Path-Planning for Autonomous Parking with Dubins Curves

Christoph Siedentop* Robert Heinze* Dietmar Kasper* Gabi Breuel*
Cyrill Stachniss†

Abstract: Autonomous cars can offer numerous services to its users, one is fully autonomous parking. There are different challenges involved in autonomous parking on parking spaces such as efficient path planning under constraints or limited sensing of today's cars. In this paper, we present a path planning approach for automated car parking in unstructured environments. Our system is able to find paths without imposing additional restrictions on either the environment, the final parking position, the number of direction switches, nor the length of the path. Our method consists of a lattice grid search, which yields kinematically-feasible paths and a subsequent optimization step to obtain a dynamically-desirable solution. The main contribution of this paper is in the construction of edges for the grid search through Dubins Curves. We implemented and throughly tested our system both in simulation and on a real Mercedes-Benz E-Class instrumented for automated driving. The parking solutions are found quickly and allow for an accurate execution so that parking is possible for small- and medium-sized parking lots. We believe that this paper is a viable approach towards fully automated parking.

Keywords: Automated Parking, Path Planning, Dubins Curves

1 Introduction

In the last decade, fully-automated research vehicles have become better and better. After an autonomous journey on highways and city streets, these vehicles will need to park. Because of the unstructured environment, maneuvering for parking is significantly different to trajectory planning on a road.

Finding a path for a car-like vehicle in an obstructed environment is hard. Simultaneously, one has to find a solution that does not violate the kinematic restrictions of the vehicle and the restrictions imposed by the environment. One kinematic restriction is the minimal curvature. The environment imposes the restriction that the vehicle does not collide with obstacles or moves outside the drivable area.

When the vehicle in question is an autonomous car additional objectives become important. First, what is kinematically-feasible might not be dynamically-feasible: the minimal curvature is speed-dependent and the change in steering is never instantaneous. Secondly, often human occupants are present in the vehicle and might wish to forgo speed for safety or comfort.

Our objective for this paper is two-fold: Find a path if it exists and improve it if possible. The application of interest is finding a path that can park a car from an initial position into a desired parking spot. We choose a hybrid approach for addressing this challenge. First, we construct a discrete grid in the configuration space and then search it with A* search connecting vertices with kinematically-feasible edges. To ensure this, we use the Dubins Car-Model, which provides a shortest-path closed-form solution between any two states. The A* search provides us with a shortest path between start and target configuration. We then pass this path to an

*Daimler AG, Autonomous Driving, Böblingen, Germany, E-Mail: first.last@daimler.com
†Institute for Geodesy and Geoinformation, University of Bonn, Germany, E-Mail: cyrill.stachniss@igg.uni-bonn.de

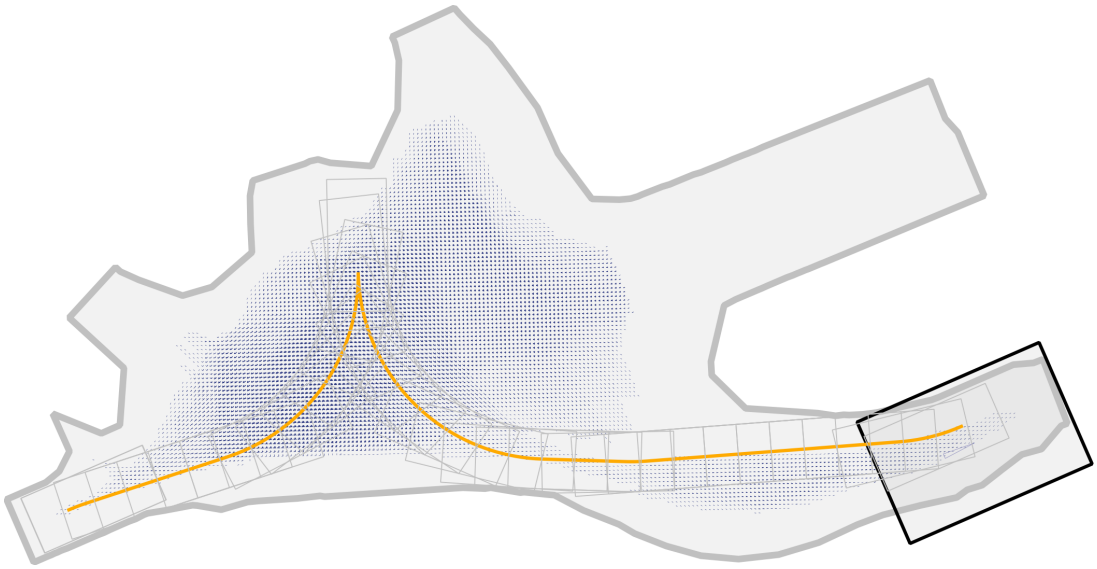

Figure 1: A typical parking scenario. The car is tasked of finding a drivable path from the bottom left to the black box in the bottom right. A solution is plotted in orange, with boundary boxes of the vehicle plotted every meter. In this example the found path consists of just 4 states connected with Dubins Curves. The grid consists of more than 200k states, 50k of which were visited during search and only 24 states were expanded. This is equivalent to a branching factor of 2000. The image shows only the visited states in blue.

optimisation engine. We formulate a cost function that improves the path in Frenet coordinates, minimizing the total jerk for comfortable movements, minimizing curvature for dynamics and maximises the distance to obstacles for a feeling of safety to occupants. The constraints are that the kinematics of the car may not be violated and that no obstacles are hit.

Our evaluation shows that this approach yields solutions in small- and medium-sized parking environments (see Fig. 1) and allows the car to park both forwards and backwards. Our approach finds solutions if the car can park in a single move, and if multi-turn maneuvering is necessary.

2 Literature Review

Path planning for parking can be approached from various directions. Automated parking systems found in production-series cars are based on geometric curves [18]. Yet, all currently available (mid 2015) systems can only deal with standard situations like parallel parking and orthogonal parking.

While these systems are solved through geometry, the opposite approach are graph-based planning approaches. These are not restricted to specific situations and can potentially solve any parking situation. Graph-search based approaches are challenged by two problems. The first is the potentially large number of states in the graph, caused by a high-dimensional state space. Published paper tend to use a four- or five-dimensional space. Typical choices for the dimensions are vehicle pose in $\langle x, y, \theta \rangle$ supplemented with current steering angle ξ and drive mode $d \in \{\texttt{forwards}, \texttt{backwards}\}$. The second challenge is how to connect vertices (states) in the graph. Three approaches exist: Ziegler et al. [20] construct a tree starting with the current vehicle state and generating neighboring states by forward-predicting the motion model. The second variant was proposed in [15]. Here the authors transformed the metric system into (2,4)-chained form, which then spans a regular grid. This technique from control theory

is used to great effect and offers a profound, clean solution. However, care must be taken in the implementation to avoid singularities in the transformation and also requires substantial computing resources. The third variant is the construction of a lattice grid, where states are connected by off-line solving the boundary-value problem of the motion model. Examples for this approach are [10, 8, 12, 6, 2].

3 System Overview

We aim at building a parking system for an automated car. Our test vehicle is a standard Mercedes-Benz E-Class equipped with additional sensors, processors and an interface to the actuators (steering, brakes and acceleration, and drive selection).

The vehicle's pose is provided by a localisation framework and obstacles are provided by radar sensors. The localisation method and sensor setup are described in more detail in [13].

The map of the environment and the final parking position are discovered through previous recordings of paths driven by a user. A set of trajectories is converted to a drivable area using our previously published approach [17]. The end poses of those user-driven trajectories also provide us with candidate parking positions. At run-time a user can select one of those parking spots. The vehicle will then calculate a path towards the selected parking spot and try to execute it. If new obstacles are sensed, the system will first re-plan around them using the optimisation step. If that fails, a complete re-planning with an updated environment map is started.

4 Algorithm Description

A complete description of our algorithm must cover both phases: grid search and optimisation. The grid search uses a regular grid in a three dimensional state space, where the sophistication lays in the edges. And then the solution is improved by means of convex optimisation.

4.1 Dubins Grid

When constructing a grid for the A* search with a constrained vehicle, such as a car, we must take care that the graph is well-connected. We chose to use Dubins' model of a car to generate edges that model the behaviour of our vehicle.

4.1.1 Vertices

The vertices in the Dubins Grid are configurations (i.e. the state) of the car: Pose in $\langle x, y, \theta \rangle$. These are created in regular intervals, although there is no inherent need for this regularity by the algorithm.

4.1.2 Edges

The edges are what motivates the name Dubins Grid. We connect neighboring vertices by Dubins Curves. Dubins Curves are the shortest path that a Dubins Car can take between any two states in an obstacle-free space. The Dubins Car uses a bicycle model with the explicit definition that it can only go forwards and that it has a minimal turning radius. Dubins [3] showed that the shortest path his car could take would only ever consist of three basic movements: straight forward ("S"), full right turn ("R"), or full left turn ("L"). He furthermore showed that the shortest path between two positions is composed of exactly three of these basic movements and that there are only six of these so-called *words* in total: Four with a straight segment in the middle (RSR, RSL, LSR, LSL) and two with a curved segment in the middle (RLR, LRL).

For our application of Dubins Curves it is important that a *closed-form solution* for all six possibilities exist. Finding the shortest of the six can be done through a simple min-operation or through more sophisticated means [1, 16].

To connect the Dubins Grid, we connect those vertices with edges whose Dubins Curves are collision-free with the map boundary and obstacles. Finally, we define the Dubins Curves both for a forwards and a backwards driving car. This means the direction of movement does not change in an edge. Only in a vertex is a change of direction possible. We connect two states with the shortest Dubins Curves.

4.1.3 Heuristics

As a heuristic we chose the length of the Dubins Curves between a vertex u and the closest goal vertex t. It is important to note that this is an inadmissible heuristic as it can overestimate the cost to the goal. This overestimation occurs when the shortest path includes a change in direction. Reeds-Shepp Curves [14] are an extension of Dubins Curves, where changes in direction are possible. As such, they would provide an admissible heuristic but are computationally too costly.

The Dubins Curve heuristic captures the non-holonomic constraints of the vehicle and provides a distance metric (if done both forwards and reverse) to the configuration space. The other heuristic incorporated into the search is often one that captures the map. This is sometimes called an obstacle-sensitive heuristic. This can either be done through Voronoi graphs [20] or based on a 2D-Dijkstra search through the map [10]. We chose to implement Dijkstra for its simplicity. However, in the tested scenarios it did not yield any runtime improvement. This was even the case in a very long scenario.

The result of the search is a list of 3D states which are connected through Dubins Curves. To complement the search we run optimisation on the path to smooth it.

4.2 Optimisation

Because the generated path found by the graph search above is composed of concatinated Dubins Curves, optimisation is highly beneficial to yield smooth and comfortable trajectories. We chose to implement a cost function and constrains with the following aims. There are two constraints: (1) The car may not collide with any obstacle or the boundary of the map. (2) The curvature may not exceed the maximal curvature as imposed by the vehicle constraints.

While a multitude of criteria to optimise are imaginable we concentrated on those that are necessary in our opinion for a human occupant to appreciate the maneuver. Primarily this is the change in curvature and the change in orientation. Additionally, we would like our optimised solution to track the original solution as best as possible.

4.2.1 Constraints

The first constraint is that the vehicle should not hit any obstacles.Second, the path must remain drivable and thus the curvature must remain below a threshold.

Obstacles We consider two types of obstacles. Real obstacles are provided by extrinsic sensors – an array of radar or LIDAR sensors which covers the complete surrounding of the vehicle. These are treated as point-shaped obstacles and updated multiple times a second. The second type of obstacles is the boundaries of the map. We treat both in the same framework using a Delaunay triangulation as described in [17]. The gist of our previous work is that through the triangulation, point obstacles can easily be added and removed.

Moving obstacles are treated as static obstacles and updated at regular intervals. In practise, this was sufficient for typical low-speed parking maneuvers.

The vehicle is modeled in the same way as in [19] through disks which cover the vehicle as best as possible. We selected three disks with diameters equal to the vehicle width.

Curvature Constraint We model the vehicle kinematics through the bicycle model commonly used for car-like robots. The vehicle is fully described by a base length L, the distance between the front and rear axle, and a maximal steering angle of the front wheels ($\xi_{\max}$). The maximal curvature is: $K_{max} = \dfrac{\tan \xi_{max}}{L}$.

4.2.2 Objective Function

Our variables for the optimisation are expressed in Frenet coordinates [4]. That is, we only optimise the offset to the original path. These offsets are denoted χ_i. The path is then represented as $\chi = \{\chi_i\}$ for $i = 1..n$ (n being the number of points on the path.). We express the objective function as a sum of three separate terms. $\mathcal{K}(\chi)$ represents the sum of curvature changes along the path, $\mathcal{O}(\chi)$ is the sum of orientation changes, and finally, $\mathcal{D}(\chi)$ expresses the distance to the original path.

The objective function is then: $f(\chi) = \mathcal{K}(\chi) + \mathcal{O}(\chi) + \mathcal{D}(\chi)$

Curvature Cost In detail we first have $\mathcal{K}$ the curvature costs. Those are defined as $\mathcal{K}(\chi) = \sum_{i=1}^{n-1} d(K_i, K_{i+1})$, where K_i is the curvature at χ_i and $d(\cdot, \cdot)$ the Euclidean distance. K_i is calculated with the points immediately preceding and following point i with Menger's formula [9].

Orientation Cost The delta in orientation is $\mathcal{O}(\chi) = \sum_{i=1}^{n-1} d(\psi_i, \psi_{i+1})$ with ψ_i being the orientation in χ_i. $d(\cdot, \cdot)$ is again the Euclidean distance except it is computed modulus 2π.

Hausdorff-Distance to Original Path With χ_i being the signed distance to the initial path, we reward the optimisation to find a solution close to the initial solution with the following costs: $\mathcal{D}(\chi) = \sum_i \|\chi_i\|_2$

4.2.3 Implementation Details

We employ the NLOpt-Library [5] to optimise the initial solution obtained from search using the above specified costs. All summands from the cost function are differentiable with regards to χ, so we are using the SLSQP-algorithm [7].

Cursory inspection makes us suspect that the objective function is convex. The optimisation typically converges after around 10 iterations.

5 Experiments and Evaluation

Results found in the first step through search can be seen in Fig. 1 (p. 2). The scenario shown in the figures is an actual parking space. We also evaluated our approach on a set of synthetic scenarios. Those were chosen to cover different situations that a parking car could encounter. There was a mix of both highly constrained and very large scenarios. We also sought situations that were very easy (simple forward-driving) and those that were very complicated to park in (requiring more than three gear changes). The experiments were performed on a 2014 Mercedes-Benz E-Class.

The effects various parameters have can be seen in figure 2. The length of the determined path seems reasonably independent of both spacing (resolution) of the grid and the numbers of headings represented in the grid. On the other hand, longer Dubins Curves as edges improve the solution.

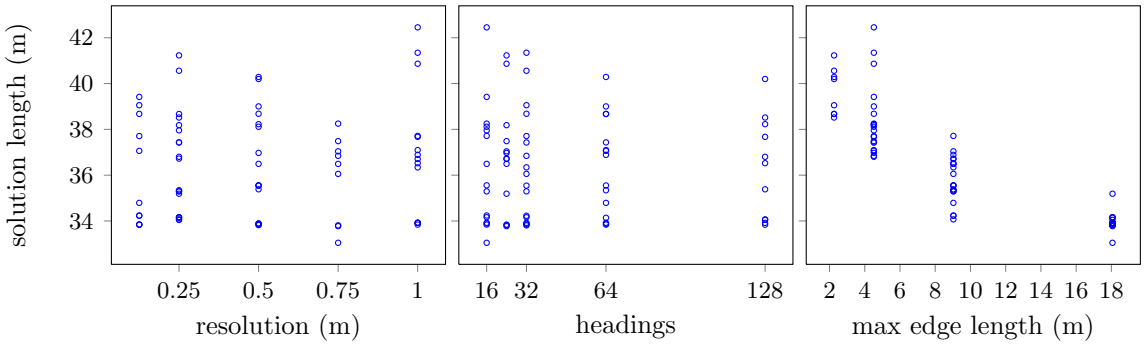

Figure 2: Resolution and number of headings show no correlation with the solution length. Maximal edge length in the state lattice clearly correlates with the solution length.

The performance depends most on the number of vertices in the graph, which is a result of grid resolution. This can be seen in figure 3. Because the planning time increases with the chosen grid resolution, we select a coarse resolution. As can be seen from figure 2, this does not effect the quality of solution. Thus an optimal choice of parameters is composed of a coarse resolution and a long maximal edge length.

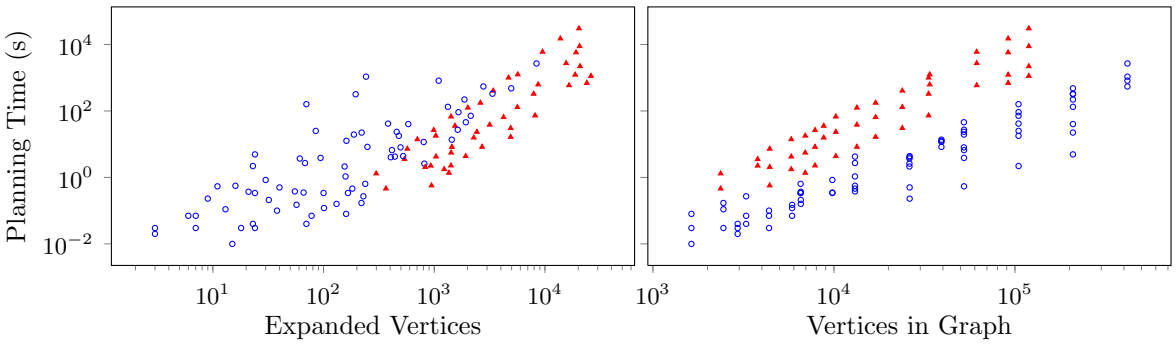

Figure 3: The number of vertices in the graph and the number of vertices that were expanded influences the planning time. Triangles (red) are from a longer scenario, circles (blue) from a shorter one (see Fig. 1).

5.1 Problems

Controllability Maximal steering angles mean that the controller has no leverage to apply control input. It is therefore difficult to exactly track a computed path. There are two solutions: Relax the model of the car by giving it a slightly larger turning circle. This would give the controller some space. Or, by using re-planning such that if the controller does not perfectly trace the desired path, while on maximal curvature, the path can be updated.

Dubins Curves Dubins Curves have a continuous gradient (C^1 continuous) but the curvature is discrete. This yields provably shortest-length paths but does not allow for dynamics of the car. Thus for a car to perfectly track the Dubins Curves, it would have to stop in the change

points. We mitigate this problem by optimising the path for dynamic characteristics in the described second step.

Minimizing gear-changes Directional gear changes always occur on a node, i.e. for a specific node an incoming edge is in the forward gear and departing edge is in reverse gear. This means that no cost can be assigned to the gear-change. This in turn means that the path cannot be optimised for the number of gear-changes.

We would argue that our algorithm is cleaner and simpler because we do not allow these gear-changes on edges and just in nodes. The Reeds-Shepp Curves could be used as edges if so desired. A simpler extension would be to make our graph bipartite and split all nodes into a forward and a backward node. In the forward node the vehicle would be in the forward gear, and the same for the backward node. Forward and backward nodes at the same position would then be connected through a gear-changing edge. This way a cost could be assigned to switching gears and even different costs could be assigned to driving forwards and backwards.

5.2 Future Work

In the future we decided to extend our approach to a four-dimensional space, in order to penalize directional gear-changes and backwards movement. Also, we are considering re-planning approaches such as D* [11] to cope with situations where our sensors cannot see far-away obstacles.

6 Conclusion

In this paper we have presented a comprehensive solution to finding paths for an autonomous parking scenario. We described the two aspects of our solutions: Finding an initial, drivable path through search in grid connected through Dubins curves and, secondly, improving the drivable path to allow higher velocities and more comfortable trajectories. The second step is done through optimisation.

The benefits of our approach is a simplified, yet kinematically correct, discretized search that is resolution complete. As such, we are guaranteed to find a path if one exists. But we are not doing more than necessary: Our initially found path is drivable but not smoothly drivable. We consider a smooth ride optional and clearly favour a non-smooth parking solution to no parking solution. In practise, optimisation can always improve the path. The optimised path becomes dynamically-feasible and the vehicle is able to track the path at higher speeds.

References

[1] Gyusang Cho and Jinkyung Ryeu. An efficient method to find a shortest path for a car-like robot. In *Intelligent Control and Automation*, pages 1000–1011. Springer, 2006.

[2] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 29(5):485–501, 2010.

[3] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, pages 497–516, 1957.

[4] Jean Frédéric Frenet. Sur les courbes à double courbure. *Journal de Mathématiques pures et appliquées*, pages 437–447, 1852.

[5] Steven G. Johnson. The nlopt nonlinear-optimization package, 2011.

[6] Ross Alan Knepper and Alonzo Kelly. High performance state lattice planning using heuristic look-up tables. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3375 – 3380, October 2006.

[7] Dieter Kraft. Algorithm 733: Tomp–fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software*, 20(3):262–281, 1994.

[8] Rainer Kümmerle, Dirk Hähnel, Dmitri Dolgov, Sebastian Thrun, and Wolfram Burgard. Autonomous driving in a multi-level parking structure. In *IEEE Intern. Conf. on Robotics and Automation*, pages 3395–3400. IEEE, 2009.

[9] Jean-Christophe Léger. Menger curvature and rectifiability. *Annals of mathematics*, 149:831–869, 1999.

[10] Maxim Likhachev and Dave Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8):933–945, 2009.

[11] Maxim Likhachev, David I Ferguson, Geoffrey J Gordon, Anthony Stentz, and Sebastian Thrun. Anytime Dynamic A*: An Anytime, Replanning Algorithm. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 262–271, 2005.

[12] Mihail Pivtoraiko and Alonzo Kelly. Constrained motion planning in discrete state spaces. In *Field and service robotics*, pages 269–280. Springer, 2006.

[13] Matthias Rapp, Markus Hahn, Markus Thom, Jürgen Dickmann, and Klaus Dietmayer. Semi-markov process based localization using radar in dynamic environments. In *IEEE 18th Intern. Conf. on Intelligent Transportation Systems (ITSC)*, 2015.

[14] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[15] Martin Rufli and Roland Siegwart. On the design of deformable input-/state-lattice graphs. In *IEEE Intern. Conf. on Robotics and Automation (ICRA)*, pages 3071–3077, May 2010.

[16] Andrei M Shkel and Vladimir Lumelsky. Classification of the dubins set. *Robotics and Autonomous Systems*, 34(4):179–202, 2001.

[17] Christoph Siedentop, Viktor Laukart, Boris Krastev, Dietmar Kasper, Andreas Wedel, Gabi Breuel, and Cyrill Stachniss. Autonomous parking using previous paths. In *Advanced Microsystems for Automotive Applications 2015 (AMAA)*, pages 3–14. Springer, 2016.

[18] Hélène Vorobieva, Nicoleta Minoiu-Enache, Sebastien Glaser, and Saïd Mammar. Geometric continuous-curvature path planning for automatic parallel parking. In *Intern. Conf. on Networking, Sensing and Control (ICNSC)*, pages 418–423. IEEE, 2013.

[19] Julius Ziegler, Philipp Bender, Thao Dang, and Christoph Stiller. Trajectory planning for Bertha—a local, continuous method. In *IEEE Intelligent Vehicles Symposium Proceedings*, pages 450–457, 2014.

[20] Julius Ziegler, Moritz Werling, and Jens Schröder. Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. In *IEEE Intelligent Vehicles Symposium*, pages 787–791, 2008.