Cornell Big Red 2003

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 the major improvements of the Cornell RoboCup team for the 2003 competition. A version of trajectory generation for omnidirectional vehicles is presented, which takes limited friction and weight transfer into account. The second part covers the implementation and testing of inertial sensors, an augmented motor control algorithm, and a description of the new wheel design.

1 Introduction

The world championship in Padua, Italy, was the fifth time that the Cornell RoboCup team competed in the small-size robotics soccer league. Every year the hardware is built from scratch, allowing Cornell to make major changes to the design and implement new technology.

The focus of this paper is on the major improvements to our system, mainly the trajectory generation and the local control. We feel that these changes will greatly benefit all participating teams in future competitions, irrespective of the particular approach used for the other subsystems such as computer vision and strategy.

The goal for 2003 was to play a very fast but controlled game. Cornell's robots from 2002 were among the fastest at the competition in Japan, but at high velocities they lacked control. This made precise passes and shots on goal quite difficult. This problem was tackled on all levels of the system. One way to achieve this objective was to reduce system latency by optimizing the vision system. The TCP/IP network was replaced with a serial connection of consistent and lower latency. The wheel design was changed to give better traction and therefore more stability at high velocities. Inertial sensors were put on-board the robots in order to be able to quickly correct for deviations from the optimal trajectory. Trajectory generation itself was re-written with the properties of the new robot hardware in mind. These improvements complemented each other and made it possible to defend our title in Italy.

2 Overview Robot Control

The robot control is organized in a hierarchical structure, see also [2]. On the highest level, the so called "AI" makes the strategic decisions where to place the

robots, based on the current status of the system and the situation on the field. A certain level of randomness is introduced in the decision making processes to prevent opponents from focusing in on, and exploiting, a particular weakness of our strategy. The desired destinations for each robot are passed on to the control module, which implements obstacle avoidance and generates robot paths.

Obstacle avoidance adjusts a robot's destination until the path from the initial to the final position is obstacle free. During that process it calls trajectory generation, which in turn computes the robot paths. When an obstacle free path is found, the corresponding set of velocity commands is sent to the robot. The on-board control hardware and firmware tries to track the given trajectory as closely as possible.

The advantage of this system is that strategy is decoupled from the hardware. As long as the robots are capable of moving and shooting the ball, they can execute the commands from the strategy module. In order to make better strategic decisions, AI can query the control module whether a certain path is possible or what the time to destination would be.

The following sections will explain the changes to trajectory generation and local control in more detail.

3 Trajectory Generation

High level robot control is done by solving a relaxed optimal control problem. Given the robot's current state (position and velocity) and the desired destination, trajectory generation calculates the minimum time path to get to the destination with zero final velocity. This primitive is being called by several parts of the AI and allows for a very modular approach.

The previous version of trajectory generation was written in 2000 for the first generation of omnidirectional robots, see [1]. Back then, robots were a lot slower and the main bottleneck were power limitations, meaning that the robots experienced reduced acceleration at higher speeds. The trajectory generation was developed with these motor characteristics in mind.

Over the years, the robot hardware underwent significant changes. The robots became much faster and more agile. The use of four drive motors instead of three meant that more power was available to accelerate the robots. From 2002 on, Cornell's robots were friction limited, not power limited. In the lower velocity ranges (< 1 m/s) they could spin out their wheels at any time.

With the increase in acceleration, another effect became more significant: weight transfer. This means that during an acceleration phase the weight distribution on the wheels changes due to the inertia of the robot mass. When accelerating in the forward direction, the normal force of the front wheels is reduced while the at the same time the rear wheels are loaded more heavily.

A new trajectory generation was developed which takes into account these two major effects, friction limitation and weight transfer. This section explains the ideas behind the new trajectory generation and how the algorithm works.

3.1 Derive Acceleration Envelope

The first assumption is that within the used velocity range the motors can provide infinite torque. The acceleration of the robot is then limited by the maximum friction force between the wheels and the ground. Assuming Coulomb friction, the maximum friction force $f_{i,\max}$ for each wheel is directly proportional to the coefficient of friction μ and the normal force n_i between wheel and ground. The normal force n_i is a function of the mass of the robot and the acceleration, due to the weight transfer mentioned above. Note that a detailed model of friction will be much more complicated than Coulomb friction, due to the omni-directional wheels; the assumption of Coulomb friction should be seen as a first order approximation.

The robot is modeled as a rigid body, with the center of mass (CM) directly above the geometric center. The robot has four drive modules, which are arranged with a 90 ° displacement relative to each other, see Figure 1. The global

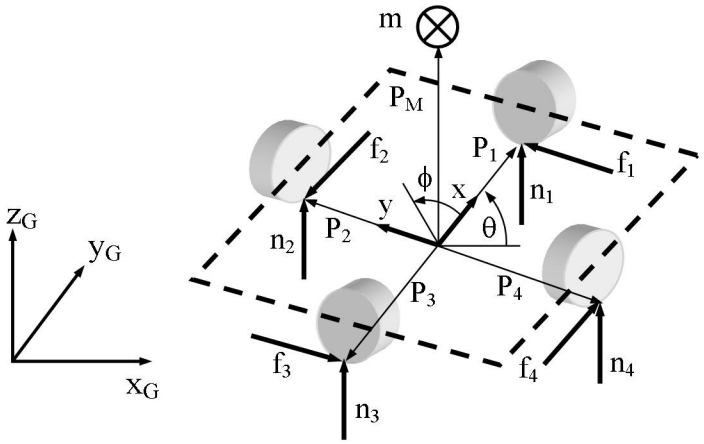

Fig. 1. Free body diagram of a robot

coordinate system is defined by x_{G}, y_{G}, and z_{G}. The robot frame of reference is defined by x and y. The angle θ is the rotation of the local coordinates with respect to the global ones, ϕ is the direction the robot is accelerating in. The mass of the robot is m, the forces f_i are the friction forces which accelerate the robot. The positions of the wheels with respect to the CM are defined by the vectors $\boldsymbol{P}_i$.

Taking force and moment balances about the robot's center of mass yields the equations of motion of the vehicle, see [4]. Inserting the relations for the maximum friction forces and solving the resulting system of equations yields the

accelerations in the robot's frame of reference

$$\ddot{x} = \frac{-\mu l g(\mu h[(u_3 - u_1)(u_4 + u_2)] + 2l(u_2 - u_4))}{8l^2 + 2\mu^2 h^2 (u_3 + u_1)(u_4 + u_2)} \tag{1}$$

$$\ddot{y} = \frac{-\mu l g(\mu h[(u_3 + u_1)(u_4 - u_2)] + 2l(u_3 - u_1))}{8l^2 + 2\mu^2 h^2 (u_3 + u_1)(u_4 + u_2)} \tag{2}$$

$$\ddot{\theta} = \frac{\mu m l g(\mu^2 h^2 [u_1 u_2 u_3 + u_1 u_2 u_4 + u_1 u_3 u_4 + u_2 u_3 u_4] + l^2 \sum_i u_i)}{J(4l^2 + \mu^2 h^2 (u_3 + u_1)(u_4 + u_2))} \tag{3}$$

where h is the height of the CM above ground, l the distance of the wheels to the center of the robot and J the moment of inertia of the robot. The control efforts u_i are defined as

$$u_i = \frac{f_i}{f_{i,\max}}, \quad u_i \in [-1, 1] \tag{4}$$

Equations (1), (2), and (3) are solved numerically to find the acceleration envelope, which is defined as the set of all feasible combinations of $\ddot{x}$, $\ddot{y}$, and $\ddot{\theta}$. The result is depicted in Figure 2. Since the envelope is symmetric, only the upper half ($\ddot{\theta} > 0$) is shown. All combinations of accelerations inside the

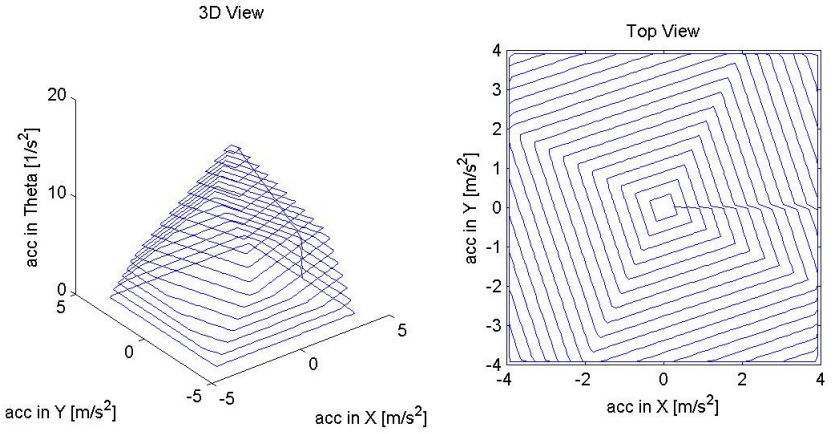

Fig. 2. Acceleration envelope

envelope can be executed by the robot, it is just a matter of choosing the proper combination of control efforts u_i.

Trajectory generation is run in a global frame of reference. Since the derived envelope is valid in the robot's coordinate system, the global acceleration is a function of the orientation θ of the robot. Further, the linear acceleration is also

a function of the direction ϕ the robot is accelerating in. These dependencies on the robot's orientation and course make the control problem more difficult to solve. In order to find a computationally cheap algorithm, the problem is relaxed by limiting the envelope in two steps. The first step is to take the subset of accelerations that is common to all ϕ, see also [1]. The second step is to limit $\ddot{\theta}$ to a value that corresponds to the operational angular accelerations. This reduces the envelope for $\ddot{x}$ and $\ddot{y}$ to a cylinder, see Figure 3. These relaxations

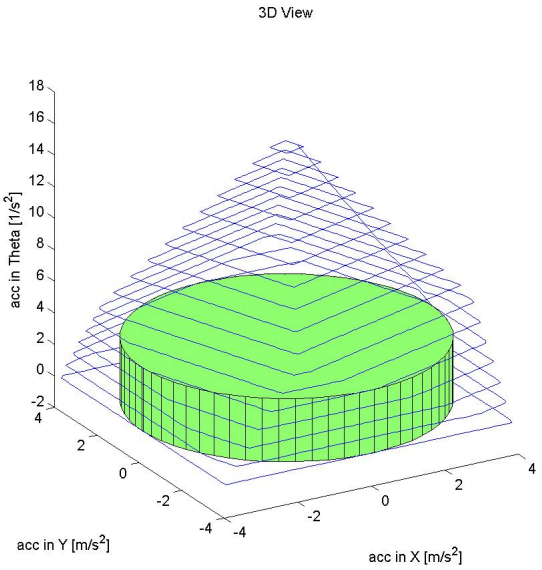

Fig. 3. Restricting $\ddot{x}$ and $\ddot{y}$ to a cylinder

also decouple $\ddot{\theta}$ from $\ddot{x}$ and $\ddot{y}$. In the following, an algorithm is proposed to find a trajectory for the two linear degrees of freedom (DOF), x and y. Calculating a path for θ is a special, simplified case that is not covered here.

3.2 Solve Optimal Control Problem

The optimal control problem is to find a minimum time trajectory for the following system:

$$\ddot{x} = q_{\mathrm{x}}(t) \tag{5}$$

$$\ddot{y} = q_{\mathrm{y}}(t) \tag{6}$$

with boundary and initial conditions

$$x(0) = 0, \quad x(t_{\mathrm{f}}) = x_{\mathrm{f}}, \quad \dot{x}(0) = \dot{x}_0, \quad \dot{x}(t_{\mathrm{f}}) = 0 \tag{7}$$

$$y(0) = 0, \quad y(t_\mathrm{f}) = y_\mathrm{f}, \quad \dot{y}(0) = \dot{y}_0, \quad \dot{y}(t_\mathrm{f}) = 0 \tag{8}$$

subject to constraints on the control effort and the state itself

$$\sqrt{q_\mathrm{x}^2(t) + q_\mathrm{y}^2(t)} \leq a_\mathrm{max} \tag{9}$$

$$\sqrt{\dot{x}^2 + \dot{y}^2} \leq v_\mathrm{max} \tag{10}$$

where q_x and q_y are the control efforts in x and y directions, t_f is the minimum time, a_max is the maximum linear acceleration, and v_max is the maximum linear velocity. Note that the speed of the robot is limited. This is due to the fact that the motors are designed for a certain maximum number of revolutions per minute. Going beyond that speed means risking damage to the motors. Also, at some point the assumption of unlimited motor torque breaks down.

The maximum speed and acceleration of the two DOFs are coupled. Since the objective is to reach the desired state in minimum time, the solution occurs on the boundary of the constraints, (9) and (10), see also [3]. The following has to hold true for the control efforts:

$$\sqrt{q_\mathrm{x}^2(t) + q_\mathrm{y}^2(t)} = a_\mathrm{max} \quad \text{if} \quad \sqrt{\dot{x}^2 + \dot{y}^2} < v_\mathrm{max} \tag{11}$$

$$\sqrt{q_\mathrm{x}^2(t) + q_\mathrm{y}^2(t)} = 0 \quad \text{if} \quad \sqrt{\dot{x}^2 + \dot{y}^2} = v_\mathrm{max} \tag{12}$$

The problem is further simplified by coupling maximum acceleration and top speed as follows

$$q_\mathrm{x,max} = a_\mathrm{max} \cos\alpha \tag{13}$$

$$q_\mathrm{y,max} = a_\mathrm{max} \sin\alpha \tag{14}$$

$$\dot{x}_\mathrm{max} = v_\mathrm{max} \cos\alpha \tag{15}$$

$$\dot{y}_\mathrm{max} = v_\mathrm{max} \sin\alpha \tag{16}$$

$$\alpha \in [0, \pi] \tag{17}$$

where the role of α will be explained shortly. This limitation ensures that the constraints on the state and control effort are met. At the same time, it decouples the two linear degrees of freedom, x and y, which can now be solved separately.

In general, the solutions for x and y directions have different minimum times to reach the final state. In order to find a minimum time trajectory for both DOFs, the solutions have to be synchronized. This can be achieved by varying α.

The next step is to find the minimum time solution to a single DOF z. At this point it is possible to distinguish five possible cases, see Figure 4. At any given time, the problem falls into one of these cases, which are defined by the initial conditions of the state vector (v_0 is the initial velocity) and the desired final state. The solution is found by stepping through all applicable cases until the desired state (desired position with zero final velocity) is reached.

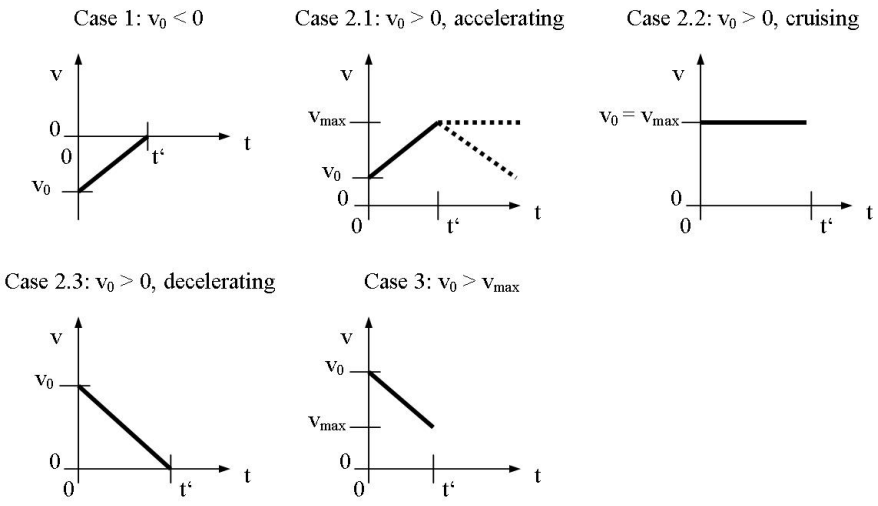

Fig. 4. Possible optimal control cases

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

Each case yields a required execution time t' and acceleration a'. Knowing the initial position and velocity v_0 of the vehicle, the trajectory can be easily calculated by integrating the acceleration a'.

After doing this for both the x and y directions, the solutions have to be synchronized by adjusting the parameter α. The total execution times $t_{\mathrm{f},i}$ for both directions are monotone and continuous functions of α. Thus, a bisection algorithm can be used to adjust α, such that $t_{\mathrm{f,x}} = t_{\mathrm{f,y}}$. Once the proper control efforts for x and y are found, the entire trajectory can be generated.

4 Local Sensing

The steady increase in robot acceleration and top speed makes it increasingly harder to control the robots. A good example of this difficulty is Cornell's 2002 system. While the robots were able to accelerate at about 4 m/s^2 and move at 1.8 m/s, there were some problems regarding the stability of the motion. At high speeds the robots weren't following the desired orientation very well, making precise passes or shots at the goal very difficult.

One of the reasons for the severity of this problem is system latency. It takes too long to detect a deviation over the vision system and correct for it. By that

time, the robots are already too far out of position. In a situation where the robots are supposed to kick the ball, this causes the shot to miss.

The encoders of the drive motors cannot detect this problem since at high speeds the robots are skidding. Each wheel by itself is well controlled while the robot as a whole is out of control. In order to improve local control we decided to put rate gyroscopes on our robots. The measurement range of these rate gyros is 10 rad/s at 300 Hz bandwidth. The rate gyros are temperature compensated and have a negligible drift after the initial calibration, i.e. after finding the 0 rad/s offset. Figure 5 shows the block diagram of the control loops on the robot.

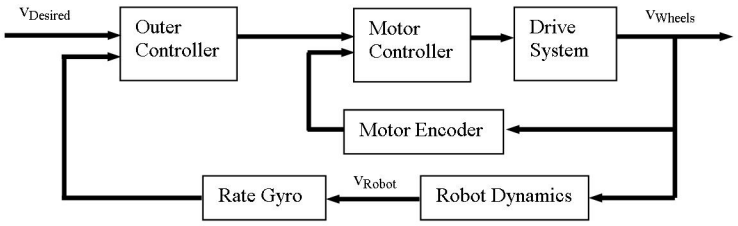

Fig. 5. Local control loops

The signal v_{Desired} contains the desired velocities for all DOFs, v_{Wheels} contains the four wheel velocities, and v_{Robot} contains the robot velocities.

The inner loop contains the motor control and the sensor readouts from the encoders. In previous years the desired velocities were sent directly to this loop, the motor control was then responsible for turning the wheels at the right velocities and making the robot move.

In order to integrate the rate gyros into the system, a second control loop was wrapped around the motor control. The outer loop takes the commands from AI as inputs, compares them with the measured robot velocities and adjusts the input to the motor control accordingly. Therefore the robots can compensate for slipping wheels in order to stay on the commanded trajectory.

In order to quantify the effect of local sensing, robots were tested with and without rate gyros and the results compared. The first test was an open loop pattern, designed to check the drive system before every game. The robot moves forward, executes a circle and comes back exactly the same way. Ideally, the robot stops at the same location from where it started. Figure 6 shows the results of the test, the trajectory of the robot in x and y coordinates. It can be seen that local sensing helps the robot to follow the commanded trajectory. Without the rate gyro, the radius of the circle becomes too small and the robot overshoots. When backtracking, the robot doesn't make it back to the point from where it started.

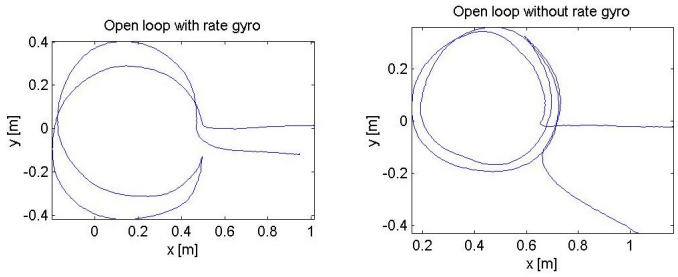

Fig. 6. Results of open loop test mode

While this is some indication that local sensing helps executing motion commands, in a real game situation the robot's motion is constantly monitored and corrected. In order to test the system under these conditions, the robots were run closed loop (i.e. with vision) in a rectangular test pattern. The robots had to move from corner to corner of a rectangle while doing a 180 ° turn on each leg. The position of the robots as seen by vision was recorded, Figure 7 shows the results from the experiment. The robot without the rate gyro has larger variance

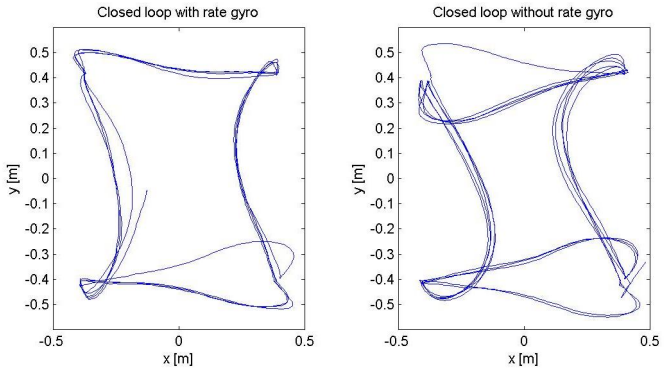

Fig. 7. Results of closed loop rectangle test

in its trajectories. Also, the curves are more noticeable, meaning that it couldn't follow the rectangular trajectory as well as the robot with the rate gyro.

These graphs give a first impression of the effectiveness of the rate gyro. A more objective metric is a comparison between the predicted and the actual robot path. Prediction uses recently sent velocity commands to estimate the

robots position in the future. This is done to compensate for latency in the system. If the robot follows the commanded trajectory perfectly, predicted and actual positions will match up.

The metric $\boldsymbol{E}$ for how good the robot followed the trajectory was defined as

$$\boldsymbol{E} = \frac{1}{N} \sum_{k=1}^{N} |\boldsymbol{P}_{\text{act}}[k+L] - \boldsymbol{P}_{\text{pred}}[k]| \tag{18}$$

where N is the number of sample points, $\boldsymbol{P}_{\text{act}}$ is the vector of the actual robot position, $\boldsymbol{P}_{\text{pred}}$ is the vector of the predicted position, and L is the system latency in frames. Note that in order to be in the same time frame, the actual positions have to be shifted by L. The averaged results of several tests are shown in Table 1.

Table 1. Trajectory errors of rate gyro test

Type	Error in x [m]	Error in y [m]	Error in θ [rad]
no rate gyro	0.024	0.026	0.059
rate gyro	0.019	0.022	0.050

This shows that using a rate gyro reduces the tracking errors of the robots by about 20 %. It should be noted, that the improvements are not limited to the coordinate of the sensor θ, but also extend to the linear DOFs, x and y.

5 Motor Torque Control

In previous years, voltage control was used for the local motor control loops. The controller itself was a modified PI controller, whose output was the duty cycle of a pulse-width modulated (PWM) signal. This signal triggered the power electronics for the motors.

The latest generation of robots, however, uses torque control, which has several advantages. First of all, it is possible to limit the torque individually for each wheel, preventing the wheels from slipping. The torque applied to a motor is directly proportional to the current, which generates heat. Excessive heat can easily destroy the brushes of a motor. Since we were operating our motors at the upper limits of the specifications, we wanted to control how much current was going through the motor at any given time. In the rest of this section, a simple way of implementing torque control is presented.

The equation of motion of the system, consisting of one motor with an attached rotary inertia, can be written as

$$J\dot{\omega} = T \tag{19}$$

where J is the moment of inertia, ω is the wheel velocity in rad/sec and T is the applied motor torque. The control law takes the error e in motor speed and produces a desired torque T_{des}.

$$e = \omega_{\text{des}} - \omega \tag{20}$$

$$T_{\text{des}} = k_{\text{p}} e + k_{\text{i}} \int_0^t e d\tau \tag{21}$$

The instantaneous torque T_{inst} that is available at the motor is a function of the applied voltage u, the motor speed ω and the motor coefficients ψ and ν given by the manufacturer.

$$T_{\text{inst}} = \nu u - \psi \omega \tag{22}$$

The PWM duty cycle f is the ratio of "motor on" to "motor off" times. Whenever the signal is high, the motor produces the torque T_{inst}. Whenever it is low, the motor is floating and producing no torque at all. The effective torque T_{eff} is computed by taking the average torque over one period of the PWM signal, i.e. it is just the fraction of the "on" times:

$$T_{\text{eff}} = f T_{\text{inst}} \tag{23}$$

The motor will produce the desired amount of torque when T_{eff} matches T_{des}. This can be achieved by adjusting f. Substituting (21) and (22) into (23) leads to the following expression:

$$f = \frac{k_{\text{p}} e + k_{\text{i}} \int_0^t e d\tau}{\nu u - \psi \omega} \tag{24}$$

This duty cycle f is then simply sent to the power electronics to drive the motor.

6 Wheel Design

The problem of maintaining control at high speeds was approached from different angles. Additional local sensing and new control algorithms went hand in hand with improved hardware. Building on the experience from the 2002 competition the major objectives for the mechanical design team were to increase controllability, acceleration, and top speed.

The wheels used in the 2001/2002 competitions are easy to implement but they don't give enough traction to sustain a controlled robot motion at high velocities. Further, they are rather wide, using up much of the single most limited resource on the robots, space.

The solution to these problems was the design of a new wheel. Requirements were a high coefficient of friction with the ground and little space consumption. First tests were done using aluminum rollers, which really dig into the carpet

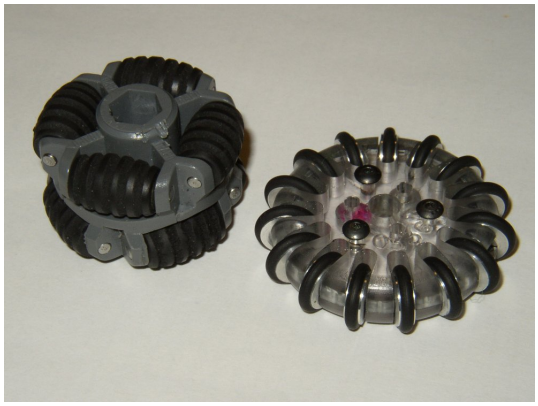

Fig. 8. 2002 wheel (left), 2003 wheel (right)

and yield excellent traction. The drawback is excessive wear on the carpet, which was considered unsportsmanlike and unacceptable.

The final solution is a wheel with a single row of in-line rollers, see Figure 8. The rollers at the perimeter of the wheel are rubber coated for higher friction. The rollers are thin enough to dig into the carpet and yield much higher traction than the previous wheel designs. At the same time, the thickness of the wheels is reduced by about 50 % in comparison to the wheels from 2002, allowing more room for the motors and the solenoid kick.

7 Acknowledgments

The authors would like to thank the members of the 2003 Cornell RoboCup team, especially the ones who went to the competition: Eugene Byrne, Patrick Dingle, Leonard Evansic, Joseph Loomis, Sergei Lupashin, Gregory Peng, Yuval Shavit, Carlo Soracco, and Kenneth Sterk. Our success in Italy would not have been possible without their dedication and efforts.

References

1. Kalmàr-Nagy, T., Ganguly, P., D'Andrea, R.: "Near-Optimal Dynamic Trajectory Generation and Control of an Omnidirectional Vehicle", submitted to International Journal of Robotics Research
2. D'Andrea, R., Kalmàr-Nagy, T., Ganguly, P., Babish, M.: "The Cornell RoboCup Team", Stone P., Balch T., Kraetzschmar (Eds), Robocup 2000: Robot Soccer World Cup IV, Springer Verlag, Berlin, 2001
3. Bryson, Arthur E., Ho, Y.-C.: "Applied Optimal Control", John Wiley & Sons, 1975
4. Moon, Francis C.: "Applied Dynamics", Wiley-Interscience, 1998