The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios

Julius Ziegler and Christoph Stiller

Department of Measurement and Control (MRT)
University of Karlsruhe
76131 Karlsruhe, Germany
{ziegler|stiller}@mrt.uka.de

Abstract— **We present a method for motion planning in the presence of moving obstacles that is aimed at dynamic on-road driving scenarios. Planning is performed within a geometric graph that is established by sampling deterministically from a manifold that is obtained by combining configuration space and time. We show that these graphs are acyclic and shortest path algorithms with linear runtime can be employed. By reparametrising the configuration space to match the course of the road, it can be sampled very economically with few vertices, and this reduces absolute runtime further. The trajectories generated are quintic splines. They are second order continuous, obey nonholonomic constraints and are optimised for minimum square of jerk. Planning time remains below 20 ms on general purpose hardware.**

I. INTRODUCTION

Autonomous cars are useful, since they offer the potential to improve traffic safety by preventing accidents caused by human failure. Of prime importance in their development are motion planning algorithms that are capable of dealing with other road users and outside traffic participants, like pedestrians.

While motion planning is a difficult task in general, its application to intelligent vehicles moving in traffic imposes requirements that make it even more challenging: Road going vehicles obey nonholonomic motion laws that must be accounted for during planning. Moreover, they often operate at speeds that make a consideration of vehicle dynamics essential, and bounds on acceleration and jerk are required to maintain controllability and driving comfort. Finally, the presence of other traffic participants requires to consider the environment as dynamic, and consequently, a motion planner should not only generate purely geometrical paths but time parametrised trajectories. This is an indispensable requirement even for everyday driving scenarios like merging into moving traffic or overtaking.

Our approach refines the concept of the state lattice that has been introduced in [1]. A space-time manifold is constructed by combining configuration space and time axis. From this manifold, we sample a spatiotemporal state lattice. This new type of state lattice falls into the category of acyclic graphs, for which linear time shortest path solvers exist. Adding the time dimension to the sampling space increases the number of graph vertices required to express vehicle motion considerably, an effect we counter by reparametrising the workspace in a way that aligns the lattice to the course of the road. While this sacrifices some generality of motions realisable by the planner, it yields a drastic reduction in the number of configuration space samples required to describe vehicle dynamics. This allows us to meet the requirements defined above within a real time planner.

The rest of this article is structured as follows. In section II, we will give a short overview of seminal contributions in the field of trajectory planning, directing the focus on methods that have been applied in the automotive domain. In section III, we will develop the main concept of this work, namely the application of the state lattice principle to space time planning. Section IV complements this idea by proposing means of increasing performance of this method that can be applied when planning in on-road scenarios. We demonstrate motion planning results for two challenging scenarios in section V, and close with a summary of findings and a discussion of the methodology in section VI.

II. RELATED WORK

In the last decade, practical motion planning research has focused on sampling based methods. Two main fields have emerged, random sampling based methods and methods sampling the configuration space deterministically.

The most influential random sampling based method has been the technique of rapidly exploring random trees (RRT) [2]. The RRT algorithm uses a simple stochastic scheme to construct a geometric tree that explores the free space quickly. The method generalises very easily in terms of configuration space dimensionality and dynamic model selection, since only a forward dynamic model is required. It has been applied successfully not only to path planning, but also for generation of time-parametrised trajectories [3], and was employed on board an intelligent vehicle [4]. However, with regard to the scenario we are considering, some difficulties remain: To be efficient, the method relies heavily on nearest neighbour (NN) searching, and efficient methods for NN searching impose constraints on the choice of metric. Typically, the Euclidean metric is employed, and weak generalisations are feasible [5]. RRT is highly sensitive to the choice of NN-metric [6], and gives good results only if the metric is a proper approximation of the real travel cost. The problem with this is twofold: If the free space has narrow passages, the chosen metric may poorly approximate the real travel cost, since the robot must go a loop way. Then, search may take a long time to complete. In some scenarios, the cost function that is desired to be minimised may even be completely different from a practical NN-metric. This is the case for highly dynamic driving scenarios, where travel cost is based on energy type functions (square of jerk, square of curvature). For these reasons, the otherwise well-proven RRT algorithm cannot fulfil our requirements.

Another method of motion planning that has already been brought to great success in a practical automotive application [1], [7] works by sampling from the state space in a deterministic fashion. This is typically done in a periodic form (*state lattice*). Each sample becomes a vertex in a graph, and edges are generated that connect each vertex to a finite number of neighbours. The result is a geometric graph that has a state space sample attached to every vertex, and each edge has a geometric representation of a path connecting its adjacent vertices. Each edge can be assigned an arbitrary cost. This cost does not have to be related to a metric within the work space, and cost functions motivated by path energy or safety requirements are feasible. The shortest path is found within the graph with standard graph searching methods that draw on the dynamic programming paradigm.

With this work, we improve upon the original planning strategies [8] employed on board the autonomous car ANNIEWAY [9] during the DARPA Urban Challenge of 2007, a competition that was arranged for autonomous vehicles and set in an urban scenario.

III. MOTION PLANNING USING SPATIOTEMPORAL STATE LATTICES

In section III-A, we will review the concept of conventional state lattices, and explain briefly how they can be employed for motion planning in static environments. After generalising the concept to spatiotemporal state lattices in section III-B, we will point out properties that can be exploited when using them for planning in the presence of moving obstacles (section III-C).

A. Nontemporal state lattices

In state lattice planning [1], [7], the continuous configuration space is reduced to a geometric graph by sampling from a Cartesian grid. Each discrete sample becomes a vertex and a graph is obtained by connecting each vertex to a finite number of neighbours. If the motion model employed is nonholonomic, creating a geometric representation of an edge requires for solving a boundary value problem. This problem is sometimes called local planning, and several approaches to its solution have been proposed, like combinations of straights with arcs [10], clothoids [11] and polynomial clothoids [12]. Many local planners require for iterative solutions, and clever schemes must be employed to ensure fast operation [13].

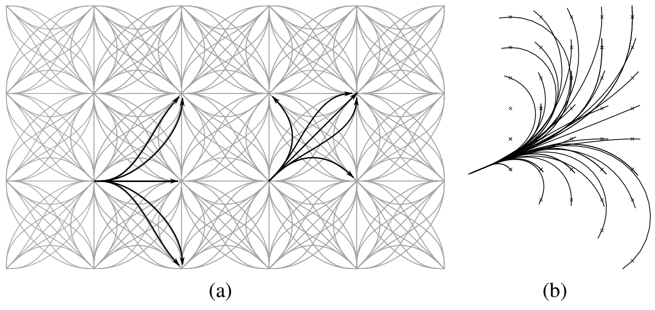


Fig. 1: (a): A nontemporal state lattice on a 2D workspace. Part of the canonical control set is displayed in black. (b): The right hand side shows part of a possible control set at finer discretisation, where the configuration space includes curvature.

The order of the boundary conditions depends on requirements imposed by the motion model underlying the planning. A car modelled as purely kinematic requires for continuous orientation along the solution path, and hence, edges must meet boundary conditions for position and orientation. If a car is modelled as a dynamic system, one will typically require trajectories to be second order continuous, since this prevents discontinuous steering input. Solving boundary value problems is, in general, a difficult task, but due to the periodic nature of the state lattice, it only has to be done for a small, canonical set of edges (*control set*, see figure 1) and can be preponed to an offline phase. This tells the method apart from probabilistic road maps (PRM) [14], [15], where a graph is obtained by picking random samples from the configuration space. The rationale for both methods is the same, namely the reduction of motion planning to the shortest path problem on graphs, for which fast methods exist. Their efficiency is owed to the dynamic programming (DP) paradigm [16], *i.e.* search algorithms keep track of intermediate vertices that a shortest path has already been found to.

Figure 1 illustrates state lattices. Note that in figure 1(a), the configuration space contains orientation, and that eight distinctive vertices, one for each discrete orientation, reside at each grid point. The control set in figure 1(b) was designed for a configuration space that contains curvature [17].

B. Spatiotemporal state lattices

Spatiotemporal state lattices are the result of combining configuration space and time into a single manifold, and then applying discretisation analogously to the preceding section. To illustrate this concept, we will first consider the simplistic case of a one dimensional configuration space.

Consider a vehicle travelling with varying velocity in $\mathbb{R}$. Its state is described by its distance from the origin, l. In the spirit of the state lattice approach, we constrain the state space to a discrete subset of $\mathbb{R}$ by sampling it at equidistant positions, with a sampling resolution of Δl. Assuming that obstacles are moving within the state space, we also have to discretise in the time dimension, t. This is

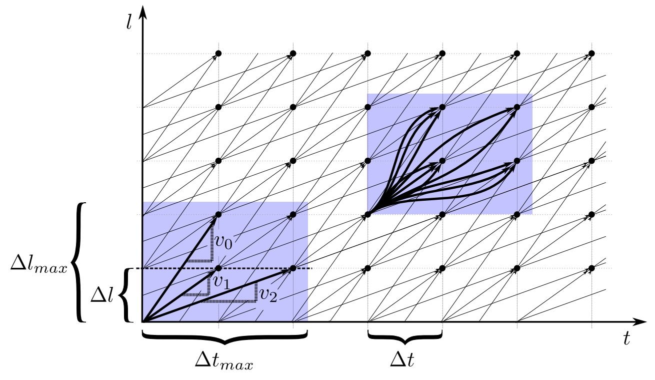


Fig. 2: A spatiotemporal state lattice over a one dimensional workspace. In the lower left shaded area, a possible control set for paths in $\mathcal{C}^0$ is depicted, the upper right one depicts part of one designed for higher order continuity, consisting of quintic polynomials.

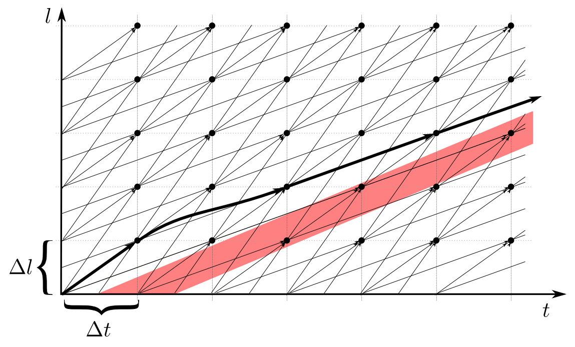


Fig. 3: Planning with a moving obstacle in the space time manifold established for a one dimensional workspace. The shaded area is covered by a moving object. A trajectory is shown that is composed of elements of the control set. Shortest paths can be found by relaxing vertices from left to right.

necessary, since, in a dynamically changing workspace, DP graph search algorithms will not only require to keep track of what positions are reachable, but also in what time they can be reached.

Figure 2 depicts a spatiotemporal state lattice over the described one dimensional workspace. Let us first assume that the vehicle can move at piecewise constant, positive velocities. Figure 2 makes plausible that these velocities should be multiples of integral fractions and $\frac{\Delta l}{\Delta t}$, with Δt being the resolution of time discretisation, in the example $v_0 = 2\frac{\Delta l}{\Delta t}$, $v_1 = 1\frac{\Delta l}{\Delta t}$, $v_2 = \frac{1}{2}\frac{\Delta l}{\Delta t}$. Since only a local connectedness of the graph is desired, one can constrain the maximum difference in l and t that can occur on a single edge transition. For later reference, we will call these constraints Δl_{max} and Δt_{max} (see figure 2). The required degree of continuity and, consequently, the required boundary conditions for the edges, can be chosen arbitrarily, to allow for smooth transitions between different velocities or accelerations.

For our experiments, we chose quintic polynomials to geometrically represent edges, which leads to paths in $\mathcal{C}^2$ and boundary conditions for position, velocity, acceleration and time. Quintic polynomials are attractive for planning dynamic driving manoeuvres, because, on top of being in a suitable continuity class, they have been proven to be optimal control trajectories with respect to minimum squared jerk [18]. Given the boundary conditions, coefficients can be computed very quickly. Closed form expressions exist to describe the integral of squared jerk and for maximum speed, acceleration and speed along the trajectory [19]. Quintic splines have been used for automotive motion planning before [20], albeit only to describe kinematic paths without time parametrisation.

C. Motion planning using spatiotemporal state lattices

For planning in the presence of moving obstacles, it is necessary to predict their positions into the future. Obstacles can then be transferred to the space time manifold, as figure 3 illustrates for the one dimensional case. The shaded area is occupied by a small object that moves left at roughly the speed of $\frac{1}{2}\frac{\Delta l}{\Delta t}$. A trajectory is found within the spatiotemporal lattice that does not collide with the obstacle.

To deal with obstacles efficiently, we create a mapping between a discrete space-time obstacle map and the set of all edges in the graph. This can be done in the offline graph generation phase. Then, edges blocked by obstacles can be invalidated quickly by a single run over the obstacle map. Note that this method scales well, so that increasing the number of obstacles has little impact on overall processing time.

Edge weights are derived from the integral of squared jerk of their geometric representations, as opposed to arc length. This improves safety, controllability and driving comfort. As we will discuss in section VI, this kind of edge weighting is hard to implement with many other search based planning methods.

Graph based motion planning algorithms usually employ shortest path algorithms that maintain vertices visited in a partially ordered data structure. Algorithms belonging to this class include Dijkstra's algorithm and A* search, as well as algorithms derived from these two, like Stentz' D* [21] and focused D* [22]. Keeping visited vertices ordered during search is not required for spatiotemporal lattices. They belong to the class of directed acyclic graphs (DAG), since transitions that go backward in time (and would create a cycle) do not exist. Hence, sorting vertices by time yields a topological ordering in advance, and vertices can be just processed in this order. In figure 3, this means that vertices can be relaxed from left to right. The resulting algorithm is linear in the number of vertices n, as opposed to Dijkstra's general scheme which is in $\mathcal{O}(n \log n)$. Since maintaining vertices in a partially ordered set is the most time consuming part of Dijkstra's algorithm, runtime drops drastically in absolute figures as well, if the faster algorithm is used. Note that all feasible graphs embedded into the space time manifold are acyclic, and hence, the faster algorithm can be employed to search these. A detailed description of the DAG

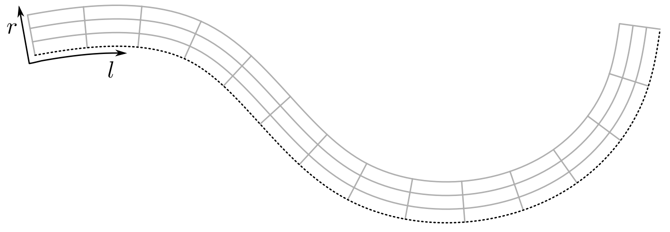


Fig. 4: Reparametrisation of the Cartesian plane. The dotted line indicates the original run of the road, (X, Y). The grey structure illustrates the result of discretising the new parameters, l and r.

shortest path algorithm appears in [23].

IV. A LANE ADAPTED REPARAMETRISATION OF THE WORKSPACE

The principle of spatiotemporal state lattices developed in the preceding sections generalises naturally to two dimensions. Doing this naïvely, however, bears dimensionality problems due to the required dense sampling of the state space. Note that, in comparison with [7] the dimensionality of the sampling space for the state lattice rises from 3 (2D position and orientation, in [1], curvature is consider additionally) to 7 (2D position, 2D velocity, 2D acceleration and time), due to moving from a kinematic to a higher order dynamic model and the incorporation of time. With dimensionality rising, it becomes ever harder to cover a continuous space with a small number of samples. In this section, we present an efficient way of sampling the configuration space that is adapted to the special case of navigating on a road whose run is known a prioi, *e.g.* from digital map data.

Given a continuous, two times piecewise differentiable, arc length s parametrised representation $(X(s), Y(s))$ of the course of the road, we define the following reparametrisation (l, r) of the 2D workspace, where (x, y) are standard Cartesian coordinates, $l(t)$ is the distance travelled along the road, and $r(t)$ is the lateral offset towards the road centre:

$$x(t) = X(l) - rY'(l) \quad (1)$$
$$y(t) = Y(l) + rX'(l). \quad (2)$$

This is a base change towards a local orthogonal coordinate system that has its abcissa aligned with the road for any l. It defines a two dimensional manifold as depicted in figure 4. As described earlier, differential boundary conditions of up to second order are required for edge generation. We therefore need to transform them through equations (1) and (2): Given $\dot{l}$, $\dot{r}$, $\ddot{l}$ and $\ddot{r}$, by application of the chain rule we obtain

$$\dot{x} = \dot{l}X'(l) - \dot{r}Y(l) - r\dot{l}Y'(l) \quad (3)$$
$$\dot{y} = \dot{l}Y'(l) + \dot{r}X(l) + r\dot{l}X'(l) \quad (4)$$

and

$$\ddot{x} = \ddot{l}X' + \dot{l}^2X'' - \ddot{r}Y - (2\dot{r}\dot{l} + r\ddot{l})Y' - \dot{r}\dot{l}^2Y'' \quad (5)$$
$$\ddot{y} = \ddot{l}Y' + \dot{l}^2Y'' - \ddot{r}X - (2\dot{r}\dot{l} + r\ddot{l})X' - \dot{r}\dot{l}^2X''. \quad (6)$$

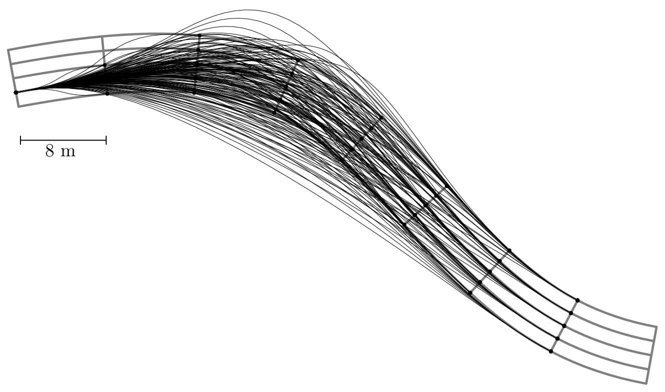


Fig. 5: State transitions on the transformed grid. The successors of one vertex are shown in black.

We now restrict parameters $l, r, \dot{l}, \dot{r}, \ddot{l}$ and $\ddot{r}$ to a discrete, grid like set (the vertices of the search graph) and transform them through equations (1) - (6). The resulting $x, y, \dot{x}, \dot{y}, \ddot{x}$ and $\ddot{y}$, together with discrete values for time t, are used as boundary values to calculate quintic polynomial trajectories as described in section III-B. To assert dynamic and kinematic feasibility, a respective edge is only added to the graph, if velocity, acceleration and jerk stay within bounds defined in advance. In the effort to further reduce the number of vertices, some ad hoc reductions can be applied to the sets of discrete parameters: r is constrained to an interval so as to restrict all vertices of the lattice to be within the bounds of the road. We set $\dot{r} = 0$ and constrain $\dot{l}$ to be positive, since we wish the vehicle to make progress along the road, while crosswise motion is to be avoided. Second derivatives $\ddot{l}$ and $\ddot{r}$ of the untransformed coordinates are set to zero at the grid points.

Figure 5 gives an impression of the graph we used for our experiments by displaying successor edges of a single vertex. The outdegree of vertices is approximately 200. Note that the procedure proposed in this section compromises, in part, the ability to reduce edge geometries to a small canonical control set (as discussed in section III-A). However, edge geometries can still be reused among edges only differing by a time offset.

V. EXPERIMENTS

We demonstrate planning in two scenarios that have been setup in a simulation environment. In both scenarios, the road is approximately 130 metres long and 6 metres wide. The first scenario demonstrates overtaking with oncoming traffic, see figure 6(a). Initially, the vehicle was travelling slow, and it had to arrive at the end of the road going at the same, moderate speed of approximately 4 $\frac{m}{s}$. In the second scenario (figure 6(b)), the vehicle approaches a junction at 10 $\frac{m}{s}$ and merges into traffic travelling at 4 $\frac{m}{s}$. The figures show the fastest trajectory found, respectively. The dots on the trajectories have the same temporal distance of 0.25 s. The set of parameters used to generate the search graph is summarised in table I. Graph statistics are shown in the same table for the overtaking scenario. The offline creation of the

$\Delta l \cdot \mathrm{m}$	8
$\Delta t \cdot \mathrm{s}$	2.5
$\Delta r \cdot \mathrm{m}$	1.2
$\Delta l_{max} \cdot \mathrm{m}$	60
$\Delta t_{max} \cdot \mathrm{s}$	30
$\dot{l} \cdot \frac{\Delta l}{\Delta t}$	$\frac{4}{3}, \frac{3}{2}, 3, \frac{7}{2}, 4, 5, 6$
$\ddot{l}$	0
$\dot{r}$	0
$\ddot{r}$	0

#vertices	2976
#edges	171379
average out degree	~200

TABLE I: Parameters and statistics of the search graph.

graph took about 4.5 s, runtime for the actual search was below 20 ms for both scenarios. The experiment was timed on a 2 GHz laptop computer.

VI. Summary and Discussion

We have presented an algorithm for motion planning that is capable of planning in dynamic on-road scenarios. The trajectories generated are time parametrised and meet the kinematic and dynamic requirements of road going passenger vehicles. Main contributions are the generalisation of state lattices to space time planning, and the transfer of the lattice concept to a a lane adapted workspace parametrisation. We believe that, from a practical standpoint, only by applying the latter method the first one becomes feasible.

The performance - in terms of runtime - of our method is mostly owed to the problem adapted sampling of the state space, and the resulting reduction in the number of vertices and edges required to express feasible motions of the vehicle. The graph is searched exhaustively on each query. This is in contrast to other lattice based approaches [7] that gain their speed by relying on proper heuristic cost functions to guide the search into promising parts of the graph. We believe the independence from cost functions to be advantageous, since their design (*i.e.*, finding a good lower bound on the actual cost to go) is a difficult task. It is especially difficult to give a good lower bound on path cost if paths are penalised by energy type functionals, like the square of jerk. Note that a good cost function should take the position of obstacles into account. To our knowledge, there is no search based, heuristically accelerated motion planning method using path cost other than arc length to date. Energy type cost functionals are feasible though, when doing an exhaustive search, as has been demonstrated by this work.

There are several directions for future work. An important property of the state lattice is the fidelity of the control set. Fidelity can be improved by either increasing sampling density, and hence, the number of vertices, or by increasing the number of edges by allowing for "long" transitions between vertices that are far apart in the state space. How to find the right balance between these two options is an open question that we will address in the future.

As described earlier, by setting up the graph in a way that allows for exhaustive searching, we have gained flexibility in choice of the edge weights (and hence, the optimality criterion for complete trajectories) not found in other methods. We believe that this can be exploited to imprint useful properties upon the solution paths by incorporating prior knowledge into the edge weights. Part of this knowledge will be provided from higher level perception- and decision processes in the intelligent vehicles system architecture. We are confident to obtain a situation adapted trajectory that meets safety and comfort requirements.

VII. Acknowledgements

We gratefully acknowledge support of this work by the German Research Foundation (DFG) within the transregional collaborative research centre 28 "Cognitive Automobiles".

References

[1] M. Pitvoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 2005.

[2] M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," tech. rep., Computer Science Dept., Iowa State University, 1998.

[3] M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *International Conference on Robotics and Automation*, 1999.

[4] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, "Motion planning for urban driving using RRT," in *International Conference on Intelligent Robots and Systems*, 2008.

[5] A. Atramentov and S. LaValle, "Efficient nearest neighbor searching for motion planning," in *International Conference on Robotics and Automatisation*, 2002.

[6] P. Cheng and S. LaValle, "Reducing metric sensitivity in randomized trajectory design," in *Intelligent Robots and Systems*, 2001.

[7] M. Likhachev and D. Ferguson, "Planning long dynamically-feasible maneuvers for autonomous vehicles," in *Proceedings of Robotics: Science and Systems IV*, 2008.

[8] J. Ziegler, M. Werling, and J. Schröder, "Navigating car-like vehicles in unstructured environment," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2008.

[9] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. v. Hundelshausen, O. Pink, C. Frese, and C. Stiller, "Team AnnieWAY's autonomous system," *International Journal of Field Robotics Research*, 2008.

[10] J. Reeds and R. Shepp, "Optimal paths for a car that goes both forward and backward," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 144–154, 1991.

[11] T. Fraichard and A. Scheuer, "From Reeds and Shepp's to continuous-curvature paths," *IEEE Transactions on Robotics and Automation*, vol. 20(6), 2004.

[12] H. Delingette, M. Hebert, and K. Ikeuchi, "Trajectory generation with curvature constraint based on energy minimization," in *International Conference on Intelligent Robots and Systems*, 1991.

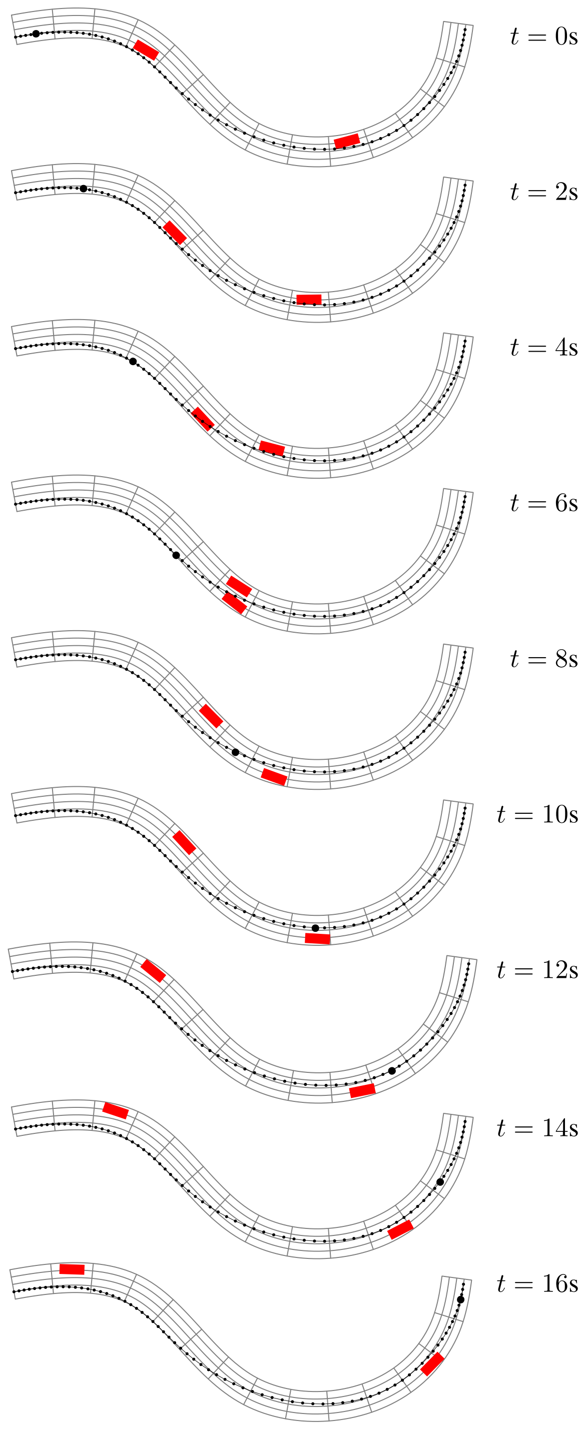
[13] T. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *International Journal of Robotics Research*, vol. 26, pp. 141–166, 2007.

[14] L. E. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, 1996.

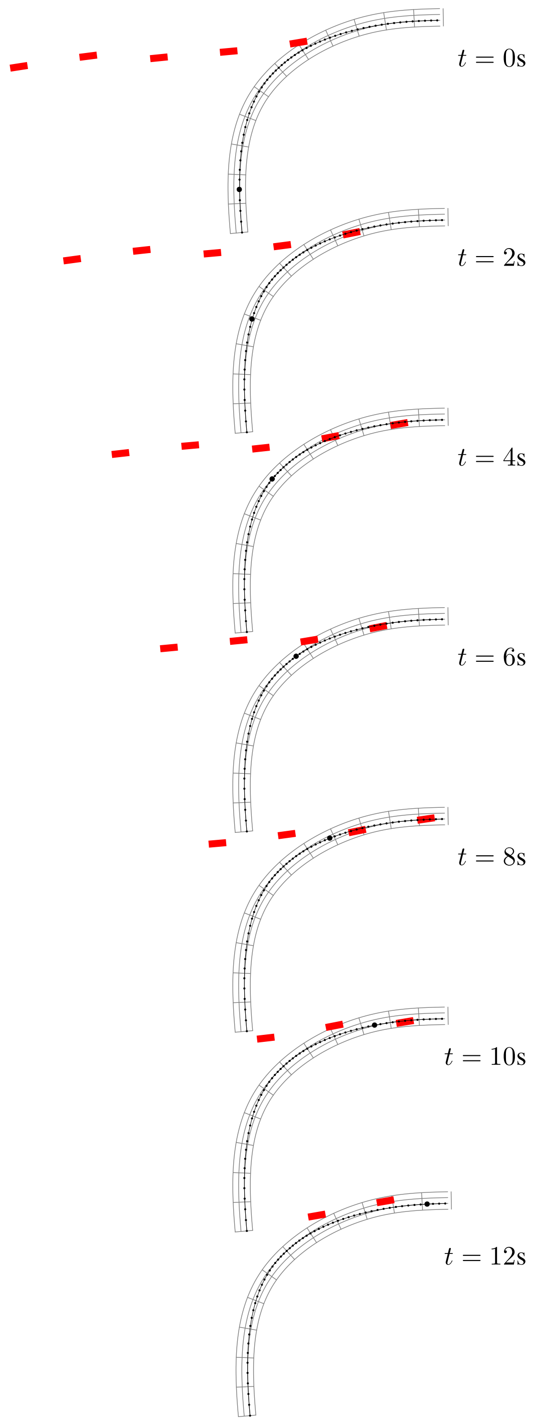
[15] I. Sucan, J. Kruse, M. Yim, and L. Kavraki, "Kinodynamic motion planning with hardware demonstrations," in *International Conference on Intelligent Robots and Systems*, 2008.

[16] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[17] C. Stiller, S.Kammel, I. Lulcheva, and J.Ziegler, "Probabilistic methods for environment perception of cognitive automobiles," *at - Automation*, vol. 11, pp. 568–574, 2008.



(a)



(b)

Fig. 6: Sequences demonstrating overtaking with oncoming traffic (a) and merging into slower traffic (b). The shortest time trajectory found is shown in black. Small dots on the trajectory are 0.25 s apart. The large dot is the vehicle executing the trajectory. Moving obstacles are displayed in red.

[18] A. Takahashi, T. Hongo, Y. Ninomiya, and G. Sugimoto, "Local path planning and motion control for AGV in positioning," in *International Workshop on Intelligent Robots and Systems*, 1989.

[19] R. Andersson, "Aggressive trajectory generator for a robot ping-pong player," *Control Systems Magazine*, vol. 9, pp. 15–21, 1989.

[20] A. Piazzi, C. Lo Bianco, M. Bertozzi, A. Fascioli, and A. Broggi, "Quintic G^2-splines for the iterative steering of vision-based autonomous vehicles," *Transactions on Intelligent Transportation Systems*, vol. 3, pp. 27–36, 2002.

[21] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *International Conference on Robotics and Automation*, 1994.

[22] A. Stentz, "The focussed D* algorithm for real-time replanning," in *International Joint Conference on Artificial Intelligence*, 1995.

[23] U. Manber, *Introduction to Algorithms*. Addison-Wesley, 1989.