Available online at www.sciencedirect.com
SCIENCE DIRECT®
Robotics and Autonomous Systems 54 (2006) 13–22
Robotics and Autonomous Systems
www.elsevier.com/locate/robot

Trajectory generation and control for four wheeled omnidirectional vehicles

Oliver Purwin*, Raffaello D'Andrea

Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853, USA

Received 1 October 2004; received in revised form 4 October 2005; accepted 7 October 2005
Available online 28 November 2005

Abstract

This paper describes an algorithm to calculate near-optimal minimum time trajectories for four wheeled omnidirectional vehicles, which can be used as part of a high-level path planner. The algorithm is based on a relaxed optimal control problem. It takes limited friction and vehicle dynamics into account, as encountered in high-performance omnidirectional vehicles. The low computational complexity makes the application in real-time feasible. An implementation of the algorithm on a real vehicle is presented and discussed.

Keywords: Trajectory generation; Real-time path planning; Omnidirectional vehicle; Bang-bang control

1. Introduction

Omnidirectional vehicles have some desirable properties: they are very maneuverable, able to navigate tight quarters, and have few constraints on path planning. The small-size league of the annual RoboCup competition is an example of a highly dynamic environment where omnidirectional vehicles have been employed extremely successfully since 2000 [5,6].

Path planning in general is a difficult task, especially when considering vehicle dynamics and moving obstacles. Rapidly changing environments require a re-computation of the path in real-time. There are many approaches to solve this problem, which are based on different assumptions about the hardware and the environment. Paromtchik and Rembold [10] and Muñoz et al. [11], for example, used a sequence of splines to generate a path which includes waypoints. The splines contained time information, so that the vehicle's desired velocity could be limited depending on the hardware used. Faiz and Agrawal [12] approximated the set of all feasible states of the system with polytopes, which took the dynamics and other constraints into account. Moore and Flann [7] presented a trajectory generation algorithm for an off-road vehicle. The basis for the path generator was a set of mission goals that had to be achieved. They used an A* algorithm to determine trajectories as a combination of steps, ramps, decaying exponentials, and sinusoidal functions. Watanabe et al. [8] used a resolved acceleration approach on their omnidirectional robotic platform. They inverted the dynamics of the system and implemented a PI or PD controller with a feedforward term to minimize the error between desired and achieved trajectory. Liu et al. [13] implemented a method called trajectory linearization control (TLC), which is based on linearization along the desired trajectory and inversion of the dynamics. Kalmár-Nagy et al. [4] developed a trajectory generation algorithm which computed a minimum time path based on the dynamics of the vehicle and the motor characteristics.

With the recent advancements in robot hardware, some of the basic assumptions for generating optimal paths have changed. Robots can accelerate at much higher rates than they used to. Some robots are no longer power but friction limited, since the drive motors can deliver enough torque to make the wheels slip even at high velocities. Furthermore, due to the high accelerations the effect of weight transfer becomes more significant. Weight transfer can cause the normal forces between the wheels and the ground to change, thus altering the available amount of traction and the maximum acceleration.

The objective of this paper is to present a trajectory generation algorithm for high-performance omnidirectional vehicles. The algorithm computes the minimum time trajectory from a given initial state to a given final state while taking limited friction and weight transfer into account. The output

* Corresponding author.
E-mail address: op24@cornell.edu (O. Purwin).
URL: http://www.people.cornell.edu/pages/op24/ (O. Purwin).

0921-8890/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2005.10.002

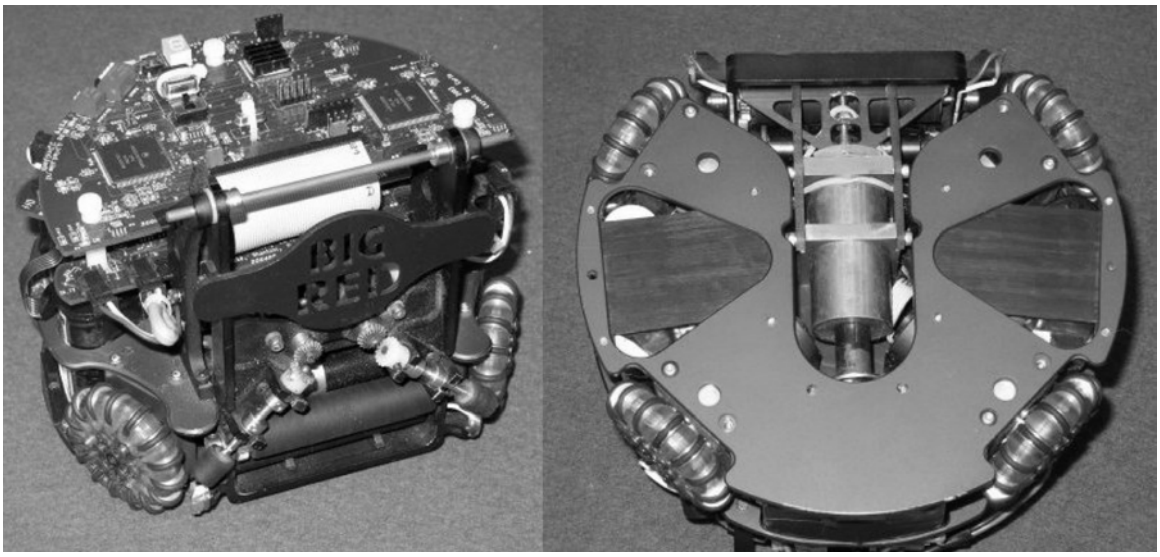

Fig. 1. 2003 Cornell RoboCup robot (left), robot wheelbase (right).

is a sequence of velocities for the vehicle, which have to be tracked by low-level control. This algorithm can readily be used as part of a high-level path planning algorithm, for example as the obstacle-free guidance system in [9].

The algorithm is designed for vehicles with four powered wheels moving on a plane surface. These vehicles are overactuated, which means that different combinations of control inputs can have the same net effect on the system. The problem of overactuated vehicles is discussed in [16] and several techniques are given of how to allocate the control efforts in order to cause the desired system response.

In the derivation of the presented algorithm ideal control of the actuators is assumed, i.e., motor torques can be chosen arbitrarily within the physical limits. On a real vehicle the wheel forces would be determined by a dedicated controller [5], which is beyond the scope of this paper.

This paper is organized in the following way. Section 2 describes the assumptions made and covers the derivation of the vehicle equations of motion. In Section 3 the model of the vehicle dynamics is simplified and the rotational and translational degrees of freedom (DOFs) are decoupled. The result is an acceleration profile, which is independent of the vehicle orientation. Section 4 presents a solution to the relaxed optimal control problem, i.e., finding a minimum time solution to drive the system to the desired final destination given the previously derived vehicle characteristics. Section 5 describes the performance of the algorithm in simulation, while Section 6 covers results from the implementation on a real vehicle of the Cornell RoboCup system.

2. Vehicle dynamics

The basis for the following derivations is a four wheeled omnidirectional vehicle; see Fig. 1. The drive modules are equally spaced at 90°. The center of mass (CM) is assumed to be exactly above the geometrical center of the drive system.

2.1. Motor characteristics

The main assumption here is that the acceleration of the vehicle is friction limited, which means that the maximum acceleration can be achieved over the entire velocity range. This is a reasonable approximation for vehicles which are equipped with strong drive motors and do not have much room to accelerate or decelerate, as discussed in [5].

2.2. Friction force and weight transfer

Friction has been modelled as Coulomb friction. For more sophisticated friction models, see Olsson et al. [14], for example. The maximum acceleration force a wheel can exert is $f_a = \mu n$, where n is the normal force between the wheel and the ground, and μ is the coefficient of friction. In general, the normal force n is not only a function of vehicle mass and geometry but it depends on the current acceleration vector of the vehicle. This effect is called weight transfer [1]. It means that during an acceleration phase the weight distribution on the wheels changes due to the inertia of the robot mass. For example, when accelerating in the forward direction the normal force of the front wheels is reduced while at the same time the rear wheels are loaded more heavily.

2.3. Derivation of equations of motion

The first step is to derive the equations of motion which govern the vehicle's dynamics. Fig. 2 depicts the free-body diagram of the vehicle. The global coordinate system is defined by x, y, and z. The vehicle frame of reference is defined by x_r, y_r, and z_r. The angle θ is the rotation of the vehicle in the x–y plane, i.e., it is the rotation of the local coordinates with respect to the global coordinate system. The mass of the vehicle is m. The forces n_i are the normal forces between the wheels and the ground, and the forces f_i are the friction forces. The positions of the wheels with respect to the CM are defined by the vectors $\mathbf{P}_i$:

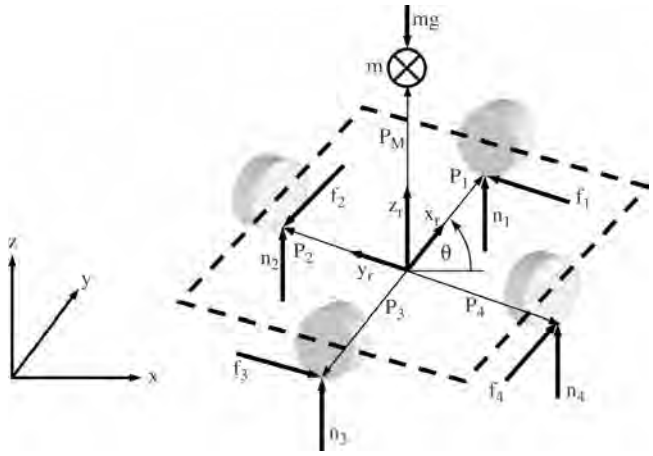

Fig. 2. Free-body diagram.

$$\mathbf{P}_1 = l\begin{bmatrix}1\\0\\0\end{bmatrix},\ \mathbf{P}_2 = l\begin{bmatrix}0\\1\\0\end{bmatrix},\ \mathbf{P}_3 = l\begin{bmatrix}-1\\0\\0\end{bmatrix},\ \mathbf{P}_4 = l\begin{bmatrix}0\\-1\\0\end{bmatrix}. \tag{1}$$

The parameter l describes the distance from the geometrical center to the wheels. The driven directions $\mathbf{D}_i$ of the wheels are orthogonal to the position vectors $\mathbf{P}_i$

$$\mathbf{D}_1 = \begin{bmatrix}0\\1\\0\end{bmatrix},\ \mathbf{D}_2 = \begin{bmatrix}-1\\0\\0\end{bmatrix},\ \mathbf{D}_3 = \begin{bmatrix}0\\-1\\0\end{bmatrix},\ \mathbf{D}_4 = \begin{bmatrix}1\\0\\0\end{bmatrix}. \tag{2}$$

The CM of the vehicle is at

$$\mathbf{P}_\mathrm{M} = h\begin{bmatrix}0\\0\\1\end{bmatrix}. \tag{3}$$

The rotation matrix $\mathbf{R}(\theta)$ relates the local (vehicle) frame of reference (FOR) to the global (Newtonian) FOR:

$$\mathbf{R}(\theta) = \begin{bmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}. \tag{4}$$

Taking the force and moment balance in the global FOR yields two sets of equations that define the vehicle dynamics:

$$\begin{bmatrix}m\ddot{x}\\ m\ddot{y}\\ m\ddot{z}\end{bmatrix} = \mathbf{R}(\theta)\left(\sum_i f_i\mathbf{D}_i + \sum_j n_j\begin{bmatrix}0\\0\\1\end{bmatrix} + mg\begin{bmatrix}0\\0\\-1\end{bmatrix}\right) \tag{5}$$

$$\begin{bmatrix}J_x\ddot{\theta}_x\\ J_y\ddot{\theta}_y\\ J\ddot{\theta}\end{bmatrix} = \mathbf{R}(\theta) \times \left(\sum_i(-\mathbf{P}_\mathrm{M}+\mathbf{P}_i)\times f_i\mathbf{D}_i + \sum_j(-\mathbf{P}_\mathrm{M}+\mathbf{P}_j)\times n_j\begin{bmatrix}0\\0\\1\end{bmatrix}\right) \tag{6}$$

where J is the moment of inertia. Subscripts $\bullet_x$ and $\bullet_y$ indicate rotation about the coordinates x and y respectively. At this point the assumption is made that the normal forces are always positive, i.e., the vehicle does not tip. Thus

$$\ddot{z} = \ddot{\theta}_x = \ddot{\theta}_y = 0. \tag{7}$$

The equations of motion in the x–y plane are

$$m\ddot{x} = \cos\theta(f_4 - f_2) - \sin\theta(f_1 - f_3) \tag{8}$$

$$m\ddot{y} = \sin\theta(f_4 - f_2) + \cos\theta(f_1 - f_3) \tag{9}$$

$$J\ddot{\theta} = l\sum_i f_i \tag{10}$$

where the wheel forces f_i can be arbitrarily chosen within the limits posed by the maximum friction forces

$$|f_i| \le f_{i,\max} = \mu n_i. \tag{11}$$

The acceleration envelope $\mathcal{A}_0$ is defined as the boundary of the set of all feasible combinations of $\ddot{x}$, $\ddot{y}$, and $\ddot{\theta}$. Within the envelope, every combination of $\ddot{x}$, $\ddot{y}$, and $\ddot{\theta}$ can be achieved by the vehicle.

In order to find the acceleration envelope of the vehicle, (8) through (10) have to be solved in terms of the friction forces f_i, which are functions of the normal forces n_i. Therefore, expressions for the normal forces have to be found first. Eqs. (5) and (6) only yield six equations for the seven unknowns. In addition to the equations of motion it is assumed that the normal forces are distributed as follows:

$$n_1 + n_3 = n_2 + n_4. \tag{12}$$

For a derivation of (12), see Appendix A. In order to find explicit expressions for the normal forces n_i, the moment balances (6) have to be pre-multiplied by $\mathbf{R}^{-1}(\theta)$. The inverse of $\mathbf{R}(\theta)$ always exists, since $\mathbf{R}(\theta)$ is non-singular for all $\theta \in [0, 2\pi]$ [3]. With assumption (7) this leads to

$$f_1 h - f_3 h + n_2 l - n_4 l = 0 \tag{13}$$

$$f_2 h - f_4 h - n_1 l + n_3 l = 0. \tag{14}$$

The system of equations consisting of (12)–(14), and the force balance in the z direction from (5) is solved for n_i in order to yield

$$\begin{bmatrix}n_1\\n_2\\n_3\\n_4\end{bmatrix} = \frac{1}{4l}\begin{bmatrix}2h(f_2-f_4)+lmg\\ 2h(f_3-f_1)+lmg\\ 2h(-f_2+f_4)+lmg\\ 2h(-f_3+f_1)+lmg\end{bmatrix}. \tag{15}$$

At this point the control inputs u_i are introduced, which are a measure of how much torque the motors provide.

$$f_i = u_i f_{i,\max}, \qquad u_i \in [-1, 1]. \tag{16}$$

Eqs. (11) and (15) are substituted into (16), which leads to implicit expressions for f_i. At the same time, the terms $f_2 - f_4$ and $f_3 - f_1$ are replaced by acceleration terms from (8) and (9):

$$\begin{bmatrix}f_1\\f_2\\f_3\\f_4\end{bmatrix} = \frac{m\mu}{4l}\begin{bmatrix}u_1(-2h(\ddot{x}\cos\theta+\ddot{y}\sin\theta)+lg)\\ u_2(-2h(-\ddot{x}\sin\theta+\ddot{y}\cos\theta)+lg)\\ u_3(2h(\ddot{x}\cos\theta+\ddot{y}\sin\theta)+lg)\\ u_4(2h(-\ddot{x}\sin\theta+\ddot{y}\cos\theta)+lg)\end{bmatrix},$$
$$u_i \in [-1, 1]. \tag{17}$$

Table 1
Sample values for an omnidirectional vehicle

Parameter	μ	g	m	J	l	h
Value	0.8	9.81 m/s^2	2.7 kg	0.0085 m^2 kg	0.08 m	0.05 m

The goal is to find expressions for $\ddot{x}$, $\ddot{y}$, and $\ddot{\theta}$ which are only functions of u_i, μ, m, h, l, g, and θ. In order to achieve this, (17) is substituted back into the equations of motion (8)–(10). The result is a system of coupled differential equations, which is only dependent upon accelerations and the control efforts. All terms containing normal or friction forces have been eliminated.

$$\begin{aligned}\ddot{x} = \frac{\mu}{4l}[&lg((u_4-u_2)\cos\theta-(u_1-u_3)\sin\theta)\\&+2h(\ddot{x}(u_1+u_3-u_2-u_4)\cos\theta\sin\theta\\&+\ddot{y}((u_2+u_4)\cos^2\theta+(u_1+u_3)\sin^2\theta))] \qquad (18)\\ \ddot{y} = \frac{\mu}{4l}[&lg((u_4-u_2)\sin\theta+(u_1-u_3)\cos\theta)\\&+2h(\ddot{x}(-(u_2+u_4)\sin^2\theta-(u_1+u_3)\cos^2\theta)\\&+\ddot{y}(-u_1-u_3+u_2+u_4)\cos\theta\sin\theta)] \qquad (19)\\ \ddot{\theta} = \frac{\mu m}{4J}[&lg(u_1+u_2+u_3+u_4)\\&+2h(\ddot{x}((u_3-u_1)\cos\theta+(u_2-u_4)\sin\theta)\\&+\ddot{y}((u_4-u_2)\cos\theta+(u_3-u_1)\sin\theta))]. \qquad (20)\end{aligned}$$

Solving (18) through (20) explicitly for $\ddot{x}$, $\ddot{y}$, and $\ddot{\theta}$ yields

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{\theta}\end{bmatrix} = \mathbf{R}(\theta)\frac{lg\mu}{4l^2+h^2\mu^2(u_2+u_4)(u_1+u_3)}$$
$$\begin{bmatrix} l(u_4-u_2)+0.5\mu h(u_1-u_3)(u_2+u_4)\\ l(u_1-u_3)+0.5\mu h(u_2-u_4)(u_1+u_3)\\ m/J\left(h^2\mu^2\sum_i \frac{u_1u_2u_3u_4}{u_i}+l^2\sum_i u_i\right)\end{bmatrix},$$
$$u_i \in [-1,1]. \qquad (21)$$

These nonlinear equations in u_i describe the set of admissible accelerations of the vehicle in the global frame of reference. The acceleration envelope is discussed in more detail in the next section.

3. Simplification of the acceleration envelope

Since (21) is nonlinear in u_i, a numerical approach is chosen in order to find the envelope. Discretizing the control efforts u_i, solving the equations using sample parameters, and plotting the results yields a discretized set of admissible accelerations. This approximation can be arbitrarily close to the continuous envelope, depending on the resolution of the discretization. Table 1 holds the parameters for the omnidirectional vehicle depicted in Fig. 1, which is also used in Section 6 to show the implementation of the trajectory generation on an actual vehicle. Fig. 3 shows the envelope for the sample case with $\theta = 0$. Depending on the given parameters the dimensions vary, but the characteristic shape is always the same. The resulting

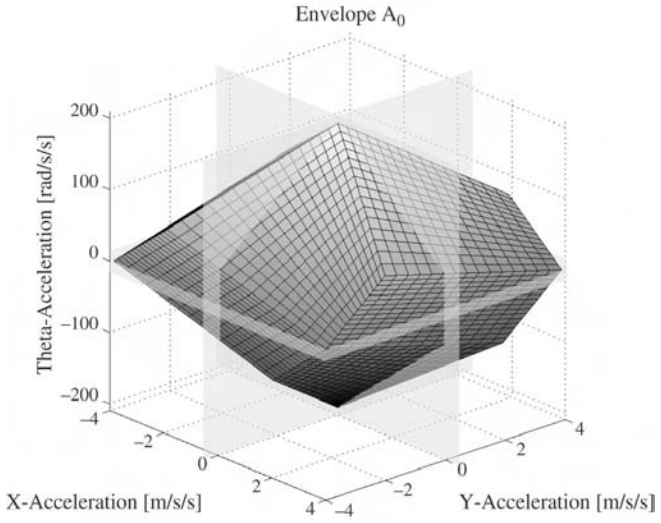

Fig. 3. Acceleration envelope $\mathcal{A}_0$.

acceleration envelope $\mathcal{A}_0$ is not rotationally symmetric, which means that the maximum magnitude of the acceleration is determined by the direction of the acceleration vector. This dependency makes it harder to compute the vehicle's minimum time trajectory. In order to find a closed form solution, the acceleration envelope is restricted, similarly to [4]. By taking the intersection of the acceleration envelopes for all directions θ, the influence of the orientation is removed:

$$\mathcal{Q} = \bigcap_{\theta\in[0,2\pi]} \mathbf{R}(\theta)\mathcal{A}_0. \qquad (22)$$

$\mathcal{Q}$ defines the admissible set of accelerations that the vehicle can achieve, independent of the current orientation θ. The result for the sample envelope is depicted in Fig. 4. Within $\mathcal{Q}$, any arbitrary combination of the three acceleration components can be chosen. It should be noted that the accelerations are coupled through the control efforts u_i. In order to make the optimal control problem in Section 4 faster to solve for real-time applications, the envelope is further simplified by decoupling the rotational DOF θ from the two linear DOFs. This is achieved by imposing a limit on the rotational and translational accelerations: $|\ddot{\theta}| \le \ddot{\theta}_{\max}$ and $\sqrt{\ddot{x}^2+\ddot{y}^2} \le a_{\max}$, where $a_{\max}$ is the maximum linear acceleration at $\ddot{\theta} = \ddot{\theta}_{\max}$. This reduces the envelope to a cylinder with radius $a_{\max}$, as illustrated in Fig. 4. The presented algorithm therefore assumes independent control of translation and rotation, as described in [5]. However, the algorithm can easily be extended to not making this simplification.

In the following section, the resulting optimal control problem for the simplified envelope is posed and a solution is presented.

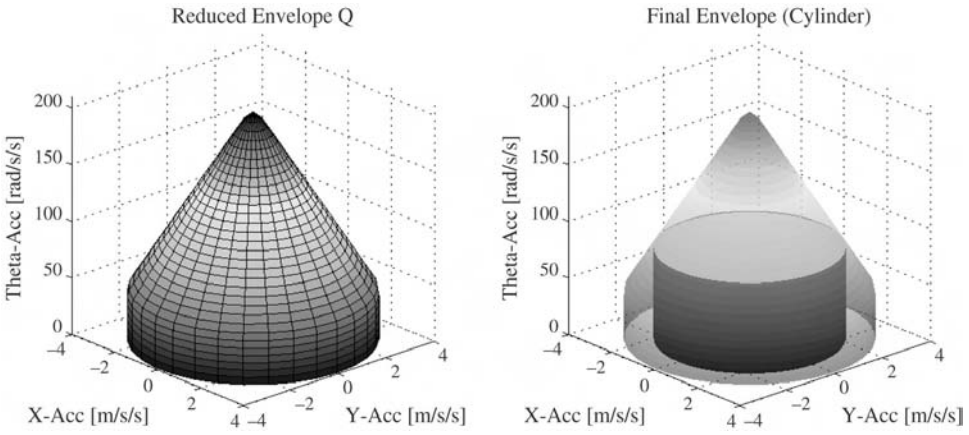

Fig. 4. Reduced envelope $\mathcal{Q}$ (left), final cylindrical envelope (right).

4. The optimal control problem

For the remainder of this paper the vehicle rotation will be neglected, since it can be treated as a simplification of the translational cases. The objective is to find the minimum time path for the two linear DOFs, x and y, from a given initial state to a desired final state. The dynamics have been simplified to a double integrator subject to limitations on the acceleration and velocity. The state of the system consists of positions and velocities. The final velocity is always chosen to be zero in order to avoid discontinuities in the solutions when approaching the desired final state; see [2,4]. It should be noted that when applying the algorithm in practice the vehicle hardly ever slows down to zero velocity since the destination will be changed continually depending on the environment [5,6].

By definition, all accelerations inside the acceleration envelope can be achieved. With the simplifications made in Section 3 individual wheel forces are no longer required to describe the system dynamics. The inputs to the simplified system are the accelerations in x and y directions instead. The allocation of torques to the wheels is the task of a separate controller, which is not within the scope of this paper.

The optimal control problem is to minimize the time t_f for the system

$$\ddot{x}(t) = q_x(t) \tag{23}$$

$$\ddot{y}(t) = q_y(t) \tag{24}$$

with initial and final conditions

$$x(0) = 0, \quad x(t_\mathrm{f}) = x_\mathrm{f}, \quad \dot{x}(0) = \dot{x}_0, \quad \dot{x}(t_\mathrm{f}) = 0 \tag{25}$$

$$y(0) = 0, \quad y(t_\mathrm{f}) = y_\mathrm{f}, \quad \dot{y}(0) = \dot{y}_0, \quad \dot{y}(t_\mathrm{f}) = 0 \tag{26}$$

subject to constraints on the control effort and the state

$$\sqrt{q_x^2(t) + q_y^2(t)} \le a_\mathrm{max} \tag{27}$$

$$\sqrt{\dot{x}^2(t) + \dot{y}^2(t)} \le v_\mathrm{max} \tag{28}$$

where $q_x(t)$ and $q_y(t)$ are the control efforts in the x and y directions, t_f is the execution time, x_f and y_f are the final positions, and $\dot{x}_0$ and $\dot{y}_0$ are the initial velocities. The maximum acceleration a_max is the radius of the cylindrical envelope from the previous section (see Fig. 4); the maximum velocity v_max is defined by the motor specifications.

In order to make the problem more tractable, each DOF is handled independently at first. In the end, the final times of both DOFs are synchronized. Introducing a general DOF w, the problem can be written as

$$\ddot{w}(t) = q_w(t) \tag{29}$$

$$w(0) = 0, \quad w(t_\mathrm{f}) = w_\mathrm{f}, \quad \dot{w}(0) = \dot{w}_0, \quad \dot{w}(t_\mathrm{f}) = 0 \tag{30}$$

$$|\dot{w}(t)| \le v_{w,\mathrm{max}}, \quad |q_w(t)| \le a_{w,\mathrm{max}}. \tag{31}$$

The minimum time solution to this problem occurs on the boundary of the velocity and/or acceleration constraints [2]. This means that at any time the vehicle is following one of three strategies:

- accelerating: $q_w(t) = a_{w,\mathrm{max}}$
- decelerating: $q_w(t) = -a_{w,\mathrm{max}}$
- cruising: $|\dot{w}(t)| = v_{w,\mathrm{max}}$, $q_w(t) = 0$.

In order to find a complete solution the problem can be broken down into a combination of several distinct cases. Each case represents a possible state or condition of the system. These conditions are mutually exclusive: at any given time the system is in one and only one of these cases, depending on $\dot{w}_0$, w_f, $v_{w,\mathrm{max}}$, and $a_{w,\mathrm{max}}$. Each case has a control effort associated with it (e.g. case 1: $q_w(t) = a_{w,\mathrm{max}}$). The strategy is to apply this control effort until the conditions for a different case are satisfied or the final destination is reached with zero final velocity. The complete solution is therefore a sequence of cases.

The problem is normalized to $w_\mathrm{f} \ge 0$ by inverting the signs of w_f and $\dot{w}_0$ if w_f is negative. Thus, the possible cases are reduced to the five cases shown in Fig. 5. Case 1 covers all initial conditions where $\dot{w}_0 < 0$: the vehicle is initially moving away from the destination and has to accelerate with $q_w(t) = a_{w,\mathrm{max}}$ in order to reverse direction. Case 3 is applicable if $\dot{w}_0 > v_{w,\mathrm{max}}$. The vehicle is moving too fast and has to decelerate with maximum control effort. This case is possible since $v_{w,\mathrm{max}}$ is not necessarily a hard physical constraint, but rather a desired maximum velocity which should not be exceeded. Furthermore, the constraint $v_{w,\mathrm{max}}$ can be decreased artificially as part of the synchronization process of the final times; see below. For

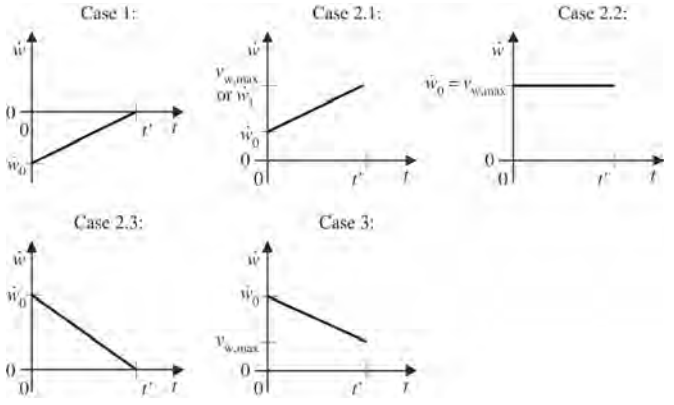

Fig. 5. Possible optimal control cases.

all other instances there are three choices left (acceleration, deceleration, coasting) depending on how far the destination is and how fast the vehicle moves initially. These three choices are covered by cases 2.1, 2.2, and 2.3. A possible sequence could look as follows.

Example 1. The vehicle is far away from the destination and moving slowly towards it. It will accelerate until it reaches $v_{w,\max}$ (case 2.1), cruise at $v_{w,\max}$ (case 2.2), and finally decelerate such that it reaches the destination with exactly zero final velocity (case 2.3).

Given the applied control effort and the initial conditions, it is possible to find a closed form solution for the final state of each case. The following list contains the requirements for each particular case, the applied control effort $q_w(t)$ associated with that case, the execution time t', the travelled distance $w' = w(t')$, and the final velocity $\dot{w}' = \dot{w}(t')$.

Case 1. $\dot{w}_0 < 0$

The initial velocity is negative; the vehicle has to accelerate with maximal control effort until it reaches $\dot{w}(t) = 0$.

$$q_w(t) = a_{w,\max}, \qquad t' = -\frac{\dot{w}_0}{a_{w,\max}} \tag{32}$$

$$w' = \dot{w}_0 t' + \frac{a_{w,\max}}{2} t'^2 = -\frac{\dot{w}_0^2}{2a_{w,\max}}, \quad \dot{w}' = 0. \tag{33}$$

Case 2.1. $v_{w,\max} > \dot{w}_0 \geq 0$ and $w_f > \frac{\dot{w}_0^2}{2a_{w,\max}}$.

The vehicle has to accelerate, either because the destination is far away or the initial velocity is small. This case has two subcases. In subcase I, case 2.1 is followed by case 2.2; the vehicle reaches $v_{w,\max}$ and is cruising. In subcase II, case 2.1 is followed by case 2.3 and the vehicle decelerates until it reaches the final destination; see Fig. 6. Define t_1 as the time when the vehicle has to start decelerating in order to avoid overshooting the destination. Also define $w_1 = w(t_1)$ and $\dot{w}_1 = \dot{w}(t_1)$. It follows that

$$\begin{aligned} w_1 &= \dot{w}_0 t_1 + \frac{a_{w,\max} t_1^2}{2}, \qquad t_1 = \frac{\dot{w}_1 - \dot{w}_0}{a_{w,\max}} \\ &= \dot{w}_0 \frac{\dot{w}_1 - \dot{w}_0}{a_{w,\max}} + \frac{(\dot{w}_1 - \dot{w}_0)^2}{2a_{w,\max}} \end{aligned} \tag{34}$$

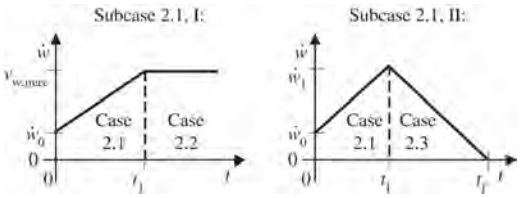

Fig. 6. The subcases of case 2.1.

$$w_f - w_1 = \frac{\dot{w}_1^2}{2a_{w,\max}}. \tag{35}$$

Adding (34) and (35) and solving for $\dot{w}_1$ yields

$$\dot{w}_1 = \sqrt{w_f a_{w,\max} + \frac{\dot{w}_0^2}{2}} \tag{36}$$

where $\dot{w}_1$ has to be positive by definition. The decision, which of the two subcases is applicable, is based on a comparison of the time t_I to reach $v_{w,\max}$ and the time t_{II} when the vehicle has to decelerate.

$$t_I = \frac{v_{w,\max} - \dot{w}_0}{a_{w,\max}}, \qquad t_{II} = \frac{\dot{w}_1 - \dot{w}_0}{a_{w,\max}}. \tag{37}$$

If $t_I < t_{II}$ then the vehicle reaches $v_{w,\max}$ and

$$q_w(t) = a_{w,\max}, \quad t' = t_I \tag{38}$$

$$w' = \dot{w}_0 t' + \frac{a_{w,\max}}{2} t'^2 = \frac{v_{w,\max}^2 - \dot{w}_0^2}{2a_{w,\max}}, \ \dot{w}' = v_{w,\max}; \tag{39}$$

otherwise it has to brake before it reaches $v_{w,\max}$ and

$$q_w(t) = a_{w,\max}, \qquad t' = t_{II} \tag{40}$$

$$w' = w_1 = \frac{w_f}{2} + \frac{\dot{w}_0^2}{2a_{w,\max}}, \quad \dot{w}' = \dot{w}_1. \tag{41}$$

Case 2.2. $\dot{w}_0 = v_{w,\max}$ and $w_f > \frac{\dot{w}_0^2}{2a_{w,\max}}$.

The vehicle is cruising at maximum velocity until it has to decelerate.

$$q_w(t) = 0, \qquad t' = \frac{w_f}{v_{w,\max}} - \frac{v_{w,\max}}{2a_{w,\max}} \tag{42}$$

$$w' = w_f - \frac{v_{w,\max}^2}{2a_{w,\max}}, \qquad \dot{w}' = v_{w,\max}. \tag{43}$$

Case 2.3. $v_{w,\max} \geq \dot{w}_0 > 0$ and $w_f \leq \frac{\dot{w}_0^2}{2a_{w,\max}}$.

The vehicle has to decelerate until it reaches zero final velocity.

$$q_w(t) = -a_{w,\max}, \qquad t' = \frac{\dot{w}_0}{a_{w,\max}} \tag{44}$$

$$w' = \frac{\dot{w}_0^2}{2a_{w,\max}}, \qquad \dot{w}' = 0. \tag{45}$$

Case 3. $\dot{w}_0 > v_{w,\max}$.

The vehicle moves faster than the allowed maximum velocity. This can be caused by the iterative solution procedure presented

below. The vehicle has to decelerate until it reaches the allowed velocity.

$$q_w(t) = -a_{w,\max}, \qquad t' = \frac{\dot{w}_0 - v_{w,\max}}{a_{w,\max}} \tag{46}$$

$$w' = \frac{1}{2a_{w,\max}}(\dot{w}_0^2 - v_{w,\max}^2), \qquad \dot{w}' = v_{w,\max}. \tag{47}$$

The procedure to find the complete solution is as follows.

(1) Define initial and final states, set $t = 0$
Normalize: $\dot{w}_0 = \text{sign}(w_\text{f})\dot{w}_0, \quad w_\text{f} = \text{sign}(w_\text{f})w_\text{f}$
(2) Check which case is applicable, based on the initial conditions
(3) Compute: $q_w(t)$, t', w', and $\dot{w}'$
(4) Set: $t = t + t'$, $w_\text{f} = w_\text{f} - w'$, $\dot{w}_0 = \dot{w}'$
Normalize: $\dot{w}_0 = \text{sign}(w_\text{f})\dot{w}_0$, $w_\text{f} = \text{sign}(w_\text{f})w_\text{f}$
(5) If the destination is not reached with zero final velocity: Go to (2)
(6) Total time to destination $t_{\text{f},w} = t$.

The result of this algorithm is a minimum time trajectory for a single DOF. Given a particular instance of the problem (23) through (28), the solutions in x and y will yield different execution times $t_{\text{f},x}$ and $t_{\text{f},y}$ in general. In order to get the minimum time to destination for the vehicle the solutions have to be synchronized. This is done by adjusting the maximum allowed control effort and velocity for both DOFs via the parameter $\alpha \in (0, \pi/2)$:

$$a_{x,\max} = a_{\max}\cos\alpha, \qquad v_{x,\max} = v_{\max}\cos\alpha \tag{48}$$

$$a_{y,\max} = a_{\max}\sin\alpha, \qquad v_{y,\max} = v_{\max}\sin\alpha. \tag{49}$$

Eqs. (48) and (49) satisfy the constraints (27) and (28). If either $x_\text{f} = 0$ or $y_\text{f} = 0$ with zero initial velocity, no synchronization is needed. Thus the exclusion of 0 and $\pi/2$ from α. The execution times $t_{\text{f},x}$ and $t_{\text{f},y}$ are continuous and strictly monotonously increasing/decreasing functions of α [17]. Therefore it is possible to use a bisection algorithm to find α.

(1) Initial guess: $\alpha = \pi/4$, $\alpha_\text{min} = 0$, $\alpha_\text{max} = \pi/2$
(2) Find minimum time trajectories for both x and y coordinates, given α
(3) If $|t_{\text{f},x} - t_{\text{f},y}|$ is sufficiently small, keep the solutions and stop the search
(4) If $(t_{\text{f},x} > t_{\text{f},y})$, set $\alpha = (\alpha - \alpha_\text{min})/2$, $\alpha_\text{max} = \alpha$
(5) If $(t_{\text{f},x} < t_{\text{f},y})$, set $\alpha = (\alpha_\text{max} - \alpha)/2$, $\alpha_\text{min} = \alpha$
(6) Go back to step (2).

Using bisection the difference between the two execution times can be made arbitrarily small. An alternative is to keep the number of iterations constant, effectively limiting the search time in a real-time application. The optimal value α_opt is defined as the α for which $t_{\text{f},x} = t_{\text{f},y}$. The difference between α_opt and α is bounded by

$$|\alpha - \alpha_\text{opt}| \leq \frac{\pi}{2^{N+1}} \tag{50}$$

where N is the number of iterations of the bisection. Therefore, the order of the search algorithm is $O(\log_2 N)$. It should be noted that the solution depends continuously on the initial and final conditions, i.e., small changes in the initial and final conditions only cause small changes in the solution. This property makes the solution robust to disturbances and noise.

5. Performance of the algorithm in simulation

This section describes how the implementation of the algorithm performs. One important aspect is the computation time. The recursive algorithm presented in the previous chapter was implemented in C ++ and tested on a Pentium 4 (1.7 GHz clock speed). A Monte Carlo simulation yielded average computation times of 94 μs, with a standard deviation of 28 μs. Note that there are guarantees on the maximum number of operations per iteration, since a sequence can only consists of a maximum of five cases in a row.

Another point is the quality of the solutions of the proposed algorithm in terms of the execution time t_f. The derivation of the algorithm involves several simplifications and assumptions, which greatly reduce the required computational effort but at the same time increase the execution times of the found trajectories. The solution of the proposed algorithm $t_{\text{f,approx}}$ is compared against two different benchmarks. Both are computed using RIOTS [18], an optimal control toolbox written in Matlab and C. The Matlab code for the simulation can be found at the author's web site [19]. It should be noted that parts of the simulation can only be run in conjunction with the RIOTS engine which is a commercial product and is therefore not included.

RIOTS can solve optimal control problems with constraints on the state and the control effort. The first benchmark is the execution time $t_{\text{f,full}}$ of the full problem, i.e., (21) subject to the velocity constraint (28). The second benchmark is the execution time $t_{\text{f,2D}}$ of the two-dimensional problem (23) to (28). The parameters of the sample vehicle (Fig. 1, Table 1) are used in all simulations. The maximum velocity and acceleration are limited to $v_\text{max} = 2.0$ m/s and $a_\text{max} = 3.92$ m/s^2. The velocity limit is required to maintain the friction limit assumption. The maximum acceleration is found by limiting the rotational acceleration to $\ddot{\theta}_\text{max} = 44.9$ rad/s^2 and determining the intersection with the envelope (see Fig. 4). If the rotational acceleration was any larger it would reduce the maximum allowed translational acceleration.

A Monte Carlo simulation was performed by generating random initial conditions and computing $t_{\text{f,approx}}$, $t_{\text{f,full}}$, and $t_{\text{f,2D}}$. Fig. 7 shows the simulation results. The abscissa depicts the ratio r_tf of the execution times $t_{\text{f,full}}/t_{\text{f,approx}}$ or $t_{\text{f,2D}}/t_{\text{f,approx}}$ respectively. The ordinate shows the fraction n_i/n_tot, where n_tot is the total number of solutions. The variable n_i is the number of solutions for which the execution time ratio $t_{\text{f,RIOTS}}/t_{\text{f,approx}}$ is smaller than or equal to r_tf. That means that a fraction of n_i/n_tot of all solutions have a ratio of final times of r_tf or better. Summarizing, the abscissa depicts the quality of the results, while the ordinate shows what fraction of the solutions achieves that quality.

As expected, the solution of the full problem yielded the smallest execution times, since it used the unreduced accelera-

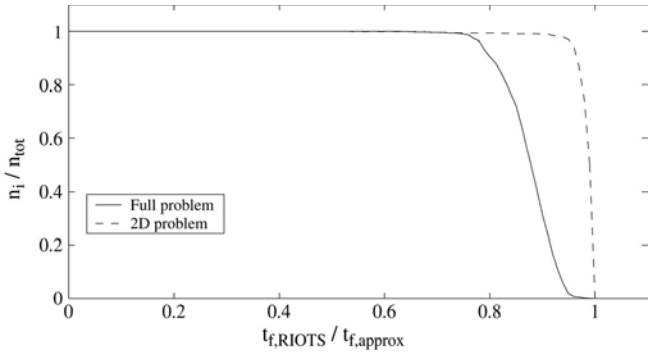

Fig. 7. Comparison of execution times.

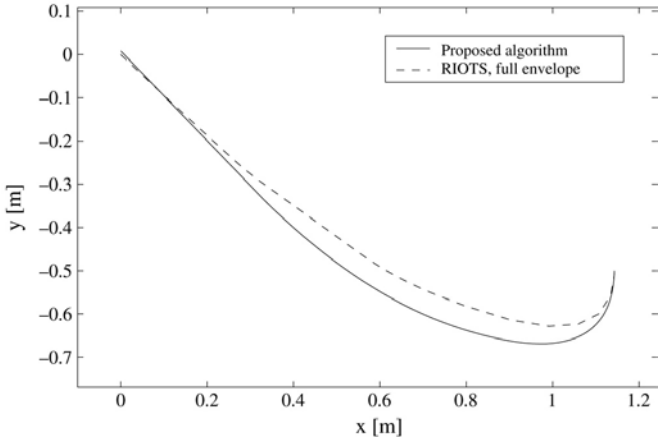

Fig. 8. Comparison of trajectories.

tion envelope. Due to the simplifications and relaxations the execution times $t_{f,approx}$ are longest, but not by much. More than 94% of $t_{f,approx}$ are within 96% of $t_{f,2D}$. When comparing to the solutions of the full problem, more than 85% of $t_{f,approx}$ are within 82% of $t_{f,full}$. On the other hand, the reduction in computation time is significant: on the same computer the average RIOTS solution (both the full and the 2D solution) took more than 2 minutes while the simplified algorithm was executed in about 100 μs.

Fig. 8 shows a comparison of a RIOTS trajectory (using the full envelope) and the trajectory generated by the presented algorithm. Plotted are x versus y positions of the following representative example:

$$x(0) = 1.143 \text{ m},\ x(t_f) = 0.0 \text{ m},\ \dot{x}(0) = 0 \text{ m/s},\ \dot{x}(t_f) = 0 \text{ m/s} \quad (51)$$

$$y(0) = 0.5 \text{ m},\ y(t_f) = 0.0 \text{ m},\ \dot{y}(0) = -1.0 \text{ m/s},\ \dot{y}(t_f) = 0 \text{ m/s}. \quad (52)$$

6. Implementation

The new trajectory generation algorithm was implemented on the Cornell RoboCup system. RoboCup is a game of completely autonomous robotic soccer. The main sensor of the system is an overhead camera which takes pictures of the playing field at a rate of 60 Hz. The vision system

Table 2
Destinations for implementation test

Destination	x-Coord (m)	y-Coord (m)
A	−1.0	−0.5
B	1.0	−0.5
C	0.0	0.5

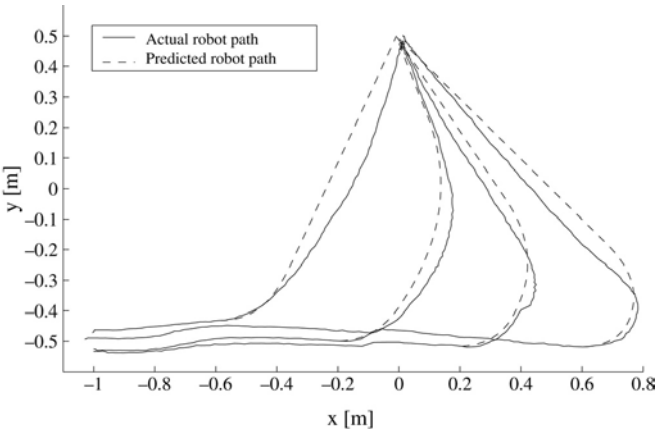

Fig. 9. Implementation on robot.

analyzes these frames and determines the robots' positions and velocities. This data is passed on to the strategy module, which makes the decisions where to move the robots. Trajectory generation then computes minimum time paths, which are recomputed every frame in order to give immediate feedback. The velocity commands corresponding to these paths are sent to the robots. The robots track the velocity commands using essentially PI controllers for the wheel speeds.

In order to test the performance of the proposed trajectory generation algorithm, a robot was commanded to move along a line from A to B, using trajectory generation. The rotation was held fixed at $\theta = 0$ rad. When the robot crossed a particular x coordinate x_c, the final destination was changed to C, so that the robot had to alter its course while moving. Four different situations were tested, with $x_{c,1} = -0.6$ m, $x_{c,2} = -0.2$ m, $x_{c,3} = 0.2$ m, and $x_{c,4} = 0.6$ m. Table 2 contains the destinations, and Fig. 9 depicts the results. Video clips of the vehicle executing the test pattern can be found at [19]. The figure shows two paths for each x_c. The solid lines stand for the paths actually taken by the robots. The trajectories and vehicle commands are recomputed every frame to compensate for process and sensor noise. The dashed lines are the paths that were computed in the frame when the destination was changed from B to C.

The theoretical and actual paths deviate at most by about 0.1 m. This might seem a large error in comparison to the length of the entire trajectory, but it should be kept in mind that the vehicle is not tracking the dashed trajectories for more than one frame. The trajectories are recomputed every frame, which means that errors enter only in the form of slightly changed initial conditions. On the other hand, for a short time horizon ($\approx$ 0.3 s) the deviations between precomputed and actually

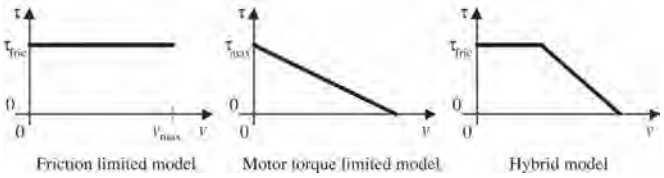

Fig. 10. Torque-speed graphs.

taken trajectories are about an order of magnitude smaller. The errors are sufficiently small for effective obstacle avoidance and ball control in the small-size league of the RoboCup competition, where the algorithm was very successfully applied [5].

7. Conclusion

A trajectory generation algorithm for omnidirectional vehicles has been presented. The algorithm takes the vehicle dynamics, limited friction, and weight transfer into account. It is tailored to high-performance vehicles that are mainly friction limited. It offers a computationally efficient way to calculate minimum time trajectories, which are close to the optimal solutions. In order to prove the feasibility of the concept, the algorithm was successfully applied to a real vehicle of the Cornell RoboCup system.

During the derivation of this algorithm certain assumptions were made about the location of the center of mass and the wheel positions. In practice, these locations are subject to manufacturing tolerances and therefore are not perfectly well known. These deficiencies could be the topic of further investigations. In the case that the CM is not centered, for example, additional terms are introduced in (6).

During the derivation of the vehicle dynamics it was assumed that the motors could always provide sufficient torque to make the wheels slip. In reality, this assumption will break down for most vehicles at high speeds. An interesting approach would be to combine limited friction with a motor model such as presented in [4]. This would mean that at low speeds the motors can provide enough torque τ to make the wheels slip (τ_{fric}), but if the wheel velocity v gets larger than a certain threshold the maximum torque drops linearly; see Fig. 10.

Also, this paper could be extended to non-zero final velocities. This would require finding means of handling the discontinuities that arise when approaching the final destination. There still remains great potential for future research in the area of trajectory generation for omnidirectional vehicles.

Appendix A. Assumption about vehicle weight distribution

During the derivation of the vehicle equations of motion it is assumed that

$$n_1 + n_3 = n_2 + n_4. \tag{A.1}$$

This is motivated by the rigid pillar problem as presented in [15]. A vehicle with four wheels is statically undetermined if the wheels and suspension are considered to be rigid. In order to derive (A.1), the wheels are treated as linear springs with a very large spring constant k, such that $\ddot{z} = \ddot{\theta}_x = \ddot{\theta}_y = 0$ still holds to first order. The exact magnitude of the spring constant is not relevant (as long as it is positive and not infinite), since it will cancel out. The vehicle chassis is modelled as a rigid body, as shown in Fig. A.1.

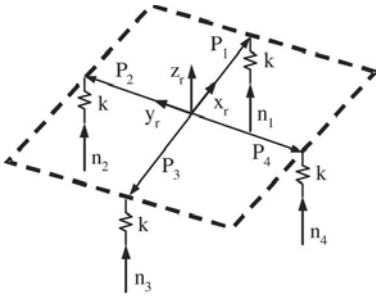

Fig. A.1. Normal force model.

For small rotations, the displacements d_i of the springs can be written as functions of the linear robot displacement z_r and the two rotations about the x_r and y_r axes, $\theta_{x,\text{r}}$ and $\theta_{y,\text{r}}$ respectively.

$$d_1 = z_\text{r} - l\theta_{y,\text{r}} \tag{A.2}$$
$$d_2 = z_\text{r} + l\theta_{x,\text{r}} \tag{A.3}$$
$$d_3 = z_\text{r} + l\theta_{y,\text{r}} \tag{A.4}$$
$$d_4 = z_\text{r} - l\theta_{x,\text{r}} \tag{A.5}$$
$$kd_i = n_i. \tag{A.6}$$

Solving (A.2) through (A.6) for the normal forces and adding n_1 and n_3 (n_2 and n_4 respectively) yields

$$n_1 + n_3 = 2kz_\text{r} \tag{A.7}$$
$$n_2 + n_4 = 2kz_\text{r} \tag{A.8}$$

which completes the derivation of (A.1).

References

[1] T.D. Gillespie, Fundamentals of Vehicle Dynamics, Society of Automotive Engineers, 1992.
[2] A.E. Bryson, Y.-C. Ho, Applied Optimal Control, John Wiley & Sons, 1975.
[3] C.F. Moon, Applied Dynamics, Wiley-Interscience, 1998.
[4] T. Kalmár-Nagy, R. D'Andrea, P. Ganguly, Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle, Robotics and Autonomous Systems 46 (2004) 47–64.
[5] O. Purwin, R. D'Andrea, Cornell Big Red 2003, in: D. Polani, A. Bonarini, B. Browning, K. Yoshida (Eds.), Robocup 2003: Robot Soccer World Cup VII, in: Lecture Notes in Artificial Intelligence, Springer, Berlin, 2003.
[6] R. D'Andrea, T. Kalmár-Nagy, P. Ganguly, M. Babish, in: P. Stone, T. Balch, Kraetzschmar (Eds.), The Cornell RoboCup Team, in: Robocup 2000: Robot Soccer World Cup IV, Springer, Berlin, 2001.
[7] K.L. Moore, N.S. Flann, A six-wheeled omnidirectional autonomous mobile robot, Control Systems Magazine, IEEE 20 (6) (2000) 53–66.
[8] K. Watanabe, Y. Shiraishi, S.G. Tzafestas, J. Tang, T. Fukuda, Feedback

control of an omnidirectional autonomous platform for mobile service robots, Journal of Intelligent and Robotic Systems 22 (3–4) (1998) 315–330.
[9] E. Frazzoli, M.A. Dahleh, E. Feron, Real-time motion planning for agile autonomous vehicles, Journal of Guidance, Control, and Dynamics 25 (1) (2002) 116–129 (14).
[10] I.E. Paromtchik, U. Rembold, A practical approach to motion generation and control for an omnidirectional mobile robot, in: Proceedings of IEEE International Conference on Robotics and Automation, vol. 4, 1994, pp. 2790–2795.
[11] V. Muñoz, A. Ollero, M. Prado, A. Simón, Mobile robot trajectory planning with dynamic and kinematic constraints, in: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 4, 1994, pp. 2802–2807.
[12] N. Faiz, S.K. Agrawal, Trajectory planning of robots with dynamics and inequalities, in: Proceedings of the 2000 IEEE International Conference on Robotics and Automation, vol. 4, 2000, pp. 3976–3982.
[13] Y. Liu, X. Wu, J. Zhu, J. Lew, Omni-directional mobile robot controller design by trajectory linearization, in: Proceedings of the American Control Conference 2003, vol. 4 (4–6), 2003, pp. 3423–3428.
[14] H. Olsson, K.J. Aström, C. Canudas de Wit, M. Gäfvert, P. Lischinsky, Friction Models and Friction Compensation.
[15] C.M. Bender, D.C. Brody, B.K. Meister, Quantised three pillar problem, ArXiv Quantum Physics e-prints (February) (2003) arXiv:quant-ph/0302097. http://adsabs.harvard.edu/cgi-bin/nph-bib_query?bibcode=2003quant.ph..2097B&db_key=PRE.
[16] J. Plumlee, Multi-input ground vehicle control using quadratic programming based control allocation techniques, M.S. Thesis, Auburn University, Auburn, Alabama, August 2004.
[17] O. Purwin, R. D'Andrea, Continuity and monotonicity properties of an optimally controlled double integrator with state and input constraints, TR2005-2001, Cornell University Library Technical Reports and Papers, 2005, http://techreports.library.cornell.edu.
[18] A. Schwartz, E. Polak, Y. Chen, RIOTS 95, Optimal Control Toolbox for Matlab V6.5.
[19] http://control.mae.cornell.edu/Purwin/PhDWork.htm.

Oliver Purwin was born in Frankfurt, Germany, in 1975. He received a Master of Engineering degree from Cornell University in 2001 and graduated with a Diplom-Ingenieur in mechanical engineering from the Darmstadt University of Technology in 2002. Since then he has been a graduate student in the Department of Mechanical Engineering at Cornell University. He has been involved with the Cornell Autonomous Robot Soccer team in 2001 (third place), 2003 (first place), and 2005 (second place). His main research interest is system identification and control of an autonomous helicopter.

Raffaello D'Andrea was born in Pordenone, Italy, in 1967. He received his BASc. degree in Engineering Science from the University of Toronto in 1991, and M.S. and Ph.D. degrees in Electrical Engineering from the California Institute of Technology in 1992 and 1997, respectively. Since then, he has been with the Department of Mechanical and Aerospace Engineering at Cornell University, where he is an Associate Professor. He is also Vice President of Systems Architecture for Distrobot Systems Incorporated.

He has been a recipient of the University of Toronto W.S. Wilson Medal in Engineering Science, an IEEE Conference on Decision and Control best student paper award, an American Control Council O. Hugo Schuck Best Paper award, an NSF CAREER Award, and a DOD sponsored Presidential Early Career Award for Scientists and Engineers. He was the system architect and faculty advisor of the RoboCup world champion Cornell Autonomous Robot Soccer team in 1999 (Stockholm, Sweden), 2000 (Melbourne, Australia), 2002 (Fukuoka, Japan), and 2003 (Padova, Italy), and third place winner in 2001 (Seattle, USA). His recent collaboration with Canadian artist Max Dean, "The Table", an interactive installation, appeared in the Biennale di Venezia in 2001. It was on display at the National Gallery of Canada in Ottawa, Ontario, in 2002 and 2003, and is now part of its permanent collection.