Holonomic Control of a robot with an omni-directional drive.

Raul Rojas and Alexander Gloye Förster

This paper shows how to control a robot with omnidirectional wheels, using as example robots with four motors, and generalizing to n motors. More than three wheels provide redundancy: many combinations of motors speeds can provide the same Euclidean movement. Since the system is over-determined, we show how to compute a set of consistent and optimal motor forces and speeds using the pseudoinverse of coupling matrices. This approach allows us also to perform a consistency check to determine whether a wheel is slipping on the floor or not. We show that it is possible to avoid wheel slippage by driving the robot with a motor torque under a certain threshold or handle it and make high accelerations possible.

1 Introduction

Omnidirectional wheels have become popular for mobile robots, because they allow them to drive on a straight path from a given location on the floor to another without having to rotate first. Moreover, translational movement along any desired path can be combined with a rotation, so that the robot arrives to its destination at the correct angle.

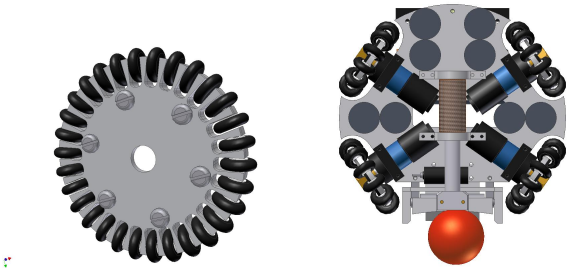

Figure 1: Our omnidirectional wheel design (left) and four-wheeled robot (right).

Omnidirectional wheels mostly based on the same general principle: while the wheel proper provides traction in the direction normal to the motor axis, the wheel can slide frictionless in the motor axis direction. In order to achieve this, the wheel is built using smaller wheels attached along the periphery of the main wheel. Fig. 1 (left) shows an example of the kind of wheels that we have been using for our RoboCup small size and middle size omnidirectional robots since 2002 [3, 9]. Our wheel is a variation of the so-called Swedish wheels, which use rollers with a rotation direction which is not parallel nor perpendicular to the motor axis.

Two or more omnidirectional wheels are used to drive a robot. Each wheel provides traction in the direction normal to the motor axis and parallel to the floor. The forces add up and provide a translational and a rotational motion for the robot. If it were possible to mount two orthogonally oriented omnidirectional wheels right under the center of a robot with a circular base, then driving the robot in any desired direction (without rotation) would be trivial. To give the robot a speed (v_x, v_y), with respect to a cartesian coordinate system attached to the robot, each wheel would just have to provide one of the two speed components.

However, since the wheels and motors need some space, this simple arrangement is not possible (the robot would be also very unstable!). The wheels are usually mounted on the periphery of the chassis. More than two wheels can be used, which makes it also easier to cancel any rotational torque which could make difficult to drive the robot on a straight path. Popular configurations are three and four-wheeled omnidirectional robots. Fig. 1 (right) shows the CAD design of the omnidirectional robot which we used at RoboCup 2004 in Lisbon [3].

In the next sections we present first the physical fundamentals we need to control robots with omnidirectional drive. We present then the energy saving and non slipping control of a 4 wheeled symmetrical robot in ideal circumstances. Unfortunately, we have unpredictable wheel slippage and not fully definable physical robot characteristics. Our goal is to drive fast and precise which is impossible under the described conditions. Next we present our approach for handling wheel slippage under high velocities and accelerations. We correct its impacts by a system control loop allowing precise and fast movement at the same time. In Chapter 5 we present some related work.

2 Physical Fundamentals

Each wheel can move the robot forward, but since they are located on the periphery of the robot, they can also rotate the robot's frame. In order to derive the relationship between the motors' torques and the movement of the robot, we need to analyze the geometry of the problem.

Let us use a robot with $n \geq 3$ wheels, as shown in the diagram (Fig. 2).

All the angles of the motor axis are measured relative to the x direction in the coordinate system of the robot. Call the angles of the motor axis for the n wheels $\theta_1, \theta_2, \ldots, \theta_n$. The driving direction of the i-th wheel is therefore $\theta_i + \pi/2$.

When the n motors are activated, we obtain n traction forces $F_1, F_2, \ldots, F_n$ from the motors, which add up to a translational force and a rotational torque. Each traction force F_i is the torque of the motor multiplied by the radius of the wheel. The sum of the forces depends on the exact wheel arrangement.

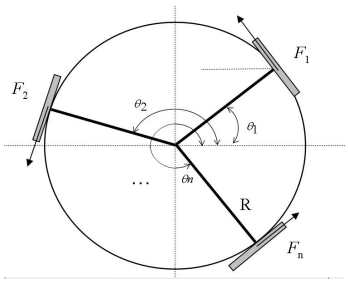

Figure 2: Arrangement of n wheels and distribution of forces.

2.1 Force Coupling Matrix

We are interested in the movement of the robot along the x and y direction. In order to simplify the expressions we will derive, we consider the instantaneous acceleration and velocity of the robot with respect to its own reference frame. For example, a robot moving forward will have a certain positive velocity in the y direction and zero in the x direction. We call the translational velocity and the angular velocity of the robot the "Euclidean magnitudes", different from the individual motor speeds and accelerations.

The translational acceleration a and the angular acceleration $\dot{\omega}$ of the center of mass of the robot (which we assume is located at the geometrical center of our circular robot), are given by

$$a = \frac{1}{M}(F_1 + F_2 + \ldots + F_n)$$

$$\dot{\omega} = \frac{R}{I}(f_1 + f_2 + \ldots + f_n)$$

where M is the mass and R is the radius of the robot, f_i denotes the magnitude of the force F_i, for $i = 1, \ldots, n$, and I is the moment of inertia. The computation is possible using this expression, because the forces are tangent to the circular frame of the robot and point in the same rotational direction, so that we can work just with the magnitudes of the force vectors. The magnitudes $f_1, f_2, \ldots, f_n$ can be positive or negative, according to the direction of rotation of the motor (counterclockwise or clockwise). The positive rotation directions are as shown in Fig. 2.

We can compute the x and y components of the robot's acceleration, by considering the respective components of each force:

$$Ma_x = -f_1 \sin\theta_1 - f_2 \sin\theta_2 - \ldots - f_n \sin\theta_n$$

$$Ma_y = f_1 \cos\theta_1 + f_2 \cos\theta_2 + \ldots + f_n \cos\theta_n$$

For a homogeneous cylinder $I = \frac{1}{2}MR^2$, for a ring $I = MR^2$. For any mass distribution strictly between a concentration of mass in the middle and concentration in the periphery, $I = \alpha MR^2$, with $0 \leq \alpha \leq 1$. We can express the above acceleration equations as a matrix vector multiplication

$$\begin{pmatrix} a_x \\ a_y \\ \dot{\omega} \end{pmatrix} = \frac{1}{M} \begin{pmatrix} -\sin\theta_1 & -\sin\theta_2 & \cdots & -\sin\theta_n \\ \cos\theta_1 & -\cos\theta_2 & \cdots & \cos\theta_n \\ \frac{MR}{I} & \frac{MR}{I} & \cdots & \frac{MR}{I} \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$$

We can use $I = \alpha MR^2$ and simplify this matrix further by using the same units (meters per second) for the planar and angular acceleration. Instead of operating with $\dot{\omega}$ we can work with $R\dot{\omega}$. The new expression is then

$$\begin{pmatrix} a_x \\ a_y \\ R\dot{\omega} \end{pmatrix} = \frac{1}{M} \begin{pmatrix} -\sin\theta_1 & -\sin\theta_2 & \cdots & -\sin\theta_n \\ \cos\theta_1 & \cos\theta_2 & \cdots & \cos\theta_n \\ \frac{1}{\alpha} & \frac{1}{\alpha} & \cdots & \frac{1}{\alpha} \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$$

We call the $3 \times n$ matrix in the expression above the force coupling matrix C_α.

Given any four motor states (and the associated torques) it is then straightforward to compute the acceleration in the x and y directions, as well as the tangential acceleration of the robot's frame periphery.

We are assuming here that the wheels cannot slip, that is, all the torque from the motors is transmitted to the robot via the floor. This is an unrealistic assumption which we discuss later.

2.2 Euclidean magnitudes

We can compute the final velocities of the wheels, and the velocity of the robot on the plane, as well as its angular velocity, by integrating the movement equations with respect to time. However, we have to think of the robot in Euclidean space, compute its trajectory there, and derive from this the velocity of each individual wheel. First let us look at the geometry of the problem.

Let us group the individual speeds of the four motors in the vector $(v_1, v_2, \ldots, v_n)^{\mathrm{T}}$ and the Euclidean velocity and tangential rotational speed of the robot in the vector $(v_x, v_y, R\omega)^{\mathrm{T}}$. If the robot is moving as determined by the vector $(1, 0, 0)^{\mathrm{T}}$, this means that it is moving sideways without rotating. When the robot moves with speed 1 to the right, the i-th wheels rotate with speed $-\sin\theta_i$. This is easy to see from the diagram in Fig. 3. The large wheel provides one of the components of the horizontal movement (that is, $-\sin\theta_i$), while the small peripheral wheels provide the other orthogonal component (i.e. $\cos\theta_i$).

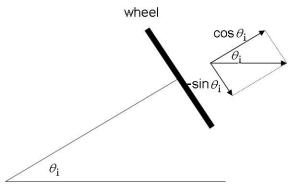

Figure 3: Rotation of large and small wheels, when the robot moves sideways with speed 1. The main wheel rotates with speed $-\sin\theta_i$, the small wheels with speed $\cos\theta_i$.

The same kind of computation can be done when the robot is moving forward without rotating. The wheel movement is then the component $\cos\theta_i$. Using the convention that the positive rotation direction is the direction of the right-hand thumb when we hold the motor axis in the hand, we obtain the following

expression for the correspondences between the Euclidean and motor speeds:

$$\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} -\sin\theta_1 & \cos\theta_1 & 1 \\ -\sin\theta_2 & \cos\theta_2 & 1 \\ \vdots & \vdots & \vdots \\ -\sin\theta_n & \cos\theta_n & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ R\omega \end{pmatrix}$$

The matrix in this expression is very similar to the transpose of the coupling matrix C_α. This matrix, which we denote by D, is the velocity coupling matrix. If the rank of the coupling matrix is at least three, then for any given Euclidean combination $(v_x, v_y, \omega R)^{\mathrm{T}}$, there is a combination of motor speeds which can produce such movement.

Let us denote the acceleration vector $(a_x, a_y, R\dot{\omega})^{\mathrm{T}}$ by a, the force vector $(f_1, f_2, \ldots, f_n)^{\mathrm{T}}$ by f, the vector $(v_x, v_y, R\omega)^{\mathrm{T}}$ by v, and the vector of motors speeds $(v_1, v_2, \ldots, v_n)^{\mathrm{T}}$ by m. Then the following identities hold

$$a = C_\alpha f \quad \text{and} \quad m = Dv$$

Integrating over a time interval Δt, we have $\Delta v = \Delta t \times a$, that is,

$$\Delta m = \Delta t \times D C_\alpha f.$$

The motors have tick counters which allow to measure their speed in real time with the on-board electronics. For the purpose of controlling the robot, we want to know how measured wheel speeds $m = (v_1, v_2, \ldots, v_n)^{\mathrm{T}}$ map to the Euclidean magnitudes $v = (v_x, v_y, R\omega)^{\mathrm{T}}$, that is, we would like to invert the expression $m = Dv$. This is not possible in general because the matrix D is not a square one, and therefore, is not invertible. However, we can look for the pseudoinverse matrix D^+ and with $m = Dv$ the equation

$$D^+ m = (D^+ D)v = I_3 v = v$$

is valid. It can be easily shown that the matrix has always a rank of 3. It can also be proven that when the rank of an $n \times 3$ matrix D is three, the product D^+D is the 3×3 identity matrix I_3.

The matrix D^+, therefore, maps motor speeds to Euclidean velocities correctly. This is a formal proof of what we have been arguing in this paper, that three omnidirectional wheels, at three different angles, are sufficient for omnidirectional driving. The rest of the wheels $(n-3)$ provide redundancy to the system.

Summarizing: given the force coupling matrix C_α and the velocities coupling matrix D, we can transform:

- motor forces into Euclidean accelerations: $a = C_\alpha f$
- Euclidean accelerations into motor forces: $f = C_\alpha^+ a$
- Euclidean speeds into motor speeds: $m = Dv$
- motor speeds into Euclidean speeds: $v = D^+ m$

3 Accurate Driving

It is possible to control an omnidirectional robot perfectly with the previously introduced methods if the friction between the wheels and the floor is infinite. However, in the real world the friction and thereby the acceleration of the robot is limited. First, we show how a slipping wheel can be detected by evaluating the wheel velocities. With this information, the motor forces can be reduced to prevent slippage.

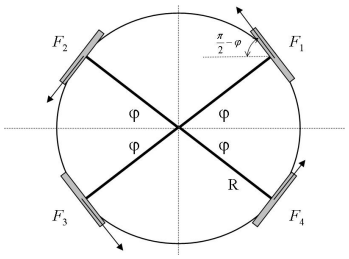

Figure 4: Arrangement of the wheels and distribution of forces in a symmetrical robot with 4 wheels. The same angle φ for each motor axis is used.

3.1 Identifying slipping wheels

Let us consider a symmetrical robot (see Fig 4). Let us call m the four-dimensional vector $(v_1, v_2, v_3, v_4)^{\mathrm{T}}$ of tangential motor speeds, D the velocities coupling matrix, and v the three-dimensional vector $(v_x, v_y, R\omega)^{\mathrm{T}}$. The fact that the Euclidean velocities of the robot and the tangential velocities of the motors are connected by the expressions

$$m = Dv \quad \text{and} \quad v = D^+ m$$

gives us the possibility of testing for inconsistencies in the motors speed and thus detect wheel slippage.

The controller on the robot gets the desired vector v by radio communication and transforms v into the necessary motor currents. After some time, the tick counters in the motors provide a vector of current motor speeds m'. We can test if wheels are slipping using the in-built redundancy of our motor values. Since $v' = D^+ m'$ and $m' = Dv'$, then it must be true that $m' = DD^+ m'$. If not, then one or more wheels are slipping on the ground. It would be extremely unlikely that they all slip at a rate which allows the expression to remain valid.

A simple computation shows that for a symmetrical robot with angle φ:

$$DD^+ = \frac{1}{4} \begin{pmatrix} 3 & 1 & -1 & 1 \\ 1 & 3 & 1 & -1 \\ -1 & 1 & 3 & 1 \\ 1 & -1 & 1 & 3 \end{pmatrix}$$

Checking spinning wheels can be done very easily by multiplying with this matrix, because we should obtain $(I - DD^+)m' = 0$. The matrix $I - DD^+$ is given by

$$I - DD^+ = \frac{1}{4} \begin{pmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{pmatrix}$$

The consistency check, as this matrix shows, reduces to the test

$$v_1 - v_2 + v_3 - v_4 = 0$$

or equivalently

$$v_1 + v_3 = v_2 + v_4$$

that is, the sum of the speeds of motor 1 with its opposite motor 3, is equal to the sum of the speed of motor 2 with its

opposite motor 4 (see Fig. 4 for the numeration of the motors). The speeds used before, that is $f_r = (1,1,1,1)^{\mathrm{T}}$ for rotation, $f_f(1,-1,-1,1)^{\mathrm{T}}$ for going forward, and $f_s(-1,-1,1,1)^{\mathrm{T}}$ for driving sideways, all fulfill the above consistency check. Any linear combination of them passes also the consistency check.

It is interesting to note that this slippage check does not depend on the angle φ. It is a universal test for omnidirectional four motor autonomous robots, with wheels placed symmetrically at an angle φ.

Slippage check can be also computed for any other configuration of n motors. We just have to compute the matrix $(I - DD^{+})$, and this provides the check we need. Consistent motor speeds m satisfy all the condition $(I - DD^{+})m = 0$.

In the case of an asymmetrical robot with motor axis at angles φ and θ, we can also compute algebraically the matrix $(I - DD^{+})$.

3.2 Energy-saving Driving

If we detect that the wheel rotation is not consistent, that is, that one or more wheels are wasting energy or slipping by false motor forces, can we do something about it? As this section shows, under certain conditions we can correct the motor control values without control losing time. Let us consider a symmetrical robot with the same angle φ for all motor axes with respect to the horizontal.

First, note that there are motor torque combinations which do not accelerate the robot. One such combination, for a symmetrical robot, is $f_k = (1,-1,1,-1)^{\mathrm{T}}$. The wheels pull in the forward and backward direction simultaneously. In such a case, the Euclidean magnitudes are zero, i.e.

$$a = C_\alpha f_k = 0$$

This means that the vector f_k is an element of the kernel of C_α (the set of vectors mapped to zero by this matrix).

Interestingly, if we have two combinations f and f' of motor forces which produce the same Euclidean accelerations a, then $a = C_\alpha f$ and $a = C_\alpha f'$. Therefore

$$C_\alpha(f - f') = 0$$

is true. This means that $f - f'$ belongs to the kernel of C_α. Also, if a vector g belongs to the kernel of C_α, then the vector of forces f produces the same accelerations as the vector of forces $f + g$ because

$$a = C_\alpha(f + g) = C_\alpha f + C_\alpha g = C_\alpha f$$

For a robot with four wheels the rank of the matrix C_α is exactly 3 (see chapter 2). Since the sum of the rank of an $n \times m$ matrix and the dimension of the kernel is equal to $\max(n, m)$, this means that the dimension of the kernel of C_α is one. This in turn implies that any vector in the kernel of C_α is of the form λf_k. Curiously enough, this is equivalent with every row in the matrix $(I - DD^{+})$ computed in the previous section for a four-wheeled symmetrical robot. What this means is that motor speeds $m = (v_1, v_2, v_3, v_4)^{\mathrm{T}}$ are consistent if they are orthogonal to the kernel of the matrix C_α, that is, to all vectors of the form λf_k. Motor speeds are inconsistent if they include a projection in the direction of the vector f_k. The solution? If we have inconsistent motor speeds $m = (v_1, v_2, v_3, v_4)^{\mathrm{T}}$, we compute the projection of m in the direction of the unit vector $\frac{1}{2} f_k$ and subtract it from m. The corrected m is thus:

$$m' = m - (m \cdot \frac{1}{2}(1,-1,1,-1)^{\mathrm{T}})\frac{1}{2}(1,-1,1,-1)^{\mathrm{T}}$$

The correction can be simplified to

$$m = m - \frac{v_1 - v_2 + v_3 - v_4}{4}(1,-1,1,-1)^{\mathrm{T}}$$

We can visualize this result as follows: in the four-dimensional space of motor values, there is a three dimensional subspace of consistent and non-slipping motor values (a three-dimensional hyperplane). Normal to this hyperspace we have the vector of wheel speeds $(1,-1,1,-1)$. Anytime wheels are slipping, we are wasting energy because the vector of motor values contains a component in the $(1,-1,1,-1)$ direction. What we have to do is to orthogonally project the vector of motor values back onto the hyperplane of consistent motor values, by just subtracting the component in the $(1,-1,1,-1)$ direction.

Incidentally, when we map Euclidean accelerations a to motor forces f, we always obtain consistent results because we use the expression

$$f = C_\alpha^{+} a$$

Since C_α and C_α^{+} are both of rank three, accelerations a (in three-dimensional space) are mapped one-to-one to forces f (in a three-dimensional subspace of the four-dimensional space). If this three dimensional subspace would include any element g of the kernel of C_α, then there would exist an acceleration a, different from zero, such that $g = C_\alpha^{+} a$. Since $C_\alpha g = 0$, then we would not have $a = C_\alpha C_\alpha^{+} a$, as we should have.

Also, if g belongs to the kernel of C_α, any force of the form $f + g$ is equivalent to f, because $C_\alpha f = C_\alpha(f + g)$. For an acceleration a, there is a unique force f such that $f = C_\alpha^{+} a$. Since f is always smaller in length than $f + g$ (because g is orthogonal to the subspace where f lives), we have the optimal situation where the forces computed require the minimal consumption of energy. Motors do not waste energy, when force combinations in the kernel of C_α are avoided.

4 Driving a real robot fast and accurate

Assume that we want to drive our robot forward, trying to avoid any significant wheel slippage. Assume that it has been determined experimentally that when the voltage for the DC motors is a maximum of 2 Volts the wheels will not slip (we can do this by holding the robot, and increasing the voltage until the wheels start to slip). We would like to drive as fast as possible (that is, with the maximum possible voltage for the motors) but without slipping. What we have to do then is to start the motors with $V_0 = 2$ and let the robot roll forward. After a few milliseconds, the induced current E in the rotor decreases the effective voltage on the rotor's solenoid to $V_0 - E$, and the motor torque correspondingly. We can now increase the value of the voltage to $V_1 = V_0 + E$, and now the motor torque corresponds to the effective voltage $V_0 + E - E = V_0$. Repeating this adjustment periodically, allows us to drive the motors with the maximum possible torque which does not let the wheels slip.

However, this consideration depends on a perfect mass distribution so that all wheels exert the same pressure on the floor. This assumption cannot be guaranteed when the robot is driving fast, because acceleration can lift sometimes the front or side wheels from the floor, or at least diminish the pressure they exert on the floor. Slipping wheels are a fact of life and they have to be handled in the PID controller.

4.1 PID controller

In our previous system, we sent the desired final linear velocity (v_x, v_y) to the robot and accelerated the wheels using one PID controller for each direction. A third PID controller received the desired final angular velocity ω and controlled the wheels. All PID controllers were interleaved. The wheel accelerations overlapped and we obtained the desired robot behavior. However, all parameters for the PIC controller must be set to adequate values to achieve fast acceleration of the robot in combination with low slippage. This can be done by experiments, heuristics or learning methods [6].

The above method works well if the robot accelerates slowly. The PID corrects the deviance from the desired behavior autonomously. The disadvantage of this method occurs in the case of a slipping wheel because the controller corrects then the consequences in a false way. An example: let us send $v = (0, 1, 1)$ as command to a still standing symmetrical robot. For a short time Δt the velocities are $\Delta m = \Delta t(1, -1, -1, 1)$. If the first wheels slips its force does not take effect on the robot, but it accelerates very fast. After Δt the tick counters measure $\Delta t(2, -1, -1, 1)$ as wheel velocities. This value is calculated by using D^+ back to robot velocity $v' = \Delta t(-\frac{1}{\sin\varphi}, \frac{1}{\cos\varphi}, 1)$. On the other hand, the effect of the slipping wheel on the robot is $f = \lambda(0, -1, -1, 1)$ with $\lambda > 0$ and robot's acceleration is given by:

$$a = \frac{\lambda}{M}(\sin\varphi, 3\cos\varphi, -\frac{1}{\alpha})$$

For the short time Δt we have the velocity

$$v'' = \Delta t a = \frac{\Delta t \lambda}{M}(\sin\varphi, 3\cos\varphi, -\frac{1}{\alpha}).$$

If we compare the measured velocity v' with the real velocity v'' we see that they are opposed to each other. Thatswhy the correction by the PID controller will go into the false direction because its calculation is based on the measured velocity.

There are two possibilities to avoid this problem is either to detect slipping wheels or to use one PID controller for each motor and not for every degree of freedom of the robot, what we do in our present system.

4.2 Control loop and correction

Normally, the robot is controlled and observed externally. In our system, for example, a camera is receiving images from the the top of the robot and the system calculates the exact position and orientation of the robot (see [3] for a full description). There is also always a delay in the system between sending commands to the robot and observing their impact. That means, that the robot always conducts commands which were calculated based on old data and that it can move in the meantime more than half a meter. In order to cancel this effect we are using a prediction system. It predicts the position and orientation of the robot after the delay, based on the behavior of the robot and the last sent commands [1].

However, the prediction system can be used also for correcting the robot movement in other cases. The expected behavior o the robot can be compared with its real one and if the robot behaves strange, its control commands can be corrected [5]. The desired wheel velocities $m = (v_1, v_2, v_3, v_4)$ are calculated from the given velocity vector $v = (v_x, v_y, R\omega)$ by $m = Dv$ with the velocity coupling matrix D. But to compensate the driving error the velocity vector v is modified. This modification is learned by observing the behavior of the robot over a certain time.

4.3 Driving without one motor

It can happen that a robot loses one of the four motors because of a hardware failure. The advantage of a four motor robot is that it can recover from such a loss. It can continue to drive accurately and can still behave as an omnidirectional robot.

Assume, for example, that the first motor in Fig. 4 malfunctions. The wheel can still roll, but the motor does not provide torque. In that case in the set of motor forces $f = (f_1, f_2, f_3, f_4)^{\mathrm{T}}$, f_1 is always zero. However, the three remaining motors can still drive the robot forward, sideways, and can let it rotate. Furthermore, the necessary motor torques are independent. The vector $\hat{f}_f = (0, 0, -1, 1)^{\mathrm{T}}$ of motor forces moves the robot forward without letting it rotate, the vector $\hat{f}_s = (0, -1, 1, 0)^{\mathrm{T}}$ moves the robot sideways to the right without letting it rotate, and the vector of forces $\hat{f}_r = (0, 1, 0, 1)^{\mathrm{T}}$ rotates the robot counterclockwise, without displacing its center of mass. It is easy to see that these (and multiples of them) are the only torque combinations with such properties. Note also that for every desired direction there is always another inactive motor beside the malfunctioning one.

As before, we can decouple the displacement of the robot from the rotation. We modify the control strategy for the motor values. We can drive the robot in the direction (v_x, v_y) by setting a combination of forces proportional to

$$\frac{v_x}{\sin(\varphi)}\hat{f}_s + \frac{v_y}{\cos(\varphi)}\hat{f}_s.$$

The robot will accelerate in the desired direction, and the two inactive motors will roll passively, without causing major problems. On top of that movement, we can let the robot rotate using any multiple of the vector of forces $\hat{f}_r$. Of course, the robot will be only half as fast as before, because we will be using only half of the motor power, but the robot will be still manageable.

Fig. 5 shows an experiment of a symmetrical robot with one non-working motor (motor 1) with and without correction of the robot velocities commands (as explained in 4.2). As can be seen, the robot moves accurate in the top-right and down-left direction also in the uncorrected case, because in these directions motor one is not involved in the movement.

If two, even three, motors are damaged it is still possible to drive the robot, but it ceases to be omnidirectional. In that case the high-level control must be changed in order to control the robot.

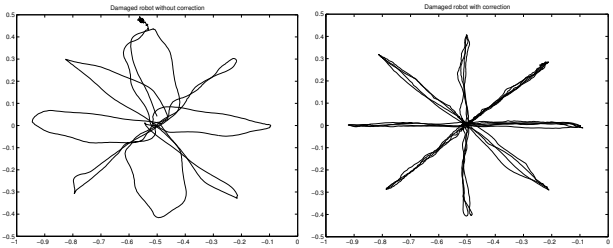

Figure 5: Star driving robot without one motor controlled without modifications of the desired motor velocities (left) and with correction (right). The wheels of the robot are symmetrical arranged with an angle φ of 45^o.

5 Related Work

Many researchers use PID controllers to reach the desired motor speed [2, 4, 7]. They use an inverse kinematics model for three or four wheels to calculate the desired wheel velocity or a dynamic nonlinear model of the robot [16]. Other researchers even use PID controllers for the translation and the rotation of the robot [14, 8]. Oubbati et al have replaced the PID controllers for the motors with a recurrent neural network to convert the desired wheel velocities to PWM commands [11]. The robots in these papers steer very slowly, hence wheel slippage does not appear and can be let unconsidered in these models.

The very fast Cornell Big Red team uses a simple dynamic motor model and an inverse kinematic model to generate bang-bang trajectories to move very fast to a given location with a three or four wheeled omnidirectoinal robot [10, 12]. The acceleration limit in there models is only the torque of the motors with respect to the velocity and not the friction between the floor and the wheel. For control, the robots are equipped with gyroscopes to measure the movement of the robot [13]. This method is much more accurate than measuring the movement indirectly with motor tick counters, because slippage can not influence the calculation.

Williams et al. measure the friction between omnidirectional wheels and carpet and also between wheels and paper[15]. They use a simple friction model for simulation and compare the results of the simulation with a fast accelerating and thereby slipping real three wheeled omnidirectional robot. The intention of the paper is to model and understand the sliding dynamics problem and not the real-time control of the robot, in contrast to the here presented work.

6 Conclusion

In this article we showed how to optimally control an omnidirectional robot. We presented the physical and theoretical foundations for controlling a n-wheeled robot and demonstrated practical experiences and problems. The major problems when driving an omnidirectional robot are to detect and handle wheel slippage and to handle the failure of motors. We presented methods for both avoiding and handling wheel slippage, as well as how to drive accurately without one motor. We are using all of the mentioned approaches successfully in our small size and middle size RoboCup robots. It is of course a required but not a sufficient share for best results in playing robot soccer.

Acknowledgements

This work was supported by the German Research Foundation SSP 1125 and partially funded by EU project FP6511931.

References

[1] S. Behnke et al.: "Predicting away the Delay", *RoboCup-2003: Robot Soccer World Cup VII*, Springer-Verlag, 2004.

[2] D. J. Daniel et al.: "Kinematics and Open-Loop Control of an Ilonator-Based Mobile Platform", *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1985.

[3] A. Egorova et al.: "FU Fighters Small Size Team 2004", *RoboCup-2004: Robot Soccer World Cup VIII*, Springer-Verlag, 2005.

[4] D. Feng et al.: "The Servo-Control System for an Omnidirectional Mobile Robot", *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1989.

[5] A. Gloye: "Learning Methods for Autonomous Mobile Robots", PhD thesis, Freie Universität Berlin, 2005.

[6] A. Gloye et al.: "Learning to Drive and Simulate Autonomous Mobile Robots", *RoboCup-2004: Robot Soccer World Cup VIII*, Springer-Verlag, 2005.

[7] L. Huang et al.: "Design and Analysis of a Four-wheel Omnidirectional Mobile Robot", *Proc. of the 2nd Int. Conf. on Autonomous Robots and Agents*, 2004.

[8] Y. Liu et al.: "Omni-Directional Mobile Robot Controller Design by Trajectory Linearization." *Proc. of the 2003 American Control Conference*, 2003.

[9] F. v. Hundelshausen et al.: "FU-Fighters Team Description 2003", *RoboCup-2003 - Proc. of the Int. Symposium*, 2003.

[10] T. Kalmár-Nagy et al.: "Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle", *Robotics and Autonomous Systems*, vol. 46, no. 1, Elsevier, 2004.

[11] M. Oubbati et al.: "Velocity Control of an Omnidirectional RobCup Player with Recurrent Neural Networks", *RoboCup-2005 - Proc. of the Int. Symposium*, preprint, 2006.

[12] O. Purwin et al.: "Trajectory Generation and Control for Four Wheeled Omnidirectional Vehicles", *Proc. American Control Conference*, 2005.

[13] O. Purwin et al.: "RoboCup 2003 Team Description: Cornell Big Red 2003", 2003.

[14] K. Watanabe et al.: "Feedback Control of an Omnidirectional Autonomous Platform for Mobile Service Robots", *Journal of Intelligent and Robotic Systems*, vol. 22, no. 3–4 , Kluwer, 1998.

[15] R. L. Williams et al.: "Dynamic Model With Slip for Wheeled Omnidirectional Robots", *IEEE Tr. on Robotics and Automation*, vol. 18, no. 3, June 2002.

[16] L. Wilson et al.: "Design and Modeling of a Redundant Omnidirectional RoboCup Goalie", *Proc. RoboCup 2001 Int. Symposium*, 2001.

Contact

Prof. Dr. Raul Rojas
Freie Universität Berlin
Takustr. 9, D-14159 Berlin
fon: +49 30 838 75 130
fax: +49 30 838 75 109
e-mail: rojas@inf.fu-berlin.de

Dr. Alexander Gloye Förster
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Galeria 2, CH-6928 Manno-Lugano
fon: +41 58 666 66 70
fax: +41 58 666 66 61
e-mail: alexander@idsia.ch

Bild

Univ. Prof. Dr. pol. Raul Rojas is Professor of Computer Science at the Freie Universität Berlin. His field of research is the theory and applications of artificial intelligence techniques. He has published papers about computer vision, robotics, and handwriting recognition, among other topics. His robotic soccer team, the FU-Fighters, has won twice the RoboCup world championship (small-size league). Prof. Rojas is member of the Editorial Board of several journals. He received the European Academic Software Award in 2002, the RoboCup Engineering Award in 2004, and the Wolfgang von Kempelen Prize in 2005.

Bild

Dr. rer. nat. Alexander Gloye Förster studied Mathematics and Computer Science at the Freie Universität Berlin, Germany, from where he received his MS in Mathematics (Dipl.-Math.) in 1996. From 1997 to 2000, he worked as Software Developer at the Ernst-Moritz-Arndt-Universität Greifswald, and at the Graphische Systeme GmbH in Berlin. In 2000 he came back to the FU Berlin as research assistant. From 2003 to 2004 he was the team leader of the FU-Fighters small size team. After receiving his PhD degree from the FU Berlin in 2005, he moved to Lugano as a postdoc at the Dalle Molle Institute for Artificial Intelligence.