Trajectory Generation for Four Wheeled Omnidirectional Vehicles

Oliver Purwin and Raffaello D'Andrea
Sibley School of Mechanical and Aerospace Engineering
Cornell University
Ithaca, NY 14853, USA
op24@cornell.edu rd28@cornell.edu

Abstract— **This paper describes an algorithm to calculate near-optimal minimum time trajectories for omnidirectional vehicles, which can be used as part of a high-level path planner. The basis is an optimal control algorithm with relaxations to reduce the required computational effort. It takes limited friction and vehicle dynamics into account, as encountered in high-performance omnidirectional vehicles. The algorithm is written with modest computational resources in mind, so that it is possible to control an omnidirectional vehicle in real time in a fast paced environment.**

I. Introduction

Omnidirectional vehicles are becoming increasingly popular, as they have some distinct advantages over their non-holonomic counterparts: They are more maneuverable, able to navigate tight quarters, and are easier to control. 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, see [2] [3].

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. Muñoz et al. [4], 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 used hardware. Faiz and Agrawal [5] approximated the set of all feasible states of the system with polytopes, which took the dynamics and other constraints into account. Moore and Flann [6] 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. [7] 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. [8] 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. [1] 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 and the motors cannot deliver their maximum torque because the wheels are slipping. 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 is a sequence of velocities for the vehicle, which have to be tracked by low-level control. This primitive can readily be used as part of a high-level path planning algorithm, as was actually done in [9] for example.

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

II. 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 degrees. The center of mass (CM) is assumed to be exactly above the geometrical center of the drive system.

A. Motor Characteristics

The presented trajectory generation is based upon the assumption that the vehicle's acceleration is friction lim-

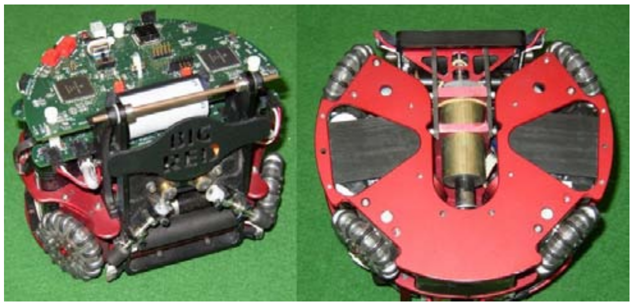
Fig. 1. 2003 Cornell RoboCup robot (left), robot wheelbase (right)

ited. This 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 don't have much room to accelerate or decelerate, for example [2].

B. Friction Force

Friction has been modelled as Coulomb friction. Although there are more sophisticated friction models, see Olsson et al. [10] for example, this approach suffices as a first order approximation, as evident by the performance of our algorithm on actual vehicles, see Section VI.

C. Derivation of Equations of Motion

In order to find the acceleration envelope it is necessary to derive the equations of motion which govern the vehicle's behavior. Fig. 2 depicts the free-body diagram of the vehicle. The global coordinate system is defined by x, y,

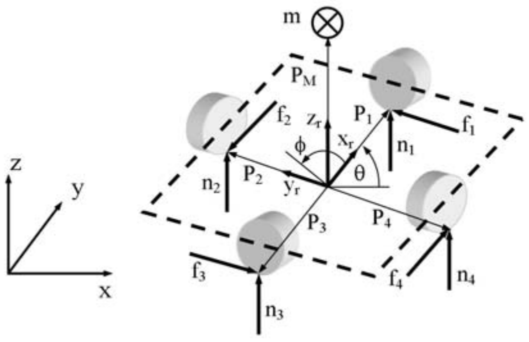

Fig. 2. Free body diagram

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 ones. The angle ϕ is the direction the vehicle is accelerating in. The mass of the vehicle is m. The forces n_i are the normal forces between the wheels and the ground, the forces f_i are the friction forces which accelerate the vehicle. The positions of the wheels with respect to the CM are defined by the vectors $\mathbf{P}_i$:

$$\mathbf{P}_1 = \begin{bmatrix} l \\ 0 \\ 0 \end{bmatrix}, \mathbf{P}_2 = \begin{bmatrix} 0 \\ l \\ 0 \end{bmatrix}, \mathbf{P}_3 = \begin{bmatrix} -l \\ 0 \\ 0 \end{bmatrix}, \mathbf{P}_4 = \begin{bmatrix} 0 \\ -l \\ 0 \end{bmatrix} \quad (1)$$

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 $\mathbf{P}_i$.

$$\mathbf{D}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \mathbf{D}_2 = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{D}_3 = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \mathbf{D}_4 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (2)$$

The CM of the vehicle is at $\mathbf{P}_M = [0\ 0\ h]^{\mathrm{T}}$. The rotation matrix $\mathbf{R}(\theta)$ relates the local (vehicle) frame of reference (FOR) to the global (Newtonian) FOR:

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

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

$$\begin{bmatrix} m\ddot{x} \\ m\ddot{y} \\ m\ddot{z} \end{bmatrix} = \mathbf{R}\left(\sum_i f_i \mathbf{D}_i + \sum_j n_j \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}\right) \quad (4)$$

$$\begin{bmatrix} J_x\ddot{\theta}_x \\ J_y\ddot{\theta}_y \\ J\ddot{\theta} \end{bmatrix} = \mathbf{R}\left(\sum_i (-\mathbf{P}_M + \mathbf{P}_i) \times f_i \mathbf{D}_i + \sum_j (-\mathbf{P}_M + \mathbf{P}_j) \times n_j \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right) \quad (5)$$

with J being the moment of inertia and subscripts $\bullet_x$ and $\bullet_y$ indicating 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

$$\ddot{z} = \ddot{\theta}_x = \ddot{\theta}_y = 0 \quad (6)$$

The equations of motion in the x-y plane are

$$m\ddot{x} = \cos\theta(f_4 - f_2) - \sin\theta(f_1 - f_3) \quad (7)$$
$$m\ddot{y} = \sin\theta(f_4 - f_2) + \cos\theta(f_1 - f_3) \quad (8)$$
$$J\ddot{\theta} = l\sum_i f_i \quad (9)$$

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

$$|f_i| \leq f_{i,max} = \mu n_i \quad (10)$$

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, (7) through (9) 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. Equations (4) and (5) only yield 6 equations for the 7 unknowns. In addition to the equations of motion it is assumed that the body of the vehicle is rigid while the wheels act as linear springs. This leads to

$$n_1 + n_3 = n_2 + n_4 \tag{11}$$

which is motivated by the rigid pillar problem as presented in [11].

In order to find explicit expressions for the normal forces n_i, the moment balances (5) have to be pre-multiplied by $\mathbf{R}^{-1}(\theta)$. The non-singular matrix $\mathbf{R}(\theta)$ describes a simple rotation about z and is therefore invertible. With (6) this leads to

$$f_1 h - f_3 h + n_2 l - n_4 l = 0 \tag{12}$$

$$f_2 h - f_4 h - n_1 l + n_3 l = 0 \tag{13}$$

The system of equations consisting of (11), (12), (13), and the force balance in the z direction from (4) is solved for n_i in order to yield

$$\begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix} = \frac{1}{4l}\begin{bmatrix} 2h(f_2 - f_4) + lmg \\ 2h(f_3 - f_1) + lmg \\ 2h(-f_2 + f_4) + lmg \\ 2h(-f_3 + f_1) + lmg \end{bmatrix} \tag{14}$$

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

$$f_i = u_i f_{i,max}, \quad u_i \in [-1, 1] \tag{15}$$

Equations (10) and (14) are substituted into (15), 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 (7) and (8):

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \frac{m\mu}{4l}\begin{bmatrix} u_1(-2h(\ddot{x}\cos\theta + \ddot{y}\sin\theta) + lg) \\ u_2(-2h(-\ddot{x}\sin\theta + \ddot{y}\cos\theta) + lg) \\ u_3(2h(\ddot{x}\cos\theta + \ddot{y}\sin\theta) + lg) \\ u_4(2h(-\ddot{x}\sin\theta + \ddot{y}\cos\theta) + lg) \end{bmatrix}, \quad u_i \in [-1, 1] \tag{16}$$

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, (16) is substituted back into (7), (8), and (9). The result is a system of coupled differential equations, where all terms containing normal or friction forces have been eliminated. Solving these equations explicitly for $\ddot{x}$, $\ddot{y}$, and $\ddot{\theta}$ yields

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} = \mathbf{R}(\theta)\frac{lg\mu}{4l^2 + h^2\mu^2(u_2 + u_4)(u_1 + u_3)} \begin{bmatrix} l(u_4 - u_2) + 0.5\mu h(u_1 - u_3)(u_2 + u_4) \\ l(u_1 - u_3) + 0.5\mu h(u_2 - u_4)(u_1 + u_3) \\ m/J(h^2\mu^2 \sum_i \frac{u_1 u_2 u_3 u_4}{u_i} + l^2 \sum_i u_i) \end{bmatrix}, \quad u_i \in [-1, 1] \tag{17}$$

Equation (17) describes the set of admissible accelerations of the vehicle in the global frame of reference. The envelope is plotted and simplified in the next section.

III. Simplifying the Acceleration Envelope

Since (17) 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 I holds the parameters for the omnidirectional vehicle depicted in Fig. 1, which is also used in Section VI 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 acceleration envelope $\mathcal{A}_0$ is not rotationally symmetric. This means that the maximum achievable acceleration is a function of the direction ϕ the vehicle is accelerating in. In order to find a closed form solution, the acceleration envelope is restricted, similar to [1]. By taking the intersection of the accelerations for all directions ϕ, the influence of the orientation is removed.

TABLE I

Sample values for an omnidirectional vehicle

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

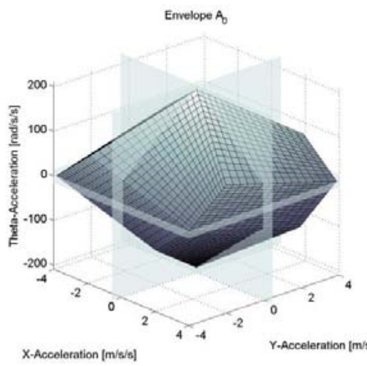

Fig. 3. Acceleration envelope $\mathcal{A}_0$

$$\mathcal{Q} = \bigcap_{\phi\in[0,2\pi]} \mathbf{R}(\phi)\mathcal{A}_0 \tag{18}$$

$\mathcal{Q}$ defines the admissible set of accelerations that the vehicle can achieve, independent of the current orientation θ or heading ϕ. 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. In order to solve the problem in real time with modest computational resources, the problem is relaxed further by decoupling the rotational

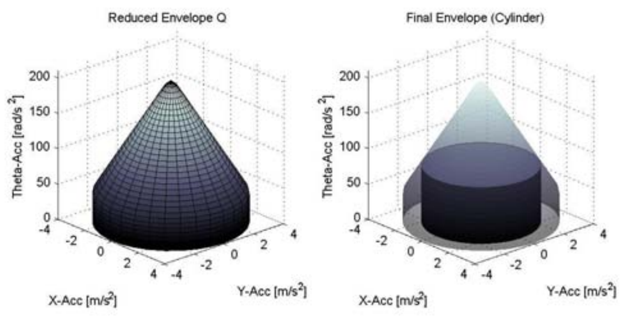

Fig. 4. Reduced envelope $\mathcal{Q}$ (left), final cylindrical envelope (right)

DOF θ from the two linear DOFs. This is achieved by imposing a limit on the rotational acceleration: $|\ddot{\theta}| \leq \ddot{\theta}_{max}$. This reduces the envelope to a cylinder with radius a_{max}, where a_{max} is the maximum linear acceleration at $\ddot{\theta} = \ddot{\theta}_{max}$, see Fig. 4. While the x and y coordinates have to be synchronized in order to get a minimum time solution, this is generally not the case for the rotation. Still, the presented algorithm can easily be extended to not making this simplification.

IV. Solve Optimal Control Problem

From now on the rotation will be neglected, since it is 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 final velocity is always chosen to be zero in order to avoid discontinuities in the solutions when getting close to the desired final state, see also [1] [12]. 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, see [3] [2].

The optimal control problem is to minimize t_f for the system

$$\ddot{x}(t) = q_x(t) \quad (19)$$
$$\ddot{y}(t) = q_y(t) \quad (20)$$

with initial and final conditions

$$x(0) = 0, \quad x(t_f) = x_f, \quad \dot{x}(0) = \dot{x}_0, \quad \dot{x}(t_f) = 0 \quad (21)$$
$$y(0) = 0, \quad y(t_f) = y_f, \quad \dot{y}(0) = \dot{y}_0, \quad \dot{y}(t_f) = 0 \quad (22)$$

subject to constraints on the control effort and the state

$$\sqrt{q_x^2(t) + q_y^2(t)} \leq a_{max} \quad (23)$$
$$\sqrt{\dot{x}^2(t) + \dot{y}^2(t)} \leq v_{max} \quad (24)$$

where $q_x(t)$ and $q_y(t)$ are the control efforts in x and y directions, t_f is the execution time, x_f and y_f are the final positions, $\dot{x}_0$ 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, both DOFs will have to be synchronized. Introducing a general DOF w, the problem can be written as

$$\ddot{w}(t) = q_w(t) \quad (25)$$

with

$$w(0) = 0, \quad w(t_f) = w_f, \quad \dot{w}(0) = \dot{w}_0, \quad \dot{w}(t_f) = 0 \quad (26)$$
$$|\dot{w}(t)| \leq v_{w,max}, \quad |q_w(t)| \leq a_{w,max} \quad (27)$$

The minimum time solution to this problem occurs on the boundary of the velocity/acceleration constraints (see also [12]). This means that at any time the vehicle is following one of three strategies:

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

In order to find a complete solution the problem can be broken down into a combination of several distinct cases. Without loss of generality, assume that $w_f \geq 0$ (the problem can easily be normalized that way). Thus, the possible cases are reduced to the ones shown in Fig. 5. Each case

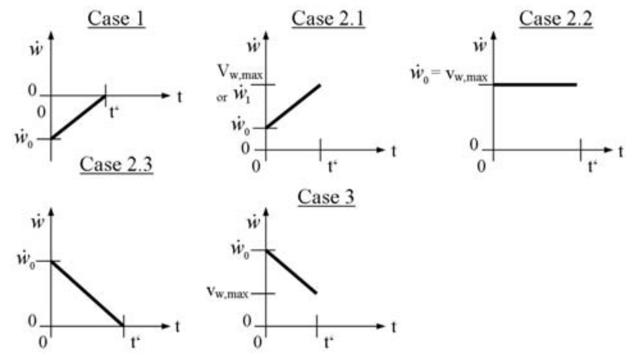

Fig. 5. Possible optimal control cases

represents a possible state or condition the system can be in, depending on $\dot{w}_0$, w_f, $v_{w,max}$, and $a_{w,max}$. These conditions are mutually exclusive, at any given time the system is in one and only one of these cases. Each case has a control effort associated with it. The system has to execute its current case until it either switches into a different case or reaches the final destination with zero final velocity. The complete solution is therefore a sequence of cases. The following list contains the requirements for each particular case, the applied control effort $q_w(t)$ while being in that case, the execution time t', the travelled distance $w' = w(t')$ and the final velocity $\dot{w}' = \dot{w}(t')$ in closed form.

Case 1: $\dot{w}_0 < 0$
The initial velocity is negative, the vehicle has to accelerate with maximal control effort until it reaches $\dot{w}(t) = 0$.

$$q_w(t) = a_{w,max}, \quad t' = -\frac{\dot{w}_0}{a_{w,max}} \quad (28)$$

$$w' = \dot{w}_0 t' + \frac{a_{w,max}}{2} t'^2 = -\frac{\dot{w}_0^2}{a_{w,max}}, \quad \dot{w}' = 0 \quad (29)$$

Case 2.1: $v_{w,max} > \dot{w}_0 \geq 0 \quad \text{AND} \quad w_f > \frac{\dot{w}_0^2}{2a_{w,max}}$
This case has 2 subcases: The vehicle has to accelerate until it reaches either $v_{w,max}$ or $\dot{w}_1$. The variable $\dot{w}_1$ is defined as the velocity at which the vehicle has to decelerate in order to avoid overshooting. 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} to reach $\dot{w}_1$.

$$t_I = \frac{v_{w,max} - \dot{w}_0}{a_{w,max}}, \quad t_{II} = \frac{\dot{w}_1 - \dot{w}_0}{a_{w,max}} \tag{30}$$

with

$$\dot{w}_1 \;=\; \sqrt{w_f a_{w,max} + \frac{\dot{w}_0^2}{2}} \tag{31}$$

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

$$q_w(t) = a_{w,max}, \quad t' = t_I \tag{32}$$

$$w' = \frac{v_{w,max}^2 - \dot{w}_0^2}{2a_{w,max}}, \quad \dot{w}' = v_{w,max} \tag{33}$$

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

$$q_w(t) = a_{w,max}, \quad t' = t_{II} \tag{34}$$

$$w' = w_1 = \frac{w_f}{2} + \frac{\dot{w}_0^2}{2a_{w,max}}, \quad \dot{w}' = \dot{w}_1 \tag{35}$$

Case 2.2: $\dot{w}_0 = v_{w,max} \quad \text{AND} \quad w_f > \frac{\dot{w}_0^2}{2a_{w,max}}$
The vehicle is cruising at maximum velocity until it has to decelerate.

$$q_w(t) = 0, \quad t' = \frac{w_f}{v_{w,max}} - \frac{v_{w,max}}{2a_{w,max}} \tag{36}$$

$$w' = w_f - \frac{v_{w,max}^2}{2a_{w,max}}, \quad \dot{w}' = v_{w,max} \tag{37}$$

Case 2.3: $v_{w,max} \geq \dot{w}_0 > 0 \quad \text{AND} \quad w_f \leq \frac{\dot{w}_0^2}{2a_{w,max}}$
The vehicle has to decelerate until it reaches zero final velocity.

$$q_w(t) = -a_{w,max}, \quad t' = \frac{\dot{w}_0}{a_{w,max}} \tag{38}$$

$$w' = \frac{\dot{w}_0^2}{2a_{w,max}}, \quad \dot{w}' = 0 \tag{39}$$

Case 3: $\dot{w}_0 > v_{w,max}$
The vehicle moves faster than its maximum velocity. This can happen either due to noise or due to the iterative solution procedure presented below. The vehicle has to decelerate until it reaches its maximum velocity.

$$q_w(t) = -a_{w,max}, \quad t' = \frac{\dot{w}_0 - v_{w,max}}{a_{w,max}} \tag{40}$$

$$w' = \frac{1}{2a_{w,max}}(\dot{w}_0^2 - v_{w,max}^2), \quad \dot{w}' = v_{w,max} \tag{41}$$

The procedure to find the complete solution is as follows:

1) Normalize the problem such that $w_f \geq 0$, set $t = 0$
2) Check which case is applicable
3) Execute the case, compute: $q_w(t)$, t', and w'
4) Renormalize: $t = t + t', \quad w_f = w_f - w', \quad \dot{w}_0 = \dot{w}'$
5) If the destination is not reached with zero final velocity: Start over with 2.
6) Total time to destination $t_{f,w} = t$

In general, when calculating trajectories for both x and y the solutions will yield different execution times, $t_{f,x}$ and $t_{f,y}$. 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 a parameter $\alpha \in (0, \pi/2)$:

$$a_{x,max} = a_{max}\cos\alpha, \quad a_{y,max} = a_{max}\sin\alpha \tag{42}$$

$$v_{x,max} = v_{max}\cos\alpha, \quad v_{y,max} = v_{max}\sin\alpha \tag{43}$$

Equations (42) and (43) satisfy the constraints (23) and (24). If either x or y are already at the desired destination with zero velocity, no synchronization is needed. Thus the exclusion of 0 and $\pi/2$ from α. The execution times $t_{f,x}$ and $t_{f,y}$ are continuous and monotonously increasing/decreasing functions of α (for a formal proof, see [14]), therefore it is possible to use a binary search algorithm to find an α which renders the difference between $t_{f,x}$ and $t_{f,y}$ arbitrarily small. With the synchronization of x and y, a feasible, albeit suboptimal, trajectory for the vehicle is found.

V. Performance of the Algorithm in Simulation

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 94×10^{-6} s, with a standard deviation of 28×10^{-6} s.

The second test is to check how good the solution of the proposed algorithm is in terms of 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 t_f. The solution of the proposed algorithm $t_{f,approx}$ is compared against two different benchmarks. Both are computed using RIOTS [13], an optimal control toolbox written in Matlab and C. The Matlab code for the simulation can be found at [14]. 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 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 $t_{f,full}$ of the full problem, i.e. (17) subject to (24). The second benchmark is the execution time $t_{f,2D}$ of the 2D problem (19) to (24). The maximum velocity and acceleration are limited to $v_{max} = 2.0$ m/s and $a_{max} = 3.92$ m/s^2.

A Monte Carlo simulation was performed by generating random initial conditions and computing $t_{f,approx}$, $t_{f,full}$, and $t_{f,2D}$. Fig. 6 shows the simulation results. The abscissa depicts a reference value r_{tf}. The variable n_i is defined as the number of solutions for which $t_{f,full}/t_{f,approx} \geq r_{tf}$ or $t_{f,2D}/t_{f,approx} \geq r_{tf}$. The total number of solutions is n_{tot}. It follows that a large n_i/n_{tot} for large r_{tf} means that the solution of the new algorithm is almost as fast as the RIOTS solutions. As expected, the solution of the

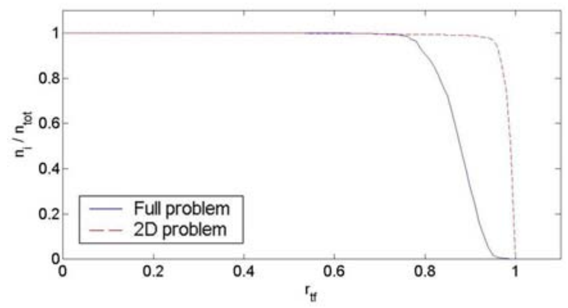

Fig. 6. Comparison of execution times

full problem yielded the smallest execution times, since it used the unreduced acceleration 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 100×10^{-6} s.

VI. Implementation

The new trajectory generation algorithm was implemented on the Cornell RoboCup system. A robot was commanded to move along a line from ($x = -1.0$ m, $y = -0.5$ m) to ($x = 1.0$ m, $y = -0.5$ m), using trajectory generation. The rotation was held fixed at $\theta = 0$ rad. When the robot was crossing a particular x coordinate x_c the final destination was changed to ($x = 0.0$ m, $y = 0.5$ m), 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. Fig. 7 depicts the results, video clips of the vehicle executing the test pattern can be found at [14]. The Figure shows two paths for each x_c. The solid lines stand for the paths actually taken by the robots. The 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. The deviation between the two is due to noise and unmodeled dynamics. The theoretical and actual paths are close, which means that the robot was able to follow the initial theoretical path. This shows the successful implementation of the proposed trajectory generation algorithm on a real vehicle.

VII. Conclusion

A trajectory generation algorithm for omnidirectional vehicles has been presented, which takes the vehicle dynamics, limited friction, and weight transfer into account. It is tailored to high-performance vehicles that are mainly

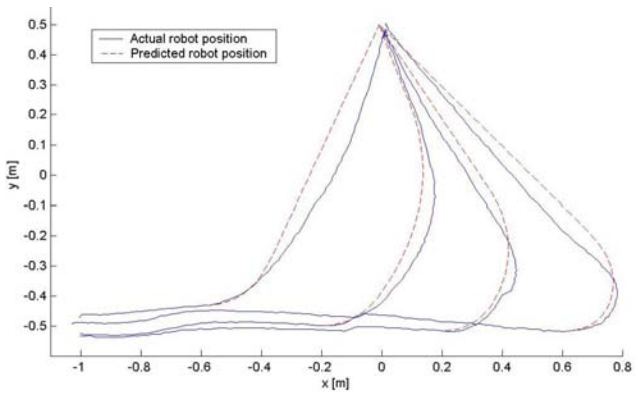

Fig. 7. Implementation on Robot

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.

References

[1] Kalmár-Nagy T., D'Andrea R., Ganguly P.: "Near-Optimal Dynamic Trajectory Generation and Control of an Omnidirectional Vehicle", Robotics and Autonomous Systems, Vol. 46, 2004, pp. 47-64
[2] Purwin O., D'Andrea R.: "Cornell Big Red 2003", in: Polani D., Bonarini A., Browning B., Yoshida K. (Eds), Robocup 2003: Robot Soccer World Cup VII, Lecture Notes in Artificial Intelligence, Springer, Berlin, 2003
[3] D'Andrea R., Kalmár-Nagy T., Ganguly P., Babish M.: "The Cornell RoboCup Team", in: Stone P., Balch T., Kraetzschmar (Eds), Robocup 2000: Robot Soccer World Cup IV, Springer, Berlin, 2001
[4] Muñoz V., Ollero A., Prado M., Simón A.: "Mobile Robot Trajectory Planning with Dynamic and Kinematic Constraints", Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 4, 1994, pp. 2802-2807
[5] Faiz N., Agrawal S.K.: "Trajectory Planning of Robots with Dynamics and Inequalities", Proceedings of the 2000 IEEE International Conference on Robotics and Automation, Vol. 4, 2000, pp. 3976-3982
[6] Moore K.L., Flann N.S.: "A Six-Wheeled Omnidirectional Autonomous Mobile Robot", Control Systems Magazine, IEEE, Vol. 20, Issue 6, Dec 2000, pp. 53-66
[7] Watanabe K., Shiraishi Y., Tzafestas S.G., Tang J., Fukuda T.: "Feedback Control of an Omnidirectional Autonomous Platform for Mobile Service Robots", Journal of Intelligent and Robotic Systems, Vol. 22, Issue 3-4, 1998, pp. 315-330
[8] Liu Y., Wu X., Zhu J., Lew J.: "Omni-directional mobile robot controller design by trajectory linearization", Proceedings of the American Control Conference 2003, Vol. 4, No. 4-6, June 2003, pp. 3423-3428
[9] Frazzoli E., Dahleh M.A., Feron E.: "Real-Time Motion Planning for Agile Autonomous Vehicles", Journal of Guidance, Control, and Dynamics, Vol. 25, No. 1, January 2002, pp. 116-129(14)
[10] Olsson H., Aström K.J., Canudas de Wit C., Gäfvert M., Lischinsky P.: "Friction Models and Friction Compensation"
[11] Bender C.B., Brody D.C., Meister B.K.: "Quantised Three Pillar Problem", submitted to Journal of Physics A, 2002
[12] Bryson A.E., Ho Y.-C.: "Applied Optimal Control", John Wiley & Sons, 1975
[13] Schwartz A., Polak E., Chen Y.: "RIOTS 95", Optimal Control Toolbox for Matlab V6.5
[14] http://control.mae.cornell.edu/Purwin/PhDWork.htm