Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea - May 21-26, 2001

Computation of Time Optimal Movements for Autonomous Parking of Non-Holonomic Mobile Platforms

Konstantin Kondak, Günter Hommel
Technische Universität Berlin, Institut für Technische Informatik
Franklinstraße 28/29, Sekretariat FR 2-2, D-10587 Berlin, Germany
{kondak, hommel}@cs.tu-berlin.de

Abstract

In this paper we present a method for calculation time-optimal movements for parking of non-holonomic mobile platforms. The parking problem is considered as specialization of the general problem of path planning for non-holonomic robots. This is formulated as a non-linear optimal control problem and solved using advanced numerical methods. The concept of artificial potential field is used for accounting for the obstacles in the environment. This method allows to compute the time-optimal control for all possible parking configurations (parallel, diagonal, row parking). The experiments show that using this method the movements for different complex situations can be calculated in a timely fashion.

1 Introduction

In this paper we address the problem of autonomous parking of non-holonomic mobile platforms. This problem is a specialization of the general problem of motion planning for non-holonomic robots. In this problem additional to the exponential complexity of the motion planning problem for holonomic robots [1] the non-holonomic constrains of the robot must be considered. For all practically important platforms these constraints can be expressed as $\sum \omega_i(\mathbf{s})\,\dot{\mathbf{s}} = 0$, where $\mathbf{s}(t)$ is the state (or configuration) vector, $\omega_i(.)$ a scalar function. A lot of elaborated methods for motion planning are based on geometric path planning in configuration space. The problem is that non-holonomic constraints are formulated for elements from the tangential space to the given configuration space. This means, that non-holonomic constraints cannot be transformed into configuration obstacles like obstacles in the environment and methods based on planning in configuration space are not applicable.

We pursue the approach to formulate the problem of motion planning as a problem of the non-linear constrained optimization and to solve this problem with proper numerical methods. The idea behind that is as follows; the movement of most robotic systems can be described by a set of state equations:

$$\dot{\mathbf{s}}(t) = \mathbf{f}(\mathbf{s}(t), \mathbf{u}(t)) \tag{1}$$

where $\mathbf{s}(t) = (s_1(t), s_2(t) \ldots s_n(t))^T$ is the system's state vector, $\mathbf{u}(t) = (u_1(t), u_2(t) \ldots u_m(t))^T$ the control vector, and $\mathbf{f}(\cdot,\cdot)$ a non-linear function, normally referred to as the system function. The problem of finding the optimal, collision-free trajectory for the robot can thus be formulated as a non-linear optimal control problem: Given a cost function, J, the actual state of the system, $\mathbf{s}_{act}$, and the desired end state, $\mathbf{s}_{end}$, find the control vector, $\mathbf{u}(t)$, which can take the system from state $\mathbf{s}_{act}$ to state $\mathbf{s}_{end}$ while minimizing J. The accounting for obstacles is based on a simple inequality imposed on an artificial potential field caused by detected obstacles in the environment. Section 3 describes this in more detail. The parameters for this inequality are the coordinates of the obstacle points. Apart from a simple numerical treatment this has the advantage that data from most conventional sensors (e.g., sonars and laser radars) can be used without any complex preprocessing. The number of obstacle points has only a very limited effect on the computation complexity of the method and thus poses no practical limitation. For the solution of the formulated optimization problem *sequential quadratic programming* (SQP) is used, which is outlined in Section 4.

In other papers of the authors [5], [6] the application of this method to the computation of the movement for a mobile robot and a simulated 2D mobile manipulator with a redundant arm were shown. In this work it is demonstrated that this general method can be used for solving a general parking problem.

In the literature a lot of interesting publication about motion planning for non-holonomic robots can be found e.g. [8], [9]. In spite of good planning results for movements in environments with moderate amount of free space these methods produce impracticable solutions for very narrow environments which are the normal case for many parking

0-7803-6475-9/01/$10.00©

situations.
This leads to the development of methods dealing directly with parking problem. Most of the methods of this group consider only the parallel parking problem and use some predefined motion patterns for achieving a sideways displacement of the platform. So e.g. in [11] the movement formed by *cos*-controls and in [4] the movement along circular arcs of minimal turn radius are used. The usage of motion patterns does not take advantage of all kinematic possibilities of a given platform and therefore for some complex situations inefficient movements are produced. In examples presented in these papers, the computed path consists of three parts (each followed with an opposite sign of linear velocity) for the parking bay which is twice as big as the car length with the depth equal to the width of the car (maximal steering angle 23° and 50°). For practical scenarios of car parking we often have to deal with parking bays which are much smaller, e.g. $120-150\%$ of the car length or smaller.
One obvious possibility to solve a general parking problem is to discretize the state (possibly also control) space and solve it using advanced search methods (e.g. A^*), so e.g. [12] describes a successful system for the parking of a wheel chair. These methods have the limitation, that in narrow cluttered environments the discretization of the configuration or action space must be quite fine for finding a solution. The fine discretization on the other hand makes the planning over large distances (from ca. $5m$) difficult since the number of states becomes very large.
We consider the problem of autonomous parking as consisting of the following three tasks:

1. Acquisition of the environmental data
2. Steering to a convenient start location
3. Performing a parking maneuver

The tasks 2. and 3. are shown in Fig. 1. This work is

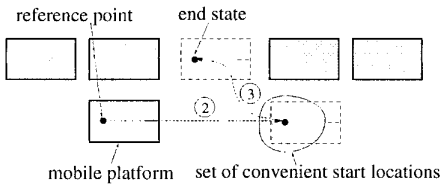

Figure 1: The tasks of autonomous parking problem

dedicated to task 3. The set of convenient start locations for parking maneuver in each concrete situation depends on the type of platform and on the method used for task 3. For our method applied to most kinds of mobile platforms, this set is quite large and can be given without any computation for all practically relevant configurations. Task 2. is in most cases elementary, it can be performed using the same method as task 3. or one of the more simple methods (e. g. movement along a straight line). Task 1. is difficult to be performed with common sensors for mobile robots like sonar or laser radars. For the practical use high safety is required. For this work the environmental data are assumed to be given.
The here described planning method computes the trajectory of the state and control vectors. The computed control values can be directly used for steering the platform in open loop mode. Because of uncertainty of the environmental data re-planning must be performed during the motion of the platform. This re-planning can be performed very fast by exploiting the fact that the solution from time t_0 can be used as a good initial estimate for the solution at time $t_0+\triangle t$. This is demonstrated among other examples for parallel and row parking in Section 5.
In Section 6 the assets and drawbacks of the method are summarized, conclusions are drawn and pointers to further work are given.

2 Model of a Non-Holonomic Mobile Platform

As a typical example of non-holonomic mobile platforms the car-like platform with drive on the rear axle will be considered. The state and control vectors for this platform can be defined as: $\mathbf{s}=(x,y,\theta)^T$ and $\mathbf{u}(\upsilon,\phi)^T$ where x and y are the coordinates of the robot's reference point (center of the rear axle), θ the orientation , υ the longitudinal velocity of the reference point and ϕ the steering angle. The movement is then described by the following system of state equations:

$$\begin{aligned} \dot{x} &= \upsilon\cos(\theta) \\ \dot{y} &= \upsilon\sin(\theta) \\ \dot{\theta} &= \upsilon/L\tan(\phi) \end{aligned} \tag{2}$$

where L is the distance between axes (wheel base).
Furthermore the following bounding box constrains for the platform are considered:

$$\begin{aligned} |\upsilon(t)| &\leq \upsilon_{max} \\ |\phi(t)| &\leq \phi_{max} \\ |\dot{\upsilon}(t)| &\leq a_{\upsilon max} \\ |\dot{\phi}(t)| &\leq v_{\phi max} \end{aligned} \tag{3}$$

The first two constraints restrict the values and the last two the change rates of the control variables. To mention here is that the violation or the missing of constraints (3) leads to computations of movements that can not be performed by a real platform.

3 Potential Field and Collision Avoidance

The collision-free movement of the platform is, as mentioned above, given with an inequality imposed on an artificial potential field created by obstacle points. The basic idea behind this representation is illustrated in Fig. 2 where the rectangle illustrates a mobile platform and the dots represent obstacle points.

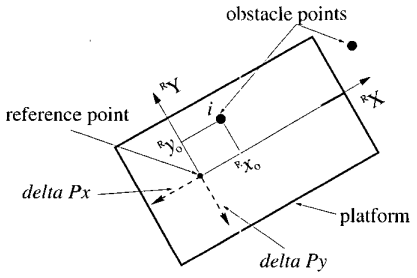

Figure 2: The platform and obstacle points

For this work the potential field caused by a particular obstacle point i has been defined as

$$P^i(\mathbf{s}) = \begin{cases} P^i_x(\mathbf{s}), \text{ if } R_Y - {}^R x^i_o \geq R_X - {}^R y^i_o \\ P^i_y(\mathbf{s}), \text{ if } R_Y - {}^R x^i_o < R_X - {}^R y^i_o \end{cases} \quad (4)$$

with

$$\begin{aligned} P^i_x(\mathbf{s}) &= P_{max}(1 - \tfrac{{}^R x^i_o}{R_Y}) \\ P^i_y(\mathbf{s}) &= P_{max}(1 - \tfrac{{}^R y^i_o}{R_X}) \end{aligned} \quad (5)$$

if the obstacle i is inside the platform, and

$$P^i_x(\mathbf{s}) = P^i_y(\mathbf{s}) \equiv 0 \quad (6)$$

if the obstacle i is outside of the platform. Here $({}^R x^i_o, {}^R y^i_o)$ are the coordinates of the obstacle point i in platform reference system, R_Y and R_X are the distances from the axes of the platform reference system to the border of the platform measured on the side the obstacle point i lies and P_{max} is a scale factor.

Given the coordinates of obstacle points, $P_i(\mathbf{s})$ obviously depends on the robot state $\mathbf{s}(t)$. A collision-free movement is thus ensured if:

$$P_i(\mathbf{s}) \leq 0 \;\; (i = 1,2,3 \ldots N) \quad (7)$$

where N is the number of obstacle points in the scene. Setting $P_i(\mathbf{s}) = 0$ for obstacles outside of the robot allows a linear superposition of the potential fields $P_i(\mathbf{s})$ and to formulate the condition for collision-free movement with only one inequality imposed on the resulting field:

$$P(\mathbf{s}) = \sum_{i=1}^{N} P_i(\mathbf{s}) \leq 0 \quad (8)$$

Observing this inequality guarantees that no collisions with detected obstacles occur. This kind of artificial potential field can easily be calculated for a large number of different robot geometries and due to the simplicity, the introduction of e.g., a safety distance, d_s, is straightforward.

In [7] we have shown for a potential field defined with eq. (4)-(6), that the obstacle point i, which violates the condition (8) causes the numerical methods, described in section 4, to change the actual computed trajectory $\mathbf{s}(t)$, so that the obstacle point i leaves the shape of the platform on the shortest possible path. It can for example be shown, that the vectors:

$$\begin{aligned} deltaPx &= (-\cos(\theta), -\sin(\theta))^T \\ deltaPy &= (\sin(\theta), -\cos(\theta))^T \end{aligned}$$

depicted in Fig. 2 are approximately the directions of changing the coordinates of the platform reference point, caused by active potential fields P_x and P_y.

4 Mathematical Formulation of the Problem and Numerical Methods

After the considerations from the previous sections the non-linear optimal control problem for computation of movement for a non-holonomic platform can be formulated as follows:

Given the state equations of the system (2), the actual and desired states $\mathbf{s}_{start}$, $\mathbf{s}_{end}$, find the control vector, $\mathbf{u}(t)$, which takes the system from state $\mathbf{s}_{start}$ to state $\mathbf{s}_{end}$ while minimizing the movement time t_{end} and satisfying the inequalities (8), (3).

At first the above formulated problem will be descretized using the *direct collocation method* [13] and transformed into the static non-linear optimization problem of the form:

$$\min F(\mathbf{x})$$

subject to:

$$\begin{aligned} c_i(\mathbf{x}) &= 0, \; i \in E \\ c_i(\mathbf{x}) &\geq 0, \; i \in I \end{aligned}$$

The vector $\mathbf{x} \in \Re^N$ contains the values of the state and the control vector at discrete time points:

$$\mathbf{x} = (\mathbf{s}^T_1, \mathbf{u}^T_1, \mathbf{s}^T_2, \mathbf{u}^T_2, \ldots, \mathbf{s}^T_M, \mathbf{u}^T_M, t_{end})^T$$

with $\mathbf{s}^T_k = (x(t_k), y(t_k), \theta(t_k))$ and $\mathbf{u}^T_k = (\upsilon(t_k), \phi(t_k))$ for $k = 1,2,\ldots,M$. The objective function is simply $F(\mathbf{x}) = t_{end}$. The functions $c_i : \Re^N \to \Re$, $i \in E$ - equality constraints - are obtained after discretization of state equations (2) and the functions $c_i : \Re^N \to \Re$, $i \in I$ - inequality constraints - after discretization of the constraints (3) and inequality (8)

for the artifical potential field. Note that the elements of the unknown vector $\mathbf{x}$ are real numbers and not functions like in the previously formulated non-linear optimal control problem.

After discretization the resulting static non-linear constrained optimization problem can be solved using *sequential quadratic programming* (SQP). The SQP-algorithm is the main part of computation. This algorithm solves the given non-linear optimization problem by iteratively solving quadratic optimization problems (one with linear constraints c_l and quadratic objective function F) which are obtained with the aid of linearization of the non-linear optimization problem. For a further description of the SQP-algorithm see [10], [2], [3].

For our computations, the numerical software package SNOPT [3] which contains the implementation of the SQP-algorithm has been used. The structure of the calculation is as follows:

1. The infinite-dimensional non-linear optimal control problem is transformed into static non-linear optimization problem by discretization using the direct collocation method
2. The resulting static non-linear optimization problem is solved using SNOPT
3. The given tolerances of the original problem are checked and if necessary the used discretization is refined. Thereafter the algorithms returns to step 1.

For our first experiments we used additionally the software package DIRCOL [13] which implements the loop formed by steps 1, 3 of the algorithm. This package has a convenient user interface which requires only the calculation of the right hand side of state equations (2) and inequalities (3), (8) to be provided. So the users, unfamiliar with numerical optimization, can start their experiments very fast, without programming and debugging the routines for discretization of the original problem. The direct implementation of steps 1 and 3 allows on the other hand a better adjustment of the problem to the solving process with the SQP-method and yields therefore better computation times.

5 Experimental Results

The results presented here were computed for a car-like platform with following dates: total length - $5m$, total width - $2m$, $L = 3m$, $\upsilon_{max} = 1m/s$, $\phi_{max} = 45°$, $a_{\upsilon max} = 0.5m/s^2$, $v_{\phi max} = 22.5 degree/s$. All computations were performed on a Pentium 400 MHz PC. The number of obstacle points in the scene does not affect the computation time significantly and varies, in the below shown examples, from 50 to 200. As the initial solution a simple geometric path (involving only the coordinates x and y) between $\mathbf{s}_{start}$ and $\mathbf{s}_{end}$ (straight ahead to the x coordinate of $\mathbf{s}_{end}$ and then perpendicular bottom-up) was used. Other components of the state and control vectors were set to 0. Alternatively a movement computed regardless of the obstacles can also be used as an appropriate initial solution. The computation times for this movement with the same algorithm are $0.3 - 0.6s$.

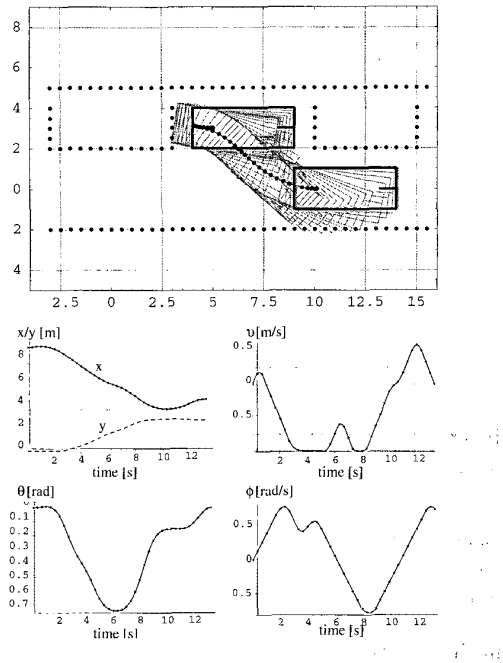

Figure 3: Parallel parking for a car-like platform, time optimal. $\mathbf{s}_{start} = (10,0,0)^T$, $\mathbf{s}_{end} = (5,3,0)^T$

In Fig. 3 and 4 the computed examples of the parallel backward parking maneuvers are shown. For both cases the actual and desired states were defined as $\mathbf{s}_{start} = (10,0,0)^T$ and $\mathbf{s}_{end} = (5,3,0)^T$. In all examples presented in this paper are $\upsilon_{start} = \upsilon_{end} = 0$, $\phi_{start} = 0$ and ϕ_{end} free. In the case of Fig. 3 there are $1.0m$ free spaces from each side of the car and in the case of Fig. 4 only $0.5m$ (the length of the parking bay is only 20% bigger than the length of the car). The progression of the longitudinal velocity υ in time shows that the movement for both cases consists mainly of two parts (except for the shot forward motion at the beginning): a longer movement backwards followed by a shorter movement forwards. Note that for the case in Fig. 4 the platform has been stopped for approx. $2s$ (for t between 12 and $14s$ $\upsilon(t) = 0$) waiting until the steering angle ϕ has reached the proper value. The change rate of the steering angle during this time has of course its maximum allowed value. According to the expectation the more difficult ma-

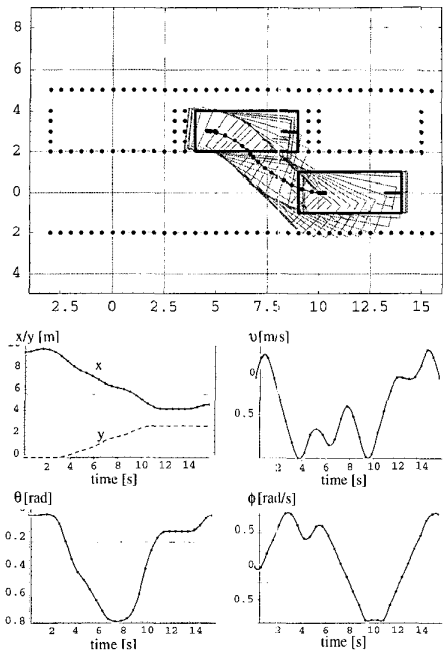

Figure 4: Parallel parking for a car-like platform, time optimal. $\mathbf{s}_{start} = (10,0,0)^T$, $\mathbf{s}_{end} = (5,3,0)^T$

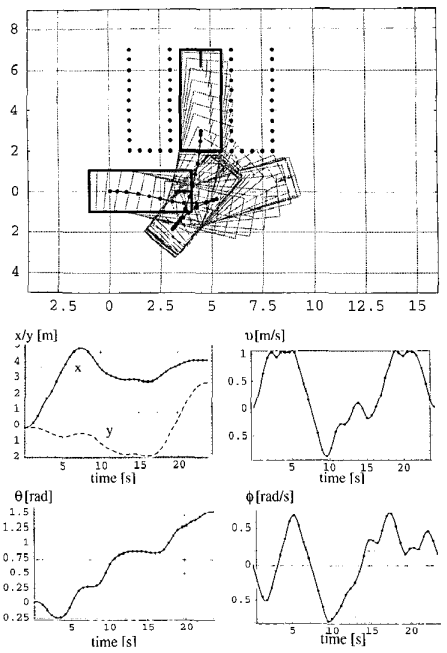

Figure 5: Row parking for a car-like platform, time optimal. $\mathbf{s}_{start} = (0,0,0)^T$, $\mathbf{s}_{end} = (4.5,3,\pi/2)^T$.

neuver in the case of Fig. 4 takes more time then the easier one in Fig. 3. The computation time for the examples in Fig. 3 and 4 was under $10s$. The re-computation (replanning) of an already computed movement with slight modification of the obstacle configurations or changing the actual position of the platform can be performed on average under $1s$. So the computation of the movement for the case in Fig. 4 with the initial solution taken from results of the computation in Fig. 3, lasts approx. $2s$ despite relative significant change of the obstacles.

In Fig. 5 and 6 the computed examples for row parking are shown. The actual and desired states were in these examples: $\mathbf{s}_{start} = (0,0,0)^T$, $\mathbf{s}_{end} = (4.5,3,\pi/2)^T$ and $\mathbf{s}_{start} = (0,0,0)^T$, $\mathbf{s}_{end} = (4.5,6,-\pi/2)^T$ respectively. The computation time for these examples was approx. $15s$. The free space from each side was $0.5m$.

Unfortunately in some difficult situations our method does not produce a result (divergence of numerical methods). So it was not possible to compute the movement for the example in Fig. 5 with the obstacles from the example in Fig. 6 (additional obstacle line below).

6 Conclusion

A method for the calculation of time optimal movement for the general parking problem has been presented. The method is based on an adjustment of a general method for non-holonomic motion planning (earlier presented in [5]) to a parking problem. The task was formulated as a nonlinear optimal control problem which is solved using advanced techniques from numerical analysis. The main advantages of the method can be summarized as follows:

- *The elegant and general mathematical problem description*; the formulation of the problem of motion planning as an optimal control problem covers not only the autonomous parking but also a big variety of other application in robotics.
- *The method computes movements for various parking situations*; the distinction between parallel, row, diagonal etc. parking is not necessary.
- *A high quality of the computed movement*; the method tries to use all kinematic facilities of the platform and produces movements which take significant less time than other methods known to us.
- *The algorithm yields not only the geometric path, but also control values* which can be directly used for steering a platform.
- *The data from common sensors can be directly used* without preprocessing.

The main disadvantage of the method is the relative long initial computation time (in shown examples up to $15s$).

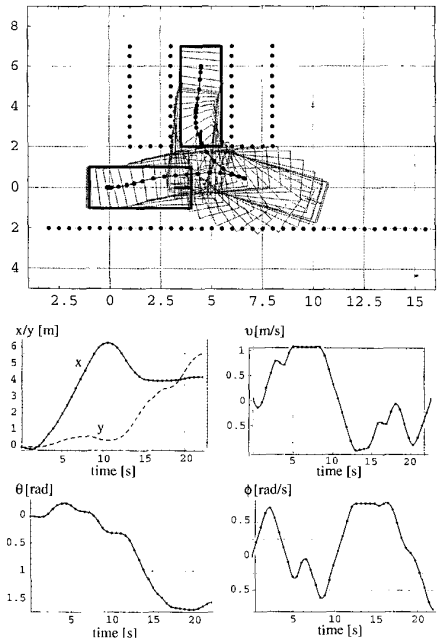

Figure 6: Row parking for a car-like platform, time optimal. $\mathbf{s}_{start} = (0,0,0)^T$, $\mathbf{s}_{end} = (4.5,6,-\pi/2)^T$.

But this is partially compensated by the quality of the solution. So it is better to spend about $10s$ for computation and afterwards complete the parking maneuver in $15s$, than moving the platform in the parking bay with a very fast method for several minutes.

Another disadvantage of the method lies in the fact, that it does not converge to a solution in some very difficult configurations. This can be detected in the computation process quite quickly and another try with a slightly changed initial solution can be started.

The applied optimization method does not necessarily converge to the global minimum, i.e., the computed solutions are only locally optimal. This is not important for practical use because, as the computed examples show, the solutions are good enough for practical use.

The initial solution is important for the convergence of the method and computation time and has an influence to which local optimum the solution will converge to. Thus we are currently investigating methods for fast generation of appropriate (not necessarily feasible) start solutions.

We are also working on decreasing the computation time through better adjustment of the problem to the SQP-algorithm. We expect also that the computation time will decrease by reducing the (currently very high) accuracy of the numerical methods.

The test of the method on a real platform is planned for the next time.

Acknowledgments

We would like to thank Philip Gill and Oskar von Stryk for helpful discussions and for providing the numerical software packages.

References

1. J. Canny. *"The Complexity of Robot Motion Planning"*. Doctoral Dissertation. The MIT Press, 1988.
2. P. E. Gill, W. Murray, M. H. Wright. *"Practical Optimization"*. Academic Press, 1981.
3. P. E. Gill, W. Murray, M. A. Saunders. *"SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization"*. Report NA 97-2, Dept. of Mathematics, University of California, San Diego.
4. K. Jiang, L. D. Seneviratne. *"A Sensor Guided Autonomous Parking System for Nonholonomic Mobile Robots"*. IEEE International Conference on Robotics and Automation 1999.
5. K. Kondak, G. Hommel, S. Horstmann, S. Kristensen. *"Computation of Optimal and Collisionfree Movements for Mobile Robots"*. Accepted for IEEE/RSJ International Conference on Intelligent Robots and Systems 2000.
6. K. Kondak, G. Hommel. *"Computation of Optimal and Collision-free Movements for Non-Holonomic Mobile Platforms Moving in Narrow, Cluttered Enviroment"*. Proposed for ICRA2001.
7. K. Kondak, G. Hommel. *"Berechnung der kollisionsfreien Bewegung für mobile Roboter mit Hilfe von Optimierungsmethoden"*. Techn. Bericht Nr. 2000-10, ISSN 1436-9915, Technische Universität Berlin.
8. J.-P. Laumond (Ed.). *"Robot Motion Planning and Control"*. Lecture Notes in Control and Information Sciences 229. Springer1998.
9. Z. Li, J. F. Canny. *"Nonholonomic Motion Planning"*. Kluwer Academic Publishers, 1993.
10. J. Nocedal, S. J. Wright. *"Numerical Optimization"*. Springer, 1999.
11. I. E. Paromtchik, C. Laugier. *"Motion Generation and Control for Parking an Autonomous Vehicle"*. IEEE International Conference on Robotics and Automation 1996.
12. M. Strobel. *"Navigation in Partially Unknown, Narrow, Cluttered Space"*. IEEE International Conference an Robotics & Automation, 1999.
13. O. von Stryk. *"User's Guide for DIRCOL: a Direct Collocation Method for the Numerical Solution of Optimal Control Problems"*. http://www-m2.ma.tum.de/~stryk