Physical Simulation of the Dynamical Behavior of Three-Wheeled Omni-directional Robots

Hamid Rajaie, Reinhard Lafrenz, Oliver Zweigle, Uwe-Philipp Käppeler, Frank Schreiber, Thomas Rühr, Andreas Tamke, and Paul Levi

Institute of Parallel and Distributed Systems,
Universität Stuttgart, 70569 Stuttgart, Germany
robocup@informatik.uni-stuttgart.de
www.robocup.informatik.uni-stuttgart.de

Abstract. Hardware simulation is a very efficient way for parameter tuning. We developed a Simulink-based simulator for the navigation components of our robotic soccer team. This physical simulation has interfaces to be interconnected with the higher levels of the real control software and is therefore able to perform an overall simulation of single robots.

1 Introduction

Development of new control strategies for the driving behavior of mobile robots and extensive parameter studies on real hardware are time-consuming. To overcome this, we developed a Simulink-based simulator for the driving behavior of our soccer robots, where physical parameters of the robot and the environment can be modeled and which allows for off-line parameter studies. This physical simulation has network interfaces to be interconnected with the higher levels of the real robot control software, so that critical movements can be detected and the higher-level control can be adapted. Therefore, the system is able to perform an overall simulation of single robots.

We use Matlab and Simulink to design an integrated environment for rapid prototyping of algorithms, simulation, and modeling. Rapid prototyping is a mean for the evaluation of different control strategies, e.g., P, PID, or Fuzzy. We use the simulation environment to model the dynamic behavior of the robot and the underlying hardware components. To analyze these models, we successfully connected the physical simulator with the CoPS behavior control software, which runs on real hardware.

With the simulator, we can also analyze and visualize the different signals, e.g. motor currents and accelerations. This can be used to achieve good sets of parameters for the higher-level behavior software. For example, in robotic soccer, the driving behavior depends highly on the characteristics and conditions of the carpet. In addition, the driving behavior changes while dribbling the ball. Typical controllers like PID have several parameters, that affect the behavior. Finding a good set in the higher-order parameter space is always a critical and time-consuming procedure which can be accelerated using the simulation system.

U. Visser et al. (Eds.): RoboCup 2007, LNAI 5001, pp. 270–277, 2008.

The simulator is able to interpret messages in CAN-bus format, which is used for the real hardware. In this way, the higher levels of the software don't need to be changed to switch between simulation and real hardware. The simulation is interconnected to the robot software by TCP/IP communication.

2 Three-Wheeled Omnidirectional Drive

2.1 Robot Geometry Model

Fig. 1 shows the top view of our mobile robot model and the coordinate systems we use in the model. The use of the following different coordinate systems make the analysis of the model easier.

1. The World Coordinate System (WCS), attached to the field, is an inertial Cartesian coordinate system with the unit vectors $\mathbf{i}_w, \mathbf{j}_w, \mathbf{k}$.
2. The Local Coordinate System (LCS) is also a Cartesian coordinate system, but its X axis is the heading of the robot (angle β in WCS). The unit vectors are $\mathbf{i}_l, \mathbf{j}_l, \mathbf{k}$.
3. A Local Polar Coordinate Systems or $LPCS_i, i = 1, \ldots, 3$ attached to each leg. The unit vectors are $\mathbf{e}_r^i, \mathbf{e}_\theta^i, \mathbf{k}$.

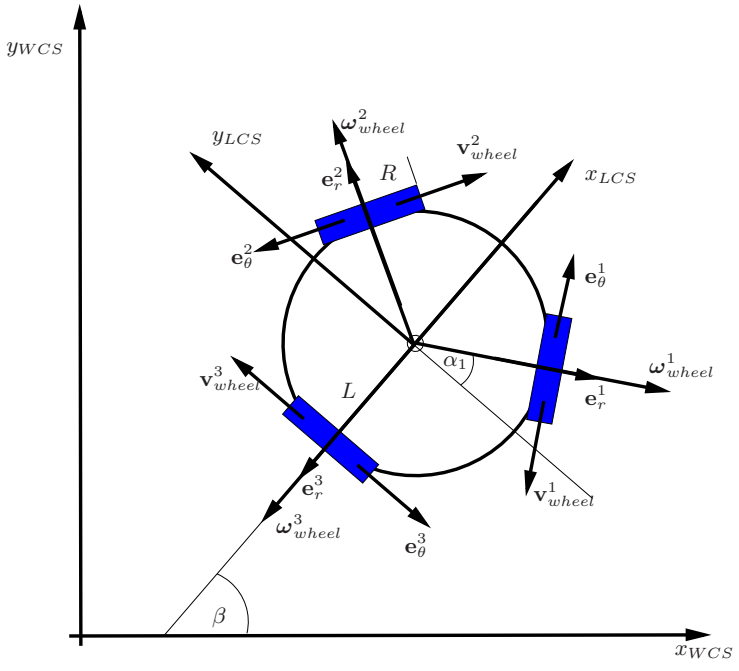

Fig. 1. Kinematics of the omnidirectional drive

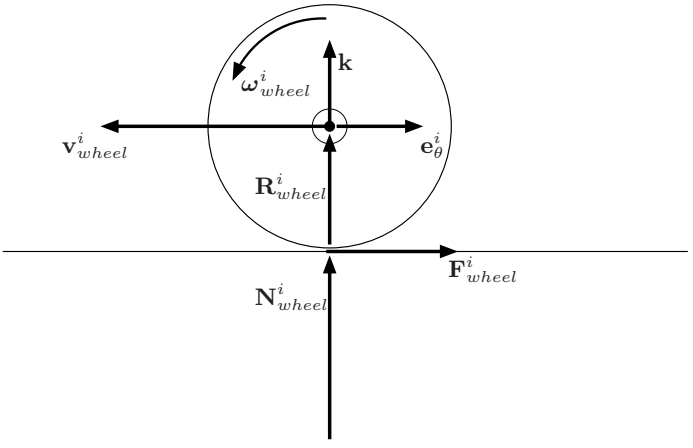

Fig. 2. Wheel model

2.2 Wheel Model

In order to access a better estimation of the dynamic behavior of the robot, we consider the effect of wheel slip. Fig. 2 shows the side view of a wheel. The view direction points to the center of the robot.

We assume that there is always a vertical force applied from the ground to each wheel, $\mathbf{N}^i_{wheel}$ as it is shown in Fig. 2. $\mathbf{N}^i_{wheel}$ is a result of the weight of the robot and its value depends on the position of the center of gravity of the robot and can be different for each wheel. We also assume that the horizontal force applied form the ground to the wheel $\mathbf{F}^i_{wheel}$ is always in the plane of the wheel, if it is not zero.

In case of a free rotation of the wheel i (no mechanical driving or breaking torque on the wheel) we can assume:

$$\mathbf{v}^i_{wheel} = \mathbf{R}^i_{wheel} \times \boldsymbol{\omega}^i_{wheel} \tag{1}$$

which leads according to Fig. 2 to

$$\mathbf{R}^i_{wheel} = R^i_{wheel}\mathbf{k}, \quad \boldsymbol{\omega}^i_{wheel} = \omega^i_{wheel}\mathbf{e}^i_r, \quad \mathbf{v}^i_{wheel} = v^i_{wheel}\mathbf{e}^i_\theta \tag{2}$$

Considering that if $\boldsymbol{\omega}^i_{wheel} \cdot \mathbf{e}^i_r$ is positive then $\mathbf{v}^i_{wheel} \cdot \mathbf{e}^i_\theta$ is negative. In this case the contact force between the wheel and the ground is zero.

$$\mathbf{F}^i_{wheel} = 0 \tag{3}$$

whereas in general

$$\mathbf{F}^i_{wheel} = F^i_{wheel}\mathbf{e}^i_\theta \tag{4}$$

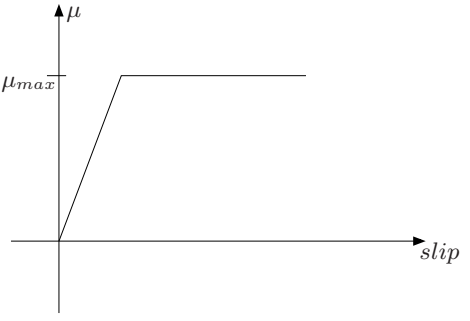

Fig. 3. Slip model

In cases where driving or breaking mechanical torques are applied on the wheel, the above equation is no longer valid, which means that slip occurs. The wheel slip velocity $\mathbf{v}_s^i$ is defined as:

$$\mathbf{v}_s^i = \mathbf{v}_{wheel}^i + \mathbf{R}_{wheel}^i \times \boldsymbol{\omega}_{wheel}^i \tag{5}$$

In order to quantify the slip, the following formula is also used as a mathematical definition.

$$\mathbf{s}^i = \frac{\mathbf{R}_{wheel}^i \times \boldsymbol{\omega}_{wheel}^i + \mathbf{v}_{wheel}^i}{\|\mathbf{v}_{wheel}^1\|} \cdot \mathbf{e}_\theta^i \tag{6}$$

In the case of a nonzero mechanical torque on the wheel axis, as a consequence of the slip, a contact force $\mathbf{F}_{wheel}^i$ between wheel and ground in the plane of wheel occurs. A simplified and useful model which relates the slip and the generated force is as follows:

$$\mathbf{F}_{wheel}^i = \mu^i \|\mathbf{N}_{wheel}^i\| \mathbf{e}_\theta^i \tag{7}$$

in which $\mathbf{F}_{wheel}^i$ is the generated force and $\mathbf{N}_{wheel}^i$ is the vertical force which is applied from the ground on the wheel i.

$$\mathbf{N}_{wheel}^i = N_{wheel}^i \mathbf{k} \tag{8}$$

In general, we assume that μ^i is a function of the slip velocity $\mathbf{v}_s^i$ according to [3]:

$$\mu^i = f(\mathbf{v}_s^i) \tag{9}$$

Using equations 7 and 9 we conclude that

$$\mathbf{F}_{wheel}^i = f(\mathbf{v}_s^i) \|\mathbf{N}_{wheel}^i\| \mathbf{e}_\theta^i \tag{10}$$

Fig. 3 shows the model we use in this paper as a relation between the slip and coefficient of friction, according to equation 9.

2.3 Kinematic and Dynamic Model

Due to the Newton's second law and according to Fig. 1 we get

$$\sum_{i=1}^{3} F^i_{wheel}\mathbf{e}^i_\theta = m_{robot}\mathbf{a}_{robot}, \qquad \sum_{i=1}^{3} r^i\mathbf{e}^i_r \times F^i_{wheel}\mathbf{e}^i_\theta = I_{robot}\boldsymbol{\alpha}_{robot} \tag{11}$$

The above two equations are written in the Local Coordinate System LCS. Considering equation 10 we can write:

$$\sum_{i=1}^{3} f(\mathbf{v}^i_s)\|\mathbf{N}^i\|\mathbf{e}^i_\theta = m_{robot}\mathbf{a}_{robot}, \qquad \sum_{i=1}^{3} r^i\mathbf{e}^i_r \times f(\mathbf{v}^i_s)\|\mathbf{N}^i\|\mathbf{e}^i_\theta = I_{robot}\boldsymbol{\alpha}_{robot} \tag{12}$$

2.4 DC Motor Model

The model which we used for a DC motor is the standard model with ideal motor and internal resistance. The rotation of the rotor of a DC motor generates an electromotive force U^i_{emf}. A linear algebraic equation relates the electromotive force and the rotor angular velocity of motor i:

$$U^i_{emf} = K_u\|\boldsymbol{\omega}^i_{motor}\| \tag{13}$$

According to Ohm's law the voltage drop due to the winding's resistance is

$$U^i_{resistance} = R^i_{motor}I^i \tag{14}$$

By applying the Kirchoff's law we get

$$U^i_{terminal} - U^i_{emf} = U^i_{resistance} \tag{15}$$

Putting equations 13 and 14 in equation 15 we conclude that for motor i

$$U^i_{terminal} - K_u\|\boldsymbol{\omega}^i_{motor}\| = R^i_{motor}I^i \tag{16}$$

Another very important relation in a DC motor is the linear relation between the mechanical load, T^i_{motor} and the electrical current I^i:

$$I^i = K_iT^i_{motor} \tag{17}$$

The equations 16 and 17 are used in our model to describe mathematically the behavior of our DC motors.

2.5 Motor Controller Model

The basic task of a motor controller is to regulate the angular velocity of the motor $\omega^i(t)$ equal to a set angular velocity $\omega^i_{motor\,set}$. We define the velocity error as

$$error^i(t) = \omega^i_{motor\,set} - \omega^i_{motor}(t) \tag{18}$$

The motor controller receives the set angular velocity from another module over the CAN bus and reads the current angular velocity from the digital encoders and calculate the error. Depending on the control strategy, in our case PID, it produces an output terminal voltage $U^i_{terminal}$ which is applied to the motor.

$$U^i_{terminal} = f^i_{PID}(error^i) \tag{19}$$

3 Simulation Package

Based on the mathematical models introduced in part 2, we implemented the Simulink model shown in Fig. 4. It consists of the following components:

The NETWORK block provides the set motor angular velocities $\omega^i_{motor_{set}}$. As Simulink runs, the code in this block creates a listening socket and waits for a request from the CoPS control software. For each velocity message from the CoPS client, the network block updates the set motor angular velocities $\omega^i_{motor_{set}}$. The velocity messages are based on the protocol which we use in our team as the communication protocol between the higher levels of the CoPS software and the motor controller hardware.

Each POWER TRANSMISSION block models a DC motor plus the planetary gearbox as well as the tooth belt and the motor controller based on equations 17 and 16. Each block gets the corresponding set angular velocity $\boldsymbol{\omega}^i_{motor_{set}}$ as input from the network block and the contact force signal $\mathbf{F}^i_{wheel}(t)$ as a feedback from the CONTACT FORCES block and outputs the wheels' angular velocities $\boldsymbol{\omega}^i_{wheel}(t)$.

The SLIP block gets the $\boldsymbol{\omega}^i_{wheel}(t)$ and $\mathbf{v}^i_{wheels}(t)$ as inputs and calculates $\mathbf{v}^i_s(t)$ based on Eq. 5.

The FRICTION COEFFICIENTS block calculates the $\mu^i(t)$ from the $\mathbf{v}^i_s$ based on Eq. 9.

The CONTACT FORCES block calculates $\mathbf{F}^i_{wheel}(t)$ based on Eq. 7. In this block the nonuniform weight distribution of the robot can be modeled.

The NEWTON block accepts the $\mathbf{F}^i_{wheel}(t)$ from the contact forces block as inputs and calculates the robot velocity $\mathbf{v}^i_{robot}(t)_{LCS}$ in the local coordinate system based on equations **??** and 11.

The FIELD block receives the $\mathbf{v}^i_{robot}(t)_{LCS}$ from NEWTON block and visualizes the play field and the robot on it.

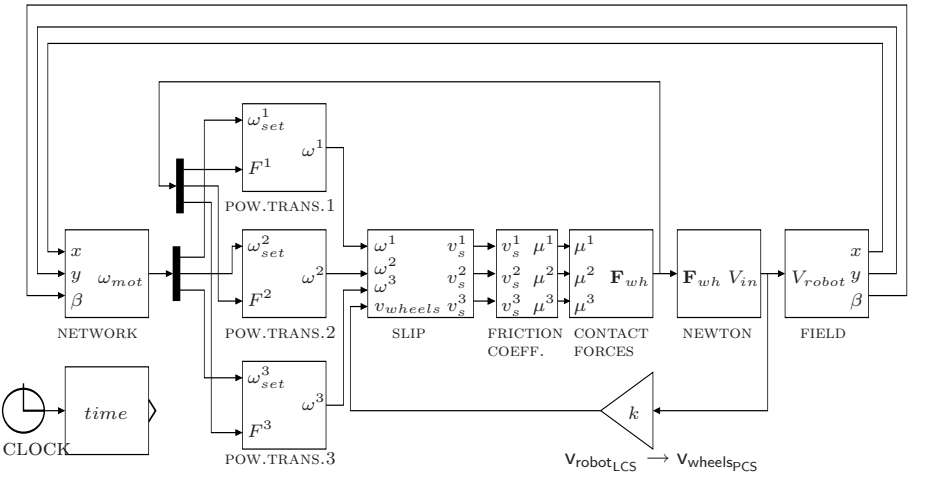

Fig. 4. Simulink model for the 3-wheel motor control

The CLOCK provides a soft real time mechanism which synchronizes the model simulation clock with the internal clock of the computer.

4 Experiments and Results

The result of the simulation for a robot with $m = 17kg$,$I = 0.227kgm^{-2}$ is shown in fig. 5. In the simulation we have assumed a non-uniform distribution of weight load on the wheels:

$$\|\mathbf{N}^1_{wheel}\| = 60.43N, \quad \|\mathbf{N}^2_{wheel}\| = 50.63N, \quad \|\mathbf{N}^3_{wheel}\| = 55.53N, \tag{20}$$

Also it is assumed that $\mu_{max} = 0.25$, as shown in Fig. 3. The initial velocity of the robot is zero, and the set velocity is

$$\mathbf{v}_{robot_{LCS}} = (2\,m/s,\ 0\,m/s,\ 0\,rad/s) \tag{21}$$

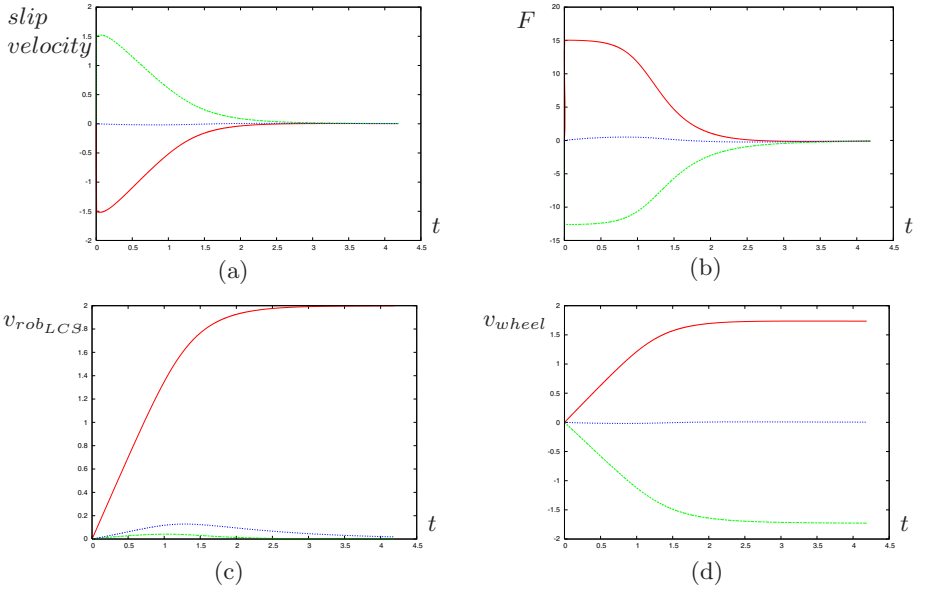

Fig. 5. Simulation result for the robot during acceleration: (a) slip velocity for the 3 wheels $[m/s]$, (b) horizontal forces for the 3 wheels $[N]$, (c) robot velocity in local coordinate system $[(m/s, m/s, rad/s)]$, (d) wheel velocities $[rad/s]$

As shown in the Fig. 5(a), at the beginning of the movement, the slip velocities of wheels 1 and 2 are high and their absolute values are not equal. The unequal slip values are a direct consequence of the non-uniform weight distribution. Therefore the traction forces on wheels 1 and 2 are not equal and therefor robot moves on a curve until the traction forces become equal. This force transition phase is shown in Fig. 5(b). As shown in Fig. 5(c),the the robot velocity reaches the set velocity $(2m/s, 0m/s, 0rad/s)$ after the transition phase.

5 Conclusion and Outlook

We presented a dynamic model for a mobile robot equipped with omnidirectional wheels, considering the wheel slip and a non-uniform distribution of weight. The network interface allows us to connect our CoPS software with the model through a the TCP/IP link. The CoPS software can communicate with the model, e.g. send *set velocity* commands and query the model for signals like velocities, positions and motor currents.

In this way, we can tune our control software for a better control over our hardware, and also implement new control strategies rapidly in our model before moving to the actual implementation in theCoPS control software.

In the near future, we want to extend the model in order to simulate not only a single robot, but a team of robots. Additionally, we want also to simulate the ball and other objects in the field, as well as components like the kicking device. Future work will also include the possibility of learning these parameters using other sensory data then odometry, e.g. the global self-localization.

References

1. Buchheim, T., Kindermann, G., Lafrenz, R., Levi, P.: A dynamic environment modelling framework for selective attention. In: Visser, et al. (eds.) Proceedings of the IJCAI 2003 Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments (2003)
2. Craig, J.J.: Introduction to Robotics - Mechanics and Control, 3rd edn. Pearson Prentice Hall (2005)
3. Williams II, R.L., Carter, B.E., Gallina, P., Rosati, G.: Dynamic Model With Slip for Wheeled Omnidirectional Robots. IEEE Transactions on Robotics and Automation 18(3), 285–293 (2002)
4. Zweigle, O., Käppeler, U.-P., Lafrenz, R., Rajaie, H., Schreiber, F., Levi, P.: Situation recognition for reactive agent behavior. In: Artificial Intelligence and Soft Computing (ASC) (2006)
5. Zweigle, O., Lafrenz, R., Buchheim, T., Käppeler, U., Rajaie, H., Schreiber, F., Levi, P.: Cooperative Agent Behavior Based on Special Interaction Nets. In: Arai, T. (ed.) Intelligent Autonomous Systems 9 (IAS-9) (2006)