First published in:

2008 IEEE Intelligent Vehicles Symposium
Eindhoven University of Technology
Eindhoven, The Netherlands, June 4-6, 2008

Navigating car-like Robots in unstructured Environments using an Obstacle sensitive Cost Function

Julius Ziegler
Department of Measurement and Control
University of Karlsruhe (TH)
76131 Karlsruhe, Germany
julius.ziegler@mrt.uka.de

Moritz Werling
Department of Applied Computer Science/Automation
University of Karlsruhe
76128 Karlsruhe Germany
moritz.werling@iai.fzk.de

Joachim Schröder
Industrial Applications of Informatics and Microsystems
University of Karlsruhe
76131 Karlsruhe Germany
joachim.schroeder@ira.uka.de

Abstract— **We propose a method for navigating a car-like vehicle within an unstructured environment. Path planning is posed as a graph search problem. The search graph is set up in a way that implies derivation of a feed forward term for a downstream closed loop controller. An informed search algorithm is used that is guided by a heuristic cost function that accounts for both kinematic constraints of the vehicle and the topology of the vehicle's free space. Configuration space obstacles are computed from an obstacle map acquired from a high definition laser range scanner and search is restricted to the collision free subset of the configuration space. The algorithm allows for solving all of the following problems: Precise parking maneuvers, narrow turns, long distance navigation. The system has been used sucessfully on board the autonomous car ANNIEWAY in the DARPA Urban Challenge competition of 2007.**

I. INTRODUCTION

Planning collision free paths is of prime importance in robotic applications. Path planning becomes more challenging when applied to a robotic system of a non-holonomic nature, like a four-wheeled vehicle, since kinematic constraints of the system must be accounted for. In this work, we present a method for planning paths for such a system that interfaces directly to a closed loop controller. Obstacles are input as a discrete map that is obtained from a laser range scanner.

In structured areas, i.e. areas where a geometrical road network description is available, paths can be generated in a straight forward way from a graph representation of this network. In case of the ANNIEWAY robot, a complex, hierarchically organised state machine generated paths from this graph in compliance with traffic regulations. In an unstructured environment, like parking lots, loading zones and offroad areas, a graph for path planning is first to be established. Some approaches to path planning generate it explicitly, like the popular probabilistic path planning (PPP) algorithm introduced in [8], that utilises a graph that is established by sampling its nodes randomly from the free configuration space.

We define an implicit graph that is expanded on the fly by an A* search algorithm. A* search is a well known concept in the domain of robotic path planning (*cf.* [3]), that allows for accelerating exploration of the search space by defining a heuristic cost function that gives expected cost-to-go for each node of the search graph. If the cost function underestimates the actual distance to the goal, A* is guaranteed to find the least-cost path. If the error of the cost function is big, A* quickly degenerates to an exponential time algorithm. This is common when a metric cost function is used and search gets stuck in a dead end configuration. We avoid this problem by designing an obstacle sensitive cost function that accounts for the topology of the free space.

Search is performed on a graph in which all paths are feasible. It is directly derived from a kinematic model of the car and not only guarantees feasibility of the generated path, but also allows for straight forward design of a combined feed forward/feed backward controller. Adding a feed forward term makes the controller react more quickly and accurate, since reaction of the vehicle to steering input is modelled seperatly from controller offset introduced by noise. A tight coupling of path planning and closed loop control turned out to be useful and yielded a very robust and capable system.

We restrict search to the collision free subset of configuration space by calculating configuration space obstacles from an obstacle map obtained from a 360°-laser range scanner. The discrete nature of this obstacle map motivated dealing with configuration space obstacles in a discrete way as well, as opposed to more traditional approaches that require obstacle input in the form of polygonal data ([8]).

As part of the ANNIEWAY project, the algorithm is used whenever the available road network definition is not sufficient to generate paths from it directly. This is the case

978-1-4244-2569-3/08/$20.00 ©2008 IEEE.

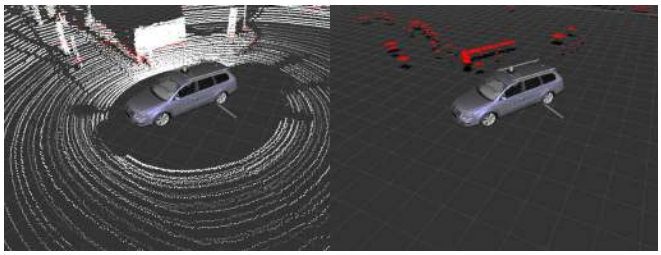

Fig. 1. Laser range scanner and obstacle mapping. Vertical structures detected within laser scanner data are mapped to a 2D-grid.

for parking maneuvers, turns and whenever recovery from an unforeseen situation is required. Parking maneuvers, in the Urban Challenge (UC), take place within special zones that are defined solely by perimeter points. No additional information for navigating within a zone is given, and it can contain static and dynamic obstalces. While it is possible to precompute a k-turn maneuver from sufficient road definition in an ad hoc way, our algorithms can plan turning maneuvers in a much more generally. Even in structured environment with detailed road geometry data, the algorithm is used as a fall back solution whenever the vehicle gets off track, the road is blocked or no sensible localisation within the given road definition is possible.

II. Configuration Space

Planning is performed within the three dimensional configuration space $\mathcal{C}$, that is spanned by a 2D-position $\vec{x}$ and orientation ψ of the robot. In the spirit of [4], our algorithm restricts $\mathcal{C}$ to a discrete space. The upstream laser scanning system detects vertical structures in the environment and accumulates these over time, so that, by continous exploration, a complete map of the environment can be obtained. The collision free subset of $\mathcal{C}$, called the free space of the robot, is computed from this discrete obstacle map.

Let n_ψ denote the number of discrete orientations of the vehicle. The 2D-obstacle map can be transfered to the discretised configuration space by convoluting it n_ψ times with a structuring kernel as depicted in figure 2. The shape of the kernel is chosen to resemble the shape of the vehicle, but is dilated by a 1 m disk to guarantee the safety distance required by UC regulations [1]. By precomputing the configuration space in discretised form, a collision check for a certain configuration can be performed quickly in $\mathcal{O}(1)$ by a simple

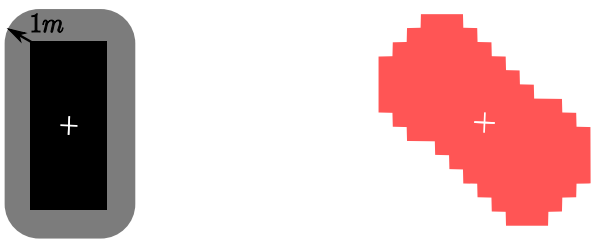

Fig. 2. Structuring elements for transferring obstacles into the discretised configuration space. Left: A 1 m safety distance is added to the shape of the robot. Right: Structuring element for one discrete angle.

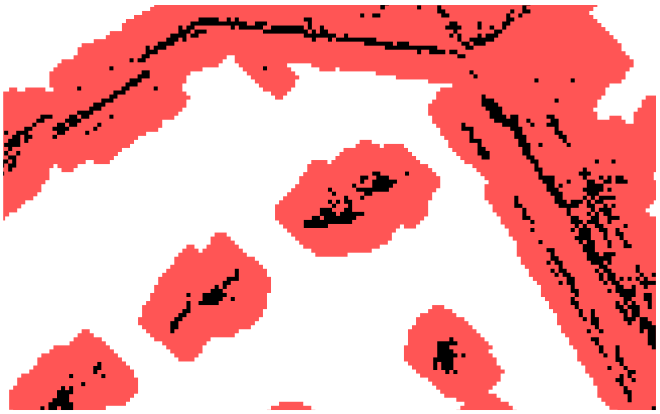

Fig. 3. Configuration space obstacles. Result of convoluting an obstacle map acquired from processing lidar data (black) with the structuring element from figure 2. Note that if the robot has the same orientation as the structuring element and is positioned in the red area, it must intersect with an obstacle.

table lookup. Figure 3 shows an xy-slice of the configuration space with configuration space obstacles for one orientation.

III. Implicit Search Graph

To guarantee feasibility of the generated path, we search a graph in which all traversions obey the kinematic constraints of the vehicle. A node of the search graph can be completely described by a tuple $(\vec{x}, \psi, \delta)$, with $\vec{x}$, ψ and δ denoting position, orientation and steering angle (i.e. the deflection of the front wheels) of an instance of the kinematic model (see figure 4). Steering angle δ is from a set of n_δ discrete steering angles that are distibuted equidistantly over the range of feasible steering: $D = \{\delta_1 \dots \delta_{n_\delta}\}$. All nodes of the graph are connected by an arc of fixed length .

We generate the successors of a node $v_p = (\vec{x}_p, \psi_p, \delta_p)$ by solving the kinematic model equations for initial values taken from v_p and $\dot{\delta} = \frac{\delta_p - \delta_i}{s}$, for each $\delta_i \in D$. This spans arcs between the nodes that closely resemble clothoid segments. It is equivalent of driving the car model over a distance s at constant speed while uniformly turning the front wheels from δ_p to δ_i. For the set of nodes $\{(\vec{0}, 0, \delta_i), \delta_i \in D\}$, this results in n_δ^2 successors, and another n_δ^2 if backward motion is allowed. Successors of other nodes can be generated

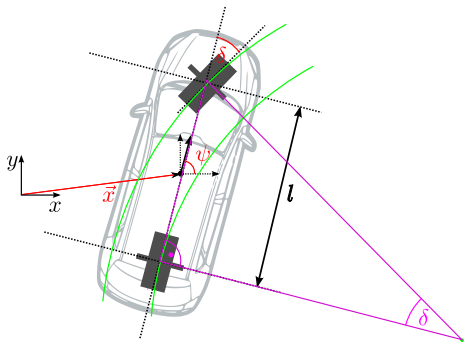

Fig. 4. Kinematic one track model. Model equations can be derived from side ratios in the pink triangle.

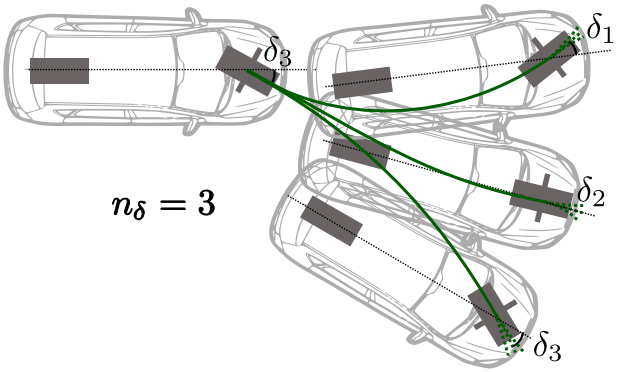

Fig. 5. Implicit search graph. Each node has n_δ successors. The connecting arcs resemble clothoid segments.

quickly from this precomputed set by subsequent rotation and translation. Figure 5 shows part of the search graph.

IV. Heuristics

To guide the search process we combined two different cost functions. The first one accounts for kinematic contraints of the vehicle, while the second one is derived from the Voronoj graph of the vehicle's free space and so incorporates knowledge of the obstacle's shapes and positions.

Local cost function

We use the so called *RTR metric* as a local cost function. RTR (rotation-translation-rotation) paths connect two configurations by two circular arcs of minimum turning radius and a straight segment tangenting both. It can be shown easily (*cf.* [8]), that for every pair of configurations a finite number of such paths can be constructed. The RTR metric is the arclength of the shortest such path. RTR paths do neither have continous curvature nor are they optimal (the optimal - in terms of arclength - solution to the local navigation problem are the so called *Reeds and Shepp paths*, *cf.* [6]), but are preferred by us due to their computational simplicity. Figure 6 illustrates RTR metric.

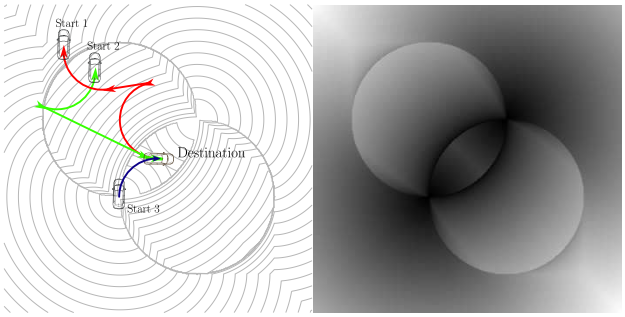

Fig. 6. Local cost function for -90° change or direction. The left hand side shows three examples for minimum RTR paths. Destination position was in the middle of the image, destination orientation was to the right. The right image shows the value of the RTR metric, evaluated for an upward starting position (bright: high value, dark: low value). Some equidistance lines are superimposed on the left image.

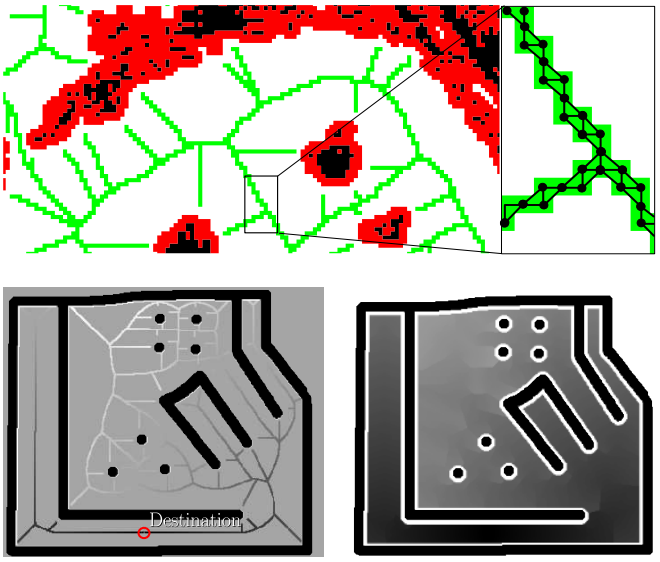

Fig. 7. Voronoj based cost function. Top: Voronoj lines (green) of the free space are generated as an 8-connected graph. Bottom left: Voronoj graph labeled using Dijkstras algorithm. Brightness corresponds to shortest-path distance to destination (red circle). Bottom right: Cost function evaluated over $\mathbb{R}^2$ by matching to the Voronoj graph (equidistant lines are superimposed to visualise gradient direction).

Voronoj based cost function

We construct a powerful, obstacle sensitive cost function based on the Voronoj graph of the free space of the vehicle. Actually, a superset of the free space is used that is invariant to the vehicles orientation. It is generated by generating configuration space obstacles for a disk shaped structure that is the intersection of all structuring elements from figure 2.

Our algorithm to calculate Voronoj lines from a binarised obstacle map is similar to [5], however, instead of using the vector distance map, we use the approximate chamfer metric to be able to label Voronoj lines using only two passes over the obstacle map. The method is derived from an algorithm ([2], [7]) for calculating the euclidean distance transform. It gives the Voronoj lines as a set of 8-connected pixels.

After matching the target position to the closest point on the Voronoj graph, Dijkstras algorithm is used to calculate the shortest path distance to the target position for every point on the graph. Cost for a position not on the graph is derived by matching to the closest point on the graph and incorporating the matching distance in a way that leads to a gradient of the cost function that is slightly sloped towards the Voronoj lines.

Using this heuristic function is appealing for several reasons. Since the Voronoj lines comprise the complete topology of the free space, search cannot get stuck in a dead end configuration, as is common with conventional, metric heuristics that do not incorporate knowledge of free space topology and therefore grossly underestimate the cost in such a case. Additionally, the Voronoj lines have - as the centers of maximum inscribing circles - the property of being at the farthest distances possible from any obstacle. This is conveyed to the planned paths, giving reserves to account

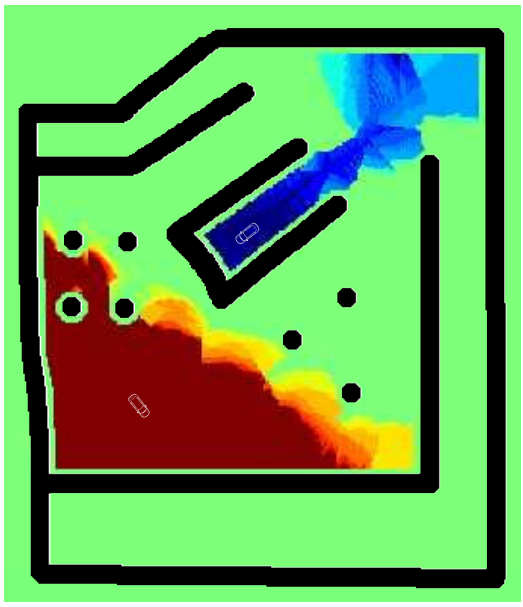

Fig. 8. Combination of cost functions. The image shows where local cost function is active for two different destination positions (red and blue). Voronoj based cost function is active in the green area. The boundary line is blurry, since local cost function is dependant on the orientation of the starting configuration. Note that, starting from green regions, cost is dominated by the necessity to maneuver around obstacles.

for control- and measurement errors.

Combination of cost functions

We combine the two cost functions into one by the maximum operator. This procedure can be justified from the admissibility principle for heuristics in the context of A* search. A heuristic is called *admissible,* if it consistently *under*estimates the cost to the target node. Consequently, combining two heuristics via the maximum operator still gives an admissible heuristic. Result of comparing both cost functions can be seen in figure 8. It coincides with the practical experience that in the vicinity of the target position, cost is dominated by the necessity to maneuver in order to reach the destination in right orientation, while cost at long distances often is caused by the necessity to avoid obstacles.

V. Closed Loop Control

Control strategy is derived from the same non-holonomic kinematic model (figure 9) as the search graph. An orbital tracking controller is employed to minimise lateral offset to the generated path, while longitudinal dynamics are subjected to a separate controller.

The lateral dynamics of the vehicle displayed in figure 9 can be described in local coordinates s, d, and $\Delta\psi$ (cf. figure 9). Expressing the dynamics with respect to covered arc length s_c of the planned path rather than time, with $\frac{\mathrm{d}}{\mathrm{d}t}() = \frac{\mathrm{d}}{\mathrm{d}s_c}() \cdot \frac{\mathrm{d}s_c}{\mathrm{d}t}$ the time-independent system becomes

$$\frac{\mathrm{d}}{\mathrm{d}s_c}\begin{bmatrix} s_c \\ d \\ \Delta\psi \end{bmatrix} = \begin{bmatrix} 1 \\ \sin\Delta\psi \cdot \frac{1-d\kappa_c(s_c)}{\cos\Delta\psi} \\ \frac{\tan\delta}{l} \cdot \frac{1-d\kappa_c(s_c)}{\cos\Delta\psi} - \kappa_c(s_c) \end{bmatrix}. \quad (1)$$

In combination with the feedback linearizing control law

$$\delta = \arctan(-lk_0 d - lk_1\Delta\psi + l\kappa_c) \quad (2)$$

with $k_0, k_1 > 0$ a stable linear error dynamics

$$\frac{\mathrm{d}}{\mathrm{d}s_c}\begin{bmatrix} d \\ \Delta\psi \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_0 & -k_1 \end{bmatrix}\begin{bmatrix} d \\ \Delta\psi \end{bmatrix} \quad (3)$$

with respect to s_c with the characteristic polynomial

$$\lambda^2 + k_1\lambda + k_0 = 0 \quad (4)$$

is given in the vicinity of the planned path. The controller input $(d, \Delta\psi, \kappa)$ is derived from the discrete representation of the planned path (nodes) via interpolation.

As can be seen in the simulation result in figure 10, the velocity-independent transient behavior (*orbits*) to different initial errors d and $\Delta\psi$ for forward (blue) and backward driving (red) is stable and velocity independent.

The longitudinal controller comprises of a linear proportional velocity and a nonlinear stopping controller. The latter asserts a constant deceleration until the vehicle arrives its final position as soon as a certain deceleration threshold is exceeded. Via a min-operator the desired accelerations of both controllers is combined and finally converted to accelerator and brake pressure values by a straightforward split-range strategy.

VI. Experiments and Results

Due to the choice of heuristics, A* search gives good and fast results for all practical path planning problems in a static environment. The Voronoj heuristics guides the search quickly towards the target even in difficult, maze like environments where conventional, metric heuristics perform poorly (figure 11). The local heuristic allows for efficiently planning parking maneuvers even where little space is available, and hence, a lot of maneuvering is required. Figure 12 gives some examples for this case.

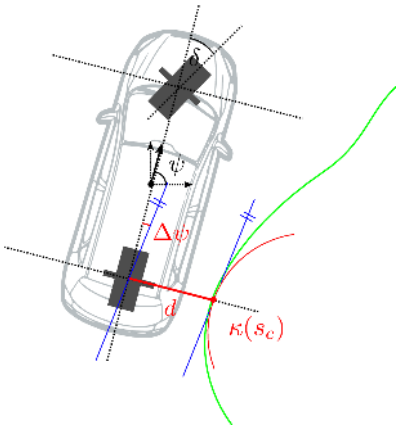

Fig. 9. Local coordinates of the one track model. Center of rear axle is matched to the closest point on the path to yield matching distance d and curvature at the matched point, $\kappa(s_c)$. $\Delta\psi$ is the difference between vehicles orientation and orientation of the path's tangent in the matched point.

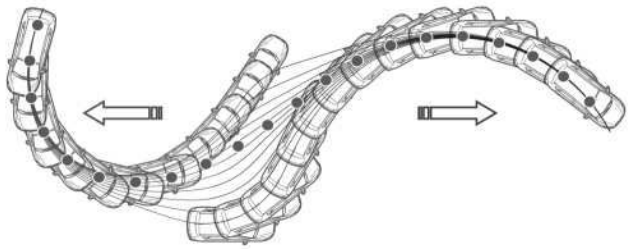

Fig. 10. Velocity-independent transient behavior (*orbits*) to different initial errors d and $\Delta\psi$ for forward (blue) and backward driving (red); the green dots represent the knots of the planned path which are the input to the controller

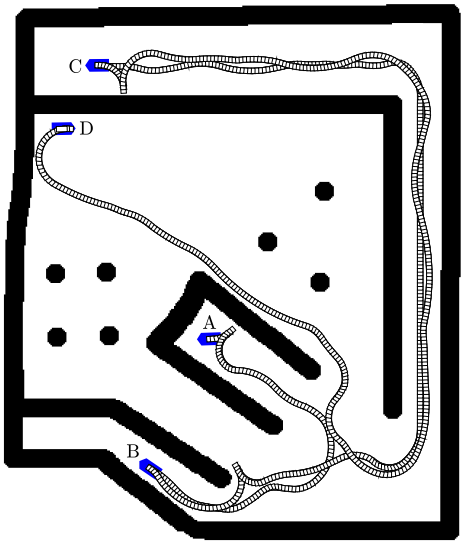

Fig. 11. Long distance navigation results. paths were planned from position A to B, B to C and C to D subsequently.

Search time remains below 2 seconds if search is restricted to an area of 200 m by 200 m. Though the environment is assumed to be static, this is fast enough to cope with slow changes in the environment by continous replanning. Additionally, to avoid collision with fast moving objects, a lower level process continously determines the free section of the planned path and, if necessary, stops by changing the control mode of the longitudinal control system. The lateral controller follows the generated paths precisely enough to implement all of the intended maneuvers. Speed was restricted to 7 m/s for the experiments and the UC competition.

VII. Summary

We have implemented a path planning system with a downstream closed loop controller that is capable of solving all of the following navigation problems: Precise parking maneuvers, narrow turns and longer distance navigation in unstructured environment. Designing search graph and vehicle controller around the same kinematic model allowed for easy integration and robust interaction of both systems.

The main contribution is the design of an obstacle sensitive cost function, which is used to accelerate the search process. It proved to be generally suited for all practically occuring

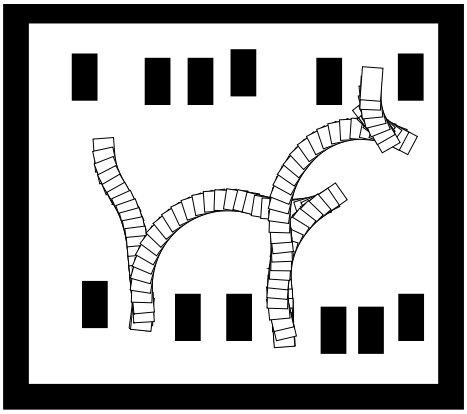

Fig. 12. Three subsequent backward parking maneuvers. Vehicle started on the left.

path planning problems, taking into account not only the vehicle's kinematic constraints, but also the topology of its free space. It has been designed to be applied on a grid-like, discrete obstacle representation that, in case of the AnnieWAY robot, is obtained from an upstream sensor system based on laser range measurements.

The complete system proved well suited to tackle the challenges posed at the DARPA Urban Challenge of 2007. Its suitability for planning both parking maneuvers and navigation in the presence of many obstacles, even in difficult, maze-like situations, easily surpasses the demands of the Urban Challenge.

VIII. Acknowledgements

The authors gratefully acknowledge the collaboration of all partners from University of Karlsruhe (TH), Technical University of Munich (TUM) and University of the German Federal Armed Forces, Munich. This work had not been possible without drawing inspiration from ongoing research of the transregional collaborative research centre 28 *Cognitive Automobiles*.

References

[1] *Urban Challenge Rules*. DARPA, Oktober 2007.
[2] Gunilla Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, 1986.
[3] Yong K. Hwang and Narendra Ahuja. Gross motion planning - a survey. *ACM Computing Surveys*, 24(3):219–291, 1992.
[4] L. Kavraki. Computation of configuration-space obstacles using the fast fourier transform. *IEEE Transactions on Robotics and Automation*, 11(3):408–413, 1995.
[5] H. Li and A. M. Vossepoel. Generation of the euclidean skeleton from the vector distance map by a bisector decision rule. In *CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 66, Washington, DC, USA, 1998. IEEE Computer Society.
[6] J.A. Reeds and R.A. Shepp. Optimal paths for a car that goes both forward and backward. *Pacific Journal of Mathematics*, 145(2):144–154, 1991.
[7] A. Rosenfeld and J. L. Pfaltz. Sequential operations in digital picture processing. *The Journal of the ACM*, 13:471–496, Oktober 1968.
[8] P. Švestka and M.H. Overmars. Motion planning for car-like robots using a probabilistic learning approach. *The International Journal of Robotics Research*, 16(2):119–143, April 1997.