velocity of the two motors in each wheel module, are controlled by on-board Elmo motor controllers. The total four Elmo controllers

Figure 3.4: The Care-O-bot©3 platform [1].

are connected to the platform PC via CAN-bus. The platform computer is in charge of everything related to the platform motion, such as path planning, obstacle avoidance, communication, and much more. To not make the path planning unnecessarily complicated, only set points for the robot configuration, meaning linear and rotational velocities, are considered in the path planning [2]. The lowest level of the platform control, from here on referred to as the under-carriage control, translates the robot set points to wheel set points and controls the platform to follow the target values.

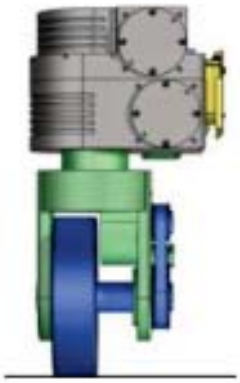


Figure 3.5: The Care-O-bot©3 wheel module [1].

3.1.1 Platform Control System

The under-carriage control system can be described by the block diagram in Figure 3.6. The commanded robot velocities and robot rotational rate

are decided on a supervisory control level and serve as the input vector $rob_{cmd} = [V_x, V_y, \dot{\theta}]_{cmd}$ to the under-carriage controller. The robot target generator block modifies the commanded robot velocities with help of estimations of the current velocities, rob_{est}, generated from Elmo measurement vectors in the robot estimator block, to smoothen the robot motions for safe operation. The wheel driveing and steering command generator translates the commanded robot velocities, rob_{target}, to corresponding wheel driving and steering references, v_{wheel} and ϕ^*. The steering angle position control, resulting in steering velocity commands, $\dot{\phi}^*$, is done independently for each wheel with impedance control in the steer control block. The wheel velocity command generator finally adds the steering velocity and drive velocity coupling and calculates the corresponding wheel rotational velocity commands, ω_{cmd}, for the Elmo drives.

The input vector is sent to the under-carriage controller, from the supervisory control layer, with an interval of approximately 20 ms, varying slightly depending on the command generator and the computational load. Each command to the under-carriage control corresponds to one control cycle, which implies that the robot references can change in each cycle, and that the wheel commands to the Elmo controllers are passed on with the same interval as the robot commands.

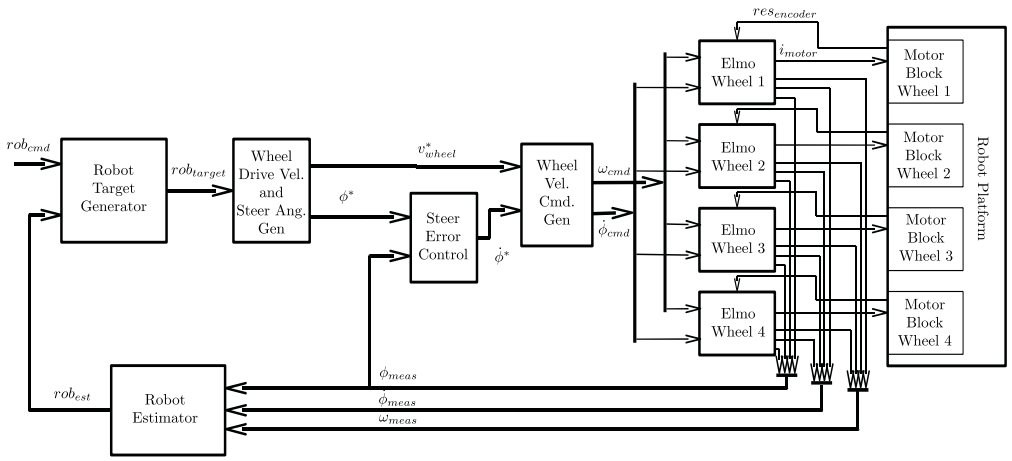


Figure 3.6: Block diagram for the under-carriage control.

The outputs from the under-carriage control computer to the Elmo wheel controllers are given in form of steering and driving velocity references for each wheel. The Elmo controllers hold two parallel velocity controllers. Each one is using a cascade structure, where the velocity is controlled with a reg-

ular PI controller in the outer loop, generating a torque command, which results in a current command to the inner current loop. The rotational rate of each motor is measured by encoders, allowing the motor velocities to be calculated and fed back, together with calculated motor positions, as measurements from the Elmo blocks. The measurements from the Elmo controllers are the only measurements available for the entire platform. Within the under-carriage controller, the measurements are used both directly for steering angle control, and as inputs to the robot state estimator, estimating the velocities and rotations of the whole platform. The Elmo modules can also be configured to send estimated measurements of the motor torques, but that feature is not used in the present version of the control.

3.1.2 Alternative Control Method

The under-carriage control structure described in Section 3.1.1 is the first out of two currently implemented under-carriage control alternatives. This first version is referred to as the geometrical variant, and the other version is called the spherical variant. As mentioned in the previous section, the wheel set points are in the geometrical variant calculated directly from the robot set points according to the steering angle constraint equations. The steering angles are then controlled independently to reach their targets.

The spherical control method is a bit more complex. Here the instantaneous center of motion (ICM), introduced in Chapter 2, for the robot is calculated. The estimated ICM is then controlled to reach its target and in the same time avoid singularities using a potential field controller. The ICM is here represented and controlled in spherical coordinates for computational singularity elimination. The spherical ICM position and velocities are mapped to the commanded steering velocities and steering angles. The steering angle changes are here results of the steering velocities, but the steering velocities are, due to the still purely kinematic system model, considered, from the coordination point of view, to follow their targets instantaneously. More about this method can be found in [2].

Even though the spherical version handles the wheel coordination much better than the geometrical, the geometrical implementation is still the more robust one. The robustness in combination with the simple structure, made the geometrical version to the one best suited to be used as the base for the new under-carriage control developed in Chapter 5.

3.2 Problems

For the geometrical control structures, the use of a purely kinematic view on the steering angle coordination, results in that wheel coordination is only guaranteed is steady state. Moving from one command to another is in the kinematic model represented by an instantaneous jump to the new target, which in reality will require an unknown number of control cycles. Since the demanded angle change might differ for the wheels, they will most likely require different number of samples to reach their targets. Fast and large set point variations might therefore cause the wheel angles to drift according to each other, resulting in an unpredictable behavior of the robot.

With the spherical control structure, the coordination is much better, coordination is here only guaranteed when the steering velocities behave as their targets. If the targets change slowly enough they can be reached fast and the wheels are kept coordinated, but for a more aggressive ICM control, the wheels will drift slightly even here, since different wheels might not reach their steering velocity targets simultaneously.

Chapter 4

System Model

An exact model of a real system is always impossible, and mostly even undesirable, to achieve. In a complex system like the Car-O-bot©3 platform, it is of interest to keep the model as simple as possible, but of course without losing valuable information. In most papers, for example [3], [8], [4], only a kinematic model is considered due to simplicity. Completely disregarding the dynamics works fine in many situations but will, at least in the Care-O-bot© 3 case, described in Section 3.2, meet some difficulties when the maneuvers become more complex. One of the main parts of this thesis is to introduce dynamics into the previous model, which will be done shortly, but since the kinematics still is the basis of the model, a detailed derivation is presented as a start.

4.1 Approximations

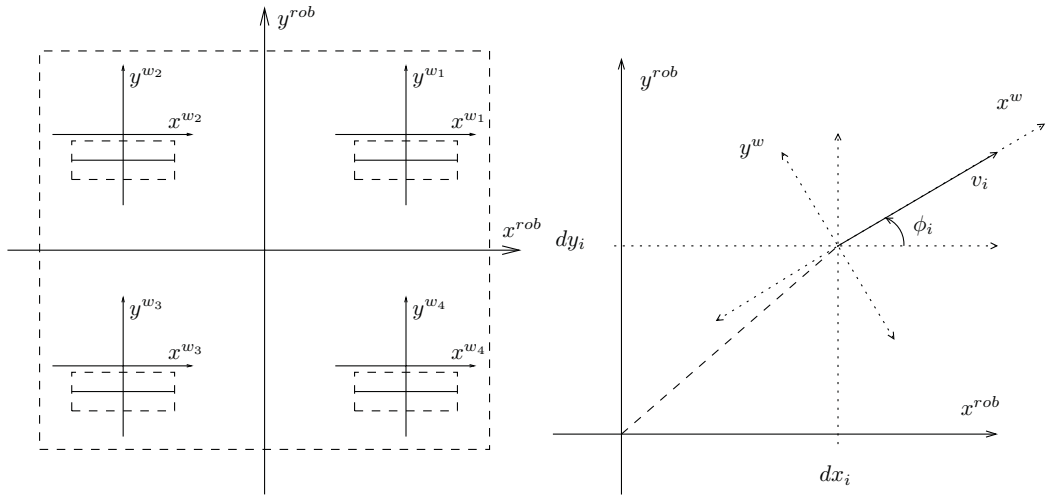
Just as in [7] some formal simplifications will be made. It is from here on assumed that the robot wheels are non deformable, which due to their solid rubber structure seems reasonable even without a detailed study; the wheel ground contact area is approximated by a single point; the robot is considered to move on a flat horizontal surface; the wheels have perfect contact to the ground; the load is supposed to be equally distributed over all wheels.

4.2 Coordinate Systems

In most papers, robot motions are described in relation to a global frame, since the goal normally is to achieve a desired position in the surrounding room. Motions of subsystems as well as the robot velocity are then mapped to the world coordinate frame, see [7],[3],[8]. In the Care-O-bot©3

under-carriage control, the robot motions are only represented as the linear velocities and rotation rate. No information about the robot's position with respect to any global frame is available; mapping the known velocities to such a frame is thereby undesirable. Instead the robot coordinate plane is chosen as the outermost frame.

The robot coordinate plane, (x^{rob}, y^{rob}), Figure 4.1(a), is defined as the coordinate system fixed to the robot platform on ground level, with the positive x-axis pointing in the platform forward direction with zero steering angle. The coordinate system's origo is placed in the geometrical midpoint of the platform. Likewise, the four wheel coordinate planes, $(x^{w_i}, y^{w_i}), i = 1..4$, are defined parallel to the robot coordinate plane, with the origo in the wheel steering rotational center, (dx_i, dy_i). The wheel offset is fixed along the negative y-axis, y^{w_i}, for all wheels. Zero steering angles on all of the wheels then result in the platform configuration shown in Figure 4.1(a). The wheel steering angle, ϕ_i, is defined as the angle between the robot coordinate plane x-axis, x^{rob}, and the wheel coordinate plane x-axis, x^{w_i}, as in Figure 4.1(b).



(a) Definition of robot and wheel coordinate planes.

(b) Definition of wheel steer angle.

Figure 4.1: Definition of robot and wheel coordinate planes and platform model labels.

4.3 Kinematics

The kinematics of a system covers the relation between the motions of different parts of a rigid body, in this case the relations between the robot and the wheel motions. The kinematics enables to study possible steady state wheel setups, not taking into account forces causing motion or forces in the system as inertia and friction. The transitions between steady states are not modeled and therefore considered to be instantaneous. A purely kinematic model will then be a good system representation when the system itself can change fast enough for the state transformation approximation to be valid.

4.3.1 Constraints

When a wheel, with the in Section 4.1 mentioned approximations, rolls on a surface, it is supposed to fulfill the pure rolling without slipping condition [7]. The linear velocity then corresponds to the wheel rotational velocity and equals the actual forward velocity of the wheel. A displacement of the wheel can then only occur in the direction of the wheel. The linear velocity of the wheel in the wheel coordinate frame is then given by:

$$v_w = \begin{bmatrix} \omega * radius \\ 0 \end{bmatrix}. \tag{4.1}$$

When four wheels are mounted on a rigid body, the pure rolling condition for each of the wheels will result in kinematic constraints relating the wheels' directions and rotational velocities to each other. A widely used approach, used in [7][4][3] etc., to realize and to visualize the relation between the coordinated wheels, is to consider each motion of the wheel base as a rotation around an instantaneous center of motion (ICM), see Figure 4.2). The motion around the ICM is seen as a circular motion known from classical physics, which results in Equation 4.2, where the forward velocity, v_j, of a point moving around the center, is orthogonal to the radius, r_j, and equal to the product of the rotational velocity, $\dot{\theta}$, and the distance from the rotation center, r_j. When driving straight the turning radius will approach infinity and all the wheels will have the same linear velocity.

$$v_j = \dot{\theta} * r_j \tag{4.2}$$

The ICM is given from the linear velocity and rotational rate of the robot, since the correlation between those entities decides the turning radius, which equals the distance to the ICM orthogonal to the forward velocity. When the robot wheels are coordinated, the ICM will be the point in which all of

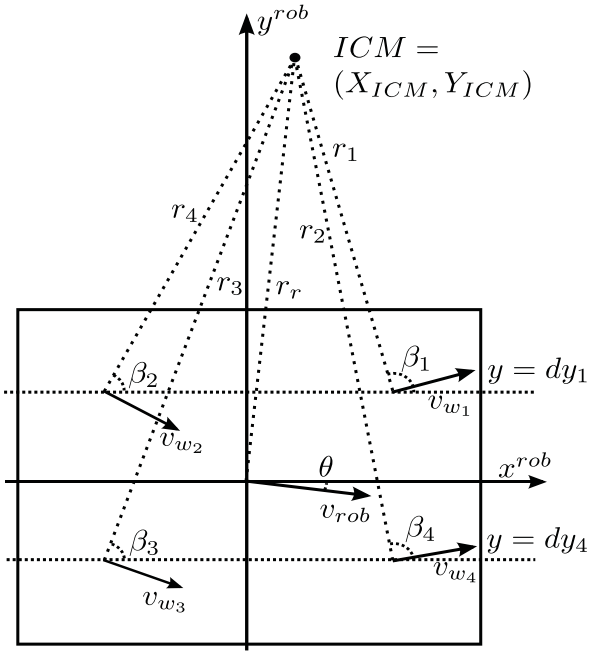


Figure 4.2: The wheel velocities/ICM relations.

the virtual extended wheel axes intersect. Thus the steering angles can be directly connected to the ICM, resulting in four constraints for the system.

Steering Constraints

As can be seen in Figure 4.3, the ICM position vector r_r equals the sum of the wheel position vector d_i, which is the location of the wheel rotational center in the robot coordinate plane, and the ICM wheel vector r_i. The Figure depicts one wheel only, but the relation holds for all wheels. The wheel turning radius and required steering angle can then be found from Equation 4.3 resulting in Equation 4.4, where β is the angle between r_i and x^{rob} as defined in Figure 4.2 and Figure 4.3, $d_i = \begin{bmatrix} dx_i \\ dy_i \end{bmatrix}$, and $r_r = \begin{bmatrix} x^{rob}_{ICM} \\ y^{rob}_{ICM} \end{bmatrix} = \begin{bmatrix} X_{ICM} \\ Y_{ICM} \end{bmatrix}$.

$$\begin{bmatrix} X_{ICM} \\ Y_{ICM} \end{bmatrix} = \begin{bmatrix} dx_i \\ dy_i \end{bmatrix} + \begin{bmatrix} r_i * cos(\beta_i) \\ r_i * sin(\beta_i) \end{bmatrix} \tag{4.3}$$

$$\beta_i = atan2(\frac{sin(\beta_i)}{cos(\beta_i)}) = atan2(\frac{Y_{ICM} - dy_i}{X_{ICM} - dx_i}) \tag{4.4}$$

For each β two different steering angle possibilities fulfill the constraints,

$\phi = \beta + \pi/2$ or $\phi = \beta - \pi/2$. Which one of those two alternatives to use depends on the desired wheel configuration.

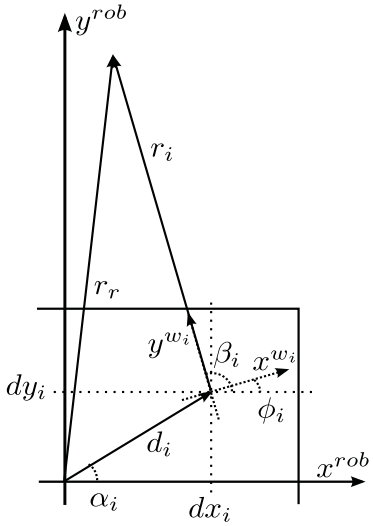


Figure 4.3: System notation definitions and vector relations.

Wheel Velocity Constraints

Since the rotational velocity around the ICM must be the same for the whole system, the different wheel ICM distances will result in different wheel velocities. For each ICM there exists two different possible robot motions; clockwise and counterclockwise rotation around the ICM. The forward rotational direction of the wheel is defined to be the positive x direction in the wheel coordinate system. Depending on which steering angle alternative chosen, the linear wheel velocity is pointing either in negative or positive x^w direction. The rotational velocities of the wheels are then dependent on both the target robot velocity and the steering angles.

The wheel offset from its rotational center will also play a part in the choice of commanded wheel velocities. Depending on which of the steering angle alternatives chosen, the wheel will either be closer to or farther away from the ICM than the wheel rotational center is. Since the distance to the ICM together with the rotational velocity around it decides the linear velocity of the wheel, the chosen steering angle will matter even for the speed of the wheel, not just the rotation direction. The resulting wheel velocity can be found from Equation 4.5, where dr is the wheel offset described in Section 3.1, which is equal for all wheels.

$$v_i = \begin{cases} \frac{\dot{\theta}}{|r_i|-dr}, \ if\ \beta_i = \phi_i - \frac{\Pi}{2} \\ \frac{\dot{\theta}}{|r_i|+dr}, \ if\ \beta_i = \phi_i + \frac{\Pi}{2} \end{cases} \tag{4.5}$$

4.3.2 Singularities

One of the problems with the classical ICM representation is the introduction of computational singularities, resulting from that the radius approaches infinity when the robot is driving straight. One way around this problem is to transfer the ICM to spherical coordinates, like in [1]. During the development of the new control method described in Chapter 5, it was highly important to easily visualize the ICM path. With a ICM represented in spherical coordinates, it is not so straight forward to quickly understand which robot motion that corresponds to a given ICM. For this work it was thereby preferred to keep the classical ICM representation.

In order to avoid the computational singularities, the classical representation was used for radius up to a certain limit, above which all motions are considered as straight. This way to represent the ICM is described more in Section 5.4. The ICM radius limiting extension to the classical representation does indeed offer a way around the computational problems, but is only good as a temporary solution. It is not continuous and the jumps it includes will cause problems in the control.

Some system singularities also exist; when the ICM is placed directly on the coordinate of a wheel, the wheel can no longer rotate around the ICM and follow the direction of the robot. When controlling the system, the real singularities should preferably be avoided if possible; more about this in Section 5.5.

4.4 Dynamics

When the transition between kinematic states has a noticeable effect on the overall system, the purely kinematic model needs to be extended with some dynamics in order to be a valid representation of the system. In the Care-O-bot©3's case, the flexible structure of the platform in combination with complex motion demands on the steering of the wheels, will, as described in previous chapter, cause the wheels to lose coordination with respect to each other. What happens is that one or more of the wheels are unable to reach the demanded steering angle fast enough, due to dynamical limitations that are not considered in the kinematic-based motion planning.

In order to include the dynamic limitations in the motion planning, a model over the reference values effect on the actual steering angles and velocities must be constructed. The steering angle change over a certain period of time will always be the integral of the actual steering velocity, thus focus for the dynamic model is to represent the relation between a commanded and measured steering velocity within a sample.

4.4.1 The Choice of Included Dynamics

Building a dynamic model can basically be done in two different ways, either directly from known dynamic equations, or constructed from experimental results. In most cases a study over which dynamic parts that have most significant impact on the system is needed in order to keep the model as simple as possible.

Setting up all the dynamic equations and convert to a single mathematic model for the Care-O-bot©3 platform, is a very complex task. Looking at the gear friction and the slip forces, the steering angles, steering velocities, and drive velocities will be linked, and the forces will be nonlinear. This approach was tried out, but the resulting model even for a single wheel module became to complicated to be valuable as an analytical model, and no usable dynamical platform model was constructed with this approach.

To only include the most limiting dynamics, the measurement based approach was chosen. As introduced in Section 3.1.2, the measured steering angles and velocities are collected by the under-carriage control from the Elmos once per sample, and the wheel velocity references are passed to the Elmos with the same time interval. Since the purpose of the dynamic model is to represent the behavior of the steering velocities during a sample, measurements for model building purpose needs to be collected with a much higher sample rate than what is done in the under-carriage control loop.

Due to the software architecture, a major system modification would be needed in order to get the required measurements. However, a Simulink model of the platform physics existed, developed in a bachelor thesis by a previous student. The simulation model, covering the whole system from Elmo reference values to actual robot motions, is the best available representation of the platform physics. Slip, friction, and steer drive coupling, as well as the total wheel forces acting on the platform is included. By temporary system modifications, no longer available, the model accuracy had been investigated and validated, see [17].

By investigating the steering velocity response to different sizes of input steps, the limiting system parameters could be detected. The top graph in Figure 4.4 shows the result of a large steering velocity reference step (yellow

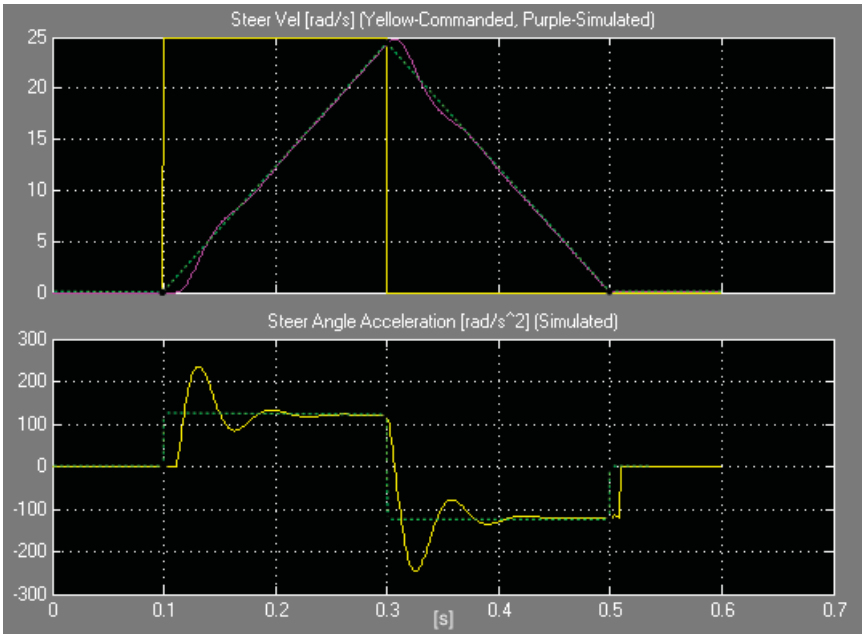


Figure 4.4: Steering angle velocity and acceleration step responses for large input step. Top graph: y-axis rad/s, x-axis seconds, yellow commanded, purple measured, dotted green approximated. Bottom graph: y-axis rad/s^2, x-axis seconds, yellow measured, green approximated.

curve). As can be seen the corresponding velocity response (purple curve) increases with an approximately constant slope until the target value is reached. It can also be seen that the response initially increases with a higher slope then later on, but overall an approximation with a ramp function (green dotted line) seems like a fairly good and simple representation.

For a smaller reference step, Figure 4.5 top graph, the target is reached faster, but the velocity behavior is the same as for the large input step, except that the initial transient here covers a more significant part of the total response. The transient effect then becomes more visual, but the ramp function approximation (green dotted line) does, just as in the Figure 4.4, seem like a good representation also here.

Decreasing the reference step even more, results in the step response in the top graph of Figure 4.6. Here the target is reached before the initial transient has disappeared, and the ramp approximation is not as good as in above cases. Comparing the ramp approximation (green line) with the simulated measurement (purple line), they both reach their target almost simultaneously, and the area under both lines are approximately equal, so

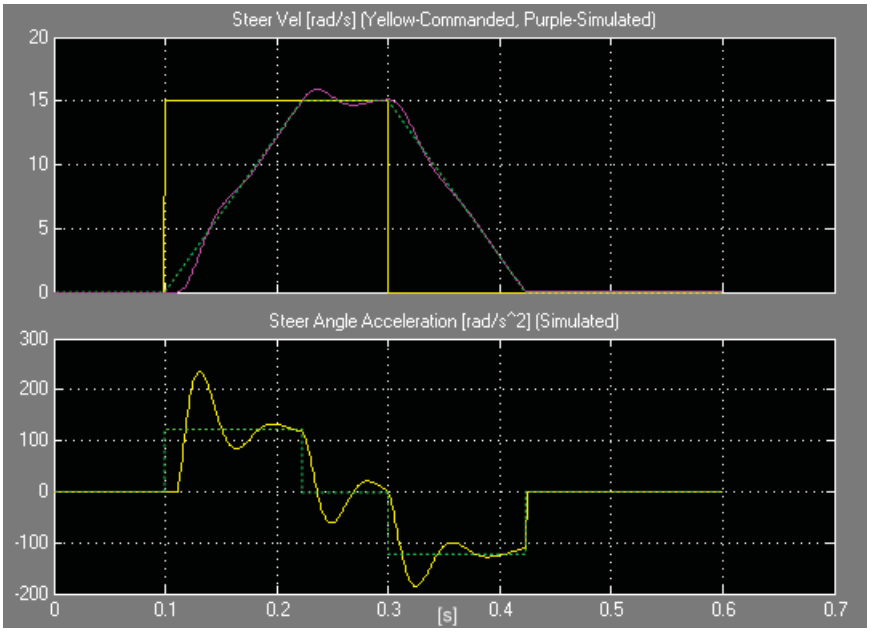


Figure 4.5: Steering angle velocity and acceleration step responses for medium input step. Top graph: y-axis rad/s, x-axis seconds, yellow commanded, purple measured, dotted green approximated. Bottom graph: y-axis rad/s^2, x-axis seconds, yellow measured, green approximated.

also in this case the ramp function seems like a valid representation.

A ramp-like velocity behavior corresponds to a constant acceleration. The velocity ramp function approximation would then give a step acceleration curve. The bottom graphs in Figures 4.4, 4.5, 4.6 show the corresponding acceleration curves for the velocity responses. The green dotted line is the approximated acceleration resulting in the top graph's velocity approximation. Just as for the velocities the approximation is best for Figure 4.4.

The source of the presented velocity and acceleration behavior is the motor current limits. In an electrical permanent magnet motor, the current through the stator windings create a magnetic flux resulting in a torque on the rotor. The current and the torque are almost linearly related. The torque produces an acceleration of the motor. The motor currents are limited due to the limited heat sink capacity of the motor. For increased performance, the rated motor current is normally allowed to be exceeded during short periods of time in order to help during for example start up, to overcome the static friction. The initial high current limit is only used for some milliseconds, and does then decrease to rated current value.

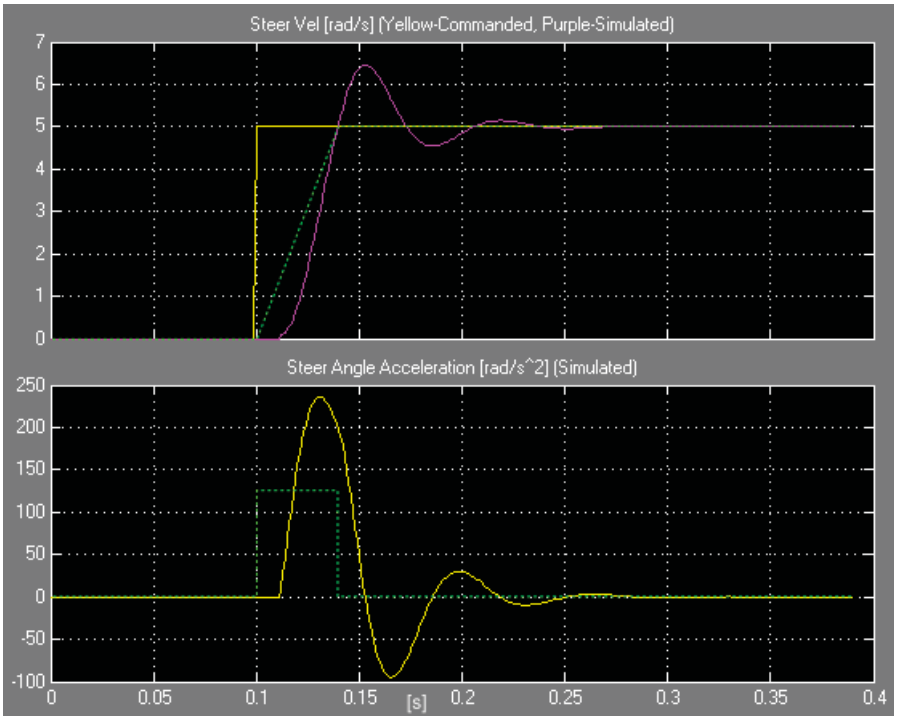


Figure 4.6: Steering angle velocity and acceleration step responses for small input step. Top graph: y-axis rad/s, x-axis seconds, yellow commanded, purple measured, dotted green approximated. Bottom graph: y-axis rad/s^2, x-axis seconds, yellow measured, green approximated.

To fully use the motor capacity and reach velocity targets as fast as possible, the motor current limit value is mostly put out to the motor when acceleration is required. The motor acceleration limit curve will then be shaped like the current limit, which explains the initial overshoot on the acceleration graphs and the initially increased slope in the velocity graphs, as can be seen in Figures 4.4, 4.5, 4.6.

From the above experiments, the constraint acceleration seems to be the main cause of the steering angle control limits. Naturally there is also a velocity limit, but for normal steering commands the reference angle does not shift more than half a revolution at once. For such small steer reference changes, the maximal velocity will not be reached before the angle reaches its target. For safety reasons the maximal allowed velocity cannot be completely disregarded, but for an as simple dynamical model it is not of interest.

The focus so far has been the steering angles, since those are the ones causing problems, but something about the drive velocity dynamics is worth

mentioning. If the smoothness of the robot motion would not be considered, the torque limitations of the drive motors would be limiting the forward motion, in the same way as the steering motor torque. Due to the much higher inertia in the drive than in the steer module, the torque limit would result in larger acceleration restrictions than it does for the steer motors. In reality, applying maximal driving acceleration to all the wheels would cause the robot to accelerate much quicker than would seem safe for the surrounding, modeling the drive in the same way as the steering would thereby not be useful. For the drive, the maximal velocity will instead have much more impact and will therefore have to be considered in the motion planning.

4.4.2 Constant Acceleration Model

In the previous section the limiting steering dynamics was specified from simulated measurements. The steering acceleration behavior was approximated by constant acceleration limits according to Equation 4.6. In order to use this model for control purposes, it needs to be delimited to a 20 ms interval, since the purpose of the model is to describe how a velocity command affects the actual steering velocity, and the velocity commands are passed on with a 20 ms interval.

$$\ddot{\phi} = \begin{cases} \ddot{\phi}_{max}, & \dot{\phi}^* > \dot{\phi} \\ 0, & \dot{\phi}^* = \dot{\phi} \\ \ddot{\phi}_{min}, & \dot{\phi}^* < \dot{\phi} \end{cases} \tag{4.6}$$

Since the reference only can be changed once per sample, there are three possible alternatives for a command; accelerate, decelerate, or keep the current velocity. Even though acceleration and deceleration are approximated as constant when a velocity change is needed, they are only applied until the target is reached or the next command is received, the total acceleration or deceleration amount within the sample can thereby be varied by the choice of velocity command. Knowing the acceleration the maximal useful velocity command can also be calculated.

The velocity command of the k'th sample $\dot{\phi}^*(k)$ will then result in that a constant acceleration is applied for a time $t_a < Ts$, resulting in a velocity ramp function within the sample like in Figure 4.7 for acceleration, and Figure 4.8 for deceleration. t_a is the time it takes to reach the commanded value, given in Equation 4.8, and Ts is the sample time. Summarizing in mathematical terms gives Equation 4.7, where a is the amplitude on the acceleration step. Since the deceleration and acceleration only differs by sign according to the simulations, Equation 4.7 covers both cases.

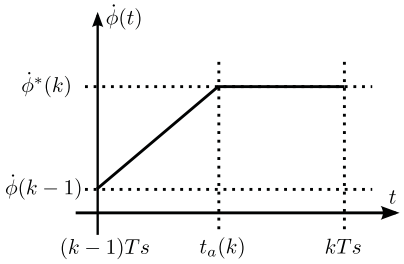


Figure 4.7: Model for acceleration command.

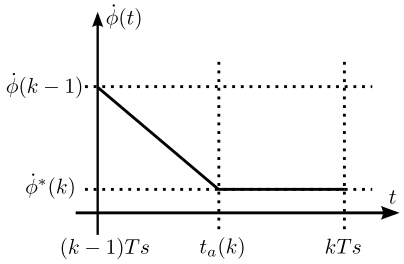


Figure 4.8: Model for deceleration command.

$$\dot{\phi}(t) - \dot{\phi}(k-1) = \begin{cases} at, & t < t_a \\ at_a, & t_a \leq t \leq Ts \end{cases} \tag{4.7}$$

$$t_a = \frac{\dot{\phi}^*(k) - \dot{\phi}(k-1)}{a} \leq Ts \tag{4.8}$$

Since the steering angle change equals the integral of the steering velocity, the angle change over one sample can be calculated by integrating the velocity according to Equation 4.9. Assuming that the inequality in Equation 4.8 is fulfilled, implies that the velocity command is reachable, thus $\dot{\phi}(k) = \dot{\phi}^*(k)$, Equation 4.8 and Equation 4.9 can be combined to Equation 4.10 for the steering angle after k samples.

$$\begin{aligned}\Delta\phi &= \phi(k) - \phi(k-1) = \int_0^{Ts} \dot{\phi}(t)\,dt \\ &= \begin{cases} \dot{\phi}(k-1)Ts + \int_0^{t_a} at\,dt + \int_{t_a}^{Ts} at_a\,dt, & \dot{\phi}(k)^* > \dot{\phi}(k-1) \\ \dot{\phi}(k-1)Ts, & \dot{\phi}(k)^* = \dot{\phi}(k-1) \\ \dot{\phi}(k-1)Ts + \int_0^{t_a} -at\,dt + \int_{t_a}^{Ts} -at_a\,dt, & \dot{\phi}(k)^* < \dot{\phi}(k-1) \end{cases}\end{aligned} \tag{4.9}$$

$$\phi(k) = \begin{cases} \phi(k-1) + \dot{\phi}(k)Ts - \frac{(\dot{\phi}(k)-\dot{\phi}(k-1))^2}{2a}, & \dot{\phi}(k)^* > \dot{\phi}(k-1) \\ \phi(k-1) + \dot{\phi}(k)Ts, & \dot{\phi}(k)^* = \dot{\phi}(k-1) \\ \phi(k-1) + \dot{\phi}(k)Ts + \frac{(\dot{\phi}(k)-\dot{\phi}(k-1))^2}{2a}, & \dot{\phi}(k)^* < \dot{\phi}(k-1) \end{cases} \tag{4.10}$$

Using this steering angel model, a transition from one velocity to another has to be built up by one or more of those ramp functions in such a way that each velocity command does not exceed the maximal within one sample reachable velocity change, meaning that the velocity command must obey Equation 4.11 at all times.

$$|\dot{\phi}^*(k) - \dot{\phi}(k-1)| \leq aTs \tag{4.11}$$

4.4.3 Model Validation

The Simulink model used for dynamic behavior investigation is, as previously mentioned, the only available representation of the system dynamics during the time between under-carriage control measurements are collected. The constructed constant acceleration model will therefore be validated within the simulation environment. Experimental validation from the real system will only be done in combination with the control implementation described in the next chapter. Such experimental results are presented in Chapter 6.

When simulations to find the limiting dynamics were made, velocity reference steps were applied for arbitrary times and the responses were investigated. For validation of the constant acceleration model, velocity referenses were applied in 20 ms intervals, which corresponds to the commands generated once per sample by the under-carriage controller. Transitions between targets not reachable within one interval were commanded as a series of reachable 20 ms commands.

As a first step the maximal, according to the model, reachable velocity command was applied for one sample, resulting in the response shown in

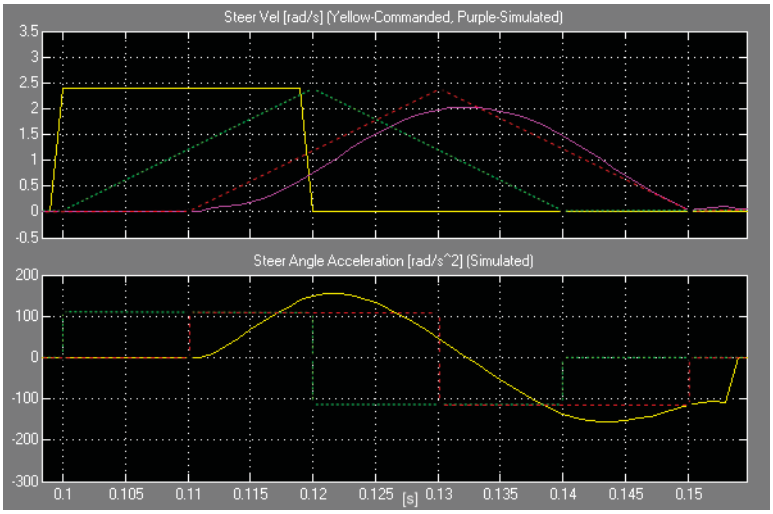


Figure 4.9: Steering angle velocity and acceleration responses to one sample maximal velocity command. Top graph: y-axis rad/s, x-axis seconds, yellow commanded, purple measured, dotted green approximated, dotted red approximated without time delay. Bottom graph: y-axis rad/s^2, x-axis seconds, yellow measured, green approximated, dotted red approximated without time delay.

Figure 4.9, and then for two samples, results in Figure 4.10. In the first case the model does not match the actual value particularly well. As can be seen in the figure, the mismatch seems to be caused mainly by a time delay in the response, which is not modeled. Since the time delay is almost half a sample long, it has a large impact on this short input step response. In Figure 4.10, the time delay impact is already much smaller and the model corresponds much better to the simulated reality.

Results for more samples corresponds well to the figures in Section 4.4.1, from which the model was constructed. The model can thereby be considered a good approximation of the system behavior, for all reference series longer than one sample. For the one sample case, it would be more accurate to add the initial time delay to the model, but the errors will be so small due to the short time, that the model error will probably not have more impact on the platform performance than unknown disturbances will. The constant acceleration model is thereby considered useful to represent the limit steering dynamics even for small reference changes.

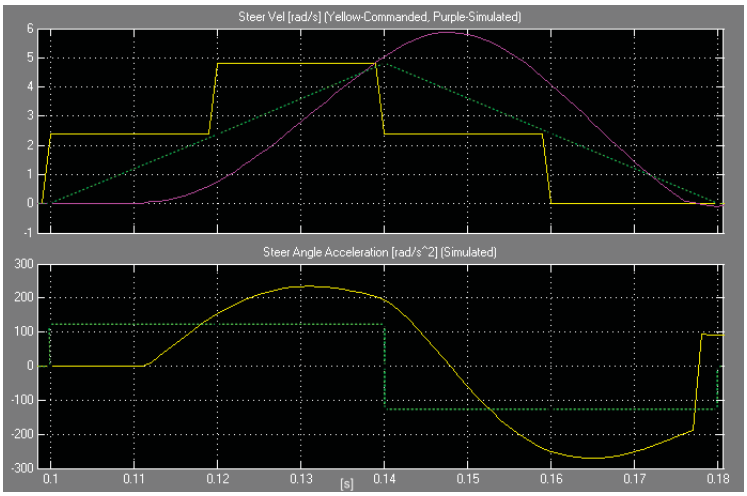


Figure 4.10: Steering angle velocity and acceleration responses to two samples maximal velocity command. Top graph: y-axis rad/s, x-axis seconds, yellow commanded, purple measured, dotted green approximated. Bottom graph: y-axis rad/s^2, x-axis seconds, yellow measured, green approximated.

Chapter 5

Constructed Control Method

As introduced in Section 4.3, wheels mounted on a common base are required to fulfill a number of kinematic constraints in order for the base to move properly. When the wheels are independently steered and driven, as in the case of the Care-O-bot©3 platform, external control for the wheel coordination is needed. The kinematic constraints describe the correlation between the linear wheel velocities, implying that both the direction and size of the velocity vectors need to be covered by the coordination control. It is reasonable to split the control into two parts, one for the velocity direction, meaning the steering angles, and one for the size of the velocity vector, which is related to the rotational velocity of the wheels. Since the wheel speeds depend on the steering angles, it is preferable to handle the steering control first and use the results in the drive control. This chapter is focused on construction of a solution to the control issues described in Section 3.2, which are related to the steering angle control. The drive control will consequently only be covered briefly.

As previously presented, the described control issues originate from the use of a purely kinematic model when choosing steering commands. The publications related to steering coordination is in most cases limited to singularity avoidance and search for better system representations of the platform kinematics. The only found publication where some wheel dynamics is included, is mostly focused on wheel-ground contact and force control [14].

As briefly mentioned in Section 2.5, Kroger et al. presents a method in [12] for synchronization of multiple degrees of freedom where dynamic constraints are considered. Their target system is a robot manipulator but the dynamic constraint model is similar to the one constructed in Section 4.4. This chapter presents a method, to handle the coordination problem by including the steering dynamic constraint model in the steering control reference calculation. The presented procedure resembles the one in [12] but is

far from identical.

5.1 Specifications of New Control Method

To solve the wheel drifting problem, the goal is to make sure that all reference values that are sent to the wheel module controllers correspond to coordinated steering angles in each sample. In this way the steering constraint errors will be minimized. Employing the ICM representation, defined in Section 4.3, the control problem is reduced to position control of the ICM within the robot coordinate plane. The task is then to:

> *Find a time optimal ICM trajectory from current to target state that guarantees that each commanded ICM is reachable within one control cycle.*

Achievable ICM points from the current ICM, depend highly on the current steering angels and steering velocities, which implies that no static mapping between those two parameters exists. An ICM point is reachable if the corresponding angle change that is required for each wheel is achievable. Depending on where in the plane the ICM is located, the needed angle changes will vary. Which wheel that will be subject to the largest steering angle change will thereby depend on the ICM location within the plane. Whether or not an angle change is achievable depends not only on the size of the demanded step, but also, due to the limited acceleration, on the current steering velocity of the wheel. Combining those aspects will lead to that different wheels will be limiting the ICM point motion at different positions and times. In Section 4.4 a dynamic model mapping the reference velocities to actual steering angle effects was developed. The idea is to include an inverse of this map in the reference value calculation, and in that way find followable wheel and ICM commands.

As mentioned in section 3.1.1 the robot references are generated from joystick output. In this work, it is assumed that

> *Only the current target shall be considered.*

When a new target is specified, the past ones do not have to be reached. No steering target velocities are given explicitly, so

> *The target velocity and acceleration are supposed to be zero.*

Deceleration to target position must therefore be considered.

5.2 Chosen Approach

In some papers, for example [4], the current ICM is defined as the intersection point between two specified wheel axes. The rest of the wheels are then coordinated according to the leading wheel pair. Problems with such a method will arise when the ICM is located right between the wheels, since no intersection point then can be found [4]. Another considerable drawback is the possible effect of disturbances on one of the leading wheels causing disturbances in the robot path. If the wheels instead could be coordinated by a global parameter, disturbances on one wheel would not affect the others as much. By separating the path planning and time aspect in the ICM control, the ICM along a chosen path corresponding to each steering angle, can be found as the wheel axes intersection with the path. When the wheels are coordinated, the intersection points will be the same for all wheels. The time aspect can then be added by defining ICM points along the path for each time instant, which is the same as a discrete time ICM trajectory.

Even though it would be possible to generate the complete ICM trajectory from the start to the end of the path, the fact that the target point might change in each cycle, and the old targets then no longer ought to be considered, would lead to that a large amount of computational power would be spent on calculating references that would never be used. Generating the trajectory step by step online will therefore be much more efficient and also simpler, since the current states then will be known. Hence, online generation of trajectories is the chosen approach.

5.3 Concept Description

In order to combine the mentioned trajectory generation idea with the existing under-carriage control architecture, some extensions are needed. The block diagram for the new the control structure including the trajectory generation is shown in Figure 5.1, which can be compared to the old version in Figure 3.6.

The system input vector $rob_{cmd} = [V_x, V_y, \dot{\theta}]_{cmd}$ represents the robot velocity command from superior control levels. The robot target generator adapts the commanded robot speed in order to avoid too sudden changes in the robot forward motion, resulting in robot velocity target vector rob_{target}. The rob_{est} vector holds a very simple estimate of the actual robot velocities, calculated directly from the wheel measurements. Both rob_{target} and rob_{est} follow the same notation as rob_{cmd}.

The trajectory generator block includes both the ICM path planning,

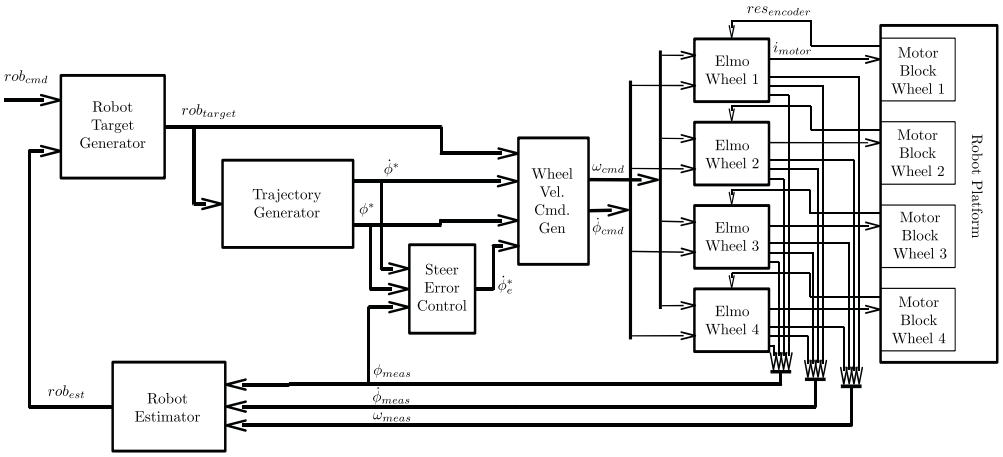


Figure 5.1: Block diagram for the new under-carriage control.

generating a path for the ICM from its current to target position, and the trajectory generator choosing a series of followable points along the path. The model developed in Section 4.4, are used to take the steering dynamics into account when choosing the trajectory points with the following three-step method:

1. Calculate the maximal achievable angle change for each wheel.
2. Derive the limiting ICM for each wheel by calculating the intersection point between the ICM path and the wheel axis that corresponds to maximal angle changes.
3. Choose the one of the four ICM candidates that is closest to the current ICM along the path.

The procedure is visualized in Figure 5.2- 5.4. Start with a known path from current to target ICM, Figure 5.2. Calculate maximal achievable ICM for each wheel, Figure 5.3. Choose the ICM candidate closest to the current one along the path, Figure 5.4.

Within the trajectory generator block the ICM trajectory points are also converted to the corresponding steering angle command vector $\phi^* = [\phi_1^*..\phi_4^*]$ and steering velocity commands $\dot{\phi}^* = [\dot{\phi}_1^*..\dot{\phi}_4^*]$, aiming to achieve the requested steering angles. The $\dot{\phi}^*$ velocities are used as feed-forward to the steering velocity controllers in the Elmo drivers. The steering error controller compensates for model errors and disturbances by adding a steering error compensation term $\dot{\phi}_e^*$ to the Elmo steering velocity command.

In the wheel velocity command generator block, the final Elmo commands are generated. The steering velocity commands $\dot{\phi}_{cmd}$ are simply the sum of $\dot{\phi}^*$ and $\dot{\phi}_e^*$, and the wheel rotational velocity commands ω_{cmd} are calculated from the steering angle commands, steering velocity commands, and robot target velocities. The Elmo drives control the motor currents i_{motor} and generate position and velocity measurements from received encoder readings $res_{encoder}$.

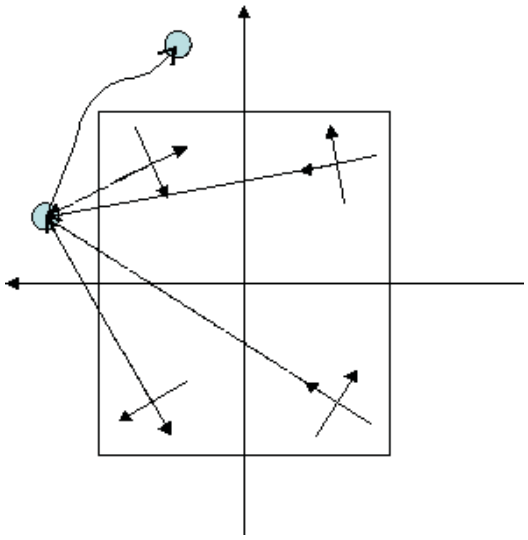


Figure 5.2: Initial state of the trajectory generator, path to target is known.

5.4 Classical ICM Representation Extension

Section 4.3.2 describes the main problem with the classical ICM representation; that straight motion places the ICM at infinity. The same section also presents the chosen solution to the problem, namely to define a circle in the robot coordinate plane where the classical representation is used for all ICM points located inside the circle, and all other points will be approximately placed on the circle. Figure 5.5 illustrates this new ICM representation. All ICM points located on the circle will then approximately correspond to straight motion orthogonal to the angle of the ICM in the coordinate plane. The radius of the limiting circle is chosen such that moving from steady state zero velocity at zero steering angle, applying maximal angle change for one cycle, will result in a steering angle change large enough to cause a replacement of the ICM, meaning move the ICM inside the circle. It shall be noticed

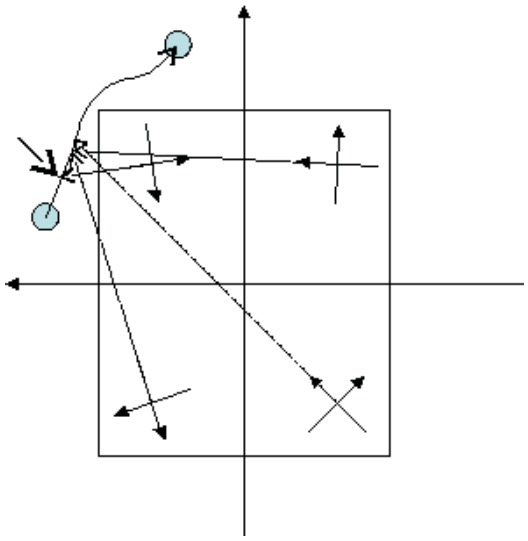


Figure 5.3: Calculation of the maximal achievable path intersection for each wheel.

that this way to represent the ICM only is a temporary solution; to achieve a well performing system a better representation must be used, preferably the spherical representation in [1].

5.5 ICM Path Planning

Choosing the curvature for the ICM is not completely straight forward, it must be continuous, to not have to worry about wheel tangents intersecting between points. Since generation time is limited, a quite simple mathematical representation, allowing for a fast calculation of the intersection points, is preferred. Choosing the path to optimize the total steering changes, avoiding problematic configurations would be most desired. Constructing such a path is a demanding task, appropriate for a future thesis. The goal here is to support the trajectory generation idea, since some path is required for the trajectory idea to work, a very simple variant is constructed; when current or target ICM is within the maximal radius, the path is given by linear interpolation between the points, Figure 5.6(a). When both current and target ICM is located on the limiting circle, the path follows the circle, Figure 5.6(b). By determining the equation for the line extending the wheel axes, each wheel axes path intersection will be found by calculation of simple polynomial intersection points.

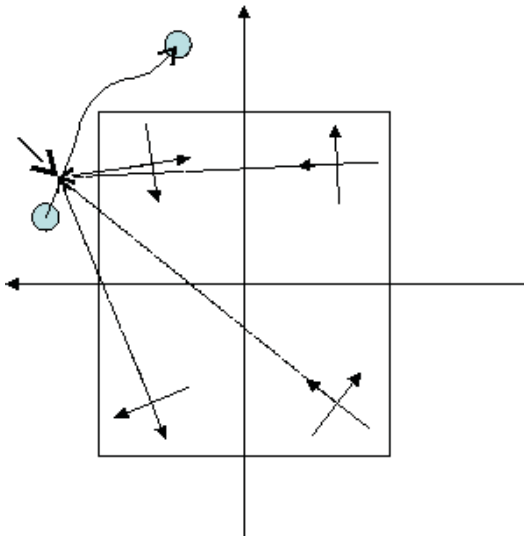


Figure 5.4: Selection of the limiting ICM candidate.

For a path between two points, there will always be two actual paths one where the turning radius is increased until the robot is moving straight and then decreased in the opposite direction, p_2, and one right where the ICM is moving straight to the target within the cycle, p_1, as can be seen in Figure 5.6(a). The first alternative together with the special ICM representation will allow for jumps, meaning that straight motion in one direction is equal to negative straight motion in the opposite direction. The points on the border circle will then correspond to the same motion for points on the opposite side of each other. Taking this into account when planning the path, jumps on the outer circle will be possible.

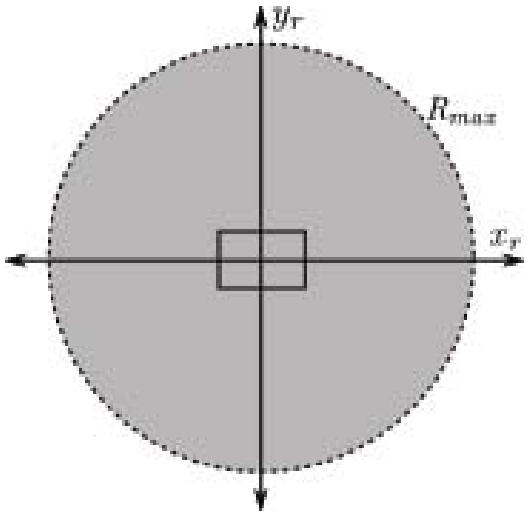
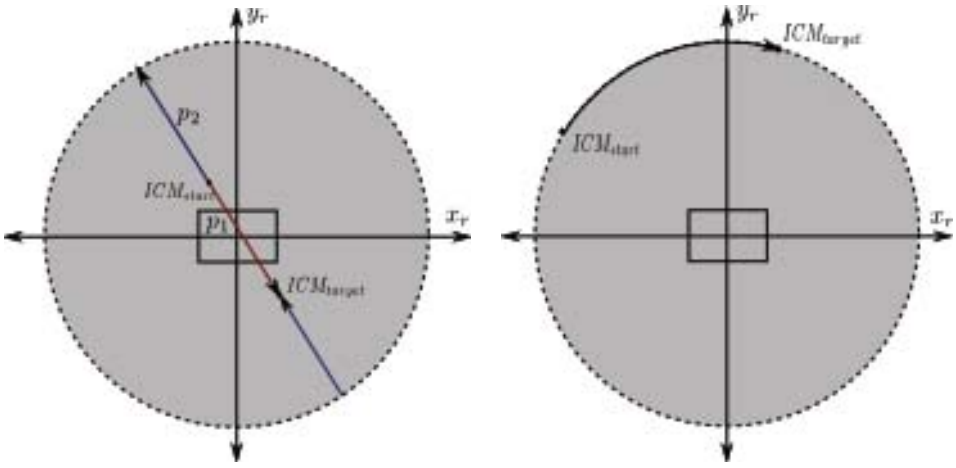


Figure 5.5: With the new ICM representation, the ICM can only be located within the gray area or en the dotted line. An ICM located on the dotted line corresponds to straight motion of the robot.



(a) Path by linear interpolation, two alternative paths exist between two points; one straight from start to target (p_1), and one trough straight robot motion (p_2) causing ICM jump.

(b) Path along radial limit, meaning that all wheels keep the same orientation when changing robot direction.

Figure 5.6: The two different kinds of ICM path types that results from the used ICM representation.

5.6 Trajectory Generation

The trajectory generator aims to add the time aspect to the path. A series of points along the path are chosen, corresponding to the ICM command in each sample. The trajectory requirements state that each of the trajectory points must be theoretically reachable, considering the wheel dynamics. In order to find the limits for the coordinated wheels, the limit for each wheel separately must be known. The separate limits can then be combined by a transition to ICM representation, and the limiting ICM points can be calculated.

5.6.1 Finding the Wheel Limits

Consider a steering angle change of one wheel according to the dynamic model in Section 4.4. Without restriction on the target steering velocity, the time optimal angle change would be achieved by applying maximal acceleration, and thereby achieve maximal average velocity, until the target angle is reached. Adding the restrictions zero target velocity and no angle overshoot, the time optimal change will be achieved by accelerating to highest possible velocity allowing for deceleration without exceeding the target angle. The velocity curve would then correspond to Figure 5.7. The steering angle change is the integral of the velocity curve, and thus corresponds to the gray area in Figure 5.7

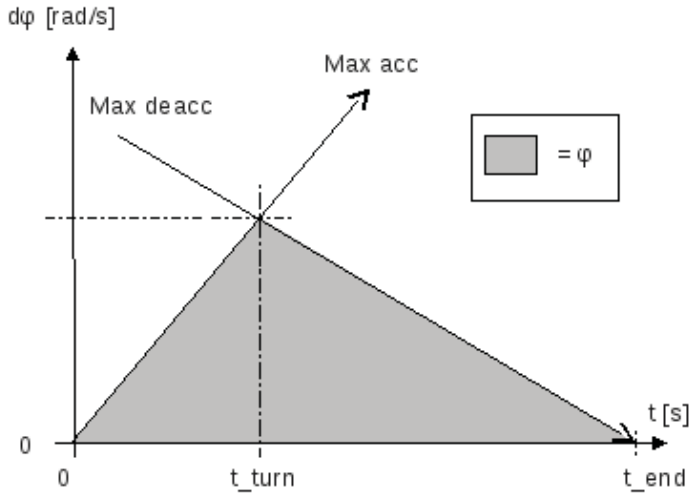


Figure 5.7: Continuous time optimal angle velocity curve.

In a discrete system, the velocity commands are given once per sample.

The change of velocity direction can thence not happen within a sample, which must be considered when choosing references. The commanded top velocity will therefore be slightly lower than for a continuous system. For a velocity to be a possible command, it must be possible to reach it within the sample, hold it until next sample, and decelerate from it to zero without exceeding the demanded angle change. The time optimal velocity curve will then correspond to Figure 5.8.

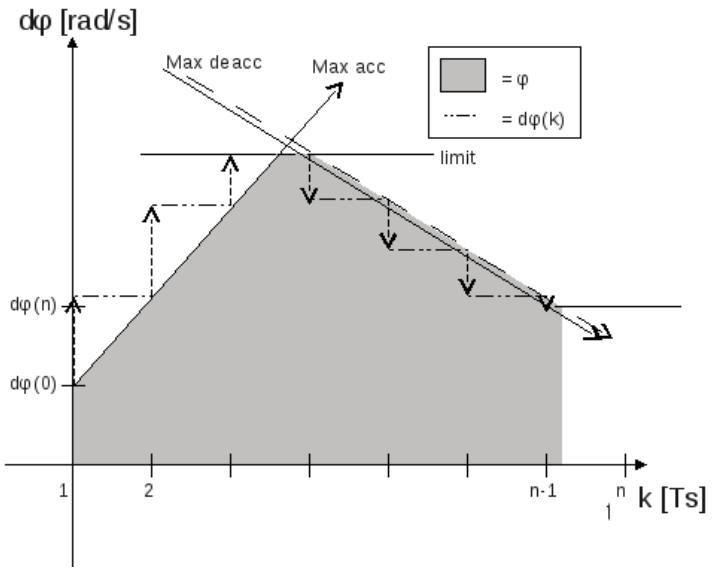


Figure 5.8: Time optimal angle velocity curve with discrete references.

According to the model in Section 4.4, the maximal acceleration and deceleration is equal, which adds simplicity. Velocity increase in one sample can then be decreased back in the next. Only deceleration from one sample ahead must hence be considered when choosing velocity command. In cases where the deceleration is lower than the acceleration, two or more samples ahead must be taken into account. The same acceleration and deceleration values also simplify calculations related to velocity zero crossing, where the same slope can be used on both sides of the zero crossing.

In reality both positive and negative angle changes occur. Since both cases follow the same procedure but with different signs, negative angle commands are then reflected to positive sign. The wheel trajectory limit algorithm then only needs to cover the positive case. The result from the algorithm is then switched back to the original sign at the end of the procedure.

When the initial steering velocity is negative, deceleration to zero will result in an angle change with wrong sign. Since the total area under the velocity curve must equal the demanded angle change, the positive area must then equal the sum of the commanded and deceleration angle change.

Constructing the Limiting Wheel Trajectories

Determining the maximal reachable steering angle and steering velocity command is a multi step procedure with the following steps:

1. Switch sign on current settings if needed; only positive changes are considered for simplicity.

2. Calculate the angle change covered by deceleration from current velocity to zero, through

$$\phi_{decel} = \int_0^{\frac{\dot{\phi}}{a}} (\dot{\phi} - at)dt = \frac{\dot{\phi}^2}{2a}. \tag{5.1}$$

3. Determine whether acceleration or deceleration is needed by the inequality Equation 5.2. If the inequality is true, the current velocity can be kept for another sample without saturation, thus acceleration might be possible. If the inequality is false, a decelerating velocity command is needed.

$$\Delta\phi - \phi_{decel} \geq \dot{\phi}Ts \tag{5.2}$$

4. Decide which steering velocity command to choose according to:

   (a) If maximal acceleration can be applied without saturation, meaning that inequality Equation 5.3 is true, $\dot{\phi}^* = \dot{\phi} + aTs$

$$\int_0^{t_{end}} \dot{\phi(t)}dt = \frac{(\dot{\phi} + aTs)Ts}{2} + \frac{(\dot{\phi} + aTs)^2}{2a} \leq \Delta\phi \tag{5.3}$$

   (b) If inequality Equation 5.3 is false, the limiting $\dot{\phi}^*$ is given by

$$\int_0^{t_{end}} \dot{\phi(t)}dt = \Delta\phi \Rightarrow \dot{\phi}^* = \frac{2a\Delta\phi + \dot{\phi}^2}{2(aTs + \dot{\phi})}. \tag{5.4}$$

(c) If $\Delta\phi = \phi_{decel}$, maximal deceleration will result in enough angle change and $\dot{\phi}^*$ is given by

$$\dot{\phi}^* = 0, \Delta\phi < \frac{aTs^2}{2}\dot{\phi} - aTs, \Delta\phi \geq \frac{aTs^2}{2}. \tag{5.5}$$

(d) The last velocity command must be zero, implementing maximal deceleration down to zero within the last sample. Limiting the deceleration is therefore done by choosing a velocity command allowing for the target angle to be reached with maximal deceleration in the next step. The velocity command is calculated in almost the same way as the limiting acceleration command in Equation 5.3, but the sign difference result in that some simplifications cannot be made in the calculations. The velocity command will then be the solution of a second order equation according to

$$\dot{\phi}^* = -\frac{aTs - \dot{\phi}}{2} + \sqrt{\frac{(aTs - \dot{\phi})^2}{2} - \frac{\dot{\phi}}{2} + a\Delta\phi}. \tag{5.6}$$

5. The goal is to relate the maximal reachable steering angles to ICM limits. The maximal angle change for each wheel, $\Delta\phi^*$, is found by integration of the estimated steering velocity within next sample according to Equation 5.7. The steering angle command limit is then the result of $\Delta\phi$ added to the current steering angle. The sign variable s is 1 when accelerating and -1 when decelerating. It shall be remembered that this only the command limits for each wheel and not the actual steering angle commands.

$$\Delta\phi^* = \int_0^{Ts} \phi(\dot{t})dt = \dot{\phi}^* Ts - s * \frac{(\dot{\phi}^* - \dot{\phi})^2}{2a} \tag{5.7}$$

6. If the sign was shifted in step 1, the resulting steering angle and steering velocity command limit must also be shifted, in order to get the right sign with respect to the demanded change.

5.6.2 Choosing the Limiting ICM

When the maximal steering angle change for each wheel is known, corresponding ICM limits can be found, as mentioned in Section 5.3, by calculating the intersection points between each of the wheel axes' tangents and the ICM path. When the ICM path is described by a simple polynomial, like presented in Section 5.5, the intersection point can be found by simple

polynomial intersection, since the wheel axes tangent can be represented by a first order equation in the robot coordinate plane. Caution must be taken when the path is described by a first order polynomial, thus the two polynomials might be parallel or close to parallel. In such case the intersection point is taken as one of the points where the path meets the radius limiting circle, and does then correspond do a straight robot motion in orthogonal direction.

The limiting ICM for the whole platform is finally chosen as the one of the calculated intersection points closes to the current ICM along the path. All wheels are then coordinated with respect to the chosen point and the corresponding velocity and angle commands are calculated.

5.7 Error control

According to the dynamic model all the wheel modules are identical. In reality minor variations in manufacturing and abrasion will cause differences in the motor and gear module dynamics. Driving in a normal household environment implies driving condition variations that in most cases will have different impact on each wheel module. A combination of multiple small variations will eventually have a noticeable effect on the wheel modules' performance. Such effects can be seen as unknown disturbances and are, due to their unknown nature, hard and undesirable to comprise for in the reference value planning. By instead adding a steering angle position control, the disturbances together with modeling errors can be suppressed at the same time as the reference value generation is kept simple.

In the previous under-carriage control version the steer angle position control was done by one impedance control for each wheel module. To acheive the main goal of this work; to present results of the presented trajectory generation methodology, construction of an optimal steering angle error control was not the main concern. For this purpose two different error control varieties were used; one simple proportional controller and one impedance controller corresponding to previous under-carriage control implementation.

5.8 Implementation

5.8.1 Description

The constructed trajectory generation method was implemented in C++ within the previous under-carriage control thread on the on-board Linux PC handling the under-carriage control. At first the outputs from the old

implementation were used, the new version was then running within the same thread, but without affecting the actual outputs to the wheels. In this way the trajectory generator was implemented and tested gradually by verifying that the logged calculation results, achieved from the new implementation with real inputs during normal operation of the robot, was reasonable. When the whole concept implementation was authenticated, the previous wheel outputs were replaced by the result from the trajectory generator.

The implemented algorithm is an implementation of the three step method presented in Section 5.3. The main section of the trajectory generation code can be seen in Figure 5.9, where the achievable ICM and steering angle limit for each wheel is found by the procedure visualized with the chart in Figure 5.10, which corresponds to the functions CalculateMaxPhi and calcIntPoint in the code.

5.8.2 Evaluation

When testing the implemented trajectory generator idea for operation of the robot some drawbacks were noticed. An overview of the control behavior related to the implementation is thereby presented.

For ICM Trajectories placed completely outside the wheel base and inside the limiting circle, or trajectories following the circle, the wheel control appeared to work properly.

Sometimes when changing from straight to turning motion, the robot accelerates momentarily, the reason for this behavior is probably numerical errors associated with the robot forward velocity calculation. The drive control set points depend on the robot velocity around the ICM, which, when the ICM is known, can be calculated either from the commanded forward velocity or from the rotational ratio of the robot. The switch between those two calculation methods in combination with the very simple drive velocity control and the discontinuous ICM representation, then cause the strange suddenly accelerations.

The very simple path planner does not aim to avoid placing ICM commands close to or on a wheel. Moving the ICM in between the wheels works fine as long as the path does not cross exactly over a wheel, if it does various behavior occurs, mostly causing the wheels to move very slowly and then suddenly much faster.

When the ICM path aims to bring the ICM from a location between the wheels to outside the wheels, problems occur. The reason is probably that different wheels try to move in different directions along the path. Since the path direction in this implementation is decided from the direction the wheels optimally moves, this unwanted behavior is most likely to occur, and

indicates that the direction of the path needs to be specified in advance.

Despite those flaws the implementation was used for more detailed analyze of the trajectory generator idea, but with some limitations on possible test cases. The results are presented in next chapter.

```
//Trajectory generator main loop
void NewSteerCtrl::GenerateTrajectory(void)
{
    //update state
    update_Phi_current(Phi_cmd);
    update_dPhi_current(dPhi_cmd);

    // calculate desired icm path
    IcmPathPlanner();
    if (icmPath.pathState == END_STATE){
        //close to setpoint, set final val
        for(int i=0; i<4; i++){
            Phi_cmd[i] = Phi_target[i];
            dPhi_cmd[i]= 0;
        }
        icmPath.icm_now[0]=icmPath.icm_end[0];
        icmPath.icm_now[1]=icmPath.icm_end[1];
    }
    else{
        double distLimit = calcDist(icmPath.icm_end);
        double x_new = icmPath.icm_end[0];
        double y_new = icmPath.icm_end[1];

        // generate limit for each wheel
        for(int i=0; i<4; i++){
            // get steering angle limit
            Phi_cmd[i] = CalculateMaxPhi(i);

            // transfer steering angle to ICM rep
            double icm_new[2];
            Phi_cmd[i] = calcIntPoint(Phi_cmd[i], i, icm_new);

            // calculate the distance between ICM candidate and current ICM
            double dist = calcDist(icm_new);
            // compare new distance with worst case
            if (dist<distLimit){
                //update if new ICM candidate is the most limiting one
                distLimit=dist;
                x_new =icm_new[0];
                y_new =icm_new[1];
            }
        }
        //save next icm target
        icmPath.icm_now[0]= x_new;
        icmPath.icm_now[1]= y_new;

        // get Phi_cmd
        ICMToWheelCmd(icmPath.icm_now[0], icmPath.icm_now[1]);
        // get corresponding dPhi_cmd
        for(int i=0;i<4;i++){
            //vel command for feed forward
            dPhi_cmd[i]= calcVelCmd(i,Phi_cmd[i]);
        }
    }
}
```

Figure 5.9: Trajectory generator main loop implementation.

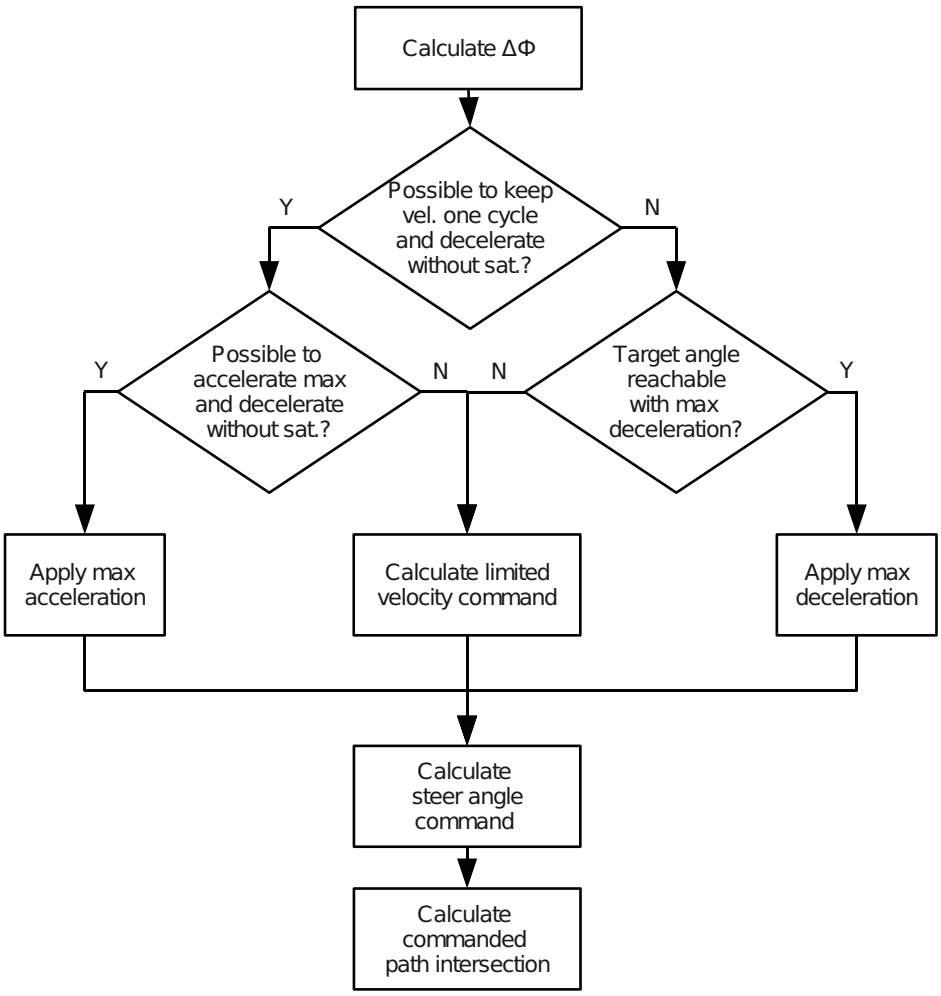


Figure 5.10: Chart over the limiting ICM algorithm.

Chapter 6

Experimental Results

The goal of this chapter is to present the trajectory generators potential to improve the wheel coordination of the Care-O-bot©3 platform. The motion patterns of interest should therefore show the problems with the old control method, and in the same way be simple enough to not cause problems with the simplified path planner of the new implementation. The drive scenario where the robot changes from straight to fully turning motion fulfills the requirements for visualization of the results. Such drive scenario corresponds to a movement from a ICM at infinity, or with the new ICM representation, an ICM located on the crossing of the y-axis and the limiting circle; to a ICM at the origin of the robot coordinate plane. Motions similar to the presented one but not necessarily including completely straight or purely turning motion will also be useful for the coordination evaluation.

All results presented in this chapter originate from measurements collected from the real robot. The robot motions are in all presented cases commanded with a joystick. Attempts to regenerate a given robot motion did consequently not result in the exact same series of robot commands. The measurements were recorded by writing parameters from the under-carriage control loop to files during runtime. The presented graphs are generated by plotting relevant parts of the collected data in Matlab.

6.1 Joint-Space Control

The purpose of this section is to present some of the drawbacks with the old control implementation, even refered to as the joint-space controller or the geomethrical controller, in order to provide a reference for comparison with the new implementation.

In the old implementation the robot motion commands were converted

directly to wheel commands as described in Chapter 5, and no ICM or axes intersection points were considered within the wheel control. In order to study the coordination and compare the ICM commands with the new implementation, the missing parameters have been calculated from known parameters subsequently.

Figure 6.1 shows an overview of the general performance of the steering angle control. The figure includes both the commanded and the measured steering angles for each wheel. The measured values follow the commanded well in most cases, which can be seen by that the two lines mostly occurs as one line in each sub figure. It can also be seen that during transients, the commanded lines lead the measured, which means that the measured are not able to follow their reference fast enough, thus two lines, for each wheel, can be seen separated in the figure.

Figure 6.1: Overview of commanded and measured steering angles for wheel 1-4 for arbitrary commands. Steering angles in radians on the y-axis and time in seconds on the x-axis.

Corresponding commanded and measured steering velocities are shown in Figure 6.2. In this figure the commanded and measured lines in each sub figure seem to follow each other well since they are seen as on e line for each wheel.

Focusing on the special problematic driving scenario presented, namely

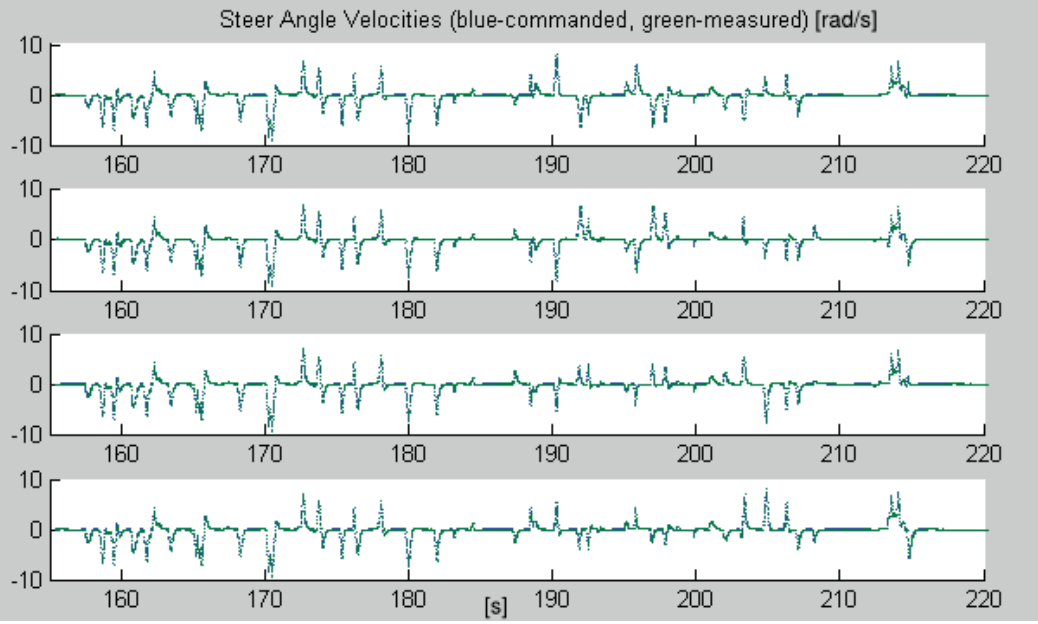


Figure 6.2: Overview of the commanded and measured steering angle velocities corresponding to the angle changes in Figure 6.1. Steering angle velocities on each of the y-axis are in rad/s and the x-axis shows the time in seconds.

changing from straight to turning motion, the ICM calculated from the robot commands are shown in Figure 6.3, where the point in (0,100) corresponds to straight motion in the positive x direction of the robot, and the point at (0,0) corresponds to that the robot is rotating around its own origin. As can be seen the commanded transition between those two endpoints are sudden, only a few intermediate points are commanded.

Studying how the under-carriage controller responds to the robot motion commands that resulted in the ICM commands in Figure 6.3, the intersection points between the virtual extended axes of the wheels, with the commanded steering angles fulfilled, can be calculated. The result is shown in Figure 6.4(a). Completely coordinated wheels should result in that all the wheel axes intersect in the same point. As can be seen in the figure, intersection points for different wheel pairs are spread out in the plane, thus the commanded steering angles does not correspond to coordinated wheels.

The intersection points of the virtual axes corresponding to the measured steering angles for the same drive scenario are shown in Figure 6.4(b). Comparing to the commanded intersection points in Figure 6.4(a) it can be seen

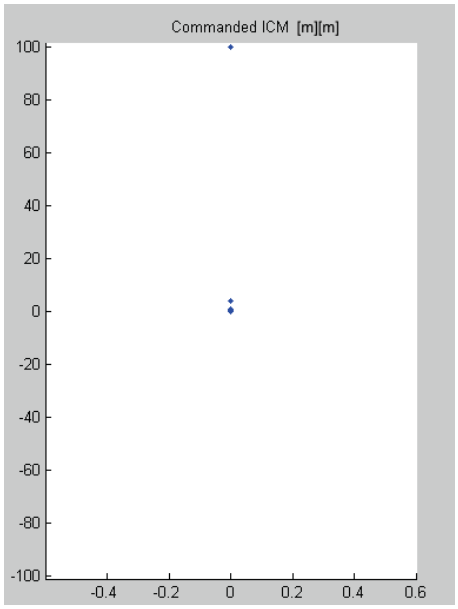
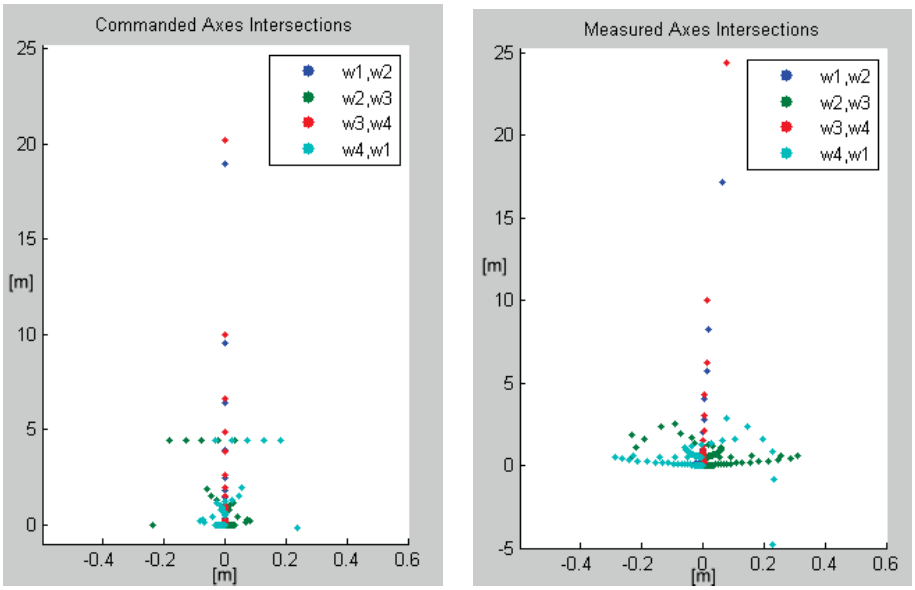


Figure 6.3: Commanded position of the ICM in the robot coordinate plane. Calculated from the commanded robot motions ordering the robot to change from driving straight, ICM in (0,100) to turning around its own center, ICM at (0,0). Both the x and y axis unit is meters.

that the measured points are more spread out then the commanded, which means that the wheels are not coordinated in reality either for the considered robot motion.

A detailed view of the steering angles and steering velocities are shown in Figure 6.5 and Figure 6.6. Once again the same drive commands are considered. It can be seen in the velocity plots, Figure 6.6, that the measured steering velocities follow their targets quite well, but the steering angles do not. It can also be seen that the steering angles do not reach their targets simultaneously, which was described in Chapter 5.



(a) Axes intersection points calculated from commanded steering angles

(b) Axes intersection points calculated from measued steering angles

Figure 6.4: Resulting virtual wheel axes intersection points from old controller for robot commands corresponding to target jump from straight forward motion to pure rotation around platform center, which is the same as the ICM commands in Figure 6.3. Coordinated wheels would result in that all the wheel axes pairs would intersect in the same point. The unit of both x and y axis is meter. The zoom of the figures are set to clearly visualize the interesting phenomena, thus some points located far away from the robot is disregarded.

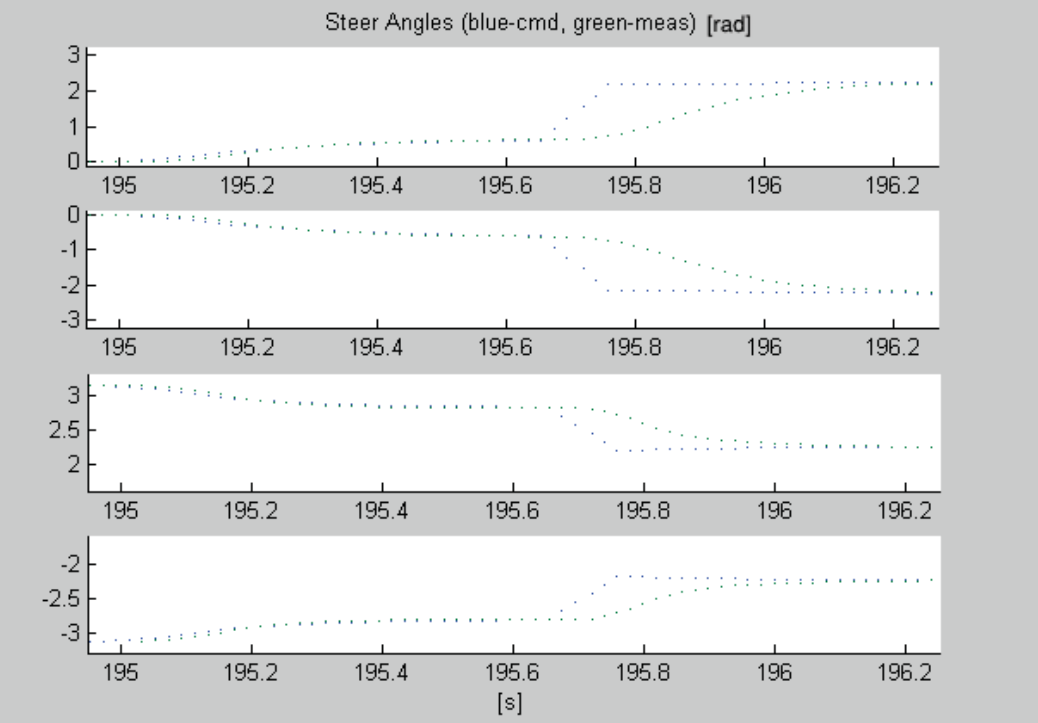


Figure 6.5: Commanded and measured steering angles from old controller corresponding to Figure 6.3. Steering angles on the y-axes are in radians, time on x-axes are in seconds. Commanded angles are leading the measured.

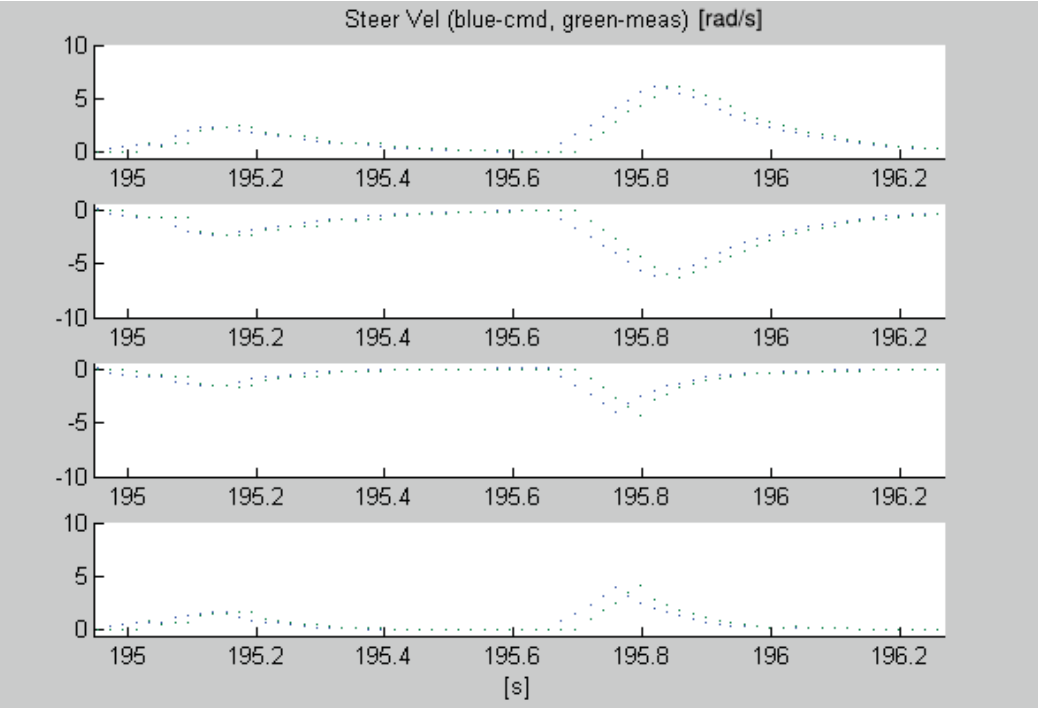


Figure 6.6: Commanded and measured steering angle velocities from old controller corresponding to Figure 6.5. Steering angle velocities on the y-axes are in radians per seconds, time on x-axes is in seconds. Commanded velocities are leading the measured.

6.2 ICM Trajectory Based Control

The results from the new control implementation was analyzed in two parts; first the work of the trajectory generation was investigated, then the effect of the actual outputs to the wheels.

The trajectory generator is, as presented in Section 5.6, supposed to generate followable points in between the globally commanded ICM points. The process is shown in Figure 6.7 where the initial ICM commands are plotted as rings, and the targets generated by the trajectory generator are represented as dots. It can be seen that the distance between the generated points are smaller close to the platform base, which coincides with the expectations since a specified distance between two points correspond to larger angle changes close to the wheel base, than further away. A zoomed version of the course of events close to the platform can be seen in Figure 6.8. It can be noticed that not all target points are reached due to the choice to disregard old commands when a new command is received.

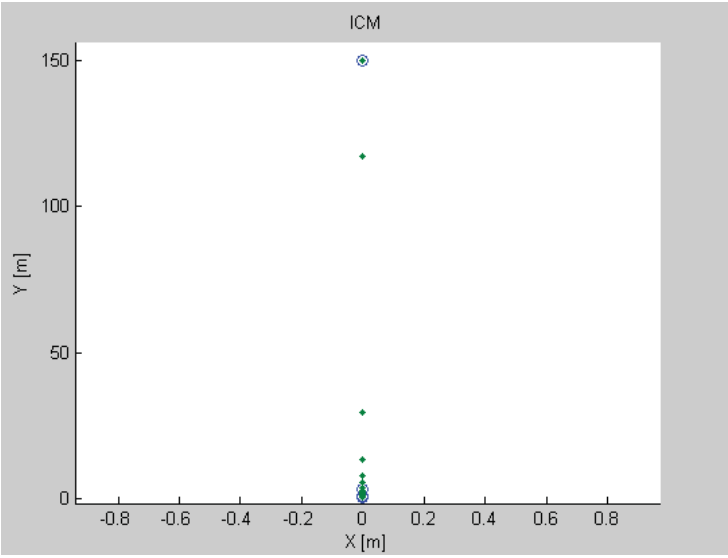


Figure 6.7: Commanded ICM trajectory from the trajectory generator, green dots, generated from commanded target points, blue rings. The unit on all axes is meters. Starting in (0,150), ending in (0,0.2).

To analyze the wheel coordination and the followability of the ICM trajectory, the wheel axes intersection points corresponding to the commanded and measured steering angles are presented in Figure 6.10(b) and Figure 6.10(a) respectively. The commanded ICM trajectory is shown in Figure

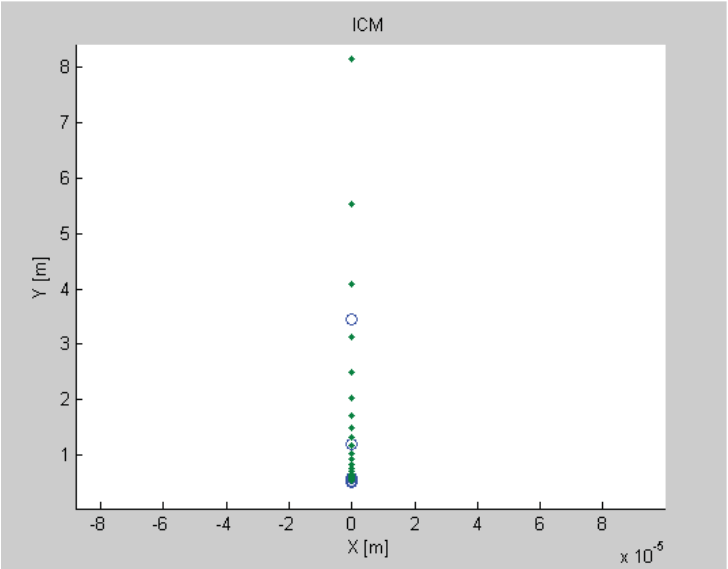


Figure 6.8: Zoom of Figure 6.7. All axis units are meters. Trajectory generator targets represented by rings, generated points represented by dots.

6.9, the ICM does in this case move along the negative y axis towards origin, which corresponds to a right turn instead of a left as in previous cases. Since each angle command are calculated directly from corresponding ICM command, all wheel axes pairs intersect in the same point, as can be seen in Figure 6.10(a), which can be compared with the commands from the old controller in Figure 6.4(a). The measured steering angles are not as well coordinated as the commanded, this can be seen by that the axes intersection points are slightly spread out for a given ICM command, Figure 6.10(b), but compared with the measurements from the old controller, Figure 6.4(b), the coordination is dramatically improved.

For the ICM trajectory to be followable, both the steering angles and steering velocities must be able to follow their references. Figure 6.11 shows the steering angles that correspond to the ICM trajectory in Figure 6.9. Three dotted lines are shown in each subfigure, one represents the trajectory generator target steering angles calculated directly from the robot commands, and the other two represents the actually commanded and measured angles respectively. It can be seen that the measured angles follow the commanded angles well for all wheels, and that they both reach the external target together for all wheels.

The steering velocities that belong to the steering angles in Figure 6.11 are shown in Figure 6.12. Here the three dotted lines in each subfigure

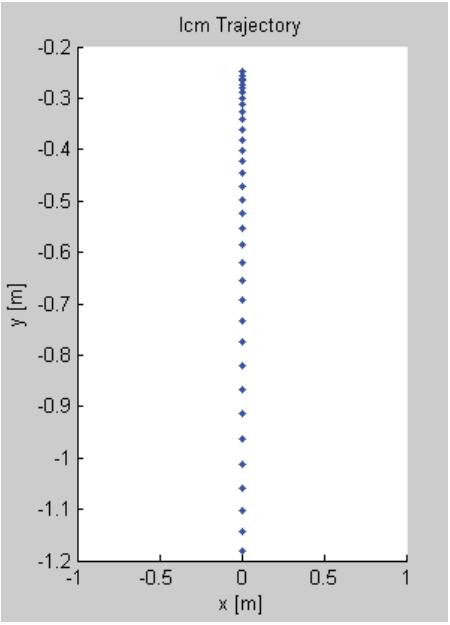


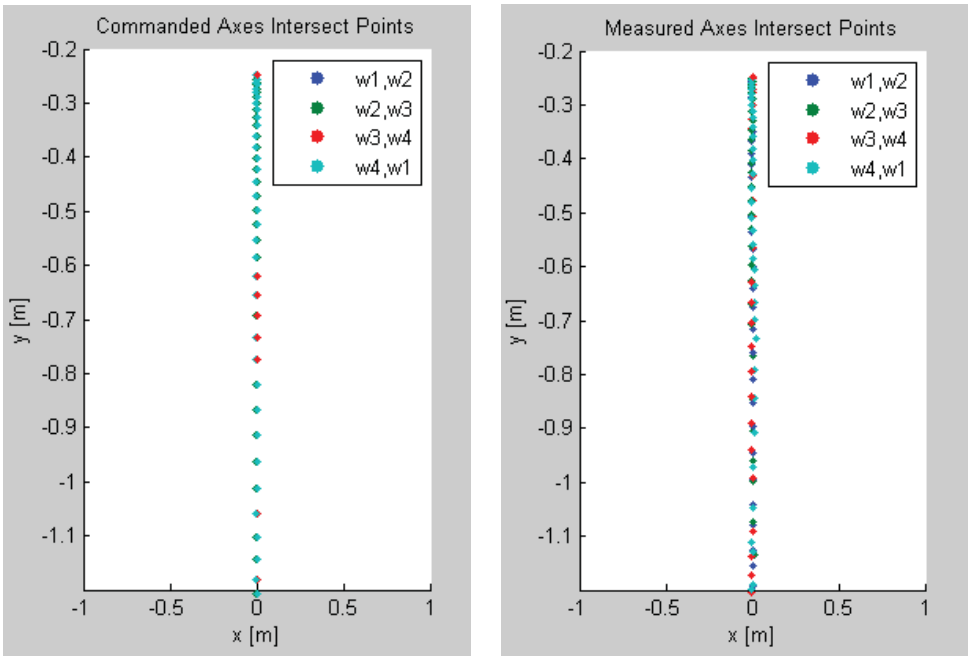
Figure 6.9: The end part of commanded ICM trajectory corresponding to straight to right turn of the robot. Axes units are meters.

correspond to the steering velocity commanded from the trajectory generator, the final commanded steering velocity including error compensation, and the measured steering velocity. The error compensation control is here done with a simple quite aggressive proportional controller. As can be seen in the figure, the steering velocities do not follow their targets as well as the steering angles, but is still quite good. The reason is most likely the included effect of the error compensation.

Figure 6.13 shows the same concept as Figure 6.11 but for a target changing more often but with smaller steps. It can be seen that even here the commands from the trajectory generator is followed very well by the measured steering angles.

6.3 Result Evaluation

Data recordings from the robot using the old under-carriage control implementation show that the in between target commanded steering angles are not coordinated. For complex maneuvers, the lack of coordination eventually leads to temporary but undesirable wheel drifting. Looking closer at the case when changing from straight motion to fully turning motion, it can be



(a) Axes intersection points calculated from commanded steering angles by the new comtroller. All wheel axes pairs intersect in the same points, which means that the commanded steering angles are coordinated.

(b) Axes intersection points calculated from measured steering angles. Compared to the intersection points from measured steering angles using the old controller, Figure 6.4(b), the wheels are much better coordinated with the new control strategy.

Figure 6.10: Resulting virtual wheel axes intersection points new old controller for robot commands corresponding to target jump from straight forward motion to sharp right turn. ICM commands for the interval plotted in Figure 6.9. Coordinated wheels would result in that all the wheel axes pairs would intersect in the same point. The unit on all axes is meters.

seen that the wheels are controlled to reach their targets independent to each other, and thus thereby not reach their targets simultaneously.

Recordings from the trajectory generator implementation show improved wheel coordination when moving from straight to fully turning robot motion. The results indicate that the trajectory generation idea does indeed improve the system's ability to coordinate the wheels. As presented in Section 5.8.2, the trajectory generation implementation is not complete, which strongly constrains the amount of possible test cases. The performance and robustness for a complete implementation is yet unknown. Even though the presented measurements indicate increased performance, some robustness issues can be realized; the trajectory generator fully depends on that

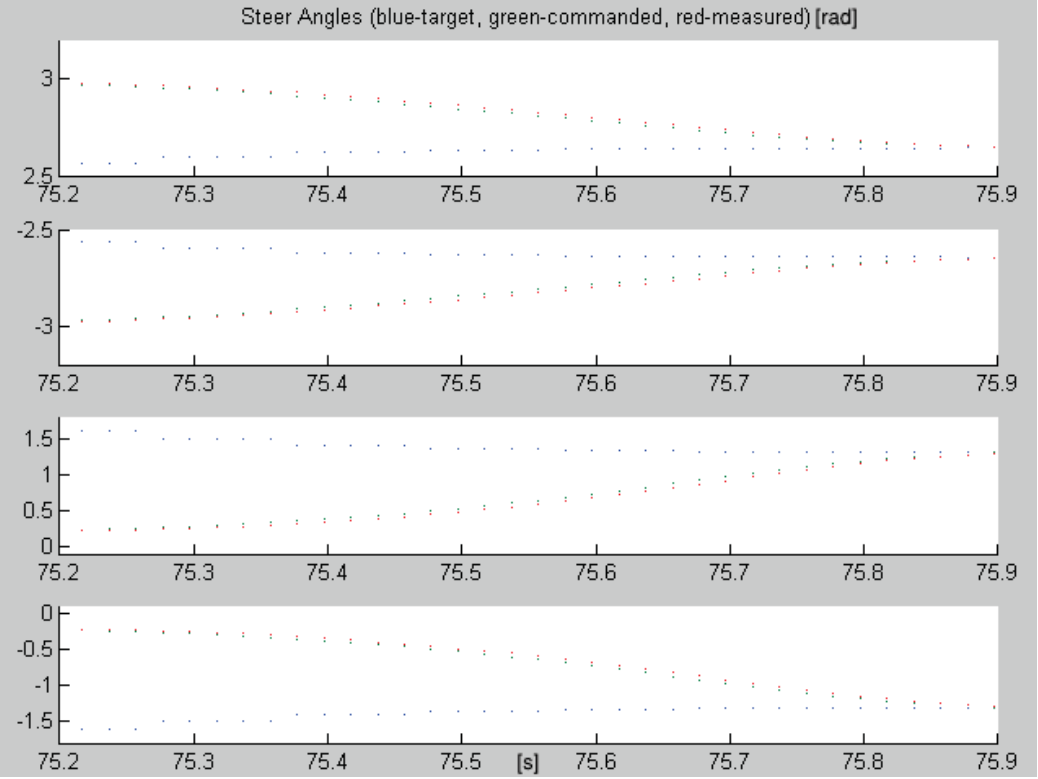


Figure 6.11: Target, commanded and measured steering angles for all the wheels (1 in the top graph and so on). Y-axis unit is radians and x-axis unit is seconds. The measured angles follow the commanded well and they both reach the target together simultaneously for all wheels.

the dynamic model is a good representation of the reality. Even though the steering error controller eliminates small model errors, no reality adaption of the wheel trajectories are available within the trajectory generator, meaning that the steer error controller must handle all model errors alone. If the steering dynamics would change, for example by driving on a heavy carpet, the generation of followable trajectories would eventually fail and problems occur. Extending the presented idea with an adaption to actual robot motion would thereby be necessary for a robust final performance.

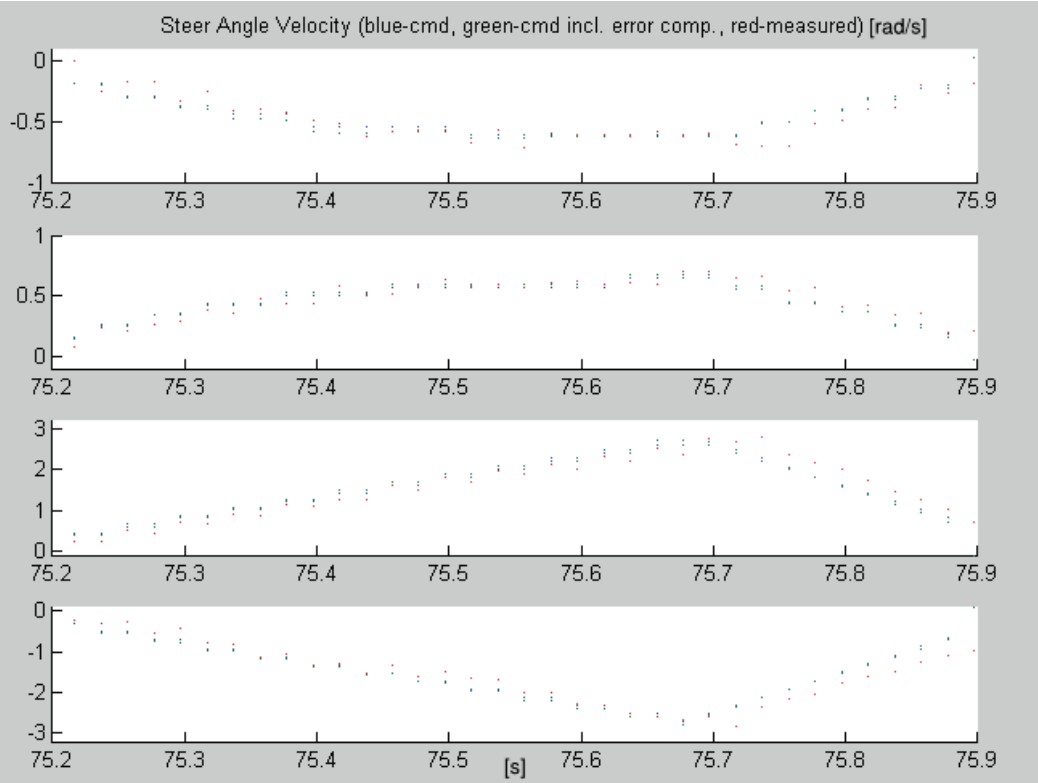


Figure 6.12: Steering angle velocities corresponding to the steering angles in Figure 6.11. The dotted lines represents commands from the trajectory generator, final commands including error compensation, and measured steering angles for all the wheel (1 in top graph and so on). The unit on the y-axes is radians per second, and the unit on the x-axes is seconds.

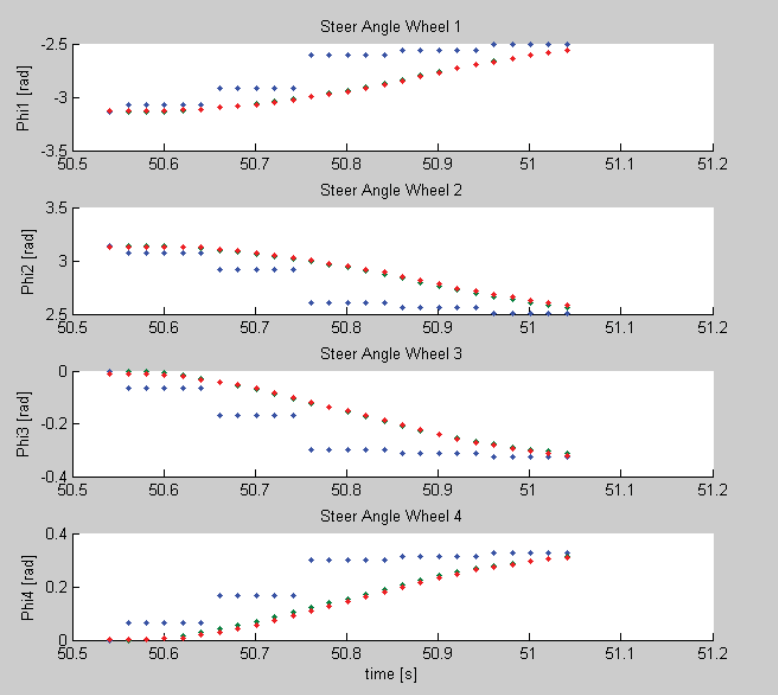


Figure 6.13: Steering angle targets, blue dots. Resulting commanded and measured angles from the new under-carriage control method are represented by red and green dots respectively. Each subfigure represents one wheel with wheel 1 in top graph and so on.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

As a tool to solve the wheel coordination issues described in Section 3.2, a steering dynamic model was constructed. To keep the model as simple as possible, only the, from the under-carriage control noticeable, limiting dynamics was included. By simulation with the available platform dynamic Simulink model, the limiting parameters could be found and modeled. The model was validated in the simulation environment, and considered representative enough to use in the new under-carriage control.

Due to the long sample time in the under-carriage control the possible approaches to improved wheel coordination is limited. Since influence on the coordination only can be done when the reference values to the motor drivers are set, once per control cycle, the best the coordination control can do is to make sure all driver references are followable for all wheels and aiming towards coordinated steer angles. To construct such a coordination control, an algorithm for online trajectory generation was successfully developed. The algorithm generates a trajectory for the robot instantaneous center of motion (ICM) in two main steps; construct a path for the ICM, generate trajectory points, within the dynamic limits, along the path. The algorithm was implemented on the robot, and experimental data shows that the wheel coordination does indeed improve with the new control approach.

The two step trajectory generator architecture is flexible and allows for many future improvements. The steering model could be replaced or extended to better capture the limiting dynamics. The currently simplest as possible path planner should preferably be replaced by a more complex version covering singularity avoidance and smooth paths generation. Research around similar areas are already taking place at the institute, mostly in con-

text of smooth ICM control, but without dynamic inclusion. Great potential to merge the trajectory generation concept with such systems is offered.

7.2 Future Work

Except from already suggested improvements on the developed control system, there are a lot more that can be done for a better under-carriage control.

7.2.1 Software Structure Modifications

The work for this thesis has been constrained to a method working without large scale system modifications. Using the currently quite long control cycle of the entire under-carriage control might not be the most optimal way to control the platform. If the wheel coordination part, including both steering angle and wheel velocities, would be separated from the rest and handled in a separate thread with a shorter cycle time, the coordination control could most likely be improved dramatically. By connecting the new thread to the robot control thread with proper real-time management, an efficient and robust under-carriage control could be achieved. The faster coordination loop would then allow for better use of the Elmo drivers abilities, such as motor torque feed-forward, which cannot be used in combination with the current control cycle time.

7.2.2 Wheel Force Control

In the current under carriage system, the wheel velocities are used as the only indication of the robot velocity. In this way, the wheel ground contact is approximately perfect. Even though higher control levels has information about robot position in relation to surrounding objects etc. from cameras and other sensors, the under-carriage control has no ability to adapt to changed drive conditions and associated potential problems, such as wheel slipping and surface deformation. By using torque measurements from the Elmo controllers and eventually even robot velocity measurements, if sensors providing such would be added, the wheel forces could be estimated, allowing for force control of the wheels.

To take the whole thing one step further, to provide ability to drive on troublesome surfaces, the force control could be extended with drive condition adaption. A state machine keeping track of drive conditions should then preferably be added to provide information to such adaption.

Bibliography

[1] C.P. Connette, A. Plott, M. Hgele, A. Verl, "Control of an Psedo-omnidirectionl, Non-holonomic, Moblie Robot based on an ICM Reprecentation in Spherical Coordinates", *IEEE Conference on Design and Control*, Cancun, Dec 2008, pp. 4976-4983

[2] C.P. Connette, C. Parlitz, B. Graf, M. Hgele, A. Verl, "The Mobility Concept of Care-O-Bot 3", *39th International Symphosium on Robotics*, Seoul, Oct 2008, pp. 746-748

[3] A.B. Reister, M.A. Unseren, "Position and constraint force control of a vehicle with two or more steerable drive wheels", *IEEE transaction on Robotics and Automatisation*, vol. 9, no. 6, Dec 1993, pp. 723-731

[4] P. Robuffo Giordano, M. Fuchs, A. Albu-Schffer, G. Hirzinger, "On the Kinematic Modeling and Control of a Mobile Platform Equipped with Steering Wheels and Movable Legs", *2009 IEEE International Conference on Robotics and Automatisation*, Kobe, Japan, May 2009, pp. 4080-4087

[5] J. Minguez, F. Lamiraux, J. Laumond, "Motion Planning and Obstacle Avoidance", *In: B. Siciliano, O. Khatib, (Eds.): Hand Book of Robotics, Springer*, 2008, ch. 35, pp. 827-852

[6] P. Morin, C. Samson, "Motion Control of Wheeled Mobile Robots", *In: B. Siciliano, O. Khatib, (Eds.): Hand Book of Robotics, Springer*, 2008, ch. 34, pp. 799-826

[7] G. Campion, G. Bastin, B. D'Andra-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots", *IEEE*, 1993, pp. 462-469

[8] B. Thuilot, B. D'Andra-Novel, A. Micaelli, "Modeling and Feedback Control of Mobile Robots Equipped with Several Steering Wheels",

IEEE Transaction on Robotics and Automation, vol. 12, no. 3, Jun 1996, pp. 375-390

[9] K. Macek, K. Thoma, R. Glatzel, R. Seigwart, "Dynamic Modeling and Parameter Identification for Autonomous Vehicle Navigation", *IEEE International Conference on Intelligent Robots and Systems*, San Diego, USA, Oct 2007, pp. 3321-3326

[10] R. K. Kankaanranta, H. N. Koivo, "Dynamics and Simulation of Compliant Motion of a Manipulator", *IEEE Transaction on Robotics and Automation*, vol. 12, no. 3, Jun 1996, pp. 375-390

[11] R. Holmberg, O. Khatib, "Development of a Holonomic Mobile Robot for Mobile Manipulation Tasks", *FSR International Conference on Field and Service Robotics*, Pittsburg, USA, Aug 1999

[12] T. Kroger, A. Tomiczek, F. M. Wahl, "Towards On-Line Trajectory Computation", *IEEE International Conference on Intelligent Robots and Systems*, Beijing, China, Oct 2006, pp. 736-741

[13] M. B. Milam, K. Mushambi, R. M. Murray, "A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems", *Conference on Decision and Control*, 2000

[14] G. Ishigami, K. Yoshida, Tohoku University Japan, "Steering Characteristics of an Exploration Rover on Loose Soil based on All-Wheel Dynamic Model", *Downloaded from IEEE Xplore*, June 2009

[15] C. Su, Y. Stepanenko, "Robust Motion/Force Control of Mechanical System with Classical Nonholonomic Constraints", *IEEE Transaction on Automatic Control*, vol. 39, no. 3, Mar 1994, pp. 609-614

[16] S. Jung, T. C. Hsia, "Explicit Lateral Force Control of an Autonomus Mobile Robot with Slip", *Downloaded from IEEE Xplore*, May 2009

[17] A. Kirsch, "Inedtifizierung und modellbasierte Regelung fur das Fahrwerk des Roboters Care-O-Bot 3", *Bachelorarbeit, Fraunhofer IPA*, Stuttgart, Germany, 2009

[18] Fraunhofer IPA, "Care-O-Bot", *www.care-o-bot.de/english/index.php*, 2009

[19] Fraunhofer IPA, "soft", *www.care-o-bot.de/english/Cob3_Software.php*, 2009

[20] Fraunhofer IPA, "Care-O-bot I", *www.care-o-bot.de/english/Care-O-bot_1.php*, 2009

[21] Fraunhofer IPA, "Care-O-bot II", *www.care-o-bot.de/english/Care-O-bot_2.php*, 2009

[22] Boston Dynamics, "Big Dog, The Most Advanced Rought Terrain Robot on Earth", *www.bostondynamics.com/robot_bigdog.html*, 2009

[23] Honda Motor Co., "ASIMO The Honda Humanoid Robot", *world.honda.com/ASIMO*, 2009

[24] iRobot Corporation, "iRobot Roomba Vacum Cleaning Robot", *www.irobot.com*, 2009

[25] Electrolux, "Trilobite 2.0", *www.trilobite.se*