Simple Path Planning Algorithm for Two-Wheeled Differentially Driven (2WDD) Soccer Robots

Gregor Novak[1] and Martin Seyr[2]

[1]Vienna University of Technology, Vienna, Austria
novak@bluetechnix.at

[2]Institute for Machine and Process Automation,
Vienna University of Technology, Vienna, Austria
seyr@impa.tuwien.ac.at

Abstract — *Well designed path planning algorithms are the key factor for moving robots. This paper describes a novel simple approach based on kinematics. The testing bed is a tiny two-wheeled robot. The robot's movement is specified by its translatoric velocity v_{R} and its angular velocity ω_{R}. The kinematic approach generates piecewise circular arcs based on a number of possible different sets of boundary conditions in the target positions.*

1 Introduction

In 1994 robot soccer was introduced with the theoretical background to develop multi-robot adaptive, co-operative, autonomous systems solving common tasks. A group of robots shall interact and self-organise autonomously in order to achieve a common goal. Further technical aspects besides co-operative and co-ordinated behaviour are miniaturisation of a complex electro-mechanical system, precise movement and optimal power efficiency. There are several categories in robot soccer: NaroSot, MiroSot, KheperaSot and HuroSot, classified by the size of the robots and the number of playing robots. In the category MiroSot the robot's size is limited to a 0.075m cube (http://www.fira.net).

Generally speaking, such a robot system has a global goal - to win a soccer game. In order to reach this goal tasks and subtasks are generated based on the actual game situation. The solution of a task or a subtask leads to a trajectory to be traced. Path planning itself is a difficult topic and a lot of highly sophisticated approaches for solving it have been published [1, 2, 3, 4, 5]. In this paper however, a simple approach based on kinematics is presented. The problem of path planning can be simplified by calculating intersection points to be reached subsequently. The presented approach has the advantage of a simple straight forward implementation, which enables a coordinated movement with comparatively small programming effort. Moreover, the algorithms require little computational performance. So this concept is ideal for hardware testing purposes.

The paper gives a short overview of the used testing bed, the robot's kinematics and the used algorithm for generating trajectories.

2 Tinyphoon - Testing Bed

As a testing bed a mobile mini robot is used, fig.1. This mini robot is two-wheeled and differentially driven (2WDD). It is characterized by a simple, compact and modular architecture. It has two DC motors controlled by pulse width modulated (PWM) voltage signals and reaches a speed of up to 2.5m/s. The whole robot with all its components fits into a cube with an edge length of 0.075m. The task of the robot's motoric unit is to optimally follow a desired trajectory.

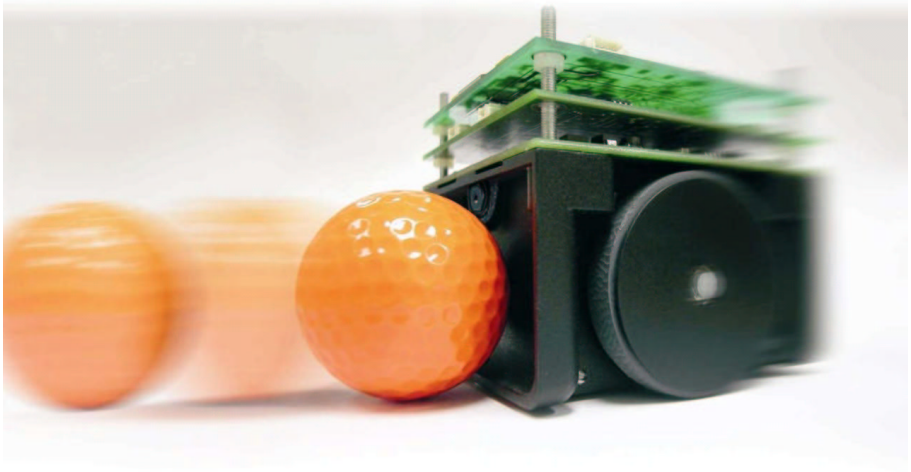

Figure 1: Tinyphoon

The mechanical unit of the mobile platform consists of the following parts:

- two wheels with rubber tyres
- two one stage transmissions
- two DC motors with magnetic two channel encoders
- Chassis with battery mounting
- Superstructure for the electronics unit

Beside the mechanical part the robot's motion unit consists of two electronic boards. One board contains the power electronics and sensors, the other one contains a micro controller XC167 by Infineon, a flash EPROM, a serial connection interface for programming, and a CAN bus interface. The CAN bus interface enables the communication between several micro controller boards for different tasks.

The modular and open architecture enables an easy implementation of additional sensors, for example for an onboard robot vision system.

The micro controller has six PWM outputs, three inputs for digital encoders, AD (analog digital) and DA (digital analog) converters and a number of free input and output pins, which generate interrupts either on rising or falling edges. Furthermore, it features CAN bus interfaces as well as asynchronal and synchronal serial interfaces. The primary task of the micro controller is to control the movement of the robot. Therefore four PWM output signals, two direction signals, and four pins for the two encoders are required. The micro controller is clocked with 40MHz.

Additionally, this board contains a Bluetooth radio module for communication with a host computer and other robots. This communication module can also be used for online configuration and debugging.

As already mentionaed the DC motors are controlled by PWM signals generated by the micro controller. Due to the fact that the motors need much more power (i.e. higher current) than the micro controller is able to provide, a driver is required. A dual full bridge driver is installed. The DC motors are controlled by feedback signals of two-channel digital encoders, which are mounted directly on the DC motors. It is planned to implement additional sensors on the power electronics-/sensor-board. These will be two two-axis acceleration sensors, a gyro sensor and a magnetic field sensor.

The algorithm itself is implemented on an additional DSP board. This board contains a Blackfin BF533 processor by Analog Devices and is linked via SPI bus to the XC167 board.

3 Kinematics

As reference values for the robot's movement the velocity v_{R} and the angular velocity ω_{R} are chosen (fig. 2). To convert these values to the velocities of the left wheel $v_{\mathrm{R,L}}$ and the right wheel $v_{\mathrm{R,R}}$, equations 1 and 2 are used.

All the necessary relations can be obtained from the kinematic model $(v_{\mathrm{R,L}}, v_{\mathrm{R,R}}) = f(v_{\mathrm{R}}, \omega_{\mathrm{R}})$,

$$v_{\mathrm{R,L}} = v_{\mathrm{R}} - \frac{B \cdot \omega_{\mathrm{R}}}{2},\ v_{\mathrm{R,R}} = v_{\mathrm{R}} + \frac{B \cdot \omega_{\mathrm{R}}}{2}; \tag{1}$$

solved for v_{R} and ω_{R}

$$v_{\mathrm{R}} = \frac{v_{\mathrm{R,L}} + v_{\mathrm{R,R}}}{2},\ \omega_{\mathrm{R}} = \frac{v_{\mathrm{R,R}} - v_{\mathrm{R,L}}}{B}. \tag{2}$$

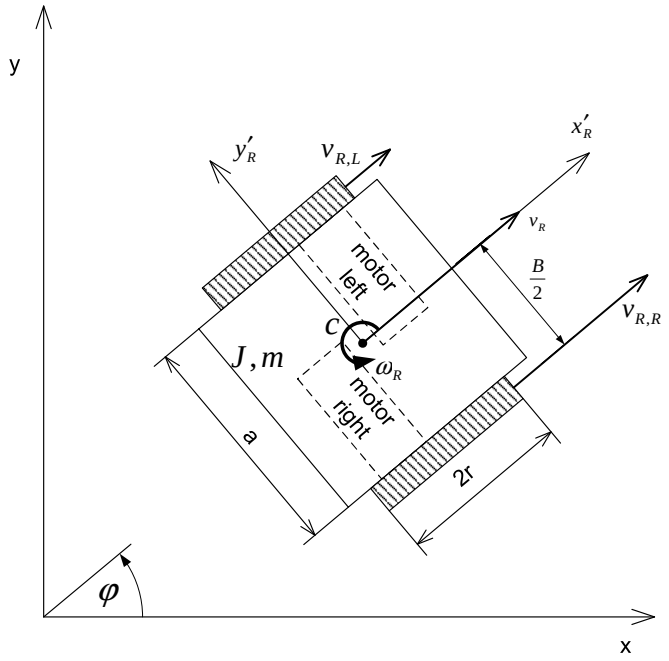

Figure 2: Kinematics and local coordinate system

The curvature ρ of the trajectory is obtained from the holonomic relation between the velocity v_{R} and the angular velocity ω_{R},

$$\rho = \frac{v_{\mathrm{R}}}{\omega_{\mathrm{R}}}. \tag{3}$$

The position of a robot moving along a trajectory is calculated with the following mechanism. The position of the robot at time 0 is $P_0 = [x_0, y_0, \varphi_0]^T$. The robot moves with the velocity v_{R} and angular velocity ω_{R}. The next position P is a function $P = f(P_0, v_{\mathrm{R}}, \omega_{\mathrm{R}}, \Delta t)$. The trajectory between the initial position and the next position at $t + \Delta t$ is a circular arc with the radius ρ.

In order to simplify the calculation a robot fixed coordinate system is introduced. The new origin is located in the center of the robot and the new x-axis (x'-axis) is the tangent to the current trajectory. Therefore the new coordinate system is shifted by x_0 in x-direction and by y_0 in y-direction. Furthermore, the new coordinate system has to be rotated by φ_0 (fig. 3).

Coordinate transformation:

$$x' = x\cos\varphi_0 + y\sin\varphi_0 - x_0\cos\varphi_0 - y_0\sin\varphi_0, \tag{4}$$

$$y' = -x\sin\varphi_0 + y\cos\varphi_0 + x_0\sin\varphi_0 - y_0\cos\varphi_0. \tag{5}$$

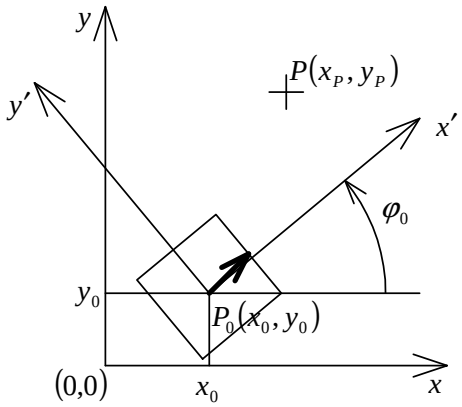

Figure 3: Coordinate Systems

Inverse transformation:

$$x = x' \cos\varphi_0 - y' \sin\varphi_0 + x_0 \tag{6}$$

$$y = x' \sin\varphi_0 + y' \cos\varphi_0 + y_0. \tag{7}$$

In the fig. 4 the initial position P_0 is located in the origin of the transformed coordinate system.

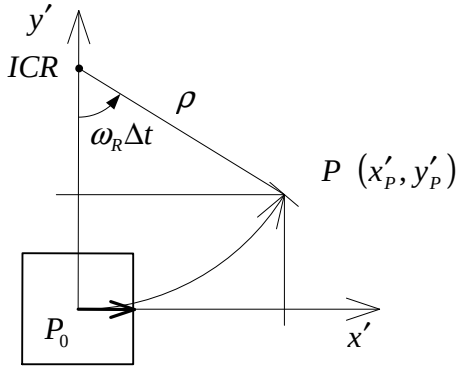

Figure 4: Circular arc around the instantaneous center of rotation

The new position after a time interval Δt is

$$x' = \rho \sin(\omega_{\mathrm{R}} \Delta t), \tag{8}$$

$$y' = \rho(1 - \cos(\omega_{\mathrm{R}} \Delta t)). \tag{9}$$

Applying the inverse coordinate transformation gives

$$x = \frac{v_{\mathrm{R}}}{\omega_{\mathrm{R}}} \left(\sin(\omega_{\mathrm{R}}\Delta t)\cos\varphi_0 - (1 - \cos(\omega_{\mathrm{R}}\Delta t)\sin\varphi_0)\right) + x_0, \tag{10}$$

$$y = \frac{v_{\mathrm{R}}}{\omega_{\mathrm{R}}} \left(\sin(\omega_{\mathrm{R}}\Delta t)\sin\varphi_0 + (1 - \cos(\omega_{\mathrm{R}}\Delta t)\cos\varphi_0)\right) + y_0 \tag{11}$$

and

$$\varphi = \omega_{\mathrm{R}}\Delta t + \varphi_0. \tag{12}$$

For $\omega_{\mathrm{R}} \to 0$ the new position is calculated from (8) and (9) using l'Hospital's rule

$$x' = v_{\mathrm{R}}\Delta t \tag{13}$$

$$y' = 0. \tag{14}$$

Applying the inverse coordinate transformation yields

$$x = v_{\mathrm{R}}\Delta t\cos\varphi_0 + x_0, \tag{15}$$

$$y = v_{\mathrm{R}}\Delta t\sin\varphi_0 + y_0 \tag{16}$$

and

$$\varphi = \varphi_0. \tag{17}$$

4 Path planning

4.1 Introduction

The control algorithm of the robot is based on a multi layer model. The bottom layer controls the mechanics in order to follow the robot's desired trajectory. The next higher layer calculates the trajectory, which is generally known as path planning. The calculation of a trajectory is based on intersection points, which are the result of solving the generated tasks. Solving the tasks is done by the next layer, which is itself a sub layer of the decision layer, [6].

The basic ability of such a robot is to reach target positions. There are various possibilities how a target position P can be described. It can be defined by arbitrary combinations of the following parameters:

- x- and y-coordinate, a position in the plane
- φ, the orientation in the target position
- v_{R}, the velocity in the target position
- T, time interval to reach the target position

4.2 Algorithms

Depending on whether the target orientation is specified or not, in this paper two different algorithms are presented. The first one calculates a path through a target position without a given orientation. Whether the target velocity and the time interval are specified or not does not affect the trajectory in this simple approach. The second algorithm first calculates a suitable intersection point to be reached using the first algorithm, and then calculates a circular arc through the target position. The target position can then be reached with the desired orientation. So the trajectory is calculated out of the target parameters and the present position, it consists of straight lines and circular arcs; the velocities and angular velocities are piecewise constant.

The robot does not have a distinguished front- or backside, both directions are equivalent. If the robot moves backwards the angle φ in the above derivations has to be changed to

$$\tilde{\varphi} = \varphi - \pi, \tag{18}$$

and the reference velocity v_{R} is multiplied by -1

$$\tilde{v}_{\mathrm{R}} = -v_{\mathrm{R}}. \tag{19}$$

The angular velocity ω_{R} remains unchanged.

4.3 Target position without specified target orientation

With this algorithm a target position for arbitrary target orientations is calculated from the parameters x_{R}, y_{R} and φ_{R} (the robot's present position) and x_{T} and y_{R} (the target position), see fig. 5.

The orientation φ_{R} has to be adjusted to face the target position first,

$$\Delta\varphi = \varphi_{\mathrm{T}} - \varphi_{\mathrm{R}}. \tag{20}$$

The angular velocity ω_{R} is calculated by

$$\omega_{\mathrm{R}} = \frac{K_{\mathrm{P}}}{\Delta T}\Delta\varphi, \tag{21}$$

where ΔT denotes the sampling time and K_{P} the gain of a P-controller.

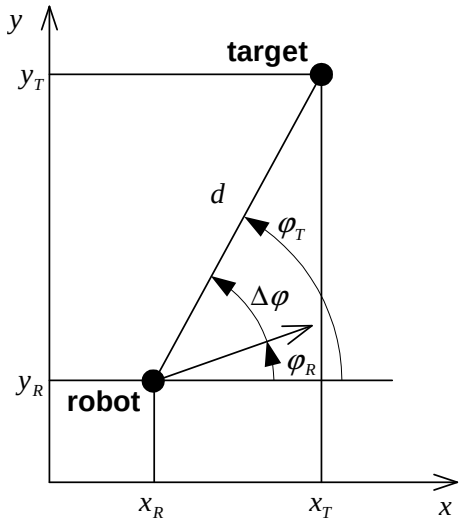

Figure 5: Target position without specified target orientation

The velocity v_{R} is increased with a constant maximum acceleration. If $\Delta\varphi$ is larger than a certain margin (to be determined experimentally), the translational acceleration starts after pivoting the robot on the spot. Otherwise there would be a significant deviation.

4.4 Target position including a specified orientation

In order to reach a target position with a specified orientation in this paper a trajectory consisting of a straight line (chapter 4.3) and a circular arc with a fixed radius is calculated, fig. 6. The radius depends on the robot's dimensions, its velocity and the goal object's dimensions (e.g. the ball).

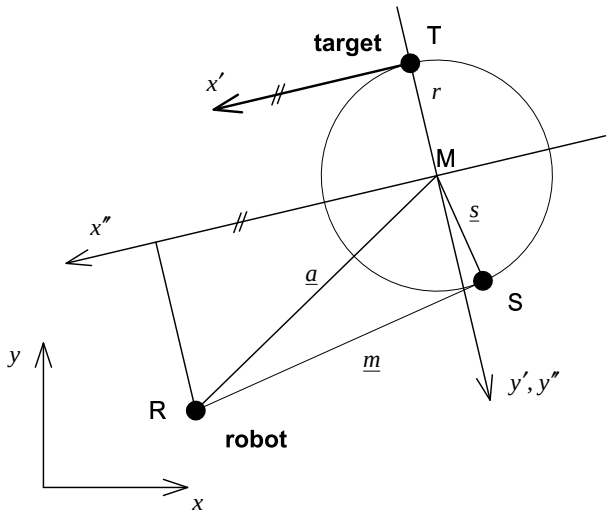

Figure 6: Trajectory for a target position with specified target orientation

For the calculation of the intersection point S the coordinate transformation equ. (4) and (5) is used. The coordinate system is rotated so that the new x-axis points into the direction of the target orientation, and its origin is shifted to the circle's center M. The intersection point S is the position where a straight line through the robot's present position R is a tangent to the circle. In equation (22) $''$ denotes the representation of a vector in terms of the $''$-coordinate system, fig. 6.

$$\underline{a}'' = [a_{x''}, a_{y''}]^T, \quad \underline{s}'' = [s_{x''}, s_{y''}]^T. \tag{22}$$

The vector from the intersection point S to the present position R reads

$$\underline{m}'' = \underline{a}'' - \underline{s}'' = [a_{x''} - s_{x''}, a_{y''} - s_{y''}]^T; \tag{23}$$

its absolute value

$$|\underline{m}''| = \sqrt{(a_{x''} - s_{x''})^2 + (a_{y''} - s_{y''})^2} \tag{24}$$

and *Pythagoras'* theorem

$$|\underline{m}''| = \sqrt{|\underline{a}''|^2 - |\underline{s}''|^2} \tag{25}$$

yield

$$(a_{x''} - s_{x''})^2 + (a_{y''} - s_{y''})^2 = a_{x''}^2 + a_{y''}^2 - s_{x''}^2 - s_{y''}^2. \tag{26}$$

Outmultiplying, replacing $s_{x''}^2 - s_{y''}^2$ with r^2, reordering and dividing by 2 gives

$$s_{x''} a_{x''} + s_{y''} a_{y''} = r^2. \tag{27}$$

Inserting equation 25 into 27 leads to a quadratic equation for $s_{y''}$

$$r^2 \left(r^2 - a_{x''}^2\right) - 2 a_{y''} r^2 s_{y''} + \left(a_{x''}^2 + a_{y''}^2\right) s_{y''}^2 = 0 \tag{28}$$

As can easily be seen from fig. 6 only the larger of the two real solutions is relevant. If there is a double solution, the present position of the robot is already on the circle, if there is a conjugate complex solution, the present position lies inside the circle, and the calculation has to be redone with a suitable smaller radius.

$$s_{y''} = \frac{a_{y''} r^2 + \sqrt{a_{x''}^4 r^2 + a_{x''}^2 a_{y''}^2 r^2 - a_{x''}^2 r^4}}{a_{x''}^2 + a_{y''}^2}. \tag{29}$$

For the x-coordinate of the intersection point S follows

$$s_{x''} = \frac{r^2}{a_{x''}} - \frac{s_{y''} a_{y''}}{a_{x''}}. \tag{30}$$

Remark I There are two possibilities to set a circle whose tangent is the target orientation. Of course the possibility nearer to the robot's present position has to be chosen.

Remark II The calculated coordinates can easily be transformed back into the original inertial or the robot-fixed coordinate system.

To follow a circular arc the robot's velocity v_{R} and angular velocity ω_{R} have to keep a constant relation. Furthermore the velocity is assumed to be kept constant, therefore the angular velocity has to be constant, too. That means that only the difference angle between the present orientation and the target orientation has to be calculated, fig. 7.

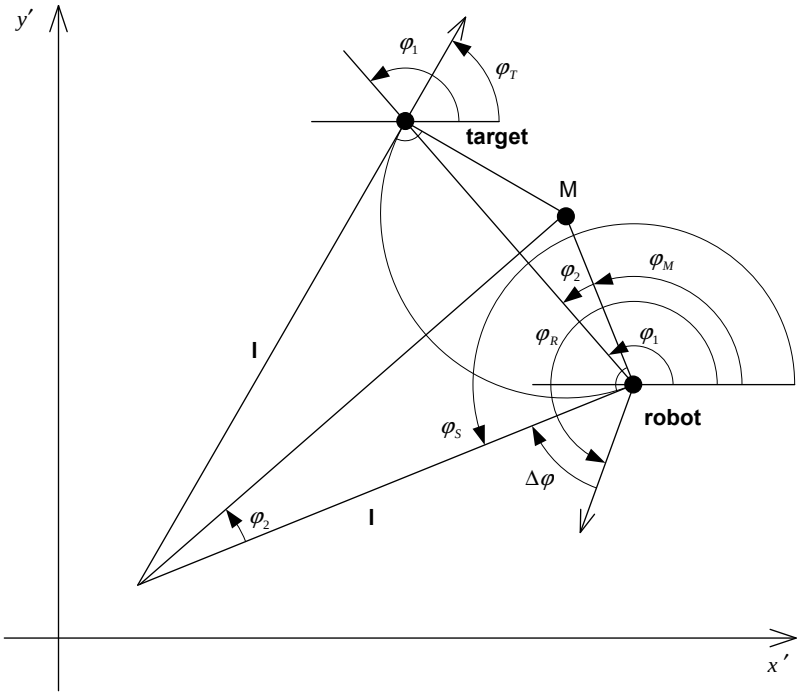

Figure 7: Angles in the circular arc

φ_{R}	actual direction
φ_{T}	target orientation
φ_{M}	angle of the vector pointing to the center of the circle
φ_{S}	tangent orientation in the starting point
φ_1	angle of the secant
φ_2	angle between secant and radius
$\Delta\varphi$	angle between actual direction and the tangent to the circle at the starting point

$$\varphi_1 = \arctan\left(\frac{y_{\mathrm{T}} - y_{\mathrm{R}}}{x_{\mathrm{T}} - x_{\mathrm{R}}}\right), \tag{31}$$

$$\varphi_{\mathrm{M}} = \arctan\left(\frac{y_{\mathrm{M}} - y_{\mathrm{R}}}{x_{\mathrm{M}} - x_{\mathrm{R}}}\right), \tag{32}$$

$$\varphi_2 = \varphi_1 - \varphi_{\mathrm{M}}, \tag{33}$$

and

$$\varphi_{\mathrm{S}} = \frac{\pi}{2} + \varphi_{\mathrm{M}}. \tag{34}$$

The angle φ_{curve} is the angle by which the robot has to be rotated to reach the desired orientation.

$$\varphi_{\mathrm{curve}} = 2(\frac{\pi}{2} - \varphi_2). \tag{35}$$

If there is a deviation from the tangential direction due to a possible disturbance the robot has to rotate by $\Delta\varphi$ first,

$$\Delta\varphi = \varphi_{\mathrm{S}} - \varphi_{\mathrm{R}}. \tag{36}$$

After a time interval $\Delta t = \frac{\varphi_{\mathrm{curve}}}{\omega_{\mathrm{R}}}$, assuming constant v_{R} and ω_{R}, the target is reached with the desired orientation.

5 Conclusions and further work

The described algorithm is very simple to implement and leads to feasible results, [7]. Nevertheless, a more sophisticated algorithm, which is based on an optimization algorithm using the calculus of variations, is currently being developed. The idea is to minimize a performance criterion based on the robot's initial position and orientation plus its target position and orientation. Furthermore, the robot's dynamic properties (e.g. nonlinear dynamic response of the power electronics or wheel slip) will be accounted for.
In future, stationary and even moving obstacles will be included in the calculation of the trajectory, as well as different combinations of target parameters.

References

[1] J.-M. Yang and J.-H. Kim. Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots. *IEEE Transactions on Robots and Automation*, (3), June 1999.

[2] A.M. Hussein and A. Elnagar. On optimal constrained trajectory planning in the plane. *International Journal of Robotics and Automation*, 14:33–38, 1997.

[3] J.V. Miro and A.S: White. Quasi-optimal trajectory planning and control of a CRS A251 industrial robot. *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, 216:343–356, 2002.

[4] J. Reuter. Mobile robots trajectories with continuously differentiable curvature: an optimal control approach. *Proceedings of the International Conference on Intelligent Robots and Systems*, 1:38–43, 1998.

[5] A. Elnagar and A. Basu. Smooth and acceleration minimizing trajectories for mobile robots. *Proceedings of the 1993 IEEE International Conference on Intelligent Robots and Systems*, 3:2215–2221, 1993.

[6] G. Novak. Robot soccer: An example for autonomous mobile cooperating robots. In *Proceedings of the First Workshop on Intelligent Solutions in Embedded Systems*, pages 107–118, Vienna, Austria, June 2003.

[7] G. Novak. *Multi Agent Systems - Robot Soccer*. PhD thesis, Vienna University of Technology, Vienna, Austria, 2002.

[8] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Number 0-521-56876-5. Cambridge University Press, Cambridge, UK, 2000.